ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Επίγνωση Πλαισίου σε Περιβάλλον Ανεξαρτήτου Πλατφόρμας

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χρήστος Κ. Ντάνος

Αθήνα, Νοέμβριος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Επίγνωση Πλαισίου σε Περιβάλλον Ανεξαρτήτου Πλατφόρμας

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χρήστος Κ. Ντάνος

Συμβουλευτική Επιτροπή : Δημήτριος Ασκούνης
Ιωάννης Ψαρράς
Γρηγόριος Μέντζας

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 15η Δεκεμβρίου 2015.

.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαρράς	Γρηγόριος Μέντζας
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ
.....	.....	.....
Βασίλειος Ασημακόπουλος	Ιωάννης Βασιλείου	Χρυσόστομος Δούκας
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Επίκ. Καθηγητής ΕΜΠ

.....
Γεώργιος Λεκάκος
Αναπ. Καθηγητής ΟΠΑ

Αθήνα, Νοέμβριος 2015

.....

Χρήστος Κ. Ντάνος

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Με την δημιουργία διασυνδεδεμένων συστημάτων ενδιάμεσου λογισμικού ανεξαρτήτου πλατφόρμας, προκύπτει μία νέα περιοχή ευκαιριών για την ανάπτυξη πανταχού-παρόντων συστημάτων διάχυτου υπολογισμού. Οι εφαρμογές με δυνατότητες επίγνωσης πλαισίου μπορούν να επεκταθούν, ώστε να μπορούν, με πρακτικό και ρεαλιστικό τρόπο, να εμπεριέχουν πολλαπλές πτυχές της αλληλεπίδρασης ανθρώπου-μηχανής στην καθημερινότητα, οι οποίες δεν περιορίζονται σε ένα μοντέλο συναγωγής πλαισίου με κέντρο την συσκευή. Η παρούσα διατριβή προτείνει ότι αυτό το μοντέλο μπορεί να επεκταθεί προς ένα άλλο, με κέντρο τον άνθρωπο, ανεξάρτητο της πλατφόρμας και της συσκευής, και το οποίο βασίζεται σε ένα προσωπικό οικοσύστημα συννέφου κατανεμημένων εφαρμογών και δεδομένων.

Το webinos, μία καινοτόμα συλλογή επεκτάσεων χρόνου εκτέλεσης ιστού, η οποία επιτρέπει σε εφαρμογές και υπηρεσίες ιστού να χρησιμοποιούνται και να διαμοιράζονται με συνέπεια και ασφάλεια σε ένα ευρύ φάσμα συγκλινόντων και διασυνδεδεμένων συσκευών, αποτελεί αυτό το οικοσύστημα. Το λειτουργικό πακέτο επίγνωσης πλαισίου του webinos, το οποίο απαρτίζεται από τον Διαχειριστή Πλαισίου και το API Πλαισίου, είναι προσβάσιμο από κάθε εφαρμογή webinos. Έπειτα από μία αυστηρή επιβολή πολιτικής δικαιωμάτων, μπορεί να συλλέξει πληροφορίες πλαισίου, είτε μέσω ενός αυτοματοποιημένου μηχανισμού αναχαίτισης κλήσεων συστήματος που πραγματοποιούνται από εφαρμογές όταν καλούν τα διάφορα API του webinos, μέσω ενός αυτοματοποιημένου μηχανισμού δειγματοληψίας σε αυτά τα API, ή μέσω προσαρμοσμένων επεκτάσεων του σχήματος του πλαισίου από συγκεκριμένες εφαρμογές. Στη συνέχεια, μπορεί να κατανείμει την πληροφορία πλαισίου από τον μηχανισμό αποθήκευσης προσωπικού συννέφου, στη μορφή απλών, διαχειρίσιμων και διαισθητικών Αντικειμένων Πλαισίου, από και προς όλες τις συσκευές με webinos που ανήκουν στον ίδιο χρήστη, ή ακόμα και σε άλλους, εξουσιοδοτημένους χρήστες.

Επιπλέον, το λειτουργικό πακέτο αυτό, παρέχει τον τρόπο για τη δημιουργία κανόνων που πυροδοτούν ενέργειες όταν συμβαίνουν γεγονότα που σχετίζονται με το πλαίσιο, όπως όταν ο χρήστης εισέρχεται στο σπίτι του ή δεν παρακολουθεί την αγαπημένη του εκπομπή στην τηλεόραση, και να εκκινήσει ουσιαστικές αλληλεπιδράσεις, οι οποίες δημιουργούν ή επικοινωνούν πληροφορίες, ή εκτελούν διεργασίες και εφαρμογές που σχετίζονται με την κατάσταση του χρήστη.

ABSTRACT

With the introduction of interconnected cross-platform middleware, a new area of opportunities for ubiquitous/pervasive computing has emerged. Context aware applications can be enhanced to practically and realistically incorporate multiple facets of human–machine interactions in everyday life that are not limited to a device-centred model for deducing context. This thesis proposes that this model can rather be extended to a human-centred, device and platform independent model, based on a personal distributed application and data cloud ecosystem.

For this to be achieved, webinos, an innovative set of web runtime extensions that enable web applications and services to be used and shared consistently and securely over a broad spectrum of converged and connected devices, is used to provide this ecosystem. The webinos Context Awareness Framework, consisting of the Context Manager and Context API, described here, is accessible to each webinos-enabled application. After strict policy enforcement, it can collect contextual information, either via an automatic mechanism that intercepts native calls made by webinos applications through the various webinos APIs, via an automatic polling mechanism to these APIs, or via custom, application-specific context schema extensions. It can then distribute the contextual information from its own personal cloud storage mechanism, in the form of simple, manageable and intuitive Context Objects, to and from all webinos-enabled devices owned by the same user, or even other, authorized users.

Furthermore, the webinos Context Awareness Framework provides means to create triggers for contextual events, such as the user entering his house, or missing his favourite TV show and initiate meaningful interactions that create and communicate information, or execute procedures and applications that are relevant to the user's situation.

Πίνακας Περιεχομένων

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

1. Εισαγωγή

1.1. Το Πρόβλημα

Η έννοια του προσωπικού υπολογιστή, η οποία εισήχθη τη δεκαετία του 1960 και η οποία έγινε μέρος της καθημερινότητας του ανεπτυγμένου κόσμου τις δεκαετίες του 1980 και 1990, αναφερόταν πάντα σε ένα μοντέλο χρήσης, κατά το οποίο ένας ηλεκτρονικός υπολογιστής είχε έναν ή περισσότερους χρήστες. Παράλληλα, η συνδεσιμότητα του συγκεκριμένου υπολογιστή, ακόμα κι όταν το διαδίκτυο έκανε την εμφάνισή του στα σπίτια και τους χώρους εργασίας, ήταν περιορισμένη σε υπηρεσίες και κατανεμημένα δεδομένα σε διαφορετικά και ανομοιογενή συστήματα, είτε στο διαδίκτυο, είτε σε τοπικά αποθηκευτικά μέσα. Αυτά τα συστήματα, όμως, εξακολουθούσαν να παρέχουν τις υπηρεσίες τους μέσω αυτού του ενός υπολογιστή, ο οποίος παρέμενε σταθερά εγκατεστημένος σε κάποιο γραφείο ή μετακινούταν από γραφείο σε γραφείο στην περίπτωση των φορητών υπολογιστών. Το λειτουργικό σύστημα του υπολογιστή αυτού επέτρεπε την εγκατάσταση και την εκτέλεση συγκεκριμένων προγραμμάτων που ήταν συμβατά με αυτό, ενώ η αποθήκευση δεδομένων αφορούσε σε κάθε εφαρμογή ξεχωριστά.

Μία σημαντική εξέλιξη στον τρόπο που εκτελούνται εφαρμογές προήλθε από το Παγκόσμιο Ιστό, όπου οι διάφορες υπηρεσίες παρείχαν διαδικτυακά την δυνατότητα αποθήκευσης δεδομένων, οι οποίες είναι προσβάσιμες από οποιοδήποτε υπολογιστή με σύνδεση στο διαδίκτυο και ένα πρόγραμμα περιηγητή. Οι εφαρμογές αυτές δίνουν τη δυνατότητα στον χρήστη να έχει πρόσβαση στο δικό του περιβάλλον χρήσης από οποιοδήποτε υπολογιστή, οπουδήποτε στον κόσμο.

Όταν, όμως, εξαπλώθηκε η χρήση έξυπνων συσκευών, από κινητά τηλέφωνα, σε ταμπλέτες, έξυπνες τηλεοράσεις και συστήματα αυτοκινήτου καθώς και η χρήση πολλαπλών συσκευών με διαφορετικά λειτουργικά συστήματα και διεπαφές, όλες με πρόσβαση στο διαδίκτυο, δημιουργήθηκε μία σειρά από ευκαιρίες, αλλά και προκλήσεις. Ο καθένας, πλέον, δύναται να κατέχει περισσότερες από μία συνδεδεμένες στο διαδίκτυο συσκευές και μεγάλο εύρος των δραστηριοτήτων του πραγματοποιείται μέσω αυτών των συσκευών. Σε πολλές περιπτώσεις, ο χρήστης επιθυμεί να εκτελεί την ίδια εφαρμογή σε διαφορετικά συστήματα και να αποθηκεύει τις ίδιες πληροφορίες σε όλες αυτές. Παράλληλα, εξαιτίας του ότι δεδομένα που προέρχονται από μία εφαρμογή

ενδέχεται να είναι χρήσιμα σε άλλες, οι λύσεις που μπορούν να δοθούν αφορούν σε περιβάλλοντα εφαρμογών από τον ίδιο κατασκευαστή λογισμικού.

Λόγω του ότι οι περισσότεροι χρήστες αυτών των συσκευών έρχονται σε συχνή επαφή με αυτές, και επειδή πολλές από αυτές τις συσκευές έχουν ενσωματωμένες μία σειρά αισθητήρων, από δέκτες GPS, σε επιταχυνσιόμετρα, αισθητήρες περιβάλλοντος φωτός, αποστασιόμετρα, πυξίδες, γυροσκόπια, κάμερες, όσο και αισθητήρες συνθηκών δρόμου, λειτουργίας κινητήρα και ρυθμίσεων αυτοκινήτων, είναι δυνατή η αξιοποίηση αυτών των δεδομένων για την εξαγωγή συμπερασμάτων για τους χρήστες από πληθώρα εφαρμογών. Ταυτόχρονα, η προσθήκη συσκευών, όπως έξυπνες τηλεοράσεις, έξυπνες οικιακές συσκευές και ολοκληρωμένα συστήματα διαχείρισης οχημάτων, έδωσαν τη δυνατότητα να καταγράφεται η συμπεριφορά και οι συνθήκες περιβάλλοντος του χρήστη σε πολλές από τις πτυχές της καθημερινότητάς του. Οι δυνατότητες αυτές έδωσαν τη ευκαιρία να αναπτυχθούν μεθοδολογίες και συστήματα για την αξιοποίηση αυτών των πληροφοριών πλαισίου του χρήστη και να παρέχουν πληθώρα αυτοματοποιημένων και μη, λειτουργιών σε εφαρμογές που αναπτύσσονται σε μεγάλο εύρος πλατφορμών.

Οι εφαρμογές επίγνωσης πλαισίου (context awareness), υπό τη παρούσα μορφή τους περιορίζονται σε δύο μορφές. Η πρώτη αναφέρεται σε εφαρμογές που συλλέγουν δεδομένα από μία συσκευή, τα αποθηκεύουν και εν τέλει παρέχουν τις υπηρεσίες τους σε αυτή τη συσκευή. Η δεύτερη αναφέρεται σε εφαρμογές διαδικτύου, όπου τα δεδομένα μπορούν να συλλέγονται από περισσότερες από μία συσκευές μέσω εφαρμογών ή υπηρεσιών του Παγκόσμιου Ιστού, να αποθηκεύονται σε απομακρυσμένους διακομιστές που διαχειρίζεται ο κατασκευαστής του λογισμικού και να παρέχονται στις συσκευές αυτές μέσω του ίδιου λογισμικού.

Τα προβλήματα που προκύπτουν από τις παρούσες υλοποιήσεις είναι τα εξής:

1. Τα δεδομένα επίγνωσης πλαισίου ή θα περιορίζονται σε μία συσκευή ή η διαχείρισή τους δε θα ανήκει στον χρήστη. Με άλλα λόγια, στη μία περίπτωση, οι εφαρμογές αποθηκεύουν τοπικά σε μία συσκευή τα δεδομένα τους και τα δεδομένα αυτά παραμένουν προσβάσιμα μόνο από αυτή τη συσκευή. Στην άλλη περίπτωση, τα δεδομένα αυτά αποθηκεύονται σε απομακρυσμένους διακομιστές που δεν ανήκουν στον χρήστη. Εξαιτίας του ότι, ακόμα και στην περίπτωση που δεν είναι

αυστηρώς ευαίσθητα, τα δεδομένα που αποθηκεύονται είναι προσωπικά και μπορούν να σκιαγραφήσουν με κάθε λεπτομέρεια τον χρήστη, τίθεται είτε ένας σημαντικός κίνδυνος παραβίασης της ιδιωτικότητας, είτε ένας αυστηρός περιορισμός του εύρους των δεδομένων και, κατά προέκταση, των δυνατοτήτων που μπορούν να κάνουν χρήση οι εφαρμογές επίγνωσης πλαισίου.

2. Κάθε εφαρμογή επίγνωσης πλαισίου έχει το δικό της οικοσύστημα, αποθηκεύει τα δικά της δεδομένα και είναι η μόνη που έχει πρόσβαση σε αυτά. Αν κάποια άλλη εφαρμογή συλλέγει τα ίδια δεδομένα, θα πρέπει να εγκαταστήσει το δικό της μηχανισμό για την απόκτηση και την αποθήκευσή τους, με αποτέλεσμα να υπάρχουν διπλότυπα και να απαιτείται διπλάσια επεξεργαστική ισχύς για την ίδια λειτουργία, πράγμα το οποίο περιορίζει το εύρος των εφαρμογών επίγνωσης πλαισίου οι οποίες μπορούν να εκτελούνται αποδοτικά ταυτόχρονα σε κάθε συσκευή.
3. Η συλλογή και αξιοποίηση των δεδομένων αυτών περιορίζεται στον χρόνο που είναι εγκατεστημένη η εφαρμογή ή η υπηρεσία που τα συλλέγει. Αυτό έχει ως αποτέλεσμα οι εφαρμογές επίγνωσης πλαισίου να χρειάζονται πάντα ένα διάστημα εκπαίδευσης, το οποίο μπορεί να κυμαίνεται μεταξύ μερικών ωρών, μέχρι και αρκετών μηνών. Σε περίπτωση που η εφαρμογή απεγκατασταθεί, ή ο χρήστης πάψει να χρησιμοποιεί την διαδικτυακή υπηρεσία, τα συμπεράσματα που έχουν εξαχθεί δε μπορούν να αξιοποιηθούν αλλιώς από τον χρήστη. Αυτό περιορίζει την χρήση των δεδομένων αυτών από τον κύκλο χρήσης της κάθε εφαρμογής.
4. Η συλλογή δεδομένων από ανομοιογενή συστήματα απαιτεί από τον προγραμματιστή να αναπτύξει διαφορετικούς μηχανισμούς άντλησης των δεδομένων επίγνωσης πλαισίου για κάθε λειτουργικό σύστημα και πλατφόρμα από την οποία επιθυμεί να συλλέξει δεδομένα. Αυτό αυξάνει κατά πολύ τις απαιτήσεις στους απαιτούμενους πόρους που χρειάζονται για την ανάπτυξή των εφαρμογών αυτών και θέτει ένα σημαντικό εμπόδιο εισόδου στην αγορά παροχής εφαρμογών επίγνωσης πλαισίου για προγραμματιστές και μικρομεσαίες εταιρίες πληροφορικής.

5. Οι ολοκληρωμένες πλατφόρμες που παρέχουν υπηρεσίες επίγνωσης πλαισίου παρέχονται ως εμπορικές υπηρεσίες για έναν κατασκευαστή λογισμικού κάθε φορά και περιορισμένο αριθμό υπολογιστικών οικοσυστημάτων, δίνοντας έμφαση σε φορητές συσκευές. Τα δεδομένα που παρέχονται για μία σουίτα εφαρμογών δεν είναι διαθέσιμα σε άλλους κατασκευαστές λογισμικού. Αυτό παρεμποδίζει τη δυνατότητα συνεργασιών είτε σε επίπεδο δεδομένων, εφόσον ο κάθε κατασκευαστής λογισμικού έχει αποκλειστική χρήση των δεδομένων που συλλέγει, είτε σε επίπεδο ανάπτυξης της πλατφόρμας, καθώς πρόκειται για εμπορικά προϊόντα κλειστού κώδικα.

1.2. Το Αντικείμενο και ο Στόχος της Διδακτορικής Διατριβής

Αντικείμενο της παρούσας διδακτορικής διατριβής είναι η μελέτη και εφαρμογή μεθόδων και πρακτικών μοντέλων συλλογής, επεξεργασίας και διάθεσης πληροφοριών επίγνωσης πλαισίου σε περιβάλλον διάχυτου υπολογισμού για χρήση σε διασυνδεδεμένες εφαρμογές ανεξάρτητες πλατφόρμας.

Στόχος της παρούσας διδακτορικής διατριβής είναι η ανάπτυξη ενός καινοτόμου μοντέλου συλλογής, αποθήκευσης, διαχείρισης και διάθεσης δεδομένων επίγνωσης πλαισίου σε ένα περιβάλλον, το οποίο στοχεύει να επιλύσει μερικά από τα πιο κρίσιμα προβλήματα που αποτρέπουν την υλοποίηση και διάδοση εφαρμογών, οι οποίες αξιοποιούν δεδομένα επίγνωσης πλαισίου ως προστιθέμενη αξία, την οποία στη συνέχεια, απολαμβάνει εξολοκλήρου ο τελικός χρήστης. Στο πλαίσιο αυτής της προσπάθειας, επιχειρείται να σχεδιαστεί και να αναπτυχθεί μία λειτουργική μονάδα, ως μέρος ενδιάμεσου λογισμικού εφαρμογών, η οποία θα προσφέρει υπηρεσίες επίγνωσης πλαισίου για εφαρμογές που προορίζονται για τελικούς χρήστες, όπου ισχύουν ταυτόχρονα τα παρακάτω:

- Τα δεδομένα επίγνωσης πλαισίου ανήκουν στον χρήστη και μόνο. Η βάση δεδομένων που αποθηκεύονται τα ευαίσθητα προσωπικά δεδομένα που αποτελούν το πλαίσιο του χρήστη βρίσκεται σε φυσικό μηχάνημα που ανήκει στον τελικό χρήστη, ή σε αποθηκευτικό χώρο στον οποίο εξασφαλίζεται ότι η πρόσβαση και το περιεχόμενο της βάσης δεδομένων ανήκει μόνο σε αυτόν.

- Όποιος προγραμματιστής επιθυμεί να αναπτύξει κάποια εφαρμογή για την ενδιάμεση πλατφόρμα, μπορεί να προσθέσει λειτουργίες και υπηρεσίες επίγνωσης πλαισίου χωρίς περιορισμούς από τον διανομέα της ενδιάμεσης πλατφόρμας. Ο διανομέας (ISP, κατασκευαστής υλικού, όπως κινητά τηλέφωνα, τηλεοράσεις κλπ), δεν θα πρέπει να εισάγει φραγμούς ως προς τη λειτουργικότητα των εφαρμογών αυτών, πέρα από τους κανόνες χρήσης της πλατφόρμας.
- Θα πρέπει να διασφαλίζεται η ορθή χρήση των δεδομένων πλαισίου από τους κατασκευαστές των εφαρμογών που τα χρησιμοποιούν. Οι εφαρμογές που επιθυμούν να αξιοποιούν τα δεδομένα πλαισίου του χρήστη πρέπει να αιτηθούν ειδική άδεια, μέσω της διαχείρισης πολιτικών ασφαλείας και ιδιωτικότητας, απευθείας από τον χρήστη.
- Τα δεδομένα πλαισίου των χρηστών θα πρέπει να είναι προσβάσιμα μόνο από τους ίδιους τους χρήστες. Οι εφαρμογές που αξιοποιούν τα δεδομένα πλαισίου του χρήστη μέσω της πλατφόρμας ενδιάμεσου λογισμικού συμμετέχουν στην πολιτική προστασίας των προσωπικών δεδομένων του χρήστη, ενώ αναπτύσσονται και εκτελούνται σε αγνωστικό περιβάλλον για τον προγραμματιστή, ως προς τα προσωπικά δεδομένα του κάθε χρήστη.
- Η πλατφόρμα ενδιάμεσου λογισμικού δε θα πρέπει να χρησιμοποιείται μόνο από εφαρμογές επίγνωσης πλαισίου, διότι αυτό περιορίζει κατά πολύ τα δεδομένα που συλλέγονται. Οι υπηρεσίες επίγνωσης πλαισίου προσφέρονται σε πλατφόρμα ενδιάμεσου λογισμικού εφαρμογών για πολλαπλές συσκευές, της οποίας οι δυνατότητες εκτείνονται πολύ πέρα από την επίγνωση πλαισίου, προσφέροντας μεγάλο εύρος εφαρμογών και συστημάτων που μπορούν να αντλήσουν και να αξιοποιήσουν τα δεδομένα αυτά.
- Θα πρέπει να διασφαλίζεται η μεγαλύτερη δυνατή ευκολία στη χρήση της πλατφόρμας επίγνωσης πλαισίου. Η συλλογή των δεδομένων επίγνωσης πλαισίου του χρήστη μπορεί να πραγματοποιείται σε όλες τις συμβατές με την πλατφόρμα ενδιάμεσου λογισμικού συσκευές οι οποίες του ανήκουν, χωρίς να χρειάζεται ο προγραμματιστής να γράψει εγγενή κώδικα για κάθε διαφορετική πλατφόρμα.

- Θα πρέπει να διευκολύνεται η επικοινωνία και η άντληση δεδομένων που προέρχονται από διαφορετικές συσκευές σε σχέση με αυτή που τα αιτείται. Η άντληση των δεδομένων από μία εφαρμογή μπορεί να αφορά σε δεδομένα που προέρχονται από οποιαδήποτε συσκευή που του ανήκει, χωρίς να χρειάζεται ο προγραμματιστής να γράψει εγγενή κώδικα για κάθε διαφορετική πλατφόρμα.
- Θα πρέπει να διασφαλίζεται η συνέχεια των δεδομένων πλαισίου, σε σχέση με τον χρήστη και όχι σε σχέση με τις εφαρμογές ή τις συσκευές. Τα δεδομένα είναι διαρκή, ακόμα κι αν απεγκατασταθεί η εφαρμογή που τα δημιούργησε, ή αποσύρει ο χρήστης τη συσκευή μέσω της οποίας δημιουργήθηκαν, εκτός αν ο ίδιος ο χρήστης επιθυμήσει τη διαγραφή τους.
- Θα πρέπει τα δεδομένα να είναι ανοιχτά προς τον χρήστη. Τα δεδομένα πλαισίου είναι προσβάσιμα ακόμα και από εφαρμογές που δεν τα δημιούργησαν, εφόσον το επιτρέψει ο χρήστης.
- Θα πρέπει να ενθαρρύνεται η δημιουργία κοινότητας προγραμματιστών που θα κάνουν χρήση της πλατφόρμας. Η υλοποίηση θα πρέπει να είναι ανοιχτού κώδικα, να παρέχεται δωρεάν και να προσφέρει εφάμιλλές δυνατότητες με αυτές των εμπορικών λύσεων, χωρίς όμως να στοχεύει στην αξιοποίησή της για λόγους προώθησης και διαφήμισης.

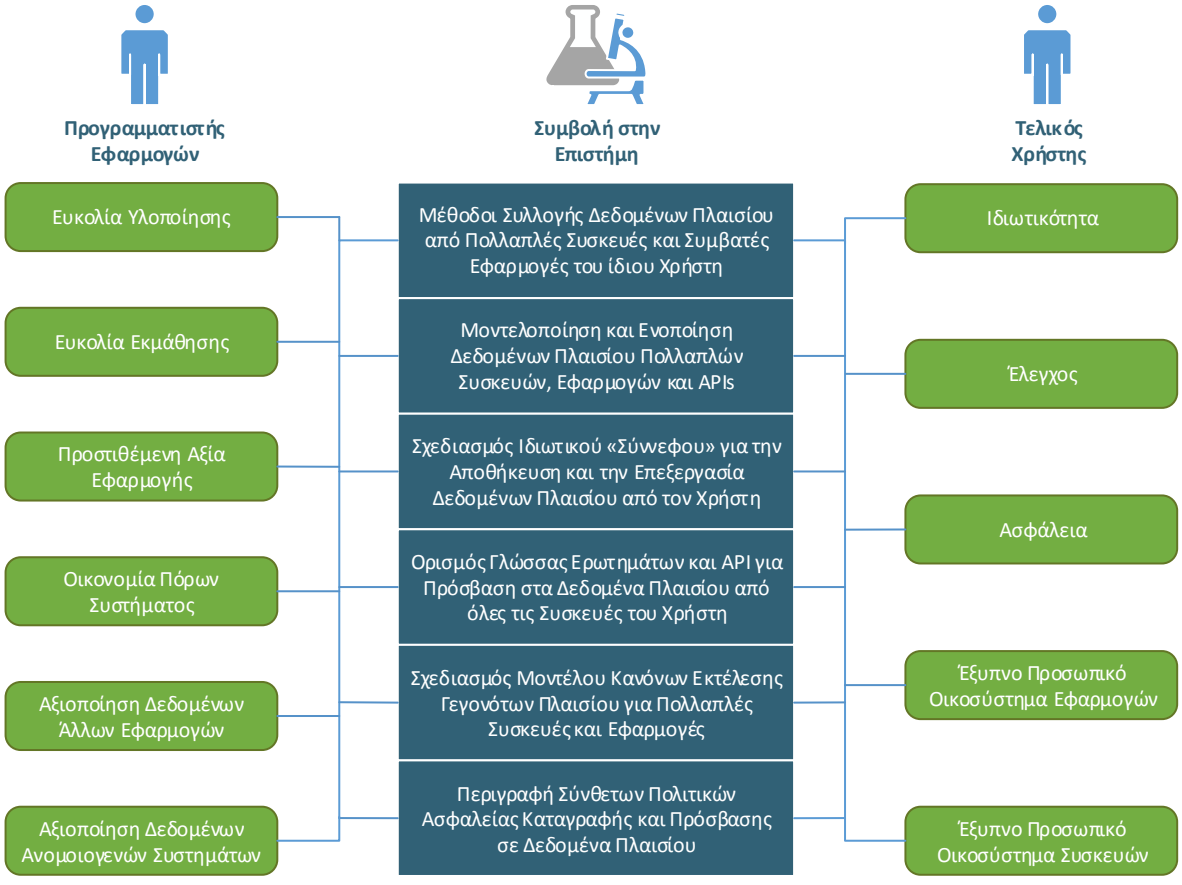
1.3. Η συμβολή της Διδακτορικής Διατριβής

Το επιστημονικό πεδίο του διάχυτου υπολογισμού και της επίγνωσης πλαισίου αποτελεί αντικείμενο μελέτης εδώ και αρκετά χρόνια, όμως μόλις τα τελευταία από αυτά έχει αρχίσει να βρίσκει και πρακτική εφαρμογή, κυρίως εξαιτίας της εξάπλωσης έξυπνων κινητών τηλεφώνων, έξυπνων συσκευών και της διασύνδεσής τους μέσω του Διαδίκτυο των Πραγμάτων (Internet of Things). Σε αυτό το νέο περιβάλλον, δίνεται η δυνατότητα σχεδιασμού καινοτόμων συστημάτων, τα οποία, λαμβάνοντας υπόψη το συμπεριφορικό προφίλ των χρηστών ως προς τη χρήση και την κατάσταση των συσκευών τους, μπορούν να παρέχουν υπηρεσίες, οι οποίες επεκτείνουν κατά πολύ τις παρεχόμενες μέχρι σήμερα υπηρεσίες επίγνωσης πλαισίου και που περιορίζονται σε μία συσκευή τη φορά ή σε μία εφαρμογή, συνήθως διαδικτυακή. Η ενοποίηση σε επίπεδο εφαρμογών των συστημάτων που ανήκουν στον ίδιο χρήστη μέσω ενός

καινοτόμου ενδιάμεσου λογισμικού διασύνδεσης συσκευών που ανήκουν στον ίδιο χρήστη, δίνει ξεκάθαρα μία τέτοια δυνατότητα. Παράλληλα, παρέχεται η δυνατότητα να δοθεί λύση στις ανησυχίες που αφορούν στην ιδιωτικότητα των τελικών χρηστών.

Συνολικά, η συμβολή της παρούσας διατριβής έγκειται στην πρόταση για την μετατόπιση, διεύρυνση και απλοποίηση των μοντέλων επίγνωσης πλαισίου από το επίπεδο των μεμονωμένων εφαρμογών ή/και των συσκευών, στο επίπεδο του χρήστη, ανεξαρτήτως εφαρμογής, συσκευής ή πλατφόρμας, παρέχοντας ταυτόχρονα μεγαλύτερη ιδιωτικότητα από αυτήν που θα ήταν αναμενόμενη από ένα κατανεμημένο σύστημα αυτού του τύπου.

Αναλυτικότερα, η συμβολή αυτή μπορεί να αναλυθεί σε δύο επίπεδα. Στο πρώτο επίπεδο βρίσκεται η Συμβολή της Διδακτορικής Διατριβής στην επιστήμη και την τεχνολογία. Στο δεύτερο επίπεδο βρίσκονται τα οφέλη της Διδακτορικής Διατριβής στους δύο ειδών χρήστες του συστήματος.



Εικόνα 1: Η συμβολή της Διδακτορικής Διατριβής και τα οφέλη των χρηστών

Για το πρώτο επίπεδο, την Συμβολή στην Επιστήμη και την Τεχνολογία επιδιώκεται να δοθεί μία ολοκληρωμένη λύση στο ζήτημα της μεθόδου συλλογής, συσσωμάτωσης, αναπαράστασης, αποθήκευσης, ερώτησης (querying) και διανομής δεδομένων Πλαισίου, για ένα περιβάλλον διασυνδεδεμένων συσκευών και εφαρμογών σε ανομοιογενή συστήματα, ως μέρος υπολογιστικής πλατφόρμας ενδιάμεσου λογισμικού παροχής εφαρμογών ιστού, ανεξαρτήτως υπολογιστικού υλικού και λειτουργικού συστήματος. Αναλυτικότερα, οι καινοτομίες σε μεθοδολογικό επίπεδο μπορούν να συνοψιστούν στα ακόλουθα αντικείμενα:

- **Καινοτόμοι Μέθοδοι Συλλογής Δεδομένων Πλαισίου από Πολλαπλές Συσκευές και Συμβατές Εφαρμογές του ίδιου Χρήστη**: Επιδιώκεται να σχεδιαστεί μία μεθοδολογία συλλογής δεδομένων πλαισίου, η οποία επιτρέπει την αυτοματοποιημένη συλλογή δεδομένων πλαισίου από όλες τις συσκευές που ανήκουν στον ίδιο χρήστη, ενώ επιτρέπεται ο εμπλουτισμός αυτών και η προσθήκη νέων από τις εφαρμογές που υλοποιούνται πάνω στην πλατφόρμα ενδιάμεσου λογισμικού και οι οποίες έχουν πρόσβαση σε αυτά.
- **Μοντελοποίηση και Ενοποίηση Δεδομένων Πλαισίου Πολλαπλών Συσκευών, Εφαρμογών και APIs**: Επιδιώκεται να μοντελοποιηθούν και να συσσωματώνονται τα δεδομένα πλαισίου που προέρχονται από την ενδιάμεση πλατφόρμα και τα οποία περιέχουν στοιχεία για πολλές διαφορετικές συσκευές, εφαρμογές, αλλά που αφορούν στον ίδιο χρήστη.
- **Σχεδιασμός Ιδιωτικού «Σύννεφου» για την Αποθήκευση και την Επεξεργασία Δεδομένων Πλαισίου από τον Χρήστη**: Μοντελοποιείται και υλοποιείται ένα προσωπικό ιδιωτικό «σύννεφο» για την αποθήκευση και ανάκληση δεδομένων πλαισίου από εφαρμογές. Ο ιδιοκτήτης των δεδομένων, δηλαδή ο τελικός χρήστης του συστήματος, έχει τη δυνατότητα να διαγράψει ανά πάσα στιγμή μέρος ή το σύνολο αυτών, αμετάκλητα.
- **Ορισμός Γλώσσας Ερωτημάτων και API για Πρόσβαση στα Δεδομένα Πλαισίου από όλες τις Συσκευές του Χρήστη**: Επιλέγεται και επιδιώκεται να τροποποιηθεί κατάλληλα υπάρχουσα γλώσσα

ερωτημάτων και ενσωματώνεται σε API που αποκλειστικό σκοπό έχει την διασύνδεση με τα δεδομένα πλαισίου στο Ιδιωτικό «Σύννεφό».

- **Σχεδιασμός Μοντέλου Κανόνων Εκτέλεσης Γεγονότων Πλαισίου για Πολλαπλές Συσκευές και Εφαρμογές**: Επιδιώκεται να σχεδιαστεί μία μονάδα για σύνταξη και διαχείριση κανόνων στο ενδιάμεσο λογισμικό, στην οποία περιγράφονται γεγονότα πλαισίου, τα οποία όταν συμβαίνουν, ενεργοποιείται μία διαδικασία σε μία ή περισσότερες συσκευές, οι οποίες εκτελούν προκαθορισμένες ενέργειες σχετικές με το γεγονός πλαισίου που περιγράφεται στον κανόνα.
- **Περιγραφή Σύνθετων Πολιτικών Ασφαλείας Καταγραφής και Πρόσβασης σε Δεδομένα Πλαισίου**: Επιχειρείται να περιγραφούν πολιτικές ασφαλείας και πρόσβασης σε δεδομένα πλαισίου, οι οποίες να μπορούν να εμπλουτίζονται δυναμικά, ανάλογα με τις απαιτήσεις δεδομένων πλαισίου των εφαρμογών από διαφορετικές συσκευές που τις δημιουργούν ή τις αξιοποιούν. Οι πολιτικές αυτές είναι εμφανείς ανά πάσα στιγμή στον ιδιοκτήτη/χρήστη των δεδομένων πλαισίου και του δίνεται η δυνατότητα να τις τροποποιήσει και να τις καταργήσει.

Τα οφέλη στους χρήστες αφορούν στην αξία που δημιουργείται, από τη μία πλευρά για τους προγραμματιστές εφαρμογών διάχυτου υπολογισμού, καθώς, από την άλλη, και τις δυνατότητες και τις ελευθερίες που παρέχονται στους τελικούς χρήστες των εφαρμογών αυτών.

Αναλυτικότερα, οι προγραμματιστές εφαρμογών επωφελούνται στους ακόλουθους τομείς:

- **Ευκολία Υλοποίησης Εφαρμογής**: Οι δυνατότητες επίγνωσης πλαισίου ενσωματώνονται σε υπολογιστική πλατφόρμα ενδιάμεσου λογισμικού παροχής εφαρμογών ιστού, με αποτέλεσμα να αποτελούν μέρος του συστήματος και να μη χρειάζεται να χρησιμοποιηθούν επιπρόσθετες βιβλιοθήκες, αποθηκευτικά μέσα και εναλλακτικές μέθοδοι διασύνδεσης με διακομιστές ή άλλες συσκευές. Όλες αυτές οι δυνατότητες παρέχονται ως υπηρεσίες στο ενδιάμεσο λογισμικό. Ταυτόχρονα, το σημείο πρόσβασης για τα δεδομένα πλαισίου και τις

υπηρεσίες κανόνων είναι ένα και ενιαίο για όλες τις συσκευές που ανήκουν στον ίδιο χρήστη.

- **Ευκολία Εκμάθησης**: Ο τρόπος με τον οποίο υλοποιεί ο προγραμματιστής τη αλληλοεπίδραση με τη Διεπαφή Προγραμματισμού Εφαρμογών (API) για τις υπηρεσίες Επίγνωσης Πλαισίου υλοποιείται με τον ίδιο τρόπο που υλοποιούνται οι υπόλοιπες Διεπαφές του ενδιάμεσου λογισμικού, ενώ παράλληλα, η γλώσσα ερωτημάτων που χρησιμοποιείται είναι ήδη διαδεδομένη, παράγοντες που επιταχύνουν σημαντικά την καμπύλη μάθησης για τις λειτουργίες αυτές.
- **Προστιθέμενη Αξία Εφαρμογής**: Οι εφαρμογές που μπορούν να αναπτυχθούν στο ενδιάμεσο λογισμικό εφαρμογών μπορούν να παρέχουν διευρυμένες υπηρεσίες επίγνωσης πλαισίου, οι οποίες είναι ήδη ενσωματωμένες στην πλατφόρμα, με αποτέλεσμα να είναι εύκολο να αξιοποιούνται δεδομένα από ανομοιογενείς πλατφόρμες (έξυπνες τηλεοράσεις, κινητά τηλέφωνα και tablets, συστήματα πολυμέσων αυτοκινήτων κ.α.) με μόνο βασικές γνώσεις προγραμματισμού ως προαπαιτούμενο. Παράλληλα, οι δυνατότητες δημιουργίας κανόνων πλαισίου, οι οποίοι ενεργοποιούνται σε όποια συσκευή προβλέπει ότι είναι καταλληλότερη η εφαρμογή, αποτελεί θέμα το οποίο παρέχεται ήδη λυμένο στο ενδιάμεσο λογισμικό.
- **Οικονομία Πόρων Συστήματος**: Τα δεδομένα επίγνωσης πλαισίου αποθηκεύονται σε απομακρυσμένη από την συσκευή βάση δεδομένων που τα συλλέγει και τα αξιοποιεί, ενώ ταυτόχρονα τα ερωτήματα εκτελούνται απομακρυσμένα, με αποτέλεσμα ο προγραμματιστής να περιορίζεται πολύ λιγότερο, κατά την ανάπτυξη, από τους πόρους συστήματος φορητών συσκευών (μπαταρία, αποθηκευτικός χώρος, επεξεργαστική ισχύς) ή μέσων με συγκεκριμένους περιορισμούς λειτουργιών (έξυπνες τηλεοράσεις, συστήματα πολυμέσων αυτοκινήτων).
- **Αξιοποίηση Δεδομένων άλλων Εφαρμογών**: Η ενοποίηση των δεδομένων επίγνωσης πλαισίου που παράγονται από εφαρμογές κάτω από τις υπηρεσίες επίγνωσης πλαισίου του ενδιάμεσου λογισμικού είναι,

υπό προϋποθέσεις, προσβάσιμα από τις υπόλοιπες εφαρμογές, πράγμα το οποίο σημαίνει ότι οι εφαρμογές μπορούν να χτίζουν η μία πάνω στη γνώση που δημιουργείται από τις υπόλοιπες, βελτιώνοντας έτσι τις υπηρεσίες που παρέχουν προς τους τελικούς χρήστες.

- **Αξιοποίηση Δεδομένων άλλων Συστημάτων**: Η συλλογή των δεδομένων πλαισίου στο ενδιάμεσο λογισμικό εφαρμογών πραγματοποιείται από όλες τις συμβατές συσκευές, ενώ το σημείο πρόσβασης σε αυτά είναι κοινό για όλες τις εφαρμογές, σε οποιαδήποτε συσκευή εκτελούνται. Αυτό δίνει τη δυνατότητα σε έναν προγραμματιστή εφαρμογών που προορίζει την εκτέλεση της εφαρμογής του σε προσωπικό υπολογιστή, για παράδειγμα, να εκμεταλλευτεί δεδομένα που προέρχονται από κάποια έξυπνη τηλεόραση, αυτοκίνητο, κάποια φορητή συσκευή ή κάποιου άλλου λειτουργικού συστήματος.

Για τον τελικό χρήστη, τα οφέλη συνοψίζονται στα ακόλουθα:

- **Ιδιωτικότητα**: Η φιλοσοφία με την οποία σχεδιάζονται οι υπηρεσίες επίγνωσης πλαισίου που περιγράφονται σε αυτή την διδακτορική διατριβή επιδιώκουν να διασφαλίσουν και σε επίπεδο πολιτικών, αλλά και σε επίπεδο σχεδιασμού την ιδιωτικότητα των δεδομένων πλαισίου του χρήστη. Λαμβάνοντας υπόψη το διευρυμένο φάσμα διεπαφών, εφαρμογών και συσκευών που ενσωματώνονται στα δεδομένα πλαισίου που αποθηκεύονται, το γεγονός ότι απαιτούνται ειδικές πολιτικές εμπιστευτικότητας και τα δεδομένα αποθηκεύονται σε μέσο το οποίο είτε ανήκει είτε έχει αποκλειστική πρόσβαση ο χρήστης, μπορεί σε μεγάλο βαθμό να διασφαλίσει την ακεραιότητα της ιδιωτικότητας του χρήστη. Στο περιβάλλον αυτό, ο προγραμματιστής εφαρμογών, ο πάροχος δικτύου ή υπηρεσιών αποθήκευσης και η ομάδα ανάπτυξης της πλατφόρμας δεν έχει πρόσβαση στα δεδομένα πλαισίου των τελικών χρηστών.
- **Έλεγχος**: Ο χρήστης έχει ανά πάσα στιγμή τη δυνατότητα να ελέγξει τα δεδομένα πλαισίου που είναι αποθηκευμένα συνολικά για όλες τις εφαρμογές, συσκευές και APIs του ενδιάμεσου λογισμικού που κατέχει

και επιλεκτικά να περιορίσει ή να επιτρέψει την πρόσβαση σε αυτά, καθώς και να τα διαγράψει από ένα μόνο σημείο πρόσβασης.

- **Ασφάλεια**: Εκμεταλλευόμενο τις πολιτικές ασφαλείας, κωδικοποίησης και πιστοποιητικών που εκδίδονται από το ενδιάμεσο λογισμικό, διασφαλίζεται σε σημαντικό βαθμό η ασφάλεια των δεδομένων και των επικοινωνιών, είτε σε κάθε συσκευή ξεχωριστά, καθώς και μεταξύ διαφορετικών συσκευών και άλλων χρηστών.
- **Έξυπνο Προσωπικό Οικοσύστημα Εφαρμογών**: Ο τελικός χρήστης μπορεί να χρησιμοποιήσει εφαρμογές οι οποίες μπορούν να ανήκουν σε ένα οικοσύστημα αλληλο-συνεργαζόμενων εφαρμογών, οι οποίες προσθέτουν, βελτιώνουν και αξιοποιούν κοινά δεδομένα πλαισίου.
- **Έξυπνο Προσωπικό Οικοσύστημα Συσκευών**: Οι δυνατότητες επέκτασης του πλαισίου του χρήστη πέρα από τα στενά όρια της συσκευής που χρησιμοποιεί ανά πάσα στιγμή, προσθέτοντας, με τη φιλοσοφία του Διαδικτύου των Πραγμάτων, συσκευές οι οποίες δεν γνωρίζουν μέχρι σήμερα την ύπαρξη η μίας της άλλης, επεκτείνουν την ευκολία με την οποία μπορούν να παρέχονται πληροφορίες ή να εκτελούνται γεγονότα πλαισίου σε διαφορετικές συσκευές από αυτές που προέρχεται η πρωτογενής πληροφορία.

1.4.Δομή της Διδακτορικής Διατριβής

Η παρούσα Διδακτορική Διατριβή αποτελείται από τα εξής κεφάλαια:

- Το Κεφάλαιο 1 αποτελεί την παρούσα εισαγωγή στο αντικείμενο και τους στόχους της διατριβής.
- Το Κεφάλαιο 2 περιγράφει τις βασικές έννοιες τις οποίες πραγματεύεται η παρούσα διδακτορική διατριβή και περιλαμβάνει τις έννοιες του Πλαισίου, του προγραμματισμού με διαχείριση πληροφορίας Πλαισίου, της συνεργατικής επίγνωσης πληροφορίας Πλαισίου, της διαχείρισης πληροφοριών Πλαισίου, των κριτηρίων συμβιβασμού και του ενδιάμεσου λογισμικού.
- Το Κεφάλαιο 3 περιγράφει την υφιστάμενη κατάσταση στους τομείς του ενδιάμεσου λογισμικού επίγνωσης πλαισίου, την σύγκριση μεταξύ διαφορετικών Ενδιάμεσων Λογισμικών Επίγνωσης Πλαισίου, την

αξιολόγηση των υπαρχόντων Ενδιάμεσων Λογισμικών Εφαρμογών, την περιγραφή του συστήματος webinos και των κύριων τεχνολογιών που χρησιμοποιεί.

- Το Κεφάλαιο 4 περιγράφει τη μεθοδολογία συλλογής απαιτήσεων συστήματος που ακολουθείται, θέτοντας τους στόχους, περιγράφοντας τον τρόπο σύνταξης των σεναρίων χρήσης και τον τρόπο καταγραφής των λειτουργικών και μη λειτουργικών απαιτήσεων για το περιγραφόμενο σύστημα.
- Το Κεφάλαιο 5 περιγράφει τις προδιαγραφές του Διαχειριστή Πλαισίου και του API Πλαισίου του webinos. Σε αυτό το σημείο παρουσιάζονται τα βασικά δομικά στοιχεία του συστήματος, παρέχοντας πληροφορίες για τους τρόπους με τους οποίους συλλέγονται δεδομένα πλαισίου, τον τρόπο που εκτελούνται ερωτήματα πλαισίου, τους αυτοματισμούς που μπορούν να εκτελεστούν, καθώς και τους τρόπους με τους οποίους διασφαλίζεται η ασφάλεια και η ιδιωτικότητα των δεδομένων αυτών.
- Το Κεφάλαιο 6 περιγράφει μία πλήρη μελέτη περίπτωσης χρήσης του Διαχειριστή Πλαισίου, η οποία εκμεταλλεύεται το σύνολο των δυνατοτήτων επίγνωσης πλαισίου που παρέχονται από το webinos, και η οποία αφορά στην επέκταση των δυνατοτήτων φορετών συστημάτων υγείας.
- Το Κεφάλαιο 7 περιέχει τα συμπεράσματα της παρούσας εργασίας.
- Το Κεφάλαιο 8 προτείνει τα επόμενα ερευνητικά βήματα που μπορούν να πραγματοποιηθούν στην κατεύθυνση που έχει δοθεί από την παρούσα διδακτορική διατριβή.
- Το Παράρτημα 1 περιέχει τις προδιαγραφές του συστήματος σε Web IDL.

2. Βασικές έννοιες

Μία σημαντική διαφορά μεταξύ του τρόπου με τον οποίο πραγματοποιείται η επικοινωνία μεταξύ ανθρώπων και μεταξύ ανθρώπου και μηχανής αφορά στην προαπαιτούμενη γνώση και εμπειρία των συστημάτων αυτών. Στην περίπτωση της επικοινωνίας μεταξύ ανθρώπων, ο πλούτος της κοινής γλώσσας, η κοινή κατανόηση του τρόπου με τον οποίο λειτουργεί ο κόσμος και μία υπονοούμενη κοινή αντίληψη καθημερινών καταστάσεων αποτελεί βασικό κριτήριο ως προς το πώς οι άνθρωποι επιτυγχάνουν να μεταδίδουν ιδέες και να αντιδρούν ανάλογα με τις περιστάσεις (24). Όταν οι άνθρωποι συνομιλούν με ανθρώπους ένα μεγάλο μέρος της μεταξύ τους επικοινωνίας στηρίζεται σε αυτό το σύνολο πρότερης υπονοούμενης γνώσης της κατάστασης στην οποία βρίσκονται. Η πρότερα γνωστή αυτή κατάσταση είναι το πλαίσιο (context) μέσα στο οποίο πραγματοποιείται η ανταλλαγή των πληροφοριών. Μία τέτοια ανταλλαγή θα οδηγήσει σε διαφορετικά συμπεράσματα ή αποφάσεις, ανάλογα με το πλαίσιο στο οποίο βρίσκεται, επιτρέποντας ένα σημαντικό βαθμό ασάφειας, το οποίο δε θα ήταν αποδεκτό στην επικοινωνία μεταξύ ανθρώπου και μηχανής. Για παράδειγμα, αν ζητήσει κάποιος από έναν άνθρωπο να φέρει γάλα από το σούπερ μάρκετ, υπονοείται το πλαίσιο του καταστήματος, στο οποίο το γάλα θα αγοραστεί για κάποιο χρηματικό ποσό, το πλαίσιο του είδους του γάλακτος, μεταξύ φρέσκου ή εβαπορέ, ανάλογα με αυτό που καταναλώνεται περισσότερο στο σπίτι, το πλαίσιο της αναμενόμενης κατανάλωσης, όπου θα αγοραστεί μία ποσότητα η οποία αναμένεται να καταναλωθεί πριν λήξει, το πλαίσιο της χωρητικότητας του ψυγείου, όπου η ποσότητα που θα αγοραστεί θα πρέπει να μπορεί να αποθηκευτεί στο ψυγείο κ.α. Όλες αυτές οι λεπτομέρειες, οι οποίες θεωρούνται δεδομένες στην επικοινωνία μεταξύ ανθρώπων, είναι παντελώς άγνωστες στα υπολογιστικά συστήματα.

Εξαιτίας του σημαντικού περιορισμού στους τρόπους με τους οποίους ένας χρήστης υπολογιστικού συστήματος μπορεί να εισάγει πληροφορίες σε αυτό, με το να παρέχεται πρόσβαση του συστήματος στο πλαίσιο του χρήστη, εμπλουτίζεται η επικοινωνία μεταξύ τους, με αποτέλεσμα το υπολογιστικό σύστημα να μπορεί να παρέχει υψηλότερης ποιότητας πληροφορίες ή να παρέχει πιο σχετικές ως προς τον χρήστη υπηρεσίες (1). Η χρήση του πλαισίου αυτού γίνεται ολοένα σημαντικότερη στα πεδία των φορητών και διάχυτων υπολογιστικών συστημάτων, όπου το πλαίσιο των χρηστών αλλάζει γοργά.

Για να γίνει αποτελεσματικότερη χρήση του πλαισίου, είναι απαραίτητο να γίνει κατανοητό τι ακριβώς είναι το πλαίσιο αυτό, πώς μπορεί να χρησιμοποιηθεί, ενώ ταυτόχρονα πρέπει να παρέχεται υποστήριξη σε επίπεδο αρχιτεκτονικής του συστήματος. Η βαθιά κατανόηση του πλαισίου επιτρέπει στους σχεδιαστές λογισμικού την επιλογή του κατάλληλου πλαισίου για χρήση από τις εφαρμογές τους και ταυτόχρονα, τον εντοπισμό των συμπεριφορών αυτών που περιέχουν πληροφορίες πλαισίου από την πλευρά του χρήστη και οι οποίες μπορούν να χρησιμοποιηθούν από τις εφαρμογές. Η δημιουργία μίας αρχιτεκτονικής για την υποστήριξη των δεδομένων πλαισίου επιτρέπει τον ευκολότερο σχεδιασμό και υλοποίηση των εφαρμογών αυτών. Σε επίπεδο αρχιτεκτονικής υποστήριξης των λειτουργιών πλαισίου, ορίζονται δύο επίπεδα, αυτά των υπηρεσιών και των αφαιρέσεων (24). Οι υπηρεσίες αφορούν στις δυνατότητες που δίνει το λογισμικό στην ανάκτηση, αποθήκευση, εξαγωγή και συμψηφισμό πληροφοριών πλαισίου. Οι αφαιρέσεις αφορούν στον τρόπο γενίκευσης των δεδομένων και των λειτουργιών πλαισίου, με τρόπο τέτοιο ώστε να εξυπηρετείται η ευκολότερη ανάπτυξη λογισμικού και η ευκρινέστερη παρουσίαση των δεδομένων πλαισίου.

Σταδιακά παρατηρείται μία μεταστροφή στα υπολογιστικά συστήματα, καθώς από σταθερούς προσωπικούς υπολογιστές σε γραφεία και σπίτια, ογκώδη συστήματα που απαιτούν χώρο και χρόνο για τη φυσική εγκατάσταση και χρήση τους, σε φορητούς υπολογιστές (laptops), υπολογιστές παλάμης (tablets), έξυπνα κινητά τηλέφωνα (smartphones) και άλλες φορητές ή φορετές συσκευές όπως ρολόγια και έξυπνες οικιακές και άλλες συσκευές. Κατ’ αυτό τον τρόπο τα υπολογιστικά συστήματα από ένα κατανεμημένο μοντέλο εξελίσσονται σε ένα μοντέλο φορητότητας. Η συνδεσιμότητά τους δεν περιορίζεται μόνο σε αυτή μέσω σταθερών συνδέσεων στο διαδίκτυο, αλλά και μέσω ασυρμάτων δικτύων με κινητούς κόμβους. Τα κινητά συστήματα περιλαμβάνουν ένα μεγάλο αριθμό προσπελάσιμων και συχνά αόρατων υπολογιστικών συσκευών. Οι συσκευές αυτές μπορεί να βρίσκονται ενσωματωμένες στο περιβάλλον ή στον χρήστη. Συνδέονται σε μία, ταχύτατα και αυθαίρετα, μεταβαλλόμενη δικτυακή υποδομή, στην οποία βρίσκονται κατανεμημένες υπηρεσίες, διαθέσιμες προς τους χρήστες.

Οι κινητές συσκευές έχουν κάποια ιδιαίτερα χαρακτηριστικά, που τις διαφοροποιούν από τους σταθερούς ηλεκτρονικούς υπολογιστές, ως προς το λογισμικό που μπορούν να υποστηρίξουν:

- Έχουν περιορισμένους υπολογιστικούς πόρους όπως: υπολογιστική ισχύ, μνήμη και μόνιμο αποθηκευτικό χώρο.
- Επικοινωνούν μεταξύ τους και με άλλους υπολογιστές ασύρματα, σχηματίζοντας δίκτυα με ακαθόριστη δομή και συχνές συνδέσεις και αποσυνδέσεις κόμβων (ad – hoc δίκτυα). Το εύρος ζώνης στην επικοινωνία των συσκευών αυτών είναι συνήθως περιορισμένο, ενώ η συνεχής διαθεσιμότητα του δικτύου δεν είναι δεδομένη.
- Ο χρήστης έχει τη δυνατότητα να κινείται στο χώρο αλλάζοντας συνεχώς το περιβάλλον λειτουργίας της συσκευής. Σε κάποιες περιπτώσεις η γεωγραφική θέση του χρήστη και περιβαλλοντικοί παράγοντες μπορεί να επηρεάζουν την εκτέλεση της εφαρμογής που εκτελείται στη συσκευή.
- Οι κινητές συσκευές έχουν περιορισμένη αυτονομία. Η λειτουργία τους εξαρτάται απόλυτα από την χωρητικότητα και το επίπεδο φόρτισης της μπαταρίας τους.

Τα συστήματα με τα παραπάνω χαρακτηριστικά ονομάζονται διάχυτα (pervasive ή ubiquitous). Ο όρος (pervasive) αναφέρεται για πρώτη φορά από τον Weiser το 1991 (13). Ο όρος περιγράφει τη διαφανή ενσωμάτωση των υπολογιστικών συσκευών στην καθημερινότητα του χρήστη, όπου η τεχνολογία χάνεται στο παρασκήνιο εστιάζοντας στο χρήστη και τις δραστηριότητές του.

Η διάδοση φθηνότερων, μικρότερων, και μεγαλύτερης ακρίβειας αισθητήρων και συσκευών έχει οδηγήσει σε μια επανάσταση στον τομέα της πληροφορικής, καθώς οι εφαρμογές που λειτουργούν σε διάχυτα συστήματα γίνονται ολοένα πιο διαδεδομένες, μπορούν να αλληλοεπιδρούν με το φυσικό περιβάλλον και να προσαρμόζονται σε μεταβαλλόμενες συνθήκες. Το σύνολο των παραγόντων του οποιουδήποτε περιβάλλοντος μέσα στο οποίο εκτελείται μια εφαρμογή ή υπηρεσία και που επηρεάζουν τη λειτουργία της ονομάζεται «πλαίσιο». Παραδείγματα αποτελούν η τοποθεσία, ο χρόνος, η θερμοκρασία, η φωτεινότητα, οι άλλοι άνθρωποι ή συσκευές σε κοντινή απόσταση κ.α. Με τη χρήση του πλαισίου, τα υπολογιστικά συστήματα μπορούν να γίνουν φιλικότερα προς τους χρήστες, πιο αποδοτικά και να παρέχουν ποιοτικότερες υπηρεσίες. Οι πληροφορίες που συλλέγονται, οι οποίες αφορούν κυρίως τις συνθήκες κάτω από τις οποίες εκτελείται μια εφαρμογή είναι ιδιαίτερα σημαντικές

για τα συστήματα, όπου το πλαίσιο μεταβάλλεται συχνά και απρόβλεπτα και οι υπηρεσίες οφείλουν να προσαρμόζονται τάχιστα και αδιάλειπτα.

Αυτή η αντίληψη του φυσικού περιβάλλοντος από τις εφαρμογές, καθώς και η χρήση των πληροφοριών που συλλέγονται για την επίτευξη των στόχων τους, είναι γνωστή ως επίγνωση πλαισίου (context-awareness) και μια σειρά από πολλά υποσχόμενες υλοποιήσεις έχουν αναπτυχθεί. Οι τεχνολογίες επίγνωσης πλαισίου αποκτούν ιδιαίτερη σημασία σε περιβάλλοντα κινητών συσκευών όπου το λειτουργικό περιβάλλον αλλάζει συνεχώς εξαιτίας της φορητότητας των συσκευών και των χαρακτηριστικών της ασύρματης τεχνολογίας επικοινωνιών. Σε τέτοια περιβάλλοντα, μέσω της επίγνωσης πλαισίου, μπορεί να επιτραπεί στις εφαρμογές να ανταποκρίνονται έξυπνα σε μεταβλητό εύρος ζώνης, αναξιόπιστες συνδέσεις και περιορισμούς όγκου δεδομένων μετάδοσης.

Οι εφαρμογές που μπορούν να γνωρίζουν το πλαίσιο και να το χρησιμοποιήσουν κατάλληλα, ώστε να προσαρμόσουν τις λειτουργίες τους σε αυτό, χαρακτηρίζονται ως εφαρμογές με επίγνωση πλαισίου. Για να αποκτήσει μια εφαρμογή χαρακτηριστικά επίγνωσης πλαισίου, χρειάζεται να αναπτυχθεί επιπρόσθετο λογισμικό. Το επιπρόσθετο αυτό λογισμικό επωμίζεται, την εύρεση, τη διαχείριση και την αξιοποίηση του πλαισίου. Σύμφωνα με τις αρχές της τεχνολογίας λογισμικού τα συστήματα με επίγνωση πλαισίου θα πρέπει να καλύπτουν τις απαιτήσεις που σχετίζονται με την ετερογένεια, την κατανομή (distribution), την διαλειτουργικότητα (interoperability) και την επεκτασιμότητα (extensibility) (77). Επιπρόσθετα θα πρέπει να σχεδιαστούν ώστε να ανταποκρίνονται σε παράγοντες όπως ξαφνικά ή απρόβλεπτα φαινόμενα, μετακίνηση φυσικών ή λειτουργικών οντοτήτων, διαμόρφωση, αβεβαιότητα, κακόβουλες ενέργειες και αποτυχίες που σχετίζονται με το φυσικό περιβάλλον (75). Η ανάπτυξη συστημάτων με αυτά τα χαρακτηριστικά είναι δύσκολη και χρονοβόρα διαδικασία για τους μηχανικούς λογισμικού.

Το ζητούμενο είναι η σχεδίαση και ανάπτυξη μιας κοινής πλατφόρμας, πάνω στην οποία να μπορούν εύκολα και ευέλικτα να αναπτυχθούν οι εφαρμογές που θα παρέχουν αυτές τις υπηρεσίες. Μία τέτοια κοινή βάση ανάπτυξης λογισμικού εξασφαλίζει την επαναχρησιμοποίηση κώδικα και ανεξαρτητοποιεί την εφαρμογή από το λογισμικό που διαχειρίζεται το πλαίσιο. Το λογισμικό υποστήριξης εφαρμογών επίγνωσης πλαισίου αποτελεί ενδιάμεσο λογισμικό μεταξύ λειτουργικού συστήματος/

συσκευής/αισθητήρων και εφαρμογής. Η ανάπτυξη κατάλληλου ενδιάμεσου λογισμικού για εφαρμογές επίγνωσης πλαισίου έχει προσελκύσει ιδιαίτερα το ενδιαφέρον της ερευνητικής κοινότητας την τελευταία δεκαετία. Η κατάλληλη αρχιτεκτονική, οι υπηρεσίες που παρέχονται αλλά και ο τρόπος αλληλεπίδρασης με το επίπεδο εφαρμογής είναι τα κυριότερα αντικείμενα της έρευνας.

2.1.Ορισμός πλαισίου

Σύμφωνα με το *The Free On-line Dictionary of Computing* πλαίσιο είναι «αυτό που περιβάλλει και δίνει νόημα σε κάτι άλλο». Αρκετές περιοχές της πληροφορικής εξετάζουν την ιδέα αυτή τα τελευταία τουλάχιστον πενήντα χρόνια (67), με στόχο να σχετιστεί η επεξεργασία πληροφοριών και οι τηλεπικοινωνίες με χαρακτηριστικά και καταστάσεις όπου αυτή η επεξεργασία πραγματοποιείται. Χαρακτηριστικά, το πλαίσιο είναι έννοια κλειδί στην Υπολογιστική Γλωσσολογία και πιο γενικά στην αλληλεπίδραση Ανθρώπου-Μηχανής. Για παράδειγμα, εδώ και αρκετά χρόνια, τα γραφικά περιβάλλοντα εφαρμογών χρησιμοποιούν δεδομένα πλαισίου για να προσαρμόσουν τα μενού τους βάσει προτιμήσεων των χρηστών.

Στο έργο που εισήγαγε πρώτο την έννοια της επίγνωσης πλαισίου (65), γίνεται αναφορά στο πλαίσιο ως τοποθεσία, ταυτότητα των κοντινών αντικειμένων και ανθρώπων, καθώς και οι αλλαγές σε αυτά τα αντικείμενα. Ως επίγνωση πλαισίου, μπορεί να οριστεί «η ικανότητα ενός συστήματος να ανακαλύπτει, να ερμηνεύει, να συμπεραίνει, να αξιοποιεί και να συλλογίζεται βάσει της περιρρέουσας πληροφορίας, ώστε να λαμβάνει αποφάσεις, να προβαίνει σε προκαθορισμένες ενέργειες και να προσαρμόζεται σε διάφορες καταστάσεις»(4).

Ένα σύστημα διάχυτου υπολογισμού, το οποίο επιδιώκει να ελαχιστοποιήσει την παρεισφρητικότητά του, θα πρέπει να έχει, κατά το δυνατό, μεγαλύτερη επίγνωση του πλαισίου του χρήστη του. Θα πρέπει να προσαρμόζει τη συμπεριφορά του με τέτοιο τρόπο ώστε να εκμεταλλεύεται τη γνώση που έχει για την κατάσταση του χρήστη και του περιβάλλοντος αυτού χώρου. Ο όγκος των πληροφοριών που το επηρεάζουν είναι πολύ μεγάλος και περιλαμβάνει από χωρικές και χρονικές παραμέτρους, δεδομένα χρήσης του συστήματος και ακόμα και συμπεράσματα που μπορούν να εξαχθούν από την συσσωμάτωση δεδομένων που δε φαίνονται σχετικά με πρώτη ματιά. Τέτοιες πληροφορίες μπορεί να αποτελούνται από τη φυσική θέση, τη φυσική κατάσταση

(θερμοκρασία σώματος, παλμοί της καρδιάς και άλλα βιομετρικά χαρακτηριστικά), τη συναισθηματική κατάσταση, τις καθημερινές συνήθειες κ.α. Ένας άνθρωπος έχει τη δυνατότητα με αυτού του είδους τις πληροφορίες να λειτουργήσει προδραστικά (proactively) , προβλέποντας με αυτό τον τρόπο τις ανάγκες του ιδίου ή κάποιου άλλου. Εξαιτίας του ότι τα υπολογιστικά συστήματα λειτουργούν κατά βάση με εντολές των χρηστών, αποτελεί ξεχωριστή και ιδιαίτερη περίπτωση η δυνατότητα ένα σύστημα να λαμβάνει μόνο του αποφάσεις, ώστε να προσαρμόζεται προνοητικά και αξιόπιστα στις ανάγκες του χρήστη, χωρίς τη συμβολή του δεύτερου.

