
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ


# Έξυπνες ροές εργασίας RESTful διαδικτιακών υπηρεσιών με τη βοήθεια του εργαλείου WebHookIt

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της


**ΓΚΟΤΣΕ ΜΠΛΕΡΙΝΑ**



**Επιβλέπων :** Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.


Αθήνα, Ιούλιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Έξυπνες ροές εργασίας RESTful διαδικτιακών υπηρεσιών με τη βοήθεια του εργαλείου WebHookIt

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΓΚΟΤΣΕ ΜΠΛΕΡΙΝΑ**

**Επιβλέπων :** Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την... .

| *(Υπογραφή)* | *(Υπογραφή)* | *(Υπογραφή)* |
|---|---|---|
| ..... | ..... | ..... |
| Θεοδώρα Βαρβαρίγου | Βασίλης Λούμος | Συμεών Παπαβασιλείου |
| Καθηγήτρια Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. |

Αθήνα, Ιούλιος 2015

.....

Γκότσε Μπλερίνα

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

# Ευχαριστίες

Για την πραγματοποίηση της συγκεκριμένης διπλωματικής θα επιθυμούσα να ευχαριστήσω την επιβλέπουσα καθηγήτρια κ. Θεοδώρα Βαρβαρίγου για την εμπιστοσύνη της. Επίσης για τις πολύτιμες συμβουλές και την υποστήριξη τους θα ήθελα να ευχαριστήσω τους Δρ. Φώτη Α. Αίσωπο και Δρ. Κωνσταντίνο Τσερπέ.

# Περίληψη

Οι RESTful διαδικτυακές υπηρεσίες έχουν καθιερωθεί πλεον στο διαδίκτυο ως οι πλεον κατάλληλες για τη μεταφορά, την ανάκτηση και την εισαγωγή μεγάλου όγκου δεδομένων στο διαδίκτυο. Δημιουργείται έτσι η ανάγκη για την ανάπτυξη εργαλείων που μπορούν να παράγουν και να διαχειρίζονται ροές εργασιών των RESTful διαδικτυακών υπηρεσιών. Ένα τέτοιο εργαλείο ανοιχτού λογισμικού είναι και το WebHookIt του οποίου τη λειτουργία αναπτύξαμε και επεκτείναμε στα πλαίσια την συγκεκριμένης διπλωματικής.

Παρακάτω αναλύουμε και εξετάζουμε λεπτομερώς τις διαδικτυακές υπηρεσίες και την εξέλιξή τους. Βασικό ρόλο σε αυτό παίζει η κατανοήση της υπηρεσιοστρεφούς αρχιτεκτονικής δηλαδή ποιους κανονές θα πρέπει να ακολουθεί μια υπηρεσία ώστε να χαρακτηρίζεται ως στοιχείο μιας υπηρεσιοστρεφούς αρχιτεκτονικής και ποια είναι η δομή μιας τέτοιας αρχιτεκτονικής. Σημαντικά επίσης για τη λειτουργία και την επικοινωνία αυτών των υπηρεσιών είναι τα πρωτόκολλα SOAP, UDDI, WSDL και WADL τα οποία θα εξεταστούν παρακάτω. Ένας πιο εξελιγμένος απόγονος των διαδικτυακών υπηρεσιών είναι οι RESTful διαδικτυακές υπηρεσίες οι οποίες ακολουθούν ένα συγκεκριμένο σύνολο αρχιτεκτονικών αρχών εστιάζοντας κυρίως στους πόρους ενός συστήματος. Επομένως, παρακάτω περιγράφουμε τη σημασία αυτών των υπηρεσιών καθώς και γιατί κυριάρχισαν στο διαδίκτυο. Έπειτα θα ασχοληθούμε με την ανάλυση του WebHookIt καθώς και των επιμέρους εργαλείων που το αποτελούν τα οποία είναι το NodeJS και η βάση δεδομένων MongoDB. Τέλος θα παρουσιάσουμε τις δυνατότητες του WebHookIt και την επέκταση των δυνατοτήτων αυτών ώστε να συμβαδίζουν με της ανάγκες και τις απαιτήσεις του project RADICAL για το οποίο εφαρμόστηκε η συγκεκριμένη πλατφόρμα.



**Λέξεις Κλειδιά:** <<RESTful διαδικτυακές υπηρεσίες,διαδίκτυο,WebookIt, υπηρεσιοστρεφής αρχιτεκτονική, SOAP, UDDI, WSDL, WADL, NodeJS,MongoDB, RADICAL>>

# Abstract

RESTful web services have been established as the most suitable for sending, retrieving and storing a big amount of data on the web. The above statement indicates the need to develop tools that can generate and manage workflows of RESTful web services. Such an open source tool is the WebHookIt whose functionality is developed and expanded in this document.

Below we analyze and examine in detail the web services and their evolution. An important role plays the understanding of Service Oriented Architecture which describes the rules that should be followed by a service to be qualified as part of Service Oriented Architecture and what the structure of such architecture is. In addition, there are protocols which are very crucial for the operation and communication of these services. Such protocols are SOAP, UDDI, WSDL and WADL which will be analysed below. A more advanced descendant of web services are RESTful web services which follow a specific set of architectural principles focusing on the resources of a system. Therefore, we are going to describe the importance of these services and why they dominate the web. Then we will focus on the analysis of WebHookIt and the tools that it contains which is NodeJS and MongoDB database. At the end we will present the possibilities of WebHookIt and extend these capabilities to keep pace with the needs and requirements of RADICAL project for which this platform was implemented.



**Keywords:** << RESTful web services, web, WebHookIt, Service Oriented Architecture, SOAP, UDDI, WSDL, WADL, NodeJS,MongoDB, RADICAL >>

# Πίνακας περιεχομένων

# Ευρετήριο σχημάτων και πινάκων

# 1

# *Γενικά για τις διαδικτυακές υπηρεσίες*

.

## *1.1 Υπηρεσίες*

Οι απαιτήσεις της σημερινής κοινωνίας έχουν καταστήσει τις υπηρεσίες στο διαδίκτυο απαραίτητες για τη διεκπεραίωση των διαφόρων λειτουργιών και τη διευκόλυνση του κοινωνικού συνόλου. Η εμφάνιση του Internet of Things (Διαδίκτυο των υπηρεσιών) εισάγει εκτός από την απλή διασύνδεση βάσεων δεδομένων και τη δυνατότητα δυναμικής εξυπηρέτησης. Πλέον πόλεις και χώρες ολόκληρες βασίζονται στην ηλεκτρονική διακυβέρνηση και παροτρύνουν τους πολίτες τους να χρησιμοποιούν όλο και περισσότερο τις διαδικτυακές τους υπηρεσίες. Άλλο παράδειγμα είναι το e-Learning (ηλεκτρονική μάθηση) όπου εκατομμύρια χρήστες πλέον συμμετέχουν σε πλατφόρμες στο διαδίκτυο για να παρακολουθήσουν μαθήματα και σεμινάρια μέσω διαδικτύου. Επιπλέον η μεγάλη ανταπόκριση του ευρύ κοινωνικού συνόλου στα κοινωνικά δίκτυα έχει κάνει τη δυναμική των υπηρεσιών στο διαδίκτυο τεράστια. Μέσω των κοινωνικών δικτύων οι χρήστες μπορούν να προβάλουν τις ανάγκες, τις απόψεις και τις επιθυμίες τους ενώ οι υπηρεσίες θα πρέπει να έχουν τη "νοημοσύνη" να τις αντιληφθούν και να παρέχουν τις απαραίτητες λύσεις στα θέματα των χρηστών. Αυτός είναι και ο λόγος που διάφορες εταιρίες και και ερευνητικά κέντρα έχουν στρέψει την προσοχή τους στα κοινωνικά δίκτυα. Έτσι θα

έχουν τη δυνατότητα να αναπτύξουν υπηρεσίες και να τις εξελίξουν ανάλογα με τις ανάγκες των χρηστών.

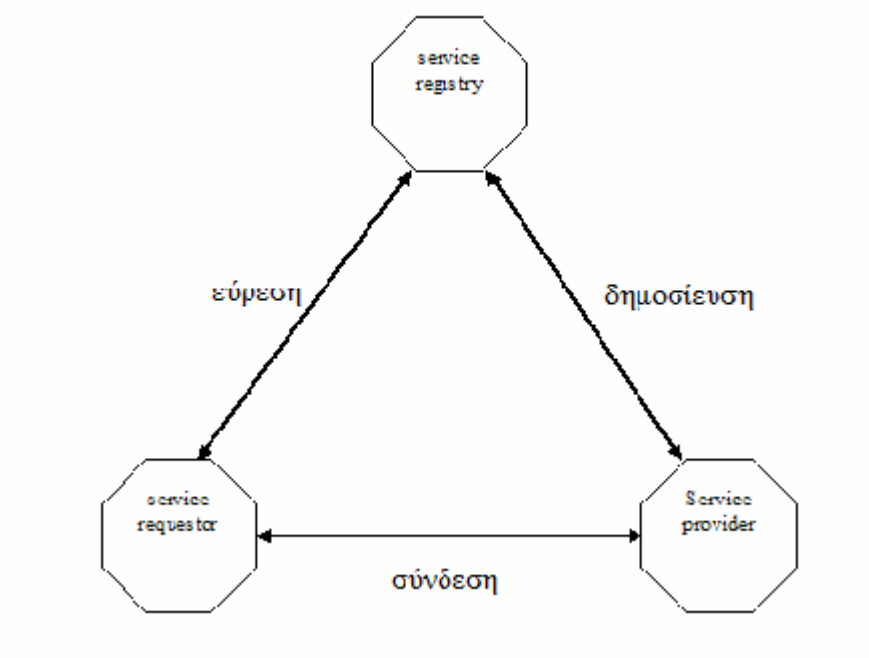
Μέσα από αυτό σταδιακά δημιουργείται μια μεγάλη ανοιχτή αγορά υπηρεσιών η οποία καθορίζει την την πολυπλοκότητα αυτών και παρέχουν τις βασικές λειτουργικότητες οι οποίες μπορούν να συνδυαστούν για να δημιουργήσουν μεγαλύτερες ροές εργασίων (workflows).