Κατά τον Schilit(65) οι τρεις σημαντικότεροι παράγοντες του πλαισίου του χρήστη είναι το πού βρίσκεται, με ποιον είναι μαζί και ποιοι πόροι βρίσκονται κοντά του. Για αυτόν το λόγο διαχώρισε το πλαίσιο στις εξής τρεις κατηγορίες

- **Υπολογιστικό πλαίσιο**: διαθέσιμοι επεξεργαστές, συσκευές προσβάσιμες για εισαγωγή και εμφάνιση από τον χρήστη, κοντινοί πόροι όπως εκτυπωτές, οθόνες, συνδεσιμότητα, κόστος υπολογισμού και επικοινωνίας, εύρος ζώνης, δυναμικότητα δικτύου κ.α.
- **Πλαίσιο χρήστη**: διαθέσιμοι επεξεργαστές, συσκευές προσβάσιμες για εισαγωγή και εμφάνιση από τον χρήστη, κοντινοί πόροι όπως εκτυπωτές, οθόνες, συνδεσιμότητα, κόστος υπολογισμού και επικοινωνίας, εύρος ζώνης, δυναμικότητα δικτύου κ.α.
- **Φυσικό πλαίσιο**: θερμοκρασία, θόρυβος, ποσοστό υγρασίας, φωτεινότητα, ταχύτητα κ.α.

Οι Chen και Kotz (16) εισήγαγαν το χρονικό πλαίσιο στα παραπάνω. Εξαιτίας της δυνατότητας που αναφέρθηκε συμωηφισμού των δεδομένων που έχουν αντληθεί, ιδιαίτερο νόημα αποκτά και η διάσταση του χρόνου, η οποία αφορά στην ώρα μέσα στην ημέρα, την ημέρα της εβδομάδας, του μήνα, καθώς και της εποχής και του έτους. Όπως και οι υπόλοιπες κατηγορίες, μπορεί να χρησιμοποιηθεί αυτούσια, αλλά αποκτά ιδιαίτερο νόημα για το πλαίσιο του χρήστη όταν συνδυάζεται με προσωπικά δεδομένα από τις άλλες κατηγορίες.

Ο Pascoe (55) ορίζει το πλαίσιο ως το υποσύνολο των φυσικών και εννοιολογικών καταστάσεων ενδιαφέροντος μιας συγκεκριμένης οντότητας. Ένας πιο

ολοκληρωμένος και γενικός ορισμός του πλαισίου, όμως, δόθηκε από τον Dey το 2001 και αναφέρει πως:

Πλαίσιο χρήστη είναι οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα είναι ένα άτομο, μια θέση ή ένα αντικείμενο που θεωρείται σχετικό με την διάδραση μεταξύ ενός χρήστη και μίας εφαρμογής, συμπεριλαμβανομένου του χρήστη και της εφαρμογής. Πλαίσιο χρήστη είναι τυπικά η τοποθεσία, η ταυτότητα και η κατάσταση των ατόμων, των ομάδων και των υπολογιστικών και φυσικών αντικειμένων (23).

Σύμφωνα με τον ορισμό του Dey, το πλαίσιο του χρήστη, ως προς τα υπολογιστικά συστήματα που μπορεί να το αξιοποιήσουν, έχει νόημα εφόσον τα δεδομένα που το απαρτίζουν μπορούν να αξιοποιηθούν ή αξιοποιούνται από τα συστήματα αυτά. Για παράδειγμα, στην περίπτωση μίας εφαρμογής ελέγχου της φωτεινότητας της οθόνης ενός συστήματος, μπορούν να χρησιμοποιηθούν τα δεδομένα που προέρχονται από κάποιον αισθητήρα περιβάλλοντος φωτός (ambient light sensor) με τα οποία είναι εφοδιασμένες πολλές συσκευές, όπως φορητοί υπολογιστές, ταμπλέτες και κινητά τηλέφωνα. Σε περίπτωση που το περιβάλλον φως είναι έντονο, η φωτεινότητα της οθόνης αυξάνεται, ενώ σε περιβάλλοντα που υπάρχει λιγότερο φως, η υψηλή φωτεινότητα δεν είναι απαραίτητη και μειώνεται. Για την εφαρμογή αυτή, τα δεδομένα που προέρχονται από τον αισθητήρα περιβάλλοντος φωτός αποτελούν δεδομένα πλαισίου του χρήστη, εφόσον προσδιορίζουν μία ιδιότητα του περιβάλλοντος του χρήστη (αν είναι φωτεινός ή σκοτεινός ο χώρος στον οποίο βρίσκεται, καθώς και η ικανότητά του να διακρίνει το περιεχόμενο της οθόνης). Αντιθέτως, τα δεδομένα που προέρχονται από τον δέκτη GPS δεν αφορούν στην συγκεκριμένη εφαρμογή και συνεπώς, τα δεδομένα αυτά δεν αποτελούν δεδομένα πλαισίου για την εφαρμογή αυτή. Με άλλα λόγια, σύμφωνα με τον ορισμό αυτό, τα δεδομένα πλαισίου περιορίζονται βάσει της χρηστικότητάς τους για κάποιο πληροφοριακό σύστημα.

Συμφωνά με την Henricksen (33), μπορούν να γίνουν οι εξής παρατηρήσεις για το πλαίσιο χρήστη:

Οι πληροφορίες πλαισίου παρουσιάζουν ένα εύρος χρονικών χαρακτηριστικών. Τα δεδομένα πλαισίου μπορούν να χαρακτηριστούν ως στατικά και δυναμικά. Τα στατικά δεδομένα πλαισίου περιγράφουν τις πτυχές ενός συστήματος διάχυτου υπολογισμού τα οποία είναι σταθερά, όπως η ημερομηνία γέννησης ενός

ατόμου. Από την άλλη πλευρά, επειδή τα συστήματα διάχυτού υπολογισμού χαρακτηρίζονται, ως επί των πλείστων, από συχνές αλλαγές, η πλειονότητα των δεδομένων είναι δυναμική. Η διατηρησιμότητα των δυναμικών δεδομένων πλαισίου παρουσιάζει μεγάλη διαφοροποίηση ανάλογα με το είδος τους. Για παράδειγμα, σχέσεις μεταξύ συναδέλφων ή συγγενών, διαρκούν από μήνες και χρόνια, έως και παραμένουν για πάντα αμετάβλητες, ενώ η γεωγραφική τοποθεσία και η δραστηριότητα ενός χρήστη μπορεί να μεταβάλλεται από λεπτό σε λεπτό. Τα χαρακτηριστικά διατηρησιμότητας της κάθε πληροφορίας πλαισίου επηρεάζουν και τα μέσα με τα οποία συλλέγονται τα δεδομένα αυτά. Ενώ είναι λογικό να συλλέγεται το μεγαλύτερο μέρος των στατικών πληροφοριών πλαισίου άμεσα από τους ίδιους τους χρήστες, τα δεδομένα πλαισίου που αλλάζουν συχνά θα πρέπει να συλλέγονται με άλλους, έμμεσους τρόπους, όπως για παράδειγμα με τη χρήση αισθητήρων. Συχνά, στις εφαρμογές διάχυτου υπολογισμού, περισσότερο ενδιαφέρον έχει η παρούσα ευμετάβλητη κατάσταση του πλαισίου του χρήστη, απ' ό,τι η διαχρονική και αμετάβλητη.

Οι πληροφορίες πλαισίου δεν είναι τέλειες. Ένα δεύτερο χαρακτηριστικό των πληροφοριών πλαισίου είναι ότι δεν είναι τέλειες. Οι πληροφορίες μπορεί να είναι εσφαλμένες στην περίπτωση που αποτυγχάνουν να αποτυπώσουν την πραγματική κατάσταση του κόσμου που μοντελοποιούν, ασυνεπείς εάν περιέχουν αντιφατικές πληροφορίες, ή ατελείς εάν κάποια χρήσιμα δεδομένα δεν είναι γνωστά. Αυτά τα προβλήματα έχουν τις ρίζες τους σε έναν αριθμό από αίτια. Κατά πρώτον, τα περιβάλλοντα διάχυτου υπολογισμού είναι εξαιρετικά δυναμικά, το οποίο σημαίνει ότι οι πληροφορίες που τα περιγράφουν μπορούν σε σύντομο χρονικό διάστημα να απαρχαιώνονται. Αυτό το πρόβλημα συνδυάζεται και με το γεγονός ότι συχνά, οι πηγές, τα αποθετήρια δεδομένων και οι καταναλωτές των δεδομένων πλαισίου είναι κατανεμημένα και τα νέα δεδομένα που δημιουργούνται χρειάζονται επεξεργασία, ώστε να μετατραπούν σε μία μορφή που θα είναι χρήσιμη από την εφαρμογή. Αυτοί οι παράγοντες μπορούν να προκαλέσουν μεγάλες καθυστερήσεις μεταξύ της παραγωγής και της χρήσης δεδομένων πλαισίου. Κατά δεύτερο, οι παραγωγοί δεδομένων πλαισίου, όπως αισθητήρες, αλγόριθμοί μετατροπής και χρήστες μπορεί να παρέχουν εσφαλμένες πληροφορίες. Αυτό αποτελεί χαρακτηριστικό πρόβλημα στην περίπτωση που η πληροφορία πλαισίου πρέπει να έχει συναχθεί από ακατέργαστα δεδομένα αισθητήρων, ή όταν η δραστηριότητα ενός ατόμου προκύπτει ως συμπέρασμα από

άλλα δεδομένα πλαισίου όπως η τοποθεσία και τα επίπεδα θορύβου. Τέλος, ασυνέχειες και αποτυχίες μπορεί να σημάνουν ότι η διαδρομή από τον παραγωγό των δεδομένων πλαισίου έως τον καταναλωτή του διακόπτεται, με αποτέλεσμα ένα μέρος ή ακόμα και ολόκληρο το πλαίσιο να είναι άγνωστο στην εφαρμογή.

Οι πληροφορίες πλαισίου έχουν πολλές εναλλακτικές αναπαραστήσεις. Μεγάλο μέρος των πληροφοριών πλαισίου προκύπτει από αισθητήρες. Συνήθως υπάρχει μεγάλο κενό μεταξύ των δεδομένων που προέρχονται από τις εξόδους των αισθητήρων και τα επίπεδα των πληροφοριών που είναι χρήσιμα στις εφαρμογές. Αυτό το κενό πρέπει να γεφυρώνεται με διάφορα είδη επεξεργασίας των δεδομένων πλαισίου. Για παράδειγμα, ο αισθητήρας τοποθεσίας (GPS) μπορεί να παρέχει απλές συντεταγμένες, ενώ η εφαρμογή να ενδιαφέρεται για την ταυτότητα ενός κτιρίου ή δωματίου μέσα στο οποίο βρίσκεται ο χρήστης. Επιπροσθέτως, οι απαιτήσεις μπορεί να διαφοροποιούνται μεταξύ ανεξάρτητων εφαρμογών. Συμπερασματικά, ένα μοντέλο πλαισίου θα πρέπει να είναι σε θέση να εξυπηρετήσει διαφορετικές αναπαραστάσεις του ίδιου πλαισίου σε διαφορετικές μορφές και σε διαφορετικά επίπεδα αφαίρεσης. Παράλληλα, θα πρέπει να είναι σε θέση να συλλάβει τις σχέσεις που υπάρχουν μεταξύ αυτών των διαφορετικών αναπαραστάσεων.

Οι πληροφορίες πλαισίου είναι ιδιαίτερα αλληλένδετες. Πολλές από τις πληροφορίες πλαισίου εκφράζονται ως σχέσεις μεταξύ οντοτήτων, όπως μεταξύ ανθρώπων, των συσκευών τους και των καναλιών επικοινωνίας τους. Αυτές οι σχέσεις μπορεί να απαντούν ερωτήσεις, για παράδειγμα, όπως «ποια συσκευή ανήκει σε ποιον», «ποιο κανάλι επικοινωνίας χρησιμοποιείται για τι» και «ποιες συσκευές και χρήστες είναι σε κοντινή απόσταση μεταξύ τους». Παρόλα αυτά, υπάρχουν και άλλες, λιγότερο προφανείς σχέσεις μεταξύ πληροφοριών πλαισίου. Οι πληροφορίες αυτές μπορεί να σχετίζονται με κάποιους κανόνες παραγωγής τους, οι οποίοι περιγράφουν πώς οι πληροφορίες αυτές αποκτώνται συνδυάζοντας έναν η περισσότερους τύπους δεδομένων. Για παράδειγμα, η παρούσα δραστηριότητα ενός χρήστη μπορεί να εξαχθεί από άλλες πληροφορίες πλαισίου, όπως η τοποθεσία στην οποία βρίσκεται και το ιστορικό περασμένων παρόμοιων δραστηριοτήτων. Ένα τέτοιο είδος σχέσης, όπου τα χαρακτηριστικά μίας εξαγόμενης πληροφορίες είναι άρρηκτα συνδεδεμένα με τις ιδιότητες των πληροφοριών από τις οποίες εξάγονται, αποτελεί μία σχέση εξάρτησης.

Επιπλέον, επεκτείνοντας και τον παραπάνω συλλογισμό, σύμφωνα με τον Gu (32), το πλαίσιο μπορεί να διαχωριστεί σε δύο κατηγορίες, στο άμεσο και το έμμεσο, ανάλογα με τον τρόπο με τον οποίο έχει αποκτηθεί.

Η **άμεση** πληροφορία πλαισίου έχει απευθείας αποκτηθεί ή παραχωρηθεί από έναν προμηθευτή πλαισίου, ο οποίος μπορεί να είναι είτε μία εσωτερική πηγή, όπως για παράδειγμα ένας πάροχος εσωτερικής υπηρεσίας, είτε μία εξωτερική πηγή όπως για παράδειγμα ένας διακομιστής πληροφοριών. Το άμεσο πλαίσιο μπορεί περαιτέρω να κατηγοριοποιηθεί σε ορισμένο πλαίσιο και αισθητό. Το αισθητό αποκτάται από φυσικούς αισθητήρες, όπως η θερμοκρασία ενός δωματίου, με τη χρήση ενός θερμομέτρου ή ενός εικονικού αισθητήρα, για παράδειγμα μία μετεωρολογική υπηρεσία διαδικτύου. Το ορισμένο πλαίσιο όπως μαρτυρά η ονομασία του ορίζεται από τον ίδιο τον χρήστη, για παράδειγμα «αγαπημένες ταινίες».

Η **έμμεση** πληροφορία πλαισίου προέρχεται από την ανάλυση και τον συνδυασμό του άμεσου πλαισίου μέσω της διαδικασίας ενός συλλογισμού. Για παράδειγμα, η τρέχουσα κατάσταση ενός ατόμου (λούζεται) μπορεί να εξαχθεί από την τοποθεσία του (μπάνιο), την κατάσταση του θερμοσίφωνα (αναμμένος), την κατάσταση της ντουζιέρας (νερό τρέχει) και την κατάσταση της πόρτας (κλειστή).

Μία άλλη διάκριση γίνεται ανάμεσα σε ενεργητικό και παθητικό πλαίσιο (16). Στην πρώτη περίπτωση έχουμε την αυτόματη προσαρμογή της εφαρμογής στο νέο πλαίσιο, δηλαδή αυτόματες εκτελέσεις εργασιών ή αλλαγή συμπεριφοράς με την λήψη της νέας πληροφορίας. π.χ. αναπροσαρμογή των οδηγιών σε ένα σύστημα GPS αν για οποιονδήποτε λόγο δεν ακολουθηθούν οι αρχικές οδηγίες. Στην δεύτερη περίπτωση η εφαρμογή μετά την ενημέρωσή της με το νέο πλαίσιο ενημερώνει τον χρήστη ή αποθηκεύει την πληροφορία για μελλοντική χρήση, περιμένοντας κάποια εντολή.

Διαφορετικές κατηγορίες πλαισίου έχουν διαφορετικά χαρακτηριστικά, δηλαδή το αισθητό πλαίσιο είναι κυρίως δυναμικό με διατηρησιμότητα από δευτερόλεπτα έως ώρες, ενώ το ορισμένο πλαίσιο είναι πιθανότερο να είναι στατικό. Από την κατηγοριοποίηση διαφόρων πληροφοριών πλαισίου και γνωρίζοντας τα χαρακτηριστικά τους, είναι δυνατό να πραγματοποιηθούν αποτελεσματικότεροι συλλογισμοί πλαισίου, πάνω σε διαφορετικά είδη πλαισίου, ώστε να λυθούν προβλήματα αξιοπιστίας ή αντικρουόμενων πληροφοριών. Για παράδειγμα,

γνωρίζουμε ότι το ορισμένο πλαίσιο είναι πιο αξιόπιστο από το αισθητό πλαίσιο και μπορεί να χρησιμοποιηθεί κατά προτεραιότητα, εάν και τα δύο είναι διαθέσιμα.

2.2. Προγραμματισμός με Διαχείριση Πληροφορίας Πλαισίου

Ένα νέο σχετικά πεδίο, στο οποίο παρουσιάζεται αυξημένο ενδιαφέρον ως προς τη χρήση δεδομένων πλαισίου είναι τα φορητά υπολογιστικά συστήματα. Ενώ υπήρξε ένα πρώτο κύμα όπου οι φορητοί υπολογιστές βασιζόταν σε φορητούς υπολογιστές γενικής χρήσης και κατά βάση στη χρήση της τοποθεσίας τους, ένα δεύτερο κύμα έδωσε έμφαση σε πολύ πιο μικρές φορητές συσκευές και σε ένα ενδιαφέρον να σχετιστούν αυτές με την περιβάλλουσα κατάσταση χρήσης τους. Αυτές οι συσκευές περιγράφονται ως μικροί φορητοί υπολογιστές και ορίζονται ως υπολογιστικές συσκευές, οι οποίες λειτουργούν και χρησιμοποιούνται εν κινήσει, ενώ χαρακτηρίζονται από μία μετατόπιση από τη γενική χρήση προς την υποστήριξη συγκεκριμένων σκοπών. Τέτοιες συσκευές απαρτίζονται από Προσωπικούς Ψηφιακούς Οδηγούς (PDAs), έξυπνα κινητά τηλέφωνα (Smartphones) και φορετούς υπολογιστές (wearable computers) (67).

Αυτή η αντίληψη του φυσικού περιβάλλοντος από τις εφαρμογές, καθώς και η χρήση των πληροφοριών που συλλέγονται για την επίτευξη των στόχων τους, έχει παράγει μια σειρά από πολλά υποσχόμενες . Όπως προκύπτει, οι τεχνολογίες επίγνωσης πλαισίου αποκτούν ιδιαίτερη σημασία σε περιβάλλοντα κινητών συσκευών όπου το λειτουργικό περιβάλλον αλλάζει συνεχώς εξαιτίας της φορητότητάς τους και των χαρακτηριστικών ασύρματης τεχνολογίας επικοινωνιών που διαθέτουν. Σε αρχικό στάδιο, η επίγνωση πλαισίου αφορούσε μεθόδους αντίληψης της τοποθεσίας, όπως για παράδειγμα τους δέκτες GPS. Βέβαια η τοποθεσία είναι μόνο η μία όψη του φυσικού περιβάλλοντος και όπως συνήθως διαφαίνεται, χρησιμοποιείται πολλές φορές ως μία προσέγγιση ενός πολυπλοκότερου πλαισίου. Πέρα από την τοποθεσία, η έρευνα προσανατολίζεται και προς την αντίληψη άλλων χαρακτηριστικών που συνεισφέρουν στο πλαίσιο, εξαιτίας του ότι όσο περισσότερα γνωρίζει μία υπέρ-κινητή συσκευή ως προς τις συνθήκες χρήσης της, τόσο πιο πολύ μπορεί να φανεί χρήσιμη στον χρήστη της.

Ο όρος που χρησιμοποιήθηκε είναι «προγραμματισμός επίγνωσης πλαισίου» (context-aware computing) και ο οποίος εισήχθη και συζητήθηκε για πρώτη φορά από

τους Schilit και Theimer (66) το 1994, όπου το όρισαν ως λογισμικό που «προσαρμόζεται σύμφωνα με την τοποθεσία χρησιμοποίησης του, τους ανθρώπους και τα αντικείμενα που βρίσκονται κοντά του, όπως επίσης και στις αλλαγές που υφίστανται τα αντικείμενα αυτά προϊόντος του χρόνου.» Η πρώτη όμως ευρεία έρευνα πάνω στον προγραμματισμό επίγνωσης πλαισίου μπορεί να θεωρηθεί ότι πραγματοποιήθηκε νωρίτερα, το 1992 και ήταν το Olivetti Active Badge (76). Από τότε αρκετοί ορισμοί έχουν δοθεί, ορισμένους εκ των οποίων θα αναφερθούν παρακάτω.

Οι Hull et al. (5) και Pascoe et al. (55) ορίζουν ως προγραμματισμό επίγνωσης πλαισίου την ικανότητα των υπολογιστικών συσκευών να ανιχνεύουν και να αισθάνονται, να ερμηνεύουν και να αντιδρούν σε θέματα του τοπικού περιβάλλοντος του χρήστη και των ίδιων των συσκευών. Ο Dey (5) αναφέρει πως η γνώση του πλαισίου του χρήστη οδηγεί σε αυτοματοποίηση του λειτουργικού συστήματος ενώ ο Salber (23) ορίζει ως επίγνωση πλαισίου την ικανότητα παροχής μέγιστης ευελιξίας μίας υπολογιστικής υπηρεσίας βάσει ανίχνευσης του πλαισίου σε πραγματικό χρόνο. Ακόμα, ο Ryan (56) ορίζει ως εφαρμογές επίγνωσης πλαισίου, τις εφαρμογές εκείνες που λαμβάνουν ερεθίσματα από αισθητήρες περιβάλλοντος και παρέχουν στους χρήστες ένα φάσμα φυσικών και λογικών επιλογών, σύμφωνα με τις παρούσες ανάγκες και δραστηριότητές τους. Τέλος, ο Brown (13) τις ορίζει ως εφαρμογές που παρέχουν αυτόματα πληροφορίες ή/και πράττουν σύμφωνα με το παρόν πλαίσιο του χρήστη όπως αυτό έχει ανιχνευθεί από αισθητήρες.

Ένας πλήρης και συνοπτικός ορισμός προέρχεται από τον Dey (2001) και αναφέρει πως:

Ένα σύστημα έχει επίγνωση πλαισίου εάν χρησιμοποιεί το πλαίσιο ώστε να παρέχει σχετικές πληροφορίες ή/και υπηρεσίες στον χρήστη, όπου η σχετικότητά τους εξαρτάται από την εργασία του χρήστη. (24)

Για παράδειγμα, η παρούσα κατάσταση μια πυξίδας σε ένα κινητό τηλέφωνο, η παρούσα κατάσταση του αισθητήρα του περιβάλλοντος φωτός μιας φορητής συσκευής ή η τοποθεσία και τα κοντινότερα ασύρματα δίκτυα χρησιμοποιείται με σκοπό μια αυτόματη προσαρμογή (την αύξηση της φωτεινότητας, την ενεργοποίηση της ασύρματης σύνδεσης) ή την ενεργοποίηση μια συναφούς επιλογής (επισήμανση γεωγραφικών σημείων ενδιαφέροντος κοντά στον χρήστη, ή η εκτύπωση ενός εγγράφου στον πλησιέστερο εκτυπωτή).

2.3. Συνεργατική Επίγνωση Πληροφορίας Πλαισίου

Στα σύγχρονα κινητά κατανεμημένα υπολογιστικά συστήματα, στα οποία οι χρήστες αλληλεπιδρούν με πληθώρα διαφορετικών κινητών ή σταθερών υπολογιστών κατά τη διάρκεια της μέρας, ο υπολογισμός δεν πραγματοποιείται σε ένα μοναδικό σημείο, σε ένα συγκεκριμένο περιβάλλον, αλλά, αντιθέτως, εκτείνεται σε πολλές καταστάσεις, όπως το γραφείο, η αίθουσα συνεδριάσεων, το σπίτι, το αεροδρόμιο, το αεροπλάνο κλπ. Ένας σημαντικός παράγοντας που επηρεάζει αυτόν τον τρόπο λειτουργίας είναι το ταχέως μεταβαλλόμενο περιβάλλον εκτέλεσης στο οποίο εκτίθενται οι χρήστες και οι εφαρμογές μακρόχρονης εκτέλεσης. Όσο οι χρήστες μετακινούνται, το σύνολο των κινητών και σταθερών αντικειμένων με τα οποία αλληλεπιδρούν ενδέχεται να αλλάζουν, δημιουργώντας ένα άκρως δυναμικό περιβάλλον εκτέλεσης στο οποίο η τοποθεσία είναι σημαντική. Όταν περισσότεροι από έναν χρήστες βρίσκονται στο ίδιο περιβάλλον, ή μοιράζονται κοινά ενδιαφέροντα ή προτιμήσεις, οι φορητές συσκευές που έχουν αποτελούν κόμβους που επεξεργάζονται παρόμοιες πληροφορίες πλαισίου (66).

Οι κόμβοι αυτοί, οι οποίοι κινούνται προς διάφορες θέσεις μπορούν να διαδώσουν τις πληροφορίες αυτές σε συνεργασία. Σε αυτή την περίπτωση, ο πρώτος κόμβος που θα συλλέξει ή θα τα επεξεργαστεί μία πληροφορία πλαισίου μπορεί να την μεταδώσει στη συνέχεια και σε παραπλήσιους κόμβους, όταν αυτοί το ζητήσουν. Με την συνεργατική επίγνωση πληροφορίας πλαισίου υποδηλώνεται η κατανόηση μίας πληροφορίας πλαισίου από ολόκληρη την ομάδα κόμβων με κοινό ενδιαφέρον για αυτήν, με αποτέλεσμα να παρέχεται ανώτερη ποιότητα πληροφορίας πλαισίου σε κάθε έναν κόμβο ξεχωριστά(3),(12). Ως συνεργατική πληροφορία θεωρείται η πληροφορία πλαισίου που αποκτάται μέσω δικτύου μεταξύ αισθητήρων και των συσκευών υψηλού επιπέδου και μπορεί να χρησιμοποιηθεί για να αυξήσει την κοινή κατανόηση για όλους τους γειτονικούς κόμβους ενός εξ αυτών, να βελτιώσει τη διαθεσιμότητα των προσφερόμενων υπηρεσιών, ή να ενισχύσει την αξιοπιστία της πληροφορίας, μέσω των πρόσθετων πληροφοριών από τους γειτονικούς κόμβους.

2.4. Διαχείριση Πληροφοριών Πλαισίου

Ενώ αρχικά η κοινότητα της πληροφορικής αντιμετώπιζε το ζήτημα του πλαισίου ως θέμα που αφορούσε στην γεωγραφική τοποθεσία του χρήστη (24), τα

τελευταία χρόνια η έννοιά του θεωρείται ότι δεν αποτελεί απλώς μία κατάσταση, αλλά μέρος μίας διαδικασίας, στην οποία εμπλέκονται και οι χρήστες. Κατ' αυτό τον τρόπο, σύνθετα και γενικά μοντέλα έχουν προταθεί (10), τα οποία υποστηρίζουν εφαρμογές που το χρησιμοποιούν για να προσαρμόζουν διεπαφές, να επιλέγουν τα σύνολο των δεδομένων που είναι σχετικά με την εφαρμογή, να αυξήσουν την ακρίβεια της ανάκτησης πληροφοριών, να ανακαλύψουν υπηρεσίες, να κάνουν την αλληλεπίδραση με τον χρήστη πιο σιωπηρή, ή να χτίσουν έξυπνα περιβάλλοντα. Τα συστήματα που έχουν επίγνωση πλαισίου ασχολούνται με τη συλλογή, την αποθήκευση, το φιλτράρισμα και τον συνδυασμό των πληροφοριών πλαισίου, όπως και την εκτέλεση εντολών βάσει της πληροφορίας που τα δεδομένα αυτά παρέχουν. Τα συστήματα αυτά επιφορτίζονται

- την απόκτηση και συλλογή των πληροφοριών πλαισίου (Context Acquisition),
- την αφαιρετική προσαρμογή τους, αναπαράσταση και αποθήκευσή τους (Context Handling) και
- την κατανόησή τους, μέσω ενός συλλογισμού και προσαρμογής πλαισίου (Context Reasoning)

Ο διαχωρισμός του εντοπισμού και της χρήσης των πληροφοριών πλαισίου είναι απαραίτητος για να αυξηθεί η επεκτασιμότητα και επαναχρησιμοποίηση ενός συστήματος επίγνωσης πλαισίου. Η παρακάτω πολυεπίπεδη αρχιτεκτονική (Πίνακας 1) επεκτείνει τα επίπεδα του εντοπισμού και της χρήσης του πλαισίου, προσθέτοντας και τη διερμηνεία του, όπως και τον συλλογισμό του (2).

Εφαρμογή
Αποθήκευση/Διαχείριση
Προ-επεξεργασία
Ανάκτηση ακατέργαστων δεδομένων
Αισθητήρες

Πίνακας 1: Πολυεπίπεδη αρχιτεκτονική συστημάτων επίγνωσης πλαισίου

Καθώς η δραστηριότητα του χρήστη είναι κρίσιμη για πολλές εφαρμογές, η έρευνα στην επίγνωση πλαισίου έχει επικεντρωθεί περισσότερο στα πεδία της επίγνωσης γεωγραφικής θέσης και επίγνωσης δραστηριοτήτων.

Η έννοια της διαχείρισης πληροφοριών και δεδομένων πλαισίου συνήθως αναφέρεται στον συντονισμό και την παράλληλη διακίνηση και διανομή τους. Οι διακομιστές ή οι υπηρεσίες που επωμίζονται τη διαχείριση αυτών των δεδομένων αναλαμβάνουν να τα αποθηκεύουν και να παρέχουν πρόσβαση σε αυτά, ώστε να είναι δυνατή η ανάκτηση, η σύγκριση και η ανανέωση της πληροφορίας. Μέχρι πρότινος, οι προγραμματιστές, καθώς και οι υπόλοιποι επιστήμονες των Η/Υ, όριζαν τους αισθητήρες και τον τρόπο συλλογής των δεδομένων, και καθόριζαν τις αντίστοιχες συμπεριφορές που θα έπρεπε να έχει το σύστημα, είτε άμεσα στον πηγαίο κώδικα, είτε έμμεσα μέσω άλλων εργαλείων.

Ο Winograd (59) περιγράφει τρία διαφορετικά μοντέλα διαχείρισης πλαισίου για τον συγχρονισμό πολλαπλών διεργασιών και συστατικών στοιχείων:

- *Μικροεφαρμογές (Widgets)*: Προερχόμενο από τα ομώνυμα στοιχεία γραφικού περιβάλλοντος, μία μικροεφαρμογή είναι ένα συστατικό λογισμικού που παρέχει ένα κοινό περιβάλλον εργασίας για έναν αισθητήρα υλικού. Οι μικροεφαρμογές κρύβουν τις χαμηλού επιπέδου λεπτομέρειες ανίχνευσης και διευκολύνουν την ανάπτυξη των εφαρμογών, εξαιτίας της δυνατότητάς τους να επαναχρησιμοποιούνται. Λόγω της ενθυλάκωσης σε μικροεφαρμογές, είναι δυνατή η ανταλλαγή πληροφοριών ανάμεσα σε μικροεφαρμογές που παρέχουν το ίδιο είδος δεδομένων πλαισίου, για παράδειγμα την ανταλλαγή μίας μικροεφαρμογής ραδιοσυχνοτήτων με μία μικροεφαρμογή κάμερας, με σκοπό την συλλογή δεδομένων τοποθεσίας. Οι μικροεφαρμογές ελέγχονται συνήθως από έναν διαχειριστή μικροεφαρμογών (widget manager). Η προσέγγιση των στενά συνδεδεμένων μικροεφαρμογών αυξάνει την αποτελεσματικότητα, αλλά δεν είναι αξιόπιστη όσον αφορά στις βλάβες των στοιχείων τους.
- *Υπηρεσίες Δικτύου*: Αυτή η πιο ευέλικτη προσέγγιση μοιάζει με την αρχιτεκτονική του διακομιστή δεδομένων πλαισίου. Αντί ενός γενικού διαχειριστή μικροεφαρμογών, διαφορετικές μεταξύ τους τεχνικές

αναζήτησης χρησιμοποιούνται για την εύρεση υπηρεσιών δικτύου. Αυτή η προσέγγιση παρέχει μεγαλύτερη αξιοπιστία από την μία πλευρά, αλλά από την άλλη, δεν είναι τόσο αποτελεσματική όσο μια αρχιτεκτονική μικροεφαρμογών, λόγω της πιθανής πολυπλοκότητας του δικτύου και του πλήθος των στοιχείων από τα οποία αυτό θα αποτελείται.

- *Μοντέλο μαυροπίνακα (blackboard)*: Σε αντίθεση με την επικεντρωμένη στις διαδικασίες θεώρηση των μικροεφαρμογών, καθώς και του μοντέλου που βασίζεται στην αναζήτηση υπηρεσιών δικτύου, το μοντέλο του μαυροπίνακα παρουσιάζει μία θεώρηση επικεντρωμένη στα δεδομένα. Σε αυτή την ασύμμετρη προσέγγιση, οι διαδικασίες αναρτούν μηνύματα σε ένα κοινό μέσο, το επονομαζόμενο μαυροπίνακα, και εγγράφονται σε αυτό ώστε να ενημερωθούν όταν κάποιο συγκεκριμένο γεγονός συμβεί. Τα πλεονεκτήματα αυτού του μοντέλου είναι η απλότητα που παρέχει στην πρόσθεση νέων πηγών δεδομένων πλαισίου, καθώς και η δυνατότητα που παρέχει για εύκολη διαμόρφωση αυτών. Ανασταλτικό παράγοντα, όμως, αποτελεί το γεγονός ότι υπάρχει η ανάγκη ενός κεντρικού διακομιστή, ο οποίος θα φιλοξενεί τον μαυροπίνακα, όπως και η έλλειψη αποτελεσματικότητας στην επικοινωνία, καθώς απαιτούνται δύο λυκίσκοι (hops) για κάθε επικοινωνία.

2.4.1. Συλλογή Δεδομένων Πλαισίου

Η πληροφορία πλαισίου αντλείται από μια σειρά από διαφορετικές πηγές δεδομένων, όπως αισθητήρες θέσης, αισθητήρες καιρού ή κίνησης, οθόνες υπολογιστών και δικτύων, καθώς και την κατάσταση των υπολογιστικών ή ανθρώπινων υπηρεσιών. Ενώ τα ακατέργαστα δεδομένα του αισθητήρα μπορεί να είναι επαρκή για μερικές εφαρμογές, σε πολλές άλλες, αυτά τα ακατέργαστα δεδομένα είναι που απαιτούνται ώστε να μετασχηματιστούν ή να συντηχθούν με άλλα δεδομένα του αισθητήρα για να είναι χρήσιμα. Συγκεντρώνοντας πολλές εισόδους αισθητήρων μπορούμε να αποκομίσουμε πλαίσιο υψηλότερου επιπέδου, με αποτέλεσμα οι εφαρμογές να μπορούν να προσαρμοστούν με μεγαλύτερη ακρίβεια.

Μια βασική πρόκληση στον τομέα της διεισδυτικής υπολογιστικής είναι η συλλογή ακατέργαστων δεδομένων από χιλιάδες διαφορετικούς αισθητήρες, η

επεξεργασία των δεδομένων ώστε να αποτελέσουν πληροφορίες πλαισίου, καθώς και η διάδοση των πληροφοριών αυτών σε εκατοντάδες διαφορετικές εφαρμογές που εκτελούνται σε χιλιάδες συσκευές παράλληλα, με την κλιμάκωση σε μεγάλους αριθμούς πηγών, εφαρμογών, και χρηστών. Με αυτό τον τρόπο εξασφαλίζονται πληροφορίες πλαισίου από μη εγκεκριμένες χρήσεις και ικανοποιείται η ιδιωτικότητα των χρηστών. (17)

Ως *προμηθευτές δεδομένων πλαισίου* θεωρούνται οι πηγές από τις οποίες μπορεί να προέρχονται τα δεδομένα πλαισίου. Σε ένα σύστημα παρέχονται πληροφορίες είτε κατευθείαν από τον χρήστη, είτε από βάσεις δεδομένων, είτε από αισθητήρες. Έτσι, με βάση την πηγή προέλευσης της πληροφορίας, γίνεται και η κατηγοριοποίηση του είδους πλαισίου.

Η σχεδίαση του συστήματος διαχείρισης των προμηθευτών, στην αρχιτεκτονική ενός συστήματος επίγνωσης πλαισίου, περιλαμβάνει τα εξής σημεία:

- **Επικοινωνία προμηθευτών εφαρμογής**: Είτε με την μέθοδο εγγραφής – ενημέρωσης, μέσω αποστολής γεγονότων είτε με την μέθοδο ερωτοαπαντήσεων (Q&A).
- **Συχνότητα ενημέρωσης**: Καθορισμός συχνότητας ενημέρωσης της εφαρμογής από το σύστημα αισθητήρων, είτε με απευθείας ορισμό συχνότητας, είτε με φίλτρα γεγονότων. Τα φίλτρα καθορίζουν, σύμφωνα με τη ρύθμιση που έχει κάνει ο χρήστης, ποια γεγονότα είναι αρκετά σημαντικά, ώστε να ενημερωθεί η εφαρμογή.
- **Εύρεση κατανεμημένων προμηθευτών, σε περιβάλλον διάχυτου προγραμματισμού**: Σε διάχυτα περιβάλλοντα υπάρχουν δύο περιπτώσεις κατανομής προμηθευτών ως προς την εφαρμογή. Στην πρώτη περίπτωση, δεν υπάρχουν ορατοί προμηθευτές, οπότε η εφαρμογή πρέπει να πραγματοποιήσει αναζήτηση σε ένα κατανεμημένο δίκτυο προμηθευτών και να ανιχνεύσει τους κατάλληλους. Στην δεύτερη περίπτωση, η εφαρμογή έχει ορατούς προμηθευτές, αλλά δε γνωρίζει ποιος από όλους θα της παρέχει τη ζητούμενη πληροφορία πλαισίου. Ανεξάρτητα από την περίπτωση, πρέπει να υπάρχουν κατάλληλοι μηχανισμοί εύρεσης και επικοινωνίας με τους προμηθευτές του πλαισίου.

Επιπρόσθετα, η μέθοδος με την οποία γίνεται η συλλογή των δεδομένων πλαισίου είναι ιδιαιτέρως σημαντική στη σχεδίαση συστημάτων επίγνωσης πλαισίου, γιατί προκαθορίζει, τουλάχιστον σε κάποιο βαθμό, το αρχιτεκτονικό στυλ του συστήματος. Ο Baldauf (8)παρουσιάζει τρεις διαφορετικές προσεγγίσεις όσον αφορά στον τρόπο απόκτησης πληροφοριών πλαισίου:

- **Άμεση πρόσβαση αισθητήρα**: Αυτή η προσέγγιση χρησιμοποιείται πιο συχνά σε συσκευές που έχουν ενσωματωμένους αισθητήρες. Το λογισμικό του χρήστη συλλέγει τις επιθυμητές πληροφορίες απευθείας από τους αισθητήρες, δηλαδή δεν υπάρχει ενδιάμεσο επίπεδο για την απόκτηση και την επεξεργασία των δεδομένων του αισθητήρα. Οι οδηγοί για τους αισθητήρες είναι ενσωματωμένοι στην εφαρμογή και για αυτόν τον λόγο αυτή η στενά συζευγμένη μέθοδος χρησιμοποιείται μόνο σε σπάνιες περιπτώσεις. Συνεπώς, δεν είναι κατάλληλη για κατανεμημένα συστήματα, λόγω του ότι στερείται το συστατικό που θα την κάνει ικανή να διαχειριστεί ταυτόχρονα προσπελάσεις δεδομένων πολλαπλών αισθητήρων.
- **Υποδομή Ενδιάμεσου Λογισμικού**: Ο σύγχρονος σχεδιασμός λογισμικού χρησιμοποιεί μεθόδους ενθυλάκωσης για τον διαχωρισμό, για παράδειγμα επιχειρηματική λογική (business logic) και γραφικών διεπαφών χρήστη. Η προσέγγιση ενδιάμεσου λογισμικού εισάγει μια πολυεπίπεδη αρχιτεκτονική στα συστήματα επίγνωσης πλαισίου, με σκοπό την απόκρυψη χαμηλού επιπέδου λεπτομέρειες ανίχνευσης.
- **Διακομιστής Πλαισίου**: Το επόμενο λογικό βήμα είναι να επιτρέπεται σε πολλαπλούς χρήστες να έχουν πρόσβαση σε απομακρυσμένες πηγές δεδομένων. Αυτή η κατανεμημένη προσέγγιση επεκτείνει την αρχιτεκτονική που βασίζεται στο ενδιάμεσο λογισμικό, με την εισαγωγή ενός συστατικού πρόσβασης απομακρυσμένης διαχείρισης. Οι λειτουργίες συγκέντρωσης των δεδομένων που εκτελούνταν από τους αισθητήρες μεταφέρονται τώρα στον διακομιστή πλαισίου για τη διευκόλυνση της ταυτόχρονης πολλαπλής προσπέλασης. Εκτός από την επαναχρησιμοποίηση των αισθητήρων, η χρήση ενός διακομιστή πλαισίου έχει το πλεονέκτημα της ανακούφισης των συσκευών των χρηστών από εντατικές διαδικασίες που αναλώνουν πόρους. Αν υπολογιστεί επίσης πως

η πλειοψηφία των τερματικών συσκευών που χρησιμοποιούνται σε συστήματα επίγνωσης πλαισίου είναι κινητές και φορητές συσκευές με περιορισμένη υπολογιστική ισχύ, χωρητικότητα μνήμης, ενέργεια, αυτό αποτελεί μία σημαντική πτυχή. Παράλληλα, κατά τον σχεδιασμό ενός συστήματος επίγνωσης πλαισίου που βασίζεται στην αρχιτεκτονική χρήστη – διακομιστή πρέπει να δοθεί έμφαση στην επιλογή κατάλληλων πρωτοκόλλων, τις επιδόσεις του δικτύου, την ποιότητα των παρεχόμενων υπηρεσιών κ.α.

Το πρώτο επίπεδο (Πίνακας 1) απαρτίζεται από μία συλλογή διαφόρων αισθητήρων. Αξίζει να σημειωθεί ότι με τη λέξη «αισθητήρας» δεν εννοούνται μόνο αισθητήρες υλικού, αλλά και κάθε είδους πηγή δεδομένων τα οποία μπορούν να προσφέρουν πληροφορίες πλαισίου. Ανάλογα με τον τρόπο με τον οποίο συλλέγονται τα δεδομένα, οι αισθητήρες μπορούν να χωριστούν σε τρεις κατηγορίες (38):

- **Φυσικοί Αισθητήρες** : Οι αισθητήρες πιο συχνής χρήσης είναι οι φυσικοί αισθητήρες. Στις μέρες μας υπάρχουν διαθέσιμοι αισθητήρες ικανοί να συλλάβουν σχεδόν κάθε μορφή φυσικού δεδομένου.
- **Εικονικοί Αισθητήρες**: Οι εικονικοί αισθητήρες συλλέγουν πληροφορίες πλαισίου που πηγάζουν από εφαρμογές λογισμικού και υπηρεσίες. Για παράδειγμα, είναι δυνατό να εξακριβωθεί η τοποθεσία ενός ατόμου, όχι μόνο χρησιμοποιώντας συστήματα εντοπισμού (φυσικοί αισθητήρες) αλλά και μέσω εικονικών αισθητήρων. Για παράδειγμα, ένα ηλεκτρονικό ημερολόγιο, ηλεκτρονικά μηνύματα κλπ. Άλλο ένα παράδειγμα δεδομένων πλαισίου που μπορούν να συλλεχθούν από εικονικούς αισθητήρες είναι ενέργειες του χρήστη από κινήσεις του ποντικιού του υπολογιστή ή του πληκτρολογίου.
- **Λογικοί Αισθητήρες**: Αυτοί οι αισθητήρες κάνουν χρήση διαφόρων πηγών πληροφοριών και συνδυάζουν φυσικούς και εικονικούς αισθητήρες με πρόσθετες πληροφορίες από βάσεις δεδομένων ή άλλες πηγές με σκοπό την επίλυση συνθετότερων ζητημάτων. Για παράδειγμα, ένας λογικός αισθητήρας μπορεί να δομηθεί ώστε να εντοπίζει τη θέση του χρήστη, αναλύοντας τις εισόδους σε έναν

υπολογιστή, παράλληλα με μία βάση δεδομένων για τις πληροφορίες τοποθεσίας του Η/Υ.

Το είδος της πρωτογενούς πληροφορίας πλαισίου μπορεί να προέρχεται από διαφορετικούς φυσικούς αισθητήρες. Για παράδειγμα (68).:

- **Φως**: Φωτοδίοδοι, Αισθητήρες χρωμάτων, υπέρυθροι, υπεριώδεις αισθητήρες, κλπ
- **Οπτική**: Ψηφιακές κάμερες, καταγραφείς βίντεο
- **Επιτάχυνση**: Φωτοκύτταρα, Διακόπτες Υδραργύρου, Επιταχυνσιόμετρα
- **Ακουστική**: Μικρόφωνα
- **Τοποθεσία**: GPS, GSM, WiFi
- **Επαφή**: Ενσωματωμένοι αισθητήρες επαφής
- **Θερμοκρασία**: Θερμόμετρα
- **Φυσικά Χαρακτηριστικά**: Βιομετρητές που καταγράφουν αντίσταση δέρματος, αρτηριακή πίεση κλπ

Το δεύτερο επίπεδο (Πίνακας 1) ευθύνεται για την ανάκτηση των ακατέργαστων δεδομένων. Κάνει χρήση των κατάλληλων οδηγών των φυσικών αισθητήρων και των Διεπαφών Προγραμματισμού Εφαρμογών (APIs) των εικονικών και των λογικών αισθητήρων. Η λειτουργία ερωτημάτων συχνά εφαρμόζεται σε επαναχρησιμοποιήσιμα στοιχεία λογισμικού, τα οποία κάνουν τις πληροφορίες χαμηλού επιπέδου πρόσβασης στο υλικό των αισθητήρων διαθέσιμες, παρέχοντας πιο γενικές μεθόδους. Χρησιμοποιώντας διεπαφές για τα στοιχεία που ευθύνονται για παρόμοιους τύπους δεδομένων πλαισίου, αυτά τα στοιχεία μπορούν να αποτελέσουν στοιχεία συναλλαγών μεταξύ διαφορετικών αντικειμένων. Έτσι, για παράδειγμα, είναι εφικτό να αντικατασταθεί ένα σύστημα Ταυτοποίησης Ραδιοσυχνοτήτων (RFID) από ένα Παγκόσμιας Σύστημα Θεσιθεσίας (GPS), χωρίς να πραγματοποιηθεί κάποια σημαντική μετατροπή σε αυτό ή στα ανώτερα επίπεδα.

2.4.2. Συλλογισμός Πλαισίου

Το τρίτο επίπεδο, αυτό της προ-επεξεργασίας, το οποίο δεν υλοποιείται σε όλα τα συστήματα επίγνωσης πλαισίου, μπορεί να παρέχει χρήσιμες πληροφορίες εφόσον τα ακατέργαστα δεδομένα είναι υπερβολικά ασαφή. Το επίπεδο αυτό ευθύνεται για τον

συλλογισμό και την ερμηνεία των πληροφοριών πλαισίου. Οι αισθητήρες που ερωτώνται στα κατώτερα επίπεδα, πολλές φορές, παρέχουν τεχνικής φύσης δεδομένα, τα οποία δεν είναι χρήσιμα για τους προγραμματιστές εφαρμογών. Έτσι, το επίπεδο αυτό ανάγει τα δεδομένα του δεύτερου επιπέδου σε πιο αφαιρετικές δομές. Οι μετασχηματισμοί που πραγματοποιούνται συμπεριλαμβάνουν την εξαγωγή και την κβάντωση των δεδομένων. Για παράδειγμα, η ακριβής τοποθεσία βάσει GPS ενός ατόμου ενδέχεται να μην είναι χρήσιμη για μία εφαρμογή, αλλά το όνομα του δωματίου στο οποίο βρίσκεται να είναι.

Σε συστήματα επίγνωσης πλαισίου τα οποία απαρτίζονται από πολλές διαφορετικές πηγές δεδομένων, τα μεμονωμένα στοιχεία πλαισίου μπορούν να συνδυαστούν, σε αυτό το επίπεδο, σε πληροφορίες υψηλότερου επιπέδου. Η διαδικασία αυτή ονομάζεται επίσης «συσσωμάτωση» (aggregation) ή «σύνθεση» (composition). Μία μοναδική τιμή από έναν αισθητήρα συχνά δεν είναι σημαντική για μία εφαρμογή, ενώ ο συνδυασμός, είτε περισσοτέρων από μία τιμές από τον ίδιο αισθητήρα, είτε μεταξύ τιμών διαφορετικών αισθητήρων, να είναι πιο πολύτιμη ή και ακριβής. Με αυτή τη λογική, ένα σύστημα είναι ικανό να καθορίσει, για παράδειγμα, εάν ο χρήστης βρίσκεται σε εσωτερικό ή εξωτερικό χώρο, αναλύοντας πολλά διαφορετικά φυσικά δεδομένα, όπως η θερμοκρασία, το φως ή ακόμα, το αν το άτομο συμμετέχει σε κάποια συνάντηση, αξιοποιώντας μία ανάλυση του περιβάλλοντος θορύβου και της τοποθεσίας. Για να γίνει εφικτή μία τέτοια ανάλυση, πρέπει να εφαρμοστούν διάφορες στατιστικές μέθοδοι και συχνά είναι απαραίτητη μία φάση εκπαίδευσης του συστήματος.

Είναι προφανές ότι αυτή η λειτουργία αφαίρεσης μπορεί επίσης να εφαρμοστεί απευθείας από την εφαρμογή. Παρόλα αυτά, για λόγους επεξεργαστικής ισχύος και αποθηκευτικού χώρου, είναι πολλές φορές συμφέρον να πραγματοποιείται στον διακομιστή πλαισίου. Η ενθυλάκωση προωθεί την επαναχρησιμοποίηση και ως εκ τούτου, απλουστεύει την ανάπτυξη των εφαρμογών πελάτη. Παράλληλα, κάνοντας αυτούς τους συλλέκτες προσβάσιμους από απόσταση με την χρήση υπηρεσιών δικτύου, μειώνεται ο φόρτος στο δίκτυο, εφόσον μία μόνο αίτηση για δεδομένα είναι απαραίτητη για να υπάρξει πρόσβαση σε δεδομένα υψηλού επιπέδου, αντί να χρειάζεται ξεχωριστή σύνδεση σε κάθε έναν από του αισθητήρες ξεχωριστά, ενώ ταυτόχρονα χρειάζονται λιγότεροι πόροι συστήματος για την αποθήκευση στο σύστημα-πελάτη.

Το πρόβλημα με την εξεύρεση συγκρούσεων που μπορεί να συμβούν όταν χρησιμοποιούνται πολλαπλές πηγές δεδομένων πρέπει να επιλύεται σε αυτό το επίπεδο. Για παράδειγμα, όταν ένα σύστημα ενημερώνεται για την τοποθεσία ενός χρήστη από τις συντεταγμένες της φορητής του συσκευής και από μία κάμερα που εντοπίζει το άτομο, ίσως είναι δύσκολο να αποφασιστεί από το σύστημα ποια πληροφορία να χρησιμοποιήσει. Συχνά, αυτή η σύγκρουση προσεγγίζεται χρησιμοποιώντας διαφορετικά δεδομένα όπως χρονική σφραγίδα και κανόνες επίλυσης συγκρούσεων.

Στο τέταρτο επίπεδο, την «αποθήκευση και διαχείριση» των πληροφοριών πλαισίου, τα δεδομένα που συγκεντρώνονται οργανώνονται και προσφέρονται μέσω κάποιας δημόσιας διεπαφής προς το σύστημα-πελάτη. Τα συστήματα πελάτη μπορούν να αποκτήσουν πρόσβαση με δύο διαφορετικούς τρόπους, σύγχρονα και ασύγχρονα. Με τον σύγχρονο τρόπο, το σύστημα-πελάτης πραγματοποιεί αιτήματα προς τον διακομιστή πλαισίου για αλλαγές, μέσω απομακρυσμένων μεθόδων κλήσης. Συνεπώς, στέλνει ένα μήνυμα το οποίο ζητά κάποιο είδος προσφερόμενων δεδομένων και παύει έως ότου παραλάβει την απάντηση από τον διακομιστή. Στον ασύγχρονο τρόπο, η εργασία πραγματοποιείται μέσω ενός συστήματος εγγραφών. Κάθε σύστημα ή εφαρμογή-πελάτης εγγράφεται σε συγκεκριμένα γεγονότα τα οποία τα ενδιαφέρουν. Όταν αυτά τα γεγονότα συμβαίνουν, το σύστημα-πελάτης μπορεί απλώς να ενημερώνεται, ή ακόμα να καλείται μία μέθοδος σε αυτό. Στις περισσότερες περιπτώσεις, η ασύγχρονη προσέγγιση είναι καταλληλότερη, λόγω της ταχύτητας των αλλαγών στο σχετικό πλαίσιο. Η τεχνική των συνεχών περιοδικών αιτημάτων καταναλώνει πολλούς πόρους συστήματος, καθώς τα δεδομένα πλαισίου θα πρέπει να ζητώνται σχετικά συχνά και η εφαρμογή θα πρέπει να ελέγχει από μόνη της τις αλλαγές αυτές, χρησιμοποιώντας κάποιο είδος ιστορικού δεδομένων πλαισίου.

Το πρόγραμμα-πελάτης υπάρχει στο πέμπτο επίπεδο, το επίπεδο «εφαρμογής». Η εμφανής αντίδραση στα διαφορετικά γεγονότα και περιστατικά πλαισίου πραγματοποιούνται σε αυτό το επίπεδο. Μερικές φορές, η ανάκτηση πληροφοριών, η διαχείριση και ο συλλογισμός πλαισίου που αφορά στη συγκεκριμένη εφαρμογή ενθυλακώνεται στη μορφή αντιπροσώπων (agents), οι οποίοι επικοινωνούν με τον διακομιστή πλαισίου και δρουν ως ένα επιπλέον επίπεδο μεταξύ της προ-επεξεργασίας και της εφαρμογής (18). Ένα παράδειγμα λογικής πλαισίου στην πλευρά του συστήματος-πελάτη είναι το σύστημα απεικόνισης της συσκευής. Καθώς ο αισθητήρας

περιβάλλοντος φωτός αντιλαμβάνεται πολύ φως, ανεβάζει την φωτεινότητα της οθόνης για να ανταπεξέλθει.

Απαιτείται η ύπαρξη και η δημιουργία εφαρμογών που θα μπορούν να προσαρμόζουν την συμπεριφορά τους ανάλογα με τα υπάρχοντα δεδομένα. Η ύπαρξη ενός κοινού μοντέλου, το οποίο ασχολείται με την αοριστία και την ανακρίβεια των δεδομένων, διευκολύνει τούς προγραμματιστές στη δημιουργία νέων υπηρεσιών και εφαρμογών σε τέτοια περιβάλλοντα καθώς και στην επαναχρησιμοποίηση αυτών των μεθόδων συλλογισμού και εξαγωγής συμπερασμάτων πάνω στην αβεβαιότητα (62).

Τα δεδομένα πλαισίου που προέρχονται από τους προμηθευτές πλαισίου είναι χαμηλού εννοιολογικού επιπέδου. Πρόκειται για αριθμητικές τιμές και τεχνικές ή ειδικές ακατέργαστες πληροφορίες, οι οποίες από μόνες τους δεν καθίστανται χρήσιμες. Για το λόγο αυτό ένα σύστημα επίγνωσης πλαισίου πρέπει να παρέχει μηχανισμούς ερμηνείας του πλαισίου, μετατρέποντάς το σε μορφή που να είναι αξιοποιήσιμη από την εφαρμογή. Ακόμα και έτσι, όμως, υπάρχουν περιπτώσεις κατά τις οποίες και η κατεργασμένη πληροφορία ενός προμηθευτή δεν είναι ιδιαίτερα χρήσιμη από μόνη της. Σε αυτές τις περιπτώσεις, πρέπει το σύστημα να προχωρήσει σε συλλογή πληροφοριών, των οποίων ο συνδυασμός θα οδηγεί σε ένα συμπέρασμα, χρήσιμο για το χρήστη. Το ζητούμενο, δηλαδή, είναι ο συνδυασμός του χαμηλού εννοιολογικού επιπέδου πλαισίου και η εξαγωγή υψηλότερου επιπέδου σημασιολογικής πληροφορίας. Τις περισσότερες φορές το πλαίσιο υψηλού εννοιολογικού επιπέδου δεν μπορεί να αποκτηθεί άμεσα, αλλά συμπεραίνεται, συνδυάζοντας την πληροφορία που παρέχουν οι προμηθευτές, μέσω κάποιου συλλογισμού πλαισίου, μίας διαδικασίας, δηλαδή, εξαγωγής συμπερασμάτων. Η διαδικασία συλλογισμού πλαισίου προϋποθέτει την ύπαρξη ενός ευφυούς και έμπειρου συστήματος.

Εκτός από την εκμετάλλευση των πληροφοριών πλαισίου, είτε πρόκειται για δεδομένα, είτε για εξαγόμενα συμπεράσματα απευθείας από τον χρήστη, οι πληροφορίες αυτές μπορούν να χρησιμεύσουν και για την αυτόματη προσαρμογή της συμπεριφοράς του περιβάλλοντος των εφαρμογών στο πλαίσιο στο οποίο αυτές βρίσκονται. Αντί να παρέχει μια ενιαία υπηρεσία, ανεξάρτητα από τις συνθήκες του χρήστη, το σύστημα επίγνωσης πλαισίου μπορεί να προσαρμοστεί στην τρέχουσα κατάσταση. Για παράδειγμα, η προσαρμογή της συμπεριφοράς μιας συσκευής στις ανάγκες και τις ιδιομορφίες ενός συγκεκριμένου χρήστη, όπου ως πλαίσιο είναι η

τοποθεσία του, όπως αυτή προκύπτει από τον δέκτη GPS, το συνδεδεμένο δίκτυο WiFi, οι πληροφορίες κυψέλης κινητής τηλεφωνίας, οι αισθητήρες περιβάλλοντος φωτός, το μικρόφωνο, η κάμερα κ.α., είναι εφικτή δυναμώνοντας την φωτεινότητα της οθόνης κατά τη διάρκεια της μέρας, ή αλλάζοντας σε φθηνότερο επικοινωνιακό πάροχο, όταν αυτό είναι δυνατό.

2.4.3. Διαχείριση Πλαισίου

Ένα σημαντικό μέρος του συλλογισμού πλαισίου για την εξαγωγή συμπερασμάτων αποτελεί η δυνατότητα του συστήματος να χρησιμοποιήσει την εμπειρία που έχει συλλέξει από τη χρήση του συστήματος ή το πλαίσιο του χρήστη και της συσκευής, ώστε να συμπεράνει με ακρίβεια το παρόν πλαίσιο. Αυτό κάνει την αποθήκευση των δεδομένων πλαισίου, είτε πρωτογενών, είτε εξαγόμενων, απαραίτητη, ώστε να παρέχονται υπηρεσίες μελλοντικής ανάκλησης, επεξεργασίας και συμψηφισμού αυτών των δεδομένων. Μάλιστα, είναι σημαντικό, λόγω του μεγάλου όγκου δεδομένων που ενδέχεται να προέρχεται από κάποιους αισθητήρες, όπως για παράδειγμα του δέκτη GPS, ο οποίος παράγει ένα αντικείμενο τοποθεσίας ανά δευτερόλεπτο, να φιλτράρονται ή να συμψηφίζονται δεδομένα που παρέχουν την ίδια ή παρόμοια πληροφορία. Στο παράδειγμα του δέκτη GPS, όταν ο χρήστης δεν έχει μετακινηθεί από μία τοποθεσία για μία ώρα, είναι ασύμφορο να αποθηκεύονται 3600 καταχωρήσεις τοποθεσίας με τις ίδιες ή πανομοιότυπες συντεταγμένες. Αν' αυτού, μπορεί να πραγματοποιείται δειγματοληψία και συμψηφισμός, όπου το σύστημα καταχωρεί μία μέτρηση κάθε μερικά λεπτά και στην περίπτωση που είναι παρόμοια με την προηγούμενη, να αντικαθιστά την χρονική σφραγίδα με ένα εύρος μεταξύ της αρχικής και της πιο πρόσφατης χρονικής σφραγίδας που ο χρήστης δεν έχει μετακινηθεί. Επιπλέον, εξαιτίας του ότι υπάρχει πληθώρα διαφορετικών δεδομένων που συλλέγονται από το σύστημα διαχείρισης πλαισίου, θα πρέπει να υπάρχουν διαφορετικοί τρόποι αναπαράστασης της πληροφορίας πλαισίου, ανάλογα με το είδος της.

Είτε η πληροφορία προέρχεται απευθείας από τους προμηθευτές, είτε είναι απόρροια συνδυασμού δεδομένων και συλλογισμού πλαισίου, ο τρόπος με τον οποίον αυτή θα διαχειριστεί από το σύστημα ή την συσκευή είναι πολύ σημαντικός τόσο για την άμεση αξιοποίησή της, όσο και για την δυνατότητα μελλοντικής ανάκτησης και

επαναχρησιμοποίησής της. Η διαχείριση πληροφοριών πλαισίου απαιτεί ένα μοντέλο που ορίζει, φιλτράρει και αποθηκεύει τα δεδομένα πλαισίου, ενώ είναι επαρκώς γενικό, περιλαμβάνοντας όλα τα γνωρίσματα της πληροφορίας και παρέχοντας μία ευέλικτη μέθοδο αναπαράστασης. Γενικότερα, για τη διαχείριση του πλαισίου, ο εκάστοτε προγραμματιστής της εφαρμογής μπορεί να επιλέξει την τεχνική που είναι πιο εύκολη και πιο χρήσιμη να εφαρμοστεί, ανάλογα με την περίπτωση. Δύο κύριοι τρόποι διαχείρισης του συλλεγμένου πλαισίου είναι:

Σύνδεση των οδηγών των αισθητήρων απευθείας στην εφαρμογή: Σε ορισμένες εφαρμογές, που ακολουθούν αυτόν τον τρόπο διαχείρισης, οι οδηγοί των αισθητήρων είναι απευθείας συνδεδεμένοι με την εφαρμογή. Σε αυτή την περίπτωση οι προγραμματιστές των εφαρμογών πρέπει να γράψουν κώδικα ικανό να χειρίζεται τις λειτουργίες των αισθητήρων, χρησιμοποιώντας οποιοδήποτε πρωτόκολλο αυτοί τους υπαγορεύουν. Δύο προβλήματα εντοπίζονται σε αυτή την τεχνική. Το πρώτο είναι πως καθίσταται ιδιαίτερα δύσκολη η ανάπτυξη μιας εφαρμογής επίγνωσης πλαισίου, καθώς απαιτείται από τους προγραμματιστές να αντιμετωπίσουν την πιθανή πολυπλοκότητα της διαδικασίας απόκτησης των δεδομένων πλαισίου. Το δεύτερο πρόβλημα είναι πως αυτή την τεχνική δεν υποστηρίζει καλές και εύχρηστες τεχνολογίες λογισμικού. Δεν επιβάλει διαχωρισμούς ανάμεσα στην σημασιολογία της εφαρμογής και τις χαμηλού επιπέδου λειτουργίες της απόκτησης των δεδομένων πλαισίου από τους αισθητήρες. Το γεγονός αυτό, οδηγεί στην απώλεια της γενικότητας καθιστώντας αδύνατες, την επαναχρησιμοποίηση των αισθητήρων από άλλες εφαρμογές, καθώς και την ταυτόχρονη χρησιμοποίηση τους από πολλαπλές εφαρμογές.