## *1.2 Υπηρεσιοστρεφής Αρχιτεκτονική*

Προηγουμένως μιλήσαμε γενικά για υπηρεσίες διαδικτύου. Οι υπηρεσίες όμως αυτές δεν χαρακτηρίζουν μια υπηρεσιοστρεφή αρχιτεκτονική. Μια υπηρεσιοστρεφής αρχιτεκτονική είναι ουσιαστικά μια συλλογή υπηρεσιών. Οι υπηρεσίες αυτές πρέπει να επικοινωνούν μεταξύ τους. Η επικοινωνία μπορεί να περιλαμβάνει είτε απλή μεταφορά δεδομένων ή θα μπορούσε να περιλαμβάνει δύο ή περισσότερες υπηρεσίες συντονισμένες ώστε να εκτελούν μια συγκεκριμένη δραστηριότητα. Επίσης χρειάζονται και κάποια μέσα για τη διασύνδεση των υπηρεσιών μεταξύ τους.

Ουσιαστικά είναι μια τεχνική που περιλαμβάνει μια αλληλεπίδραση μεταξύ "χαλαρά" συνδεδεμένων υπηρεσιών που λειτουργούν ανεξάρτητα. Μια υπηρεσιοστρεφής αρχιτεκτονική για να είναι αποτελεσματική, χρειάζεται υπηρεσίες των οποίων οι λειτουργίες είναι σαφώς καθορισμένες, αυτόνομες, και μην εξαρτώνται από το πλαίσιο ή την κατάσταση των άλλων υπηρεσιών. Κάθε υπηρεσία σε τέτοιες αρχιτεκτονικές είναι μια μονάδα εργασίας που γίνεται διαθέσιμη από έναν πάροχο, ώστε να παρέχει τα επιθυμητά τελικά αποτελέσματα στον καταναλωτή υπηρεσίας. Βάσει αυτού, η υιοθέτηση σε μια εφαρμογή μιας SOA δομής σημαίνει αυτόματα την αποδοχή κάποιων σχεδιαστικών αρχών και πρόσθετων τεχνολογιών ως βασικού τμήματος του τεχνικού περιβάλλοντός της. Η τεχνολογία των διαδικτυακών υπηρεσιών είναι η πιο συνήθης τεχνολογία σύνδεσης για τις υπηρεσιοστρεφείς αρχιτεκτονικές.

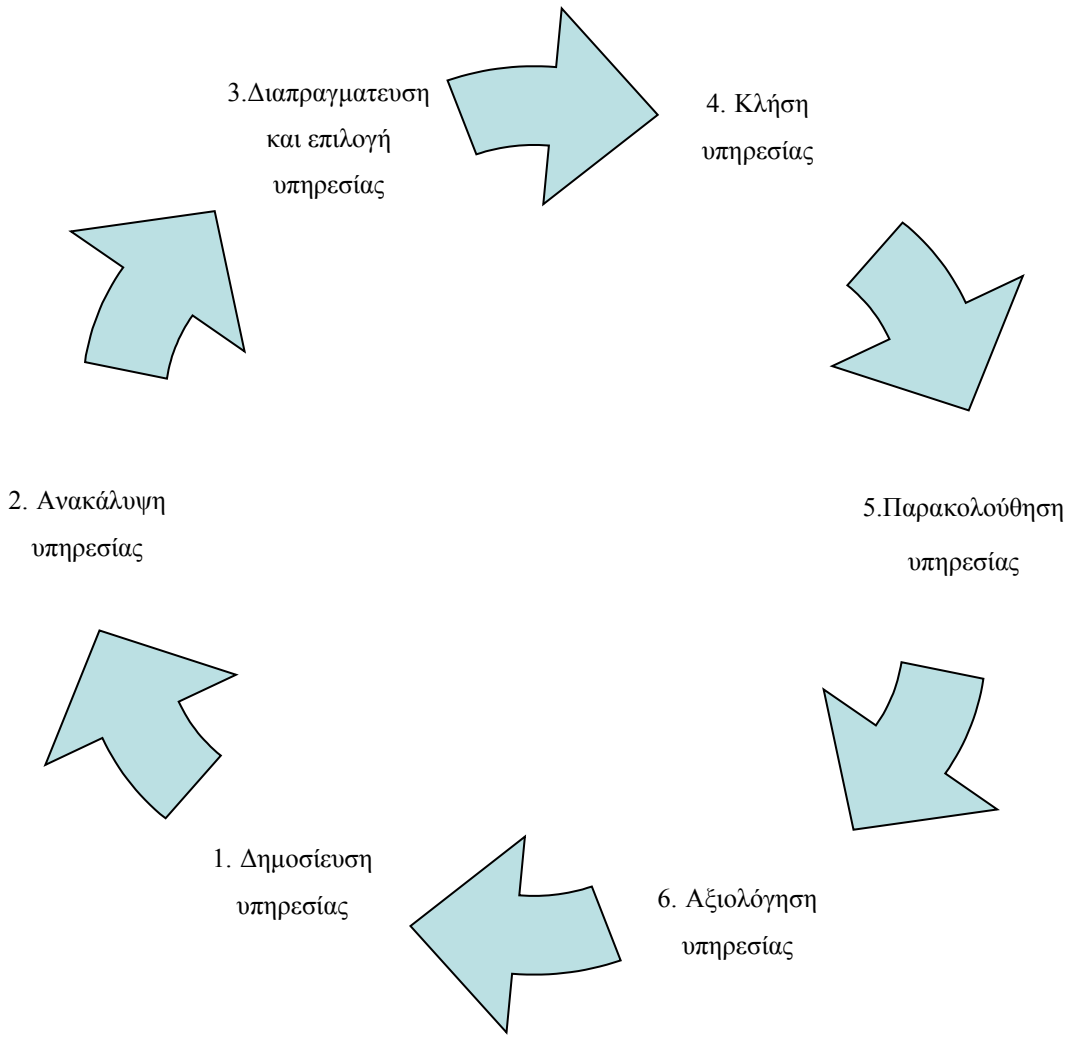
Οι βασικοί ρόλοι και λειτουργίες στην αρχιτεκτονική αυτή παρουσιάζονται στο Σχήμα 1 (Menychtas, 2009):




**Εικόνα 1: Δομή Υπηρεσιοστρεφούς Αρχιτεκτονικής**

Όπως παρατηρούμε και από το σχήμα υπάρχει μια σχέση εξυπηρετητή(server) – πελάτη(client) ανάμεσα στον πάροχο υπηρεσιών( service provider ) τον πελάτη που ζητά την υπηρεσία( Service requestor). Ο πάροχος δέχεται μηνύματα και κλήσεις από τους ζητούντες την υπηρεσία και αναλόγως παρέχει την υπηρεσία. Ο πάροχος επίσης δημιουργεί την περιγραφή της υπηρεσίας (service description) και τη δημοσιεύει σε κάποιο κατάλογο-οδηγό υπηρεσιών Universal Description, Discovery and Integration – UDDI). Από την άλλη πλευρά, ο πελάτης αναζητά μια συκεκριμένη υπηρεσία μέσω της περιγραφής της σε κάποιο κατάλογο υπηρεσιών (service registry) και στη συνέχεια καλεί την επιθυμητή υπηρεσία. Επομένως ο ρόλος του καταλόγου είναι να φέρει τον πάροχο και τον πελάτη σε επαφή. Έπειτα, ο εξυπηρετητής και ο πελάτης συνεχίζουν χωρίς να χρειάζονται την παρέμβαση του καταλόγου.

**Κύκλος ζωής των υπηρεσιών σε SOA**




Ο κύκλος ζωής των υπηρεσιών σε SOA είναι γενικά προδιαγεγραμμένος και εμπεριέχει τα παρακάτω στάδια:

- Δημοσίευση υπηρεσίας: Ο πάροχος υπηρεσιών θέτει μια δημιουργηθείσα υπηρεσία δημόσια διαθέσιμη, ώστε να μπορεί να ανακαλυφθεί από τους ενδιαφερόμενους πελάτες.
- Ανακάλυψη υπηρεσίας: Ο πελάτης ανακαλύπτει την επιθυμητή υπηρεσία μέσω συγκεκριμένων πρωτοκόλλων και μητρώων.

- Διαπραγμάτευση και επιλογή υπηρεσίας: Ο πελάτης διαπραγματεύεται με τον πάροχο τις παραμέτρους κλήσης και τη χρέωση της υπηρεσίας. Κατόπιν συμφωνίας, ο πελάτης επιλέγει τη συγκεκριμένη υπηρεσία για χρησιμοποίηση
- Κλήση υπηρεσίας: Ο πελάτης καλεί την υπηρεσία δίνοντας δεδομένα εισόδου και περιμένοντας την απόκριση και τα ζητούμενα αποτελέσματα.
- Παρακολούθηση υπηρεσίας: Ο πάροχος (ή και ο πελάτης) μπορούν να παρακολουθούν καθ' όλη τη διάρκεια της κλήσης τη χρησιμοποίηση των πόρων (ή τις τιμές των παραμέτρων κλήσης)
- Αξιολόγηση υπηρεσίας: Μετά το πέρας της κλήσης και σύμφωνα με την παρακολούθηση κατά τη διάρκειά της, οι δύο συμβαλλόμενοι μπορούν να αξιολογήσουν κατά πόσο τηρήθηκε η προαναφερθείσα συμφωνία.

## *1.3 Διαδικτυακές Υπηρεσίες*

Ο πιο γνωστός τύπος υπηρεσιών της υπηρεσιοστρεφούς αρχιτεκτονικής είναι οι διαδικτυακές υπηρεσίες. Οι διαδικτυακές υπηρεσίες είναι αυτόνομες, προσαρμόσιμες, κατανεμημένες, δυναμικές εφαρμογές που μπορούν να περιγραφούν, να δημοσιευθούν, να βρεθούν και να κληθούν μέσω του δικτύου για τη δημιουργία προϊόντων και διαδικασιών. Μπορούν να περιγράφουν έναν τυποποιημένο τρόπο για την ενσωμάτωση των εφαρμογών βασισμένων στο διαδίκτυο χρησιμοποιώντας τα ανοιχτά πρότυπα XML, SOAP, WSDL και UDDI πάνω σε διαδικτυακά πρωτόκολλα. Το XML χρησιμοποιείται για να επισημάνει τα δεδομένα, το SOAP χρησιμοποιείται για τη μεταφορά των δεδομένων, το WSDL χρησιμοποιείται για να περιγράψει τις διαθέσιμες υπηρεσίες και το UDDI για να καταχωρήσει ποιες υπηρεσίες είναι διαθέσιμες.

Σε αντίθεση με τα παραδοσιακά μοντέλα πελάτη/εξυπηρετητή όπως οι διαδικτυακοί εξυπηρετητές ή το σύστημα διαδικτυακών σελίδων,οι διαδικτυακές υπηρεσίες δεν

παρέχουν στον χρήστη ένα γραφικό περιβάλλον(GUI). Οι διαδικτυακές υπηρεσίες αντίθετα μοιράζονται δεδομένα επιχειρηματικής λογικής και διαδικασίες μέσα από μια προγραμματική υπηρεσία μέσω ενός δικτύου. Η διεπαφή των εφαρμογών, όχι οι χρήστες. Οι προγραμματιστές μπορούν στη συνέχεια να προσθέσουν την διαδικτυακή υπηρεσία σε ένα GUI (όπως σε μια ιστοσελίδα ή ένα εκτελέσιμο πρόγραμμα) για να προσφέρουν ειδική λειτουργικότητα στους χρήστες.

Οι διαδικτυακές υπηρεσίες επιτρέπουν διαφορετικές εφαρμογές από διαφορετικές πηγές να επικοινωνούν μεταξύ τους χωρίς χρονοβόρες διαδικασίες. Οι υπηρεσίες μπορούν να συμπεριλαμβάνουν προγράμματα, αντικείμενα, μηνύματα και αρχεία και επειδή όλη η επικοινωνία είναι σε XML, οι διαδικτυακές υπηρεσίες δεν είναι δεμένες με κάποιο λειτουργικό σύστημα ή γλώσσα προγραμματισμού. Για παράδειγμα, η Java μπορεί να μιλήσει με Perl, εφαρμογές των Windows μπορούν να μιλήσουν με εφαρμογές UNIX.

Επομένως διαδικτυακή υπηρεσία θεωρείται μια υπηρεσία που:

- Είναι διαθέσιμη μέσω του διαδικτύου ή ιδιωτικού δίκτυου (intranet).
- Χρησιμοποιεί ένα τυποποιημένο σύστημα ανταλλαγής μηνυμάτων XML.
- Δεν είναι εξαρτημένη από κάποιο λειτουργικό σύστημα ή κάποια γλώσσα προγραμματισμού.
- Αυτό-περιγράφεται μέσω μιας κοινής XML γραμματικής.
- Μπορεί να εντοπιστεί μέσω μιας απλής μηχανής εύρεσης.

### *1.3.1 Χαρακτηριστικά*

- **Βασίζεται σε XML**

Οι διαδικτυακές υπηρεσίες χρησιμοποιούν τη γλώσσα XML για την αναπαράσταση δεδομένων και τη μεταφορά δεδομένων στρωμάτων. Με τη χρήση της XML εξαλείφεται κάθε δέσμευση δικτύωσης, λειτουργικού συστήματος, ή πλατφόρμας. Έτσι, οι εφαρμογές που είναι βασισμένες σε διαδικτυακές υπηρεσίες έχουν πολύ υψηλή διαλειτουργική εφαρμογή στο επίπεδο του πυρήνα τους.

- **Είναι "Χαλαρά" συνδεδεμένες**

Ένας καταναλωτής μιας διαδικτυακής υπηρεσίας δεν συνδέεται με την εν λόγω υπηρεσία άμεσα. Η διεπαφή της διαδικτυακής υπηρεσίας μπορεί να αλλάξει με την πάροδο του χρόνου, χωρίς να θέτει σε κίνδυνο την ικανότητα του πελάτη να αλληλεπιδράσει με την υπηρεσία. Ένα στενά συνδεδεμένο σύστημα συνεπάγεται ότι ο πελάτης και η λογική του εξυπηρετητή είναι στενά δεμένες μεταξύ τους, πράγμα που σημαίνει ότι αν μία διεπαφή αλλάξει, η άλλη θα πρέπει επίσης να ενημερωθεί. Η υιοθέτηση μιας" χαλαρής" συζευγμένης αρχιτεκτονικής τείνει να καταστήσει τα συστήματα λογισμικού πιο εύχρηστα και επιτρέπει απλούστερη ενσωμάτωση μεταξύ των διαφορετικών συστημάτων.

- **Είναι σύνθετες**

Οι αντικειμενοστραφείς τεχνολογίες όπως η Java δημοσιεύουν τις υπηρεσίες τους μέσω ατομικών μεθόδων. Μια μεμονωμένη μέθοδος είναι μια πολύ λεπτή επέμβαση για να παρέχει κάποια χρήσιμη δυνατότητα σε εταιρικό επίπεδο. Η οικοδόμηση ενός προγράμματος Java από το μηδέν απαιτεί τη δημιουργία πολλών μεθόδων που αποτελούν μικρά μέρη και στη συνέχεια συγκροτούνται σε μια σύνθετη υπηρεσία που καταναλώνεται είτε από έναν πελάτη ή άλλη υπηρεσία. Οι λειτουργίες και οι διασυνδέσεις που δημοσιεύουν πρέπει να είναι σύνθετες.

- **Έχουν την ικανότητα να είναι σύγχρονες ή ασύγχρονες**

Ο συγχρονισμός αναφέρεται στη δέσμευση του πελάτη στην εκτέλεση της υπηρεσίας. Στις σύγχρονες επικλήσεις, ο πελάτης σταματάει και περιμένει την υπηρεσία να ολοκληρώσει την λειτουργία του πριν συνεχίσει. Στις ασύγχρονες λειτουργίες επιτρέπει σε έναν πελάτη να επικαλεστεί μια υπηρεσία και στη συνέχεια να εκτελέσει άλλες λειτουργίες. Ασύγχρονοι πελάτες ανακτούν το αποτέλεσμά τους σε μεταγενέστερο χρονικό σημείο, ενώ σύγχρονοι πελάτες λαμβάνουν το αποτέλεσμά τους, όταν η υπηρεσία έχει ολοκληρωθεί. Η ασύγχρονη ικανότητα αποτελεί βασικό παράγοντα για την δυνατότητα χαλαρά συζευγμένων συστημάτων.

- **Υποστηρίζουν απομακρυσμένες διαδικασίες κλήσης (RPCs)**

Οι υπηρεσίες αυτές επιτρέπουν στους πελάτες να επικαλεστούν διαδικασίες, λειτουργίες και μεθόδους σε απομακρυσμένα αντικείμενα χρησιμοποιώντας ένα πρωτόκολλο που βασίζεται σε XML. Οι απομακρυσμένες διαδικασίες δημοσιεύουν τις παραμέτρους εισόδου και εξόδου που μια διαδικτυακή υπηρεσία πρέπει να υποστηρίζει. Η αναπτυξιακή συνιστώσα μέσω του Enterprise JavaBeans (EJBs) και

του .NET έχει γίνει όλο και περισσότερο ένα μέρος των αρχιτεκτονικών και των επιχειρήσεων αναπτύξης κατά τα τελευταία δύο χρόνια. Και οι δύο τεχνολογίες διανέμονται και είναι προσβάσιμες μέσω μιας ποικιλίας μηχανισμών RPC. Μια διαδικτυακή υπηρεσία υποστηρίζει RPC με την παροχή υπηρεσιών από μόνη της, ισοδύναμες με εκείνες ενός παραδοσιακού στοιχείου, ή με τη μετάφραση των εισερχόμενων επικλήσεων σε μια επίκληση ενός EJB ή .NET στοιχείου.

- **Υποστηρίζει την ανταλλαγή εγγράφων**

Ένα από τα βασικά πλεονεκτήματα της XML είναι γενικό τρόπο του που αντιπροσωπεύουν όχι μόνο δεδομένα, αλλά και σύνθετα έγγραφα. Τα έγγραφα αυτά μπορεί να είναι απλά, όπως όταν εκπροσωπούν μια τρέχουσα διεύθυνση, ή μπορεί να είναι πολύπλοκα, όπως όταν αντιπροσωπεύουν ένα ολόκληρο βιβλίο ή ένα RFQ. Οι διαδικτυακές υπηρεσίες υποστηρίζουν τη διαφανή ανταλλαγή εγγράφων για τη διευκόλυνση της ολοκλήρωσης των λειτουργιών.

### *1.3.2 Χρησιμότητα*

Οι διαδικτυακές υπηρεσίες φαίνονται όλο και πιο διαδεδομένες. Οι λόγοι που καθιστούν αυτές τις υπηρεσίες όλο και πιο ιδανικές είναι οι εξής;

- **Δημοσιεύουν την υπάρχουσα λειτουργία στο δίκτυο:**

Μια διαδικτυακή υπηρεσία είναι μια μονάδα διαχειριζόμενου κώδικα που μπορεί να γίνει απομακρυσμένη κλήση μέσω HTTP, δηλαδή, μπορεί να ενεργοποιηθεί με τη χρήση αιτημάτων HTTP. Έτσι, οι διαδικτυακές υπηρεσίες δίνουν τη δυνατότητα να δημοσιεύσουν τη λειτουργικότητα του υπάρχοντα κώδικα σας μέσω του δικτύου. Από τη στιγμή που εκτίθεται στο δίκτυο, άλλη εφαρμογή μπορεί να χρησιμοποιήσει τη λειτουργικότητα του προγράμματός αυτού.

- **Σύνδεση διαφορετικών εφαρμογών, δηλαδή Διαλειτουργικότητα:**

Οι διαδικτυακές υπηρεσίες επιτρέπουν διαφορετικές εφαρμογές να μιλούν ο ένας στον άλλο και να μοιράζονται δεδομένα και υπηρεσίες μεταξύ τους. Άλλες εφαρμογές μπορούν επίσης να χρησιμοποιήσουν τις υπηρεσίες αυτές . Για παράδειγμα, η VB ή εφαρμογές .NET μπορεί να μιλούν σε διαδικτυακές υπηρεσίες Java και το αντίστροφο. Έτσι, οι διαδικτυακές υπηρεσίες χρησιμοποιούνται για να κάνουν την πλατφόρμα εφαρμογών και τεχνολογίας ανεξάρτητη.