Χρησιμοποίηση διακομιστών και απόκρυψη επιμέρους λειτουργιών των αισθητήρων: Σε αυτή την περίπτωση ένας διακομιστής έχει σχεδιαστεί έτσι ώστε να λαμβάνει τα δεδομένα με δειγματοληψία από τους αισθητήρες. Οι διακομιστές αυτοί διαχωρίζουν τις λειτουργίες των αισθητήρων από την εφαρμογή. Αυτή η τεχνική δίνει λύση και στα δύο προαναφερθέντα προβλήματα. Πλέον, οι προγραμματιστές δεν είναι αναγκασμένοι να απασχολούνται με τις ιδιαιτερότητες των λειτουργιών του κάθε αισθητήρα. Η χρήση των διακομιστών διαχωρίζει την σημασιολογία της εφαρμογής από τις χαμηλού επιπέδου λειτουργίες του αισθητήρα, κάνοντας ευκολότερη στον σχεδιαστή της εφαρμογής, την ανάπτυξη μιας εφαρμογή επίγνωσης πλαισίου και επιτρέποντας σε πολλαπλές εφαρμογές την χρησιμοποίηση ενός και μόνο διακομιστή πλαισίου.

Παρόλα αυτά αυτή η τεχνική εισάγει δύο καινούρια προβλήματα. Πρώτον, οι εφαρμογές που χρησιμοποιούν αυτούς τους διακομιστές πρέπει να είναι ενεργητικές, ζητώντας πληροφορία πλαισίου όποτε χρειάζεται, μέσω ενός μηχανισμού δειγματοληψίας. Το βάρος πέφτει πλέον στην εφαρμογή να αποφασίσει πότε υπάρχουν αλλαγές στο πλαίσιο και πότε αυτές οι αλλαγές παρουσιάζουν ενδιαφέρον. Δεύτερον, αυτοί οι διακομιστές αναπτύσσονται ανεξάρτητα για κάθε αισθητήρα ή τύπο αισθητήρα. Κάθε διακομιστής διατηρεί μια διαφορετική διεπαφή για κάθε εφαρμογή ώστε να αλληλοεπιδρά με αυτή. Αυτό απαιτεί από την εφαρμογή να αντιμετωπίζει κάθε διακομιστή με διαφορετικό τρόπο, αρκετά όμοια με το να αντιμετωπίζει διαφορετικούς αισθητήρες. Το γεγονός αυτό μπορεί να επηρεάσει την ικανότητα της εφαρμογής να ξεχωρίζει την σημασιολογία της εφαρμογής από την απόκτηση των πληροφοριών πλαισίου.

Ιδανικά, θα ήταν επιθυμητό η διαχείριση του πλαισίου να πραγματοποιείται με τον ίδιο τρόπο που γίνεται η διαχείριση της εισόδου δεδομένων από έναν χρήστη, μέσω των εργαλείων των διεπαφών του χρήστη, με την υποστήριξη των οποίων οι σχεδιαστές των εφαρμογών διαχειρίζονται τα δεδομένα που εισάγει ένας χρήστης. Αυτό θα πρέπει να συμβαίνει ταυτόχρονα με ένα σημαντικό επίπεδο αοριστίας, ώστε να επιτρέπεται στους προγραμματιστές να μην απασχολούνται με τον τρόπο με τον οποίο αποκτήθηκαν τα δεδομένα πλαισίου. Η αοριστία αυτή ονομάζεται αφαιρετικότητα μικροεφαρμογής (widget abstraction) ή διαδράστης (interactor) (23). Η αφαιρετικότητα της μικροεφαρμογής προσφέρει πολλά πλεονεκτήματα τόσο στην είσοδο από πληκτρολόγιο ή ποντίκι, όσο και στην είσοδο από στυλό ή ομιλία, καθώς και με τις αντισυμβατικές συσκευές εισόδου που χρησιμοποιούνται στην εικονική πραγματικότητα. Με αυτόν τον τρόπο διευκολύνεται ο διαχωρισμός της σημασιολογίας της εφαρμογής από τις χαμηλού επιπέδου λειτουργίες της διαχείρισης της πληροφορίας. Για παράδειγμα, μία εφαρμογή δεν χρειάζεται να διαφοροποιεί την λειτουργία της αν χρησιμοποιούμε αντί για το ποντίκι, ένα στυλό - laser για να δείχνουμε. Ακόμα, πρέπει να υποστηρίζεται η επαναχρησιμοποίηση της ίδιας μικροεφαρμογής, επιτρέποντας σε πολλαπλές εφαρμογές να δημιουργούν τα δικά τους στιγμιότυπα. Επίσης, πρέπει να περιέχεται όχι μόνο ένας μηχανισμός δειγματοληψίας αλλά και ένας μηχανισμός ειδοποίησης, ώστε οι εφαρμογές να μπορούν να αποκτούν την πληροφορία εισόδου ενώ αυτή εμφανίζεται. Τέλος, σε ένα δοσμένο εργαλείο, όλες οι μικροεφαρμογές έχουν κοινή εξωτερική διεπαφή. Αυτό σημαίνει πως η εφαρμογή

μπορεί να συμπεριφέρεται σε όλες τις μικροεφαρμογές με παρόμοιο τρόπο, μην έχοντας να αντιμετωπίσει τις πιθανές διαφορές που θα υπάρχουν ανάμεσα στο καθένα ξεχωριστά.

Σε προηγούμενα συστήματα έχουν αναπτυχθεί τεχνολογίες οι οποίες διαχειρίζονταν τις πληροφορίες πλαισίου, όπως και τα οποιοδήποτε δεδομένα εισόδου (70),(48), χρησιμοποιώντας διακομιστές για την υποστήριξη, τόσο των μηχανισμών δειγματοληψίας όσο και των αντίστοιχων μηχανισμών ειδοποίησης. Ειδικότερα, η λειτουργία ειδοποίησης απάλλασσε την εφαρμογή από την υποχρέωση να λαμβάνει δείγματα, μέσω του διακομιστή, ώστε να αποφασίζει πότε έχει πραγματοποιηθεί μια σημαντική αλλαγή των δεδομένων πλαισίου. Παρόλα αυτά, τα συστήματα αυτά αντιμετώπιζαν πρόβλημα ως προς τον σχεδιασμό εξειδικευμένων διακομιστών, λόγω της έλλειψης μιας κοινής διεπαφής. Έτσι η κάθε εφαρμογή συνδεόταν με τον κάθε διακομιστή με διαφορετικό τρόπο. Αυτό είχε ως αποτέλεσμα να χρησιμοποιείται ένας ελάχιστος αριθμός διακομιστών. Αναλύοντας την υλοποίηση των μικροεφαρμογών για την διαχείριση του πλαισίου, αιτιολογείται το πώς οι μικροεφαρμογές επίγνωσης πλαισίου παρέχουν τα ίδια πλεονεκτήματα όπως τις αντίστοιχες μικροεφαρμογές της γραφικής διεπαφής του χρήστη. Από τα παραπάνω είναι προφανές πως η χρήση της αφαίρεσης που προσφέρουν οι μικροεφαρμογές αποτελεί ένα βήμα μπροστά ως προς την αξιοποίηση των πληροφοριών πλαισίου από τις εφαρμογές. Παρά ταύτα, υπάρχουν ακόμα διαφορές στον τρόπο με τον οποίο οι πληροφορίες πλαισίου συλλέγονται και επεξεργάζονται, διαφορές οι οποίες χρήζουν την δημιουργία ενός ενδιάμεσου λογισμικού υποστήριξης της αρχιτεκτονικής των μικροεφαρμογών.

2.5.Κριτήρια συμβιβασμού

Επιλέγοντας την κατάλληλη αρχιτεκτονική για τη διαχείριση δεδομένων πλαισίου, είναι χρήσιμο να ληφθούν υπόψη διάφορα κριτήρια, τα οποία αποτελούν συμβιβασμούς πάνω σε περιορισμούς που τίθενται από τις εκάστοτε συνθήκες εκτέλεσης των αλγορίθμων διαχείρισης των δεδομένων πλαισίου (81)

- **Αποδοτικότητα**: Όλη η τεχνολογία υπολογιστικών συστημάτων υπόκειται σε περιορισμούς στον χώρο και τον χρόνο. Για διαδραστικές εφαρμογές, ο περιορισμός κλειδί είναι η αποδοτικότητα στον χρόνο, και λόγω εύρους ζώνης (bandwidth), αλλά και λόγω λανθάνουσας περιόδου (latency). Μερικές αρχιτεκτονικές επιτρέπουν την δημιουργία ταχέων

οδών για τη διαβίβαση δεδομένων, ενώ άλλες προσθέτουν επίπεδα στην δομή των επικοινωνιών, τα οποία περιορίζουν τις βελτιστοποιήσεις που μπορούν να πραγματοποιηθούν. Με δεδομένες τις σημερινές ταχύτητες των δικτύων και της επεξεργασίας, η αποδοτικότητα σε πολλές περιπτώσεις δεν περιορίζεται. Για παράδειγμα, μία εφαρμογή που χρησιμοποιεί πληροφορίες για το ποιος βρίσκεται σε έναν φυσικό χώρο, χρειάζεται μόνο μερικά byte δεδομένων και μπορεί να ανεχτεί καθυστερήσεις οι οποίες μπορεί και να μετρούνται σε δευτερόλεπτα.

- **Δυνατότητα παραμετροποίησης**: Ένα πιο δύσκολο κριτήριο για να μετρηθεί και στο οποίο δεν ισχύει ο νόμος του Moore, είναι η δυσκολία με την οποία παραμετροποιούνται συστήματα τα οποία συμπεριλαμβάνουν πολλές διεργασίες και συσκευές. Συχνά, αυτό αποτελεί και την Αχίλλειο πτέρνα αρχιτεκτονικών περίπλοκων συστημάτων. Από τη στιγμή που θα έχει πραγματοποιηθεί η παραμετροποίηση, τα μέρη του συστήματος λειτουργούν αποτελεσματικά, όμως η διαδικασία πρόσθεσης ή τροποποίησης των ήδη υπαρχόντων είναι περίπλοκη και επιρρεπής σε καταρρεύσεις. Σε πολλές περιπτώσεις, αλλαγές των παραμέτρων απαιτούν πλήρη επανεκκίνηση του συστήματος και δεν μπορούν να λειτουργήσουν επιτόπου. Καθώς τα πληροφοριακά συστήματα έχουν μετακινηθεί προς ένα δίκτυο-κεντρικό περιβάλλον με τη δυναμική προσθήκη και αφαίρεση διεργασιών, η δυνατότητα παραμετροποίησης έχει γίνει σημαντική αιτία ανησυχίας.
- **Ευρωστία**: Ένα σχετικό ζήτημα με τη δυσκολία παραμετροποίησης είναι η δυσκολία διαχείρισης της αστοχίας των μερών του συστήματος. Οι παραδοσιακές προγραμματιστικές μέθοδοι παρέχουν μηχανισμούς ελέγχου σφαλμάτων, αλλά σε γενικές γραμμές αυτοί διαχειρίζονται τα «αναμενόμενα σφάλματα» και δε διαχειρίζονται ολικές αστοχίες και αποσυνδέσεις των συστημάτων με «χάρη». Απλοί μηχανισμοί ελέγχου σφαλμάτων αρκούν για συστήματα στα οποία μία διεργασία διαχειρίζεται ολόκληρο το σύνολο των ελεγκτών του συστήματος, όπως συμβαίνει με τα λειτουργικά συστήματα κλασσικών σταθμών εργασίας, όμως δεν αναβαθμίζονται σε συστήματα με ανεξάρτητα κατανεμημένα συστήματα σε δίκτυο. Ένα εύρωστο σύστημα πρέπει να εξακολουθεί να λειτουργεί

στην περίπτωση που ένα από τα μέρη του δυσλειτουργεί, κολλά, επιστρέφει λάθος δεδομένα εξόδου, εξαφανίζεται ή επανεκκινείται. Δεν υπάρχει κάποια οικουμενική λύση στο θέμα, αλλά η επιλογή της αρχιτεκτονικής μπορεί να επηρεάσει και αρνητικά, αλλά και θετικά αυτό τον στόχο.

- **Απλότητα**: Εν κατακλείδι, η δυσκολία κλειδί είναι το ανθρώπινο μυαλό. Ένα σύστημα το οποίο απαιτεί μια περίπλοκη κατανόηση από τους κατασκευαστές που επιθυμούν να αξιοποιήσουν τις υπηρεσίες του, θα χρησιμοποιηθεί μόνο από αυτούς που έχουν την προσήλωση και το κίνητρο να το κατακτήσουν. Ο Παγκόσμιος Ιστός είναι ένα χαρακτηριστικό παράδειγμα αυτού. Τα πρωτόκολλα HTML και HTTP είναι σαφώς λιγότερο ισχυρά από πολλά πρωτόκολλα μορφοποίησης, υπερκειμένου και επικοινωνιών που προηγήθηκαν. Παρόλα αυτά, η απλότητά τους επέτρεψε τη δημιουργία ενός άλλου είδους προγραμματιστή και μίας διαφορετικής περιοχής χρήστης. Με άλλα λόγια, η απλότητα ήταν και το κλειδί της επιτυχίας του Παγκόσμιου Ιστού.

2.6.Ενδιάμεσο λογισμικό

Σύμφωνα με τους Orfali, Harkey και Edwards(53):

Το ενδιάμεσο λογισμικό είναι ένας ευρύς όρος που καλύπτει όλα τα κατανεμημένα λογισμικά που χρειάζονται για την υποστήριξη της συνεργασίας πελατών και διακομιστών....Πού αρχίζει το ενδιάμεσο λογισμικό και πού τελειώνει; Αρχίζει με την ομάδα API στην πλευρά του πελάτη που χρησιμοποιείται για την κλήση μιας υπηρεσίας, και καλύπτει την μεταβίβαση του αιτήματος στο δίκτυο και την αντίστοιχη απάντηση.

Σε ένα διαμοιρασμένο υπολογιστικό σύστημα το ενδιάμεσο λογισμικό ορίζεται ως το στρώμα λογισμικού που βρίσκεται ανάμεσα στο λειτουργικό σύστημα και τις εφαρμογές σε κάθε τοποθεσία του συστήματος(43). Με την βοήθεια του ενδιάμεσου λογισμικού παρέχεται μια κοινή προγραμματιστική αοριστία στα διεσπαρμένα συστήματα. Για να το καταφέρει αυτό πρέπει να παρέχει υψηλότερα επίπεδα προγραμματισμού μέσω APIs, σε σχέση με τα «sockets» που παρέχονται από το λειτουργικό σύστημα. Αυτό μειώνει δραστικά το βάρος από τους προγραμματιστές των εφαρμογών αφού τους απαλάσσει από τον κουραστικό και επιρρεπή σε λάθη

προγραμματισμό. Όπως χαρακτηριστικά αναφέρει ο David E. Bakken (7) ο κλασικός ορισμός ενός λειτουργικού συστήματος είναι "*το λογισμικό που κάνει το υλικό χρησιμοποιήσιμο"*. Με βάση τον παραπάνω ορισμό προχωράει και ορίζει το ενδιάμεσο λογισμικό ως «το λογισμικό που κάνει ένα διεσπαρμένο σύστημα προγραμματίσιμο».

Επηρεαζόμενες από την ανάπτυξη των δικτυακών εφαρμογών, οι τεχνολογίες ενδιάμεσου λογισμικού γίνονται όλο και πιο σημαντικές. Χρήστες αλληλοεπιδρούν μέσα από ποικιλία συσκευών, τα χαρακτηριστικά και οι δυνατότητες των οποίων εκτείνονται σε ένα μεγάλο εύρος. Ανάμεσα σε έναν υπολογιστή υψηλής απόδοσης, ένα έξυπνο κινητό τηλέφωνο και μία ταμπλέτα, οι διαφορές στο εύρος ζώνης (bandwidth), στην τοπική επεξεργαστική ισχύ (local processing power) και στην ευκρίνεια της οθόνης (screen capacity) είναι εξαιρετικά μεγάλες. Για το λόγο αυτό, οι διάφορες τεχνολογίες ενδιάμεσου λογισμικού καλούνται να καλύψουν ένα μεγάλο εύρος λειτουργικών συστημάτων, το οποίο περιλαμβάνει από λογισμικά Η/Υ και τηλεοράσεων μέχρι τα αντίστοιχα κινητών τηλεφώνων και αυτοκινήτων καθώς και να διευκολύνουν την μεταξύ τους επικοινωνία σε περιπτώσεις δικτύων.

Το ενδιάμεσο λογισμικό μπορεί να διακριθεί στις εξής κατηγορίες (36):

- **Βιβλιοθήκες**: Πρόκειται για ένα γενικευμένο σύνολο από υλοποιημένους αλγορίθμους, σχετικούς με κάποιο θέμα. Οι βιβλιοθήκες στοχεύουν αποκλειστικά στην επαναχρησιμοποίηση κώδικα.
- **Πλαίσια**: Τα πλαίσια στοχεύουν κυρίως στην επαναχρησιμοποίηση αρχιτεκτονικού σχεδιασμού. Παρέχουν μια βασική αρχιτεκτονική δομή, για την ανάπτυξη συγκεκριμένης κατηγορίας εφαρμογών. Επιπλέον ένα πλαίσιο παρέχει τρόπους για προσαρμογή μίας εφαρμογής σύμφωνα με τις ανάγκες ή τις προτιμήσεις του σχεδιαστή.
- **Εργαλεία**: Αυτά υλοποιούνται πάνω σε κάποιο πλαίσιο παρέχοντας, ένα σύνολο από επαναχρησιμοποιήσιμα συστατικά, τα οποία προσθέτουν επιπλέον λειτουργικότητα.
- **Υποδομές**: Αφορούν σε ένα σύνολο από αξιόπιστες και προσβάσιμες τεχνολογίες που λειτουργούν ως βάση για την ανάπτυξη άλλων συστημάτων. Οι υποδομές παρέχουν σχήματα, πρωτόκολλα και άλλα πρότυπα, τα οποία μπορεί να χρησιμοποιήσει ένα σύστημα που θα χτιστεί πάνω στην υποδομή. Χαρακτηριστικά παραδείγματα υποδομών

αποτελούν οι πλατφόρμες ενδιάμεσου λογισμικού J2EE, CORBA και DCOM.

Υπάρχουν δύο κατηγορίες ενδιάμεσου λογισμικού(58): το γενικό ενδιάμεσο λογισμικό και το ειδικό ενδιάμεσο λογισμικό. Το γενικό ενδιάμεσο λογισμικό είναι λογισμικό που σχετίζεται με υπηρεσίες που ζητούνται από όλους τους πελάτες και τους διακομιστές. Κανονικά περισσότερο από αυτό το ενδιάμεσο λογισμικό αφορά τα λειτουργικά συστήματα. Το τυπικό λογισμικό που υπάγεται σ' αυτό το πλαίσιο περιλαμβάνει:

- Λογισμικό για την εκτέλεση διεργασιών όπως η μεταφορά ακατέργαστων δεδομένων χαρακτήρων στο Διαδίκτυο.
- Λογισμικό που συγχρονίζει πανομοιότυπα αντίγραφα αρχείων.
- Λογισμικό που διαχειρίζεται μία κατανεμημένη συλλογή αρχείων.

Ο ρόλος του ενδιάμεσου λογισμικού είναι να διευκολύνει τον σχεδιασμό εφαρμογών, παρέχοντας κοινές προγραμματιστικές αοριστίες, καλύπτοντας την ετερογένεια και την διασπορά του υλικού και των λειτουργικών συστημάτων και κρύβοντας τις προγραμματιστικές λεπτομέρειες χαμηλού επιπέδου.

2.6.1. Είδη ενδιάμεσου λογισμικού

Το ενδιάμεσο λογισμικό που έχει αναπτυχθεί μπορεί να χωρισθεί σε ορισμένες κατηγορίες που διαφέρουν ανάλογα με την προγραμματιστική αοριστία που παρέχει και τα είδη ετερογένειας που είναι διαθέσιμα πίσω από το δίκτυο και το υλικό (7).

Διεσπαρμένες πλειάδες: Η διεσπαρμένη σχεσιακή βάση δεδομένων προσφέρει την αοριστία των διεσπαρμένων πλειάδων και είναι η πλέον διαδεδομένη μορφή ενδιάμεσου λογισμικού. Η γλώσσα βάσης δεδομένων που χρησιμοποιεί (SQL) επιτρέπει στους προγραμματιστές να χειριστούν αυτά τα σύνολα πλειάδων (μία βάση δεδομένων) σε μία γλώσσα παρόμοια της αγγλικής όμως με μια διαισθητική σημασιολογία και μία αυστηρή μαθηματική θεμελίωση, βασισμένη στην θεωρία συνόλων και τον κατηγορηματικό λογισμό. Ακόμα, προσφέρουν συνήθως ετερογένεια ανάμεσα στις γλώσσες προγραμματισμού, αν και τις περισσότερες φορές προσφέρεται από ελάχιστη έως καθόλου ετερογένεια σε εμπορικές εφαρμογές. Επίσης, οι

διεσπαρμένες σχεσιακές βάσεις δεδομένων προσφέρουν την αοριστία της συναλλαγής. Οι οθόνες επεξεργασίας συναλλαγών, Transaction Processing Monitors (TPM), χρησιμοποιούνται συνήθως για την end-to-end διαχείριση των πόρων των ερωτημάτων των πελατών, ειδικά από την πλευρά του διακομιστή, τη διαχείριση της διαδικασίας και τη διαχείριση συναλλαγών πολλαπλών δεδομένων.

Για παράδειγμα, το Linda είναι ένα πλαίσιο (framework) που παρέχει την αοριστία των διεσπαρμένων πλειάδων και ονομάζεται Tuple Space (TS). Τα API's του Linda παρέχουν σχεσιακή πρόσβαση στο TS, χωρίς όμως καμία σχεσιακή σημασιολογία. Ακόμα παρέχει χωρική αποσύνδεση επιτρέποντας στις διαδικασίες κατάθεσης και απόσυρσης να μην γνωρίζει η μία της ταυτότητα της άλλης. Ακόμα προσφέρει χρονική αποσύνδεση επιτρέποντας τους να μην έχουν αλληλεπικαλυπτόμενες διάρκειες ζωής.

Το Jini είναι ένα Java framework για έξυπνες συσκευές, ιδιαίτερα οικιακές. Το Jini είναι δομημένο πάνω και σχετίζεται πολύ με το TS του Linda.

Απομακρυσμένη κλήση διαδικασιών (Remote Process Calls): Το ενδιάμεσο λογισμικό της απομακρυσμένης κλήσης διαδικασίας (RPC) επεκτείνει την διεπαφή της κλήσης μιας διαδικασίας προσφέροντας στους προγραμματιστές την αοριστία του να είναι σε θέση να επικαλεστούν μια διαδικασία το σώμα της οποίας είναι σε ένα άλλο τμήμα ενός δικτύου. Τα RPC συστήματα είναι συνήθως σύγχρονα και ως εκ τούτου δεν προσφέρουν καμία δυνατότητα παραλληλισμού χωρίς τη χρήση πολλαπλών νημάτων και έχουν συνήθως περιορισμένες παροχές χειρισμού εξαιρέσεων.

Ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος (Message-Oriented Middleware): Το ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος παρέχει την αοριστία μιας ουράς μηνυμάτων που μπορεί να γίνει προσβάσιμη μέσω ενός δικτύου. Είναι μια γενίκευση του πολύ γνωστού λειτουργικού συστήματος: του γραμματοκιβωτίου.

Είναι πολύ ευέλικτο για το πώς μπορεί να ρυθμιστεί με την τοπολογία των προγραμμάτων που καταθέτουν και αποσύρουν τα μηνύματα από μια συγκεκριμένη ούρα. Πολλά προϊόντα (MOM) προσφέρουν ουρές με επιμονή, αναπαραγωγή, ή εκτέλεση σε πραγματικό χρόνο. Το ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος προσφέρει το ίδιο είδος χωρικής και χρονικής αποσύνδεσης που προσφέρει και το Linda.

Ενδιάμεσο λογισμικό διεσπαρμένου αντικειμένου (DOM): Το ενδιάμεσο λογισμικό διεσπαρμένου αντικειμένου παρέχει την αοριστία ενός αντικειμένου που είναι απομακρυσμένο, στις μεθόδους όμως του οποίου μπορεί να γίνει επίκληση ακριβώς όπως θα γινόταν σε ένα αντικείμενο εάν βρισκόταν στον ίδιο χώρο διεύθυνσης του καλούντος.

Τα διεσπαρμένα αντικείμενα διαθέτουν στον προγραμματιστή της εφαρμογής όλα τα πλεονεκτήματα της τεχνολογίας λογισμικού των τεχνικών προσανατολισμένου αντικειμένου, όπως ενθυλάκωση, κληρονομικότητα και πολυμορφισμός.

2.6.1. Γενικές Απαιτήσεις Αρχιτεκτονικής Ενδιάμεσου Λογισμικού

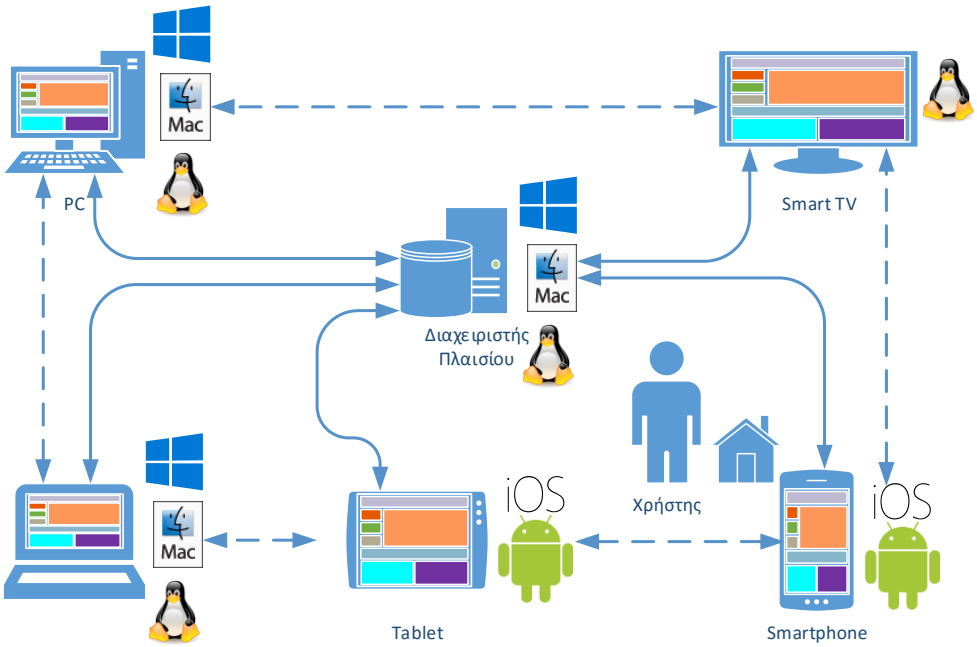
Σύμφωνα με τις προδιαγραφές, όπως έχουν οριστεί για το παρόν μοντέλο, το ενδιάμεσο λογισμικό πάνω στο οποίο μπορεί να υλοποιηθεί μία πλατφόρμα Επίγνωσης Πλαισίου οφείλει να ικανοποιεί κάποιες γενικές απαιτήσεις αρχιτεκτονικής:

- Το ενδιάμεσο λογισμικό εφαρμογών θα πρέπει να είναι ανεξάρτητο πλατφόρμας, δηλαδή να υποστηρίζονται τα περισσότερα, αν όχι όλα, τα διαδεδομένα λειτουργικά συστήματα και συσκευές, όπως συστήματα που εκτελούν Windows, MacOS, Linux, Android και Linux, καθώς και τις περισσότερες ευρέως διαδεδομένες κατηγορίες συσκευών που τα χρησιμοποιούν, όπως επιτραπέζιους υπολογιστές, φορητούς υπολογιστές, έξυπνα κινητά τηλέφωνα, ταμπλέτες και έξυπνες τηλεοράσεις.
- Οι συσκευές που ανήκουν στον ίδιο χρήστη θα πρέπει να διασυνδέονται μέσω αυτού του ενδιάμεσου λογισμικού και σε επίπεδο πλατφόρμας, αλλά και σε επίπεδο εφαρμογών. Αυτό σημαίνει ότι θα πρέπει να υπάρχει επίγνωση από το ενδιάμεσο λογισμικό και του συνόλου των υποστηριζόμενων συσκευών του χρήστη, αλλά και των συμβατών εφαρμογών που εκτελούνται σε αυτές
- Η πλατφόρμα ενδιάμεσου λογισμικού θα πρέπει να παρέχει ένα ενιαίο οικοσύστημα εφαρμογών και υπηρεσιών το οποίο είναι μοναδικό για

κάθε μοναδικό χρήστη των συμβατών συσκευών και εφαρμογών που είναι κατανεμημένες σε αυτές.

- Τα δεδομένα, ειδικότερα αυτά που αφορούν στο πλαίσιο, θα πρέπει να αποθηκεύονται σε μία κοινή, για το οικοσύστημα του κάθε χρήστη, Βάση Δεδομένων, η οποία είναι προσβάσιμη από το σύνολο των συμβατών συσκευών που ανήκουν στον χρήστη.
- Οι ρυθμίσεις ασφάλειας και ιδιωτικότητας της πλατφόρμας, των εφαρμογών που εκτελούνται σε αυτή, καθώς και των δεδομένων και υπηρεσιών στα οποία δύναται να παρέχεται πρόσβαση θα πρέπει να παρέχεται τοπικά για κάθε συσκευή, αλλά και κεντρικά για το οικοσύστημα χρήστη, μέσα από υπηρεσίες που παρέχονται ειδικά, ξεχωριστά και τοπικά σε κάθε τέτοιο οικοσύστημα.
- Η υλοποίηση θα πρέπει να είναι ανοιχτού κώδικα ώστε να παρέχεται η δυνατότητα περεταίρω βελτίωσης, ανάπτυξης και προέκτασης της πλατφόρμας, ενώ ταυτόχρονα θα δίνεται η δυνατότητα ανάπτυξης εφαρμογών χωρίς τους περιορισμούς που συναντώνται σε υλοποιήσεις κλειστού κώδικα.

Η γενική αρχιτεκτονική του κάθε οικοσυστήματος χρήστη εμφανίζεται στην Εικόνα 2.

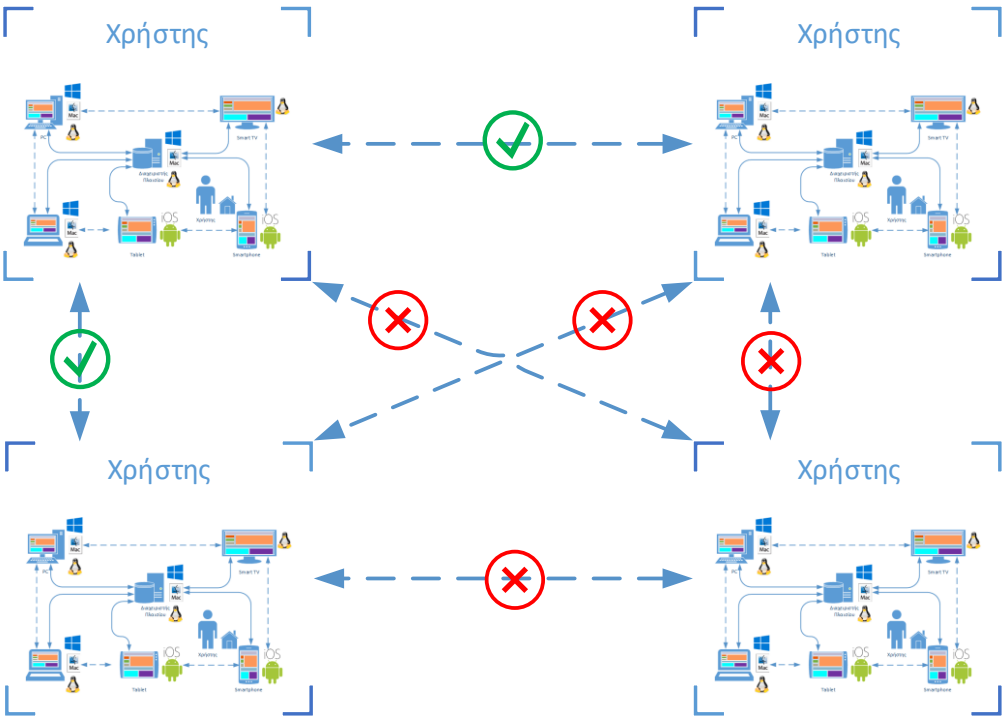


Εικόνα 2: Γενική Αρχιτεκτονική Οικοσυστήματος Χρήστη

Επιπλέον των παραπάνω, απαιτείται να υπάρχει:

- Δυνατότητα επικοινωνίας μεταξύ των εφαρμογών διαφορετικών οικοσυστημάτων χρηστών. Η επικοινωνία αυτή θα πρέπει να μπορεί να εκτελείται είτε ευθέως μεταξύ των εφαρμογών, είτε εμμέσως, με την πρόσβαση σε κοινά δεδομένα.
- Η επικοινωνία μεταξύ των οικοσυστημάτων χρηστών θα πρέπει να υλοποιείται διομότιμα (peer-to-peer), χωρίς τη συμμετοχή κεντρικών διακομιστών, ώστε να διασφαλίζεται η ιδιωτικότητα των δεδομένων που ανταλλάσσονται με τον αποτελεσματικότερο δυνατό τρόπο.
- Η πολιτική ασφαλείας και ιδιωτικότητας που αφορά στην επικοινωνία μεταξύ διαφορετικών οικοσυστημάτων χρηστών θα πρέπει να υλοποιείται σε ένα αυστηρό μοντέλο λευκής λίστας με χρονικό περιορισμό ή περιορισμό χρόνου εκτέλεσης, σε επίπεδο συγκεκριμένων εφαρμογών και ειδών δεδομένων. Η πρόσβαση σε συγκεκριμένα δεδομένα ή λειτουργίες εφαρμογών θα πρέπει να παρέχεται από ενέργεια του χρήστη από τον οποίο αιτείται η πρόσβαση, για τον χρόνο εκτέλεσης της συγκεκριμένης ενέργειας ή/και για προκαθορισμένο χρονικό διάστημα.

Η γενική αρχιτεκτονική του μοντέλου διασύνδεσης και διαλειτουργικότητας διαφορετικών οικοσυστημάτων χρηστών εμφανίζεται στην Εικόνα 3.



Εικόνα 3: Γενική Αρχιτεκτονική διασύνδεσης πολλαπλών οικοσυστημάτων χρηστών

2.6.2. Ενδιάμεσο Λογισμικό Επίγνωσης Πλαισίου

Ενώ οι εφαρμογές επίγνωσης πλαισίου έχουν αποτελέσει δημοφιλές θέμα συζήτησης και έρευνας εδώ και τουλάχιστον δεκαπέντε χρόνια, ελάχιστες υπηρεσίες είναι διαθέσιμες στους απλούς χρήστες. Το να κατασκευαστούν αυτά τα συστήματα εξακολουθεί να δείχνει ότι είναι υπερβολικά χρονοβόρα και περίπλοκη διαδικασία, λόγω της περιορισμένης υποστήριξης σε επίπεδο υποδομών. Μία τέτοια υποδομή απαιτεί τα ακόλουθα χαρακτηριστικά (32):

Ένα κοινό μοντέλο πλαισίου, το οποίο μπορεί να διαμοιραστεί μεταξύ συσκευών και υπηρεσιών. Ο διαμοιρασμός των πληροφοριών πλαισίου είναι σημαντικός, καθώς άλλες οντότητες μπορούν α αναγνωρίσουν πλαίσια στο ίδιο τομέα έξυπνου χώρου, ή και μεταξύ διαφορετικών τομέων. Σε παλιότερα συστήματα, οι πληροφορίες πλαισίου που αναπαριστώνταν ως strings ή είχαν μοντελοποιηθεί ως προγραμματιστικά αντικείμενα της Java ήταν πολύ δύσκολο να διαμοιραστούν αποτελεσματικά. Οι οντολογίες έχουν χρησιμοποιηθεί εκτενώς σε πολλές περιοχές όπως η Τεχνητή Νοημοσύνη και τον Σημασιολογικό Ιστό. Μία οντολογία παρέχει ένα λεξιλόγιο για την αναπαράσταση της γνώσης σε κάποιο τομέα και χρησιμεύει στο να περιγράψει συγκεκριμένες καταστάσεις αυτού του τομέα. Εφαρμόζοντας οντολογίες

και τεχνολογίες Σημασιολογικού Ιστού σε περιβάλλοντα διάχυτου υπολογισμού, μπορεί να χρησιμοποιηθεί ένα μοντέλο οντολογικής αναπαράστασης του πλαισίου, όπως η OWL (61), μία γλώσσα οντολογικής σήμανσης (markup), η οποία επιτρέπει το διαμοιρασμό των πληροφοριών πλαισίου, ορίζοντας συγκεκριμένα πλαίσια/αντικείμενα μέσω σημασιολογικού προσδιορισμού τους. Με αυτό τον τρόπο ο συλλογισμός πλαισίου γίνεται εφικτός. Μέσω του συλλογισμού αυτού, πολλά υψηλού επιπέδου αόριστα πλαίσια μπορούν να παραχτούν μέσω χαμηλού επιπέδου σαφή πλαίσια.

Ένα σύνολο υπηρεσιών, τα οποία εκτελούν την ανακάλυψη, την ανάκτηση, την ερμηνεία και τον διαμοιρασμό του πλαισίου. Μία πρόκληση στην δημιουργία αυτών των δομών είναι στον σχεδιασμό του κατάλληλου σύνολο υπηρεσιών που θα αναλαμβάνουν διαφορετικά καθήκοντα. Μερικά από αυτά τα καθήκοντα ενδέχεται να αναφέρονται σε συγκεκριμένες εφαρμογές. Με αυτό τον τρόπο μπορεί να σχεδιαστεί ένα ενδιάμεσο λογισμικό επίγνωσης πλαισίου, το οποίο αναλαμβάνει αυτά τα καθήκοντα. Ένα στοιχείο υπηρεσίας εκτελεί ένα συγκεκριμένο καθήκον και αλληλοεπιδρά με άλλα στοιχεία, με σκοπό να εξυπηρετήσει εφαρμογές επίγνωσης πλαισίου. Αυτή η αρχιτεκτονική επιτρέπει στο μοντέλο πλαισίου να συλλέγει πληροφορίες πλαισίου από διαφορετικούς παρόχους πλαισίου, να ερμηνεύει το πλαίσιο μέσω ενός συλλογισμού πλαισίου και να παρέχει τα παραγόμενα δεδομένα σε λειτουργίες και ώθησης (push) και έλξης (pull). Με αυτά τα στοιχεία υπηρεσιών, οι υπηρεσίες επίγνωσης πλαισίου μπορούν εύκολα να δημιουργηθούν χρησιμοποιώντας διαφορετικές υπηρεσίες για να προσφέρεται πρόσβαση σε διαφορετικά είδη πλαισίου, με διαφορετικά επίπεδα πολυπλοκότητας.

Η προσέγγιση αυτή γίνεται απαραίτητη στην περίπτωση των εφαρμογών επίγνωσης πλαισίου, διότι οι πληροφορίες πλαισίου έχουν ορισμένα ιδιαίτερα χαρακτηριστικά που τις καθιστούν δύσκολες στον χειρισμό από ένα υπολογιστικό σύστημα. Για το λόγο αυτό, η κάθε εφαρμογή που χρησιμοποιεί πληροφορία πλαισίου απαιτεί την ανάπτυξη ξεχωριστών υποσυστημάτων, ανεξάρτητων της εφαρμογής, που θα υλοποιούν τις διαδικασίες επεξεργασίας και διαχείρισης του πλαισίου.

Τα προβλήματα αυτά οδήγησαν τους σχεδιαστές εφαρμογών επίγνωσης πλαισίου σε μία διαφορετική αρχιτεκτονική προσέγγιση. Σύμφωνα με αυτή, η διαχείριση και η επεξεργασία του πλαισίου αναλαμβάνονται εξ ολοκλήρου από ένα

ενδιάμεσο λογισμικό, ανεξαρτήτως της εφαρμογής. Το ενδιάμεσο λογισμικό είναι μια κατηγορία τεχνολογίας λογισμικού σχεδιασμένο για να βοηθάει στην διαχείρηση της πολυπλοκότητας και την ετερογένεια που υπάρχει στα διεσπαρμένα συστήματα. Με τον τρόπο αυτό παρέχονται στην εφαρμογή έτοιμες υπηρεσίες και εργαλεία προσαρμογής. Έτσι, η εφαρμογή σχεδιάζεται πάνω στο ενδιάμεσο λογισμικό, δίχως ο προγραμματιστής να χρειάζεται να ασχοληθεί με την υλοποίηση λειτουργιών διαχείρισης και επεξεργασίας του πλαισίου. Τελικά, το ενδιάμεσο λογισμικό που διαχειρίζεται πλαίσιο και προσδίδει χαρακτηριστικά επίγνωσής του στην εφαρμογή ονομάζεται Context–Aware Middleware (CAM) (85).

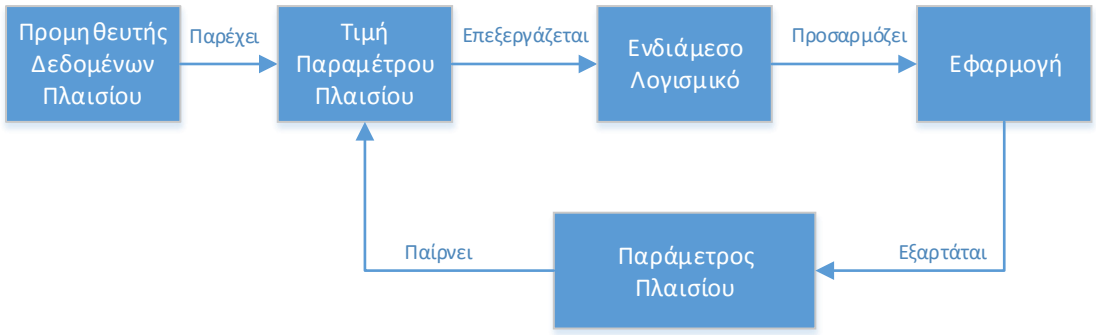
Οι παραδοσιακές πλατφόρμες ενδιάμεσου λογισμικού έλυναν προβλήματα όπως η κλιμάκωση και η ετερογένεια και παρείχαν διαφάνεια στην κατανομή και στο διαμοιρασμό των πόρων. Παρείχαν στους σχεδιαστές, ένα υψηλό επίπεδο αφαίρεσης κρύβοντας τις λεπτομέρειες της κατανομής στην ανάπτυξη του συστήματος. Σε αυτές τις τεχνολογίες το ενδιάμεσο λογισμικό κτίζονταν σαν ένα ξεχωριστό σύστημα που έπαιρνε είσοδο από την εφαρμογή και παρείχε διαφάνεια στους χρήστες και τους σχεδιαστές των εφαρμογών (15).

Οι τεχνολογίες αυτές έχουν χρησιμοποιηθεί με επιτυχία για στατικά κατανεμημένα συστήματα, σχεδιασμένα για ενσύρματα δίκτυα. Παρόλα αυτά, δεν μπορούν να αντεπεξέλθουν επαρκώς στις απαιτήσεις των κινητών εφαρμογών που μπορούν να παρέχουν υπηρεσίες οπουδήποτε, κάθε στιγμή. Αυτές οι τεχνολογίες υποθέτουν μεγάλο διαθέσιμο εύρος ζώνης, καθώς και σταθερή διαθεσιμότητα. Σε κινητά συστήματα αυτές οι υποθέσεις δεν ισχύουν. Επιπλέον, αντικειμενοστραφείς πλατφόρμες ενδιάμεσου λογισμικού υποστηρίζουν κυρίως σύγχρονη επικοινωνία, ενώ σε κινητά περιβάλλοντα είναι πολύ συνηθισμένο ο διακομιστής και ο χρήστης να μην είναι ταυτόχρονα συνδεδεμένοι. Το ενδιάμεσο λογισμικό που προορίζεται για εφαρμογές επίγνωσης πλαισίου πρέπει να είναι σχεδιασμένο με βάση τα χαρακτηριστικά των διάχυτων συστημάτων και των συσκευών που τα συνθέτουν.

Η προσέγγιση ενός ενδιάμεσου λογισμικού με στατική συμπεριφορά δεν είναι η κατάλληλη για ένα CAM (14). Το CAM πρέπει να ανιχνεύει τις αλλαγές που γίνονται στο περιβάλλον του και να προσαρμόζει τις υπηρεσίες του. Το ενδιάμεσο λογισμικό, λαμβάνοντας συγκεκριμένη πληροφορία από την εφαρμογή και το περιβάλλον, θα μπορούσε να πάρει πιο αποτελεσματικές αποφάσεις, παρέχοντας ποιοτικότερες υπηρεσίες. Επιπλέον, η εφαρμογή μπορεί να τροποποιήσει την εσωτερική δομή του

CAM σύμφωνα με τις ανάγκες της (απορρόφηση - absorption). Για παράδειγμα, η τροποποίηση μπορεί να αφορά το σύνολο των πόρων που παρακολουθούνται ή τους κανόνες προσαρμογής στο πλαίσιο. Οι υπηρεσίες διαχείρισης ενός CAM μπορεί να παρέχονται κεντρικά από κάποια συσκευή με αυξημένες υπολογιστικές δυνατότητες ή αποκεντρικοποιημένα, από το ενδιάμεσο λογισμικό, που βρίσκεται πάνω στη κινητή συσκευή. Στην πρώτη περίπτωση οι σύνθετες διαδικασίες επιτελούνται σε κάποιους στατικούς εξυπηρέτες, οι οποίοι παρέχουν υπηρεσίες διαχείρισης του πλαισίου στους κινητούς χρήστες, όπου εκτελούνται οι εφαρμογές. Στη δεύτερη περίπτωση, όλη η διαχείριση και επεξεργασία του πλαισίου γίνεται από το ενδιάμεσο λογισμικό που είναι εγκατεστημένο στην κινητή συσκευή.

Η κεντρικοποιημένη αρχιτεκτονική είναι πιο εύκολη στην υλοποίηση, καθώς το ενδιάμεσο λογισμικό δεν εξαρτάται από τους υπολογιστικούς πόρους της συσκευής για την οποία προορίζεται. Αυτή η προσέγγιση όμως, έχει περιορισμένες δυνατότητες κλιμάκωσης και δεν μπορεί να εφαρμοστεί σε διάχυτα δίκτυα μεγάλης γεωγραφικής έκτασης. Επιπλέον, δεν είναι ανεκτική σε σφάλματα καθώς εξαρτάται από ένα κεντρικό συστατικό. Τα CAMs με κεντρικοποιημένη αρχιτεκτονική σχεδιάζονται κυρίως για τη δημιουργία «Ευφυών χώρων» (intelligent spaces) ή «Ενεργών χώρων» (active spaces). Οι φυσικοί χώροι είναι περιορισμένες γεωγραφικά περιοχές, όπως δωμάτια, γραφεία, εργαστήρια, οχήματα κ.α., οι οποίοι περιλαμβάνουν φυσικά αντικείμενα, ετερογενείς διασυνδεδεμένες συσκευές και χρήστες που επιτελούν ένα σύνολο δραστηριοτήτων. Οι ευφυείς χώροι ή ενεργοί χώροι είναι φυσικοί χώροι, που περιλαμβάνουν μία υποδομή λογισμικού επίγνωσης πλασιίου που συντονίζει την αλληλεπίδραση των χρηστών με το περιβάλλον τους (63). Ένα παράδειγμα της αρχιτεκτονικής αυτής φαίνεται στο Διάγραμμα 1.



Διάγραμμα 1: Διαχείριση Πληροφορίας Πλαισίου με Ενδιάμεσο Λογισμικό

Παρά την απλούστευση που παρέχει στην ανάπτυξη εφαρμογών επίγνωσης πλαισίου, η αρχιτεκτονική αυτή δε λύνει όλα τα προβλήματα και ειδικότερα αυτά που προκύπτουν από την ετερογένεια των διεσπαρμένων συστημάτων. Αρχικά, υπάρχει ένα κενό ανάμεσα στην θεωρία και στην πράξη. Πολλές δημοφιλείς υπηρεσίες ενδιάμεσου λογισμικού χρησιμοποιούν ιδιόκτητα API's, συνήθως κάνοντας τις εφαρμογές να εξαρτώνται από έναν μόνο προϊόν προμηθευτή, και ιδιόκτητα και αδημοσίευτα πρωτόκολλα, καθιστώντας δύσκολο για τους διαφορετικούς προμηθευτές να αναπτύξουν διαλειτουργικές εφαρμογές. Για παράδειγμα, αρκετά συστήματα διαχείρισης βάσεων δεδομένων υποστηρίζουν SQL πλατφόρμες αποκλειστικής εκμετάλλευσης από περιορισμένους φορείς και αντίστοιχα πρωτόκολλα, τα οποία δεν είναι διαθέσιμα σε πολλές δημοφιλείς πλατφόρμες (όπως της Oracle, της IBM κ.α.). Ως αποτέλεσμα, περιορίζονται οι δυνατότητες ενός χρήστη να συνδέεται σε συστήματα που παρουσιάζουν ετερογένεια με το δικό του. Ακόμα και αν μία τεχνολογία ενδιάμεσου λογισμικού είναι σχετικά σύγχρονη, όταν ένας προγραμματιστής ενδιαφέρεται να αναπτύξει την εφαρμογή του, έχει να υπολογίσει το ρίσκο και την δυνατότητα του συγκεκριμένου ενδιάμεσου λογισμικού να παραμένει επικαιροποιημένο και να εξελιχθεί παράλληλα με την τεχνολογία. Για παράδειγμα, πολλές εφαρμογές που γράφτηκαν βασισμένες σε τεχνολογίες ενδιάμεσου λογισμικού που δεν εξελίχθηκαν και ξεπεράστηκαν, αναγκαστικά υλοποιήθηκαν από την αρχή, ώστε να ενσωματώνονται σε σύγχρονα συστήματα.

Δεύτερο, το μεγάλο πλήθος υπηρεσιών ενδιαμέσου λογισμικού αποτελεί από μόνο του ανασταλτικό παράγοντα για την χρησιμοποίηση τους. Αυτό καθώς ακόμα και ένας μικρός αριθμός από διαφορετικά ενδιάμεσα λογισμικά μπορεί να οδηγήσει σε υψηλή προγραμματιστική πολυπλοκότητα. Γι' αυτό το λόγο οι προγραμματιστές

πρέπει να επιλέξουν έναν πολύ μικρό αριθμό υπηρεσιών που θα ικανοποιούν τις ανάγκες τους για λειτουργικότητα κ.α.

Τρίτο, ενώ οι υπηρεσίες ενδιάμεσου λογισμικού ανεβάζουν το επίπεδο αοριστίας που απαιτεί ο προγραμματισμός διασκορπισμένων εφαρμογών, επιβαρύνουν ακόμα τον προγραμματιστή με τις δύσκολες αποφάσεις σε ότι αφορά τον σχεδιασμό.

3. Υφιστάμενη κατάσταση

Μέχρι σήμερα έχουν αναπτυχθεί αρκετά συστήματα ενδιάμεσου λογισμικού, τα οποία αφορούν σε εφαρμογές επίγνωσης πλαισίου πολλαπλών συσκευών. Στη δεκαετία του '*90*, η έρευνα στόχευε κυρίως σε εφαρμογές επίγνωσης πλαισίου για συγκεκριμένο σκοπό. Στη συνέχεια ('00) έγινε μία μετατόπιση της έρευνας, προς το πρόβλημα της σχεδίασης και ανάπτυξης ενδιάμεσου λογισμικού για την υποστήριξη εφαρμογών επίγνωσης πλαισίου. Αρχικά αναπτύχθηκε μια μεγάλη ποικιλία πλατφορμών ενδιάμεσου λογισμικού, οι οποίες δεν αφορούσαν σε διαδικτυακές εφαρμογές, οπότε και δεν είχαν σχεδιαστεί ώστε να λειτουργούν σε τέτοιο περιβάλλον. Σήμερα, η διαχείριση του πλαισίου και οι διαδικασίες που τη συνθέτουν, αποτελούν θέμα πολλών εργασιών και οι οποίες στην πλειονότητα τους είναι βασισμένες στο διαδίκτυο και σε κατανεμημένες υπηρεσίες.

3.1.Ενδιάμεσα Λογισμικά Επίγνωσης Πλαισίου

Η πρώτη σημαντική προσέγγιση προς τη κατεύθυνση αυτής της αρχιτεκτονικής, προήλθε από τον W. Schilit (80). Ο Schilit στη διδακτορική του διατριβή πρότεινε ένα γενικό πλαίσιο για εφαρμογές επίγνωσης πλαισίου, επικεντρώνοντας κυρίως σε πλαίσιο που αφορούσε στη γεωγραφική θέση του χρήστη. Ο B. Schilit (66) πρότεινε ένα πλαίσιο με κεντρικοποιημένη αρχιτεκτονική. Το πλαίσιο όριζε ένα περιβάλλον σαν ένα σύνολο από ονόματα παραμέτρων και τιμών για τις παραμέτρους. Στατικοί διακομιστές διαχειρίζονταν το περιβάλλον και μετέφεραν την πληροφορία, στους χρήστες (εφαρμογές) που είχαν ενδιαφερθεί πιο πριν, κάνοντας εγγραφή στους διακομιστές. Τυπικά, αντιστοιχούσε ένα περιβάλλον ανά χρήστη και επιπρόσθετα περιβάλλοντα για συγκεκριμένες οντότητες όπως δωμάτια, ομάδες ατόμων κ.α. Παρόμοιες προσεγγίσεις με αυτήν του B. Schilit προτάθηκαν ως πλαίσια που δρουν ως ενδιάμεσο λογισμικό που συλλέγει τις πληροφορίες πλαισίου από τους αισθητήρες και παρέχει μεταφρασμένη την πληροφορία στην εφαρμογή, μέσω ενός συγκεκριμένου API.

Η Hewlett Packard πρότεινε ένα πλαίσιο με το όνομα "An environment for situational Computing" (25). Η ιδέα βασίζεται στην ύπαρξη ενός εξυπηρετητή για την συλλογή και επεξεργασία των δεδομένων πλαισίου. Ο εξυπηρετητής παρέχει ένα σύνολο υπηρεσιών, για την μετατροπή των φυσικών δεδομένων που παρείχαν οι

αισθητήρες, σε γεγονότα πλαισίου. Οι υπηρεσίες αυτές, είναι υπεύθυνες για τη συλλογή της πληροφορίας από τους αισθητήρες και την παροχή της σε κατανοητή και χρήσιμη μορφή στην εφαρμογή. Θεμελιώδης για την έρευνα στα συστήματα επίγνωσης πλαισίου θεωρείται η εργασία του Anind K. Dey από το GeorgiaTech. Ο Dey στο πλαίσιο της διδακτορικής του διατριβής πρότεινε μια γενική αρχιτεκτονική για την υποστήριξη και ανάπτυξη εφαρμογών επίγνωσης πλαισίου. Υλοποίησε μια υποδομή βασισμένη σε αυτήν την αρχιτεκτονική και ένα εργαλείο με το όνομα Context Toolkit (23). Η αρχιτεκτονική του Context Toolkit βασίζεται σε μια αντικειμενοστραφή προσέγγιση και περιλαμβάνει τρεις τύπους αντικειμένων: widgets, servers ή aggregators και interpreters.

Το προγραμματίδιο πλαισίου (context widget) είναι μια μονάδα λογισμικού που παρέχει στην εφαρμογή πληροφορία πλαισίου από το περιβάλλον εκτέλεσής της. Ο ρόλος των προγραμματιδίων είναι να απομονώσουν την εφαρμογή από τις διαδικασίες συλλογής της πληροφορίας πλαισίου. Κρύβουν την πολυπλοκότητα των αισθητήρων, παρέχοντας αφαιρετικά την πληροφορία, ώστε να μπορεί να χρησιμοποιηθεί από την εφαρμογή.

Οι διακομιστές πλαισίου (context servers) είναι υπεύθυνοι για τη συλλογή της πληροφορίας πλαισίου που αφορά σε μία συγκεκριμένη οντότητα, όπως για παράδειγμα ένας χρήστης. Αποτελούν υποκλάσεις των προγραμματιδίων και κληρονομούν όλες τις ιδιότητες και τις μεθόδους ενός προγραμματιδίου. Ο διακομιστής πλαισίου κάνει εγγραφή στο προγραμματίδιο που τον ενδιαφέρει, αυτό δηλαδή που παρέχει την πληροφορία που σχετίζεται με κάποια οντότητα και λειτουργεί σαν ένα πληρεξούσιο (proxy) μεταξύ προγραμματιδίου και εφαρμογής. Οι διερμηνείς πλαισίου (context interpreters) μεταφράζουν τα δεδομένα πλαισίου σε κατανοητή μορφή για την εφαρμογή. Μπορούν να εκτελέσουν μετατροπές της πληροφορίας σε διάφορους τύπους αναπαράστασης και να συνδυάσουν διαφορετικούς τύπους πληροφορίας συνθέτοντας νέες πληροφορίες πλαισίου. Κάθε ένα από τα παραπάνω αντικείμενα μπορεί να εκτελεστεί αυτόνομα. Τα αντικείμενα μπορούν να δημιουργηθούν όλα σε έναν κόμβο ή σε πολλούς διαφορετικούς. Για την επικοινωνία μεταξύ των αντικειμένων χρησιμοποιείται HTTP και XML. Ωστόσο και άλλες τεχνολογίες μπορούν να χρησιμοποιηθούν. Η βασική υλοποίηση είχε γίνει σε Java, αλλά οι μηχανισμοί που χρησιμοποιούνται είναι ανεξάρτητοι από τη γλώσσα προγραμματισμού. Από τότε που παρουσιάστηκε μέχρι και σήμερα, το Context Toolkit

ενέπνευσε πολλούς ερευνητές, οι οποίοι επέκτειναν ή στήριξαν το έργο τους στις ιδέες του Dey. Οι αρχιτεκτονικές αρχές του Context Toolkit βρήκαν μεγάλη απήχηση στην ερευνητική κοινότητα και αποτέλεσαν τη βάση για αρκετά πλαίσια ανάπτυξης εφαρμογών επίγνωσης πλαισίου που προτάθηκαν από τότε.

Οι Hong και Landay (12) επέκτειναν τη δουλειά του Dey, μελετώντας την ιδέα μιας υποδομής υπηρεσιών, ενδιάμεσου λογισμικού. Πρότειναν μία γενική αρχιτεκτονική, όπου η διαχείριση του πλαισίου παρέχεται ως ένα σύνολο υπηρεσιών από μια υποδομή ενδιάμεσου λογισμικού. Τα widgets αντικαθίστανται από τεχνικές εύρεσης υπηρεσιών μεταξύ των διάχυτων κόμβων. Αυτή η προσέγγιση παρέχει ανοχή σε σφάλματα που μπορεί να προκύψουν από την κατάρρευση κάποιου widget, αλλά χάνει σε αποτελεσματικότητα, λόγω της αυξημένης δικτυακής επικοινωνίας που απαιτεί μεταξύ των συστατικών του συστήματος.

Ο Winograd (81) συνέκρινε διαφορετικές αρχιτεκτονικές εκδοχές για το χτίσιμο συστημάτων επίγνωσης πλαισίου και όρισε κριτήρια και συμβιβασμούς στην επιλογή του αρχιτεκτονικού μοντέλου. Κατέληξε στο ότι, μία προσέγγιση μαυροπίνακα (blackboard-based) είναι πιο ευέλικτη, από ό,τι αρχιτεκτονικές βασισμένες σε προγραμματίδια, όπως το Context Toolkit. Σε αυτήν την αρχιτεκτονική, οι εφαρμογές γράφουν μηνύματα σε ένα κοινό μέσο, το μαυροπίνακα και εγγράφονται σε κάποια υπηρεσία ειδοποίησης, ώστε να ειδοποιηθούν, όταν συμβούν συγκεκριμένα γεγονότα.

Η Daniela Petrelli (57) περιγράφει ένα απλό αρχιτεκτονικό πλαίσιο που έχει χρησιμοποιηθεί για την ανάπτυξη δύο εφαρμογών επίγνωσης πλαισίου για μουσεία. Το πλαίσιο στοχεύει στην υποστήριξη εφαρμογών σε έξυπνους χώρους, όπως ένα μουσείο. Παρέχει μία υπηρεσία βασισμένη σε απλούς κανόνες για την επιλογή των πληροφοριών που θα σταλούν στη συσκευή του χρήστη. Όλη η διαχείριση γίνεται από έναν κεντρικό εξυπηρετητή που δέχεται τα δεδομένα πλαισίου από την εφαρμογή και το περιβάλλον εκτέλεσης και στέλνει δεδομένα στη συσκευή του χρήστη.

Το CoolAgent (19) είναι ένα πλαίσιο βασισμένο σε μια αρχιτεκτονική με πολλαπλούς πράκτορες. Υποστηρίζει υπηρεσίες συλλογής και διαμοιρασμού πλαισίου και εξαγωγής συμπερασμάτων. Το πλαίσιο μοντελοποιείται με χρήση οντολογιών και RDF (Resource Description Framework)(44). Οι υπηρεσίες παρέχονται μέσω πρακτόρων με συγκεκριμένους ρόλους, ενώ η εξαγωγή συμπερασμάτων γίνεται τοπικά πάνω σε μία βάση δεδομένων πλαισίου, με ένα σύστημα κανόνων υλοποιημένο σε

Prolog. Η βάση δεδομένων βρίσκεται σε έναν κεντρικό κόμβο εξυπηρετητή, που διαχειρίζεται τα δεδομένα πλαισίου και τα παρέχει στις εφαρμογές.

Στην δημοσίευση του Michael Samulowitz και λοιπών (64) προτείνεται ένα πλαίσιο που στοχεύει στην ανακάλυψη και αυτόματη εκτέλεση κατάλληλων υπηρεσιών σε διάχυτα περιβάλλοντα. Στο σύστημα CAPEUS που υλοποιήθηκε, η επιλογή της κατάλληλης υπηρεσίας προς εκτέλεση γίνεται με βάση τις ανάγκες του χρήστη, που καθορίζονται από το πλαίσιο και τις παρεχόμενες υπηρεσίες. Οι ανάγκες του χρήστη και οι διαθέσιμες υπηρεσίες κωδικοποιούνται υπό μορφή «περιορισμών», δίνοντας τιμές σε κάποια γνωρίσματα του πλαισίου. Οι περιορισμοί καταχωρούνται μέσα σε έγγραφα τα οποία μεταφέρονται για να βρεθεί η κατάλληλη υπηρεσία. Τα έγγραφα ονομάζονται CAPs (Context Aware Packets) και περιέχουν επιπλέον ένα πλάνο, που υποδεικνύει από ποιους παρόχους υπηρεσιών πρέπει να περάσει το πακέτο. Κάθε πάροχος παραλήπτης ελέγχει αν η υπηρεσία που παρέχει ικανοποιεί τους περιορισμούς του CAP. Επιπλέον, μπορεί να κάνει υποθέσεις για τον κατάλληλο πάροχο και να του προωθήσει το CAP. Η εύρεση του κατάλληλου προμηθευτή πλαισίου, ακολουθεί την ίδια φιλοσοφία, με την εύρεση της κατάλληλης υπηρεσίας, όπως περιεγράφηκε παραπάνω. Παρόλα αυτά, τα CAPs δεν μπορούν να χρησιμοποιηθούν για την ανίχνευση μεταβολών στο πλαίσιο.

Το Context Fabric (37) είναι ουσιαστικά η συνέχεια της εργασίας (36) από το Hong. Σε αυτήν την εργασία προτείνεται ένα αρχιτεκτονικό πλαίσιο για την ανάπτυξη και υποστήριξη εφαρμογών επίγνωσης πλαισίου. Τα βασικά δομικά συστατικά της αρχιτεκτονικής είναι: μία βάση δεδομένων για την αποθήκευση και μοντελοποίηση του πλαισίου, μία γλώσσα αναπαράστασης (Context Specification Language) και μηχανισμοί ασφαλείας για την εμπιστευτικότητα των δεδομένων.

Το Solar (17) αποτελεί μία υποδομή για τη συλλογή και το διαμοιρασμό του πλαισίου σε διάχυτα περιβάλλοντα. Η πληροφορία πλαισίου που παράγεται από τις πηγές μοντελοποιείται σε γεγονότα. Η εφαρμογή μπορεί να ορίσει ένα γράφο από τελεστές (φίλτρα, μετατροπείς κ.α.), οι οποίοι μεσολαβούν μεταξύ των προμηθευτών και της εφαρμογής. Τα γεγονότα διασχίζουν τους κόμβους του γράφου και παραδίδονται στις εφαρμογές που έχουν ενδιαφερθεί. Τα χαρακτηριστικά του πλαισίου που επιθυμεί η εφαρμογή, περιγράφονται με μία συγκεκριμένη γλώσσα αναπαράστασης.