- **Τυποποιημένο πρωτόκολλο:**

Οι διαδικτυακές υπηρεσίες χρησιμοποιούν τυποποιημένο πρωτόκολλο βιομηχανικού προτύπου για την επικοινωνία. Όλα τα τέσσερα στρώματα (Υπηρεσία Μεταφορών, XML μηνύματα, Περιγραφή υπηρεσιών και Service Discovery στρώματα) χρησιμοποιούν το καλά καθορισμένο πρωτόκολλο στη στοίβα πρωτοκόλλου διαδικτυακών υπηρεσιών.

- **Χαμηλό κόστος της επικοινωνίας:**

Οι διαδικτυακές υπηρεσίες χρησιμοποιούν τα SOAP πάνω από το πρωτόκολλο HTTP για την επικοινωνία, έτσι ώστε να μπορούν να χρησιμοποιούν το υπάρχον χαμηλό κόστος του διαδικτύου για την υλοποίηση των Web Services.

### *1.3.3 Στοίβα πρωτοκόλλου διαδικτυακών υπηρεσιών*

Η αρχιτεκτονική των διαδικτυακών υπηρεσιών μπορεί επίσης να εξετασόμενης στοίβας πρωτοκόλλου. Η στοίβα αυτή εξακολουθεί να εξελίσσεται, αλλά σήμερα έχει τέσσερα κύρια στρώματα.

- **Υπηρεσία μεταφοράς**

Αυτό το στρώμα είναι υπεύθυνο για την μεταφορά των μηνυμάτων μεταξύ των εφαρμογών. Επί του παρόντος, αυτό το στρώμα περιλαμβάνει το πρωτόκολλο Hypertext Transfer (HTTP), το πρωτόκολλο Simple Mail Transfer Protocol (SMTP), το πρωτόκολλο μεταφοράς αρχείων (FTP), και νεότερα πρωτόκολλα, όπως το Blocks Extensible Exchange Protocol (BEEP).

- **Μηνυμάτων XML**

Αυτό το στρώμα είναι υπεύθυνο για την κωδικοποίηση μηνυμάτων σε μια κοινή μορφή XML, έτσι ώστε τα μηνύματα μπορεί να γίνει κατανοητή σε κάθε άκρο.

- **Περιγραφή υπηρεσίας**

Αυτό το στρώμα είναι υπεύθυνο για την περιγραφή της δημόσια διεπαφής σε μια συγκεκριμένη διαδικτυακή υπηρεσία. Επί του παρόντος, περιγραφή των υπηρεσιών γίνεται μέσα από την υπηρεσία Web Description Language (WSDL).

- **Ανακάλυψη υπηρεσίας**

Αυτό το στρώμα είναι υπεύθυνο για τη συγκέντρωση των υπηρεσιών σε ένα κοινό μητρώο, και την παροχή εύκολης λειτουργικότητας για δημοσίευση/εύρεσης. Επί του παρόντος, η ανακάλυψη των υπηρεσιών γίνεται μέσα από Universal Description, Discovery, και Integration (UDDI).

Δεδομένου ότι οι υπηρεσίες Ιστού εξελίσσονται, μπορούν να προστεθούν επιπλέον στρώσεις, και επιπλέον τεχνολογίες μπορεί να προστεθεί σε κάθε στρώμα.

## *1.4 Simple Object Access Protocol(SOAP)*

Το SOAP (Simple Object Access Protocol) είναι ένα πρωτόκολλο επικοινωνίας σχεδιασμένο να επικοινωνεί μέσω του διαδικτύου. Το SOAP μπορεί να επεκτείνει το HTTP για μηνύματα XML. Ακόμη έχει τη δυνατότητα να ανταλλάξει αρχεία ή να καλέσει κάποια απομακρυσμένη διαδικασία. Ένα άλλο πλεονέκτημα που έχει είναι ότι είναι ανεξάρτητο πλατφόρμας και γλώσσα. Το SOAP είναι ο XML τρόπος να προσδιορίσει κάποιος τι πληροφορίες στέλνονται και πώς.

Το SOAP μπορεί να χρησιμοποιηθεί σε πολλά συστήματα μηνυμάτων και μπορεί να μεταφερθεί μέσω μιας μεγάλης ποικιλίας πρωτοκόλλων, ωστόσο στην αρχή το SOAP εστίαζε στις κλήσεις απομακρυσμένων διαδικασιών μέσω HTTP.

Το SOAP δίνει τη δυνατότητα στις εφαρμογές πελάτη να συνδέονται εύκολα σε απομακρυσμένες υπηρεσίες και να καλούν απομακρυσμένες μεθόδους.

### *1.4.1 Δομή*

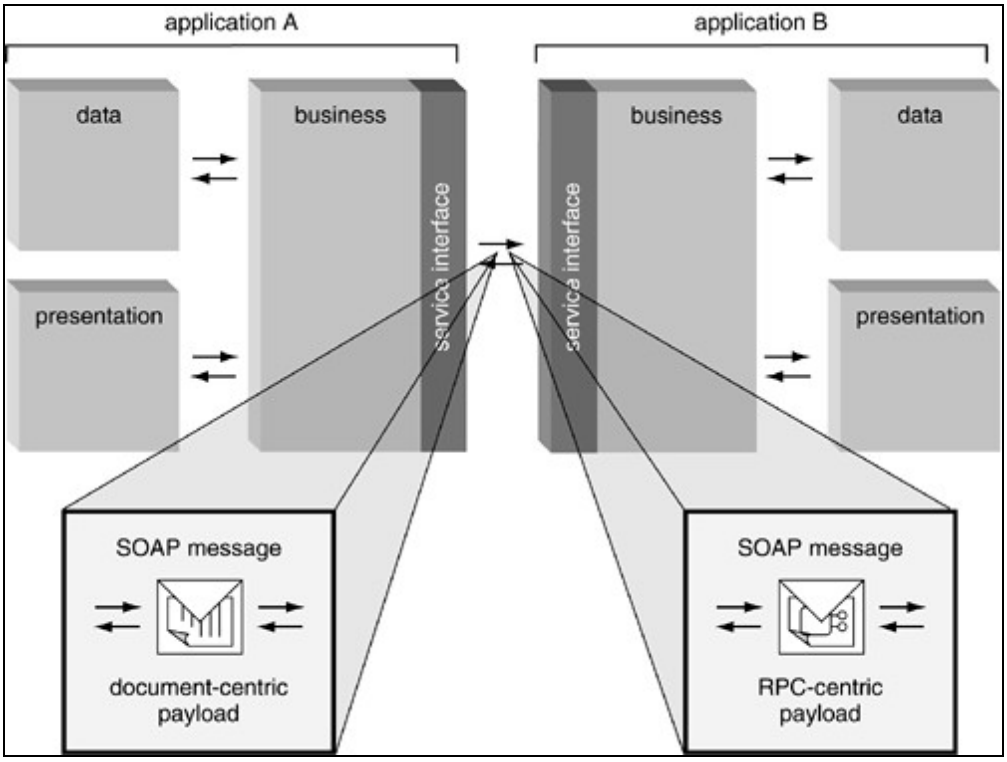
Ένα μήνυμα SOAP είναι ένα απλό XML αντικείμενο που περιέχει τα ακόλουθα αντικείμενα:

- **Envelope: (Υποχρεωτικό):** Καθορίζει την αρχή και το τέλος του μηνύματος.
- **Header(Προαιρετικό)**: Περιέχει κάθε προαιρετική παράμετρο του μηνύματος που χρησιμοποιείται για να επεξεργαστεί το μήνυμα,
- **Body(Υποχρεωτικό)**: Περιέχει τα XML δεδομένα που περιλαμβάνουν το μήνυμα που πρέπει να σταλεί.
- **Fault(Προαιρετικό):** Ένα προαιρετικό αντικείμενο Fault που παρέχει τις πληροφορίες για τα λάθη που προέκυψαν κατά την επεξεργασία του μηνύματος.

```
<?xml version="1.0"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Header>
   ...
   ...
</SOAP-ENV:Header>
<SOAP-ENV:Body>

   ...
   ...
   <SOAP-ENV:Fault>
      ...
      ...
   </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP_ENV:Envelope>
```




**Εικόνα 2: Χρήση SOAP Πρωτοκόλλου σε εφαρμογές**

## *1.5 Web Services Description Language (WSDL)*

Το WSDL είναι ένα πρωτόκολλο που βασίζεται σε XML για την ανταλλαγή πληροφοριών σε μη κεντρικά και κατανεμημένα περιβάλλοντα. Περιγράφει το τρόπο με τον οποίο μπορούμε να αποκτήσουμε πρόσβαση στις διαδικτυακές υπηρεσίες και τι λειτουργίες θα εκτελέσει. Χρησιμοποιεί και αποτελεί αναπόσπαστο κομμάτι του UDDI.

Η WSDL 1.1 υποβλήθηκε ως ένα W3C Note από την Ariba, IBM και τη Microsoft για την περιγραφή υπηρεσιών για το W3C XML Activity σε πρωτόκολλα XML το Μάρτιο του 2001.

Το WSDL χρησιμοποιείται συχνά σε συνδυασμό με το SOAP και το XML Schema για την παροχή διαδικτυακών υπηρεσιών μέσω του διαδικτύου. Ένα πρόγραμμα πελάτη συνδέεται σε μια διαδικτυακή υπηρεσία και μπορεί να διαβάσει το WSDL για να καθορίσει ποιες λειτουργίες είναι διαθέσιμες στον εξυπηρετητή. Οποιοιδήποτε ειδικοί τύποι δεδομένων που μπορεί να χρησιμοποιηθούν είναι ενσωματωμένοι στο αρχείο WSDL με τη μορφή του XML Schema. Ο πελάτης μπορεί στη συνέχεια να

χρησιμοποιήσει το SOAP για να καλέσει στην πραγματικότητα μία από τις λειτουργίες που αναφέρονται στο WSDL

### *1.5.1 Δομή*

Ένα αρχείο WSDL περιέχει τα ακόλουθα στοιχεία:

Definition: Είναι το ριζικό στοιχείο όλων των εγγράφων WSDL. Ορίζει το όνομα της διαδικτυακής υπηρεσίας, δηλώνει πολλαπλά ονόματα που χρησιμοποιούνται σε όλο το υπόλοιπο του εγγράφου, και περιέχει όλα τα στοιχεία των υπηρεσιών που περιγράφονται εδώ.

Τύποι δεδομένων: Οι τύποι δεδομένων που θα χρησιμοποιηθούν στα μηνύματα είναι σε μορφή XML σχήματα.

Message: Πρόκειται για ένα αφηρημένο ορισμό των δεδομένων, με τη μορφή ενός μηνύματος που παρουσιάζεται είτε ως ολόκληρο το έγγραφο ή ως προσδιορισμοί για τη περιγραφή σε μια κλήση μεθόδου.

Operation: Είναι ο αφηρημένος ορισμός της λειτουργίας για ένα μήνυμα, όπως η ονομασία μιας μεθόδου, η ουρά μηνυμάτων, ή η επιχειρηματική διαδικασία, η οποία θα δέχεται και θα επεξεργάζεται το μήνυμα.

Port type : Είναι ένα αφηρημένο σύνολο λειτουργιών αποτυπωμένα σε ένα ή περισσότερα end-points, που καθορίζουν τη συλλογή των εργασιών για μια σύνδεση. Η συλλογή των εργασιών, όπως είναι αφηρημένη, μπορεί να αντιστοιχιστεί σε πολλαπλές μεταφορές μέσω διαφόρων συνδέσεων.

Binding: Είναι οι συγκεκριμένες μορφές πρωτοκόλλων και δεδομένων για τις λειτουργίες και τα μηνύματα που καθορίζονται για ένα συγκεκριμένο τύπο θύρας.

Port: Είναι ένας συνδυασμός ενός binding και μιας διεύθυνσης δικτύου, παρέχοντας τη διεύθυνση στόχο της επικοινωνίας υπηρεσιών.

Service: Πρόκειται για μια συλλογή σχετικών end-points( τελικών σημείων ) που περιλαμβάνει τους ορισμούς των υπηρεσιών σε ένα αρχείο. Οι υπηρεσίες χαρτογραφούν το binding με τη Port(θύρα) και περιλαμβάνουν οποιοδήποτε ορισμό επεκτασιμότητας.

Εκτός από αυτά τα βασικά στοιχεία, η προδιαγραφή WSDL καθορίζει επίσης τα ακόλουθα στοιχεία χρησιμότητας.

Documentation: Αυτό το στοιχείο χρησιμοποιείται για να παρέχει ευανάγνωστο documentation και να μπορούν να περιληφθούν μέσα σε οποιοδήποτε άλλο στοιχείο WSDL.

Import: Αυτό το στοιχείο χρησιμοποιείται για την εισαγωγή άλλων εγγράφων WSDL ή XML Schema.

Τα μέρη του WSDL συνήθως παράγονται αυτόματα με τη χρήση εργαλείων που είναι ενημερωμένα για τις διαδικτυακές υπηρεσίες.

Αυτή είναι η βασική δομή του εγγράφου WSDL:

```
<definitions>
  <types>
    definition of types.....
  </types>

  <message>
    definition of a message....
  </message>

  <portType>
    <operation>
      definition of a operation.....
    </operation>
  </portType>

  <binding>
    definition of a binding....
  </binding>
```

```
  <service>
    definition of a service....
  </service>
</definitions>
```

Ένα έγγραφο WSDL μπορεί επίσης να περιέχει και άλλα στοιχεία, όπως στοιχεία επέκτασης και ένα στοιχείο υπηρεσίας που καθιστά δυνατό να ομαδοποιηθούν οι ορισμοί των διαφόρων διαδικτυακών υπηρεσιών σε ένα ενιαίο έγγραφο WSDL

## *1.6 Web Application Description Language(WADL)*

Η Web Application Description Language(WADL) είναι μια XML περιγραφή των διαδικτυακών εφαρμογών βασισμένων σε HTTP που είναι αναγνώσιμη από τις μηχανές (συνήθως REST διαδικτυακές υπηρεσίες). Η WADL μοντελοποιεί τους πόρους που παρέχονται από μια υπηρεσία και τις σχέσεις μεταξύ τους. Η WADL αποσκοπεί στην απλούστευση της επαναχρησιμοποίησης των διαδικτυακών υπηρεσιών που βασίζονται στην υπάρχουσα HTTP αρχιτεκτονική του διαδικτύου. Είναι ανεξάρτητη πλατφόρμας και γλώσσας και έχει ως στόχο την προώθηση της επαναχρησιμοποίησης των εφαρμογών πέρα από τη βασική λειτουργία σε ένα διαδικτυακόφυλλομετρητή.

Η WADL υποβλήθηκε στην World Wide Web Consortium(W3C)

από την Sun Microsystems στις 31 Αυγούστου 2009, αλλά η W3C δεν σχεδιάζει ακόμη να τη τυποποιήσει.Η WADL είναι το REST ισοδύναμο της *Web* Services Description Language( WSDL), το οποίο μπορεί επίσης να χρησιμοποιηθεί για να περιγράψει REST διαδικτυακές υπηρεσίες.

## *1.7 Universal Description, Discovery, and Integration (UDDI)*

Το UDDI(**Universal Description, Discovery, and Integration**) βασίζεται σε XML και χρησιμοποιείται για την περιγραφή, τη δημοσίευση και την εύρεση διαδικτυακών υπηρεσιών. Αποτελεί μια προδιαγραφή για ένα κατανεμημένο μητρώο διαδικτυακών υπηρεσιών. Είναι ένα ανεξάρτητο πλατφόρμας και ανοιχτό framework που μπορεί να επικοινωνήσει με πρωτόκολλα SOAP, CORBA και Java RMI. Το UDDI χρησιμοποιεί το WSDL για να περιγράψει τις διεπαφές των διαδικτυακών υπηρεσιών. Επιτρέπει τις υπηρεσίες να ανακαλύπτουν η μια την άλλη και να προσδιορίζουν πως να αλληλεπιδρούν μέσω του διαδικτύου.

Περιέχει δύο τμήματα:

- Ένα μητρώο των μεταδεδομένων όλων διαδικτυακών υπηρεσιών, συμπεριλαμβανομένου ενός δείκτη για την περιγραφή WSDL της υπηρεσίας.

- Ένα σύνολο WSDL ορισμών τύπων θυρών για το χειρισμό και την αναζήτηση του εν λόγω μητρώου.

# 2

# *RESTful διαδικτυακές υπηρεσίες*

Το REST ορίζει ένα σύνολο αρχιτεκτονικών αρχών με το οποίο μπορείτε να σχεδιάσετε τις διαδικτυακές υπηρεσίες που εστιάζουν στους πόρους ενός συστήματος, συμπεριλαμβανομένου του τρόπου που αντιμετωπίζονται οι καταστάσεις των πόρων και μεταφέρονται μέσω HTTP σε ένα ευρύ φάσμα πελατών γραμμένα σε διαφορετικές γλώσσες. Αν μετρήσουμε τον αριθμό των διαδικτυακών υπηρεσιών που το χρησιμοποιούν αυτό, το REST έχει γίνει τα τελευταία χρόνια το κυρίαρχο σχεδιαστικό μοντέλο των διαδικτυακών υπηρεσιών. Στην πραγματικότητα, το REST είχε τόσο μεγάλο αντίκτυπο στο διαδίκτυο που έχει κυρίως εκτοπίσει τα SOAP και την σχεδίαση διεπαφών βασισμένων σε WSDL, διότι είναι πολύ πιο απλουστευμένο στην χρήση.

Μια συγκεκριμένη εφαρμογή ενός REST Web Service ακολουθεί τέσσερις βασικές αρχές σχεδιασμού:

- Αποκλειστική χρήση της μεθόδου HTTP.
- Ανεξαρτησία κατάστασης.
- Δημοσίευση URI όμοια με δομή καταλόγου.
- Μεταφέρει XML, JavaScript Object Notation (JSON), ή και τα δύο.

Παρακάτω επεκτείνονται αυτές οι 4 αρχές και παρατίθεται ο λόγος για τον οποίο είναι σημαντικό για τους σχεδιαστές των REST διαδικτυακών υπηρεσιών.

## *2.1 Αποκλειστική χρήση της μεθόδου HTTP*

Ένα από τα βασικά χαρακτηριστικά των REST Web service είναι η ρητή χρήση των HTTP μεθόδων με έναν τρόπο που ακολουθεί το πρωτόκολλο όπως έχει οριστεί από το RFC 2616. Το HTTP GET, για παράδειγμα, είναι ορισμένο σαν μια μέθοδος παραγωγής δεδομένων που είναι προορισμένη να χρησιμοποιηθεί από την εφαρμογή πελάτη για να εξάγει κάποιο πόρο, να μεταφέρει δεδομένα από ένα εξυπηρετητή διαδικτύου, ή να εκτελέσει μια κλήση (query) αναμένοντας ότι ο εξυπηρετητής διαδικτύου θα ψάξει και θα απαντήσει με ένα σύνολο από τους πόρους που αντιστοιχούν.

Το REST ζητάει από τους προγραμματιστές να χρησιμοποιήσουν αποκλειστικά τις μεθόδους HTTP και έτσι υπάρχει συνέπεια με τον ορισμό του πρωτοκόλλου. Αυτή η βασική σχεδιαστική αρχή του REST καθιερώνει μια αντιστοιχία ένα προς ένα μεταξύ των λειτουργιών παραγωγής(create), διαβάσματος(read), ανανέωσης(update) και διαγραφής(delete) (CRUD) και HTTP μεθόδων. Σύμφωνα με αυτή την αντιστοίχηση:

- Για να δημιουργηθεί ένας πόρος στον εξυπηρετητή, χρησιμοποιούμε POST.
- Για να λάβουμε έναν πόρο, χρησιμοποιούμε GET.
- Για να αλλάξει η κατάσταση ενός πόρου ή να ανανεωθεί, χρησιμοποιούμε PUT.
- Για να αφαιρεθεί ή να διαγραφεί ένας πόρος, χρησιμοποιούμε DELETE.