Η Gaia (63)είναι μια υποδομή ενδιάμεσου λογισμικού, που επεκτείνει τα τυπικά λειτουργικά συστήματα παρέχοντας χαρακτηριστικά επίγνωσης πλαισίου στις εφαρμογές. Στοχεύει στην ανάπτυξη και υποστήριξη εφαρμογών επίγνωσης πλαισίου για ενεργούς χώρους. Παρέχει υπηρεσίες για εύρεση και χρήση υπολογιστικών πόρων, κατανομή της πληροφορίας πλαισίου και εξαγωγή συμπερασμάτων. Το πλαίσιο μοντελοποιείται σε κατηγορήματα τεσσάρων πεδίων και χρησιμοποιούνται κανόνες κατηγορηματικής λογικής πρώτης τάξεως για την εξαγωγή συμπερασμάτων. Οι κανόνες γράφονται σε DAML+OIL.

Στην δημοσίευση των Joao Pedro Sousa και David Garlan (71) προτείνεται το AURA, μία υποδομή για την ανάπτυξη εφαρμογών επίγνωσης πλαισίου. Βασίζεται στην ιδέα ότι, στις εφαρμογές επίγνωσης πλαισίου ο πιο περιορισμένος και κρίσιμος πόρος είναι η προσοχή του χρήστη. Η προσοχή ως παράμετρος του πλαισίου, έχει την έννοια της συγκέντρωσης και της διάθεσης χρόνου από το χρήστη για να ασχοληθεί με την εφαρμογή του. Για το λόγο αυτό στοχεύει στην ανάπτυξη εφαρμογών, που προσαρμόζονται αυτόματα στις αλλαγές του περιβάλλοντος, ελαχιστοποιώντας την συμμετοχή του χρήστη. Το AURA στοχεύει στη παροχή τριών βασικών υπηρεσιών: διαφανή επιλογή του κατάλληλου μοντέλου αλληλεπίδρασης του χρήστη με μια διαδικασία, ενημερότητα για το πλαίσιο, πρόβλεψη της επιθυμητής ενέργειας και πραγματοποίηση της, χωρίς την συμμετοχή του χρήστη.

Στη δημοσίευση των Licia Capra, Wolfgang Emmerich και Cecilia Mascolo (14) προτείνεται το CARISMA, ένα πλαίσιο ενδιάμεσου λογισμικού για εφαρμογές επίγνωσης πλαισίου. Το σύστημα βασίζεται στην παροχή πολλαπλών υλοποιήσεων της ίδιας υπηρεσίας, ώστε να παρέχεται με διαφορετικό τρόπο ανάλογα με το πλαίσιο. Οι διαφορετικοί τρόποι με τους οποίους παρέχεται μια υπηρεσία ονομάζονται πολιτικές. Η συμπεριφορά του ενδιάμεσου λογισμικού περιγράφεται ως ένα σύνολο συσχετίσεων μεταξύ των παρεχόμενων υπηρεσιών, των πολιτικών με τις οποίες παρέχονται οι υπηρεσίες και τις συνθήκες του πλαισίου, που πρέπει να ισχύουν για την εφαρμογή μιας πολιτικής.

Η σημασιολογική πληροφορία για το πλαίσιο της εφαρμογής κωδικοποιείται σε XML έγγραφα, σχηματίζοντας το προφίλ της εφαρμογής. Όταν εκτελείται μία υπηρεσία της εφαρμογής, το ενδιάμεσο λογισμικό ελέγχει το προφίλ της εφαρμογής και το συγκρίνει με το τρέχον πλαίσιο. Ανάλογα με αυτό, επιλέγει την κατάλληλη

πολιτική για την εκτέλεση της υπηρεσίας. Το προφίλ της εφαρμογής μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης. Επιπλέον, χρησιμοποιείται ένας μηχανισμός για την επίλυση συγκρούσεων στην επιλογή της κατάλληλης πολιτικής, βασισμένος σε μια μικροοικονομική προσέγγιση.

Το πλαίσιο ενδιάμεσου λογισμικού Hydrogen (35) υιοθετεί μία αρχιτεκτονική διομότιμης επικοινωνίας (peer-to-peer), όπου όλα τα τμήματα για τη συλλογή και διαχείριση του πλαισίου βρίσκονται πάνω σε κινητή συσκευή. Η αρχιτεκτονική του είναι τριών επιπέδων και είναι επεκτάσιμη για να μπορεί να συμπεριλάβει όλων των ειδών τις πληροφορίες πλαισίου. Το σύστημα είναι ανεξαρτητοποιημένο από κεντρικούς κόμβους παροχής υπηρεσιών και παρέχει τη δυνατότητα στην εφαρμογή να λειτουργεί, ακόμα και όταν η συσκευή δεν είναι συνδεδεμένη σε κάποιο δίκτυο. Παρέχει υπηρεσίες για συλλογή του πλαισίου από κατανεμημένους προμηθευτές και άλλους κόμβους. Το ενδιάμεσο λογισμικό έχει τη δυνατότητα να αποθηκεύσει περιορισμένη ποσότητα πληροφορίας πλαισίου. Η πληροφορία μπορεί να μεταφερθεί μεταξύ των εφαρμογών με μία υπηρεσία διαμοιρασμού του πλαισίου. Όταν οι συσκευές βρεθούν σε κοντινή απόσταση πραγματοποιείται ασύρματα διαμοιρασμός πληροφορίας, ώστε να περιλαμβάνεται σε παραπάνω από έναν κόμβους. Η επικοινωνία μεταξύ των εφαρμογών γίνεται πάνω από TCP/IP, με XML μηνύματα.

Το SOCAM (Service Oriented Context-Aware Middleware) (32) είναι επίσης ένα πλαίσιο που στοχεύει στη συλλογή και παροχή της πληροφορίας πλαισίου. Ακολουθεί τη φιλοσοφία ενός κεντρικού υποσυστήματος για τη διαχείριση του πλαισίου. Ο διερμηνέας πλαισίου (context interpreter) συλλέγει τα δεδομένα από κατανεμημένους αισθητήρες, τα επεξεργάζεται και τα παρέχει στις εφαρμογές μέσω ερωτήσεων. Οι υπηρεσίες, που μπορούν να χρησιμοποιήσουν οι εφαρμογές, βρίσκονται στην κορυφή της αρχιτεκτονικής, είτε τοπικά στη συσκευή είτε κατανεμημένες στο δίκτυο. Οι εφαρμογές μπορούν να κάνουν χρήση διαφορετικών επιπέδων πλαισίου ρωτώντας την κεντρική μονάδα ή ακούγοντας τα γεγονότα που στέλνουν οι προμηθευτές. Η επικοινωνία και εδώ βασίζεται σε Java RMI.

Στη δημοσίευση των Xiaohang Wang, Jin Song Dong, ChungYau Chin, Sanka Ravipriya Hettiarachchi και Daqing Zhang (82), την ίδια ερευνητική ομάδα που πρότεινε το SOCAM, οι οντολογίες του συστήματος επεκτείνονται για να συμπεριλάβουν πλαίσιο για ευφυείς χώρους. Το νέο πλαίσιο ονομάζεται Semantic

Space και στοχεύει στην ανάπτυξη και υποστήριξη εφαρμογών επίγνωσης πλαισίου για ευφυείς χώρους.

Το πλαίσιο SCI (30) παρέχει υπηρεσίες ενδιάμεσου λογισμικού για διαχείριση του πλαισίου, όπως αναζήτηση, σύνθεση και παροχή πληροφορίας. Το SCI στοχεύει κυρίως στη δυναμική σύνθεση της πληροφορίας, την ανοχή σε σφάλματα και την κλιμάκωση σε έξυπνους χώρους. Την πληροφορία διαχειρίζονται κεντρικοί διακομιστές και μοντελοποιείται σε κατευθυνόμενους γράφους. Οι εφαρμογές κάνουν ερωτήσεις για την απόκτηση της πληροφορίας.

Το CMF (Context Manager Framework) (42) είναι ένα πλαίσιο διαχείρισης πλαισίου που στοχεύει στη συλλογή, ερμηνεία και παροχή της πληροφορίας πλαισίου. Το σύστημα διαχειρίζεται κυρίως πλαίσιο από το περιβάλλον (θόρυβο, θέση, θερμοκρασία κ.α.), το οποίο μοντελοποιείται με χρήση οντολογιών. Χρησιμοποιείται ασαφής λογική και Μπεϋζιανή μάθηση για τη εξαγωγή υψηλότερου σημασιολογικά πλαισίου από χαμηλού επιπέδου δεδομένα. Το CMF ακολουθεί κλασσική ιεραρχική αρχιτεκτονική, με τα κομμάτια του συστήματος τοποθετημένα σε επίπεδα. Η αρχιτεκτονική περιλαμβάνει τέσσερις λειτουργικές οντότητες: το διαχειριστή του πλαισίου, τους διακομιστές που παρέχουν το πλαίσιο (προμηθευτές), τις υπηρεσίες αναγνώρισης του πλαισίου και την εφαρμογή. Οι διακομιστές είναι κατανεμημένοι, ενώ ο διαχειριστής του πλαισίου, λειτουργεί ως μια κεντρικοποιημένη μονάδα κάτω από την εφαρμογή. Στις αρμοδιότητές του είναι να αποθηκεύει κατάλληλα τα δεδομένα που του στέλνουν οι προμηθευτές και να παρέχει την εξαγόμενη πληροφορία στις εφαρμογές. Οι υπηρεσίες αναγνώρισης χρησιμοποιούνται για την ερμηνεία του πλαισίου με βάση μια καθορισμένη οντολογία και βρίσκονται κατανεμημένες στο περιβάλλον λειτουργίας.

Το JCAF (Java Context Awareness Framework) (9) υποστηρίζει παράλληλα και την υποδομή αλλά και το προγραμματιστικό πλαίσιο για την ανάπτυξη εφαρμογών επίγνωσης στη γλώσσα Java. Πληροφορίες πλαισίου ελέγχονται από διαφορετικές υπηρεσίες από τις οποίες οι χρήστες μπορούν και τις να δημοσιεύσουν αλλά και να τις ανακτήσουν. Η επικοινωνία βασίζεται στην Java RMI.

Το ενδιάμεσο λογισμικό PACE (34) παρέχει πληροφορίες πλαισίου και επιλογές διαχείρισης μαζί με ένα προγραμματιστικό οδηγό και εργαλεία, για την βοήθεια των εφαρμογών επίγνωσης πλαισίου στις διαδικασίες αποθήκευσης,

πρόσβασης και χρησιμοποίησης των δεδομένων πλαισίου που διαχειρίζεται το PACE. Ακόμα, υποστηρίζει την λήψη αποφάσεων από τις εφαρμογές, με βάση τις προτιμήσεις του χρήστη.

Το CAMUS είναι μία υπολογιστική υποδομή για δικτυακά, ευφυή ρομπότ με δυνατότητες επίγνωσης πλαισίου (40, 49). Υποστηρίζει μια πληθώρα δεδομένων πλαισίου, όπως πληροφορίες χρήστη, τοποθεσίας, περιβάλλοντος και συλλογισμού πλαισίου. Παρά ταύτα, το σύστημα αυτό σε βασίζεται σε διαδικτυακές υπηρεσίες και δουλεύει σε κλειστά περιβάλλοντα.

Το Citron είναι ένα πλαίσιο απόκτησης πληροφοριών πλαισίου για προσωπικές συσκευές(83). Συλλέγει πληροφορίες για έναν χρήστη και τον περιβάλλοντα χώρο του και τις χρησιμοποιεί για να προσαρμόσει ανάλογα την συμπεριφορά των εφαρμογές που εκτελούνται στην προσωπική του συσκευή.

Το CASS (54) είναι ένα ακόμα πλαίσιο που υιοθετεί την αρχιτεκτονική με έναν κεντρικό κόμβο για την επεξεργασία του πλαισίου. Οι εφαρμογές-χρήστες είναι ανεξαρτητοποιημένες από την επεξεργασία του πλαισίου και επικοινωνούν με τον κεντρικό κόμβο λαμβάνοντας την επεξεργασμένη πληροφορία. Το ενδιάμεσο λογισμικό περιλαμβάνει υπηρεσίες για επικοινωνία με κατανεμημένους αισθητήρες, μια μηχανή εξαγωγής συμπερασμάτων, διερμηνέα του πλαισίου και μία βάση δεδομένων για την αποθήκευση και την ανάκτηση της πληροφορίας. Ο πελάτης περιλαμβάνει τμήματα για επικοινωνία με τις υπηρεσίες του ενδιάμεσου λογισμικού.

Το σύστημα WASP (Web Architectures for Services Platforms) (22) είναι μια υποδομή για την ανάπτυξη και υποστήριξη εφαρμογών επίγνωσης πλαισίου, που παρέχει υπηρεσίες διαδικτύου, κινητής ομιλίας και δίκτυα κινητών τηλεφώνων τρίτης γενιάς. Στην αρχιτεκτονική του WASP ένα κεντρικό υποσύστημα (Context Interpreter) είναι υπεύθυνο για τη συλλογή και παροχή της πληροφορίας. Η πληροφορία παρέχεται στο υποσύστημα παρακολούθησης (Monitor Module), το οποίο υποστηρίζεται από ένα σύνολο μονάδων αποθήκευσης, για καταχώρηση της πληροφορίας για μελλοντική χρήση. Για την διασύνδεση μιας εφαρμογής με τη πλατφόρμα WASP, χρησιμοποιείται η γλώσσα WSL, στην οποία κωδικοποιούνται η συμπεριφορά και τα στοιχεία της εφαρμογής. Η αναπαράσταση της πληροφορίας γίνεται με χρήση οντολογιών. Η διαχείριση των οντολογιών γίνεται με την OWL.

Το CoWSAMI είναι ένα ενδιάμεσο λογισμικό που υποστηρίζει εφαρμογές επίγνωσης πλαισίου σε διάχυτα περιβάλλοντα (26). Παρέχει έναν διαχειριστή πλαισίου, το οποίο αναλαμβάνει τη διαχείριση των προμηθευτών πλαισίου. Έχοντας μία σχεσιακή αναπαράσταση των πληροφοριών πλαισίου, ο ορισμός τους γίνεται ανάλογα με τον προμηθευτή που τις παρέχει. Το έργο inContext (74) παρέχει διάφορες τεχνικές υποστήριξης ομάδων συνεργασίας στον τομέα της επίγνωσης πλαισίου. Είναι σχεδιασμένο για διαδικτυακά περιβάλλοντα ομαδικής εργασίας. Το inContext παρέχει μεθόδους μοντελοποίησης, αποθήκευσης, συλλογισμού και ανταλλαγής πληροφοριών πλαισίου ανάμεσα σε υπηρεσίες, μέσω διαδικτύου.

Το πλαίσιο ESCAPE (74) είναι ένα διαδικτυακό σύστημα διαχείρισης δεδομένων πλαισίου που εξειδικεύεται σε ομαδική εργασία και έκτακτες καταστάσεις, διαχείριση κινδύνου, καταστροφής κλπ. Το ESCAPE είναι σχεδιασμένο έτσι ώστε να παρέχει υπηρεσίες περιβάλλοντος χρήστη (front-end) στις κινητές συσκευές και παρασκηνιακές υπηρεσίες (back-end) στα αντίστοιχα συστήματα. Για τις κινητές συσκευές το ESCAPE αποτελείται από λειτουργίες ανίχνευσης δεδομένων, καθώς και ανταλλαγής τους, μέσω του διαδικτύου και οι οποίες εκτελούνται μέσω ενός κατά περίσταση (ad hoc) δικτύου κινητών συσκευών. Από την άλλη, το μέρος των υπηρεσιών παρασκηνίου περιλαμβάνει μια διαδικτυακή υπηρεσία αποθήκευσης και ανταλλαγής πληροφοριών πλαισίου ανάμεσα σε διαφορετικές συσκευές.

Το AmbieSensen (41) προσβλέπει στο μέλλον των ευφυών χώρων σε επίπεδα ύπαρξης και λειτουργίας, είτε πρόκειται για ανθρώπους, είτε για προγραμματιστικά περιβάλλοντα. Είναι βασισμένο στην ιδέα ότι πληροφορίες και δεδομένα αντιστοιχίζονται καθημερινά σε όλα τα αντικείμενα και τους χώρους που μας περιστοιχίζουν. Το όραμα του είναι: « Η σωστή πληροφόρηση στον σωστό χρήστη, υπό τις επικρατούσες συνθήκες». Η προσαρμοστικότητα αυτή επιτυγχάνεται με μία αρχιτεκτονική δύο επιπέδων. Το πρώτο επίπεδο είναι ένα ενδιάμεσο λογισμικό, το οποίο υλοποιεί έναν γενικό μηχανισμό για την συλλογή και διαχείριση πληροφοριών πλαισίου. Το δεύτερο επίπεδο προσφέρει μία αυτοματοποιημένη αξιολόγηση της παρούσας κατάσταση μέσω Συλλογισμού Κατά Περίσταση (Case-Based Reasoning).

Αυτή τη στιγμή ο έλεγχος και η ασφάλεια των δεδομένων που διαχειρίζονται οι εφαρμογές επίγνωσης πλαισίου είναι η κύρια έγνοια, τόσο των προγραμματιστών όσο και των τελικών χρηστών. Οι υπάρχουσες τεχνολογίες ενδιαμέσων λογισμικών δεν

βοηθούν προς αυτή την κατεύθυνση, καθώς η χρήση και η προστασία των πληροφοριών που χρησιμοποιούνται από πολλαπλές συσκευές ολοένα και δυσκολεύει τον μέσο χρήστη, παρόλη την αλματώδη αύξηση της αποθήκευσης των δεδομένων των εφαρμογών σε περιβάλλοντα "σύννεφου".

3.2.Σύγκριση Ενδιάμεσων Λογισμικών Επίγνωσης Πλαισίου και υποψήφιων Ενδιάμεσων Λογισμικών Εφαρμογών

Σε αυτό το σημείο θα πρέπει να εξεταστεί αν τουλάχιστον κάποιο από τα Ενδιάμεσα Λογισμικά Επίγνωσης Πλαισίου μπορεί να ικανοποιήσει τις απαιτήσεις που έχουν προκύψει για την παρούσα έρευνα. Από τα παραπάνω έχει προκύψει ότι για να συμβεί κάτι τέτοιο, θα πρέπει το λογισμικό αυτό να ικανοποιεί ταυτόχρονα τουλάχιστον τα παρακάτω σημεία:

- **Να παρέχεται για πολλαπλές συσκευές**: Το λογισμικό αυτό θα πρέπει να μπορεί να εγκατασταθεί ταυτόχρονα σε περισσότερες από μία συσκευές με διαφορετικές λειτουργίες, παρέχοντας ένα επίπεδο αφαίρεσης ως προς τις δυνατότητές του. Το σημείο αυτό στη βιβλιογραφία αφορά συνήθως σε δυνατότητες Διαδικτύου των Πραγμάτων (IoT), καθώς δίνει τη δυνατότητα συλλογής δεδομένων από πολλαπλές συσκευές.
- **Να παρέχεται για πολλαπλές πλατφόρμες**: Το λογισμικό αυτό θα πρέπει να μπορεί να εκτελεστεί σε ανομοιογενή περιβάλλοντα λογισμικού και λειτουργικά συστήματα. Σε γενικές γραμμές το σημείο αυτό δεν ικανοποιείται, παρά μόνο σε κάποιες περιπτώσεις, όπου η χρήση γλωσσών προγραμματισμού, όπως η Java, το επιτρέπουν.
- **Τα δεδομένα που παράγονται να ανήκουν στον χρήστη**: Το λογισμικό αυτό θα πρέπει να αποθηκεύει τα δεδομένα σε χώρο που ανήκει στον χρήστη. Ιδανικά, ο χώρος αυτός δεν βρίσκεται στην ίδια συσκευή από την οποία συλλέγονται.
- **Να παρέχεται με δυνατότητες διαλειτουργικότητας μεταξύ εφαρμογών**: Το λογισμικό αυτό θα πρέπει να προβλέπει και να υποστηρίζει την επικοινωνία και τον διαμοιρασμό δεδομένων και λειτουργιών μεταξύ διαφορετικών εφαρμογών, ιδανικά και μεταξύ διαφορετικών συσκευών.

- **Να παρέχει μία ανεξάρτητη, κοινή Βάση Δεδομένων**: Το λογισμικό αυτό θα πρέπει να αποθηκεύει τα δεδομένα επίγνωσης πλαισίου σε μία Βάση Δεδομένων η οποία είναι ανεξάρτητη των εφαρμογών και είναι κοινή για την πλατφόρμα ενδιάμεσου λογισμικού.

	Παρέχεται για πολλαπλές συσκευές (IoT)	Παρέχεται για πολλαπλές πλατφόρμες	Δεδομένα που ανήκουν στον χρήστη	Διαλειτουργικότητα μεταξύ εφαρμογών	Ανεξάρτητη κοινή βάση δεδομένων
Context Toolkit				•	•
CoolAgent	•			•	
CAPEUS	•		•		
Context Fabric	•		•	•	
Solar	•				•
Gaia	•			•	
Aura	•				
CARISMA			•		•
Hydrogen			•	•	•
SOCAM	•		•		
CoBrA	•				•
SCI	•				
CMF			•		
JCAF	•	•			
Citron			•		
e-SENSE	•		•		
HCoM	•			•	
MoCA	•		•		
SIM	•		•		
CROCO	•		•	•	
Hydra3	•				
SALES	•	•			•
Feel@Home	•				•

Πίνακας 2: Σύγκριση Ενδιάμεσων Λογισμικών Επίγνωσης Πλαισίου

Εκτός των παραπάνω σημείων, σε δευτερεύοντα βαθμό εξετάζονται και κριτήρια όπως η διαθεσιμότητα, υπό τη μορφή ανοιχτού κώδικα, αν προορίζεται για

μη βιομηχανικές εφαρμογές κ.α. Συνοπτικά, τα σημαντικότερα ενδιάμεσα λογισμικά εμφανίζονται, ως προς αυτά τα κριτήρια, στον Πίνακας 2.

Εφόσον δεν διαφαίνεται από τη βιβλιογραφία ή τις υπάρχουσες εφαρμογές ενδιάμεσου λογισμικού επίγνωσης πλαισίου ότι ικανοποιούνται οι γενικές απαιτήσεις για την παρούσα πλατφόρμα επίγνωσης πλαισίου, θα πρέπει να εξεταστούν τα υπάρχοντα ή προτεινόμενα ενδιάμεσα λογισμικά εφαρμογών, πάνω στα οποία θα μπορούσε αυτή να αναπτυχθεί. Δυνητικά, οποιοδήποτε από αυτά θα μπορούσε να χρησιμοποιηθεί αν υπήρχε πρόσβαση στον πηγαίο κώδικά τους και οι απαραίτητοι, πολύ αυξημένοι, πόροι για τον σχεδιασμό και την ανάπτυξη των προ απαιτούμενων της πλατφόρμας.

Σε γενικές γραμμές, υπάρχει πληθώρα ενδιάμεσων λογισμικών εφαρμογών πάνω στα οποία θα μπορούσε να αναπτυχθούν πλατφόρμες επίγνωσης πλαισίου, όπως τα PhoneGap (3), Enyo, MoSync(48), NEXT, RhoMobile Suite (52), QuickConnectFamily (61), iUI, Kivy, Pega AMP, iPFaces και webinos (29). Αυτά χαρακτηρίζονται από μία έμφαση στις φορητές συσκευές και κυρίως έξυπνα κινητά τηλέφωνα. Βάσει των γενικών απαιτήσεων που περιεγράφηκαν, μερικά από αυτά παρέχουν την επιθυμητή επέκταση των δυνατοτήτων τους σε υπολογιστές ή έξυπνες τηλεοράσεις. Ταυτόχρονα, κάποια παρέχουν δυνατότητες πρόσβασης σε Βάσεις Δεδομένων, συνήθως στην ίδια συσκευή, όπως είναι το ζητούμενο, αλλά χωρίς να παρέχουν κάποιο επίπεδο αγνωστικής αφαίρεσης ως προς την πρόσβαση σε αυτές τις υπηρεσίες σε απομακρυσμένους διακομιστές. Μερικά παρέχονται και ως εφαρμογές πελάτη-διακομιστή ή μπορούν να συγχρονιστούν με κάποιο εξειδικευμένο διακομιστή εφαρμογών, για εφαρμογές σε γραφεία, μουσεία ή άλλους κοινόχρηστους χώρους, όμως δεν προβλέπεται ο διακομιστής αυτός να ανήκει στον χρήστη, ενώ πρέπει να αναπτύσσεται ξεχωριστή υλοποίηση για κάθε εφαρμογή. Παράλληλα, δεν σχεδιάζονται για να παρέχουν κάποιο επίπεδο αγνωστικής αφαίρεσης για την διαλειτουργικότητα μεταξύ εφαρμογών ή εφαρμογών σε διαφορετικές συσκευές, καθώς στις περισσότερες περιπτώσεις η πρόβλεψη αφορά στην υλοποίηση απλών εφαρμογών που περιορίζονται σε μία συσκευή και τους αισθητήρες της, ενώ δεν σχεδιάζονται ώστε να λαμβάνουν υπόψη το ενδεχόμενο η πλατφόρμα να χρησιμοποιείται από τον ίδιο χρήστη σε πολλαπλές συσκευές. Τέλος, πολλές από αυτές παρέχονται ως τελικά εμπορικά προϊόντα, αποκλείοντας έτσι την χρήση τους για διανομή εφαρμογών από τρίτους.

Μεταξύ των υποψηφίων ενδιάμεσων λογισμικών εφαρμογών που εξετάστηκαν για την ανάπτυξη και την ενσωμάτωση της προτεινόμενης πλατφόρμας Επίγνωσης Πλαισίου, οι απαιτούμενες προδιαγραφές ικανοποιούνται επαρκώς μόνο από την πλατφόρμα webinos (29).

3.3.webinos

Το webinos είναι ένα έργο χρηματοδοτούμενο από την ΕΕ που είχε αρχική διάρκεια τρία χρόνια (από τον Σεπτέμβριο του 2010 έως τον Δεκέμβριο του 2013). Μετά τη λήξη του έργου μετατράπηκε σε Foundation με σκοπό την περεταίρω εξέλιξή του. Έχει παραπάνω από είκοσι συνεργάτες σε όλη την Ευρώπη ανάμεσα στους οποίους συγκαταλέγονται ακαδημαϊκά ιδρύματα, εταιρείες βιομηχανικής έρευνας, εταιρείες ανάπτυξης λογισμικού, αυτοκινητοβιομηχανίες κ.α. Στόχος του είναι να προσφέρει μια πλατφόρμα που θα υποστηρίζει την γρήγορη δημιουργία καινοτόμων και ασφαλών δικτυακών εφαρμογών για κάθε είδους έξυπνες συσκευές, όπως κινητά τηλέφωνα, Η/Υ (φορητούς και μη), τηλεοράσεις και συσκευές ενσωματωμένες σε οχήματα. Προσδοκία του είναι να παρέχει ένα δικτυακά εφαρμόσιμο ομογενοποιημένο πλαίσιο πολλαπλών συσκευών, πολλαπλών χρηστών και πολλαπλών λειτουργικών περιβαλλόντων (75).

Πολλαπλών συσκευών σημαίνει πως οι διάφορες συσκευές που τρέχουν στο webinos έχουν την δυνατότητα να επικοινωνούν μεταξύ τους. Πολλαπλών χρηστών σημαίνει πως ένας χρήστης με τις συσκευές που έχει στην διάθεση του μπορεί εύκολα να επικοινωνήσει με τις συσκευές ενός άλλου χρήστη. Πολλαπλών λειτουργικών περιβαλλόντων σημαίνει πως το webinos είναι προσαρμοσμένο έτσι ώστε να εκτελείται σε διαφορετικά μεταξύ τους λειτουργικά περιβάλλοντα, καθώς, παρά το γεγονός πως το αυτοκίνητο και η τηλεόραση εκτελούν διαφορετικά λογισμικά, μπορούν και οι δύο συσκευές να εκτελούν το webinos. Επίσης δικτυακά εφαρμόσιμες δηλώνει πως προγραμματίζεται χρησιμοποιώντας δικτυακές τεχνολογίες : html+css+javascript. Τέλος, ομογενοποιημένο σημαίνει πως διαφορετικά πεδία έχουν την δυνατότητα να ανταλλάσσουν μηνύματα, αρκετά όμοια με τη διαδικασία του ηλεκτρονικού ταχυδρομείου.

3.3.1. Το όραμα του webinos

Το webinos οραματίζεται δικτυακές εφαρμογές να τρέχουνε και να υλοποιούνται σε ένα εύρος συνδεδεμένων συσκευών οι οποίες αλληλεπιδρούν απρόσκοπτα μεταξύ τους για να συνεισφέρουν στην ανάπτυξη του διαδικτύου και να διευκολύνουν τους χρήστες με πιο ελκυστικές, καινοτόμες και αξιόπιστες εφαρμογές. Για να το πετύχει αυτό, το webinos ορίζει και παρέχει μια πλατφόρμα ανοιχτής πηγής (open source platform) με πρακτική εφαρμογή και προσιτή σε όλους, της οποίας οι κύριοι στόχοι είναι (27):

- Παροχή μιας ασφαλούς πλατφόρμας για δικτυακές τεχνολογίες σε κινητές συσκευές, οικιακά μέσα (TV), Η/Υ και συσκευές οχημάτων.
- Ορισμός και δόμηση τεχνολογικά τεκμηριωμένων βασικών εργαλείων, όπου αυτά δεν προϋπάρχουν, για την εύκολη και αποδοτική ανάπτυξη εφαρμογών σε διαφορετικές συσκευές
- Ορισμός και δόμηση ενός δικτυακά εφαρμόσιμου πλαισίου ασφαλείας το οποίο απευθύνεται στις ανάγκες των χρηστών και των παρόχων των υπηρεσιών, στις υποκείμενες πλατφόρμες και στις απαιτήσεις των συσκευών, ενώ είναι τεχνολογικά τεκμηριωμένο και εύκολα αναπτύξιμο.

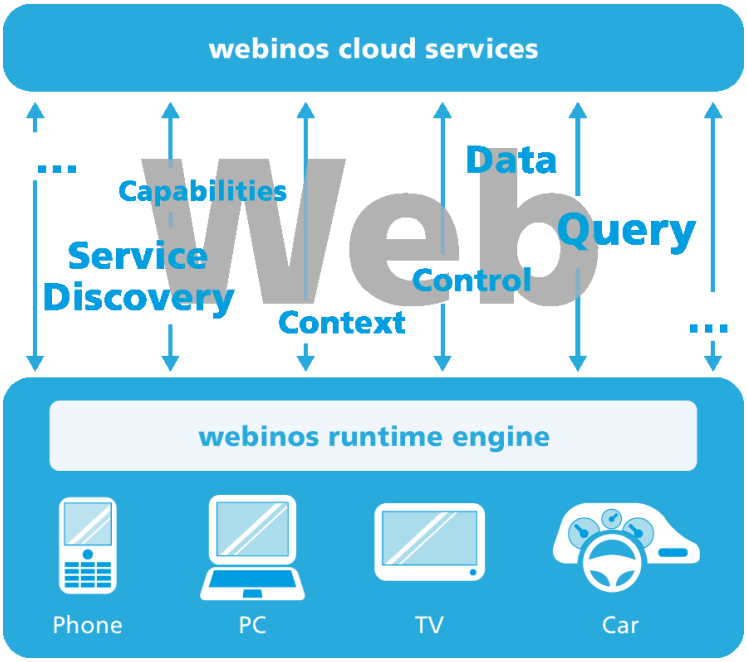
3.3.2. Τα βασικά χαρακτηριστικά του webinos

Για την επίτευξη των παραπάνω στόχων το webinos , καταρχάς, παρουσιάζει τα παρακάτω κύρια χαρακτηριστικά (75):

- Το webinos βασίζεται στα επιτεύγματα της διαδικτυακής κοινότητας και επεκτείνει το open source web runtime environment.
- Το webinos προσφέρει ένα κοινό σύνολο διεπαφών προγραμματισμού εφαρμογών (APIs) έτσι ώστε να επιτρέπει εύκολη πρόσβαση σε cross-user, cross-service και cross-device λειτουργικότητα με έναν ανοιχτό και ασφαλή τρόπο.
- Το webinos στοχεύει στην διευκόλυνση του προγραμματισμού των εφαρμογών προσφέροντας μια ενιαία εικονική συσκευή που θα αποτελείται από το σύνολο των συσκευών του χρήστη

- Οι εφαρμογές μπορούν να θέσουν σε λειτουργία και να χρησιμοποιήσουν όλες τις δυνατότητες και τους πόρους που διαθέτει το υλικό (hardware) μιας συγκεκριμένης συσκευής
- Οι εφαρμογές μπορούν να έχουν ασφαλή πρόσβαση σε ιδιωτικά και μη ιδιωτικά δεδομένα και άλλες υπηρεσίες στο «σύννεφο» και στα κοινωνικά δίκτυα, όπως και σε στοιχεία στο τερματικό του χρήστη
- Το webinos δημιουργεί ανοιχτές προδιαγραφές και open source reference implementations που δείχνουν την σκοπιμότητα των προδιαγραφών και απλοποιούν την προσαρμογή τους από την βιομηχανία.
- Οι εφαρμογές χτίζονται μία φορά από τους παρόχους λογισμικού και μπορούν να παραταχθούν οπουδήποτε.

Οι επιλεγμένες γλώσσες που χρησιμοποιούνται στο webinos υποστηρίζουν πλήρως τις διαδικτυακές τεχνολογίες, συγκεκριμένα HTML, CSS και JavaScript. Το webinos καλείται να διευθετήσει συγκεκριμένες προκλήσεις όπως είναι οι πρόβλεψη και προσαρμογή της ασφάλειας σε ένα μεγάλος εύρος συσκευών, υπηρεσιών και δικτύων. Εξίσου σημαντική είναι η ασφάλεια των προσωπικών δεδομένων των χρηστών ανεξαρτήτως της συσκευής ή της εφαρμογής που χρησιμοποιούν.



Εικόνα 4:Γενική εικόνα webinos (77)

Στην Εικόνα 4 παρουσιάζεται η γενική αρχιτεκτονική ιδέα του webinos.

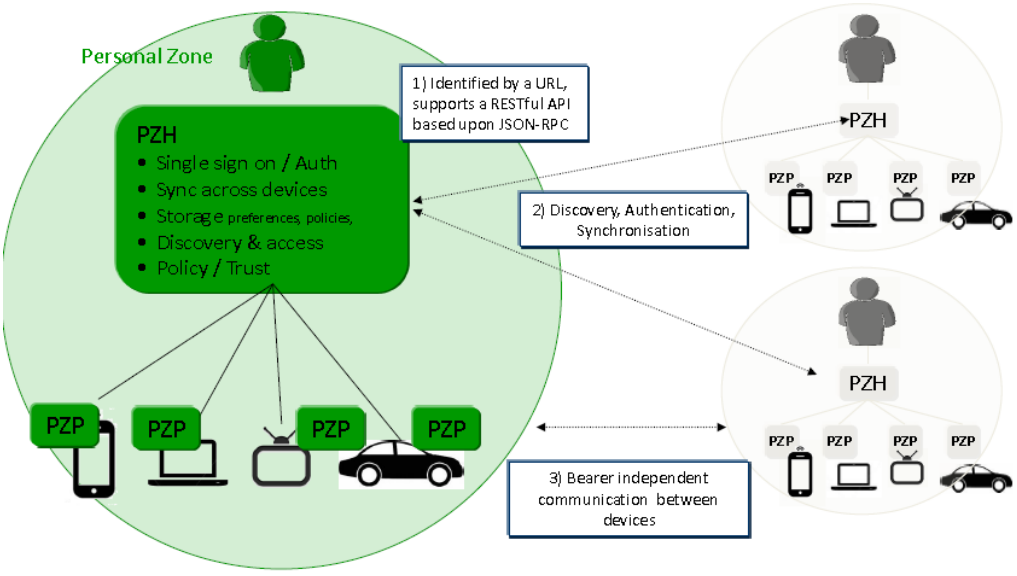
3.3.3. Καινοτομίες του webinos

Για να πετύχει όλους τους παραπάνω στόχους το webinos χρησιμοποιεί τον ακόλουθο σχεδιασμό, προτείνοντας παράλληλα για την επίτευξή του τις απαραίτητες καινοτομίες (51).

Προσωπική Ζώνη (Personal Zone - PZ)

Ο κάθε χρήστης μπορεί να έχει περισσότερες από μία συσκευές ενεργοποιημένες στο webinos. Το σύνολο αυτών των συσκευών αποτελεί την Προσωπική του Ζώνη (PZ). Η Προσωπική Ζώνη δρα ως ένα ξεχωριστό δίκτυο σε σχέση με τα υποκείμενα φυσικά δίκτυα και πρωτόκολλα. Σκοπός της είναι να παρέχει εύκολη πρόσβαση σε τοπικές και απομακρυσμένες υπηρεσίες, απλοποιώντας το έργο ενός προγραμματιστή εφαρμογών. Επίσης επιτρέπει την εύκολη ανίχνευση συσκευών και υπηρεσιών και παρέχει επικοινωνιακά μονοπάτια βασισμένα σε έμπιστες σχέσεις, αποσυνδεδεμένα από τις υποκείμενες τεχνολογίες.

Η Προσωπική Ζώνη υλοποιείται πάνω σε μία διανεμημένη αρχή η οποία αποτελείται από έναν Κόμβο Προσωπικής Ζώνης (Personal Zone Hub – PZH) και πολλαπλά Πληρεξούσια Προσωπικής Ζώνης (Personal Zone Proxy – PZP).



Εικόνα 5: Η Προσωπική Ζώνη στο webinos

Κόμβος Προσωπικής Ζώνης (Personal Zone Hub - PZH)

Ο κάθε Κόμβος Προσωπικής Ζώνης (PZH) χρησιμοποιείται για τη δημιουργία μίας και μόνο Προσωπικής Ζώνης. Είναι το κύριο στοιχείο πάνω στο οποίο συγχρονίζονται και ταυτοποιούνται όλες οι συσκευές ενός χρήστη. Σκοπός του, πέρα από τη δημιουργία της Προσωπικής Ζώνης, είναι να επιτρέπει την ασφαλή επικοινωνία των συσκευών εντός της Προσωπικής Ζώνης, καθώς και την ανίχνευση και πρόσβαση σε συσκευές άλλων Προσωπικών Ζωνών.

Πρακτικά, η υλοποίηση βασίζεται σε μία τοποθέτηση του PZH στο προσωπικό σύννεφο του webinos, προϋποθέτοντας ότι αυτή είναι ασφαλής, ακόμα και αν γίνεται σε εξυπηρετητή τρίτου. Οι βασικές λειτουργίες του PZH μέσω της διεπαφής του είναι να προσθέτει μία συσκευή στην Προσωπική Ζώνη, να ανακαλεί πιστοποιητικά για κάποια ταυτοποιημένη συσκευή και να ελέγχει την κατάσταση ταυτοποίησής της.

Για να επιτρέψει εξωτερική πρόσβαση σε μία Προσωπική Ζώνη, το webinos καθορίζει τον Κόμβο Προσωπικής Ζώνης ως μια υπηρεσία η οποία είναι προσιτή μέσω του Διαδικτύου, ο οποίος είναι διαθέσιμος οποιαδήποτε στιγμή, αντίθετα με τις προσωπικές συσκευές οι οποίες μπορούν να απενεργοποιηθούν ή να βρίσκονται εκτός εύρους επικοινωνίας. Υπάρχει απόλυτη αντιστοιχία μεταξύ των Προσωπικών Ζωνών και των Κόμβων Προσωπικών Ζωνών. Κάθε PZH ανήκει σε έναν μοναδικό χρήστη και ταυτοποιείται μέσω συγκεκριμένου URL. Για παράδειγμα, όταν ο χρήστης χρησιμοποιεί έναν δημόσιο υπολογιστή, το PZH του επιτρέπει να αποκτήσει πρόσβαση στις συσκευές της Προσωπικής του Ζώνης και τις υπηρεσίες αυτής κατά τη διάρκεια της περιόδου περιήγησης του. Επιτρέπει, επιπρόσθετα, την πρόσβαση σε άλλους χρήστες οι οποίοι έχουν λάβει το δικαίωμα από τον ιδιοκτήτη της Προσωπικής Ζώνης για αυτή την ενέργεια.

Το PZH παρέχει υποστήριξη ανακάλυψης άλλων PZH με βάση τα όνομα, ψευδώνυμο ή διεύθυνση email των χρηστών τους. Υπάρχει βέβαια πάντα η δυνατότητα να περιοριστεί η εξεύρεση του PZH από άλλους. Αυτό καθορίζεται από τον κάθε χρήστη.

Οι λειτουργίες του PZH μπορούν να συνοψιστούν στα ακόλουθα:

- Αποτελεί μία καθορισμένη οντότητα στην οποία μπορούν να σταλούν και να μεταδοθούν όλα τα μηνύματα, σαν μια προσωπική ταχυδρομική

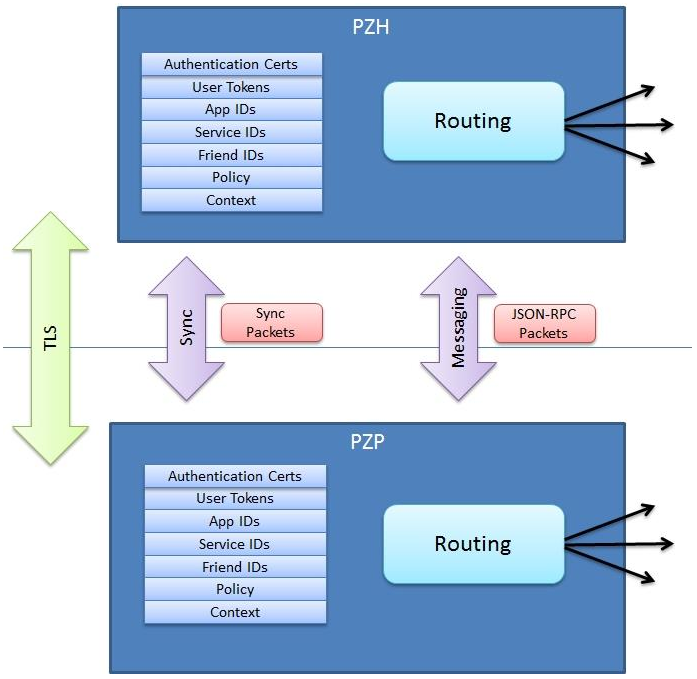
θυρίδα

- Περιέχει τα πιστοποιητικά αναγνώρισης PZP/PZH από άλλους χρήστες οι οποίοι θεωρούνται έμπιστοι
- Περιέχει τα αναγνωριστικά και πιστοποιητικά εφαρμογών, οι οποίες έχουν πρόσβαση στη Προσωπική Ζώνη
- Περιέχει τα αναγνωριστικά και πιστοποιητικά υπηρεσιών, οι οποίες έχουν πρόσβαση στη Προσωπική Ζώνη
- Περιέχει τα αναγνωριστικά και πιστοποιητικά συνδεδεμένων συσκευών, οι οποίες έχουν πρόσβαση στην Προσωπική Ζώνη
- Περιέχει όλους τους κανόνες πολιτικής που χρησιμοποιούνται
- Πραγματοποιεί την ταυτοποίηση του χρήστη
- Αναλαμβάνει την δημιουργία ασφαλούς συνεδρίας με το κάθε PZP για την μετάδοση μηνυμάτων και τον συγχρονισμό δεδομένων
- Παρέχει υπηρεσία δημιουργίας συνεδρίας για ασφαλή μετάδοση μηνυμάτων μεταξύ των εφαρμογών και των υπηρεσιών
- Παρέχει ασφαλή κοινωνική δικτύωση χρησιμοποιώντας τα πιστοποιητικά που έχουν ανταλλαχθεί μεταξύ έμπιστων χρηστών
- Παρέχει υπηρεσία σύνδεσης σε άλλες δικτυακές υπηρεσίες χρησιμοποιώντας το PZH ως ασφαλή κόμβο

Πληρεξούσιο Προσωπικής Ζώνης (Personal Zone Proxy - PZP)

Ένα PZP εκτελείται σε κάθε συσκευή με webinos και είναι υπεύθυνο για την επικοινωνία της με το PZH. Αυτή επιτυγχάνεται μέσω ενός ασφαλούς TLS καναλιού που ταυτοποιεί αμοιβαία και τις δύο πλευρές, και σε περίπτωση σφάλματος στην επικοινωνία μπορεί να γίνει έμμεση ταυτοποίηση μέσω της σύνδεσης του με ένα ήδη ταυτοποιημένο PZP. Στην περίπτωση που το PZH δεν είναι διαθέσιμο λόγω σφάλματος στη σύνδεση με το δίκτυο, είναι δουλειά του PZP να λειτουργήσει στη θέση του και να αποθηκεύσει τις απαραίτητες πληροφορίες και γεγονότα συγχρονίζοντας αυτές αργότερα με το PZH, όταν η σύνδεση είναι πλέον δυνατή (virgin PZP). Επιπρόσθετα, το PZP ευθύνεται για κάθε ανίχνευση χρησιμοποιώντας φορείς τοπικού υλικού. Ο βασικός του ρόλος είναι η εξαγωγή των APIs της συσκευής ως υπηρεσίες με έναν απόλυτα ασφαλή τρόπο. Ακόμα, το PZP συνδέεται στο χρόνο εκτέλεσης του webinos (webinos runtime, WRT) με μία διαμορφωμένη εφαρμογή διαδικτυακής υποδοχής, η οποία αποτελεί ασφαλή δίαυλο (secure channel). Το WRT είναι το τμήμα του webinos όπου όλες οι συμβατές εφαρμογές ανήκουν και εκτελούνται. Τέλος, μετά τον ορισμό της εξωτερικής διεπαφής στο PZP, μπορεί να χρησιμοποιηθεί ένα εικονικό PZP για να

γίνει δυνατή την πρόσβαση του webinos σε συσκευές, στις οποίες αυτό δεν μπορεί να εφαρμοστεί πλήρως, και με αυτό τον τρόπο να υπάρχει πρόσβαση σε υπηρεσίες διαθέσιμες σε αυτές τις συσκευές.



Εικόνα 6: Επικοινωνία μεταξύ PZH και PZP (78)

Στην Εικόνα 6 παρουσιάζεται η αρχιτεκτονική του PZH και του PZP, καθώς και η ενδιάμεση συνομιλία τους με χρησιμοποίηση του JSON-RPC (JavaScript Object Notation-Remote Procedure Call) το οποίο είναι θεμελιωμένο πάνω σε ασφαλείς, TLS (Transport Layer Security) συνεδρίες.

Εικονικό Πληρεξούσιο Προσωπικής Ζώνης (virtual PZP)

Το εικονικό πληρεξούσιο προσωπικής ζώνης (εικονικό PZP, στιγμιότυπο υπηρεσιών PZH) δημιουργήθηκε μεταγενέστερα των υπολοίπων συστατικών μίας προσωπικής ζώνης και ο ρόλος του αρχικά ήταν να «ελαφρύνει» της λειτουργίες του PZH. Αυτό διότι εξ αρχής η ιδέα ήταν το ενδιάμεσο λογισμικό να μην επιβαρύνει σημαντικά τους πόρους των συσκευών. Για το σκοπό αυτό, η υλοποίησή του γίνεται σε έναν διακομιστή του διαδικτύου και όχι απαραίτητα εντός της προσωπικής ζώνης ενός χρήστη.

Η ιδέα της Προσωπικής Ζώνης ξεκίνησε και αναπτύχθηκε κατά τη φάση της σχεδίασης του webinos, καθώς αποδείχτηκε ότι μπορεί να λειτουργήσει ως μία χρήσιμη παραδοχή που θα απλοποιεί την ανάπτυξη εφαρμογών. Έτσι, οι προγραμματιστές

μπορούν πιο εύκολα να διαχειριστούν την πολυπλοκότητα που έχει ένα σύστημα εφαρμογών πολλαπλών συσκευών που προορίζεται για χρήση σε διαφορετικές πλατφόρμες. Αρχικά, είχε υποτεθεί πως κάθε προσωπική ζώνη θα περιελάμβανε μία συσκευή και θα είχε μόνο έναν ιδιοκτήτη. Αποτέλεσμα της υπόθεσης αυτής ήταν οι προσωπικές ζώνες να θεωρούνται μικρά, περισσότερο απομονωμένα συστήματα συσκευών όπου μόνο ένας χρήστης ήταν παρών κάθε στιγμή. Αυτό βοήθησε στον αρχιτεκτονικό σχεδιασμό και στην ασφάλεια, καθώς η αυθεντικοποίηση μπορούσε να βασιστεί εν μέρει στις ταυτότητες των συσκευών, και έτσι προχώρησε το πρόγραμμα σημαντικά.

Παρόλα αυτά, έγινε ξεκάθαρο πως πρόκειται για μία υπεραπλούστευση, αφού πολλές οικιακές συσκευές ανήκουν και χρησιμοποιούνται από κάποιους ανθρώπους από κοινού και δεν έχουν έναν μοναδικό ιδιοκτήτη (π.χ. ένα οικογενειακό PC ή μία τηλεόραση). Αν γινόταν η υπόθεση πως αυτά ήταν μέρη μίας ζώνης θα προκαλούνταν προβλήματα στις εφαρμογές και τις υπηρεσίες που θα απαιτούσαν χρησιμοποίηση ταυτοτήτων και αυθεντικοποιήσεων. Για τους παραπάνω λόγους, ο σχεδιασμός διαφοροποιήθηκε στη συνέχεια του προγράμματος, ώστε να επιτρέπει στις συσκευές να εκτελούνται με διαφορετικά πληρεξούσια. Δυστυχώς, η προσέγγιση αυτή απαιτεί τα πληρεξούσια που βρίσκονται στην ίδια συσκευή να είναι απομονωμένα το ένα από το άλλο σε διαφορετικούς λογαριασμούς χρηστών. Οι λογαριασμοί αυτοί έχουν δείξει πως έχουν χαμηλή χρηστικότητα. Για παράδειγμα σε μία οικιακή ρύθμιση, αυτό απαιτεί κάθε χρήστης να είναι συνδεδεμένος ώστε να μπορούν τα πληρεξούσιά του να εκτελέσουν και να δεχτούν απομακρυσμένες εντολές. Επίσης, έτσι δεν είναι δυνατή η λειτουργία του webinos σε συσκευές και συστήματα που δεν παρέχουν την δυνατότητα χρησιμοποίησής τους από πολλαπλούς λογαριασμούς. Αυτός είναι ο λόγος που υλοποιήθηκε η λύση των πολλαπλών πληρεξουσίων ανά συσκευή.

Η έννοια του χρήστη και του λογαριασμού χρήστη είναι ασαφής και αποτυγχάνει να αποτυπώσει φυσικά πρόσωπα, πράγμα το οποίο περιπλέκει τις υλοποιήσεις που στοχεύουν στο να παρέχουν εξατομικευμένες υπηρεσίες. Η εισαγωγή δυνατοτήτων αλλαγής χρήστη από την αρχή της υλοποίησης μίας πλατφόρμας, ενώ από τη μία πλευρά μπορεί να φανεί ότι επιλύει κάποια από τα ζητήματα σύνδεσης των δεδομένων εφαρμογών με φυσικά πρόσωπα, ταυτόχρονα περιπλέκει τις υλοποιήσεις και από την πλευρά της πλατφόρμας και ενδέχεται να οδηγήσει στη μερική ικανοποίηση των

απαιτήσεων του συστήματος, αλλά και από την πλευρά του προγραμματιστή εφαρμογών που θα καλείται να λάβει υπόψη αυτή την αυξημένη περιπλοκότητα.

Υπηρεσία webinos (webinos Service)

Μία υπηρεσία webinos είναι μία συλλογή από λειτουργίες και γεγονότα, τα οποία είναι προσβάσιμα από μία εφαρμογή που υποστηρίζει το webinos. Αυτές πάντα εμφανίζονται στον προγραμματιστή ως σετ από συναρτήσεις JavaScript, ανεξάρτητα του λειτουργικού συστήματος ή της πλατφόρμας στην οποία είναι εγκατεστημένη η εφαρμογή.

Υπάρχει το ακόλουθο σύνολο από τύπους υπηρεσιών webinos (78):

1. *Εγγενή API συσκευής*: Αυτά μπορούν να χρησιμοποιηθούν στην ίδια συσκευή που είναι εγκατεστημένη η εφαρμογή, ενώ η υλοποίηση μπορεί να παρέχεται μέσω μίας σύνδεσης JavaScript με τον εγγενή κώδικα, χρησιμοποιώντας τεχνολογίες προσθέτων, όπως το NPAPI (Netscape Plugin Application Programming Interface), ή προγραμματισμένες προεκτάσεις στην μηχανή JavaScript. Η πρόσβαση σε αυτά τα API παρέχεται και διαχειρίζεται από το PEP (Policy Enforcement Point) που ορίζεται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos, εντός του PZP.
2. *APIs με δυνατότητα απομακρυσμένης πρόσβασης σε έξυπνες συσκευές που εκτελούν το webinos*: Πρόκειται για APIs στα οποία δίνεται η δυνατότητα απομακρυσμένης πρόσβασης, χρησιμοποιώντας JSON-RPC. Ένα τέτοιο API παρέχεται μέσω ενός PZP και η πρόσβαση σε αυτό παρέχεται και διαχειρίζεται από το PEP (Policy Enforcement Point) που ορίζεται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos.
3. *APIs με δυνατότητα απομακρυσμένης πρόσβασης σε μη-έξυπνες συσκευές που εκτελούν το webinos*: Παρόμοια με το παραπάνω, πρόκειται για συσκευές, οι οποίες δεν είναι έξυπνα κινητά, ηλεκτρονικοί υπολογιστές ή tablet, αλλά μικρές συσκευές που λειτουργούν ως αισθητήρες, άλλα που μπορούν να λειτουργήσουν ως PZP και ως αποτέλεσμα, μπορούν να ταυτοποιηθούν στην Προσωπική Ζώνη και να επικοινωνήσουν με JSON-RPC.
4. *Πολύ απλές συσκευές με δυνατότητα απομακρυσμένης πρόσβασης*: Παρόμοια με το παραπάνω, όμως η συσκευή είναι ακόμα πιο απλή και δε δύναται να επικοινωνήσει με το webinos απευθείας. Αντ' αυτού, μία ξένια συσκευή παρουσιάζεται ως οδηγός webinos (driver ενός μίνι PZP), το οποίο μπορεί να επικοινωνήσει με εγγενή

κώδικα με την πολύ απλή συσκευή και να μετακωδικοποιήσει τις δικατευθυντικές επικοινωνίες σε πρωτόκολλα του webinos.

5. *APIs με δυνατότητα απομακρυσμένης πρόσβασης σε διακομιστή*: Πρόκειται για υπηρεσίες ιστού (web services), οι οποίες είναι προσβάσιμες σε JavaScript, μέσω JSON-RPC. Αυτές φιλοξενούνται σε ένα Κόμβο Προσωπικής Ζώνης (PZH), ενώ πρόσβαση σε αυτά τα API παρέχεται και διαχειρίζεται από το PEP (Policy Enforcement Point) που ορίζεται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos, εντός του PZH.
6. *APIs που φιλοξενούνται από εφαρμογές*: Μία πλήρης εφαρμογή, η οποία φιλοξενείται μέσω του webinos χρόνου εκτέλεσης (webinos Runtime, WRT), μπορεί να εμφανίζει εξωτερικές υπηρεσίες, υπό τη μορφή APIs σε JavaScript, στις οποίες μπορεί να έχουν πρόσβαση άλλες εφαρμογές.

Μία υπηρεσία webinos θα πρέπει να λαμβάνει υπόψη τις ακόλουθες προδιαγραφές του webinos:

- *Ανακάλυψη*: Μία υπηρεσία θα πρέπει να είναι ανακαλύψιμη και να μπορεί να περιγράψει τον εαυτό της στην εφαρμογή, σε συμφωνία με την προδιαγραφή ανακάλυψης
- *Ανταλλαγή μηνυμάτων*: Μία υπηρεσία θα πρέπει να είναι σε θέση να λάβει και να απαντήσει σε εισερχόμενα μηνύματα RPC

3.3.4. Διεπαφές Προγραμματισμού Εφαρμογών του webinos

Για την επίτευξη του στόχου της ανάπτυξης εφαρμογών επίγνωσης πλαισίου, που θα μπορούν να εκτελούνται τόσο σε διαφορετικές πλατφόρμες, όσο και σε διαφορετικές συσκευές, είναι απαραίτητη η πρόσβαση του προγραμματιστή σε ένα κατανεμημένο περιβάλλον από Διεπαφές Προγραμματισμού Εφαρμογών (APIs).

Οι εφαρμογές που εκτελούνται μέσα σε μία Προσωπική Ζώνη του webinos μπορούν, παράλληλα, να μοιράζονται την κατάσταση της ίδιας εφαρμογής η οποία πιθανώς να εκτελείται σε μία άλλη συσκευή του ιδίου χρήστη, με τον ίδιο ακριβώς τρόπο που ανταλλάσσουν πληροφορίες εντός της Προσωπικής Ζώνης. Επιπρόσθετα, η χρήση των κλήσεων απομακρυσμένης διαδικασίας (RPCs), που πραγματοποιούνται

από το WRT στο εκτιθέμενο API άλλων συσκευών εντός της προσωπικής ζώνης, θα επιτρέπει στους προγραμματιστές να αυξάνουν το εύρος των λειτουργιών της εφαρμογής τους.

Αφού ερευνήθηκε σειρά σεναρίων, περιπτώσεων χρήσης και απαιτήσεων από την ομάδα του webinos, συμφωνήθηκε η αρχιτεκτονική του webinos να απαιτεί την ύπαρξη μίας ομάδας API συσκευών. Αρχικά χρησιμοποιήθηκαν πολλά API τα οποία ήδη υπήρχαν και μπορούσαν να καλύψουν κάποιες ανάγκες. Στη συνέχεια όμως το webinos προχώρησε στον σχεδιασμό νέων API τα οποία θα κάλυπταν συγκεκριμένες απαιτήσεις. Ο νέος σχεδιασμός έγινε προσπαθώντας πάντα να μην υπάρχει μεγάλη διαφοροποίηση από τα ήδη υπάρχοντα API.

Τα API που σχεδιάστηκαν και παρέχονται από το webinos είναι:

WEBINOS CORE INTERFACE

Προσδιορίζει την κοινή διεπαφή μέσω της οποίας μπορεί ο χρήστης να έχει πρόσβαση σε όλα τα API του webinos και σε όλες τις πληροφορίες της προσωπικής ζώνης. Αυτό το API ορίζει:

- Την κεντρική διεπαφή του webinos, μέρος του παγκόσμιου αντικειμένου παραθύρου.
- Τις πληροφορίες της προσωπικής ζώνης, όπως η κατάσταση σύνδεσης, το όνομα του προσωπικού κόμβου, το φιλικό όνομα του πληρεξουσίου της Προσωπικής Ζώνης και το αναγνωριστικό εφαρμογών.

APPLAUNCHER API

Επιτρέπει την ενεργοποίηση των εφαρμογών webinos τοπικά στην συσκευή.

Οι δυνατότητες που παρέχονται στον χρήστη είναι:

- Ενεργοποίηση/απενεργοποίηση των εγκατεστημένων εφαρμογών webinos
- Ενεργοποίηση/απενεργοποίηση κοινοποίησης στους χρήστες όταν μία εφαρμογή webinos προσπαθεί να ενεργοποιήσει μία άλλη εφαρμογή.
- Ενεργοποίηση/απενεργοποίηση της δυνατότητας της εφαρμογής να ανιχνεύει εγκατεστημένες εφαρμογές.

Η εκτέλεση του API παρέχει μηχανισμούς στις εφαρμογές webinos ώστε να ελέγχουν αν μια συγκεκριμένη εφαρμογή webinos έχει εγκατασταθεί στην συσκευή.

APPSTATE SYNCHRONISATION API

Αποτελεί μία διεπαφή για ενεργοποίηση και διαχείριση συγχρονισμού εφαρμογών. Η διεπαφή παρέχει ένα σύνολο λειτουργιών για την διαχείριση των κοινόχρηστων συγχρονισμένων αντικειμένων. Απλοποιεί από την τρέχουσα εφαρμογή μηχανισμούς ανταλλαγής δεδομένων και διευκολύνει την ανάπτυξη διεσπαρμένων εφαρμογών, μειώνοντας την πολυπλοκότητα σε απλές λειτουργίες ανάγνωσης και εγγραφής των ιδιοτήτων των αντικειμένων. Η διεπαφή περιλαμβάνει τη δημιουργία και αναζήτηση αντικειμένων στην μορφολογία ανταλλαγής δεδομένων JSON και εγγραφή ή διαγραφή για αλλαγή κατάστασης. Οι αλλαγές των αντικειμένων που δημιουργούνται από αυτή την διεπαφή ανιχνεύονται και συγχρονίζονται αυτόματα σε όλους τους συμμετέχοντες, π.χ. η ρύθμιση των ιδιοτήτων ή η αλλαγή των τιμών θα επηρεάζει όλα τα αντίγραφα των αντικειμένων. Ο τρόπος συγχρονισμού μπορεί να ρυθμιστεί ώστε οι συμμετέχοντες να είναι ξεχωριστές οντότητες της ίδιας εφαρμογής που εκτελείται στις διαφορετικές συσκευές του χρήστη (π.χ. διαμοιρασμένες εφαρμογές σε διάφορες συσκευές). Με περαιτέρω ρύθμιση, ένα αντικείμενο μπορεί να συγχρονιστεί παράλληλα για πολλούς διαφορετικούς χρήστες ή εφαρμογές (π.χ. συνομιλία ανάμεσα σε εφαρμογές).

AUTHENTICATION API

Ο ρόλος του API ταυτοποίησης είναι να παρέχει στις εφαρμογές πληροφορίες για το εάν ο τρέχων χρήστης έχει ταυτοποιηθεί καθώς και τη λειτουργία του να αιτείται επαναταυτοποίηση, εάν χρειαστεί, κατά τη διαρκεί λειτουργίας της εφαρμογής. Σκόπιμα δεν αποκαλύπτει πληροφορίες ταυτότητας του χρήστη, αλλά μπορεί να παρέχει λεπτομέρειες για την μέθοδο ταυτοποίησης, οι οποίες μπορούν να αποκαλύψουν πληροφορίες για την συσκευή.

CONTACTS API

Το API επαφών καθορίζει τις υψηλού επιπέδου διεπαφές που χρειάζονται για να υπάρξει πρόσβαση στην ενοποιημένα ατζέντα επαφών ενός χρήστη. Για το σκοπό αυτό χρησιμοποιεί δύο διαφορετικές διεπαφές: Μία διεπαφή επαφών που παρέχει τη μέθοδο πρόσβασης στην ατζέντα και μία διεπαφή επαφών η οποία συλλέγει τις ξεχωριστές πληροφορίες κάθε επαφής, οι οποίες μπορούν να επιστραφούν ύστερα από μία πετυχημένη λειτουργία ανάγνωσης.

CONTEXT API

Το API πλαισίου καθορίζει τις διεπαφές υψηλού επιπέδου που είναι απαραίτητες για να υπάρξει πρόσβαση στα δεδομένα πλαισίου του χρήστη. Ο χρήστης πρέπει να ενεργοποιήσει τον διαχειριστή πλαισίου στην συσκευή του για να μπορεί να παρακολουθήσει τα αντικείμενα πλαισίου του. Το API αυτό υποστηρίζει δύο βασικούς τρόπους πρόσβασης:

- Μέσω ερώτησης στον αποθηκευτικό χώρο δεδομένων πλαισίου και ανακτώντας τα δεδομένα από τα αποτελέσματα της ερώτησης
- Με εγγραφή ώστε να λαμβάνονται αμέσως δεδομένα πλαισίου μόλις ένα γεγονός πραγματοποιηθεί

Το API παρέχει και την δυνατότητα να προγραμματίσει κλήσεις API, με σκοπό την ανανέωση των διαθέσιμων δεδομένων πλαισίου, ακόμα και αν η εφαρμογή δεν εκτελείται.