Ένα ατυχές σχεδιαστικό μειονέκτημα έμφυτο σε πολλά Web APIs είναι η χρήση των HTTP μεθόδων για ανεπιθύμητους λόγους. Το URI αίτημα σε μια HTTP GET αίτηση, για παράδειγμα, συνήθως προσδιορίζει έναν συγκεκριμένο πόρο. Είτε το string της κλήσης σε ένα URI αίτημα συμπεριλαμβάνει ένα σύνολο από παραμέτρους που προσδιορίζουν τα κριτήρια αναζήτησης που χρησιμοποιούνται από τον εξυπηρετητή για να βρει ένα σύνολο από τους αντίστοιχους πόρους. Τουλάχιστον, με αυτόν τον τρόπο το HTTP/1.1 RFC περιγράφει το GET. Αλλά υπάρχουν πολλές περιπτώσεις μη ελκυστικών WEB APIs που χρησιμοποιούν HTTP GET για να ενεργοποιήσουν μια συναλλαγή με τον εξυπηρετητή για παράδειγμα για να κάνει εγγραφές σε κάποια βάση δεδομένων. Σε αυτές τις περιπτώσεις το GET URI αίτημα

δεν χρησιμοποιείται σωστά. Αν το Web API χρησιμοποιεί το GET για να προκαλέσει απομακρυσμένες διαδικασίες μοιάζει με αυτό:

GET /adduser?name=Robert HTTP/1.1

Αυτό δεν είναι σχεδιαστικά προσιτό γιατί η παραπάνω διαδικτυακή μέθοδος υποστηρίζει μια λειτουργία που αλλάζει κατάσταση στο HTTP GET. Θα μπορούσαμε να πούμε ότι, η παραπάνω αίτηση HTTP GET έχει παρενέργειες. Αν επιτύχει, το αποτέλεσμα θα είναι να προσθέσει έναν νέο χρήστη στα αποθηκευμένα δεδομένα. Το πρόβλημα εδώ είναι κυρίως σημασιολογικό. Οι διαδικτυακοί εξυπηρετητές έχουν σχεδιαστεί να απαντούν στα HTTP GET αιτήματα ανακτώντας πόρους που αντιστοιχούν στο μονοπάτι (ή τα κριτήρια της κλήσης) στο URI αίτημα και επιστρέφει αυτά ή την αναπαράσταση της απάντησης αλλά όχι για να προσθέσει μια εγγραφή σε μια βάση.

Εκτός από την σημασιολογία, το άλλο πρόβλημα με το GET είναι ότι για να ενεργοποιήσει διαγραφή, μετατροπή, ή προσθήκη μιας εγγραφής σε μια βάση, ή να αλλάξει μια server-side κατάσταση κατά κάποιο τρόπο, προσκαλεί εργαλεία caching (crawlers) και μηχανές αναζήτησης για να εκτελέσει μια server-side αλλαγή αθέμητα απλή . Ένας απλός τρόπος να ξεπεραστεί το πρόβλημα είναι να θέσουμε τα ονόματα και τις τιμές των παραμέτρων του URI αιτήματος σε XML ετικέτες. Οι προκύπτουσες ετικέτες, μια αναπαράσταση της οντότητας που πρέπει να δημιουργηθεί, θα μπορεί να σταλεί στο σώμα ενός HTTP POST του οποίου το URI αίτημα είναι ο προορισμένος γονέας της οντότητας.

ΠΡΙΝ:

GET /adduser?name=Robert HTTP/1.1

ΜΕΤΑ:

POST /users HTTP/1.1

Host: myserver

Content-Type: application/xml

<?xml version="1.0"?>

<user>

<name>Robert</name>

</user>

Η παραπάνω μέθοδος είναι υποδειγματική ενός RESTful αιτήματος: σωστή χρήση του HTTP POST και συμπερίληψη του payload στο σώμα του αιτήματος. Στο τέλος, το αίτημα μπορεί να εκτελεστεί προσθέτοντας τον συμπεριλαμβανόμενο πόρο στο σώμα ως δευτερεύον του προσδιορισμένου πόρου στο URI αίτημα. Σε αυτή την περίπτωση ο καινούργιος πόρος πρέπει να προστεθεί ως το παιδί του /users. Αυτή η περιοριστική σχέση ανάμεσα στην καινούργια οντότητα και τον πατέρα, όπως χαρακτηρίστηκε από το POST αίτημα, είναι ανάλογο με τον τρόπο που ένας φάκελος υπάγεται στον φάκελο γονέα. Ο πελάτης αρχικοποιεί την σχέση ανάμεσα στην οντότητα και στον γονέα και προσδιορίζει την καινούργια οντότητα URI στο POST αίτημα.

Μια εφαρμογή πελάτη ίσως τότε να πάρει μια αναπαράσταση του πόρου που χρησιμοποιεί το καινούργιο URI, σημειώνοντας ότι τουλάχιστον λογικά ο πόρος βρίσκεται κάτω από το /users, όπως φαίνεται παρακάτω.

GET /users/Robert HTTP/1.1

Host: myserver

Accept: application/xml

Χρησιμοποιώντας το GET με αυτόν τον τρόπο είναι κατηγορηματικός γιατί το GET είναι μόνο για την ανάκτηση δεδομένων. Το GET πρέπει να είναι ελεύθερο από παρενέργειες.

Μια παρόμοια αναδόμηση της διαδικτυακής μεθόδου επίσης χρειάζεται να εφαρμοστεί σε περιπτώσεις που μια λειτουργία ανανέωσης υποστηρίζεται πάνω στο HTTP GET, όπως φαίνεται παρακάτω:

GET /updateuser?name=Robert&newname=Bob HTTP/1.1

Αυτό αλλάζει το όνομα των ιδιοτήτων του πόρου. Ενώ το string του αιτήματος μπορεί να χρησιμοποιηθεί για μια τέτοια λειτουργία, και το παράδειγμα 4 είναι απλό, αυτό το σχέδιο query-string-as-method-signature τείνει να σπάσει όταν χρησιμοποιείται για πιο σύνθετες λειτουργίες. Επειδή ο στόχος είναι η αποκλειστική χρήση των HTTP μεθόδων, μια προσέγγιση πιο κοντά στο REST, στέλνει ένα αίτημα HTTP PUT για να ανανεώσει τους πόρους, στην θέση του HTTP GET, για τους ίδιους λόγους που δηλώθηκαν πιο πάνω.

PUT /users/Robert HTTP/1.1

Host: myserver

Content-Type: application/xml

<?xml version="1.0"?>

<user>

<name>Bob</name>

</user>

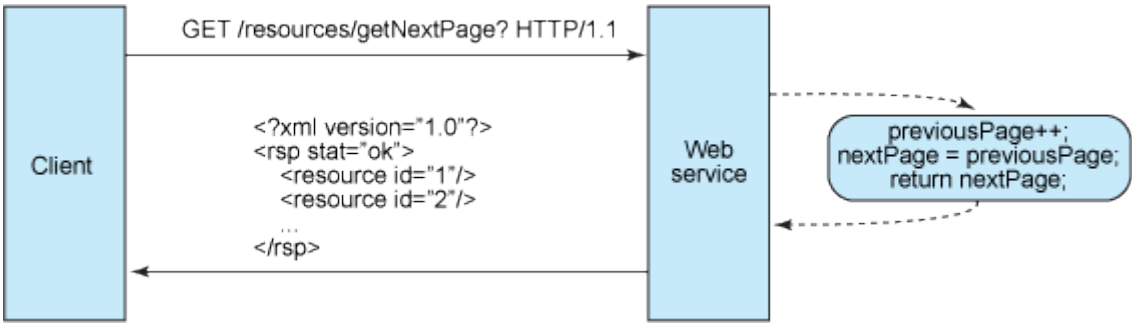
Χρησιμοποιώντας το PUT για την αντικατάσταση των αρχικών πόρων, παρέχει μια πιο καθαρή διεπαφή που είναι συνεπής με την αρχές του REST και με τους ορισμούς των μεθόδων HTTP. Το αίτημα PUT στο Παράδειγμα 5 είναι αποκλειστικό με την έννοια ότι δείχνει στους πόρους που πρέπει να ανανεωθούν προσδιορίζοντας το στο URI αίτημα και με την έννοια ότι μεταφέρει μια καινούργια αναπαράσταση των πόρων από τον πελάτη στον εξυπηρετητή στο σώμα του αιτήματος PUT αντί να μεταφέρει τις ιδιότητες ως ένα λυτό σύνολο από ονόματα και τιμές παραμέτρων στο URI αίτημα. Στο παράδειγμα 5 βλέπουμε επίσης ως αποτέλεσμα την μετονομασία του πόρου από Robert σε Bob, και πράττοντας αυτό επιβαρύνει το URI του σε /users/Bob. Σε ένα REST Web service, μεταγενέστερα αιτήματα του πόρου που χρησιμοποιούν το παλιό URI θα προκαλέσουν 404 Not Found σφάλμα.

Ως γενική σχεδιαστική αρχή, εξυπηρετεί να ακολουθούνται οι κατευθυντήριες γραμμές του REST για να χρησιμοποιούμε τις μεθόδους HTTP αποκλειστικά χρησιμοποιώντας ουσιαστικά στα URI αντί για ρήματα. Σε ένα RESTful Web service, τα ρήματα –POST, GET, PUT. και DELETE- είναι ήδη καθορισμένα από το πρωτόκολλο. Επίσης ιδανικά, για να διατηρούμε την διεπαφή γενικευμένη και να επιτρέπεται στους πελάτες να είναι σαφείς για τις λειτουργίες που επικαλούνται, η διαδικτυακή υπηρεσία πρέπει να προσδιορίσει περισσότερα ρήματα ή απομακρυσμένες διαδικασίες, όπως /adduser ή /updateuser. Αυτή η γενική σχεδιαστική αρχή επίσης εφαρμόζεται στο σώμα του HTTP αιτήματος, που σκοπεύει να χρησιμοποιηθεί για να μεταφέρει την κατάσταση των πόρων , όχι για να φέρει το όνομα την απομακρυσμένης μεθόδου ή της απομακρυσμένης διαδικασίας που θα επικαλεσθεί.

## *2.2 Ανεξαρτησία κατάστασης*

Τα REST Web services χρειάζεται να κλιμακωθούν για να φτάσουν σε υψηλής απόδοσης απαιτήσεις. Συγκροτήματα από εξυπηρετητές με εξισορρόπηση φορτίου και δυνατότητες ανακατεύθυνσης, proxy, και gateways είναι τυπικά κατανεμημένα με έναν τρόπο που σχηματίζουν μια τοπολογία υπηρεσιών , που επιτρέπει τα αιτήματα να προωθηθούν από τον έναν εξυπηρετητή στον άλλο καθώς χρειάζεται να μειωθεί ο συνολικός χρόνος απάντησης μιας κλήσης μιας διαδικτυακής υπηρεσίας. Χρησιμοποιώντας ενδιάμεσους εξυπηρετητές για να βελτιωθεί η κλίμακα, χρειάζεται οι πελάτες των διαδιακτυακών υπηρεσίων REST να στείλουν καθολικά, ανεξάρτητα αιτήματα. Αυτό σημαίνει ότι πρέπει να στείλει αιτήματα που περιέχουν όλα τα δεδομένα που χρειάζονται για συμπληρωθούν έτσι ώστε όλα τα επιμέρους στοιχεία στους ενδιάμεσους εξυπηρετητές να μπορούν να προωθηθούν, να δρομολογηθούν και ισορροπήσουν το φορτίο χωρίς καμία κατάσταση να κρατηθεί τοπικά ανάμεσα στα αιτήματα.

Ένα ολοκληρωμένο, ανεξάρτητο αίτημα δεν απαιτεί εξυπηρετητή, καθώς επεξεργάζεται το αίτημα για να ανακτήσει οποιοδήποτε περιεχόμενο ή κατάσταση. Μια εφαρμογή (ή πελάτης) διαδικτυακής υπηρεσίας REST περιέχει μέσα στους HTTP headers και στο σώμα(body) ενός αιτήματος όλες τις παραμέτρους, το περιεχόμενο, και τα απαιτούμενα δεδομένα από το server-side στοιχείο που θα εξάγει την απάντηση. Η ανεξαρτησία της κατάστασης υπό αυτή την έννοια βελτιώνει την απόδοση της διαδικτυακής υπηρεσίας και απλοποιεί τον σχεδιασμό και την εφαρμογή του server-side στοιχείου γιατί η έλλειψη κατάστασης στον εξυπηρετητή εξαλείφει την ανάγκη να συγχρονιστούν τα δεδομένα του session με εξωτερικές εφαρμογές.



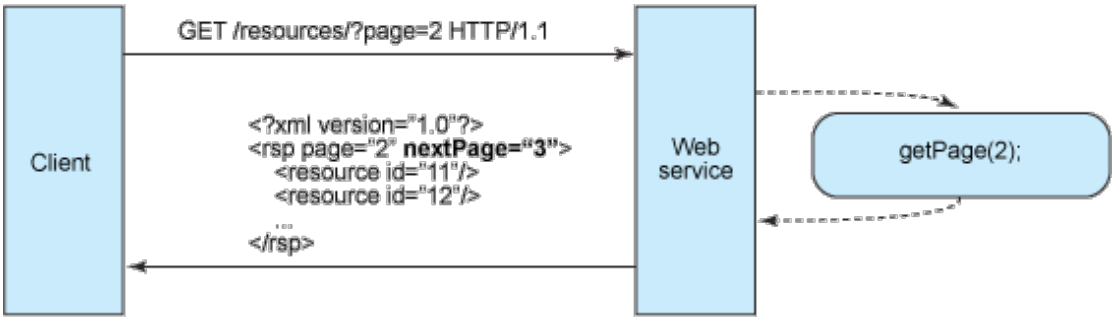

**Εικόνα 3: Σχέδιο εξαρτημένης κατάστασης**

Η Εικόνα 4 απεικονίζει μια υπηρεσία εξαρτημένης κατάστασης στην οποίο η εφαρμογή μπορεί να αιτήσει την επόμενη σελίδα σε ένα συνολικό αποτέλεσμα πολλαπλών σελίδων, θεωρώντας ότι η υπηρεσία παρακολουθεί πού η εφαρμογή σταματάει κατά την πλοήγηση στο σύνολο. Σε αυτό τον σχεδιασμό, η υπηρεσία αυξάνει και αποθηκεύει μια PreviousPage (προηγούμενη σελίδα) παράμετρο κάπου ώστε να μπορεί να απαντήσει σε κάποιο αίτημα.

Υπηρεσίες εξαρτημένης κατάστασης σαν αυτές γίνονται πολύπλοκες. Σε ένα περιβάλλον Java Platform, Enterprise Edition (Java EE) οι υπηρεσίες ανεξάρτητης κατάστασης απαιτούν πολλή εκ των προτέρων εξέταση για την αποτελεσματική αποθήκευση και επιτρέπει τον συγχρονισμό των session data σε ένα cluster από Java EE containers. Σε αυτό το είδος περιβάλλοντος, υπάρχει ένα πρόβλημα γνωστό στα servlet/JavaServer Pages (JSP) και Enterprise JavaBeans(EJB) προγραμματιστές που συχνά δυσκολεύονται να βρουν βασικές αιτίες για java.io.NotSerializableException κατά την διάρκεια αντιγραφής των session. Είτε προκύπτει από ένα servlet container κατά την διάρκεια μιας HttpSession αντιγραφής είτε προκύπτει από ένα EJB container κατά την διάρκεια μιας εξαρτημένης κατάστασης EJB αντιγραφής, είναι ένα πρόβλημα που μπορεί να κοστίσει στους προγραμματιστές πολλές μέρες να εντοπίσουν το αντικείμενο που δεν εφαρμόζει το Serializable κάποιες φορές σε ένα πολύ σύνθετο γράφο από αντικείμενα που αποτελούν την κατάσταση του εξυπηρετητή. Επιπλέον, ο συγχρονισμός των sesssion προσθέτει ένα overhead, που επηρεάζει την απόδοση του εξυπηρετητή.

Από την άλλη πλευρά,τα server-side στοιχεία ανεξάρτητης κατάστασης είναι λιγότερο πολύπλοκα στον σχεδιασμό, στο γράψιμο, και συνθέτουν εξυπηρετητές ισορροπημένου φορτίου. Μια υπηρεσία ανεξάρτητης κατάστασης όχι μόνο αποδίδει καλύτερα, αλλά μεταφέρει και την ευθύνη της συντήρησης της κατάστασης στην

εφαρμογή-πελάτη. Σε μια Restful διαδικτυακή υπηρεσία, ο εξυπηρετητής είναι υπεύθυνος για να παράγει απαντήσεις και να παρέχει μια διεπαφή που να παρέχει την δυνατότητα στον πελάτη να διατηρήσει την κατάσταση της εφαρμογής μόνος του. Για παράδειγμα, στην αίτηση για ένα πολυσέλιδο σύνολο αποτελεσμάτων, ο πελάτης πρέπει να συμπεριλάβει τον ακριβή αριθμό σελίδας που επιθυμεί να λάβει αντί να ζητάει απλά την επόμενη.




**Εικόνα 4: Σχέδιο ανεξάρτητης κατάστασης**

Μια διαδικτυακή υπηρεσία ανεξάρτητης κατάστασης παράγει μια απάντηση που συνδέει την επόμενη σελίδα στο σύνολο και αφήνει τον πελάτη να κάνει αυτό που χρειάζεται για να κρατήσει αυτή την τιμή. Αυτή η πλευρά του σχεδιασμού των RESTful διαδικτυακών υπηρεσιών μπορεί να διαχωριστεί σε δύο σύνολα ευθυνών ως ενός υψηλού επιπέδου διαχωρισμός που ξεκαθαρίζει ακριβώς πώς μια υπηρεσία ανεξάρτητης κατάστασης μπορεί να διατηρηθεί:

**Εξυπηρετητής**

- Παράγει απαντήσεις που περιλαμβάνουν συνδέσμους σε άλλους πόρους για να επιτρέψουν στις εφαρμογές να πλοηγηθούν ανάμεσα στους σχετικούς πόρους. Αυτός ο τύπος απάντησης ενσωματώνει συνδέσμους. Παρόμοια, αν η αίτηση είναι για έναν γονέα ή για έναν πόρο container, τότε μια τυπική RESTful απάντηση μπορεί επίσης να συμπεριλαμβάνει συνδέσμους στα παιδιά των γονέων ή σε δευτερεύοντες πόρους έτσι ώστε να παραμείνουν συνδεδεμένοι.
- Παράγει απαντήσεις που υποδεικνύουν αν μπορούν να βελτιώσουν την απόδοσή τους με το να μειώσουν τον αριθμό αιτήσεων για διπλούς πόρους και να απαλείψουν κάποιες αιτήσεις τελείως. Ο εξυπηρετητής το κάνει αυτό με το να συμπεριλάβει ένα Cache-Control και ένα Last-Modified(μια τιμή ημερομηνίας) HTTP response header.