DEVICE INTERACTION API

Το μοντέλο αυτό παρέχει ένα μηχανισμό για την αλληλεπίδραση με τον τελικό χρήστη, μέσω ιδιοτήτων όπως:

- Δόνηση
- Ηχητική ειδοποίηση
- Φωτεινή ένδειξη
- Ενδείξεις φόντου

DEVICE STATUS API

Αυτό το API χρησιμοποιεί ένα μοντέλο δένδρου για να επιτρέπει στους προγραμματιστές να έχουν πρόσβαση στα διάφορα κομμάτια πληροφορίας σε μία συσκευή. Σε αυτό το μοντέλο χρησιμοποιεί τους όρους: Αντικείμενο, Εξάρτημα και Ιδιότητα. Όπου ένα αντικείμενο μπορεί να περιέχει ένα ή περισσότερα εξαρτήματα, και ένα εξάρτημα μία ή περισσότερες ιδιότητες. Ακόμα, γίνεται χρήση και δύο ειδικών εξαρτημάτων: "_default" and "_active", τα οποία μπορούν να χρησιμοποιηθούν από τον προγραμματιστή ως πληρεξούσια στο API.

DISCOVERY API

Χρησιμοποιείται όχι μόνο για την ανακάλυψη τοπικών υπηρεσιών, αλλά και για την ενεργοποίηση της ανίχνευσης απομακρυσμένων υπηρεσιών. Συγκεκριμένα ενεργοποιεί την δυνατότητα ανίχνευσης εκτεθειμένων υπηρεσιών:

- Στη συσκευή
- Σε οντότητες απευθείας συνδεόμενες στη συσκευή
- Σε οντότητες διαθέσιμες στο ίδιο τοπικό δίκτυο
- Σε ασφαλείς υπηρεσίες καταχωρημένες σε μία Προσωπική Ζώνη

Αυτό το επιτυγχάνει με την χρήση ενός *αντικειμένου υπηρεσιών* (service object) και την δημιουργία ενός ασφαλούς διαύλου επικοινωνίας, εφόσον έχει ήδη εγκατασταθεί η εφαρμογή και ο χρήστης της συσκευής είναι ταυτοποιημένος και εξουσιοδοτημένος να χρησιμοποιήσει το API.

THE GENERIC ACTUATOR API

Το API αυτό παρέχει στις εφαρμογές ένα API για να ελέγχουν τους ενεργοποιητές της ίδιας της συσκευής, συνδεδεμένους με τη συσκευή ή μίας άλλης συσκευής. Το API δεν γνωρίζει τις υποκείμενες μεθόδους για να ανιχνεύει και να επικοινωνεί με τους ενεργοποιητές, για το λόγο αυτό θα πρέπει να χρησιμοποιείται σε συνδυασμό με υπηρεσίες ανίχνευσης και σύνδεσης. Οι υπηρεσίες ενός ενεργοποιητή μπορούν να εντοπιστούν στην Προσωπική Ζώνη του χρήστη ή να μοιραστούν με το υπάρχον δίκτυο. Αυτή τη στιγμή αρκετοί διαφορετικοί ενεργοποιητές ορίζονται, αλλά το API μπορεί εύκολα να επεκταθεί με επιπλέον τύπους.

Αυτό είναι ένα πειραματικό API και θέματα ασφάλειας και ιδιωτικότητας δεν έχουν διευκρινιστεί πλήρως. Αν γίνει δυνατή η πρόσβαση σε ενεργοποιητές ιδιωτικού απορρήτου ή ασφάλειας, ο πράκτορας του χρήστη πρέπει, είτε να πάρει άδεια πρόσβασης μέσω της διεπαφής του χρήστη, είτε να ελέγχει την πρόσβαση μέσω μίας προσυμφωνημένης σχέσης εμπιστοσύνης με τους χρήστες.

App2App MESSAGING API

Το API αυτό καθορίζει τις κατάλληλες διεπαφές για την δημιουργία, αποστολή και λήψη μηνυμάτων ανάμεσα σε εφαρμογές στο webinos. Παρέχει γενικά αρχέτυπα μηνυμάτων τα οποία μπορούν να χρησιμοποιηθούν σε διαφορετικά σενάρια

εφαρμογών. Τα μηνύματα είναι έμμεσα, το οποίο σημαίνει ότι οι εφαρμογές δεν απευθύνονται η μία στην άλλη. Αντί αυτού, χρησιμοποιούν ένα κανάλι επικοινωνίας για να κατευθύνουν τα μηνύματα στις συνδεδεμένες εφαρμογές, οι οποίες χρησιμοποιούν ένα μοναδικό όνομα ως κλειδί για την εύρεση και σύνδεση με το κανάλι. Το συγκεκριμένο API μπορεί να χρησιμοποιηθεί και από τρίτους προγραμματιστές εφαρμογών, οι οποίοι εκμεταλλευόμενοι τις ιδιότητες που παρέχει το webinos, με το σύστημα διαχείρισης μηνυμάτων και το υποκείμενο δικτυακό μοντέλο, μπορούν να παρέχουν προσαρμοσμένα, με βάση τα μηνύματα, πρωτόκολλα.

MESSAGING API

Το API μηνυμάτων δίνει πρόσβαση στις παρακάτω δυνατότητες:

- Αποστολή, μηνυμάτων μέσω διαφορετικών τεχνολογιών (SMS, MMS, IM, email)
- Αναζήτηση μηνυμάτων σε διαφορετικούς φακέλους
- Εγγραφή για ειδοποιήσεις ενόψει προσεχών γεγονότων μηνυμάτων.

Το API αυτό είναι ανάγνωσης μόνο και δεν επιτρέπει τη διαχείριση μηνυμάτων ή φακέλων.

MEDIACONTENT API

Αυτό το API παρέχει λειτουργίες για την ανίχνευση περιεχομένου πολυμέσων (όπως βίντεο, εικόνες, μουσική, κλπ) που είναι διαθέσιμα στην συσκευή. Είναι δυνατή και η αναζήτηση συγκεκριμένων αρχείων πολυμέσων με την χρήση φίλτρων. Επίσης το API αυτό υποστηρίζει την ρύθμισή ορισμένων χαρακτηριστικών πολυμέσων. Στο webinos εμφανίζεται ελαφρώς διαφοροποιημένο, με την προσθήκη μιας μεθόδου για την ακύρωση ασύγχρονων λειτουργιών.

NAVIGATION API

Το API πλοήγησης του webinos παρέχει έναν μηχανισμό αλληλεπίδρασης με εγκατεστημένα λογισμικά πλοήγησης. Είναι υποπροϊόν του API οχήματος, καθώς καθορίζει τις λειτουργίες αλληλεπίδρασης με υπηρεσίες δορυφορικής πλοήγησης. Ακόμα, χρησιμοποιεί μια μέθοδο παροχής σημείων ενδιαφέροντος στην υπηρεσία παρακολουθώντας μέσω αυτών αν η πλοήγηση παραμένει ενεργή.

NFC API

Το Near Field Communication (NFC) είναι ένα διεθνές πρότυπο (ISO/IEC 18092) το οποίο καθορίζει μία διεπαφή και ένα πρωτόκολλο για απλή ασύρματη διασύνδεση από ζεύγη συσκευών που βρίσκονται σε κοντινή απόσταση και λειτουργούν στα 13.56 MHz. Υπάρχουν τρεις ομάδες από πιθανά σενάρια εφαρμογών για NFC: Το πρώτο είναι να τοποθετείται μία συσκευή κοντά σε μια ασύρματη ετικέτα με σκοπό την ανταλλαγή ψηφιακών πληροφοριών ή δεδομένων. Το δεύτερο είναι να τοποθετούνται δύο συσκευές κοντά η μία στην άλλη για να γίνει μεταξύ τους ανταλλαγή δεδομένων. Η τρίτη είναι η διεκπεραίωση πληρωμών μέσω κινητών τηλεφώνων, κρατώντας τα κοντά σε σημεία πώλησης, αντί της χρήσης κάποιου είδους κάρτας.

PAYMENT API

Αυτό το API παρέχει γενικές λειτουργίες ηλεκτρονικής αγοράς με σκοπό να είναι δυνατή η πληρωμή εντός της εφαρμογής. Δεν είναι συνδεδεμένο με ένα συγκεκριμένο πάροχο υπηρεσιών ηλεκτρονικών πληρωμών και είναι σχεδιασμένο ώστε να μπορεί να λειτουργήσει με διάφορες υπηρεσίες όπως GSMA, OneAPI, BlueVia, Android Payment API και PayPal.

THE REMOTE UI API

Οποιαδήποτε συσκευή που χρησιμοποιεί το webinos μπορεί να παρέχει πρόσβαση στην διεπαφή του χρήστη της με τον ίδιο τρόπο που μπορεί να παρέχει οποιαδήποτε άλλη υπηρεσία. Η πρόσβαση στην υπηρεσία διέπεται από την πολιτική των ρυθμίσεων και στην τοπική και στην απομακρυσμένη συσκευή. Δεν δημιουργείται απευθείας πρόσβαση σε απομακρυσμένα αντικείμενα. Βασικές τιμές από το απομακρυσμένο DOM μπορούν να συλλεχθούν, αλλά όχι αντικείμενα.

SECURE ELEMENT API

Το API αυτό επιτρέπει την επικοινωνία ανάμεσα στην εφαρμογή δικτύου και μιας «έξυπνης» κάρτας ή άλλων ασφαλών στοιχείων με την χρήση του Πρωτοκόλλου Εφαρμογής Μονάδων Δεδομένων (APDU). Το APDU είναι ένα σύντομο μήνυμα που αναπαρίσταται από bytes. Τα μηνύματα APDU είναι είτε εντολές είτε αποκρίσεις.

TV CONTROL API

Αυτή η διεπαφή παρέχει τα μέσα για να αποκτηθεί μία λίστα πηγών τηλεόρασης, καναλιών κλπ. Το τηλεοπτικό κανάλι μπορεί να προβληθεί σε αντικείμενο HTMLVideoElement. Εναλλακτικά, το API παρέχει τα μέσα για να ελέγχεται η διαχείριση των καναλιών του προϋπάρχοντος υλικού για την προβολή τηλεόρασης, επιτρέποντας να επιλέγεται ένα κανάλι ή να παρακολουθούνται αλλαγές στα κανάλια που δημιουργούνται με διαφορετικό τρόπο. Το αντικείμενο της τηλεόρασης είναι διαθέσιμο στο χώρο του webinos , π.χ. webinos.tv

VEHICLE API

Αυτό το API προβάλλει δεδομένα οχήματος, τα οποία είναι διαθέσιμα σε ένα δίαυλο του οχήματος. Αυτός ο δίαυλος είναι συνήθως το σημείο πρόσβασης από όπου η κεντρική μονάδα συλλέγει τα δεδομένα του οχήματος. Κάποια δεδομένα από άλλους δίαυλους κατευθύνονται στον δίαυλο πάνω από την κεντρική οδό, όπως η ταχύτητα.

WEBINOS WIDGET API

Ορίζει την κοινή διεπαφή μικροεφαρμογής (widget). Βασίζεται στα χαρακτηριστικά του W3C Widget Specifications και συμπληρώνει την W3C Widget Interface.

THE GENERIC SENSOR API

Το API δεν γνωρίζει τις υποκείμενες μεθόδους για να ανιχνεύει και να επικοινωνεί με τους αισθητήρες. Για τον λόγο αυτό θα πρέπει να χρησιμοποιείται σε συνδυασμό με υπηρεσίες ανίχνευσης και σύνδεσης. Οι υπηρεσίες ενός αισθητήρα μπορούν να εντοπιστούν στην Προσωπική Ζώνη του χρήστη ή να μοιραστούν με το υπάρχον δίκτυο.

Το API αποτελείται από δύο διεπαφές:

- Μια διεπαφή αισθητήρα που παρέχει χαρακτηριστικά για τους αισθητήρες και μια μέθοδο για τη διαμόρφωση ενός επιλεγμένου αισθητήρα.
- Ένα επιπέδου 3 DOM γεγονός που παρέχει δεδομένα αισθητήρα.

Αυτή η λίστα με τύπους αισθητήρων που υποστηρίζονται από το API μπορεί εύκολα να επεκταθεί με επιπρόσθετους τύπους αισθητήρων.

THE Web NOTIFICATIONS API

Αυτό το API ειδοποιήσεων βασίζεται στο αντίστοιχο του W3C. Δεν προσδιορίζει ακριβώς πώς το μέσο ενός χρήστη θα προβάλει αυτές τις ειδοποιήσεις, καθώς η καταλληλότερη αναπαράσταση εξαρτάται από την συσκευή στην οποία εκτελείται το webinos. Όταν αναφέρεται αυτό το API σε αναπαράσταση ειδοποιήσεων στην επιφάνεια εργασίας, γενικά εννοείται μία στατική περιοχή εξωτερικά της περιοχής αναπαράστασης της εφαρμογής, αλλά μπορεί να πάρει αρκετές μορφές, στις οποίες συμπεριλαμβάνονται:

- Μία γωνία της οθόνης
- Μία περιοχή μέσα σε έναν προκαθορισμένο χώρο της διεπαφής του συστήματος
- Την κεντρική οθόνη μίας κινητής συσκευής

Δεν ορίζει ακριβώς το τρόπο με τον οποίο το μέσο ενός χρήστη θα προβάλει την ειδοποίηση, και είναι σχεδιασμένο ώστε να είναι ευέλικτο στις διαφορετικές επιλογές αναπαράστασης. Είναι σχεδιασμένο ώστε να είναι συμβατό, όσο αυτό είναι δυνατό, με τις ήδη υπάρχουσες πλατφόρμες και τεχνολογίες ειδοποιήσεων, αλλά και να μην εξαρτάται από αυτές. Εφ' όσον οι κοινές πλατφόρμες δεν παρέχουν την ίδια λειτουργία, αυτό παρέχει ένδειξη για το ποια γεγονότα είναι εγγυημένα και ποια όχι. Εν προκειμένω, οι ενδείξεις όπως περιγράφονται εδώ μπορούν να περιέχουν μόνο κείμενο και εικονίδια. Στο μέλλον, οι ειδοποιήσεις προερχόμενες από περιεχόμενο διαδικτύου θα μπορούν να περιέχουν και οι ίδιες περιεχόμενο του διαδικτύου.

Εν αντιθέσει με την πρωτότυπη έκδοση του W3C, το μοντέλο γεγονότων ειδοποιήσεων είναι αξιόπιστο. Το αντικείμενο ειδοποιήσεων προσφέρει ένα γεγονός πατήματος (click) και οι εφαρμογές μπορούν να αυξήσουν τις λειτουργίες τους ακούγοντας για αυτό το γεγονός.

Τα παρακάτω προϋπάρχοντα API χρησιμοποιούνται από το webinos έτοιμα και μη τροποποιημένα (referenced APIs):

THE W3C CALENDAR MODULE

Αποτελεί το API που παρέχει πρόσβαση στον χρήστη σε υπηρεσίες και εφαρμογές ημερολογίου.

THE W3C DEVICE ORIENTATION – EVENT SPECIFICATION

Ορίζει αρκετά νέα DOM (Document Object Models) τύπου γεγονότα, τα οποία παρέχουν πληροφορίες για τον φυσικό προσανατολισμό και την κίνηση της συσκευής που χρησιμοποιείται.

THE W3C FILE API

Χρησιμοποιείται για την αναπαράσταση αρχείων σε δικτυακές εφαρμογές, όπως για την επιλογή τους και πρόσβαση στα δεδομένα τους.

THE W3C FILE API: WRITER

Εξαρτάται και από άλλα API, όπως το προηγούμενο, και είναι σχεδιασμένο ώστε να χρησιμοποιείται σε συνδυασμό μαζί τους και ορίζει τον τρόπο με τον οποίο πραγματοποιείται εγγραφή σε αρχεία δικτυακών εφαρμογών.

THE W3C FILE API: DIRECTORIES AND SYSTEM

Όπως και το παραπάνω εξαρτάται και χρησιμοποιείται σε συνδυασμό με το W3C File Writer API, το οποίο με τη σειρά του δομείται πάνω στο W3C File API. Το συγκεκριμένο ορίζει τον τρόπο πλοήγησης και ιεραρχίας σε αρχεία δικτυακών εφαρμογών.

THE W3C GEOLOCATION API

Παρέχει πρόσβαση σε πληροφορίες γεωγραφικής τοποθεσίας που σχετίζονται με την ενεργή συσκευή.

THE W3C MEDIA CAPTURE AND STREAMS API

Το συγκεκριμένο API παρέχει πρόσβαση του χρήστη σε υπηρεσίες της ενεργοποιημένης συσκευής που σχετίζονται με τη λήψη στιγμιότυπου φωτογραφίας ή βίντεο και εν γένει όλες τις δυνατότητες μιας συσκευής να συλλέξει οπτικοακουστικό υλικό.

THE W3C WebRTC API

Επιτρέπει την δημιουργία διομότιμης σύνδεσης (peer-to-peer) ανάμεσα σε διαφορετικά προγράμματα περιήγησης, καθώς και την μετάδοση δεδομένων μέσω αυτών των συνδέσεων.

Τα παραπάνω API χρησιμοποιούνται από τις εφαρμογές webinos μέσω των παρακάτω διαδικασιών του webinos:

Η εφαρμογή που εκτελείται στην πλατφόρμα του webinos έχει πρόσβαση στα API μέσω ενός JavaScript δεσμού. Παρόλα αυτά, αυτό δεν σημαίνει ότι υπάρχει άμεση πρόσβαση στις υποκείμενες λειτουργίες της συσκευής, αλλά ότι η επικοινωνία γίνεται

μέσω μίας κλήσης JSON RPC στο αντίστοιχο PZP. Το PZP έχει εφαρμοστεί ως κώδικας σε node.js, έχοντας έτσι τη δυνατότητα παροχής πρόσβασης στα χαρακτηριστικά και στις λειτουργίες της συσκευής. Αυτή η αρχιτεκτονική προσέγγιση παρουσιάζει τα παρακάτω πλεονεκτήματα:

- Όλες οι κλήσεις API γίνονται μέσω μίας διεπαφής, η οποία μπορεί να αποτελέσει σημείο όπου θα επιβάλλεται η αντίστοιχη πολιτική από τα πλαίσια πολιτικής και ασφάλειας
- Ορίζεται μία RPC διασύνδεση για όλες τις λειτουργίες των API. Έτσι, η μέθοδος πρόσβασης στα API από απομακρυσμένες συσκευές δεν διαφέρει σημαντικά από την αντίστοιχη των τοπικών API, ενισχύοντας την διαδραστικότητα και την διαλειτουργικότητα του συστήματος
- Εφόσον ο κώδικας της συσκευής εντοπίζεται κυρίως στο PZP και η επικοινωνία σε αυτό γίνεται μέσω γνωστών πρωτοκόλλων, η ανάγκη για επέκταση του προγράμματος περιήγησης μειώνεται, ενισχύοντας έτσι την ευελιξία του περιβάλλοντος του webinos

3.3.5. Αρχιτεκτονική Ασφαλείας του webinos

Δύο από τους πρωτεύοντες ρόλους του webinos είναι η ιδιωτικότητα και η ασφάλεια. Υπάρχει μία σειρά από ανησυχίες που αφορούν σε ζητήματα ιδιωτικότητας. Το webinos διατηρεί προσωπικές πληροφορίες και υποστηρίζει την πρόσβαση σε ευαίσθητα API, όπως της γεωγραφικής τοποθεσίας, τα ημερολόγια και τις επαφές. Ως εκ τούτου, ένα ολοκληρωμένο σύστημα ελέγχου πρόσβασης και έξυπνα και απλά στοιχεία ελέγχου της ιδιωτικότητας είναι απαραίτητα. Από άποψη ασφάλειας, το webinos μπορεί να θεωρηθεί ένας προσοδοφόρος στόχος επίθεσης, λόγω της υποστήριξης που παρέχει για ένα API πληρωμών και την αποθήκευση των διαπιστευτηρίων χρήστη για πρόσβαση σε διαδικτυακές υπηρεσίες (27). Το webinos μπορεί επίσης να χρησιμοποιηθεί σε ένα εταιρικό περιβάλλον, πράγμα που σημαίνει ότι τα πολύτιμα δικαιώματα πνευματικής ιδιοκτησίας της εταιρείας θα μπορούσε να είναι παρόντα σε μια συσκευή webinos. Ως αποτέλεσμα, το webinos πρέπει να προστατεύει την ακεραιότητα της πλατφόρμας και να παρέχει ασφαλή αποθήκευση των δεδομένων του χρήστη και των εφαρμογών.

Επιπρόσθετα, το webinos στοχεύει στο να αποτελεί διάχυτη υπολογιστική πλατφόρμα, όπως περίπου και τα προγράμματα περιήγησης. Αυτό σημαίνει ότι μία επιτυχημένη επίθεση στο σύστημα θα μπορούσε να επηρεάσει όλες τις συσκευές που τη χρησιμοποιούν. Αυτό, σε συνδυασμό με την αναπόφευκτη περιπλοκότητα αυτών των συστημάτων είναι γνωστό ως το «πρόβλημα του ενδιάμεσου λογισμικού» (21). Ένα ακόμα πρόβλημα, το οποίο χρήζει πρωτότυπης υλοποίησης στοιχείων ελέγχου ιδιωτικότητας είναι η διαχείριση των δεδομένων πλαισίου. Οι πληροφορίες για την τοποθεσία του χρήστη, τις δραστηριότητές του, τις κοντινές συσκευές και τους κοινωνικούς του γράφους είναι ευαίσθητα δεδομένα και πρέπει να προστατεύονται. Αυτό μπορεί, από τη μία, να φέρει πολύ μεγάλα οφέλη, προσαρμόζοντας την εμπειρία του χρήστη βάσει αυτών των δεδομένων, αλλά μπορεί ταυτόχρονα να δημιουργήσει συνδέσεις μεταξύ φίλων και τοπικών υπηρεσιών (29). Παρόλα αυτά, υπάρχει μία σύγκρουση μεταξύ της χρήσης και της αποθήκευσης δεδομένων πλαισίου και της προστασίας του ιδιωτικού απορρήτου. Είναι σημαντικό να απαντηθεί το πώς είναι εφικτό να αξιοποιηθούν τα οφέλη του ενός, χωρίς να χαθεί το δεύτερο.

Το webinos στοχεύει στο να αντιμετωπίσει αυτά τα ζητήματα μέσω δύο δραστηριοτήτων: την μελέτη των προσδοκιών των χρηστών ως προς την ασφάλεια και την ιδιωτικότητα και την ανάπτυξη μίας ολοκληρωμένης αρχιτεκτονικής ασφαλείας.

Η μελέτη των προσδοκιών των χρηστών ως προς την ασφάλεια και την ιδιωτικότητα περιείχε την εκτέλεση διαφόρων δραστηριοτήτων ανάλυσης ασφάλειας και χρηστικότητας. Διάφορες «περσόνες» (60) δημιουργήθηκαν για να περιγράψουν την αρχετυπική συμπεριφορά που ενδέχεται να έχουν οι χρήστες του webinos. Αυτές οι περσόνες προέκυψαν από μία ποικιλία διαφορετικών πηγών δεδομένων και δομήθηκαν χρησιμοποιώντας σενάρια από διαφορετικά κοινωνικά πλαίσια, στα οποία ενδέχεται να χρησιμοποιούνται εφαρμογές του webinos. Αυτές οι περσόνες συμβολίζουν τελικούς χρήστες, προγραμματιστές εφαρμογών του webinos, καθώς και άτομα, τα οποία ενώ δεν αποτελούν τελικούς χρήστες του webinos, ενδέχεται να επηρεαστούν από αυτό. Αυτές οι προοπτικές είναι σημαντικές, επειδή, αντίθετα από τις παραδοσιακές πλατφόρμες ενδιάμεσου λογισμικού, οι οποίες συνήθως είναι προσβάσιμες μόνο από τους προγραμματιστές εφαρμογών, οι τελικοί χρήστες αναμένεται να διεκδικούν τις δικές τους προσδοκίες ασφάλειας και ιδιωτικότητας όταν διαχειρίζονται τις προσωπικές τους ζώνες.

Εκτός από το να ενημερώνουν τις θεωρήσεις πάνω στην χρηστικότητα του webinos, αναπτύχθηκαν περσόνες για να αναπαραστήσουν τους διαφορετικούς τρόπους με τους οποίους μπορεί να πραγματοποιηθεί μία επίθεση προς το webinos. Αυτές οι περσόνες επίθεσης βασίστηκαν σε δεδομένα ανάλυσης απειλών ανοιχτού κώδικα (20, 52) και δημοσιευμένες ιστορικές περιπτώσεις καταδικασμένων χάκερ (6). Για να επιβεβαιωθούν οι περσόνες επίθεσης, αναπτύχθηκαν περιπτώσεις κατάχρησης (69), με σκοπό να διερευνηθεί το κατά πόσο αυτοί οι επιτιθέμενοι θα μπορούσαν να έχουν τα κίνητρα και τις δυνατότητες να εκμεταλλευτούν διάφορες πτυχές της πλατφόρμας webinos.

Η αρχιτεκτονική ασφαλείας του webinos μπορεί να χωριστεί στις ακόλουθες επικαλυπτόμενες περιοχές:

Επικοινωνία Προσωπικής Ζώνης

Κάθε κόμβος Προσωπικής Ζώνης (Personal Zone Hub) μπορεί να λειτουργήσει ως αρχή έκδοσης πιστοποιητικών για των ζώνη, εκδίδοντας πιστοποιητικά για κάθε ένα πληρεξούσιο σύστημα. Η επικοινωνία μεταξύ οποιωνδήποτε συσκευών μέσα στην Προσωπική Ζώνη μπορεί να αυθεντικοποιείται και να κρυπτογραφείται αμοιβαία, χρησιμοποιώντας τυπική ασφάλεια επιπέδου μεταφοράς (Transport Layer Security, TLS). Ο κόμβος της Προσωπικής Ζώνης διασφαλίζει ότι κάθε συσκευή έχει ένα αντίγραφο όλων των γνωστών πιστοποιητικών, έτσι ώστε να υποστηρίζεται η διομότιμη επικοινωνία. Σε περίπτωση που μία συσκευή χαθεί ή κλαπεί, ο χρήστης μπορεί, έχοντας πρόσβαση στον κόμβο του, να ανακαλέσει τα πιστοποιητικά της συσκευής αυτής και να την αφαιρέσει από την Προσωπική Ζώνη.

Επικοινωνία μεταξύ Προσωπικών Ζωνών

Οι συσκευές με webinos σε μία Προσωπική Ζώνη μπορούν να επικοινωνήσουν με αυτές σε άλλες ζώνες, μέσω ενός συνδυασμού μηχανισμών ανταλλαγής πιστοποιητικών και αντιστοίχισης συσκευών, οι οποίοι εξασφαλίζουν ότι οι οντότητες έχουν ταυτοποιηθεί με ασφάλεια, χωρίς να χρειάζεται το webinos να στηρίζεται σε έμπιστους τρίτους φορείς.

3.3.6. Πολιτικές Ελέγχου Πρόσβασης για εφαρμογές

Οι εφαρμογές του webinos, σε συμμόρφωση με τα παρόντα πρότυπα, απαιτούν ρητά δικαιώματα για να έχουν πρόσβαση στους πόρους των συσκευών, όμως αυτό επεκτείνεται ενσωματώνοντας επιπλέον πολιτικές προστασίας προσωπικών δεδομένων, οι οποίες διέπουν τον τρόπο με τον οποίο οι εφαρμογές υπόσχονται ότι θα χρησιμοποιήσουν τα δεδομένα για τα οποία ζητούν πρόσβαση. Αφού δοθούν τα δικαιώματα, οι ανάλογοι κανόνες ελέγχου πρόσβασης δημιουργούνται αυτόματα σε XACML. Αυτοί μπορούν τότε να επεκταθούν ώστε να περιέχουν κανόνες που αφορούν στον διαμοιρασμό δεδομένων μεταξύ εφαρμογών, χρηστών και συσκευών. Το webinos συγχρονίζει τις πολιτικές αυτές μεταξύ όλων των συσκευών μέσα στην Προσωπική Ζώνη, έτσι ώστε οι χρήστες να μη χρειάζεται να παίρνουν συνεχώς τις ίδιες αποφάσεις.

3.3.7. Ασφαλής Αποθήκευση

Διάφορα είδη αποθηκευμένων δεδομένων του webinos απαιτούν ασφάλεια, μεταξύ των οποίων οι Πολιτικές Ελέγχου Πρόσβασης, τα διαπιστευτήρια, τα κλειδιά και τα δεδομένα πλαισίου. Το webinos, εκμεταλλευόμενο τα τοπικά χαρακτηριστικά των υποστηριζόμενων συσκευών, κωδικοποιεί τα δεδομένα, ελαχιστοποιώντας το κόστος σε περίπτωση που μία συσκευή κλαπεί ή χαθεί (79).

3.4. Node.js

Το Node.js, το οποίο επίσης ονομάζεται Node, είναι ένα περιβάλλον ανάπτυξης λογισμικού (κυρίως διακομιστών) για JavaScript, πάνω στο οποίο είναι χτισμένο το webinos. Βασίζεται στην μηχανή εκτέλεσης JavaScript της Google, V8. Η V8 και το Node υλοποιούνται κατά βάση σε C και C++, δίνοντας έμφαση στην απόδοση και στη χαμηλή κατανάλωση μνήμης. Ενώ, όμως, σκοπός της V8 είναι να υποστηρίζει κυρίως την εκτέλεση JavaScript σε ένα πρόγραμμα περιήγησης, όπως του Google Chrome, το Node στοχεύει στην υποστήριξη για διεργασίες μακράς εκτέλεσης διακομιστών(73). Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών.

Σε αντίθεση με τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων, μία διεργασία Node δεν στηρίζεται στην πολυνηματικότητα, αλλά σε ένα

μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου. Μία διεργασία διακομιστή του Node μπορεί να θεωρηθεί ως ένας μονο-νηματικός δαίμονας που ενσωματώνει τη μηχανή εκτέλεσης JavaScript με τρόπο τέτοιο, ώστε να υποστηρίζει παραμετροποίηση. Αυτό είναι διαφορετικό από τα περισσότερα συστήματα διαχείρισης συμβάντων για άλλες γλώσσες προγραμματισμού, τα οποία είναι σε μορφή βιβλιοθηκών. Το Node υποστηρίζει τη διαχείριση γεγονότων στο επίπεδο της γλώσσας προγραμματισμού.

Η JavaScript είναι μία εξαίρετη επιλογή για αυτή την προσέγγιση, καθώς υποστηρίζει επανακλήσεις από γεγονότα. Για παράδειγμα, όταν ένα πρόγραμμα περιηγητή φορτώνει ολόκληρο ένα έγγραφο, ή ο χρήστης κάνει κλικ σε ένα κουμπί, ή ολοκληρώνεται ένα αίτημα Ajax, ένα γεγονός ενεργοποιεί μία επανάκληση. Η λειτουργική φύση της JavaScript κάνει πολύ εύκολη τη δημιουργία ανώνυμων αντικειμένων συναρτήσεων τα οποία μπορούν να εγγραφούν ως χειριστές γεγονότων (event handlers).

Οι προγραμματιστές εφαρμογών που ασχολούνται με πολλαπλές πηγές εισόδου και εξόδου, όπως διακομιστές δικτύου, οι οποίοι χειρίζονται πολλαπλές συνδέσεις με συστήματα-πελάτες, χρησιμοποιούν για πολλά χρόνια ήδη πολυνηματικές τεχνικές προγραμματισμού. Αυτές οι τεχνικές έγιναν δημοφιλείς επειδή επέτρεπαν στους προγραμματιστές να διαιρέσουν τις εφαρμογές τους σε δραστηριότητες που συνεργάζονται μεταξύ τους και εκτελούνται ταυτόχρονα. Αυτό όχι μόνο έκανε την προγραμματιστική λογική ευκολότερη στη κατανόηση, στην υλοποίηση και στη συντήρηση, αλλά επίσης, επέτρεψε την ταχύτερη και πιο αποδοτική εκτέλεση της.

Η προσέγγιση εισόδων/εξόδων του Node είναι αυστηρή. Οι ασύγχρονες αλληλεπιδράσεις δεν είναι η εξαίρεση, αλλά ο κανόνας. Κάθε διεργασία εισόδου/εξόδου την χειρίζονται συναρτήσεις υψηλού επιπέδου, δηλαδή συναρτήσεις που παίρνουν συναρτήσεις ως παραμέτρους, που καθορίζουν τι πρέπει να εκτελεστεί όταν υπάρχει κάτι για να εκτελεστεί. Μόνο σε σπάνιες περιπτώσεις έχει χρειαστεί προγραμματιστές στο Node να δημιουργήσουν συναρτήσεις για ευκολία, οι οποίες να εκτελούνται σύγχρονα. Ένα τέτοιο παράδειγμα είναι συναρτήσεις για μετακίνηση ή μετονομασία αρχείων. Σε γενικές γραμμές, όταν ενεργοποιούνται διεργασίες οι οποίες ενδέχεται να χρειάζονται τη χρήση δικτύου ή διαχείρισης αρχείων, ο έλεγχος επιστρέφεται απευθείας στον καλούντα. Όταν συμβαίνει κάτι ενδιαφέρον, για παράδειγμα αν γίνουν διαθέσιμα δεδομένα από έναν υποδοχέα δικτύου (network

socket), ή μία ροή δεδομένων εξόδου γίνεται έτοιμη για εγγραφή, ή ένα σφάλμα συμβαίνει, η κατάλληλη συνάρτηση επανάκλησης καλείται.

Το Node είναι ένα από τα πιο δημοφιλή πλαίσια και περιβάλλοντα που υποστηρίζουν την ανάπτυξη εφαρμογών JavaScript για διακομιστές. Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα βιβλιοθηκών ειδικά για το Node, ή που είναι συμβατές με αυτό. Μεταξύ αυτών, είναι εργαλεία, όπως το node-mysql, το node-couchdb και η tingoDB. Αυτά παίζουν έναν σημαντικό ρόλο, υποστηρίζοντας την ασύγχρονη αλληλεπίδραση με σχεσιακές ή NoSQL βάσεις. Πολλά πλαίσια παρέχουν ένα πλήρες περιβάλλον ανάπτυξης διαδικτυακών εφαρμογών, όπως το Connect και το Express, οι οποίες είναι συγκρίσιμες με τα πρωτόκολλα Rack και Rails στον κόσμο της Ruby. Ο διαχειριστής πακέτων του Node, το npm, επιτρέπει την εγκατάσταση βιβλιοθηκών και των εξαρτήσεών τους. Επιπλέον, πολλές βιβλιοθήκες είναι διαθέσιμες για την εκτέλεση JavaScript από την πλευρά του προγράμματος-πελάτη, οι οποίες συμμορφώνονται με το σύστημα λειτουργικών μονάδων CommonJS και οι οποίες λειτουργούν και με το Node.

Με δεδομένο ότι στις περισσότερες περιπτώσεις ανάπτυξης διαδικτυακών εφαρμογών η γνώση της JavaScript είναι προαπαιτούμενο για την ανάπτυξη προχωρημένων αλληλεπιδράσεων διεπαφών χρήστη, η επιλογή χρήσης μίας προγραμματιστικής γλώσσας για όλα είναι πειρασμός. Η αρχιτεκτονική του Node.js διευκολύνει τη χρήση μίας πολύ εκφραστικής, λειτουργικής γλώσσας προγραμματισμού για ανάπτυξη στον διακομιστή, χωρίς να θυσιάζει τις επιδόσεις και να αγνοεί την επικρατούσα προγραμματιστική τάση.

3.5.NoSQL – MongoDB – TingoDB

Μία **NoSQL** βάση παρέχει έναν μηχανισμό για την αποθήκευση και ανάκτηση δεδομένων που έχουν μοντελοποίηση διαφορετική από τις σχέσεις πινάκων που υπάρχουν στις σχεσιακές βάσεις. Κίνητρα για την προσέγγιση αυτή περιλαμβάνουν την απλότητα του σχεδιασμού, την οριζόντια κλιμάκωση και τον καλύτερο έλεγχο της διαθεσιμότητας. Οι βάσεις δεδομένων αυτές είναι ιδιαίτερα βελτιστοποιημένοι αποθηκευτικοί χώροι ζευγών κλειδιών-τιμών, οι οποίες προορίζονται κυρίως για ενέργειες απλής ανάκτησης και προσάρτησης δεδομένων, ενώ ένα RDBMS προορίζεται ως αποθηκευτικό μέσο δεδομένων γενικής χρήσης. Έτσι, θα υπάρξουν

κάποιες εργασίες όπου η NoSQL βάση είναι ταχύτερη, ενώ μερικές άλλες όπου μία RDBMS βάση θα είναι ταχύτερη. Οι βάσεις δεδομένων NoSQL απολαμβάνουν μία σημαντική και αυξανόμενη ζήτηση από τη βιομηχανία, ειδικά σε περιπτώσεις μεγάλων δεδομένων (big data) και εφαρμογών ιστού πραγματικού χρόνου. Τα συστήματα NoSQL διαβάζονται και ως «Not Only SQL» με σκοπό να δοθεί έμφαση στο ότι μπορούν να επιτρέψουν την εκτέλεση ερωτημάτων σε γλώσσες τύπου SQL.

Σύμφωνα με το θεώρημα CAP (28), είναι αδύνατο για ένα διαμοιρασμένο υπολογιστικό σύστημα να παρέχει ταυτόχρονα και τις τρεις ακόλουθες εγγυήσεις:

- **Συνοχή**: όλοι οι κόμβοι βλέπουν τα ίδια δεδομένα την ίδια στιγμή
- **Διαθεσιμότητα:** κάθε αίτημα γίνεται αποδέκτης μίας απάντησης για το κατά πόσο ήταν επιτυχές ή όχι
- **Ανοχή Κατάτμησης**: το σύστημα εξακολουθεί να λειτουργεί, ακόμα κι αν υπάρχει αυθαίρετη απώλεια μηνυμάτων ή αστοχία από την πλευρά του συστήματος

Βάσει αυτού, οι NoSQL βάσεις θυσιάζουν την συνοχή για χάρη της διαθεσιμότητας και της ανοχής κατάτμησης. Εμπόδιο για την μεγαλύτερη υιοθέτηση της χρήσης NoSQL βάσεων είναι η έλλειψη πλήρους υποστήριξης συναλλαγών ACID (ατομικότητα, συνέπεια, απομόνωση, μονιμότητα), ένα σύνολο ιδιοτήτων το οποίο εγγυάται ότι οι συναλλαγές στην βάση δεδομένων λειτουργούν αξιόπιστα. Επίσης, άλλα προβλήματα είναι ότι δε μπορούν να χρησιμοποιηθούν γλώσσες ερωτημάτων χαμηλού επιπέδου, η έλλειψη τυποποιημένων διεπαφών και το γεγονός ότι έχουν πραγματοποιηθεί ήδη πολύ μεγάλες επενδύσεις από επιχειρήσεις στη χρήση και την ανάπτυξη της SQL (31).

Η **mongoDB** (47) είναι μία βάση δεδομένων εγγράφων (document oriented database), η οποία έχει αναπτυχθεί αρχικά από την 10gen το 2007. Η mongoDB είναι ένα σύστημα NoSQL βάσης δεδομένων ανεξάρτητη πλατφόρμας. Η αποθήκευση των εγγραφών σε ένα πίνακα της βάσης μπορεί να έχει τη μορφή δεδομένων αντικειμένων json (η mongoDB τα ονομάζει BSON), πράγμα το οποίο επιτρέπει πιο εύκολη και γρήγορη ενσωμάτωση των δεδομένων σε συγκεκριμένους τύπους εφαρμογών, ενώ δεν είναι απαραίτητο ο πίνακας να έχει συγκεκριμένο αριθμό πεδίων όπως στις σχεσιακές βάσεις, όπου εγγραφές συχνά περιέχουν πολλά κενά εξαιτίας του σταθερού αριθμού

πεδίων του πίνακα. Με λίγα λόγια το σχήμα της βάσης δεν είναι απαραίτητα σταθερό και μεταβάλλεται δυναμικά.

Επιπλέον, τα ερωτήματα που γίνονται για τη συλλογή των δεδομένων από τη βάση είναι δυναμικά και δεν απαιτούν συγκεκριμένους δείκτες, γράφονται σύμφωνα με την γλώσσα ερωτημάτων εγγράφων της mongoDB (document-based query language). Αποφεύγονται τα ερωτήματα ενώσεων (join) που είναι και η αιτία που δεν μπορεί να κλιμακωθούν οι σχεσιακές βάσεις. Η mongoDB είναι προτιμότερο να χρησιμοποιείται σε εφαρμογές όπου τα δεδομένα αλλάζουν πολύ γρήγορα και σημαντικό παράγοντα αποτελεί η ταχύτητα. Μία άλλη βάση τέτοιου τύπου είναι η couchdb, όπου είναι προτιμότερο να χρησιμοποιείται σε δεδομένα που δεν αλλάζουν γρήγορα και τα ερωτήματα είναι προκαθορισμένα π.χ. σε CMS και CRM.

Μερικά από τα βασικά χαρακτηριστικά της mongoDB είναι:

1. **Ερωτήματα κατά περίσταση**: Η mongoDB υποστηρίζει αναζήτηση βάσει πεδίου, περιοχής τιμών και κανονικές εκφράσεις.

2. **Ευρετηρίαση**: Κάθε πεδίο της mongoDB μπορεί να καταχωρηθεί σε ευρετήριο. Τα ευρετήρια της mongoDB είναι παρόμοια με αυτά στα RDBMS. Επίσης διαθέσιμα είναι και δευτερεύοντα ευρετήρια.

3. **Αναδιπλασιασμός**: Η mongoDB παρέχει μεγάλη διαθεσιμότητα και αυξημένο όγκο διαμεταγωγής, με τη χρήση αντιγράφων συνόλων (replica sets). Ένα αντίγραφο συνόλου απαρτίζεται από δύο ή περισσότερα αντίγραφα των δεδομένων. Κάθε αντίγραφο μπορεί να λειτουργήσει στο ρόλο του πρωτεύοντος ή του δευτερεύοντος αντιγράφου ανά πάσα στιγμή. Το πρωτεύον αντίγραφο εκτελεί όλες τις εντολές εγγραφής και ανάγνωσης από προεπιλογή. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των δεδομένων στο πρωτεύον χρησιμοποιώντας ενσωματωμένες λειτουργίες αναδιπλασιασμού. Όταν το πρωτεύον αντίγραφο αστοχεί, το αντίγραφο συνόλου εκτελεί αυτόματα μία διαδικασία εκλογής για να καθορίσει ποιο δευτερεύον αντίγραφο θα πρέπει να οριστεί πρωτεύον. Τα δευτερεύοντα αντίγραφα μπορούν να εκτελέσουν και αυτά εντολές ανάγνωσης, όμως τα δεδομένα είναι εν τέλει συνεπή έτσι κι αλλιώς.

4. **Εξισορρόπηση Φόρτου**: Η mongoDB κλιμακώνεται οριζόντια χρησιμοποιώντας «θραύση» (sharding). Ο χρήστης επιλέγει ένα κλειδί θραύσματος, το οποίο καθορίζει πώς τα δεδομένα σε μία συλλογή θα κατανεμηθούν. Τα δεδομένα χωρίζονται σε περιοχές (με βάση το κλειδί θραύσματος και κατανέμονται μεταξύ πολλαπλών θραυσμάτων. Η mongoDB μπορεί να εκτελεστεί σε πολλαπλούς διακομιστές, εξισορροπώντας το φόρτο και/ή αναδιπλασιάζοντας τα δεδομένα, ώστε να διατηρείται το σύστημα σε εκτελέσιμη κατάσταση σε περίπτωση που υπάρχει αστοχία υλικού. Η αυτόματη παραμετροποίηση είναι εύκολη στην εγκατάσταση, ενώ νέα μηχανήματα μπορούν να προστεθούν σε μία βάση δεδομένων που ήδη εκτελείται.

5. **Αποθήκευση Αρχείων**: Η mongoDB μπορεί να χρησιμοποιηθεί ως σύστημα αρχείων, εκμεταλλευόμενη τις δυνατότητες εξισορρόπησης φόρτου και αναδιπλασιασμού δεδομένων σε περισσότερες από μία συσκευές, με σκοπό την αποθήκευση αρχείων. Αυτή η λειτουργία, η οποία ονομάζεται GridFS, περιλαμβάνεται οδηγούς της mongoDB και παρέχεται χωρίς δυσκολία για διάφορες γλώσσες προγραμματισμού. Η mongoDB εκθέτει συναρτήσεις για διαχείριση αρχείων και περιεχομένου για προγραμματιστές. Σε ένα σύστημα mongoDB σε πολλαπλά μηχανήματα, τα αρχεία μπορούν να κατανεμηθούν και να αντιγραφούν πολλές φορές μεταξύ μηχανών με διαφάνεια. Με αυτό τον τρόπο, δημιουργείται ένα σύστημα με εξισορρόπηση φόρτου, το οποίο είναι ανεκτικό σε σφάλματα.

6. **Συσσωμάτωση**: Το MapReduce μπορεί να χρησιμοποιηθεί για την ομαδική επεξεργασία δεδομένων και λειτουργιών συσσωμάτωσης. Το πλαίσιο συσσωμάτωσης επιτρέπει στους χρήστες να αποκτούν το είδος των αποτελεσμάτων για το οποίο χρησιμοποιείται ο όρος GROUP BY στην SQL.

7. **Εκτέλεση JavaScript στο διακομιστή**: Η JavaScript μπορεί να χρησιμοποιηθεί για ερωτήματα, συναρτήσεις συσσωμάτωσης (όπως το MapReduce), ενώ οι εντολές αυτές μπορούν να αποστέλλονται απευθείας στη βάση στην οποία πρόκειται να εκτελεστούν.

8. **Προσαρμοσμένες Συλλογές**: Η mongoDB υποστηρίζει συλλογές σταθερού μεγέθους οι οποίες ονομάζονται «προσαρμοσμένες συλλογές» (capped collections). Αυτό το είδος συλλογών διατηρεί τη σειρά εγγραφής δεδομένων και, όταν το προκαθορισμένο μέγεθος έχει συμπληρωθεί, λειτουργεί ως κυκλική ουρά.

Η **TingoDB** είναι μία ενσωματωμένη NoSQL JavaScript βάση δεδομένων για το Node.js και το node-webkit. Το API και οι δυνατότητές του έχουν σχεδιαστεί για να είναι συμβατά προς τα πάνω με την MongoDB και τον οδηγό της για το node.js. Η TingoDB δεν είναι ένα API για την MongoDB επειδή χρησιμοποιεί το ίδιο συντακτικό. Πρόκειται για ένα ακριβές αντίγραφο, το οποίο επιτρέπει την ανάπτυξη εφαρμογών οι οποίες μπορούν με διαφάνεια να υποστηρίξουν και την MongoDB, αλλά και μία ελαφριά ενσωματωμένη μηχανή βάσεων δεδομένων. Είναι δυνατόν να υιοθετηθούν και κάποιες παράγωγες βιβλιοθήκες που εξαρτώνται από την MongoDB, όπως η Mongoose.js ODM βιβλιοθήκη, η οποία μπορεί να συνδεθεί με την TingoDB χρησιμοποιώντας τον οδηγό Tungus.

Το σύνολο των υποστηριζόμενων, αυτή τη στιγμή, λειτουργιών περιλαμβάνουν αναζήτηση, ευρετηρίαση, ατομικές ενημερώσεις, ομαδοποίηση και το MapReduce. Τα παραπάνω υλοποιούνται σε καθαρή JavaScript, χωρίς δυαδικές εξαρτήσεις και με επιδόσεις συγκρίσιμες με την MongoDB για βάσεις δεδομένων με λογικό μέγεθος. Η TingoDB προτείνεται ως μία λύση για μικρές έως μεσαίου μεγέθους απαιτήσεις εφαρμογών σε node.js για βάσεις δεδομένων.

Η TingoDB προσφέρει ατομικές ενημερώσεις και ταυτόχρονη πρόσβαση, ενώ εφαρμόζει ασφαλή μονιμότητα (persistence). Η TingoDB είναι πιο αργή από την MongoDB, όμως αυτό είναι λογικό για μία λύση που βασίζεται σε καθαρή JavaScript.

4. Απαιτήσεις συστήματος

Στο κεφάλαιο αυτό γίνεται περιγραφή των απαιτήσεων ενός συστήματος ενδιάμεσου λογισμικού και συγκεκριμένα αυτές που σχετίζονται με την διαχείριση του πλαισίου. Αρχικά γίνεται μια σύντομη περιγραφή των στόχων του λογισμικού. Στη συνέχεια ορίζονται οι χρήστες και αναλύονται οι λειτουργικές απαιτήσεις του λογισμικού. Τέλος αναφέρονται τα μη λειτουργικά χαρακτηριστικά που σχετίζονται με συγκεκριμένες ποιοτικές ιδιότητες και περιορισμούς του λογισμικού (85).

4.1.Στόχοι

Οι πλατφόρμες ενδιάμεσου λογισμικού παρέχουν ένα σύνολο υπηρεσιών και προγραμματιστικών διεπαφών για την ανάπτυξη και υποστήριξη εφαρμογών. Το σύνολο των υπηρεσιών εξαρτάται από το πεδίο εφαρμογής του λογισμικού, την αρχιτεκτονική προσέγγιση με την οποία σχεδιάστηκε και το είδος των εφαρμογών για τις οποίες προορίζεται. Το ενδιάμεσο λογισμικό για εφαρμογές επίγνωσης πλαισίου παρουσιάζει επιπλέον λειτουργικά χαρακτηριστικά, που σχετίζονται με τις απαιτήσεις στη διαχείριση της πληροφορίας πλαισίου. Οι βασικές απαιτήσεις για ένα ενδιάμεσο λογισμικό επίγνωσης πλαισίου, συνοψίζονται στις εξής :

1. Ένα μοντέλο για την αναπαράσταση και διαχείριση του πλαισίου. Το μοντέλο πρέπει να περιγράφει τα χρονικά χαρακτηριστικά της πληροφορίας πλαισίου.
2. Μία μέθοδο για το χειρισμό των χρονικών εξαρτήσεων του πλαισίου.
3. Ένα μηχανισμό συλλογής του πλαισίου από τους προμηθευτές. Η ενημέρωση για τις αλλαγές του πλαισίου θα πρέπει να γίνεται με ασύγχρονη επικοινωνία.
4. Ερμηνεία και μετατροπή του πλαισίου σε μορφή κατάλληλη για χρήση.
5. Αποτίμηση του πλαισίου και εξαγωγή συμπερασμάτων για την αντίδραση της εφαρμογής.
6. Έλεγχος αξιοπιστίας των δεδομένων που συλλέγονται.
7. Προσαρμογή του συστήματος στις αλλαγές του πλαισίου.
8. Δυναμική διαμόρφωση του συστήματος.
9. Προγραμματιστικότητα.

Κάποια συστήματα ενδιάμεσου λογισμικού προορίζονται για περιβάλλοντα στα οποία η εφαρμογή μπορεί να προσπελάσει και να χρησιμοποιήσει υπηρεσίες από κατανεμημένους εξυπηρετητές ή άλλες εφαρμογές. Οι υπηρεσίες παρέχονται συνήθως

από στατικές μονάδες λογισμικού σε έξυπνους χώρους ή από κινητές εφαρμογές. Σε τέτοιες συνθήκες απαιτείται ένας μηχανισμός εύρεσης ή επιλογής της κατάλληλης υπηρεσίας, η οποία μπορεί να κληθεί από την κινητή εφαρμογή του χρήστη (64). Μια επιπλέον υπηρεσία αφορά στην εύρεση προμηθευτών πλαισίου. Σε κάποια συστήματα διάχυτου υπολογισμού γίνεται η θεώρηση ότι οι προμηθευτές του πλαισίου δεν είναι πάντα γνωστοί στην εφαρμογή. Επίσης μπορεί να μην είναι πάντα προφανές ποιος, από μια λίστα γνωστών προμηθευτών παρέχει το επιθυμητό πλαίσιο. Έτσι, η συλλογή της πληροφορίας πρέπει να γίνει από άγνωστους ή επιλεγμένους προμηθευτές. Ο προμηθευτής πρέπει να αναζητηθεί ή να επιλεγεί με βάση τα χαρακτηριστικά του. Σε αυτές τις συνθήκες απαιτούνται επιπλέον υπηρεσίες για εύρεση ή επιλογή του κατάλληλου προμηθευτή (46).

Σε επόμενο στάδιο, πραγματοποιείται η ανάλυση των εννιά βασικών απαιτήσεων για μία υπηρεσία ενδιάμεσου λογισμικού η οποία επικεντρώνεται στην διαχείριση του πλαισίου και την ανάπτυξη εφαρμογών επίγνωσης πλαισίου. Η υπηρεσία μπορεί να ενσωματωθεί σε οποιοδήποτε συμβατό λογισμικό και λειτουργεί χωρίς επίβλεψη. Παρακάτω περιγράφονται κάποια σενάρια χρήσης του ενδιάμεσου λογισμικού για να γίνει πιο κατανοητή η λειτουργία του.

4.2.Σενάρια Χρήσης

Μια εφαρμογή σχεδιάζεται ώστε να παρέχει ένα σύνολο υπηρεσιών στο χρήστη. Οι υπηρεσίες αυτές μπορεί να αφορούν σε υπηρεσίες διαδικτύου, όπως εμπορικές συναλλαγές, προσπέλαση ιστοσελίδων ή τοπικές λειτουργίες όπως, ηλεκτρονική ατζέντα, διαχείριση τοπικών δεδομένων και χειρισμός εισερχόμενων κλήσεων και SMS αν πρόκειται για κινητά τηλέφωνα κ.α.

Η εφαρμογή αποκτά χαρακτηριστικά επίγνωσης πλαισίου αν οι λειτουργίες της δεν εκτελούνται πάντα με τον ίδιο τρόπο και μόνο με επιλογή του χρήστη, αλλά η εκτέλεσή τους εξαρτάται από το πλαίσιο. Η προσαρμοζόμενη λειτουργία της εφαρμογής επιτυγχάνεται με την προσθήκη ενδιάμεσου λογισμικού. Η υπηρεσία ενδιάμεσου λογισμικού αναλαμβάνει να παρακολουθεί το πλαίσιο και να προσαρμόζει τη λειτουργία της εφαρμογής. Η υπηρεσία μπορεί να υποστηρίξει την συμπεριφορά επίγνωσης πλαισίου της εφαρμογής με τρεις τρόπους:

- **Αυτόματη εκτέλεση λειτουργιών**: Δοσμένου του ανάλογου πλαισίου του χρήστη ή της συσκευής του, δίνεται η δυνατότητα στην εφαρμογή να εκτελέσει κάποια λειτουργία που σχετίζεται με αυτό. Ένα παράδειγμα είναι η εμφάνιση του προγράμματος ενός κινηματογράφου όταν ο χρήστης εισέρχεται σε αυτόν ή η παρουσίαση πληροφοριών ενός εκθέματος όταν βρίσκεται ο χρήστης μπροστά του σε ένα μουσείο.
- **Προσαρμοσμένη εκτέλεση λειτουργιών**: Η εκτέλεση μία λειτουργίας από τον χρήστη μπορεί να προσαρμοστεί με βάση το παρόν πλαίσιο του χρήστη. Για παράδειγμα, όταν ο χρήστης κινείται με τα πόδια και ορίσει έναν προορισμό στο πρόγραμμα πλοήγησής της φορητής συσκευής του, η διαδρομή που εμφανίζεται αφορά σε κίνηση με τα πόδια και Μέσα Μαζικής Μεταφοράς, ενώ όταν βρίσκεται στο αυτοκίνητό του, η διαδρομή αφορά σε κίνηση με ιδιωτικό όχημα.
- **Αυτοδιαμόρφωση**: Η υπηρεσία μπορεί να διαμορφώνει αυτόματα την εσωτερική της δομή ανάλογα με το πλαίσιο. Για παράδειγμα, είναι δυνατό μία φορητή συσκευή να μπαίνει σε κατάσταση αθόρυβης λειτουργίας όταν ο χρήστης εισέρχεται σε μία κινηματογραφική αίθουσα ή είναι η ώρα που είναι προγραμματισμένη κάποια επαγγελματική συνάντηση. Επιπλέον, μπορεί να εκτελεί κάποιες εφαρμογές στον υπολογιστή του σπιτιού όταν ο χρήστης εισέρχεται σε αυτό ή να χαμηλώνει η φωτεινότητα της οθόνης μίας συσκευής όταν διαπιστώνεται ότι ο περιβάλλοντας χώρος είναι σκοτεινός.

4.3.Καταγραφή Λειτουργικών Απαιτήσεων

Οι λειτουργικές απαιτήσεις περιγράφουν τις λειτουργίες που παρέχει το ενδιάμεσο λογισμικό στους πράκτορες (actors) με τις οποίες αλληλοεπιδρά. Οι λειτουργικές απαιτήσεις αφορούν κυρίως στην υποστήριξη της εφαρμογής που εκτελείται πάνω στο ενδιάμεσο λογισμικό. Επιπλέον παρέχονται και λειτουργίες που αφορούν τους χρήστες των εφαρμογών επίγνωσης πλαισίου. Στη συνέχεια ορίζονται οι πράκτορες και γίνεται η περιγραφή των λειτουργικών απαιτήσεων.

4.3.1. Πράκτορες του συστήματος

Στην περίπτωση ενός ενδιάμεσου λογισμικού με δυνατότητες επίγνωσης πλαισίου οι πράκτορες με τους οποίους αλληλοεπιδρά το σύστημα είναι οι ακόλουθοι:

- **Ο προγραμματιστής της εφαρμογής**: Ο προγραμματιστής αναλαμβάνει να συνδέσει την εφαρμογή με το ενδιάμεσο λογισμικό και να προσαρμόσει κατάλληλα τη λειτουργία της. Για το λόγο αυτό χρησιμοποιεί τις προγραμματιστικές διεπαφές που αυτό παρέχει.
- **Ο προμηθευτής του πλαισίου**: Προμηθευτής είναι οποιαδήποτε οντότητα παρέχει στην υπηρεσία τιμές για τις παραμέτρους του πλαισίου. Προμηθευτής μπορεί να είναι και η εφαρμογή, αφού κάποιες παράμετροι του πλαισίου μπορεί να σχετίζονται με λειτουργικά χαρακτηριστικά της. Οι προμηθευτές ανιχνεύουν τις αλλαγές στο πλαίσιο και χρησιμοποιούν το μηχανισμό επικοινωνίας που παρέχει η υπηρεσία, για να στείλουν τις τιμές των παραμέτρων.
- **Ο χρήστης της εφαρμογής**: Ο χρήστης μπορεί να χρησιμοποιήσει τις διεπαφές που παρέχει η υπηρεσία για να τροποποιήσει κάποια εσωτερικά χαρακτηριστικά της. Επιπλέον μπορεί να λειτουργήσει σαν προμηθευτής πληροφοριών πλαισίου, καταχωρώντας τις προτιμήσεις του από μια γραφική διεπαφή της εφαρμογής.

4.3.2. Λειτουργικές Απαιτήσεις

Επειδή η χρήση δεδομένων πλαισίου δεν υπόκειται σε κάποια δεδομένη μεθοδολογία διαχείρισής τους, απαιτείται ένα μοντέλο διαχείρισης και αναπαράστασης της πληροφορίας. Το μοντέλο χρησιμοποιείται για την αποθήκευση του πλαισίου σε κατάλληλη μορφή που διευκολύνει τη διαχείρισή του. Επιπλέον καθορίζεται ο τρόπος που υλοποιούνται οι διαδικασίες διαχείρισης του πλαισίου.

Παράλληλα, επειδή η πληροφορία πλαισίου είναι σε μεγάλο βαθμό χρονικά εξαρτημένη, είναι απαραίτητο να εφαρμόζεται ένα ενιαίο **μοντέλο χειρισμού των χρονικών εξαρτήσεων**. Το παρόν πλαίσιο του χρήστη ή της συσκευής ενδέχεται να περιέχει γεγονότα που συνέβησαν στο παρελθόν, ενώ η σειρά με την οποία

πραγματοποιήθηκαν να μεταβάλλει τα συμπεράσματα που εξάγονται. Το μοντέλο του πλαισίου πρέπει να περιλαμβάνει τις κατάλληλες αφαιρέσεις για το χειρισμό των χρονικών εξαρτήσεων της πληροφορίας. Επιπλέον, όπως σε κάθε αλληλεπιδραστικό σύστημα, ο χρόνος υπεισέρχεται και στην περιγραφή της αλληλεπίδραση της υπηρεσίας με το περιβάλλον. Το μοντέλο πρέπει να παρέχει τη δυνατότητα για χρονική περιγραφή, των ενεργειών προσαρμογής της εφαρμογής. Για το χειρισμό των χρονικών εξαρτήσεων απαιτείται μία κατάλληλη μέθοδος που θα περιλαμβάνει χρονικά χαρακτηριστικά στην επεξεργασία του πλαισίου.

Η πληροφορία πλαισίου συλλέγεται από ετερογενείς πηγές δεδομένων. Η **συλλογή της πληροφορίας πλαισίου** απαιτεί έναν κοινό μηχανισμό επικοινωνίας του ενδιάμεσου λογισμικού με τους προμηθευτές, για την απόκτηση του πλαισίου. Η επικοινωνία σε επίπεδο ενδιάμεσου λογισμικού θα πρέπει να είναι ανεξάρτητη από τη φύση της πηγής (συσκευή, λογισμικό, προτιμήσεις χρήστη) και τα τεχνικά χαρακτηριστικά της. Ο μηχανισμός επικοινωνίας πρέπει να μεταφέρει τις τιμές του πλαισίου από τους προμηθευτές στο ενδιάμεσο λογισμικό διατηρώντας τη συνέπεια και τη χρονική τους πληροφορία.

Οι αλλαγές στο πλαίσιο πρέπει να γίνονται άμεσα αντιληπτές από το ενδιάμεσο λογισμικό προκειμένου να αποτιμηθούν και να προσαρμοστεί ανάλογα η εφαρμογή. Με δεδομένο ότι μεταβολές συμβαίνουν αρκετά συχνά και απρόβλεπτα, η **ενημέρωση πρέπει να γίνεται και με ασύγχρονο τρόπο**. Ο μηχανισμός ειδοποίησης πρέπει να είναι ανεξάρτητος του τύπου της πληροφορίας που μεταφέρεται στα γεγονότα. Επιπλέον, πρέπει να παρέχεται η δυνατότητα στο προγραμματιστή της εφαρμογής να προσθέσει πιο σύνθετους μηχανισμούς ειδοποίησης. Η ενημέρωση για το πλαίσιο θα μπορούσε εναλλακτικά να γίνεται με απ' ευθείας ερωτήσεις στους προμηθευτές. Αυτή η μέθοδος δεν μπορεί να χρησιμοποιηθεί σαν βασικός μηχανισμός συλλογής των δεδομένων. Οι τιμές των γνωρισμάτων του πλαισίου, στην πλειοψηφία τους, μεταβάλλονται ταχύτατα και οι ερωτήσεις μπορεί να οδηγήσουν σε ασυνεπή πληροφορία. Επιπλέον οι συνεχείς ερωτήσεις θα οδηγούσαν σε χρονοβόρες διακοπές στη λειτουργία της εφαρμογής ή σε ανούσια εξάντληση των υπολογιστικών πόρων.

Οι προμηθευτές παρέχουν ετερογενή πληροφορία η οποία πρέπει να αναγνωριστεί και να μετατραπεί σε μία χρήσιμη αναπαράσταση. Η μορφή του πλαισίου μπορεί να είναι αριθμητικές τιμές από αισθητήρες υλικού ή συμβολοσειρές. Αυτές οι τιμές δεν μπορούν να χρησιμοποιηθούν άμεσα από το ενδιάμεσο λογισμικό. Προτού

γίνει χρήση του πλαισίου πρέπει να ερμηνευτεί και να μετατραπεί σε κατάλληλη μορφή. Συνεπώς πρέπει να υπάρχει ένας μηχανισμός απόκρυψης της ετερογένειας και της φυσικής μορφής της πληροφορίας με έναν **μηχανισμό μετατροπής του πλαισίου**.

Το ενδιάμεσο λογισμικό πρέπει να έχει **ικανότητες συλλογισμού** για την αποτίμηση του πλαισίου. Μέσω του συλλογισμού, η υπηρεσία επεξεργάζεται τις τιμές του πλαισίου και προσαρμόζει ανάλογα τη συμπεριφορά της εφαρμογής κάθε φορά που είναι απαραίτητο. Η ικανότητα συλλογισμού απαιτεί ενσωματωμένη ευφυία, ώστε να παίρνονται οι αποφάσεις που οδηγούν σε ενέργειες προσαρμογής. Με άλλα λόγια, η υπηρεσία γνωρίζοντας το εκάστοτε πλαίσιο, πρέπει να εξάγει συμπεράσματα που αφορούν την κατάλληλη συμπεριφορά της εφαρμογής.

Το πλαίσιο δεν μπορεί να θεωρηθεί πάντα αξιόπιστη πληροφορία. Συνεπώς, τα συμπεράσματα που εξάγονται από την επεξεργασία της αναξιόπιστης πληροφορίας μπορεί να είναι λανθασμένα. Μια υπηρεσία ενδιάμεσου λογισμικού πρέπει έχει τη δυνατότητα να **διαχειρίζεται την αβεβαιότητα του πλαισίου**, βγάζοντας συμπεράσματα για την ορθότητα της πληροφορίας που συλλέγεται.

Το ενδιάμεσο λογισμικό θα πρέπει να **προσαρμόζει τη λειτουργία της εφαρμογής εκτελώντας κατάλληλες ενέργειες ανάλογα με το πλαίσιο**. Για την προσαρμογή απαιτείται, επιλογή των κατάλληλων ενεργειών με βάση τα συμπεράσματα που εξάγονται από τη διαδικασία συλλογισμού. Για τον παραπάνω λόγο πρέπει να περιλαμβάνεται ένας μηχανισμός αυτόματης αντιστοίχισης των συμπερασμάτων, σε ενέργειες προσαρμογής προς την εφαρμογή.

Το ενδιάμεσο λογισμικό παρέχει, σε ένα βαθμό, διαφάνεια στην εφαρμογή. Παρόλα αυτά, η απόλυτη διαφάνεια δεν είναι επιθυμητή. Η συμπεριφορά του ενδιάμεσου λογισμικού και οι πολιτικές αλληλεπίδρασης του με την εφαρμογή πρέπει να προσαρμόζονται και στις ανάγκες του χρήστη. Ο χρήστης πρέπει να έχει τη δυνατότητα να διαμορφώνει την λειτουργία καταχωρώντας τις προσωπικές του προτιμήσεις. Η **διαμόρφωση πρέπει να γίνεται δυναμικά**, καθώς κατά τη διάρκεια εκτέλεσης της εφαρμογής οι ανάγκες του χρήστη αλλάζουν. Ο χρήστης πρέπει να έχει τη δυνατότητα, να βλέπει ένα μέρος της εσωτερικής δομής του ενδιάμεσου λογισμικού και να μπορεί να την αλλάξει ανά πάσα στιγμή. Επιπλέον, όπως αναφέρθηκε και παραπάνω, η υπηρεσία θα πρέπει να **αυτοδιαμορφώνεται**. Ανάλογα με το *πλαίσιο* η συμπεριφορά της υπηρεσίας πρέπει να προσαρμόζεται, αν αυτό είναι απαραίτητο. Αυτή

η προσαρμογή έχει σαν αποτέλεσμα την αποδοτικότερη λειτουργία και την καλύτερη υποστήριξη της εφαρμογής.

Μια σημαντική απαίτηση για το ενδιάμεσο λογισμικό είναι η παροχή ενός **προγραμματιστικού περιβάλλοντος** για την ανάπτυξη εφαρμογών. Το περιβάλλον αυτό πρέπει να περιέχει εργαλεία, που βοηθούν τον προγραμματιστή να χτίσει εφαρμογές επίγνωσης πλαισίου. Ο προγραμματιστής πρέπει να μπορεί να προσθέσει επιπλέον συστατικά που επεκτείνουν ή προσαρμόζουν την εφαρμογή πάνω στο ενδιάμεσο λογισμικό. Τα προγραμματιστικά εργαλεία μπορεί να είναι στη μορφή κάποιας βιβλιοθήκης λογισμικού συμβατής με μία γλώσσα υψηλού επιπέδου.

4.4.Καταγραφή Μη Λειτουργικών Απαιτήσεων

Στα υπολογιστικά συστήματα, οι μη λειτουργικές απαιτήσεις είναι οι απαιτήσεις που καθορίζουν κριτήρια που μπορούν να χρησιμοποιηθούν για να κρίνουν τη λειτουργία του συστήματος, αντί του να περιγράφουν συγκεκριμένες συμπεριφορές. Οι μη λειτουργικές απαιτήσεις του ενδιάμεσου λογισμικού σχετίζονται με περιορισμούς που επιβάλει το περιβάλλον στο οποίο προορίζεται να λειτουργήσει. Το ενδιάμεσο λογισμικό αποτελεί ένα αλληλεπιδραστικό σύστημα πραγματικού χρόνου. Συνεπώς περιλαμβάνει όλες τις μη λειτουργικές απαιτήσεις των συστημάτων αυτής της κατηγορίας. Επιπλέον η συμπεριφορά του συστήματος ως προς την επίγνωση πλαισίου επιβάλλει κάποιους περιορισμούς στην υλοποίηση και τη σχεδίαση του.

4.4.1.Χαρακτηριστικά του Μοντέλου Πλαισίου

Το μοντέλο του πλαισίου πρέπει να έχει τα εξής χαρακτηριστικά:

- **Απλότητα:** Οι εκφράσεις και οι περιγραφές πρέπει να είναι όσο το δυνατόν πιο απλές για να διευκολύνουν τη δουλειά του προγραμματιστή και να μπορούν χρησιμοποιηθούν και από το χρήστη.
- **Επεκτασιμότητα**: Οι παράμετροι του πλαισίου που μπορούν να χρησιμοποιηθούν πρέπει να είναι απεριόριστες και να μπορούν να προστεθούν εύκολα επιπλέον παράμετροι.

- **Εκφραστικότητα**: Το μοντέλο πρέπει να έχει μεγάλες δυνατότητες έκφρασης ώστε να μην θέτει περιορισμούς στις συνθήκες που μπορούν να περιγραφούν.
- **Γενικότητα**: Το μοντέλο πρέπει να είναι αρκετά γενικό, ώστε να μπορεί να συμπεριλάβει όλες τις ιδιότητες της πληροφορίας πλαισίου.

Επιπλέον η αναπαράσταση πρέπει να είναι τυπική, ώστε να εκφράζει με σαφήνεια την πληροφορία.

4.4.2. Αυτόνομη Αρχιτεκτονική

Το ενδιάμεσο λογισμικό επίγνωσης πλαισίου έχει σημαντική χρήση σε εφαρμογές για φορητές συσκευές και κατά προέκταση σε περιβάλλοντα διάχυτου υπολογισμού. Μια προφανής αρχιτεκτονική για την υπηρεσία είναι μια προσέγγιση πελάτη-εξυπηρετητή. Σε αυτή την αρχιτεκτονική ένα «ελαφρύ» υποσύστημα εκτελείται στην κινητή συσκευή και επικοινωνεί με την εφαρμογή. Ένα άλλο υποσύστημα βρίσκεται σε κάποιο σταθερό κόμβο και αναλαμβάνει τη συλλογή και επεξεργασία των δεδομένων πλαισίου, που αποτελούν σύνθετες υπολογιστικά διαδικασίες επικοινωνώντας με το υποσύστημα που βρίσκεται στην διάχυτη συσκευή.

Αυτή η προσέγγιση έχει χρησιμοποιηθεί αρκετά στη βιβλιογραφία, γιατί παρέχει λύση στο πρόβλημα της αδυναμίας των κινητών συσκευών να υποστηρίξουν βαριές υπολογιστικές διεργασίες (19, 22, 30, 39, 42, 57). Παρόλα αυτά, η αρχιτεκτονική με έναν κεντρικό κόμβο παροχής κρίσιμων υπηρεσιών δεν είναι η πλέον κατάλληλη αφού περιορίζει γεωγραφικά το περιβάλλον υπολογισμού σε μικρούς χώρους. Ο χρήστης δεν μπορεί να κινηθεί ελεύθερα αφού η λειτουργικότητα του ενδιάμεσου λογισμικού χάνεται μακριά από τον εξυπηρετητή που επικοινωνεί με τη συσκευή. Επιπλέον σε ένα περιβάλλον που χαρακτηρίζεται από ασταθείς και αναξιόπιστες συνδέσεις και χαμηλό εύρος ζώνης, η επικοινωνία με άλλους κόμβους, μπορεί να σταθεί εμπόδιο στη λειτουργικότητα και την απόδοση του ενδιάμεσου λογισμικού.

Συνεπώς το λογισμικό πρέπει να είναι ανεξάρτητο από κεντρικές μονάδες διαχείρισης και πρέπει να παρέχει υπηρεσίες οπουδήποτε και ανά πάσα στιγμή. Η

υπηρεσία πρέπει να λειτουργεί αυτόνομα πάνω στη συσκευή συμβάλλοντας σε ένα πλήρως αποκεντροποιημένο περιβάλλον διομότιμης επικοινωνίας.

4.4.3. Ταυτοχρονισμός

Η εκτέλεση του ενδιάμεσου λογισμικού θα πρέπει να είναι ανεξάρτητη από το υπόλοιπο σύστημα και να εκτελείται ταυτόχρονα με άλλα τμήματα λογισμικού. Οι λειτουργίες της εφαρμογής δεν πρέπει να εμποδίζονται από την υπηρεσία και το αντίστροφο. Για το λόγο αυτό, όλες οι υπολογιστικές διεργασίες θα πρέπει να εκτελούνται παράλληλα με το υπόλοιπο σύστημα.

Ο ταυτοχρονισμός επιτυγχάνεται με την εκτέλεση του ενδιάμεσου λογισμικού σε διαφορετικό νήμα ελέγχου από το υπόλοιπο σύστημα, ώστε να εκτελούνται «ψευδοπαράλληλα» στον επεξεργαστή της συσκευής. Λέγοντας ψευδοπαράλληλα εννοούμε ότι, το κάθε νήμα δεν εκτελείται σε διαφορετικό επεξεργαστή, αλλά μοιράζονται τον ίδιο. Η χρήση του επεξεργαστή εναλλάσσεται μεταξύ των νημάτων, τα οποία εκτελούνται για ορισμένο χρονικό διάστημα και μετά διακόπτονται. Η παράλληλη λειτουργία επιτρέπει στην εκτέλεση να γίνεται με απόλυτη διαφάνεια, χωρίς να επηρεάζει αισθητά, χρονικά ή λειτουργικά, την απόδοση της εφαρμογής.

4.4.4. Βελτιστοποίηση κατανάλωσης υπολογιστικών πόρων

Το χαμηλό υπολογιστικό κόστος λειτουργίας είναι πάντα επιθυμητό σε συστήματα λογισμικού. Ιδιαίτερα επιθυμητό είναι για λογισμικό που προορίζεται για διάχυτες συσκευές, όπου οι υπολογιστικοί πόροι είναι περιορισμένοι. Επιπλέον, τα αλληλεπιδραστικά χαρακτηριστικά του συγκεκριμένου ενδιάμεσου λογισμικού κάνουν ακόμα πιο κρίσιμη την απαίτηση για «ελαφριά» υπολογιστική λειτουργία. Ο ταυτοχρονισμός, ο χειρισμός των εξωτερικών γεγονότων, η λήψη αποφάσεων και οι ενέργειες προσαρμογής επιβαρύνουν υπολογιστικά τους πόρους τους συστήματος.

Η λειτουργικότητα και η απόδοση της εφαρμογής πρέπει να επηρεάζονται το λιγότερο δυνατό από την υπολογιστική επιβάρυνση που προσθέτει το ενδιάμεσο λογισμικό. Συνεπώς η υπηρεσία θα πρέπει να κάνει οικονομική διαχείριση των

απαραίτητων πόρων για το σύστημα, όπως μνήμη, επεξεργαστική ισχύ και αποθηκευτικό χώρο. Η σχεδίαση και η ανάπτυξη θα πρέπει να γίνει με γνώμονα αυτήν την απαίτηση.

4.4.5. Βελτιστοποίηση Απόδοσης

Προκειμένου το ενδιάμεσο λογισμικό να λειτουργεί σε πραγματικό χρόνο και να υποστηρίζει αποτελεσματικά την εφαρμογή απαιτείται ελάχιστη χρονική απόκριση κατά την εκτέλεσή του. Η μεγιστοποίηση της απόδοσης επιβάλει σχεδιαστικούς αλλά και προγραμματιστικούς περιορισμούς στην ανάπτυξη στης εφαρμογής. Βασικές αρχές της αντικειμενοστραφούς σχεδίασης όπως η αφαίρεση και η τμηματοποίηση (modularity) είναι απαραίτητες για σύνθετα συστήματα πραγματικού χρόνου, αλλά μπορεί να οδηγήσουν σε σημαντική μείωση της απόδοσης. Ένα λογισμικό με πολλά επίπεδα αφαίρεσης επιφέρει ένα κόστος απόδοσης, που οφείλεται στην αυξημένη επικοινωνία μεταξύ αντικειμένων στα διάφορα επίπεδα, για το χειρισμό μιας κατάστασης.

Το πρόβλημα αυτό μπορεί να διευθετηθεί σε επίπεδο υλοποίησης δημιουργώντας συντομεύσεις στην επικοινωνία μεταξύ των αντικειμένων. Έτσι ένα αντικείμενο μπορεί να έχει πρόσβαση σε ένα άλλο που κανονικά δεν θα του επιτρεπόταν. Επιπλέον μπορεί να γίνει «σύμπτυξη» κλάσεων, ώστε να μειωθεί ο αριθμός των επιπέδων μεταξύ των διαφόρων τμημάτων του λογισμικού. Αυτές οι λύσεις αυξάνουν τη χρονική απόδοση της υπηρεσίας, αλλά επιφέρουν μείωση της τμηματοποίησης στο λογισμικό. Το λογισμικό αποκτά πιο συμπαγή μορφή και χάνει την ευέλικτη και εύχρηστη δομή του. Η εύρεση της χρυσής τομής μεταξύ τμηματικής σχεδίασης και απόδοσης είναι μια κρίσιμη απαίτηση.

4.4.6. Μεταφερσιμότητα

Η υλοποίηση θα πρέπει να γίνει με κατάλληλο λογισμικό ώστε η οποιαδήποτε εφαρμογή να μπορεί να λειτουργήσει ανεξαρτήτως συσκευής. Αυτό μέχρι πρότινος δεν ήταν εφικτό, αλλά τα τελευταία χρόνια έχουν γίνει προσπάθειες και έχουν καθοριστεί κάποιες προδιαγραφές διάχυτου λογισμικού (*MIDP*), οι οποίες με το πέρασμα του χρόνου υιοθετούνται από όλο και περισσότερους κατασκευαστές συσκευών (45).

4.4.7. Αξιοπιστία

Η απαίτηση για αξιοπιστία είναι ιδιαίτερα κρίσιμη, όπως και σε όλα τα συστήματα που παρέχουν υπηρεσίες σε πραγματικό χρόνο. Η αλληλεπίδραση με το περιβάλλον μπορεί να κρύβει απρόβλεπτες καταστάσεις και δημιουργία απροσδόκητων σεναρίων. Η υπηρεσία πρέπει να λειτουργεί υπό οποιεσδήποτε συνθήκες και για κάθε πιθανό σενάριο. Επιπλέον πρέπει να ανταποκρίνεται στα γεγονότα που δέχεται, ανεξάρτητα τον αριθμό και τη σειρά με την οποία πραγματοποιούνται.

Ο ταυτοχρονισμός επιβάλλει επιπρόσθετη πολυπλοκότητα στην υπηρεσία που εύκολα μπορεί να οδηγήσει σε σφάλματα κατά την εκτέλεση. Το ενδιάμεσο λογισμικό πρέπει να χειρίζεται κατάλληλα την παράλληλη εκτέλεση των τμημάτων του και να αποτρέπει την πραγματοποίηση ανεπιθύμητων καταστάσεων (αδιέξοδα, ασιτία, κρίσιμες καθυστερήσεις). Ακόμα και στην περίπτωση σφαλμάτων ή εξαιρέσεων, ο χειρισμός τους θα πρέπει να γίνεται εσωτερικά από το ίδιο το ενδιάμεσο λογισμικό, παραμένοντας κάθε στιγμή σε λειτουργία.

4.4.8. Απλότητα

Ο σημαντικότερος ίσως παράγοντας, που κρίνει τη χρηστικότητα ενός αλληλεπιδραστικού συστήματος είναι η ανθρώπινη αντίληψη. Ένα σύστημα που είναι δύσκολο στη κατανόηση, θα χρησιμοποιηθεί λιγότερο από αυτούς που θέλουν να εκμεταλλευτούν τις υπηρεσίες του. Ένα ενδιάμεσο λογισμικό πρέπει να είναι απλό και κατανοητό στις παροχές του, καθώς και στον τρόπο που μπορεί κάποιος να τις χρησιμοποιήσει. Η απλότητα αφορά τόσο την ενσωμάτωση της υπηρεσίας σε ένα σύστημα επίγνωσης πλαισίου, όσο και την αλληλεπίδραση με το χρήστη και τον προγραμματιστή της εφαρμογής.

5. Προδιαγραφές Διαχειριστή και API Πλαισίου του webinos

5.1.Συλλογή πλαισίου

Στο webinos η θεμελιώδης δομή περιγραφής και καταχώρησης ενός τμήματος δεδομένων πλαισίου είναι το *Αντικείμενο Πλαισίου* (Context Object). Συγκεκριμένα, ένα Αντικείμενο Πλαισίου είναι η ελάχιστη μονάδα πληροφορίας ή δεδομένου που καθορίζει και είναι αρκετή για να περιγράψει ένα κομμάτι πληροφορίας πλαισίου (51). Για παράδειγμα, ενώ μία κλήση σε έναν αισθητήρα GPS μπορεί να επιστρέψει ένα πλήθος πληροφοριών ως απαντήσεις (γεωγραφικό μήκος, γεωγραφικό πλάτος, κατεύθυνση, ταχύτητα, ακρίβεια, υψομετρική ακρίβεια κλπ), ένα σχετικό αντικείμενο πλαισίου, το οποίο μπορεί να λέγεται MyLocation, θα περιέχει μόνο αυτές τις πληροφορίες που είναι απαραίτητες για τον καθορισμό της μονάδας της πληροφορίες που ζητείται. Αν η πληροφορία αυτή μπορεί να θεωρηθεί ότι περιγράφεται επαρκώς με τις δύο γεωγραφικές συντεταγμένες και την ακρίβεια, το *MyLocation* μπορεί να περιέχει μόνο αυτά. Αυτό επιτυγχάνεται με τον αποκλεισμό/φιλτράρισμα κάποιων δεδομένων ή και με την προσθήκη άλλων. Στην περίπτωση αυτή καταλήγουμε σε ένα Αντικείμενο Πλαισίου που αποτελείται από τα γεωγραφικό μήκος, το γεωγραφικό πλάτος, την ακρίβεια πλάτους και μήκους, την ακρίβεια υψομέτρου και τον χρόνο.

Το περιβάλλον πλαισίου του webinos παρέχει πρόσβαση σε δεδομένα πλαισίου ούτως ώστε να καταστούν δυνατές η σχεδίαση και η λειτουργία εφαρμογών/ υπηρεσιών με οδηγό το πλαίσιο του webinos. Το περιβάλλον του webinos είναι υπεύθυνο για τη συλλογή και αποθήκευση δεδομένων πλαισίου, μέσω του προσδιορισμού συγκεκριμένων γεγονότων σχετιζόμενα με το πλαίσιο, που συμβαίνουν μέσα στις εμπλουτισμένες με τις δυνατότητες του webinos εφαρμογές, και την παροχή εφαρμογών που έχουν ένα στρώμα πρόσβασης σε αυτά τα δεδομένα, είτε αναζητώντας τα κατά την αποθήκευση είτε με την ενημέρωση σε πραγματικό χρόνο για αλλαγές στο πλαίσιο, όταν συμβαίνουν συγκεκριμένα γεγονότα.

Μια εφαρμογή επίγνωσης πλαισίου είτε χρησιμοποιεί ένα συγκεκριμένο τμήμα, είτε έναν συνδυασμό, είτε μία χρονοσειρά δεδομένων σχετιζόμενα με το πλαίσιο, για παράδειγμα την τωρινή κατάσταση της πυξίδας ενός κινητού τηλεφώνου, την τωρινή κατάσταση του αισθητήρα φωτός περιβάλλοντος ενός φορητού υπολογιστή, ή η

γεωγραφική τοποθεσία και τα πλησιέστερα ασύρματα δίκτυα, ούτως ώστε να γίνει μία αυτόματη αναδιαμόρφωση πλαισίου (π.χ. αύξηση της φωτεινότητας της οθόνης, άδεια έναρξης ασύρματης σύνδεσης) ή να επιτραπεί μία κοντινή, γεωγραφική ή άλλη, επιλογή (π.χ. επισήμανση σημείων ενδιαφερόντων γεωγραφικά τοποθετημένα κοντά στο χρήστη ή η εκτύπωση ενός αρχείου στον πλησιέστερο εκτυπωτή).

Ένας προγραμματιστής εφαρμογών, για να επωφεληθεί από μία τέτοια λειτουργικότητα, πρέπει να έχει πρόσβαση σε μέσα απόκτησης, αποθήκευσής, φιλτραρίσματός και συνδυασμού δεδομένων πλαισίου και να εκτελεί εντολές βασισμένες στις πληροφορίες που προκύπτουν. Για διάχυτες, κατανεμημένες εφαρμογές επίγνωσης πλαισίου, ο στόχος είναι η παροχή κατάλληλου ενδιάμεσου λογισμικού που μπορεί να εκτελέσει Απομακρυσμένες Διαδικαστικές Κλήσεις (Remote Process Calls, RPCs), ενώ παράλληλα εισάγει ένα στρώμα αφαίρεσης που διευκολύνει την διαδικασία ανάπτυξης λογισμικού, αποκρύπτοντας την ετερογένεια των δικτυακών περιβαλλόντων, υποστηρίζοντας προηγμένα συνεργατικά μοντέλα μεταξύ κατανεμημένων οντοτήτων και καθιστώντας όσο το δυνατόν πιο διαφανή την υπολογιστική κατανομή. Το webinos στοχεύει στην παροχή ενός πολλαπλών πλατφορμών επίπεδο αφαίρεσης για κλήσεις RPC, αλλά την ίδια στιγμή, ενσωματώνει ένα επιπρόσθετο στρώμα αφαίρεσης δεδομένων για χρήση σε εφαρμογές επίγνωσης και ανταλλαγής πλαισίου τρίτων μερών που είναι εμπλουτισμένες με τις δυνατότητες της πλατφόρμας.

Οι βασικοί στόχοι που αφορούν στη συλλογή των δεδομένων, στην αποθήκευση και στην ανάκτηση πλαισίου είναι οι εξής:

- Προσδιορισμός δεδομένων πλαισίου σχετικά με την λειτουργία του webinos
- Ανίχνευση δεδομένων πλαισίου στο webinos (μέσω των οντοτήτων του webinos) και τις σχετιζόμενες πηγές πλαισίου
- Απόκτηση δεδομένων πλαισίου
- Αναπαράσταση δεδομένων πλαισίου στο webinos
- Αποθήκευση δεδομένων πλαισίου στο webinos
- Κατανομή δεδομένων πλαισίου μεταξύ των οντοτήτων του webinos

Η συλλογή δεδομένων πλαισίου μπορεί να πραγματοποιηθεί με τρεις τρόπους, Παρακολούθηση και Αναχαίτιση Απομακρυσμένων Διαδικαστικών Κλήσεων (RPC), μία Υπηρεσία Πλαισίου και Αντικείμενα Εφαρμογών Πλαισίου. (51)

5.1.1. Παρακολούθηση και Αναχαίτιση RPC

Έχει δημιουργηθεί ένας αυτόματος μηχανισμός, ο οποίος, με την άδεια του χρήστη μέσω του Διαχειριστή Πολιτικής (Policy Manager), μπορεί να αναχαιτίσει RPCs που πραγματοποιούνται από τις εμπλουτισμένες με τις δυνατότητες του webinos εφαρμογές στα διάφορα webinos APIs. Ο Διαχειριστής Πλαισίου (Context Manager) μετατρέπει το μήνυμα αυτό σε ένα *Αντικείμενο Πλαισίου*, επιλέγοντας πεδία του RPC που έχουν οριστεί ότι περιέχουν πληροφορία πλαισίου και δομώντας τα ως Αντικείμενα Πλαισίου, μέσω ενός εκτεταμένου λεξιλογίου πλαισίου των APIs (Context API Vocabulary). Το λεξιλόγιο αυτό είναι μία λίστα δομών και κανόνων για την αυτόματη εξαγωγή πλαισίου από τα παρακολουθούμενα RPC μηνύματα. Ακολουθώντας μία απλούστευση της δομής του περιεχομένου των RPC, παρόμοια με τον JSON-WSP ορισμό των κλήσεων αυτών, προβλέπει τα πεδία και τις ιδιότητες αυτών που εμφανίζονται στο καθένα και τα αντιστοιχίζει, όπου προκύπτουν, με τα ανάλογα πεδία σε Αντικείμενα Πλαισίου, τα οποία ορίζονται με παρόμοιο τρόπο. Έτσι, το Context API Vocabulary αποτελεί μία ευέλικτης μορφής γλώσσα περιγραφής διεπαφών (Interface Description Language) μεταξύ της κάθε RPC κλήσης του κάθε API του webinos που τις πραγματοποιεί προς οποιαδήποτε κατεύθυνση, και του κάθε συσχετιζόμενου με αυτές Αντικειμένου Πλαισίου.

5.1.2. Υπηρεσία Πλαισίου Φόντου (Context Service)

Τα Αντικείμενα Πλαισίου μπορούν να εγγραφούν για μία περιοδική συλλογή δεδομένων φόντου όταν το Πληρεξούσιο Προσωπικής Ζώνης (PZP) εκτελείται. Ορίζοντας Κανόνες Πλαισίου για την αποθήκευση των Αντικειμένων Πλαισίου, το διάστημα δειγματισμού, τις προϋποθέσεις και τις μεθόδους συλλογής, η Υπηρεσία Πλαισίου Φόντου πραγματοποιεί δειγματοληπτικές κλήσεις στα APIs που σχετίζονται μέσω του λεξιλογίου πλαισίου των APIs με τα *Αντικείμενα Πλαισίου* που στοχεύει να

συλλέξει, και χρησιμοποιώντας τον ίδιο μηχανισμό με την Παρακολούθηση και Αναχαίτηση RPC να αποκτήσει τα δεδομένα που περιέχονται σε αυτές τις κλήσεις.

5.1.3. Αντικείμενα Πλαισίου Εφαρμογών

Αντικείμενα Πλαισίου μπορούν, εκτός από αυτά που ήδη ορίζονται για τα APIs του webinos, να οριστούν και να αποθηκευτούν ανεξάρτητα από οποιαδήποτε εφαρμογή, χρησιμοποιώντας τις υπηρεσίες του API Πλαισίου. Μία εφαρμογή μπορεί να αιτηθεί την εγγραφή ενός νέου προσαρμοσμένου Αντικειμένου Πλαισίου και να αιτηθεί άδεια από τον Διαχειριστή Πολιτικής Δικαιωμάτων να ξεκινήσει να αποθηκεύει αυτά τα Αντικείμενα στη Βάση Δεδομένων Πλαισίου. Η εφαρμογή μπορεί να προσδιορίσει τη δική της διαδικασία απόκτησης δεδομένων αυτών των Αντικειμένων καθώς επίσης τη συχνότητα και τη διάρκεια ζωής τους. Ένας προγραμματιστής εφαρμογών μπορεί να χρησιμοποιήσει το Λεξιλόγιο Πλαισίου Εφαρμογών (Application Context Vocabulary) του webinos, πρόσβαση στο οποίο παρέχεται μέσω του API πλαισίου, για να ορίσει αυτούς τους προσαρμοσμένους κανόνες και δομές για την αποθήκευση Αντικειμένων Πλαισίου που βασίζονται σε δεδομένα που προέρχονται απευθείας από εφαρμογές, ή αυτών που εξάγονται από οποιαδήποτε διαδικασία η συνδυασμό προϋπαρχόντων ή νέων δεδομένων πλαισίου. Αποθηκεύοντας τους ορισμούς των αντικειμένων αυτών χρησιμοποιώντας τους ίδιους συντακτικούς κανόνες με το λεξιλόγιο πλαισίου των APIs, μπορούν να αποθηκεύονται νέα αντικείμενα, ισότιμα αυτών που προέρχονται από τα APIs στη Βάση Δεδομένων Πλαισίου του webinos, με κλήσεις αποθήκευσης αυτών απευθείας από τις εφαρμογές προς το API Πλαισίου.

Η βάση δεδομένων όπου αυτά τα Αντικείμενα Πλαισίου αποθηκεύονται με ασφάλεια βρίσκεται στο εικονικό πληρεξούσιο της Προσωπικής Ζώνης ενός χρήστη (virtual PZP, v-PZP) και είναι προσβάσιμη μέσω ενός καναλιού επικοινωνίας που ανοίγει ο κόμβος της Προσωπικής Ζώνης (PZH). Η Βάση Δεδομένων Πλαισίου περιέχει δεδομένα από διάφορες συσκευές και εφαρμογές εντός μίας Προσωπικής Ζώνης και είναι μοναδική για κάθε από αυτές. Η διαδικασία ερωτήσεων στην βάση δεδομένων επιτυγχάνεται μέσω μίας εύχρηστης και αξιόπιστης εφαρμογής δόμησης ερωτήσεων, η οποία επιτρέπει την αντιμετώπιση της Βάσης Δεδομένων Πλαισίου, ως μία βελτιστοποιημένη Βάση Δεδομένων, εστιάζοντας στο κύριο δομικό της στοιχείο,

το Αντικείμενο Πλαισίου. Ο χρήστης-προγραμματιστής μπορεί να απευθύνει ερωτήσεις απευθείας στο API Πλαισίου, με τη δυνατότητα απόκτησης οποιουδήποτε τύπου Αντικειμένων Πλαισίου, που έχει δημιουργηθεί από οποιοδήποτε API, εφαρμογή ή συσκευή μέσα στην Προσωπική Ζώνη ενός χρήστη, πάντα με την προϋπόθεση ότι έχει τα ανάλογα δικαιώματα πρόσβασης από τον Διαχειριστή Πολιτικής (Policy Manager) του webinos.

5.2.Λειτουργίες του Διαχειριστή και του API Πλαισίου

Οι βασικές λειτουργίες του Διαχειριστή Πλαισίου μπορούν να συνοψιστούν στα παρακάτω σημεία:

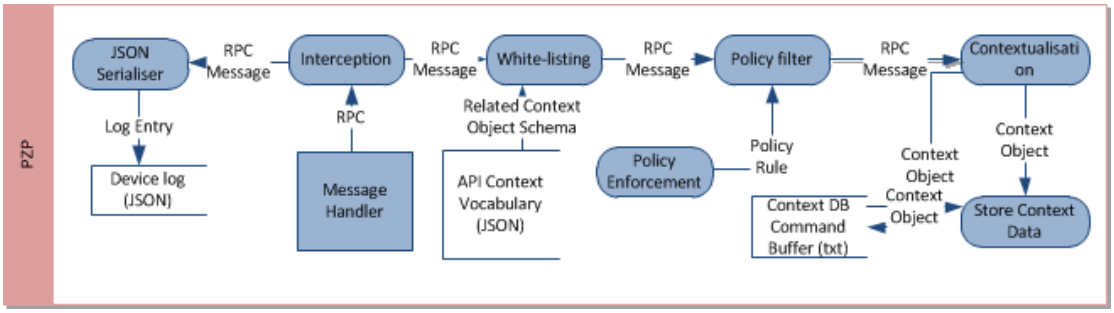
- Αυτόματα διατηρεί ένα αρχείο καταγραφής του περιεχομένου όλων των RPC κλήσεων, απογυμνωμένες από τις παραμέτρους και τα αποτελέσματα του PZP σε έναν τοπικά αποθηκευμένο JSON αρχείο ως ιστορικό.
- Περιέχει το API Context Vocabulary του webinos, το οποίο είναι μία δομημένη καταγραφή των APIs, των μεθόδων τους που εκτίθενται στο WRT, της αναμενόμενης δομής των αποτελεσμάτων τους, δομημένα ιεραρχικά κάτω από Αντικείμενα Πλαισίου που θα παρήγαγαν παραπλήσια πληροφορία πλαισίου (π.χ. το MyLocation είναι το ίδιο Αντικείμενο Πλαισίου ανεξάρτητα από το αν παράγεται από το getCurrentPosition, το watchPosition του Geolocation API ή από μία κλήση της GPS λειτουργίας του API του οχήματος).
- Παρακολουθεί και αυτόματα συλλαμβάνει RPC κλήσεις σε μεθόδους των APIs.
- Παρακολουθεί και αυτόματα συλλαμβάνει την καταγραφή των ακροατών των APIs (API Listeners) και παρακολουθεί τις κλήσεις που αυτοί επιστρέφουν, ούτως ώστε να τις αντιστοιχίσει με την κατάλληλη κλήση καταγραφής, δομώντας τες ως μοναδικές RPC κλήσεις. Οι ακροατές των APIs κατοχυρώνονται από τις εφαρμογές και λειτουργούν ώστε ασύγχρονα να λαμβάνονται αποτελέσματα όταν αυτά είναι διαθέσιμα από το webinos. Παράδειγμα αυτού αποτελεί το προαναφερθέν watchPosition, το οποίο κατοχυρώνεται από μία εφαρμογή, όπως μια εφαρμογή χαρτών, και όποτε ο αισθητήρας GPS έχει διαθέσιμο ένα γεωγραφικό στίγμα (περίπου μία φορά κάθε δευτερόλεπτο), το στίγμα αυτό επιστρέφεται στην εφαρμογή.
- Αυτόματα αναζητά κάθε RPC κλήση στο API Context Vocabulary, βρίσκει την αντιστοίχισή του με βάση το API που έκανε την κλήση, το όνομα της μεθόδου που

κλήθηκε και σε μερικές περιπτώσεις τις παραμέτρους που χρησιμοποιήθηκαν, για την αντιστοίχιση της κλήσης σε ένα Αντικείμενο Πλαισίου. Τα δεδομένα εισόδου και εξόδου της κλήσης δομούνται με βάση τους κανόνες του API Context Vocabulary, μη επιτρέποντας την καταγραφή δεδομένων όπως κωδικοί πρόσβασης και άλλες ευαίσθητες πληροφορίες που εξαιρούνται από οποιαδήποτε ερώτημα πλαισίου (context query) και δεν προκρίνονται στον Διαχειριστή Πολιτικής Δικαιωμάτων (Policy Manager).

- Τα Αντικείμενα Πλαισίου που δημιουργούνται από τις RPC κλήσεις αποθηκεύονται σε ένα αρχείο ενδιάμεσης αποθήκευσης (buffer) αν το εικονικό PZP που διατηρεί την Βάση Δεδομένων Πλαισίου δεν είναι συνδεδεμένο.
- Τα Αντικείμενα Πλαισίου που δημιουργούνται από RPC κλήσεις και αυτά που περιλαμβάνονται στον αρχείο ενδιάμεσης αποθήκευσης Αντικειμένων Πλαισίου στέλνονται στο εικονικό Πληρεξούσιο Προσωπικής Ζώνης (virtual PZP) ώστε να αποθηκευτούν στην Βάση Δεδομένων Πλαισίου.
- Επιτρέπει τον ορισμό προσαρμοσμένων Αντικειμένων Πλαισίου, απευθείας από το WRT (webinos Runtime), σε ένα ξεχωριστό Λεξιλόγιο Πλαισίου Εφαρμογών (Application Context Vocabulary) του webinos.
- Οι εφαρμογές μπορούν να αιτηθούν την αποθήκευση των δικών τους προσαρμοσμένων Αντικειμένων Πλαισίου στη Βάση Δεδομένων Πλαισίου που βρίσκεται στο εικονικό Πληρεξούσιο Προσωπικής Ζώνης (virtual PZP), χρησιμοποιώντας την ίδια δομή και αποθηκευτική θέση με όλα τα υπόλοιπα δεδομένα πλαισίου.
- Τα δεδομένα στη Βάση Δεδομένων Πλαισίου μπορούν να ερωτηθούν από το WRT χρησιμοποιώντας μία προσαρμοσμένη δομή εκτέλεσης ερωτημάτων Πλαισίου που επιτρέπει την περιγραφή απλών ή περίπλοκων ερωτημάτων που περιέχουν υποερωτήματα που σχετίζονται με Αντικείμενα Πλαισίου, αντιμετωπίζοντας τη βάση δεδομένων ως περιγραφική των Μοντέλων Αντικειμένων. Τα αποτελέσματα επιστρέφονται ως Αντικείμενα Πλαισίου.
- Μπορούν να καταχωρηθούν Κανόνες Δειγματοληψίας (Context Polling Rules). Σε αυτούς, μια κλήση σε ένα API του webinos μπορεί να προγραμματιστεί σε προκαθορισμένα χρονικά διαστήματα και το αποτέλεσμα να αναχαιτίζεται αυτόματα από τον Διαχειριστή Πλαισίου.

- Μπορούν να καταγραφούν Ακροατές Γεγονότων Πλαισίου (Context Event Listener). Ένας κανόνας μπορεί να εφαρμοστεί για το πού και όταν μία συγκεκριμένη προϋπόθεση πληρείται και ένα Ερώτημα Πλαισίου επιστρέφει ένα αποτέλεσμα που μπορεί να πυροδοτήσει ένα γεγονός στην εφαρμογή που καταχώρησε τον ακροατή.
- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής Δικαιωμάτων (Policy Manager) δίνει άδειες στα APIs και τις εφαρμογές να αποθηκεύσουν Αντικείμενα Πλαισίου στην Βάση Δεδομένων Πλαισίου στο εικονικό PZP.
- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής Δικαιωμάτων δίνει άδειες στις εφαρμογές να εκτελούν Ερωτήματα Πλαισίου στην Βάση Δεδομένων Πλαισίου στο εικονικό PZP.
- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής Δικαιωμάτων δίνει άδειες στις εφαρμογές να δημιουργήσουν και να επιβάλλουν Κανόνες Πλαισίου (Context Rules).

Ο μηχανισμός Παρακολούθησης και Αναχαίτισης RPC εμφανίζεται στην Εικόνα 7. Όταν αποστέλλεται μία κλήση RPC, ο Διαχειριστής Πλαισίου που τις παρακολουθεί, αποθηκεύει ένα αντίγραφο επιλεγμένων RPCs στο αρχείο καταγραφής και ενεργοποιεί τον Μηχανισμό Δημιουργίας Πλαισίου. Εκτελείται ερώτημα στο Λεξιλόγιο Πλαισίου των APIs (Context API Vocabulary) και ανάλογα με τη μέθοδο και το API για το οποίο έγινε η κλήση, δημιουργείται το κατάλληλο Αντικείμενο Πλαισίου. Αν υπάρχει κανόνας πολιτικής που επιτρέπει την αποθήκευση του συγκεκριμένου Αντικειμένου Πλαισίου, αποστέλλεται είτε στο εικονικό PZP μέσω του PZH, είτε στο αρχείο ενδιάμεσης αποθήκευσης (buffer file), ανάλογα με την κατάσταση σύνδεσης, για να αποθηκευτεί στη Βάση Δεδομένων Πλαισίου. Όταν υπάρχει διαθέσιμη σύνδεση στο PZH και, κατά προέκταση, στο εικονικό PZP στο οποίο βρίσκεται η Βάση Δεδομένων Πλαισίου, η κλήση αποθήκευσης πραγματοποιείται απευθείας. Αν η σύνδεση αυτή δεν είναι διαθέσιμη εκείνη τη στιγμή, και το PZP θεωρείται ότι βρίσκεται σε κατάσταση παρθένας λειτουργίας (PZP Virgin mode), τα Αντικείμενα Πλαισίου αποθηκεύονται προσωρινά στο αρχείο ενδιάμεσης αποθήκευσης, έως ότου πραγματοποιηθεί σύνδεση, όπου το σύνολο των Αντικειμένων Πλαισίου που βρίσκονται στο αρχείο αυτό αποστέλλονται μαζικά στη Βάση Δεδομένων Πλαισίου και το αρχείο ενδιάμεσης αποθήκευσης καθαρίζεται από τα περιεχόμενά του.

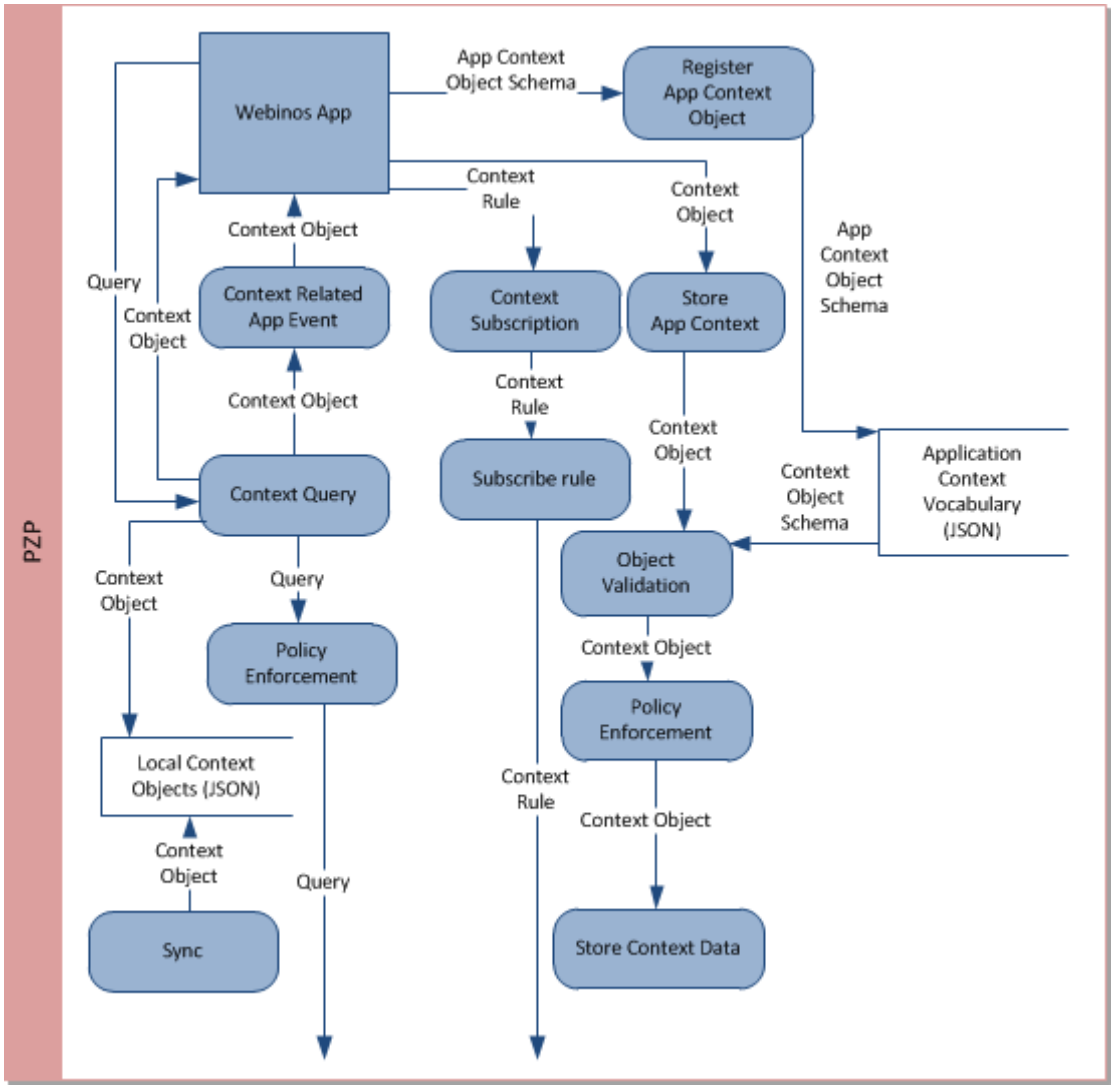


Εικόνα 7: Μηχανισμός Παρακολούθησης και Αναχαίτισης RPC

Η υπηρεσία Πλαισίου Εφαρμογών (Εικόνα 8) περιγράφεται σε ένα τοπικό επίπεδο αρχείο παρόμοιο με το Λεξιλόγιο Πλαισίου των APIs (Context API Vocabulary), το Λεξιλόγιο Πλαισίου Εφαρμογών (Application Context Vocabulary). Αν η εφαρμογή αιτείται δημιουργίας ενός νέου Αντικειμένου Πλαισίου Εφαρμογής, ή ανάκτησης ενός οποιουδήποτε Αντικειμένου Πλαισίου, καλείται ο Διαχειριστής Πολιτικής Δικαιωμάτων και αναλόγως τις άδειες που έχουν παραχωρηθεί στην εφαρμογή, το επιτρέπει ή όχι.

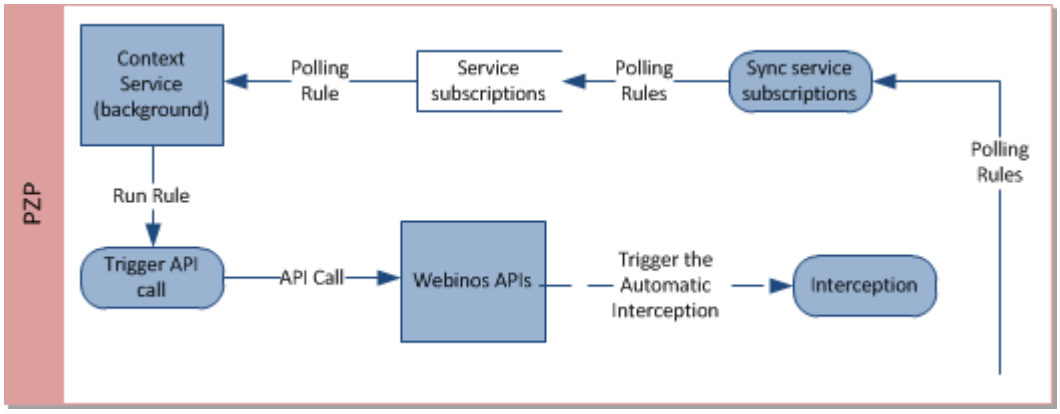
Στο ίδιο σχήμα, παρουσιάζεται ο μηχανισμός ερωτημάτων, όπου είτε όταν πυροδοτείται ένα γεγονός πλαισίου, είτε όταν μία εφαρμογή αιτείται δεδομένα μέσω ενός Ερωτήματος Πλαισίου, ερωτάται η Βάση Δεδομένων Πλαισίου, εφόσον το PZP είναι συνδεδεμένο στο PZH. Το αποτέλεσμα του ερωτήματος αυτού επιστρέφεται από το εικονικό PZP που φιλοξενεί τη Βάση Δεδομένων Πλαισίου, μέσω της κλήσης στο API Πλαισίου (Context API) που πραγματοποίησε το ερώτημα.

Επιπλέον, παρουσιάζεται ο μηχανισμός κατά τον οποίο εγγράφεται ένας Κανόνας Πλαισίου. Εφόσον η εφαρμογή έχει το δικαίωμα πρόσβασης στου κανόνες πλαισίου, έχει τη δυνατότητα να ορίσει και να αποθηκεύσει ένα αντικείμενο JSON στην Βάση Δεδομένων Πλαισίου, το οποίο περιέχει τις ρυθμίσεις για τις προϋποθέσεις πυροδότησης του κανόνα, με βάση τα Αντικείμενα Πλαισίου στα οποία έχει πρόσβαση, σύμφωνα με τον Διαχειριστή Πολιτικής Δικαιωμάτων, για την περίπτωση της καταχώρησης κανόνα πυροδότησης, ή του ρυθμού δειγματοληψίας, του Αντικειμένου Πλαισίου που θα καταγράφεται και των περιορισμών αυτού (όπως καταγραφή μόνο όταν μία τιμή είναι πάνω από ένα όριο, ή διαφορετική κατά κάποιο ποσό/ποσοστό από την ή τις προηγούμενες, τις συσκευές ή το είδος των συσκευών από τις οποίες θα προέρχονται τα δεδομένα κ.α.), στην περίπτωση καταχώρησης κανόνα δειγματισμού από την Υπηρεσία Πλαισίου Φόντου.



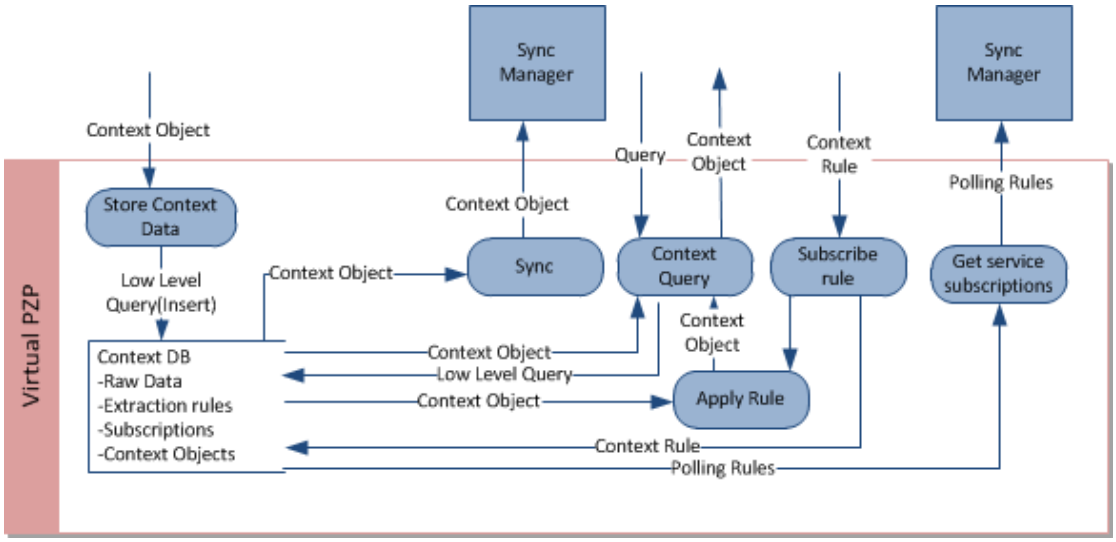
Εικόνα 8: Υπηρεσία Πλαισίου Εφαρμογών, Ερωτήματα Πλαισίου και Εγγραφή Κανόνων Πλαισίου

Η Υπηρεσία Πλαισίου Φόντου (Εικόνα 9) χειρίζεται την εκτέλεση των κανόνων δειγματοληψίας. Η υπηρεσία πυροδοτεί κλήσεις σε APIs κατά περιόδους, ακρίβεια, προϋποθέσεις και σε συγκεκριμένες συσκευές (PZP) που περιγράφονται από τους κανόνες, και έτσι πυροδοτείται επίσης ο αυτόματος μηχανισμός Παρακολούθησης και Αναχαίτισης RPC στην αντίστοιχη συσκευή. Αυτοί οι κανόνες, έχουν καταγραφεί στην Βάση Δεδομένων Πλαισίου στο εικονικό PZP και συγχρονίζονται με τις συσκευές στις οποίες είναι εφαρμόσιμοι, σε έναν επίπεδο αρχείο JSON συνδρομών υπηρεσίας (Service Subscriptions) που βρίσκεται σε αυτές. Το αρχείο αυτό αποτελεί ένα σύνολο ρυθμίσεων για την Υπηρεσία Πλαισίου Φόντου του κάθε PZP, το οποίο ερωτάται ανά τακτά χρονικά διαστήματα από την Υπηρεσία Πλαισίου Φόντου για την ενημέρωση των εσωτερικών παραμέτρων/χρονοδιαγραμμάτων/κανόνων στην υπηρεσία χρόνου εκτέλεσης.



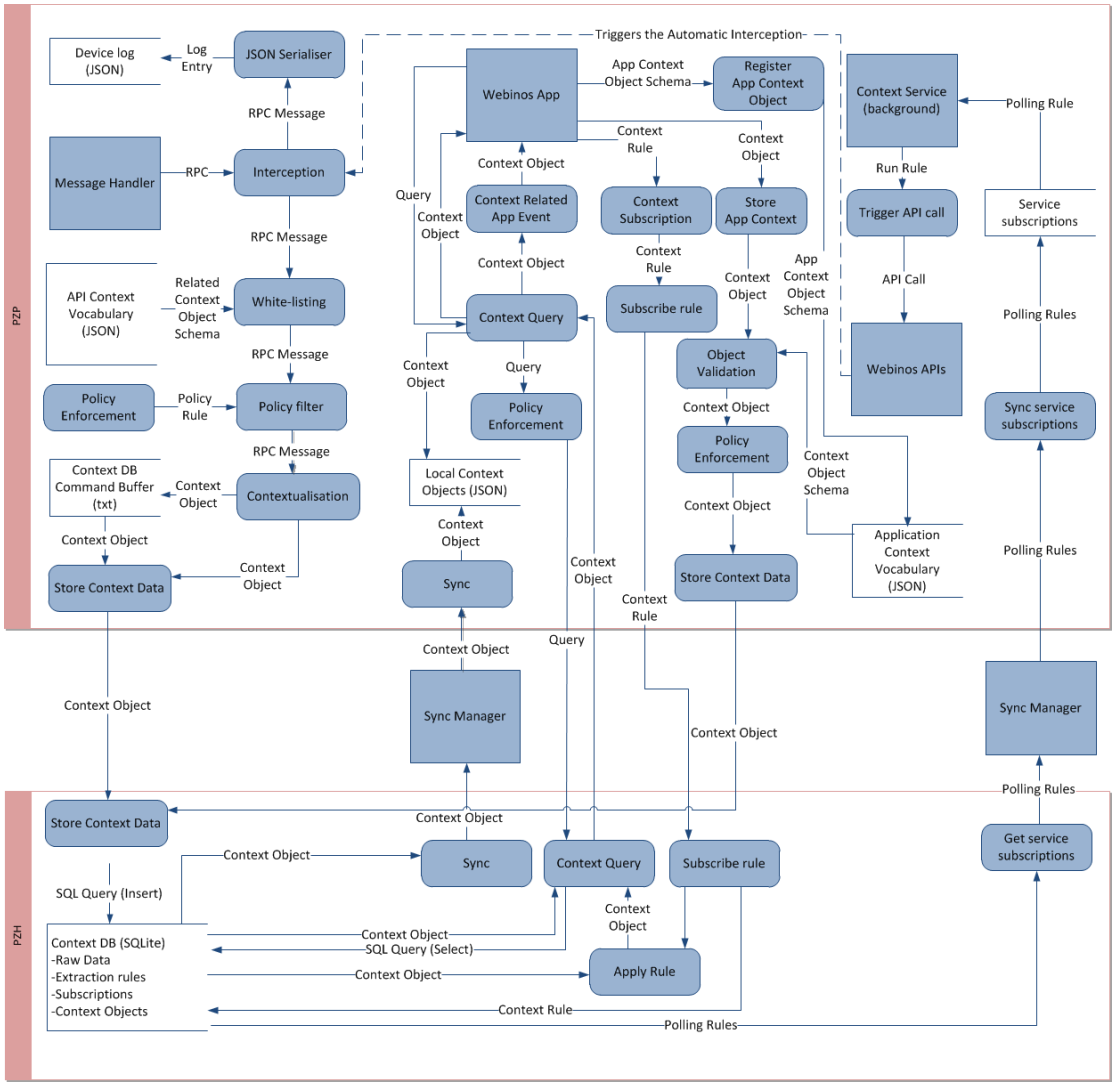
Εικόνα 9: Υπηρεσία Πλαισίου Φόντου

Το εικονικό PZP, το οποίο φιλοξενεί τη Βάση Δεδομένων Πλαισίου συνδέεται μέσω του PZH σε κάθε PZP της Προσωπικής Ζώνης. Η Υπηρεσία Πλαισίου Εικονικού PZP απεικονίζεται στην Εικόνα 10. Οι Κανόνες Πλαισίου, τα Αντικείμενα Πλαισίου, οι Συνδρομές Πλαισίου και τα ακατέργαστα Αντικείμενα Πλαισίου από την παρακολούθηση και την αναχαίτιση των RPC αποθηκεύονται και ανακτώνται εκεί. Ο Διαχειριστής Συγχρονισμού (Sync Manager) του webinos μπορεί να δρομολογήσει τα δεδομένα που πρέπει να αποθηκευτούν τοπικά σε κάθε αντίστοιχο PZP. Αυτά περιλαμβάνουν κανόνες συνδρομής, κανόνες δειγματοληψίας και τοπικά αποθηκευμένα Αντικείμενα Πλαισίου. Η Υπηρεσία Πλαισίου Εικονικού PZP αναλαμβάνει να μεταφράσει το κάθε ερώτημα πλαισίου σε γλώσσα ερωτημάτων χαμηλού επιπέδου για την Βάση Δεδομένων, καθώς και να μεταφράσει την απάντηση από τη βάση σε δομημένα, βάσει των κανόνων και των περιορισμών που έχουν τεθεί κατά την κλήση, Αντικείμενα Πλαισίου.



Εικόνα 10: Υπηρεσία Πλαισίου Εικονικού PZP και Βάση Δεδομένων Πλαισίου

Συνολικά, ο Διαχειριστής Πλαισίου, η σύνδεση μέσω του PZH με το εικονικό PZP και οι λειτουργίες του API πλαισίου εμφανίζεται στην ροή δεδομένων στην Εικόνα 11.



Εικόνα 11: Ροή δεδομένων Διαχειριστή Πλαισίου και API Πλαισίου

5.3.API του webinos που μπορούν να καταγραφούν

Τα API του webinos που περιλαμβάνονται στον αυτόματο μηχανισμό Διαχειριστή Πλαισίου για την καταγραφή πληροφοριών πλαισίου μέσω Παρακολούθησης RPC έχει αποφασιστεί να περιλαμβάνουν τα παρακάτω, έχοντας εξετάσει προσεκτικά διάφορους παράγοντες (75). Τα APIs δεν θα πρέπει να περιλαμβάνουν υπερβολικά ευαίσθητες και προσωπικές πληροφορίες, δεν θα πρέπει να παρεμβαίνουν σε ευαίσθητες λειτουργίες ασφάλειας και πολιτικής δικαιωμάτων και

θα πρέπει να ελαχιστοποιούν την ποσότητα αποθηκευμένων δεδομένων ανάλογα με την σχετική αξία του πλαισίου και την πιθανή του χρήση.

1. **Περιεχόμενο Πολυμέσων (Media Content)**: Αυτό το API παρέχει πρόσβαση στο περιεχόμενο των πολυμέσων και σε σχετικές πληροφορίες. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τα περισσότερα δεδομένα που περιέχουν την εμφάνιση ή την αναπαραγωγή των πολυμέσων σε όλες τις συσκευές, καθώς και μέτα-δεδομένων των αρχείων αυτών, αλλά όχι το ίδιο το περιεχόμενο των πολυμέσων.
2. **Γενικός Μηχανισμός Πρόσβασης (Generic Actuator):** Το API του Γενικού Μηχανισμού Πρόσβασης του webinos παρέχει στις εφαρμογές ένα API για να ελέγχουν τους μηχανισμούς πρόσβασης στην ίδια τη συσκευή, είναι συνδεδεμένοι με την συσκευή, ή βρίσκονται σε κάποια άλλη συσκευή. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τα γεγονότα που σχετίζονται με τον τύπο της πρόσβασης που καλείται και την ενέργεια που εκτελείται.
3. **Ειδοποιήσεις Ιστού (Web Notifications):** Οι προδιαγραφές των ειδοποιήσεων διαδικτύου του webinos παρέχουν ένα API για την εμφάνιση ειδοποιήσεων στους χρήστες εκτός του ορίου μιας διαδικτυακής σελίδας (όπως για παράδειγμα το σύστημα ειδοποιήσεων χρήστη του Android). Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τον τύπο του μηνύματος καθώς και το μήνυμα που εμφανίζεται, αν δεν έχει επισημανθεί ως απόρρητο.
4. **Αλληλεπίδραση Συσκευών (Device Interaction):** Αυτή η μονάδα παρέχει έναν μηχανισμό αλληλεπίδρασης με τον τελικό χρήστη μέσω χαρακτηριστικών όπως η δόνηση της συσκευής, ο ειδοποιητής της συσκευής, ο οπίσθιος φωτισμός της συσκευής, η ταπετσαρία της οθόνης. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει όλα τα γεγονότα που περιλαμβάνουν ρυθμίσεις όπως η αλλαγή του επιπέδου του οπίσθιου φωτισμού της οθόνης και αποκλείει άσχετα δεδομένα όπως η ενεργοποίηση της δόνησης της συσκευής σε συγκεκριμένη στιγμή.
5. **Κατάσταση Συσκευής (Device Status):** Αυτό το API χρησιμοποιεί ένα είδος μοντέλου δέντρου για να επιτρέψει στους προγραμματιστές να πάρουν μερικές πληροφορίες σχετικά διάφορα χαρακτηριστικά κατάστασης, λογισμικού ή υλικού της συσκευής. Οι πληροφορίες που ανακτώνται και τα δεδομένα που αιχμαλωτίζονται από τον Διαχειριστή Πλαισίου περιλαμβάνουν την κατάσταση της μπαταρίας, την κατάσταση της ασύρματης συνδεσιμότητας, την κατάσταση του τηλεφώνου κ.α.

6. **Πλοήγηση (Navigation)**: Το API πλοήγησης του webinos παρέχει τον μηχανισμό αλληλεπίδρασης με το λογισμικό πλοήγησης του αυτοκινήτου. Το API εκθέτει την λειτουργικότητα αλληλεπίδρασης με μία δορυφορική υπηρεσία πλοήγησης. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει γεγονότα πλοήγησης που περιλαμβάνουν την επιλογή ενός σημείου ενδιαφέροντος ως προορισμό και διευθύνσεις συγκεκριμένων προορισμών.
7. **Γενικός Αισθητήρας (Generic Sensor):** Το API του Γενικού Αισθητήρα του webinos παρέχει στις εφαρμογές διαδικτύου ένα API για να έχουν πρόσβαση σε δεδομένα αισθητήρων της ίδιας ή μιας άλλης συσκευής. Καθώς αυτό το API δεν γνωρίζει τις χαμηλού επιπέδου μεθόδους για την ανακάλυψης αισθητήρων και την επικοινωνία μαζί τους, ο Διαχειριστής Πλαισίου αιχμαλωτίζει δεδομένα για οποιονδήποτε αισθητήρα για τον οποίο έχει γίνει μία κλήση.
8. **Χειριστήριο Τηλεόρασης (TVControl):** Η διεπαφή αυτή παρέχει τα μέσα απόκτησης μίας λίστας πηγών τηλεόρασης, των καναλιών και των μεταδόσεων τους. Οι τηλεοπτικές μεταδόσεις καναλιών μπορούν να εμφανιστούν σε ένα αντικείμενο HTMLVideoElement. Εναλλακτικά, το API παρέχει μέσα για τον έλεγχο της διαχείρισης των καναλιών από το τοπικό λογισμικό της τηλεόρασης, επιτρέποντας την επιλογή ενός καναλιού ή την παρακολούθηση για αλλαγές καναλιών που προκαλούνται με άλλο τρόπο. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τα γεγονότα αλλαγής καναλιών και των αλλαγών στις πηγές και ροών πολυμέσων.
9. **Όχημα (Vehicle):** Το API οχήματος του webinos παρέχει πρόσβαση σε συγκεκριμένα δεδομένα που αφορούν στο όχημα, και τα οποία αιχμαλωτίζονται από το Διαχειριστή Πλαισίου, μέσω γεγονότων καταγραφής των ενεργοποιήσεων των διεπαφών του οχήματος. Αυτά τα γεγονότα περιλαμβάνουν την κατάσταση των καυσίμων, τη θέση του τιμονιού, τις αλλαγές ταχυτήτων, τις ρυθμίσεις κλίματος, τα φώτα, τη θέση του αισθητήρα στάθμευσης, την κατάσταση του υαλοκαθαριστήρα, την κατάσταση των λαδιών κινητήρα και τη θέση των καθισμάτων.

Τα APIs που αναφέρονται στις προδιαγραφές της W3C, τα οποία χρησιμοποιούνται από το webinos και καταγράφονται από τον Διαχειριστή Πλαισίου του είναι:

1. **Γεγονός Προσανατολισμού Συσκευής (DeviceOrientation Event):** Αυτός ο προσδιορισμός ορίζει DOM (Device Orientation and Motion) γεγονότα που παρέχουν πληροφορίες για τον φυσικό προσανατολισμό και την κίνηση μιας συσκευής. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τα γεγονότα που περιλαμβάνουν αλλαγές στον προσανατολισμό της συσκευής, όταν αυτές καλούνται από μία εφαρμογή.
2. **Δέσμευση Πολυμέσων και Ροών Πολυμέσων (Media Capture and Streams):** Αυτές οι προδιαγραφές ορίζουν ένα σύνολο από JavaScript APIs που επιτρέπουν στα τοπικά πολυμέσα, συμπεριλαμβανομένων ήχου και βίντεο, να ζητηθούν από μία πλατφόρμα. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει τα περισσότερα από τα δεδομένα που περιλαμβάνουν την απεικόνιση ή την αναπαραγωγή των πολυμέσων σε όλες τις συσκευές, αλλά όχι και το ίδιο το περιεχόμενο αυτών.
3. **Γεωγραφική τοποθεσία (Geolocation):** Αυτός ο προσδιορισμός ορίζει ένα API που παρέχει πρόσβαση σεναρίων (scripting) σε πληροφορίες γεωγραφικής τοποθεσίας που σχετίζονται με την συσκευή, είτε μέσω ενός δέκτη GPS, μίας διεύθυνσης IP, ενός RFID, της φυσικής διεύθυνσης (MAC) ενός ασύρματου δικτύου WiFi ή μίας συσκευής Bluetooth και κωδικών κυψελών GSM/CDMA, καθώς και μέσω εισόδου του χρήστη. Ο Διαχειριστής Πλαισίου αιχμαλωτίζει γεγονότα που σχετίζονται με την τωρινή τοποθεσία, ταχύτητα, υψόμετρο, κατεύθυνση και ακρίβεια των δεδομένων.

5.4.Αποθήκευση Αντικειμένων Πλαισίου

Τα Αντικείμενα Πλαισίου αποθηκεύονται με ασφάλεια αρχικά στην Βάση Δεδομένων Πλαισίου που βρίσκεται στο εικονικό PZP και είναι προσβάσιμο μέσω ενός καναλιού που ανοίγει το PZH. Ένα εικονικό PZP είναι ένα πολύ απλοποιημένο PZP χωρίς WRT, αλλά εκθέτει μόνο συγκεκριμένες υπηρεσίες του webinos. Το εικονικό PZP, που διατηρεί τη Βάση Δεδομένων Πλαισίου του webinos, εκθέτει τις υπηρεσίες μέσω του API Πλαισίου που σχετίζονται με λειτουργίες δημιουργίας, ανάγνωσης, επικαιροποίησης και διαγραφής (CRUD), καθώς και πιο περίπλοκες ενέργειες μέσω του API Πλαισίου. Η Βάση Δεδομένων Πλαισίου περιέχει δεδομένα από όλες τις συσκευές και τις εφαρμογές μέσα σε μία Προσωπική Ζώνη και κάθε βάση δεδομένων είναι μοναδική για αυτή την Προσωπική Ζώνη. Τα ερωτήματα στη Βάση Δεδομένων

Πλαισίου επιτυγχάνονται με ένα εύχρηστο μηχανισμό δόμησης ερωτημάτων (query builder) ειδικά για το API Πλαισίου.

5.5.Λεξιλόγιο Πλαισίου

Το Λεξιλόγιο API Πλαισίου βασίζεται σε μία δομή που ορίζει τα Αντικείμενα Πλαισίου σε σχέση με τα APIs στα οποία ανήκουν. Υπό την έννοια αυτή, η δομή ξεκινάει από τον ορισμό του API και τα δεδομένα που απαιτούνται για την αναγνώρισή του, έπειτα το όνομα του Αντικειμένου Πλαισίου που καταγράφεται. Τελικά, η δομή των μεθόδων και των δεδομένων σχετίζουν είτε την καταγραφή των δεδομένων ως μέρος ενός Αντικειμένου Πλαισίου, είτε τις αναμενόμενες αξίες για την επικύρωση μιας συγκεκριμένης μεθόδου. Ένα γενικό παράδειγμα αυτής της δομής βρίσκεται στην Εικόνα 12.

Το Λεξιλόγιο API Πλαισίου διατηρεί το όνομα του API από το οποίο αιχμαλωτίστηκε το Αντικείμενο Πλαισίου, το URI αυτού του API, ούτως ώστε να αναγνωρίσει τη μέθοδο που καλέστηκε, και τις μεθόδους που περιγράφουν το συγκεκριμένο Αντικείμενο Πλαισίου. Όπως με όλες τις κλήσεις μεθόδων του webinos, περιλαμβάνουν τις εισόδους, τις εξόδους και τα λάθη που παράγονται από τη μέθοδο. Το όνομα των μεθόδων προσδιορίζεται από τα τιμή του δεδομένου ObjectName, ενώ τα αντικείμενα εντός των “values” περιγράφονται με την ίδια δομή των αντικειμένων που εμφανίζονται στα RPCs αυτών των κλήσεων. Οποιαδήποτε είσοδος δεν πρέπει να αποθηκευτεί φέρει τη σημαία "logged" : false.

```
{ "APIname" : "Name of API",
  "URI" : "http://www.webinos.org/APIName",
  "ContextObjects" : [{
      "objectName" : "MyContextObject",
      "methods" : [{
          "Errors" : [{
              "logged" : false,
              "objectName" : "Error",
              "type" : "object",
              "values" : [{
                  "logged" : false,
                  "objectName" : "Error Field",
                  "type" : "data type"}]
      }],
      "inputs" : [{
          "logged" : true,
          "objectName" : "",
          "required" : false,
          "type" : "array",
          "values" : [{
              "logged" : true,
              "objectName" : "",
              "required" : true,
              "type" : "Object",
              "values" : [{
                  "logged" : false,
                  "objectName" : "timeout",
                  "type" : "long"}]
            }]
      }],
      "objectName" : "method name",
      "outputs" : [{
          "logged" : true,
          "objectName" : "output name",
          "type" : "object",
          "values" : [{
              "logged" : true,
              "objectName" : "An output value",
              "type" : "double"}]
        }]
    }],
  "otherMethods" : [{
      "Errors" : [],
      "inputs" : [],
      "objectName" : "A non context method",
      "outputs" : []}]}]}]
```

Εικόνα 12: Παράδειγμα του Λεξικού API Πλαισίου

Το κάθε Αντικείμενο Πλαισίου δομείται σε επίπεδο διεπαφής από το API Πλαισίου σύμφωνα με την περιγραφή στην Εικόνα 13.

WebIDL Specification

```
interface ContextObject{
      readonly attribute DOMString name;
      readonly attribute DOMString? apiUri;
      readonly attribute ApplicationURI? applicationUri;
      readonly attribute DOMString deviceId;
      readonly attribute DOMString sessionId;
      readonly attribute DOMTimeStamp timestamp;
      readonly attribute DOMString? version;
      readonly attribute ContextObjectValueArray values;
};
```

Εικόνα 13: Διεπαφή Αντικειμένου Πλαισίου

Το κάθε πεδίο που ορίζεται για κάθε Αντικείμενο Πλαισίου έχει δύο ιδιότητες, αυτές του ονόματος του πεδίου και του τύπου του (Εικόνα 14).

WebIDL Specification

```
interface ContextObjectField{
      attribute DOMString name;
      attribute DOMString type;
};
```

Εικόνα 14: Διεπαφή Πεδίου Αντικειμένου Πλαισίου

Αντίστοιχα δομείται και το κάθε αντικείμενο στο Λεξιλόγιο Πλαισίου Εφαρμογών. Εξαιτίας του ότι πρέπει να διασφαλίζεται η μοναδικότητα της κάθε εφαρμογής, δεν απαιτείται απλώς το όνομα της εφαρμογής, αλλά και ένα μοναδικό URI που την περιγράφει. Επιπλέον, εξαιτίας της πιθανότητας οι διαφορετικές εκδόσεις της εφαρμογής να οδηγήσουν σε ανομοιογένεια των δεδομένων, περιέχεται και ο αριθμός έκδοσης της εφαρμογής (ή των αντικειμένων πλαισίου εφόσον ο προγραμματιστής επιθυμεί), ώστε να αποθηκεύεται μαζί με κάθε αντικείμενο πλαισίου και να δύναται να φιλτραριστεί ανάλογα σε αναζητήσεις. Ένα αφαιρετικό παράδειγμα μίας τέτοιας καταχώρησης στο Λεξιλόγιο Πλαισίου Εφαρμογών εμφανίζεται στην Εικόνα 15.

```
{
   "APPName" : "Name of Application",
   "ContextObjects" : [
      {
         "objectName" : "MyContextObject",
         "type" : "Object",
         "values" : [
            {
               "logged" : true,
               "objectName" : "A field",
               "type" : "data type"
            }
         ]
      }
   ],
   "URI" : "http://www.vendor.org/ APPName",
   "version" : "0.5.1"
}
```

Εικόνα 15: Παράδειγμα Καταχώρησης Λεξικού Πλαισίου Εφαρμογών

Η διαχείριση των Αντικειμένων Πλαισίου Εφαρμογών πραγματοποιείται μέσω του αντίστοιχου διαχειριστή, όπως εμφανίζεται στην Εικόνα 16.

WebIDL Specification

```
interface ApplicationContextManager{
    void registerSchema(ApplicationURI applicationUri, DOMString contextObjectName, D
OMString version, ContextObjectFieldArray fields, optional SuccessCB? successCallback, opt
ional ErrorCB? errorCallback);

    void addData(ApplicationURI applicationUri, DOMString contextObjectName, ContextO
bjectValueArray values, optional SuccessCB successCallback, optional ErrorCB? errorCallbac
k);
};
```

Εικόνα 16: Διαχειριστής Πλαισίου Εφαρμογών

Η μέθοδος καταχώρησης ενός νέου αντικειμένου στο Λεξικό Πλαισίου Εφαρμογών εμφανίζεται στην Εικόνα 17.

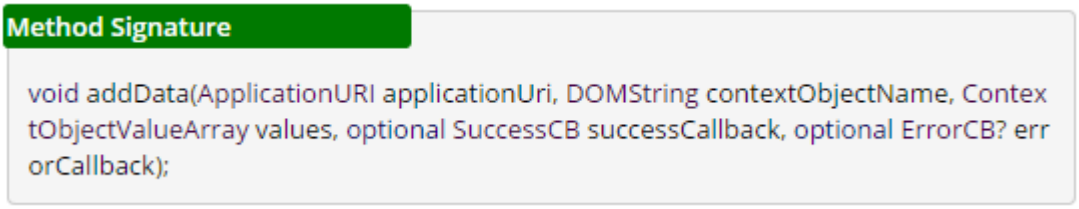
Method Signature

```
void registerSchema(ApplicationURI applicationUri, DOMString contextObjectName,
DOMString version, ContextObjectFieldArray fields, optional SuccessCB? successCall
back, optional ErrorCB? errorCallback);
```

Εικόνα 17: Μέθοδος καταχώρησης Αντικειμένου Πλαισίου Εφαρμογής

Όταν η εφαρμογή πρόκειται να προσθέσει δεδομένα, δηλαδή Αντικείμενα Πλαισίου, τα οποία έχουν οριστεί από την εφαρμογή, το πραγματοποιεί μέσω της

μεθόδου εισαγωγής Αντικειμένων Πλαισίου Εφαρμογών, η οποία εμφανίζεται στην Εικόνα 18Εικόνα 17.



Εικόνα 18: Μέθοδος εισαγωγής Αντικειμένων Πλαισίου Εφαρμογών

5.6.Ερωτήματα Πλαισίου

Ο τρόπος με τον οποίο πραγματοποιούνται τα ερωτήματα στα δεδομένα πλαισίου είναι μέσω ενός μηχανισμού ερωτημάτων που μεταφράζει τα υψηλού επιπέδου Ερωτήματα Πλαισίου σε χαμηλού επιπέδου ερωτήματα βάσης δεδομένων. Ο μηχανισμός δόμησης ερωτημάτων επιτρέπει την σύνταξη ερωτημάτων με βάση τους παρακάτω όρους (κριτήρια):

- **eq**: ισούται με
- **lt**: λιγότερο από
- **le**: λιγότερο ή ίσο με
- **gt**: μεγαλύτερο από
- **ge**: μεγαλύτερο ή ίσο με
- **starts**: ξεκινά με
- **ends**: τερματίζει σε
- **in**: ανήκει στη δοσμένη λίστα. Η τιμή πρέπει να είναι πίνακας.
- **contains**: η τιμή να περιέχει την δοσμένη τιμή, εφαρμόσιμο μόνο σε DOMString

Αυτοί οι όροι παρέχονται ως ένα επίπεδο API Αφαίρεσης Βάσεων Δεδομένων (API Database Abstraction) για κάθε υποστηριζόμενη Βάση Δεδομένων που εγκαθίσταται στο Εικονικό PZP στο οποίο βρίσκεται η Βάση Δεδομένων Πλαισίου. Αυτό το επίπεδο αφαίρεσης είναι προσβάσιμο μέσω του API Πλαισίου από κάθε εφαρμογή στην οποία έχει δοθεί δικαίωμα πρόσβασης, είτε αποθήκευσης, είτε ανάκτησης δεδομένων από την Βάση Δεδομένων Πλαισίου, δικαιώματα που παρέχονται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos. Σε δεύτερο επίπεδο, ο Διαχειριστής Πολιτικής Δικαιωμάτων ελέγχει κατά πόσο η συγκεκριμένη

ενέργεια που επιχειρείται να εκτελεστεί από την εφαρμογή έχει τα κατάλληλα δικαιώματα ως προς τα συγκεκριμένα Αντικείμενα Πλαισίου στα οποία ζητά πρόσβαση ή δικαίωμα εγγραφής.

WebIDL Specification

```
interface Context : Service {
    readonly attribute ApplicationContextManager app;

    readonly attribute ContextRuleManager rules;

    readonly attribute ScheduleManager schedule;

    void executeQuery(ContextQuery query, ContextQuerySuccessCB successCallback, opti
onal ContextQueryErrorCB? errorCallback);

    void schemaExists(DOMString uri, boolean uriIsApplication, DOMString contextObjectN
ame, SchemaExistsCB callback, optional ErrorCB? errorCallback);
};
```

Εικόνα 19: Διεπαφή Πλαισίου

Στην Εικόνα 19 εμφανίζεται η διεπαφή πλαισίου, η οποία αποτελεί προέκταση της διεπαφής υπηρεσιών της μονάδας αναζήτησης υπηρεσιών του webinos (Service Discovery). Η διεπαφή πλαισίου αποτελεί το σημείο από το οποίο εκτελούνται ερωτήματα στα δεδομένα που είναι αποθηκευμένα στην Βάση Δεδομένων Πλαισίου.

Method Signature

```
void schemaExists(DOMString uri, boolean uriIsApplication, DOMString contextObjec
tName, SchemaExistsCB callback, optional ErrorCB? errorCallback);
```

Εικόνα 20: Επιβεβαίωση ύπαρξης σχήματος

Για να είναι εφικτή η αναζήτηση στη Βάση Δεδομένων Πλαισίου για συγκεκριμένα αντικείμενα πλαισίου, παρέχεται η δυνατότητα αναζήτησης σε αυτή, για την επιβεβαίωση της ύπαρξης αυτών (Εικόνα 20).

Method Signature

```
void executeQuery(ContextQuery query, ContextQuerySuccessCB successCallback, o
ptional ContextQueryErrorCB? errorCallback);
```

Εικόνα 21: Μέθοδος Εκτέλεσης Ερωτήματος Πλαισίου

Η μέθοδος για την εκτέλεση ενός ερωτήματος Πλαισίου εμφανίζεται στην Εικόνα 21. Η διεπαφή που περιγράφει το ερώτημα εμφανίζεται στην Εικόνα 22.

WebIDL Specification

```
interface ContextQuery{
    attribute QueryFilter filter;
    attribute Object? criteria;
    attribute Object? projection;
    attribute QueryModifier? modifier;
};
```

Εικόνα 22: Διεπαφή Ερωτήματος Πλαισίου

Η γλώσσα με την οποία περιγράφονται τα ερωτήματα πλαισίου βασίζεται στη mongoDB. Αυτός είναι ο λόγος για τον οποίο χρησιμοποιούνται προαιρετικά τα projections της γλώσσας, τα οποία επιτρέπουν τον περιορισμό των πεδίων που επιστρέφονται από τα Αντικείμενα Πλαισίου σε μόνο αυτά τα οποία έχουν νόημα για τη συγκεκριμένη κλήση. Τα κριτήρια με τα οποία πραγματοποιείται ο περιορισμός των επιστρεφόμενων αντικειμένων Πλαισίου καταχωρούνται στη μεταβλητή criteria.

WebIDL Specification

```
interface QueryFilter{
    attribute DOMString name;
    attribute DOMString? apiUri;
    attribute ApplicationURI? applicationUri;
    attribute (DOMString or object)? deviceId;
    attribute (DOMString or object)? sessionId;
    attribute (DOMString or object)? timestamp;
    attribute (DOMString or object)? version;
};
```

Εικόνα 23: Διεπαφή Φίλτρου Ερωτήματος Πλαισίου

Το φίλτρο ερωτήματος (Εικόνα 23) αποτελεί τη διεπαφή με την οποία περιγράφεται το Αντικείμενο Πλαισίου το οποίο πρόκειται να επιστραφεί. Το αποτέλεσμα αποτελεί πίνακα Αντικειμένων Πλαισίου και μπορεί να απαρτίζεται από ένα ή περισσότερα Αντικείμενα. Αν πρόκειται για αντικείμενο που προέρχεται από API, τότε πρέπει να δοθεί το URI αυτού του API και αν πρόκειται για Αντικείμενο Πλαισίου Εφαρμογής, θα πρέπει να δοθεί το URI της εφαρμογής. Αν τα αποτελέσματα θα πρέπει να περιοριστούν ως προς συγκεκριμένη συσκευή, συνεδρία, χρονοσφραγίδα

ή έκδοση (για τις εφαρμογές), αυτά μπορούν να παρασχεθούν ως συνθήκες του φίλτρου.

WebIDL Specification

```
interface QueryModifier{
    attribute integer? limit;
    attribute integer? skip;
    attribute object? sort;
};
```

Εικόνα 24: Τροποποιητής Ερωτήματος Πλαισίου

Ο τροποποιητής ερωτημάτων πλαισίου (Εικόνα 24) έχει ως σκοπό τον περιορισμό των Αντικειμένων Πλαισίου που επιστρέφονται από ένα ερώτημα, με σκοπό την εξοικονόμηση πόρων. Με αυτόν, περιορίζεται ο αριθμός των επιστρεφόμενων αντικειμένων, αγνοείται συγκεκριμένος αριθμός των πρώτων αποτελεσμάτων και τα αντικείμενα ταξινομούνται βάζει κριτηρίου που ορίζεται όπως στην αντίστοιχη εντολή της mongoDB.

5.7.Συνδρομές Πλαισίου

Οι συνδρομές πλαισίου είναι τα ερωτήματα πλαισίου τα οποία ορίζονται να εκτελούνται με συγκεκριμένο χρονικό προγραμματισμό και τα οποία διαθέτουν τα αποτελέσματά τους, μέσω της Βάσης Δεδομένων Πλαισίου σε όποιες εφαρμογές έχουν τα αντίστοιχα δικαιώματα πρόσβασης. Στην Εικόνα 25 εμφανίζεται ο Διαχειριστής Συνδρομών Πλαισίου, μέσω του οποίου γίνονται οι κλήσεις.

WebIDL Specification

```
interface ScheduleManager{
    void search(DOMString? apiUri, boolean onlyOwnSchedules, ScheduleSearchCB callbac
k, optional ErrorCB? errorCallback);

    void save(Schedule schedule, optional ScheduleSaveCB? saveCallback, optional ErrorC
B? errorCallback);

    void remove(Schedule schedule, optional SuccessCB? successCallback, optional ErrorC
B? errorCallback);

    void renew(Schedule schedule, optional SuccessCB? successCallback, optional ErrorCB?
 errorCallback);
};
```

Εικόνα 25: Διαχειριστής Συνδρομών Πλαισίου

Αν η συνδρομή συνοδεύεται από μία προϋπόθεση εκτέλεσης μέσω ενός κανόνα πλαισίου, το γεγονός δε θα πυροδοτήσει την εγγραφή στη Βάση Δεδομένων Πλαισίου. Για παράδειγμα, εάν η συνδρομή αφορά στη γεωγραφική τοποθεσία μίας συσκευής και κατά προέκταση και του χρήστη, κανόνας ο οποίος έχει οριστεί να καταγράφει την τοποθεσία του χρήστη, μία λογική προϋπόθεση είναι να μπορεί να πυροδοτήσει την εγγραφή νέας τοποθεσίας, μόνο εφόσον ο χρήστης μετακινήσει τη συσκευή περισσότερο από 100 μέτρα από την προηγούμενη τοποθεσία η οποία έχει καταγραφεί για αυτή.

Η αναζήτηση μίας συνδρομής πλαισίου πραγματοποιείται με την μέθοδο στην Εικόνα 26. Η εφαρμογή πρέπει να παρέχει το URI του API για το οποίο αναζητά τις συνδρομές, καθώς και την διευκρίνιση του κατά πόσο οι συνδρομές που αναζητούνται είναι μόνο αυτές οι οποίες έχουν δημιουργηθεί από την ίδια την εφαρμογή που τις αναζητά. Το επιστρεφόμενο αποτέλεσμα είναι τα αντικείμενα της διεπαφής συνδρομής πλαισίου (Εικόνα 30) που ικανοποιούν τα κριτήρια αυτά.

Method Signature

```
void search(DOMString? apiUri, boolean onlyOwnSchedules, ScheduleSearchCB callb
ack, optional ErrorCB? errorCallback);
```

Εικόνα 26: Αναζήτηση Συνδρομής Πλαισίου

Όταν μία εφαρμογή πρόκειται να δημιουργήσει μία καινούρια συνδρομή, το κάνει με τη χρήση της μεθόδου στην Εικόνα 27, παρέχοντας ένα αντικείμενο της διεπαφής συνδρομής πλαισίου προς αποθήκευση.

Method Signature

```
void save(Schedule schedule, optional ScheduleSaveCB? saveCallback, optional Error
CB? errorCallback);
```

Εικόνα 27: Μέθοδος Εισαγωγής Συνδρομής Πλαισίου

Η μέθοδος με την οποία μία εφαρμογή μπορεί να διαγράψει μία συνδρομή πλαισίου είναι αυτή που εμφανίζεται στην Εικόνα 28.

Method Signature

```
void remove(Schedule schedule, optional SuccessCB? successCallback, optional Error
CB? errorCallback);
```

Εικόνα 28: Αφαίρεση Συνδορμής Πλαισίου

Αντίστοιχα, εξαιτίας του ορισμού χρόνου ζωής της κάθε συνδρομής, και με σκοπό την εξασφάλιση οικονομίας πόρων, οι εφαρμογές που έχουν δημιουργήσει συνδρομές πλαισίου, ή χρησιμοποιούν κάποιες άλλες, θα πρέπει να ανανεώνουν τις συνδρομές αυτές, με τη χρήση της μεθόδου στην Εικόνα 29.

Method Signature

```
void renew(Schedule schedule, optional SuccessCB? successCallback, optional ErrorC
B? errorCallback);
```

Εικόνα 29: Ανανέωση Συνδρομής Πλαισίου

Εφόσον οι Συνδρομές Πλαισίου ορίζονται από εφαρμογές, κάθε συνδρομή συνοδεύεται από το URI της εφαρμογής που την αποθηκεύει (Εικόνα 30), καθώς και τις υπόλοιπες πληροφορίες οι οποίες είναι απαραίτητες ώστε ο Διαχειριστής Πλαισίου να γνωρίζει σε ποια συσκευή να εκτελέσει μία μέθοδο ενός API που αποθηκεύει αντικείμενο πλαισίου αυτόματα.

WebIDL Specification

```
interface Schedule{

    attribute ApplicationURI applicationUri;

    attribute DOMString apiUri;

    attribute DOMString method;

    attribute object[] params;

    attribute DOMString deviceId;

    attribute integer interval;

    attribute DOMTimeStamp validUntil;
};
```

Εικόνα 30: Διεπαφή Συνδρομής Πλαισίου

5.8.Κανόνες Πλαισίου

Οι Κανόνες Πλαισίου λειτουργούν με παρόμοιο τρόπο με τις Συνδρομές Πλαισίου ως προς τον τρόπο με τον οποίο δομούνται, με τη διαφορά ότι αντί να πραγματοποιούνται κλήσεις σε APIs με σκοπό την αποθήκευση Αντικειμένων Πλαισίου στη βάση, εκτελούνται ερωτήματα πλαισίου ανά συγκεκριμένα διαστήματα στη βάση και μόνο στην περίπτωση που το αποτέλεσμά τους ικανοποιεί την συνθήκη που έχει οριστεί στο ερώτημα ενημερώνουν την εφαρμογή που τον δημιούργησε ότι ο κανόνας πυροδοτήθηκε. Η χρήση των Κανόνων Πλαισίου πραγματοποιείται μέσω του Διαχειριστή Κανόνων Πλαισίου (Εικόνα 31), ο οποίος παρέχει τη δυνατότητα αναζήτησης, αποθήκευσης, αφαίρεσης, ανανέωσης των κανόνων, καθώς και της προσθήκης και αφαίρεσης ακροατών γεγονότων πλαισίου.

WebIDL Specification

```
interface ContextRuleManager{
    void search(ContextRuleURI contextRuleUri, ContextRuleSearchCB successCallback, op
tional ErrorCB? errorCallback);

    void save(ContextRule rule, optional ContextRuleSaveCB? saveCallback, optional ErrorC
B? errorCallback);

    void remove(ContextRuleURI contextRuleUri, optional SuccessCB? successCallback, opt
ional ErrorCB? errorCallback);

    void renew(ContextRuleURI contextRuleUri, optional SuccessCB? successCallback, opti
onal ErrorCB? errorCallback);


    void addListener(ContextRuleURI contextRuleURI, DOMString listenerName, ContextRu
leEventHandler callback, optional SuccessCB? successCallback, optional ErrorCB? errorCallb
ack);

    void removeListener(ContextRuleURI contextRuleURI, DOMString listenerName, option
al SuccessCB? successCallback, optional ErrorCB? errorCallback);
};
```

Εικόνα 31: Διαχειριστής Κανόνων Πλαισίου

Η αναζήτηση κανόνων πλαισίου (Εικόνα 32) δίνει τη δυνατότητα, χρησιμοποιώντας ως είσοδο το URI ενός κανόνα, να επιστρέφεται το αντικείμενο διεπαφής κανόνα πλαισίου (Εικόνα 36), το οποίο περιέχει τον ορισμό του κανόνα αυτού.

Method Signature

```
void search(ContextRuleURI contextRuleUri, ContextRuleSearchCB successCallback,
optional ErrorCB? errorCallback);
```

Εικόνα 32: Αναζήτηση Κανόνα Πλαισίου

Η μέθοδος με την οποία αποθηκεύεται ένας κανόνας πλαισίου εμφανίζεται στην Εικόνα 33.

Εικόνα 33: Μέθοδος αποθήκευσης Κανόνα Πλαισίου

Αντίστοιχα, η μέθοδος με την οποία διαγράφεται ένας κανόνας πλαισίου εμφανίζεται στην Εικόνα 34.

Εικόνα 34: Αφαίρεση Κανόνα Πλαισίου

Εξαιτίας της συνθήκης κατά την οποία ένας κανόνας πλαισίου έχει συγκεκριμένο χρόνο ζωής, οι εφαρμογή η οποία χρησιμοποιεί τον κανόνα θα πρέπει να ανανεώνει αυτό τον χρόνο, με τη χρήση της μεθόδου που εμφανίζεται στην Εικόνα 35.

Εικόνα 35: Ανανέωση Κανόνα Πλαισίου

Η διεπαφή με την οποία περιγράφεται ο κανόνας εμφανίζεται στην Εικόνα 36. Σε αυτήν ορίζεται το URI της εφαρμογής που ζητά την ενεργοποίηση του κανόνα, το ερώτημα που θα εκτελείται και το χρονικό διάστημα για το οποίο ο κανόνας θα είναι ενεργός.

WebIDL Specification

```
interface ContextRule{
    attribute ContextRuleURI contextRuleUri;

    attribute ApplicationURI applicationUri;

    attribute DOMString description;

    attribute ContextQuery ruleQuery;

    attribute integer interval;

    attribute DOMTimeStamp validUntil;
};
```

Εικόνα 36: Διεπαφή Κανόνα Πλαισίου

Αφού αποθηκευτεί ο κανόνας, η εφαρμογή η οποία πρόκειται να παρακολουθεί το κατά πόσο η συνθήκη ή οι συνθήκες που περιέχονται ικανοποιούνται πρέπει να δηλώσει τη δημιουργία ενός ακροατή γεγονότων πλαισίου του API Πλαισίου, ο οποίος πρόκειται να εκτελέσει την μέθοδο της εφαρμογής η οποία είναι ορισμένη να πυροδοτείται όταν ικανοποιηθεί ο κανόνας (Εικόνα 37).

Method Signature

```
void addListener(ContextRuleURI contextRuleURI, DOMString listenerName, Context
RuleEventHandler callback, optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);
```

Εικόνα 37: Μέθοδος Εισαγωγής Ακροατή

Όταν ο ακροατής γεγονότων πλαισίου δεν είναι πλέον απαραίτητος, αυτός μπορεί να αφαιρεθεί με τη χρήση της μεθόδου στην Εικόνα 38.

Method Signature

```
void removeListener(ContextRuleURI contextRuleURI, DOMString listenerName, opti
onal SuccessCB? successCallback, optional ErrorCB? errorCallback);
```

Εικόνα 38: Μέθοδος Αφαίρεσης Ακορατή

Η μέθοδος της εφαρμογής στη συνέχεια δέχεται ως είσοδο το αποτέλεσμα του ερωτήματος που περιέχεται στον κανόνα υπό τη μορφή Αντικειμένων Πλαισίου που επιστρέφονται από αυτόν. Το γεγονός αυτό περιέχει το URI του κανόνα που πυροδοτήθηκε, καθώς και τα επιστρεφόμενα αντικείμενα (Εικόνα 39).

WebIDL Specification

```
interface ContextRuleEvent {
    readonly attribute ContextRuleURI contextRuleUri;
    readonly attribute ContextObjectArray? result;
};
```

Εικόνα 39: Διεπαφή γεγονότος Ακροατή

5.9.Επιβολή Πολιτικής Δικαιωμάτων

Είναι σαφές ότι οι χρήστες δεν είναι πολύ καλοί στο να αντιληφθούν τη μελλοντική αξία του να διατηρούν τις προσωπικές τους πληροφορίες ιδιωτικές, και συχνά βιάζονται να μοιραστούν την ιδιοκτησία αυτής της πληροφορίας χωρίς να εκτιμούν τη συνέπεια των πιθανών χρήσεών της (50). Με αυτό κατά νου, η πλατφόρμα του webinos διασφαλίζει ότι η ιδιοκτησία των πληροφοριών πλαισίου παραμένει στο χρήστη, ενώ τα δικαιώματα πρόσβασης σε εφαρμογές για αποθήκευση, εξαγωγή ή αναζήτηση δεδομένων πλαισίου μπορούν να δοθούν από το χρήστη μόνο στην εφαρμογή και όχι στον προγραμματιστή αυτής της εφαρμογής. Αυτό επιτρέπει στον προγραμματιστή εφαρμογών του webinos να αναπτύξει εφαρμογές οι οποίες μπορούν να χρησιμοποιήσουν Αντικείμενα Πλαισίου που είναι αποθηκευμένα από την πλατφόρμα του webinos ή οποιαδήποτε άλλη εφαρμογή στην προσωπική ζώνη του χρήστη. Για να διασφαλιστεί περαιτέρω η ιδιωτικότητα των προσωπικών δεδομένων, όλες οι συναλλαγές με τη Βάση Δεδομένων Πλαισίου παρακολουθούνται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos και συγκεκριμένα δικαιώματα πρόσβασης για ανάγνωση/εγγραφή, από και προς τη βάση δεδομένων πλαισίου, παρέχονται για κάθε εφαρμογή, τύπο και πηγή των Αντικειμένων Πλαισίου.

Επιπρόσθετα, με την περεταίρω ανάπτυξη του Διαχειριστή Πολιτικής Δικαιωμάτων και την ικανότητα να δημιουργήσει ολοένα και πιο πολύπλοκους κανόνες πολιτικής δικαιωμάτων, κάθε κλήση API πλαισίου ενεργοποιεί μια σειρά γεγονότων πολιτικής δικαιωμάτων που σχετίζονται με αυτή, στους παρακάτω άξονες:

- **Μέθοδος του API Πλαισίου:** Ο χρήστης της προσωπικής ζώνης στην οποία πραγματοποιείται η κλήση, έχει δώσει την άδειά του να πραγματοποιούνται κλήσεις για την συγκεκριμένη μέθοδο του API Πλαισίου γενικά, ή για κάποια συγκεκριμένη εφαρμογή;
- **Ανάγνωση/Εγγραφή:** Έχει δοθεί δικαίωμα ανάγνωσης ή εγγραφής δεδομένων στη Βάση Δεδομένων Πλαισίου της συγκεκριμένης προσωπικής ζώνης γενικά, για κάποια συγκεκριμένη εφαρμογή, ή αντικείμενο δεδομένων;
- **Εφαρμογή που πραγματοποιεί την κλήση:** Η εφαρμογή έχει δικαίωμα χρήσης του API Πλαισίου, συγκεκριμένων μεθόδων ατού, ή πρόσβασης σε αντικείμενα δεδομένων;
- **Αντικείμενο Δεδομένων:** Το αντικείμενο δεδομένων (Αντικείμενα Πλαισίου, Κανόνες Πλαισίου, Συνδρομές Πλαισίου κλπ.), το οποίο ζητείται επιτρέπεται να αναγνωστεί ή να εγγραφεί γενικά, ή από κάποια συγκεκριμένη εφαρμογή;

Από τη στιγμή που κάθε Προσωπική Ζώνη έχει ένα και μοναδικό χρήστη και κατά προέκταση, κάθε χρήστης έχει δική του Βάση Δεδομένων Πλαισίου, η οποία ανήκει αποκλειστικά και μόνο σε αυτόν, δεν υπάρχει λόγος κάποιος από τους παραπάνω άξονες να αφορά στον χρήστη. Μία τέτοια πρόσβαση μπορεί να δοθεί μέσω της πολιτικής δικαιωμάτων στο API Πλαισίου, μέσω των μηχανισμών του webinos που σχετίζονται έμμεσα με τον Διαχειριστή Πλαισίου, για την περίπτωση σύνδεσης με Προσωπικές Ζώνες άλλων χρηστών. Οι υπόλοιποι κανόνες που αφορούν σε συγκεκριμένη πρόσβαση προκύπτουν στη συνέχεια με τον ίδιο τρόπο, όπως όταν πρόκειται για εφαρμογή της ίδιας Προσωπικής Ζώνης του χρήστη. Θεωρείται δεδομένο, ξεκινώντας από τον σχεδιασμό, έως και την υλοποίηση και του API Πλαισίου, καθώς και των προσωπικών ζωνών του webinos και του εικονικού PZP, ότι ο κάθε χρήστης αποτελεί ξεχωριστή οντότητα, και ως προς την πλατφόρμα του webinos, αλλά και ως προς τις εφαρμογές που την αξιοποιούν.

Όταν πραγματοποιείται μία κλήση στο API Πλαισίου από μία οντότητα, ο Διαχειριστής Πολιτικής Δικαιωμάτων του webinos ελέγχει τους άξονες αυτούς, χωρίς συμμετοχή της οντότητας που πραγματοποίησε την κλήση. Σε περίπτωση που τα

ανάλογα δικαιώματα δεν έχουν δοθεί, ενημερώνεται ο χρήστης ώστε να λάβει μία απόφαση για την απόδοση του κατάλληλου δικαιώματος μεταξύ των:

α) Επιτρέπεται ή απαγορεύεται στην εφαρμογή να χρησιμοποιεί το API Πλαισίου (μόνο για αυτή τη συνεδρία ή για ένα προκαθορισμένο διάστημα);
β) Επιτρέπεται ή απαγορεύεται στην εφαρμογή (μόνο για αυτή τη συνεδρία ή ένα προκαθορισμένο διάστημα) να:
    1) Εκτελεί ερωτήματα πλαισίου για ένα συγκεκριμένο API του webinos
    2) Ορίζει, εγγράφει και εξάγει δικά της Αντικείμενα Πλαισίου Εφαρμογής
    3) Εκτελεί ερωτήματα πλαισίου σε Αντικείμενα Πλαισίου Εφαρμογής άλλης εφαρμογής
    4) Ορίζει Συνδρομές Πλαισίου
    5) Ορίζει Κανόνες Πλαισίου και Ακροατές Πλαισίου;

Ανάλογα με την επιλογή του προγραμματιστή κάθε εφαρμογής, το αίτημα μπορεί να πραγματοποιηθεί είτε στην πρώτη εκτέλεση της εφαρμογής μετά την εγκατάσταση, είτε όταν η ανάλογη ενέργεια χρειαστεί πρόσβαση στο API πλαισίου.

Ανάκληση ή τροποποίηση των δικαιωμάτων αυτών, καθώς και όλων των δικαιωμάτων που παρέχονται από τον Διαχειριστή Πολιτικής Δικαιωμάτων του webinos παρέχει το πρόγραμμα επεξεργασίας πολιτικής δικαιωμάτων (webinos Policy Editor) που παρέχεται μέσω του Πίνακα Ελέγχου του webinos (webinos Dashboard). Σε αυτό, ο χρήστης έχει τη δυνατότητα να εμφανίσει, να προσθέσει, να διαγράψει και να τροποποιήσει τους κανόνες πολιτικής δικαιωμάτων της προσωπικής του ζώνης. Οι κανόνες πολιτικής δικαιωμάτων του webinos χαρακτηρίζονται από το αν επιτρέπουν ή δεν επιτρέπουν κάποια ενέργεια ή σύνδεση. Με άλλα λόγια, σε περίπτωση που ο χρήστης επιλέξει να μην επιτρέπει σε μία εφαρμογή να εκτελεί κάποια ενέργεια, την επόμενη φορά που η εφαρμογή θα ζητήσει την ίδια ενέργεια, ο χρήστης δεν πρόκειται να ερωτηθεί ξανά.

5.10. API Βάσης Δεδομένων

Η υλοποίηση του μηχανισμού πλαισίου του webinos βασίζεται στη δημιουργία ενός API Βάσης Δεδομένων (DB API) για το webinos, το οποίο επιτρέπει την

αποθήκευση δεδομένων στην κάθε τοπική βάση των PZP για κάθε API του webinos, μεταξύ των οποίων και το API Πλαισίου. Η βάση δεδομένων αυτή έχει υλοποιηθεί σε TingoDB. Για την περίπτωση του API Πλαισίου, το υποσύνολο της βάσης που αφορά σε αυτό, αποτελεί τα δεδομένα των ενδιάμεσων μνημών (buffers), τα οποία αναμένουν την εγγραφή τους στην κεντρική mongoDB βάση δεδομένων στο στιγμιότυπο υπηρεσιών PZH (εικονικό PZP) που αντιστοιχεί στον χρήστη και αυτά που αποτελούν τα τοπικά Αντικείμενα Πλαισίου, Συνδρομές Πλαισίου και Κανόνες Πλαισίου, τα οποία συγχρονίζονται με την mongoDB και τα οποία αποθηκεύονται τοπικά για γρήγορη πρόσβαση και εξοικονόμηση πόρων δικτύου. Η κεντρική mongoDB βάση δεδομένων είναι αυτή που περιέχει το υποσύνολο δεδομένων που αποτελεί τη Βάση Δεδομένων Πλαισίου μίας προσωπικής ζώνης.

Μία εφαρμογή του webinos έχει τη δυνατότητα να κάνει εξερεύνηση των υπηρεσιών του DB API απευθείας ή μέσω των δυνατοτήτων αποθήκευσης που παρέχονται για τα APIs που θα κάνουν χρήση της. Η κάθε συλλογή (collection) μέσα στη βάση δεδομένων αφορά σε διαφορετικό API ή εφαρμογή. Οι συλλογές αυτές δεν εμφανίζονται ως ξεχωριστές υπηρεσίες από προεπιλογή, παρά μόνο αυτές που ανήκουν σε εφαρμογές και σε ειδική περίπτωση σε κάποιο API, εφόσον κριθεί απαραίτητο. Η διαχείριση πραγματοποιείται από ένα στιγμιότυπο διαχειριστή, το οποίο θα διατηρεί τις ρυθμίσεις της βάσης, της αποθήκευσης και των συλλογών, το οποίο θα χρησιμοποιείται από τον τελικό χρήστη μέσω του Πίνακα Ελέγχου του webinos (webinos Dashboard).

6. Μελέτη Περίπτωσης – Επέκταση σεναρίου Cardio Hills

Η εφαρμογή επίδειξης παρακολούθησης και καταγραφής των καρδιακών παλμών με τη χρήση του webinos δείχνει πώς ένας χρήστης μπορεί, εγκαθιστώντας μία εφαρμογή του webinos στο έξυπνο κινητό του τηλέφωνο και φορώντας ένα Bluetooth καταγραφικό των καρδιακών του παλμών, να παρακολουθεί την κατάσταση της υγείας του κατά τη διάρκεια της μέρας (84).

Η εφαρμογή παρακολούθησης και καταγραφής των καρδιακών παλμών αποτελεί ένα απλό παράδειγμα του πώς η τεχνολογία και τα API του webinos επιτρέπουν σε έναν χρήστη να εξάγει δεδομένα από πολλούς, διαφορετικού τύπου, αισθητήρες, όπως αισθητήρες κίνησης, απομακρυσμένα. Σε αυτή την περίπτωση, ο αισθητήρας είναι ένας αισθητήρας καρδιακών παλμών που τοποθετείται στο μπράτσο του χρήστη.

Παράλληλα, η επέκταση της εφαρμογής Cardio Hills με τη χρήση του Διαχειριστή και API Πλαισίου παρέχει δυνατότητες κεντρικής αποθήκευσης, συσσωμάτωσης, συνδυασμού, υπολογισμού, αυτοματοποίησης και διαμοιρασμού των δεδομένων και των δυνατοτήτων της εφαρμογής. Σκοπός της μελέτης περίπτωσης είναι να λειτουργήσει ως επίδειξη του τρόπου με τον οποίο μπορούν να χρησιμοποιηθούν όλες οι δυνατότητες του Διαχειριστή και API Πλαισίου που περιγράφονται σε αυτή τη διατριβή και από την σκοπιά του προγραμματιστή εφαρμογών επίγνωσης πλαισίου, αλλά και από την σκοπιά του τελικού χρήστη των εφαρμογών αυτών.

6.1.Εναλλακτικές λύσεις

Υπάρχουν πολλές εφαρμογές για κινητά τηλέφωνα, οι οποίες εκμεταλλεύονται τα δεδομένα από τέτοιους αισθητήρες, τα συνδυάζουν με δεδομένα που προέρχονται από τον αισθητήρα GPS και αναπαριστούν στην οθόνη τις αθλητικές δραστηριότητες του χρήστη τους. Ένα βασικό πρόβλημα με αυτές τις εφαρμογές είναι ότι τα δεδομένα παραμένουν στο κινητό τηλέφωνο ή αποστέλλονται σε κάποιο απομακρυσμένο διακομιστή. Σε περίπτωση που ο χρήστης ήθελε να μοιραστεί με ασφάλεια τα δεδομένα αυτά με έναν προσωπικό γυμναστή, ή έναν ιατρό, θα έπρεπε, στη μία περίπτωση, να εξάγει χειροκίνητα τα δεδομένα και να τα μεταδώσει σε μία αντίστοιχη συσκευή του γυμναστή ή του ιατρού, όπου ο δεύτερος θα έπρεπε να τα εισάγει στην ίδια εφαρμογή

(11). Στη δεύτερη περίπτωση, όπου τα δεδομένα αποθηκεύονται σε κάποιο απομακρυσμένο διακομιστή, μπορούν να χρησιμοποιηθούν με τρόπο που παραβιάζει την ιδιωτικότητα του χρήστη, ενώ ταυτόχρονα δεν είναι άμεσα διαθέσιμα για εξαγωγή ή αξιοποίηση τους από άλλες εφαρμογές στις οποίες ενδέχεται να επιθυμεί να δώσει πρόσβαση ο χρήστης.

Αν ο προγραμματιστής εφαρμογών επιθυμούσε να απλουστεύσει τη διαδικασία διαμοιρασμού των δεδομένων μιας τέτοιας εφαρμογής με ασφάλεια μεταξύ δύο χρηστών χωρίς τη χρήση απομακρυσμένου διακομιστή και χωρίς να χρησιμοποιήσει το webinos, θα έπρεπε να αναπτύξει ένα κοινό σύστημα αποθήκευσης για τους δύο χρήστες, καθώς και τον τρόπο ασφαλούς επικοινωνίας της εφαρμογής με άλλες συσκευές, χρησιμοποιώντας πρωτόκολλα όπως Bluetooth, NFC, ή TCP/IP. Ακόμα και σε αυτό το σενάριο, άλλες εφαρμογές δεν πρόκειται να έχουν πρόσβαση σε αυτά τα δεδομένα, επειδή η εφαρμογή θα πρέπει να χρησιμοποιεί ένα ασφαλές σύστημα αποθήκευσης, στο οποίο εφαρμογές από άλλους κατασκευαστές δεν θα έχουν πρόσβαση. Εναλλακτικά, η εφαρμογή θα πρέπει να παρέχει κάποιο ασφαλές API για την πρόσβαση στα δεδομένα, όμως, έτσι, η πρόσβαση σε αυτά θα περιορίζεται από το γεγονός ότι η εφαρμογή που τα δημιούργησε θα πρέπει είτε να εκτελείται, είτε να παρέχεται ως υπηρεσία στο κάθε σύστημα. Ταυτόχρονα, ένα τέτοιο API θα πρέπει να παρέχει δικό του ορισμό και περιγραφή των διεπαφών, της δομής των δεδομένων και των μεθόδων που παρέχει.

Συνολικά, η ανάπτυξη αυτών των δυνατοτήτων από τον προγραμματιστή κάθε εφαρμογής προσθέτει απαιτήσεις πόρων οι οποίες στην περίπτωση μικρών εφαρμογών όπως το Cardio Hills υπερβαίνουν κατά πολύ τους πόρους που είναι απαραίτητοι για την ανάπτυξη των βασικών λειτουργιών τους. Παρόλα αυτά, οι δυνατότητες αυτές μπορούν να δώσουν μεγάλη αξία σε εφαρμογές που στη βάση τους δεν χρειάζεται να είναι ιδιαίτερα περίπλοκες, ενώ ταυτόχρονα να διασφαλίζουν την ασφάλεια, την ιδιωτικότητα και την ανοιχτή επεκτασιμότητα τους.

6.2.Cardio Hills με Επίγνωση Πλαισίου

Η εφαρμογή Cardio Hills του webinos, χρησιμοποιεί όλες τις δυνατότητες του Διαχειριστή Πλαισίου του webinos, ήτοι την αναχαίτιση δεδομένων από τα APIs, τα

Αντικείμενα Πλαισίου Εφαρμογών, τις Συνδρομές Πλαισίου, τους κανόνες Πλαισίου και τα Ερωτήματα Πλαισίου του webinos.

Συνδρομή Πλαισίου

Το Cardio Hills όταν εκτελείται, ορίζει μία συνδρομή πλαισίου για την τακτική λήψη δεδομένων GPS από το API Γεωτοποθεσίας (Geolocation API) του webinos, μέσω της Υπηρεσίας Φόντου του Διαχειριστή Πλαισίου. Η συνδρομή αυτή πραγματοποιεί κλήσεις προς το webinos, οι οποίες μεταφράζονται ως κλήσεις στον αισθητήρα GPS της συσκευής. Ανάλογα με τις ρυθμίσεις που έχουν οριστεί, οι κλήσεις αυτές μπορούν να πραγματοποιούνται σε οποιοδήποτε διάστημα μεγαλύτερο του ενός δευτερολέπτου, που αποτελεί τον ελάχιστο χρόνο που είναι απαραίτητος για να ανανεωθούν τα δεδομένα GPS.

Παράλληλα, ορίζεται μία συνδρομή πλαισίου για το API Γενικού Αισθητήρα. Ο αισθητήρας σε αυτή την περίπτωση είναι το Bluetooth καταγραφικό καρδιακών παλμών. Ο ρυθμός με τον οποίο συλλέγονται τα δεδομένα ορίζεται και σε αυτή την περίπτωση από τον προγραμματιστή.

Εναλλακτικά, μπορεί να οριστεί, αυτή τη φορά μέσω του API Γεωτοποθεσίας και του API Γενικού Αισθητήρα, απευθείας ακροατές γεγονότων, οι οποίοι επιστρέφουν τις τιμές της τοποθεσίας αυστηρά κάθε δευτερόλεπτο, και του γενικού αισθητήρα όποτε είναι διαθέσιμες.

Αναχαίτιση δεδομένων από τα APIs

Οι απαντήσεις στις κλήσεις προς το API Γεωτοποθεσίας ή το API Γενικού Αισθητήρα δεν είναι απαραίτητο να λαμβάνονται απευθείας από την εφαρμογή, καθώς με την χρήση του μηχανισμού αναχαίτησης των RPC κλήσεων του webinos, τα δεδομένα γεωτοποθεσίας και του γενικού αισθητήρα γίνονται άμεσα διαθέσιμα μέσω του API Πλαισίου.

Αντικείμενα Πλαισίου Εφαρμογών

Η εφαρμογή Cardio Hills, σε περίπτωση που οι κλήσεις γίνονται απευθείας προς το API Γενικού αισθητήρα, ορίζει ένα σύνολο από Αντικείμενα Πλαισίου Εφαρμογών (π.χ. MyPulses), τα οποία περιέχουν δεδομένα από τον εξωτερικό αισθητήρα, μαζί με μία χρονοσφραγίδα. Τα Αντικείμενα Πλαισίου που προκύπτουν

από τον μετασχηματισμό που πραγματοποιείται μέσω των Λεξιλογίων API και Εφαρμογών, δηλαδή η τοποθεσία για το API Γεωτοποθεσίας και τα δεδομένα από το καταγραφικό παλμών αντίστοιχα, συγχρονίζονται με τη Βάση Δεδομένων Πλαισίου στο στιγμιότυπο υπηρεσιών PZH (εικονικό PZP). Στη συνέχεια, τα δεδομένα αυτά είναι διαθέσιμα για όποια εφαρμογή έχει δυνατότητες webinos, έχει τα κατάλληλα δικαιώματα και ανήκει σε PZP στην ίδια προσωπική ζώνη, στην αφαιρετική, ανεξαρτήτου πλατφόρμας μορφή των Αντικειμένων Πλαισίου.

Ερωτήματα Πλαισίου

Σε περίπτωση που ένας προγραμματιστής επιθυμεί να αναπτύξει μία εφαρμογή, σε οποιαδήποτε πλατφόρμα, η οποία θα παρουσιάζει ένα χάρτη με τη διαδρομή που έτρεξε ένας αθλητής, μαζί με τα δεδομένα των καρδιακών του παλμών, το μόνο που χρειάζεται είναι μία κλήση στο API Πλαισίου με τα Ερωτήματα Πλαισίου που θα επιστρέψουν τα αντικείμενα με τις γεωγραφικά στίγματα και τους καρδιακούς παλμούς. Τα ερωτήματα αυτά θα έχουν ορίσματα τα αναγνωριστικά χαρακτηριστικά των Αντικειμένων Πλαισίου που αναζητούνται (όνομα αντικειμένου, URI του API από το οποίο προέρχονται, URI εφαρμογής που τα δημιούργησε κλπ.), την συσκευή από την οποία προέρχονται τα δεδομένα και η οποία ήταν συνδεδεμένη με τον αισθητήρα καρδιακών παλμών, και τις χρονοσφραγίδες αρχής και τέλους (χρησιμοποιώντας τις συνθήκες *le* και *ge*) που ο αθλητής έκανε την προπόνησή του. Ύστερα, το μόνο που χρειάζεται είναι ο προγραμματιστής να εμφανίσει τα δεδομένα αυτά στον χάρτη.

Κανόνες Πλαισίου

Χρησιμοποιώντας το API Ειδοποιήσεων (Notification API) του webinos, το API Πλαισίου επιτρέπει στον προγραμματιστή να ορίσει έναν Κανόνα Πλαισίου, ο οποίος καταλήγει να εμφανίζει στον χρήστη μία ειδοποίηση του συστήματος (ανάλογα με την πλατφόρμα στην οποία εκτελείται), η οποία έχει νόημα ως προς το πλαίσιο του χρήστη. Για παράδειγμα, το πρόγραμμα εκγύμνασης που ακολουθεί ο χρήστης ενδέχεται να έχει έναν θερμιδικό στόχο. Χρησιμοποιώντας Αντικείμενα Πλαισίου που αφορούν στον χρήστη, όπως η ηλικία, το φύλο, το ύψος και το βάρος, και άλλα που αφορούν στη δραστηριότητα, όπως ο χρόνος έναρξης και λήξης αυτής, τη γωοτοποθεσία, συμπεριλαμβανομένων των συντεταγμένων, τις διαφοροποιήσεις του υψομέτρου και τις αντίστοιχες χρονοσφραγίδες, τους σφυγμούς και τις χρονοσφραγίδες στις οποίες καταγράφηκαν, καθώς και τις καιρικές συνθήκες, όπως θερμοκρασία και

επίπεδα υγρασίας, ο προγραμματιστής μπορεί ορίσει ένα Αντικείμενο Πλαισίου Εφαρμογής που αφορά σε ένα πρόγραμμα προπόνησης που ακολουθήθηκε (π.χ. MyTrainingSession). Το Αντικείμενο Πλαισίου αυτό, μπορεί να παρέχει έναν υπολογισμό πραγματικού χρόνου των θερμίδων που καταναλώθηκαν. Ο Κανόνας Πλαισίου που παρακολουθεί το αντικείμενο αυτό, μπορεί να ενεργοποιηθεί όταν ο θερμιδικός στόχος έχει επιτευχθεί και να παρέχει μια ειδοποίηση συστήματος, συνοδευόμενη με ήχο και δόνηση στο χρήστη. Πατώντας πάνω στην ειδοποίηση αυτή, μπορεί να εκκινήσει η εφαρμογή Cardio Hills χρησιμοποιώντας το API Εκκίνησης Εφαρμογών του webinos (AppLauncher API) και αυτομάτως να εμφανίσει τα αποτελέσματα της προπόνησης στη οθόνη, επειδή το API Εκκίνησης Εφαρμογών μπορεί να χρησιμοποιήσει το αντικείμενο πλαισίου MyTrainingSession ως όρισμα για την εκκίνηση της εφαρμογής.

Διαμοιρασμός δεδομένων πλαισίου εντός της Προσωπικής Ζώνης

Τα Αντικείμενα Πλαισίου Εφαρμογών μπορούν να διαμοιραστούν μεταξύ διαφορετικών εφαρμογών και συσκευών στην ίδια προσωπική ζώνη, εφόσον ο χρήστης το επιτρέψει μέσω του Διαχειριστή Πολιτικής Δικαιωμάτων. Αυτό σημαίνει ότι, δοθέντων των κατάλληλων δικαιωμάτων, ξεχωριστές εφαρμογές μπορούν να έχουν πρόσβαση στα ίδια δεδομένα από τους αισθητήρες για να τα συλλέξουν και για να τα αναλύσουν όταν ο αθλητής επιστρέψει στο σπίτι του. Δεν υπάρχει κάποιος περιορισμός ως προς τον αριθμό των εφαρμογών που μπορούν να έχουν πρόσβαση. Παρόλα αυτά, οι εφαρμογές αυτές δεν έχουν δικαίωμα να τροποποιήσουν ή να διαγράψουν εγγραφές που έχουν δημιουργηθεί μέσω της αναχαίτισης των κλήσεων προς τα API ή από άλλες εφαρμογές.

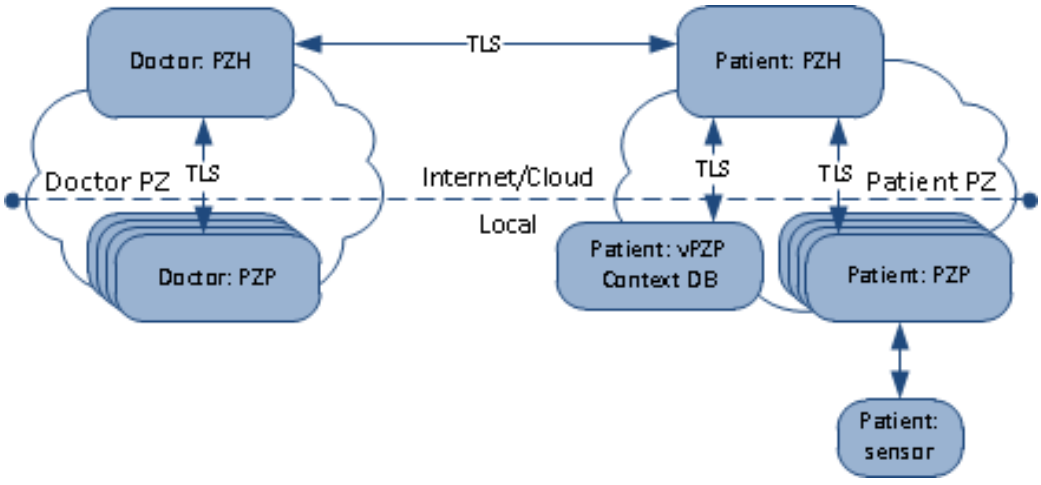
Σύνδεση με διαφορετικές Προσωπικές Ζώνες

Η εφαρμογή Cardio Hills επεκτείνει τη συνδεσιμότητα εφαρμογών και πέρα από αυτή που παρέχεται εντός μίας Προσωπικής Ζώνης. Χρησιμοποιώντας τις δυνατότητες επικοινωνίας μεταξύ διαφορετικών κόμβων προσωπικής ζώνης (PZH-PZH) και τις δυνατότητες λειτουργίας εφαρμογών του webinos μεταξύ διαφορετικών προσωπικών ζωνών, ο χρήστης έχει τη δυνατότητα να μοιραστεί τα Αντικείμενα Πλαισίου που έχουν δημιουργηθεί μέσω της εφαρμογής με όποια άτομα επιθυμεί, δίνοντας πρόσβαση στα δεδομένα αυτά σε εφαρμογές που ανήκουν σε άλλους χρήστες. Το webinos δίνει στον τελικό χρήστη τη δυνατότητα να ελέγχει πλήρως το προσωπικό

προφίλ εμπιστευτικότητάς του και να επιτρέπει μόνο στα άτομα που ο ίδιος επιθυμεί να έχουν πρόσβαση σε αυτό το μέρος των δεδομένων πλαισίου του για συγκεκριμένα χρονικά διαστήματα. Έτσι, ένας ιατρός, ο οποίος έχει μία συσκευή με webinos, η οποία αναλύει δεδομένα από βιοαισθητήρες, μπορεί να συλλέγει επιπλέον δεδομένα από πολλούς αισθητήρες εκτός από τον αισθητήρα παλμών, όπως πίεσης, κορεσμού οξυγόνου κ.α., άλλα φορετά συστήματα υγείας, καθώς και δεδομένα από το API Γεωτοποθεσίας, και να εμπλουτίσει, έτσι, το προφίλ του ασθενούς, με τα προϋπάρχοντα δεδομένα από την προσωπική ζώνη του ασθενούς.

Ανοιχτότητα

Τα δεδομένα που αποθηκεύονται στη Βάση Δεδομένων Πλαισίου και παρέχονται μέσω του API Πλαισίου στο νέφος της προσωπικής ζώνης είναι απόλυτα ανοιχτά και διαθέσιμα στον χρήστη. Αυτό έχει επιτρέψει στο webinos να σχηματίσει ένα καινοτόμο παράδειγμα διαμοιρασμού γνώσης μεταξύ διαφορετικών εφαρμογών, ακόμα και διαφορετικών κατασκευαστών εφαρμογών, ελαχιστοποιώντας την περιπλοκότητα της διαλειτουργικότητας μεταξύ εφαρμογών μέσω του webinos. Ταυτόχρονα, συγκεντρώνοντας τη συλλογή των Αντικειμένων Πλαισίου σε κάθε προσωπική ζώνη, τα δεδομένα από τα API και τις εφαρμογές μπορούν να αποθηκευτούν, χωρίς να δεσμεύονται από κάποια συγκεκριμένη εφαρμογή η συσκευή.



Εικόνα 40: Επικοινωνία Προσωπικής Ζώνης ασθενούς με Προσωπική Ζώνη ιατρού στο Cardio Hills

Τότε, και πάντα με την προϋπόθεση ότι ο χρήστης παρέχει τη συγκατάθεσή του, αυτά τα δεδομένα μπορούν να ερωτηθούν από οποιαδήποτε εφαρμογή μπορεί να τα χρησιμοποιήσει, από οποιαδήποτε συσκευή στη προσωπική ζώνη του χρήστη, ή την προσωπική χρήστη οποιουδήποτε άλλου χρήστη, εφόσον του έχει δοθεί η ρητή άδεια

από τον ιδιοκτήτη της. Από τη στιγμή που αυτή η λειτουργία είναι ενσωματωμένη στο webinos, όλα αυτά μπορούν να επιτευχθούν από τον προγραμματιστή, με πολύ λίγες και διαισθητικά απλές κλήσεις στο API Πλαισίου, το API Μηνυμάτων μεταξύ εφαρμογών (App2App Messaging API) και το API Διαχείρισης Γεγονότων (Event Handling API), όπως εμφανίζεται στην Εικόνα 40.

Ιδιωτικότητα και Ασφάλεια

Από τη στιγμή που εφαρμογή Cardio Hills χρησιμοποιεί τον Διαχειριστή και το API Πλαισίου, τα δεδομένα που δημιουργεί αποθηκεύονται στη Βάση Δεδομένων Πλαισίου, η οποία βρίσκεται στο Εικονικό Πληρεξούσιο Προσωπικής Ζώνης. Αυτό εξασφαλίζει το ότι τα δεδομένα που δημιουργούνται μέσω του API και του Διαχειριστή Πλαισίου βρίσκονται σε σύστημα στο οποίο ο τελικός χρήστης έχει την αποκλειστική πρόσβαση ο ίδιος. Μέσω της σύνθετης πολιτικής ασφάλειας που μπορεί να παρέχει το webinos και εφαρμόζει ο Διαχειριστής Πλαισίου, διασφαλίζεται ότι αν ο τελικός χρήστης επιλέξει να παρέχει πρόσβαση στη Βάση Δεδομένων Πλαισίου σε κάποια εφαρμογή, την παρέχει επιλεκτικά μόνο για τα δεδομένα και τις λειτουργίες για τις οποίες επιλέγει. Με αυτό τον τρόπο, αν μία εφαρμογή όπως το Cardio Hills, εκτός από τα δεδομένα από το καταγραφικό παλμών και το GPS αιτηθεί πρόσβαση σε δεδομένα, όπως το ιστορικό πλοήγησης ή την τηλεόραση, χωρίς αυτό να δικαιολογείται με κάποιον τρόπο από τις λειτουργίες της, ο χρήστης μπορεί την απορρίψει προσωρινά ή μόνιμα.

7. Συμπεράσματα

Σε αυτή τη διδακτορική διατριβή επιχειρήθηκε να δοθεί μία μεθοδολογικά πλήρης, υλοποιήσιμη και πρακτική λύση σε μία σειρά ζητημάτων που καταλήγουν να περιορίζουν τα συστήματα επίγνωσης πλαισίου, τα οποία ενώ σχεδιάζονται για να είναι πανταχού παρόντα, στην πραγματικότητα είναι παρόντα σε πολύ συγκεκριμένες συνθήκες εγκατάστασης και χρήσης. Ένα τέτοιο σύστημα δε θα πρέπει να περιορίζεται ούτε από την πλατφόρμα ή τη φυσική συσκευή στην οποία εκτελείται, ούτε από την εφαρμογή και τον κατασκευαστή της που το εκμεταλλεύεται, ούτε, παράλληλα, να στερεί από τον χρήστη την κυριότητα των δεδομένων του δικού του πλαισίου. Ταυτόχρονα θα πρέπει να μην παρέχεται με μία απαγορευτική περιπλοκότητα στους προγραμματιστές και τις μικρομεσαίες εταιρίες πληροφορικής, ώστε να μπορούν εισέλθουν στον χώρο των εφαρμογών επίγνωσης πλαισίου και να αρχίσουν εύκολα να αναπτύσσουν εφαρμογές με διευρυμένες δυνατότητες.

Στην πορεία προς την ανάπτυξη μίας ολοκληρωμένης λύσης, λήφθηκαν υπόψη η διαθέσιμη βιβλιογραφία στο θέμα του διάχυτου υπολογισμού, καθώς και οι διάφορες προκλήσεις που αντιμετώπισε η ομάδα υλοποίησης του webinos, η οποία παρείχε και την πλατφόρμα πάνω στην οποία αναπτύχθηκε το API και ο Διαχειριστής Πλαισίου που περιγράφηκε. Η σημαντικότερη πρόκληση ήταν η ίδια η καινοτόμος φύση του webinos, ενώ οι δυνατότητες χρήσης υπαρχόντων τεχνολογιών περιορίζονταν από τα χαρακτηριστικά των συσκευών και των λειτουργικών συστημάτων με τα οποία το webinos μπορούσε ή του επιτρεπόταν να είναι συμβατό. Περιορισμοί είχαν προκύψει και από τον συνολικό στόχο του webinos να χρησιμοποιεί το ελάχιστο των υπολογιστικών πόρων, την έλλειψη συμβατότητας πολλών υπαρχόντων προγραμματιστικών λύσεων με το οικοσύστημα του webinos, καθώς και την αστάθεια, εξαιτίας της πιλοτικής φύσης της πλατφόρμας. Παράλληλα, πολλές από τις λειτουργίες του Διαχειριστή και του API Πλαισίου, οι οποίες χρειαζόταν να αναπτυχθούν με βάση άλλες προδιαγεγραμμένες λειτουργίες και API του webinos, εξαιτίας της παράλληλης φύσης της ανάπτυξής πολλών από αυτές, έπρεπε να αναμείνουν την ωρίμανση της πλατφόρμας ώστε να δύνανται να υλοποιηθούν (72).

Παρά τις παραπάνω δυσκολίες, συνολικά επετεύχθη η ανάπτυξη ενός πλήρους μεθοδολογικού μοντέλου, το οποίο επιτυγχάνει την προσέγγιση των ζητημάτων που

ετέθησαν εξ αρχής. Αναλυτικότερα, με τον Διαχειριστή και το API Πλαισίου του webinos:

1. Τα δεδομένα επίγνωσης πλαισίου είναι προσβάσιμα από κάθε έξυπνη συσκευή, ανεξάρτητα από το λειτουργικό σύστημα που αυτή εκτελεί, ενώ ταυτόχρονα, τα δεδομένα ανήκουν εξολοκλήρου στον χρήστη. Τα δεδομένα πλαισίου δεν είναι αποθηκευμένα μόνο σε μία συσκευή, ούτε βρίσκονται σε κάποιο απομακρυσμένο διακομιστή που δεν ανήκει στον χρήστη. Αντιθέτως, βρίσκονται είτε σε διακομιστή που ανήκει στον χρήστη, είτε σε διακομιστή στον οποίο ο χρήστης έχει αποκλειστική πρόσβαση. Με αυτόν τον τρόπο, τα δεδομένα πλαισίου του χρήστη, τα οποία μπορούν να είναι από σχετικά αδιάφορα έως αρκούντως ευαίσθητα, ώστε να αποτρέπουν τον ιδιοκτήτη τους να τα μοιραστεί με κάποια πλατφόρμα ή εφαρμογή, αποθηκεύονται σε χώρο τέτοιο, ώστε να διασφαλίζεται ότι η πρόσβαση και η επεξεργασία αυτών από εφαρμογές, περιορίζεται στις ίδιες τις εφαρμογές και στις συσκευές που ανήκουν σε αυτόν.
2. Οι εφαρμογές με δυνατότητες webinos μοιράζονται ένα κοινό οικοσύστημα, αποθηκεύουν σε κοινό χώρο και μοιράζονται τα δεδομένα πλαισίου που δημιουργούν με όλες τις υπόλοιπες εφαρμογές του webinos, στις οποίες ο χρήστης παρέχει επιλεκτικά πρόσβαση. Σε δεδομένα πλαισίου, τα οποία ενδέχεται να είναι κοινά μεταξύ περισσοτέρων από μία εφαρμογές, αποφεύγεται ο κίνδυνος εμφάνισης διπλοτύπων, καθώς ενοποιείται ο τρόπος απόκτησής τους μέσω του μηχανισμού αναχαίτισης μηνυμάτων από τα API του webinos. Έτσι, για παράδειγμα, τα δεδομένα από τον αισθητήρα GPS μίας κινητής συσκευής αποθηκεύονται με έναν τρόπο, ο οποίος είναι κοινός για όλες τις εφαρμογές που πρόκειται να τα χρησιμοποιήσουν. Ταυτόχρονα, εξαιτίας της κεντρικοποιημένης αποθήκευσης των δεδομένων πλαισίου και της ύπαρξης ενός μηχανισμού για την πρόσβαση σε αυτά από τις εφαρμογές, επιτυγχάνεται σημαντική οικονομία πόρων των συστημάτων στα οποία εκτελούνται.
3. Από τη στιγμή που το webinos αποτελεί μία πλατφόρμα μόνιμου χαρακτήρα για κάθε χρήστη, η προσθήκη μίας καινούριας συσκευής ή η απεγκατάσταση μίας εφαρμογής ή υπηρεσίας που δημιουργεί δεδομένα πλαισίου δεν μειώνει τα δεδομένα που είναι προσβάσιμα για τις εφαρμογές webinos που ανήκουν στην προσωπική ζώνη του χρήστη, καθώς αποθηκεύονται σε Βάση Δεδομένων που είναι κοινή για όλες τις εφαρμογές του webinos και ο χρόνος ζωής τους είναι

ανεξάρτητος από το χρόνο ζωής των εφαρμογών ή των συσκευών που τα δημιούργησαν.

4. Η συλλογή δεδομένων επίγνωσης πλαισίου από ανομοιογενή συστήματα χρησιμοποιεί έναν κοινό μηχανισμό, με αποτέλεσμα να μην απαιτείται από κάθε προγραμματιστή εφαρμογών να αναπτύξει διαφορετικούς μηχανισμούς για κάθε λειτουργικό σύστημα και πλατφόρμα από την οποία επιθυμεί να συλλέξει δεδομένα. Με αυτό τον τρόπο είναι πιο εύκολη η είσοδος ενός προγραμματιστή ή μίας μικρομεσαίας εταιρίας πληροφορικής στον χώρο των εφαρμογών επίγνωσης πλαισίου.
5. Η πλατφόρμα του webinos παρέχει υπηρεσίες επίγνωση πλαισίου, οι οποίες δεν αποτελούν εμπορικές υπηρεσίες, δεν έχουν περιορισμό ως προς τα οικοσυστήματα που μπορούν να τις εκμεταλλευτούν και δεν περιορίζονται σε φορητές συσκευές. Τα δεδομένα που δημιουργεί μία εφαρμογή είναι δυνητικά προσβάσιμα και από όποια άλλη συσκευή ή εφαρμογή επιθυμεί ο χρήστης, δίνοντας έτσι κίνητρο για την ανάπτυξη συνεργασιών στα επίπεδα δεδομένων και ανάπτυξης λογισμικού.

Συνολικά, σε αυτή τη διδακτορική διατριβή αιτιολογήθηκε και περιεγράφηκε ο Διαχειριστής και το API Πλαισίου στο webinos, ένα καινοτόμο περιβάλλον διάχυτου υπολογισμού που παρέχεται για χρήση σε διασυνδεδεμένες εφαρμογές ανεξάρτητες πλατφόρμας, ενώ ταυτόχρονα εισήχθη στη βιβλιογραφία που αφορά στην επίγνωση πλαισίου για πληροφοριακά συστήματα, ένας καινούριος όρος, αυτός του Αντικειμένου Πλαισίου (Context Object). Οι κύριοι στόχοι που επιτεύχθηκαν με αυτή την προσέγγιση είναι μία καινοτόμα και διαισθητική διεπαφή, με την οποία ένας προγραμματιστής εφαρμογών με δυνατότητες webinos, μπορεί εύκολα να έχει πρόσβαση σε πληροφορίες πλαισίου που είναι σχετικές με την εφαρμογή που αναπτύσσει, ελαχιστοποιώντας την προσπάθεια για την απόκτηση και το φιλτράρισμα δεδομένων πλαισίου, εντός του webinos, ενός υψηλά καινοτομικού και ανεξαρτήτου πλατφόρμας προσωπικών εφαρμογών και αποθήκευσης οικοσυστήματος συννέφου.

8. Μελλοντική Εργασία

Οι μελλοντικές κατευθύνσεις της έρευνας που πραγματοποιήθηκε στο πλαίσιο αυτής της διδακτορικής διατριβής μπορούν να χωριστούν σε δύο κατηγορίες, τις βελτιώσεις και τις επεκτάσεις. Τα ζητήματα βελτιώσεων αφορούν στην αναβάθμιση της ποιότητας των υπηρεσιών και των δεδομένων που παρέχονται από το Διαχειριστή και το API Πλαισίου του webinos, ενώ οι επεκτάσεις αφορούν σε νέες λειτουργίες, οι οποίες μπορούν να χτίσουν σε κοινές πρακτικές στη βιβλιογραφία, ώστε να μπορούν οι εφαρμογές να εφαρμόσουν καινοτόμους αλγορίθμους επίγνωσης πλαισίου, βασισμένους στο κατά πολύ διευρυμένης ποικιλίας σύνολο δεδομένων πλαισίου που συλλέγει και επεξεργάζεται το webinos σε σχέση με άλλες προτεινόμενες μεθοδολογίες.

Στα ζητήματα των βελτιώσεων, είναι χρήσιμη η ανάπτυξη μίας έξυπνης αυτοματοποιημένης πολιτικής συλλογής σκουπιδιών στη Βάση Δεδομένων Πλαισίου. Εξαιτίας του πολύ μεγάλου όγκου δεδομένων που ενδέχεται να συλλέγει το webinos, ειδικά όταν υπολογίσουμε ένα σενάριο χρήσης όπου πολλαπλές συσκευές είναι συνδεδεμένες στην ίδια προσωπική ζώνη με πολλαπλές εφαρμογές για διάστημα ετών, είναι σημαντικό να αναπτυχθεί μία μεθοδολογία συμπύκνωσης των δεδομένων πλαισίου σε δύο άξονες.

Ο πρώτος άξονας αφορά στην συμπύκνωση δεδομένων σε σχεδόν πραγματικό χρόνο σε σχέση με την συλλογή τους. Επί του παρόντος, για παράδειγμα, από το API Τηλεόρασης, καταγράφονται όλα τα γεγονότα αλλαγής καναλιού όταν αυτά συμβαίνουν. Αυτό σημαίνει ότι όταν ο χρήστης κινείται προοδευτικά στη λίστα καναλιών (zapping) για να καταλήξει και να παραμείνει σε κάποιο κανάλι, κάθε αλλαγή θα καταγράφεται. Η πληροφορία, όμως, αυτή έχει περιορισμένη χρησιμότητα. Μία λύση για αυτή την περίπτωση θα ήταν η δημιουργία εξαγόμενων Αντικειμένων Πλαισίου, όπου θα καταγράφονται μόνο τα κανάλια στα οποία ο χρήστης πέρασε σημαντικό χρόνο, ακόμα κι αν ενδιάμεσα πέρναγε κι από άλλα, ενώ τα δεδομένα των ίδιων των γεγονότων αλλαγής καναλιού θα σβήνονται μετά από κάποιες ώρες ή μέρες.

Αντίστοιχα, για την περίπτωση των δεδομένων γεωτοποθεσίας, ενώ καταγράφονται όλα τα στίγματα ανά κάποιο προκαθορισμένο διάστημα, ο ρυθμός δειγματοληψίας μπορεί να μεταβάλλεται ανάλογα με τη μεταβλητότητα των

μετρήσεων. Για παράδειγμα, αν δεν υπάρξει κάποια σημαντική αλλαγή στο στίγμα, σημαίνει ότι ο χρήστης βρίσκεται σταθερά σε κάποιο σημείο, άρα δεν υπάρχει λόγος να προστίθενται νέες καταχωρήσεις. Αν, πάλι, κινείται, είναι πιο χρήσιμο να διατηρούνται περισσότερα στίγματα, ώστε να καταγραφεί η διαδρομή. Στη συνέχεια, από τα στίγματα αυτά μπορούν να προκύψουν αυτομάτως εξαγόμενα Αντικείμενα Πλαισίου, διαφορετικών τύπων, για παράδειγμα, παραμονής σε κάποια τοποθεσία, με το χρονικό διάστημα αυτής, και μετακίνησης από τοποθεσία σε τοποθεσία, με διανυσματικά δεδομένα προκαθορισμένης ακρίβειας. Όπως και στην περίπτωση των καναλιών της τηλεόρασης, τα δεδομένα των μοναδικών στιγμάτων θα διαγράφονται μετά από κάποιες ώρες ή μέρες.

Παρόμοιες ενέργειες μπορούν να πραγματοποιηθούν για τις περισσότερες μεθόδους των API του webinos, ενώ, παράλληλα, μπορεί να διερευνηθεί η περίπτωση, ακόμα και τα εξαγόμενα Αντικείμενα Πλαισίου να επιδέχονται εκ νέου συμπύκνωση, δημιουργώντας έτσι μία σχέση μεταξύ του χρόνου που έχει παρέλθει από την απόκτηση μίας πληροφορίας, της ιστορικής χρησιμότητάς της, και του βαθμού ασάφειας που μπορεί να επιδεχτεί.

Στα ζητήματα των επεκτάσεων, θα είναι χρήσιμη μία διερεύνηση της αξιολόγησης της χρησιμότητας/συνάφειας των Αντικειμένων Πλαισίου. Η χρησιμότητα/συνάφεια των Αντικειμένων Πλαισίου θα πρέπει να ορίζεται , κατ' αρχάς, αναγνωρίζοντας, κατ' αρχάς, την ταυτότητα της ενεργής συσκευής, δηλαδή της συσκευής που χρησιμοποιείται εκείνη τη στιγμή από τον χρήστη, για παράδειγμα η κλήση στο API Γεωτοποθεσίας που χρησιμοποιεί, ή τελευταία χρησιμοποίησε ο χρήστης, το tablet όταν χρησιμοποιείται, έναντι του κινητού τηλεφώνου που βρίσκεται στην τσέπη, η συσκευή που βρίσκεται σε κίνηση. Έπειτα, πρέπει να αναγνωρίζεται η προτιμώμενη συσκευή για μία συγκεκριμένη ενέργεια σε ένα συγκεκριμένο πλαίσιο, όπως η τηλεόραση στο σαλόνι, αντί αυτής στην κρεβατοκάμαρα, οι ειδοποιήσεις στην τηλεόραση όταν χρησιμοποιείται, έναντι του υπολογιστή που είναι ανοιχτός, ή το σύστημα πλοήγησης του αυτοκινήτου, αντί αυτού του κινητού τηλεφώνου όταν ο χρήστης οδηγεί. Η χρησιμότητα/συνάφεια κάθε συσκευής ή κάθε Αντικειμένου Πλαισίου θα πρέπει να αξιολογείται από τον ίδιο τον Διαχειριστή Πλαισίου, προτού επιστραφούν δεδομένα ή πυροδοτηθεί ένα γεγονός σε μία συσκευή. Σε ένα περιβάλλον στο οποίο ανήκουν πολλαπλές συσκευές που ανήκουν στον ίδιο χρήστη, και οι οποίες

μπορεί να είναι ταυτόχρονα ανοιχτές, είναι σημαντικό να μπορεί να αποσαφηνίζεται ποιο είναι το πλαίσιο της κάθε συσκευής και ποιο είναι το πλαίσιο του χρήστη.

9. Δημοσιεύσεις

1. Ntanos C, Botsikas C, Rovis G, Kakavas P, Askounis D. *A context awareness framework for cross-platform distributed applications*. Journal of Systems and Software 2014; 88:138–46.

2. Vergori P, Ntanos C, Gavelli M, Askounis D. *The webinos Architecture: A Developer's Point of View*. In: Uhler D, Mehta K, Wong JL, editors. Mobile Computing, Applications, and Services: Fourth International Conference, Mobicase 2012, Seattle, Wa, USA, October 2012. Revised Selected Papers: Springer-Verlag New York Inc; 2013. p. 391–9.

3. Su T, Lyle J, Atzeni A, Faily S, Virji H, Ntanos C et al. *Continuous Integration for Web-Based Software Infrastructures: Lessons Learned on the webinos Project*. In: Hardware and Software: Verification and Testing 2013. p. 145–50.

4. Charalabidis Y, Lampathaki F, Sarantis D, Sourouni A, Mouzakitis S, Gionis G et al. *The Greek Electronic Government Interoperability Framework: Standards and Infrastructures for One-Stop Service Provision*. In: 2008 12th Panhellenic Conference on Informatics. p. 66–70.

5. Sourouni A, Kourlimpinis G, Tsavdaris H, Askounis D, Ntanos C. *Electronic monitoring of advertising firms shareholder structure; an initiative of secretariat general of communication - secretariat general of information*. Proceedings of the European Conference on e-Government, ECEG 2008.

6. Sourouni A, Tsavdaris H, Kourlimpinis G, Ntanos C, Askounis D. *Government process reengineering in practice; A case study of G2B transaction's interoperability achievement*. Proceedings of the European Conference on e-Government, ECEG 2009.

7. Kourlimpinis G., Sourouni A.-M., Tsavdaris H., Ntanos C., Askounis D. *Reengineering of Press Distribution Monitoring in Greece: an Initiative of Greek Secretariat General of Communication and Information*. eGOVINTEROP 2007 Conference. 10th-11th of October 2007, Paris, France.

10. Βιβλιογραφία

1. Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P. Towards a Better Understanding of Context and Context-Awareness; 1707:304–7.

2. Ailisto H, Alahuhta P, Haataja V, Kyllönen V, Lindholm M, Electonics VT. Structuring context aware applications: Five-layer model and example case. In: Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing; 2002. p. 1–5 .

3. Anagnostopoulos C, Hadjiefthymiades S. On the application of epidemical spreading in collaborative context-aware computing. SIGMOBILE Mob. Comput. Commun. Rev. 2009; 12(4):43.

4. Anagnostopoulos CB, Tsounis A, Hadjiefthymiades S. Context Awareness in Mobile Computing Environments. Wireless Pers Commun 2007; 42(3):445–64.

5. ANIND K. DEY. Context-Aware Computing: The CyberDesk Project. In: National Conference on Artificial Intelligence; 1998 .

6. Atzeni A, Cameroni C, Faily S, Lyle J, Flechais I. Here's Johnny: A Methodology for Developing Attacker Personas. p. 722–7 .

7. Bakken DE. MIDDLEWARE.

8. Baldauf M, Dustdar S, Rosenberg F. A survey on context-aware systems. IJAHUC 2007; 2(4):263.

9. Bardram JE. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications: A Service Infrastructure and Programming Framework for Context-Aware Applications; 3468:98–115.

10. Bolchini C, Curino CA, Quintarelli E, Schreiber FA, Tanca L. A data-oriented survey of context models. SIGMOD Rec. 2007; 36(4):19.

11. Botsikas A. Heart Rate Monitor – Sharing data & services with your friends; 2012. Available from: URL:http://webinos.org/2012/08/17/heart-rate-monitor-sharing-data-services-with-your-friends/.

12. Brézillon P, Borges M, Pino J, Pomerol J. Context-awareness in group work: Three case studies. In: Proceedings International Conference on Decision Support Systems DSS2004; 2004. p. 115–24 .

13. Brown PJ. Triggering information by context. Personal Technologies 1998; 2(1):18–27.

14. Capra L, Emmerich W, Mascolo C. CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IIEEE Trans. Software Eng. 2003; 29(10):929–44.

15. Capra L, Quercia D. Middleware for social computing: a roadmap. J Internet Serv Appl; 2012. (vol 3).

16. Chen G, Kotz D. A Survey of Context-Aware Mobile Computing Research.

17. Chen G, Kotz D. Solar: An Open Platform for Context-Aware Mobile Applications. In: Pervasive computing: First international conference, Pervasive 2002, Zurich, Switzerland, August 26-28, 2002 : proceedings. Berlin, New York: Springer; 2002. p. 41–7 .

18. CHEN H, FININ TI, JOSHI A. An ontology for context-aware pervasive computing environments. Knowl. Eng. Rev. 1999; 18(3):197–207.

19. CHEN H, Tolia S, Sayers C, Finin T, JOSHI A. Creating Context-Aware Software Agents. p. 186–97 .

20. Common Attack Pattern Enumeration and Classification (CAPEC) [cited 2014 Jan 31]. Available from: URL:http://capec.mitre.org/.

21. Cooper A, Martin A. Towards a secure, tamper-proof grid platform. p. 8 pp-380 .

22. Costa PD, Pires LF, van Sinderen MJ, Filho JGP. Towards a service platform for mobile context-aware applications. In: First International Workshop on Ubiquitous Computing, IWUC 2004. Portugal: INSTICC Press; 2004. p. 48–62 Available from: URL:http://doc.utwente.nl/63553/.

23. Dey A, Abowd G, Salber D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Comp. Interaction 2001; 16(2):97–166.

24. Dey AK. Understanding and Using Context. Personal and Ubiquitous Computing 2001; 5(1):4–7.

25. Digest of Papers. First International Symposium on Wearable Computers. In: Wearable Computers, 1997. Digest of Papers., First International Symposium on; 1997. p. iii- .

26. Dionysis Athanasopoulos, Apostolos V. Zarras, Valerie Issarny, Evaggelia Pitoura, Panos Vassiliadis. CoWSAMI: Interface-aware context gathering in ambient intelligence environments. Pervasive and Mobile Computing 2008; 4(3):360–89. Available from: URL:http://www.sciencedirect.com/science/article/pii/S1574119207000740.

27. Fuhrhop C, Lyle J, Faily S. The Webinos Project. In: Proceedings of the 21st International Conference Companion on World Wide Web. New York, NY, USA: ACM; 2012. p. 259–62 (WWW '12 Companion). Available from: URL:http://doi.acm.org/10.1145/2187980.2188024.

28. Gilbert S, Lynch N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. SIGACT News 2002; 33(2):51–9. Available from: URL:http://doi.acm.org/10.1145/564585.564601.

29. Gionis G, Desruelle H, Blomme D, Lyle J, Faily S, Bassbouss L. "Do we know each other or is it just our Devices?": A Federated Context Model for Describing Social Activity Across Devices. In: W3C/PrimeLife Federated Social Web Europe Conference 2011; 2011 Available from: URL:http://d-cent.org/fsw2011/wp-content/uploads/fsw2011-A-Federated-Context-Model-for-Describing-Social-Activity-Across-Devices.pdf.

30. Glassey R, Stevenson G, Richmond M, NIXON P, Terzis S, Wang F et al. Towards a middleware for generalised context management. In: First International Workshop on Middleware for Pervasive and Ad Hoc Computing, Middleware 2003; 2003 .

31. Grolinger K, Higashino WA, Tiwari A, Am Capretz M. Data management in cloud environments: NoSQL and NewSQL data stores. J Cloud Comput Adv Syst Appl 2013; 2(1):22.

32. Gu T, Pung HK, Zhang DQ. A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 2005; 28(1):1–18.

33. Henricksen K, Indulska J, Rakotonirainy A. Modeling Context Information in Pervasive Computing Systems; 2414:167–80.

34. Henricksen K, Robinson R. A survey of middleware for sensor networks: state-of-the-art and future directions. In: Proceedings of the international workshop on Middleware for sensor networks; 2006. p. 60–5 .

35. Hofer T, Schwinger W, Pichler M, Leonhartsberger G, Altmann J, Retschitzegger W. Context-awareness on mobile devices - the hydrogen approach:10 pp.

36. Hong J, Landay J. An Infrastructure Approach to Context-Aware Computing. Human-Comp. Interaction 2001; 16(2):287–303.

37. Hong JI. The Context Fabric: An Infrastructure for Context-aware Computing. In: CHI '02 Extended Abstracts on Human Factors in Computing Systems. New York, NY, USA: ACM; 2002. p. 554–5 (CHI EA '02). Available from: URL:http://doi.acm.org/10.1145/506443.506478.

38. Indulska J, Sutton P. Location Management in Pervasive Systems. In: Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003 - Volume 21. Darlinghurst, Australia, Australia: Australian Computer Society, Inc; 2003. p. 143–51 (ACSW Frontiers '03). Available from: URL:http://dl.acm.org/citation.cfm?id=827987.828003.

39. Judd G, Steenkiste P. Providing contextual information to pervasive computing applications. p. 133–42 .

40. Kim H, Cho Y, Oh S. CAMUS: A middleware supporting context-aware services for network-based robots. In: Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on; 2005. p. 237–42 .

41. Kofod-Petersen A, Mikalsen M. Context: Representation and Reasoning: Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle on Applying Context-Management 2005.

42. Korpipaa P, Mantyjarvi J, Kela J, Keranen H, Malm E. Managing context information in mobile devices: A new software framework simplifies the development of context-aware mobile applications by managing raw context information gained

from multiple sources and enabling higher-level context abstractions. IEEE Pervasive Comput. 2003; 2(3):42–51.

43. Krakowiak S. What is Middleware?

44. Lassila O, Swick RR. Resource Description Framework (RDF) Model and Syntax Specification: World Wide Web Consortium W3C; 1999.

45. Li J, Meng F, Wen X, Si Y, Wang Q. Research and Design of Network Information Query System Based on MIDP. In: Hu W, editor. Advances in Electric and Electronics: Springer Berlin Heidelberg; 2012. p. 797–804 (Lecture Notes in Electrical Engineering). Available from: URL:http://dx.doi.org/10.1007/978-3-642-28744-2_103.

46. Michahelles F, Samulowitz M. Smart CAPs for Smart Its - Context Detection for Mobile Users. Personal and Ubiquitous Computing 2002; 6(4):269–75.

47. MongoDB Inc. The MongoDB 2.4 Manual; 2014 [cited 2014 Jan 23]. Available from: URL:http://docs.mongodb.org/manual/.

48. Mowbray TJ, Zahavi R. The essential CORBA: Systems integration using distributed objects. New York: Wiley; 1995.

49. Ngo HQ, Shehzad A, Liaquat S, Riaz M, Lee S. Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. In: Yang LT, editor. Embedded and ubiquitous computing: Proceedings. Berlin, Heidelberg, New York: Springer; 2004. p. 672–81 .

50. NORBERG PA, HORNE DR, HORNE DA. The Privacy Paradox: Personal Information Disclosure Intentions versus Behaviors. Journal of Consumer Affairs 2007; 41(1):100–26.

51. Ntanos C, Botsikas C, Rovis G, Kakavas P, Askounis D. A context awareness framework for cross-platform distributed applications. Journal of Systems and Software 2014; 88(0):138–46. Available from: URL:http://www.sciencedirect.com/science/article/pii/S0164121213002562.

52. Open Web Application Project (OWASP) [cited 2014 Jan 31]. Available from: URL:https://www.owasp.org/index.php/Main_Page.

53. Orfali R, Harkey D, Edwards J. The essential client-server survival guide. Nueva York: Van Nostrand Reinhold; 1994.

54. Park J, Moon M, Hwang S, Yeom K. CASS: A Context-Aware Simulation System for Smart Home. p. 461–7 .

55. Pascoe J. Adding generic contextual capabilities to wearable computers:92–9.

56. Pascoe J, Ryan N, Morse D. Issues in Developing Context-Aware Computing; 1707:208–21.

57. Petrelli D, Not E, Zancanaro M, Strapparava C, Stock O. Modelling and Adapting to Context. Personal and Ubiquitous Computing 2001; 5(1):20–4.

58. Pressman RS. Software engineering: A practitioner's approach. 7th ed. New York: McGraw-Hill Higher Education; 2010.

59. Proceedings International Conference on Decision Support Systems DSS2004; 2004.

60. Pruitt J, Adlin T. The persona lifecycle: Keeping people in mind throughout product design. Amsterdam, Boston: Elsevier; Morgan Kaufmann Publishers, an imprint of Elsevier; 2006. (The Morgan Kaufmann series in interactive technologies).

61. Pulido JRG, Ruiz MAG, Herrera R, Cabello E, Legrand S, Elliman D. Ontology languages for the semantic web: A never completely updated review. Knowledge-Based Systems 2006; 19(7):489–97. Available from: URL:http://www.sciencedirect.com/science/article/pii/S0950705106000736.

62. Ranganathan A, Al-Muhtadi J, Campbell R. Reasoning about uncertain contexts in pervasive computing environments. Znanstvena revija. Družboslovje in filozofija = Social sciences and philosophy 2004; 3(2):62–70.

63. Roman M, Hess C, Cerqueira R, Ranganathan A, Campbell R, Nahrstedt K. A middleware infrastructure for active spaces. IEEE Pervasive Comput. 2002; 1(4):74–83.

64. Samulowitz M, Michahelles F, Linnhoff-Popien C. Adaptive Interaction for Enabling Pervasive Services. In: Proceedings of the 2Nd ACM International Workshop on Data Engineering for Wireless and Mobile Access. New York, NY,

USA: ACM; 2001. p. 20–6 (MobiDe ’01). Available from: URL:http://doi.acm.org/10.1145/376868.376886.

65. Schilit B, Adams N, Want R. Context-Aware Computing Applications:85–90.

66. Schilit B, Theimer M. Disseminating active map information to mobile hosts. IEEE Network 1994 [cited 2013 Mar 1]; 8(5):22–32.

67. Schmidt A, Beigl M, Gellersen H. There is more to context than location. Computers & Graphics 1999; 23(6):893–901.

68. Schmidt A, van Laerhoven K. How to build smart appliances? IEEE Pers. Commun. 2001; 8(4):66–71.

69. Sindre G, Opdahl AL. Eliciting security requirements with misuse cases. Requirements Eng 2005; 10(1):34–44.

70. Sommerville I. Software Engineering (4th Ed.): Instructor’s Guide. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc; 1993.

71. Sousa JP, Garlan D. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. In: Proceedings of the IFIP 17th World Computer Congress-TC2 Stream/3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance; 2002. p. 29–43 .

72. Su T, Lyle J, Atzeni A, Faily S, Virji H, Ntanos C et al. Continuous Integration for Web-Based Software Infrastructures: Lessons Learned on the webinos Project. In: Hardware and Software: Verification and Testing. p. 145–50 .

73. Tilkov S, Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Comput. 2010; 14(6):80–3.

74. Truong H, Dustdar S. A survey on context-aware web service systems. International Journal of Web Information Systems 2009; 5(1):5–31.

75. Vergori P, Ntanos C, Gavelli M, Askounis D. The webinos Architecture: A Developer’s Point of View. In: Uhler D, Mehta K, Wong JL, editors. Mobile Computing, Applications, and Services: Fourth International Conference, Mobicase 2012, Seattle, Wa, USA, October 2012. Revised Selected Papers: Springer-Verlag New York Inc; 2013. p. 391–9 .

76. Want R, Hopper A, Falcão V, Gibbons J. The active badge location system. ACM Trans. Inf. Syst. 1992; 10(1):91–102.

77. webinos Project. webinos Factsheet [cited 2014 Jun 1]. Available from: URL:http://webinos.org/files/2011/06/webinos-factsheet.pdf.

78. webinos Project. WP 3.1 deliverable - Overall architecture [cited 2014 Jan 31]. Available from: URL:http://dev.webinos.org/deliverables/wp3/d31.html.

79. webinos Project. WP 3.5 deliverable - Security framework [cited 2014 Jan 31]. Available from: URL:http://dev.webinos.org/deliverables/wp3/d35.html.

80. William Noah Schilit. A System Architecture for Context-Aware Mobile Computing: Columbia University; 1995.

81. Winograd T. Architectures for Context. Human-Comp. Interaction 2001; 16(2):401–19.

82. Xiaohang Wang, Jin Song Dong, ChungYau Chin, Hettiarachchi S, Daqing Zhang. Semantic Space: An Infrastructure for Smart Spaces. IEEE Pervasive Comput. 2004; 03(03):32–9.

83. Yamabe T, Takagi A, Nakajima T. Citron: A Context Information Acquisition Framework for Personal Devices. p. 489–95 .

84. Ziran Sun. Webinos Demo Series #4: Cardio Hills – Remote Sensors; 2012. Available from: URL:http://webinos.org/2012/02/25/webinos-demo-series-4-cardio-hills-remote-sensors/.

85. Κοντογιώργης Α. ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ ΕΝΗΜΕΡΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ [ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ].

ΠΑΡΑΡΤΗΜΑ

11. Παράρτημα 1: Web IDL προδιαγραφή API Πλαισίου

webinos Context API

- ## Summary of Methods

Interface	Method
Context	void executeQuery(ContextQuery query, ContextQuerySuccessCB successCallback, ContextQueryErrorCB? errorCallback) void schemaExists(DOMString uri, boolean uriIsApplication, DOMString contextObjectName, SchemaExistsCB callback, ErrorCB? errorCallback)
ContextObject	
ContextObjectField	
ContextQuery	
QueryFilter	
QueryModifier	
ApplicationContextManager	void registerSchema(ApplicationURI applicationUri, DOMString contextObjectName, DOMString version, ContextObjectFieldArray fields, SuccessCB? successCallback, ErrorCB? errorCallback) void addData(ApplicationURI applicationUri, DOMString contextObjectName, ContextObjectValueArray values, SuccessCB successCallback, ErrorCB? errorCallback)

Interface	Method
ContextRuleManager	void search(ContextRuleURI contextRuleUri, ContextRuleSearchCB successCallback, ErrorCB? errorCallback) void save(ContextRule rule, ContextRuleSaveCB? saveCallback, ErrorCB? errorCallback) void remove(ContextRuleURI contextRuleUri, SuccessCB? successCallback, ErrorCB? errorCallback) void renew(ContextRuleURI contextRuleUri, SuccessCB? successCallback, ErrorCB? errorCallback) void addListener(ContextRuleURI contextRuleURI, DOMString listenerName, ContextRuleEventHandler callback, SuccessCB? successCallback, ErrorCB? errorCallback) void removeListener(ContextRuleURI contextRuleURI, DOMString listenerName, SuccessCB? successCallback, ErrorCB? errorCallback)
ContextRule	
ContextRuleEvent	
ScheduleManager	void search(DOMString? apiUri, boolean onlyOwnSchedules, ScheduleSearchCB callback, ErrorCB? errorCallback) void save(Schedule schedule, ScheduleSaveCB? saveCallback, ErrorCB? errorCallback) void remove(Schedule schedule, SuccessCB? successCallback, ErrorCB? errorCallback) void renew(Schedule schedule, SuccessCB? successCallback, ErrorCB? errorCallback)
Schedule	

- # Introduction

The Context API defines the high-level interfaces required to obtain access to a user's context data. User must enable Context Manager on his device in order to keep track of his context objects. The API supports two basic ways of accessing context data:

1. executing a query against the context data storage and retrieving context data through the query results. 2. subscribing to receive real time context data updates as soon as a context related event happens (via the ContextRuleManager).

The API also offers the ability to schedule API calls in order to update the context data store even if the application is not running.

- # Interfaces and Dictionaries
- ## Interface Context

This interface defines context properties. It is a context specific extension to the interface Service in the ServiceDiscovery module. This is the main interface for querying data stored in the context database.

WebIDL

```
interface Context : Service {
        readonly attribute ApplicationContextManager app;

        readonly attribute ContextRuleManager rules;

        readonly attribute ScheduleManager schedule;

        void executeQuery(ContextQuery query, ContextQuerySuccessCB
successCallback, optional ContextQueryErrorCB? errorCallback);

        void schemaExists(DOMString uri, boolean uriIsApplication,
DOMString contextObjectName, SchemaExistsCB callback, optional
ErrorCB? errorCallback);
};
```

Attributes

readonly ApplicationContextManager app

The entry point for the Application ContextObject management.

This attribute is readonly.

readonly ContextRuleManager rules

The entry point for the ContextRule management.

This attribute is readonly.

readonly ScheduleManager schedule

The entry point for the Scheduled API calls management.

This attribute is readonly.

Methods

executeQuery

Performs a ContextQuery against the context database.

Signature

```
void executeQuery(ContextQuery query, ContextQuerySuccessCB
successCallback, optional ContextQueryErrorCB? errorCallback);
```

When this method is invoked, it executes the provided query against the context storage. The context storage is a collection of context objects, each one with specific attributes, which hold context data that have been acquired over time, by identifying a number of context related events. The Query parameter that this method uses specifies what context data (i.e. from which context objects) should be retrieved.

Mediation by policy and security: this method, as it provides application with data (context) about the user, is expected to have privacy considerations. Therefore the system is able to ignore the request of an app to receive context

data if the User Privacy Policy dictates so (i.e. the user has not authorised the application to access the context data it asked for in the Query parameters).

Parameters

- **query**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextQuery
  - **Description:** The ContextQuery to execute against the Context Storage.
- **successCallback**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextQuerySuccessCB
  - **Description:** Function to be invoked if the asynchronous query operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ContextQueryErrorCB
  - **Description:** Function to be invoked if the asynchronous query operation results in errors.

`schemaExists`

Check if the Context schema is defined.

Signature

```
void schemaExists(DOMString uri, boolean uriIsApplication,
DOMString contextObjectName, SchemaExistsCB callback, optional
ErrorCB? errorCallback);
```

Parameters

- **uri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The unique URI of an Application or API depending on the uriIsApplication value.
- **uriIsApplication**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** boolean
  - **Description:** The URI provided is for an Application ContextObject if true. Otherwise it refers to an API ContextObject.

- **contextObjectName**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The alias of the Application ContextObject that will be checked for existence.
- **callback**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** SchemaExistsCB
  - **Description:** The callback function that passes the result of the search.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous query operation results in errors.

Interface ContextObject

A ContextObject is a unit for carrying data that uniquely defines a piece of contextual information.

WebIDL

```
interface ContextObject{
        readonly attribute DOMString name;
        readonly attribute DOMString? apiUri;
        readonly attribute ApplicationURI? applicationUri;
        readonly attribute DOMString deviceId;
        readonly attribute DOMString sessionId;
        readonly attribute DOMTimeStamp timestamp;
        readonly attribute DOMString? version;
        readonly attribute ContextObjectValueArray values;
};
```

Attributes

readonly DOMString name

The alias of the ContextObject. This is a short description of the contained information. For webinos APIs an indicative list is the following:

- MyContacts
- MyPositions
- MyTVChannels
- MyClimates

Each name aggregates the same data from various APIs and applications. An example of this is the MyPositions which is a ContextObjectName that aggregates the location data from both the Geolocation API and the Vehicle API.

This attribute is readonly.

readonly DOMString? apiUri

If this ContextObject was extracted from a webinos API, this is the webinos API URI, otherwise (in the case of application ContextObject) this is null.

This attribute is readonly.

readonly ApplicationURI? applicationUri

If this ContextObject was saved from an application, this is the unique application URI. Otherwise, this is null.

This attribute is readonly.

readonly DOMString deviceId

The device id that initiated the request for this ContextObject

This attribute is readonly.

readonly DOMString sessionId

The session id from which this ContextObject was extracted.

This attribute is readonly.

readonly DOMTimeStamp timestamp

The timestamp when the specific ContextObject was saved.

This attribute is readonly.

readonly DOMString? version

The version of the Application ContextObject schema. This can be null in case there is no version info available.

This attribute is readonly.

readonly ContextObjectValueArray values

The contextual data.

This attribute is readonly.

- ## Interface ContextObjectField

ContextObjectField definition. This is the definition of a field in the ContextObject

WebIDL

```
interface ContextObjectField{
        attribute DOMString name;
        attribute DOMString type;
};
```

Attributes

DOMString name

The name of the property

DOMString type

A description of the data type of the value

The developer will be able to cast the value based on this attribute. Example of possible values are:

- double: The value can contain decimals separated by a dot
- short, long, number: The value is an integer with the specified precision
- DOMString: The value is a DOMString
- DOMTimeStamp: The value is a DOMTimeStamp
- boolean: The value is boolean (true, false)
- object: The value is a JSON object
- array: The value is an array

- ## Interface ContextQuery

The query structure of the Context Manager.

WebIDL

```
interface ContextQuery{
        attribute QueryFilter filter;
        attribute Object? criteria;
        attribute Object? projection;
        attribute QueryModifier? modifier;
};
```

The querying language that is used by the Context Queries is based on mongoDB. You can find a useful comparison to the SQL syntax.

Attributes

`QueryFilter` filter

Specifies selection criteria for ContextObject's properties (except values).

Object? criteria

Specifies the selection criteria for the ContextObject values using mongoDB's query operators. To return the entire ContextObjectArray, omit this attribute. Refer to mongoDB's query tutorial for more info.

Object? projection

Specifies the ContextObject fields to return using mongoDB's projection operators. To return all fields in the matching ContextObject, omit this attribute. Refer to mongoDB's projection tutorial for more info.

`QueryModifier`? modifier

Specifies the modifications (sort, limit, skip) to apply to the results. This attribute can be omitted.

- ## Interface QueryFilter

A filter clause to limit the result set of the query based on the ContextObject parameters.

WebIDL

```
interface QueryFilter{
        attribute DOMString name;
        attribute DOMString? apiUri;
        attribute ApplicationURI? applicationUri;
        attribute (DOMString or object)? deviceId;
        attribute (DOMString or object)? sessionId;
        attribute (DOMString or object)? timestamp;
        attribute (DOMString or object)? version;
};
```

Always provide an apiUri OR an applicationUri along with the name of the ContextObject. This object can use mongoDB's query operators to describe filters, except for the name, apiUri and applicationUri which must be provided in plane <field>: <value> format.

Attributes

DOMString name

The name of the ContextObject to be queried.

DOMString? apiUri

The API URI for which the ContextObject name to be queried. This argument can be omitted only when searching for Application ContextObject, providing the applicationUri.

ApplicationURI? applicationUri

The application URI for which the ContextObject name to be queried. This argument can be omitted only when searching for API ContextObject, providing the apiUri.

union? deviceId

The device id by which results should be filtered.

union? sessionId

The session id by which results should be filtered.

union? timestamp

The timestamp by which results should be filtered.

union? version

The ContextObject schema version by which results should be filtered.

- ## Interface QueryModifier

The modifiers (sort, limit, skip) to apply to the result set of ContextObject.

WebIDL

```
interface QueryModifier{
        attribute integer? limit;
        attribute integer? skip;
```

```
        attribute object? sort;
};
```

Attributes

integer? limit

The maximum number of ContextObjects for the result ContextObjectArray. This attribute can be omitted.

integer? skip

The number of ContextObjects to skip for the result ContextObjectArray. This attribute can be omitted.

object? sort

The sort order for the result ContextObjectArray. The object is expected in mongoDB's sort format. This attribute can be omitted.

- ## Interface ApplicationContextManager

This is the entry point for the Application ContextObject management.

WebIDL

```
interface ApplicationContextManager{
        void registerSchema(ApplicationURI applicationUri, DOMString
contextObjectName, DOMString version, ContextObjectFieldArray fields,
optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);

        void addData(ApplicationURI applicationUri, DOMString
contextObjectName, ContextObjectValueArray values, optional SuccessCB
successCallback, optional ErrorCB? errorCallback);
};
```

Application developers may register their own ContextObject data by defining a schema. The application may save this kind of ContextObject which will be shared among applications and devices for later processing. The end user will be able to give access to these data based on policy manager rules. The main concept is that although the application is able to store data, the data are not owned by the application rather the user. So the user can share this kind of data either with another user or another application.

Methods

registerSchema

Register the schema of an Application ContextObject that will be available for the application to store its contextual information.

Signature

```
void registerSchema(ApplicationURI applicationUri, DOMString
contextObjectName, DOMString version, ContextObjectFieldArray
fields, optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);
```

This function updates or creates the schema of the Application ContextObject. If the schema is already defined, then it overrides it with the given one. Note that the current implementation doesn't do any version checking.

Parameters

- **applicationUri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ApplicationURI
  - **Description:** The unique identifier of the Application. This will not be required when this information will be available through the Session.
- **contextObjectName**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The alias of the ContextObject, in order to be able to retrieve it and insert it later on.
- **version**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The version of the Application ContextObject schema. It is advised to use the Semantic Versioning (http://semver.org/) schema (e.g. 0.5.1) although currently this is not taken into account.
- **fields**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextObjectFieldArray
  - **Description:** The array of ContextObjectField that describes the data that will be stored and retrieved.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous registration operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous registration operation results in errors.

`addData`

Insert an Application ContextObject in the Context database.

Signature

```
void addData(ApplicationURI applicationUri, DOMString
contextObjectName, ContextObjectValueArray values, optional
SuccessCB successCallback, optional ErrorCB? errorCallback);
```

An error may occur if the data that are being inserted does not match the register Application ContextObject schema.

Parameters

- **applicationUri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ApplicationURI
  - **Description:** The unique identifier of the Application. This will not be required when this information will be available through the Session.
- **contextObjectName**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The alias of the ContextObject, in order to be able to retrieve it later on.
- **values**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextObjectValueArray
  - **Description:** The array of ContextObjectValue that contains the data that will be stored.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: No
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous insert operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous insert operation results in errors.

- ## Interface ContextRuleManager

This is the entry point for the ContextRule management.

WebIDL

```
interface ContextRuleManager{
        void search(ContextRuleURI contextRuleUri,
ContextRuleSearchCB successCallback, optional ErrorCB?
errorCallback);

        void save(ContextRule rule, optional ContextRuleSaveCB?
saveCallback, optional ErrorCB? errorCallback);

        void remove(ContextRuleURI contextRuleUri, optional
SuccessCB? successCallback, optional ErrorCB? errorCallback);

        void renew(ContextRuleURI contextRuleUri, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);


        void addListener(ContextRuleURI contextRuleURI, DOMString
listenerName, ContextRuleEventHandler callback, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);

        void removeListener(ContextRuleURI contextRuleURI, DOMString
listenerName, optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);
};
```

Application developers may create new ContextRule in order to trigger events based on those rules. If a ContextRule is not used for a period more than a month, then it is automatically deleted. In order to renew the ContextRule, the application must call the renew() function of the ContextRule

Methods

search

Retrieve the ContextRule with the specific ContextRuleURI

Signature

```
void search(ContextRuleURI contextRuleUri, ContextRuleSearchCB
successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **contextRuleUri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleURI
  - **Description:** The unique identifier of the ContextRule.
- **successCallback**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleSearchCB
  - **Description:** Function to parse the results of the search operation. This will have either one result or null.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes

- **Type:** ErrorCB
- **Description:** Function to be invoked if the asynchronous search operation results in errors.

save

Insert or update a ContextRule.

Signature

```
void save(ContextRule rule, optional ContextRuleSaveCB?
saveCallback, optional ErrorCB? errorCallback);
```

Parameters

- **rule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRule
  - **Description:** The new ContextRule to insert. If the ContextRule already exists and the owner application is the same as the caller, the ContextRule gets updated. Otherwise an error occurs.
- **saveCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ContextRuleSaveCB
  - **Description:** Function to be invoked if the asynchronous insert operation completes successfully. The ContextRule that was saved is returned with possible modifications.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous insert operation results in errors.

remove

Remove a ContextRule.

Signature

```
void remove(ContextRuleURI contextRuleUri, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **contextRuleUri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleURI

- **Description:** The unique identifier of the ContextRule to delete. The ContextRule must exists and the owner application must be the same as the caller, otherwise an error occurs.

- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous delete operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous delete operation results in errors.

renew

Extends the validUntil of a ContextRule for a month.

Signature

```
void renew(ContextRuleURI contextRuleUri, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **contextRuleUri**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleURI
  - **Description:** The unique identifier of the ContextRule to renew.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous renew operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous renew operation results in errors.

addListener

Register a listener for a specific ContextRule.

Signature

```
void addListener(ContextRuleURI contextRuleURI, DOMString
listenerName, ContextRuleEventHandler callback, optional
SuccessCB? successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **contextRuleURI**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleURI
  - **Description:** The URI or the ContextRule to monitor.
- **listenerName**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The name of the listener to add. This name is used to identify the callback so that the application can remove or override the listener.
- **callback**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleEventHandler
  - **Description:** The callback function to process the incoming data when the ContextRule has been triggered.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous registration operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous registration operation results in errors.

`removeListener`

Unregister a listener for a specific ContextRule.

Signature

```
void removeListener(ContextRuleURI contextRuleURI, DOMString
listenerName, optional SuccessCB? successCallback, optional
ErrorCB? errorCallback);
```

Parameters

- **contextRuleURI**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleURI
  - **Description:** The URI or the ContextRule to monitor.
- **listenerName**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMString
  - **Description:** The listener name to remove.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous unregistration operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous unregistration operation results in errors.

- ## Interface ContextRule

This is the definition of a ContextRule object.

WebIDL

```
interface ContextRule{
        attribute ContextRuleURI contextRuleUri;

        attribute ApplicationURI applicationUri;

        attribute DOMString description;

        attribute ContextQuery ruleQuery;

        attribute integer interval;

        attribute DOMTimeStamp validUntil;
};
```

Attributes

ContextRuleURI contextRuleUri

The URI of the ContextRule that was triggered.

ApplicationURI applicationUri

The application URI that is the owner of this ContextRule.

DOMString description

A description of the ContextRule.

ContextQuery ruleQuery

The query that when data are retrieved, the ContextRule is being triggered.

integer interval

The interval in seconds between each ContextRule evaluation. This interval might be overridden by the Context API to avoid performance issues if the provided value is too low. ContextRuleSaveCB should be used, when saving, to determine the overridden value.

DOMTimeStamp validUntil

The timestamp until when this ContextRule is available. After that it will be removed. This is automatically set for new ContextRule to a month after the creation date.

- ## Interface ContextRuleEvent

The event that is fired when a ContextRule has been triggered.

WebIDL

```
interface ContextRuleEvent {
        readonly attribute ContextRuleURI contextRuleUri;
        readonly attribute ContextObjectArray? result;
};
```

The application may register to listen ContextRuleEvent via the ContextRuleManager.addListener() function.

Attributes

readonly ContextRuleURI contextRuleUri

The URI of the ContextRule that was triggered.

This attribute is readonly.

readonly ContextObjectArray? result

The possible resulting array of ContextObject that may occur from the ContextRule.

This attribute is readonly.

- ## Interface ScheduleManager

This is the entry point for the Scheduled API calls management.

WebIDL

```
interface ScheduleManager{
        void search(DOMString? apiUri, boolean onlyOwnSchedules,
ScheduleSearchCB callback, optional ErrorCB? errorCallback);
```

```
        void save(Schedule schedule, optional ScheduleSaveCB?
saveCallback, optional ErrorCB? errorCallback);

        void remove(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);

        void renew(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
};
```

Application developers may request a scheduled API call on Interval in order to force update of the context database. Scheduled API calls may not return fresh information if the implementation of an API imposes its own rate limits.

Methods

`search`

Search for a Schedule

Signature

```
void search(DOMString? apiUri, boolean onlyOwnSchedules,
ScheduleSearchCB callback, optional ErrorCB? errorCallback);
```

Parameters

- **apiUri**
  - **Optional:** No.
  - **Nullable**: Yes
  - **Type:** DOMString
  - **Description:** The API URI to search for. This will be ignored if null is passed, but then, onlyOwnSchedules must be set to True.
- **onlyOwnSchedules**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** boolean
  - **Description:** If true is passed, Search will return only the Schedules that are owned by this application.
- **callback**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ScheduleSearchCB
  - **Description:** Function to parse the results of the search operation.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous search operation results in errors.

`save`

Insert or update a Schedule.

Signature

```
void save(Schedule schedule, optional ScheduleSaveCB?
saveCallback, optional ErrorCB? errorCallback);
```

Parameters

- **schedule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** Schedule
  - **Description:** The new Schedule to insert. If the Schedule already exists and the owner application is the same as the caller, the Schedule gets updated. Otherwise an error occurs.
- **saveCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ScheduleSaveCB
  - **Description:** Function to be invoked if the asynchronous insert operation completes successfully. The Schedule that was saved is returned with possible modifications.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous insert operation results in errors.

`remove`

Remove a Schedule.

Signature

```
void remove(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **schedule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** Schedule
  - **Description:** The Schedule to delete. The Schedule must exist and the owner application must be the same as the caller, otherwise an error occurs.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB

- **Description:** Function to be invoked if the asynchronous delete operation completes successfully.

- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous delete operation results in errors.

renew

Extends the validUntil of a Schedule for a month.

Signature

```
void renew(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
```

Parameters

- **schedule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** Schedule
  - **Description:** The Schedule to renew.
- **successCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** SuccessCB
  - **Description:** Function to be invoked if the asynchronous renew operation completes successfully.
- **errorCallback**
  - **Optional:** Yes.
  - **Nullable**: Yes
  - **Type:** ErrorCB
  - **Description:** Function to be invoked if the asynchronous renew operation results in errors.

- ## Interface Schedule

This is the definition of a Schedule object

WebIDL

```
interface Schedule{

        attribute ApplicationURI applicationUri;

        attribute DOMString apiUri;

        attribute DOMString method;

        attribute object[] params;
```

```
        attribute DOMString deviceId;

        attribute integer interval;

        attribute DOMTimeStamp validUntil;
};
```

Attributes

ApplicationURI applicationUri

The application URI that is the owner of this Schedule.

DOMString apiUri

The webinos API URI to call

DOMString method

The webinos API method to call. The developer will be allowed to add only a specific list of methods and will not be allowed to add methods like the addEventListener.

object[] params

The parameters to pass to the API

DOMString deviceId

The device id where the API call will occur.

integer interval

The interval in seconds between each API call. This interval might be overridden by the Context API to avoid performance issues if the provided value is too low. ScheduleSaveCB should be used, when saving, to determine the overridden value.

DOMTimeStamp validUntil

The timestamp until when this Schedule is valid. After that it will be removed. This is automatically set for new Schedule to a month after the creation date.

- # Type Definitions
- ## ContextObjectFieldArray

Array of ContextObjectField.

WebIDL

```
typedef ContextObjectField[] ContextObjectFieldArray;
```

- ## ContextObjectValueArray

Array of objects. This array will contain one or more objects, that consist of field:value elements.

WebIDL

```
typedef Object[] ContextObjectValueArray;
```

- ## ContextObjectArray

Array of ContextObject.

WebIDL

```
typedef ContextObject[] ContextObjectArray;
```

- ## ContextRuleURI

The unique URI that represents a ContextRule eg. "http://epu.ntua.gr/context/rules/homeLocation".

WebIDL

```
typedef DOMString ContextRuleURI;
```

- ## ApplicationURI

The unique URI that represents an Application eg. "http://epu.ntua.gr/contextDemoApp".

WebIDL

```
typedef DOMString ApplicationURI;
```

- # Callbacks

- **SuccessCB**

Generic callback function when a function completed successfully.

WebIDL

```
callback SuccessCB = void ();
```

Signature

```
void SuccessCB();
```

- **ErrorCB**

Generic callback function when a function fails.

WebIDL

```
callback ErrorCB = void (DOMError error);
```

Signature

```
void ErrorCB(DOMError error);
```

Parameters

- **error**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMError
  - **Description:** A DOMError that describes the error occurred.

- **SchemaExistsCB**

Callback function when the Context calls the schemaExists() function.

WebIDL

```
callback SchemaExistsCB = void (boolean found, DOMString? version);
```

Signature

```
void SchemaExistsCB(boolean found, DOMString? version);
```

Parameters

- **found**

- **Optional:** No.
- **Nullable**: No
- **Type:** boolean
- **Description:** A boolean that determines if the schema is defined.

- **version**
  - **Optional:** No.
  - **Nullable**: Yes
  - **Type:** DOMString
  - **Description:** A DOMString with the current registered version of the Application or API ContextObject. If the version info is unavailable, this is null .

- **`ContextQueryErrorCB`**

Callback function when an error occurs while ContextQuery is being executed.

WebIDL

```
callback ContextQueryErrorCB = void (DOMError error);
```

Signature

```
void ContextQueryErrorCB(DOMError error);
```

Parameters

- **error**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** DOMError
  - **Description:** DOMError object detailing what went wrong; e.g. SecurityError if a security error originating from Policy Manager, NotFoundError if the requested Context Object is not found in Context DB, SyntaxError if the query was not well formed and can not be parsed, InsuficientData if aggregated Context Object can not be created due to insufficient data, or TimeoutError if context manager did not respond in time.

- **`ContextQuerySuccessCB`**

Callback function when ContextQuery executes successfully.

WebIDL

```
callback ContextQuerySuccessCB = void (ContextObjectArray data);
```

Signature

```
void ContextQuerySuccessCB(ContextObjectArray data);
```

Parameters

- **data**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextObjectArray
  - **Description:** An array of ContextObject type that contains the results of the Context Query.

- **`ContextRuleSearchCB`**

Callback function when a search for ContextRule executes successfully.

WebIDL

```
callback ContextRuleSearchCB = void (ContextRule[] rules);
```

Signature

```
void ContextRuleSearchCB(ContextRule[] rules);
```

Parameters

- **rules**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** array
  - **Description:** An array of ContextRule type that contains the results of the search.

- **`ContextRuleSaveCB`**

Callback function when a save for ContextRule executes successfully.

WebIDL

```
callback ContextRuleSaveCB = void (ContextRule savedRule);
```

Signature

```
void ContextRuleSaveCB(ContextRule savedRule);
```

Parameters

- **savedRule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRule
  - **Description:** The saved ContextRule with all possible modifications. If the interval provided is too low, the predetermined overridden ContextRule-specific interval value can be found on the returned ContextRule.

- **`ContextRuleEventHandler`**

Callback function when a ContextRule has been triggered.

WebIDL

```
callback ContextRuleEventHandler = void (ContextRuleEvent event);
```

Signature

```
void ContextRuleEventHandler(ContextRuleEvent event);
```

Parameters

- **event**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** ContextRuleEvent
  - **Description:**

- **`ScheduleSearchCB`**

Callback function when a search for Schedule executes successfully.

WebIDL

```
callback ScheduleSearchCB = void (Schedule[] schedules);
```

Signature

```
void ScheduleSearchCB(Schedule[] schedules);
```

Parameters

- **schedules**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** array
  - **Description:** An array of Schedule that contains the results of the search

- **`ScheduleSaveCB`**

Callback function when a save for Schedule executes successfully.

WebIDL

```
callback ScheduleSaveCB = void (Schedule savedSchedule);
```

Signature

```
void ScheduleSaveCB(Schedule savedSchedule);
```

Parameters

- **savedSchedule**
  - **Optional:** No.
  - **Nullable**: No
  - **Type:** Schedule
  - **Description:** The saved Schedule with all possible modifications. If the interval provided is too low, the predetermined overridden Schedule-specific interval value can be found on the returned Schedule.

- # Features

This section lists the URIs used to declare the features of this API. The feature URIs are used by the developer in:

- The application's config.xml file to declare requested features.
- As identifier for serviceType in the webinos Discovery API's findServices() method.

http://webinos.org/api/context

Acccess to the entire module. This feature provides access to the whole API. Security and Privacy enforcement may depend on the query or subscription requested by the developer.

http://webinos.org/api/context/query

Access to the query mechanism.

http://webinos.org/api/context/app

Access to the Application context manager. Enables an application to create a new Application ContextObject and retrieve ContextObjects created by the application

http://webinos.org/api/context/app/read

Access to read an Application ContextObject from an application

http://webinos.org/api/context/schedule/read

Access to the scheduling API in order to query for existing scheduled API calls

http://webinos.org/api/context/schedule/create

Enables the application to schedule a new API call

http://webinos.org/api/context/rules/read

Read context rules and execute them

http://webinos.org/api/context/rules/create

Allows the application to create context rules.

- # Full WebIDL

WebIDL

```
interface Context : Service {
        readonly attribute ApplicationContextManager app;

        readonly attribute ContextRuleManager rules;

        readonly attribute ScheduleManager schedule;

        void executeQuery(ContextQuery query, ContextQuerySuccessCB
successCallback, optional ContextQueryErrorCB? errorCallback);

        void schemaExists(DOMString uri, boolean uriIsApplication,
DOMString contextObjectName, SchemaExistsCB callback, optional
ErrorCB? errorCallback);
};

interface ContextObject{
        readonly attribute DOMString name;
        readonly attribute DOMString? apiUri;
        readonly attribute ApplicationURI? applicationUri;
        readonly attribute DOMString deviceId;
        readonly attribute DOMString sessionId;
        readonly attribute DOMTimeStamp timestamp;
        readonly attribute DOMString? version;
        readonly attribute ContextObjectValueArray values;
};

interface ContextObjectField{
        attribute DOMString name;
        attribute DOMString type;
};

typedef ContextObjectField[] ContextObjectFieldArray;

typedef Object[] ContextObjectValueArray;

callback SuccessCB = void ();
```

```
callback ErrorCB = void (DOMError error);


callback SchemaExistsCB = void (boolean found, DOMString? version);

callback ContextQueryErrorCB = void (DOMError error);

typedef ContextObject[] ContextObjectArray;


callback ContextQuerySuccessCB = void (ContextObjectArray data);


interface ContextQuery{
        attribute QueryFilter filter;
        attribute Object? criteria;
        attribute Object? projection;
        attribute QueryModifier? modifier;
};

interface QueryFilter{
        attribute DOMString name;
        attribute DOMString? apiUri;
        attribute ApplicationURI? applicationUri;
        attribute (DOMString or object)? deviceId;
        attribute (DOMString or object)? sessionId;
        attribute (DOMString or object)? timestamp;
        attribute (DOMString or object)? version;
};

interface QueryModifier{
        attribute integer? limit;
        attribute integer? skip;
        attribute object? sort;
};

interface ApplicationContextManager{
        void registerSchema(ApplicationURI applicationUri, DOMString
contextObjectName, DOMString version, ContextObjectFieldArray fields,
optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);

        void addData(ApplicationURI applicationUri, DOMString
contextObjectName, ContextObjectValueArray values, optional SuccessCB
successCallback, optional ErrorCB? errorCallback);
};

interface ContextRuleManager{
        void search(ContextRuleURI contextRuleUri,
ContextRuleSearchCB successCallback, optional ErrorCB?
errorCallback);

        void save(ContextRule rule, optional ContextRuleSaveCB?
saveCallback, optional ErrorCB? errorCallback);

        void remove(ContextRuleURI contextRuleUri, optional
SuccessCB? successCallback, optional ErrorCB? errorCallback);

        void renew(ContextRuleURI contextRuleUri, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
```

```
        void addListener(ContextRuleURI contextRuleURI, DOMString
listenerName, ContextRuleEventHandler callback, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);

        void removeListener(ContextRuleURI contextRuleURI, DOMString
listenerName, optional SuccessCB? successCallback, optional ErrorCB?
errorCallback);
};

callback ContextRuleSearchCB = void (ContextRule[] rules);

callback ContextRuleSaveCB = void (ContextRule savedRule);

interface ContextRule{
        attribute ContextRuleURI contextRuleUri;

        attribute ApplicationURI applicationUri;

        attribute DOMString description;

        attribute ContextQuery ruleQuery;

        attribute integer interval;

        attribute DOMTimeStamp validUntil;
};

interface ContextRuleEvent {
        readonly attribute ContextRuleURI contextRuleUri;
        readonly attribute ContextObjectArray? result;
};

callback ContextRuleEventHandler = void (ContextRuleEvent event);


typedef DOMString ContextRuleURI;

typedef DOMString ApplicationURI;

interface ScheduleManager{
        void search(DOMString? apiUri, boolean onlyOwnSchedules,
ScheduleSearchCB callback, optional ErrorCB? errorCallback);

        void save(Schedule schedule, optional ScheduleSaveCB?
saveCallback, optional ErrorCB? errorCallback);

        void remove(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);

        void renew(Schedule schedule, optional SuccessCB?
successCallback, optional ErrorCB? errorCallback);
};

callback ScheduleSearchCB = void (Schedule[] schedules);

callback ScheduleSaveCB = void (Schedule savedSchedule);

interface Schedule{

        attribute ApplicationURI applicationUri;
```

```
        attribute DOMString apiUri;

        attribute DOMString method;

        attribute object[] params;

        attribute DOMString deviceId;

        attribute integer interval;

        attribute DOMTimeStamp validUntil;
};
```