**Εφαρμογή πελάτη**

- Χρησιμοποιεί το Cache-Control response header για να προσδιορίσει αν μπορεί να κάνει cache ( να κάνει ένα τοπικό αντίγραφο αυτού) ή όχι. Ο πελάτης επίσης διαβάζει το Last-Modified response header και επιστρέφει την τιμή της ημερομηνίας σε ένα If-Modified-Since header για να ζητήσει από τον εξυπηρετητή αν ο πόρος έχει αλλάξει. Αυτό ονομάζεται Conditional GET, και το ένα header συνοδεύει το άλλο στο οποίο η απάντηση του εξυπηρετητή είναι ένα καθιερωμένο 304 code (Not Modified) και στέλνει το πραγματικό ζητούμενο πόρο αν δεν έχει σταλεί μέχρι εκείνη τη στιγμή. Ένα 304 HTTP response code σημαίνει ότι ο πελάτης μπορεί με ασφάλεια να χρησιμοποιήσει ένα cached, τοπικό αντίγραφο της απεικόνισης του πόρου ως το πιο σύγχρονο, με αποτέλεσμα να προσπερνάει επόμενες GET αιτήσεις μέχρι ο πόρος να αλλάξει.
- Στέλνει ολοκληρωμένες αιτήσεις που μπορούν να εξυπηρετηθούν ανεξάρτητα από άλλες αιτήσεις. Αυτό απαιτεί ο πελάτης να κάνει αποκλειστική χρήση των HTTP header όπως καθορίστηκαν από την διεπαφή της υπηρεσίας Web και να στείλει ολοκληρωμένες απεικονίσεις των πόρων στο σώμα της αίτησης. Ο πελάτης στέλνει αιτήσεις που κάνουν πολύ λίγες υποθέσεις για προγενέστερες αιτήσεις, την ύπαρξη ενός session στον εξυπηρετητή, την δυνατότητα του εξυπηρετητή να προσθέσει περιεχόμενο σε μια αίτηση, ή για την κατάσταση της εφαρμογής που διατηρείται ανάμεσα σε δύο αιτήσεις.

Αυτή η συνεργασία ανάμεσα στην εφαρμογή πελάτη και την υπηρεσία είναι ουσιώδης για να είναι ανεξάρτητο κατάστασης με μια RESTful διαδικτυακή υπηρεσία. Βελτιώνει την απόδοση αποθηκεύοντας το εύρος ζώνης και ελαχιστοποιώντας την κατάσταση της server-side εφαρμογής.

## *2.3 Δημοσίευση URI όμοια με δομή καταλόγου.*

Από την σκοπιά των πόρων που απευθύνονται στις εφαρμογές πελάτη, τα URI προσδιορίζουν πόσο διαισθητική θα είναι η RESTful διαδικτυακή υπηρεσία και αν η υπηρεσία θα χρησιμοποιηθεί με τρόπους που οι σχεδιαστές μπορούν να περιμένουν. Ένα τρίτο χαρακτηριστικό των RESTful διαδικτυακών υπηρεσιών είναι ότι όλα σχετίζονται με URI.

Τα URI των RESTful διαδικτυακών υπηρεσιών πρέπει να είναι διαισθητικά στο σημείο όπου είναι εύκολο να τα μαντέψεις. Πρέπει να σκεφτούμε τα URI ως ενός είδους αυτοπροσδιοριζόμενης διαπροσωπείας που απαιτεί μικρή, ή καμία επεξήγηση ή αναφορά για τον προγραμματιστή για να καταλάβει σε τι δείχνει και να αντλήσει σχετικούς πόρους. Η δομή ενός URI πρέπει να είναι απλή,προβλεπόμενη και εύκολα κατανοητή.

Ένας τρόπος για να επιτευχθεί αυτό το επίπεδο χρησιμότητας είναι να καθοριστούν URI με δομή καταλόγου. Αυτός ο τύπος URI είναι ιεραρχικός, προέρχεται από ένα συγκεκριμένο μονοπάτι, και η διακλάδωση από αυτό είναι υπόμονοπάτια που δημοσιεύουν τις βασικές περιοχές της υπηρεσίας. Σύμφωνα με αυτόν τον ορισμό,ένα URI δεν είναι απλώς ένα string οριοθετημένο από καθέτους, αλλά ένα δέντρο με δευτερεύουσες και πρωτεύουσες διακλαδώσεις συνδεδεμένες σε κόμβους. Για παράδειγμα, σε μια υπηρεσία νηματοειδούς συζήτησης που περιλαμβάνει θέματα από την Java μέχρι τα άρθρα, μπορεί να προσδιοριστεί ένα δομημένο σύνολο από URI όπως αυτό:

http://www.myservice.org/discussion/topics/{topic}

Η ρίζα, /discussion, έχει ένα /topics κόμβο κάτω από αυτό. Κάτω από αυτό υπάρχουν σειρές από ονόματα θεμάτων, όπως τεχνολογία, νέα κτλ.. το καθένα από τα οποία δείχνουν σε ένα νήμα συζήτησης. Μέσα σε αυτή τη δομή είναι εύκολο να λάβεις νήματα συζητήσεων απλά πληκτρολογώντας κάτι μετά το /topics/.

Σε μερικές περιπτώσεις, το μονοπάτι σε έναν πόρο προσφέρεται καλά σε μια δομή όμοια με αυτή ενός καταλόγου. Για παράδειγμα οι πόροι που είναι οργανωμένοι σύμφωνα με την ημερομηνία ταιριάζουν εξαιρετικά για να χρησιμοποιηθούν σε μια ιεραρχική σύνταξη.

Αυτό το παράδειγμα είναι διαισθητικό επειδή βασίζεται σε κανόνες:

http://www.myservice.org/discussion/2008/12/10/{topic}

Το πρώτο κομμάτι του μονοπατιού είναι ένας τετραψήφιος αριθμός που συμβολίζει τον χρόνο, το δεύτερο κομμάτι είναι ένας διψήφιος αριθμός που συμβολίζει την ημέρα και το τρίτο κομμάτι, ένας διψήφιος αριθμός που συμβολίζει τον μήνα. Οι άνθρωποι και οι μηχανές μπορούν εύκολα να παράγουν δομημένα URIs όπως αυτά γιατί είναι δομημένα σε κανόνες. Συμπληρώνοντας στο μονοπάτι τμήματα τους κάνει καλό γιατί ακολουθούν ένα συγκεκριμένο σχέδιο από το οποίο μπορούν να συντεθούν:

http://www.myservice.org/discussion/{year}/{day}/{month}/{topic}

Μερικές επιπλέον λεπτομέρειες που πρέπει να προσέχουμε όταν σκεφτόμαστε για την δομή ενός URI για μια RESTful διαδικτυακή υπηρεσία είναι:

- Να κρύψουμε τις επεκτάσεις των αρχείων της server-side scripting τεχνολογίας (.jsp, .php,.asp) αν υπάρχει, έτσι ώστε να μπορούμε να εισάγουμε κάτι άλλο χωρίς να χρειάζεται να αλλάξουμε τα URI.
- Να κρατάμε τα πάντα σε μικρά γράμματα.
- Να αντικαθιστούμε τα κενά με κάποιο σύνδεσμο ή με κάτω παύλα.
- Να αποφεύγουμε string αιτημάτων όσο το δυνατόν περισσότερο.
- Αντί να χρησιμοποιούμε το 404 Not Found κώδικα αν το αιτούμενο URI είναι για ένα μέρος του μονοπατιού, πρέπει να παρέχουμε πάντα μια σελίδα default ή έναν πόρο σαν απάντηση.

Τα URIs πρέπει να είναι στατικά έτσι ώστε όταν ο πόρος αλλάζει ή η εφαρμογή μιας υπηρεσίας αλλάζει, ο σύνδεσμος πρέπει να μένει ίδιος. Αυτό επιτρέπει την σελιδοποίηση. Είναι επίσης σημαντικό ότι η σχέση μεταξύ των πόρων που είναι κωδικοποιημένοι στο URI να παραμένει ανεξάρτητη από τον τρόπο που οι σχέσεις αναπαριστώνται εκεί που έχουν αποθηκευτεί.

## *2.4 Μεταφέρουν XML, JSON, ή και τα δύο*

Μια αναπαράσταση του πόρου τυπικά αντικατοπτρίζει την τρέχουσα κατάσταση ενός πόρου, και τα γνωρίσματά του τη στιγμή που μια εφαρμογή πελάτη στέλνει κάποιο αίτημα. Οι απεικονίσεις των πόρων λοιπόν είναι περισσότερο σαν ένα στιγμιότυπο μέσα στον χρόνο. Αυτό μπορεί να είναι κάτι τόσο απλό όσο μια απεικόνιση μιας εγγραφής σε μια βάση δεδομένων που αποτελείται από μια αντιστοίχιση ανάμεσα σε στήλες ονομάτων και XML tags, όπου τα αντικείμενα τιμές στο XML περιέχουν μια γραμμή τιμών. Ή αν το σύστημα έχει ένα μοντέλο δεδομένων, τότε σύμφωνα με τον ορισμό μια απεικόνιση πόρου είναι ένα στιγμιότυπο των γνωρισμάτων ενός από των πραγμάτων στο μοντέλο δεδομένων του συστήματος. Αυτά είναι τα αντικείμενα που θα θέλαμε μια REST διαδικτυακή υπηρεσία να εξυπηρετεί.

Το τελευταίο σύνολο περιορισμών που απευθύνεται στον σχεδιασμό μιας RESTful διαδικτυακής υπηρεσίας έχει να κάνει με τη μορφή των δεδομένων που η εφαρμογή και η υπηρεσία ανταλλάσσει στην αίτηση/απάντηση payload ή στο HTTP-body. Από εκεί φαίνεται ότι πρέπει να κρατάμε τα πράγματα απλά, να είναι ευανάγνωστα και να είναι συνδεδεμένα.

Τα αντικείμενα στο μοντέλο δεδομένων είναι συνήθως συσχετισμένα με κάποιο τρόπο, και οι σχέσεις μεταξύ των αντικειμένων του μοντέλου δεδομένων (οι πόροι) πρέπει να αντανακλούν τον τρόπο που έχουν αναπαρασταθεί για μεταφορά σε μια εφαρμογή-πελάτη. Στην υπηρεσία νηματοειδούς συζήτησης, έναν παράδειγμα απεικόνισης συνδεδεμένου πόρου μπορεί να περιλαμβάνει θέματα από την αρχική συζήτηση και τα γνωρίσματα της, και να εμπεριέχει συνδέσμους σε απαντήσεις που έχουν δοθεί σε αυτό το θέμα.

XML αναπαράσταση ενός νήματος συζήτησης:

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
    <comment>{comment}</comment>
    <replies>
        <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe"/>
        <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
    </replies>
```

</discussion>

Για να δώσουμε στις εφαρμογές πελάτη την δυνατότητα να ζητήσουν ένα συγκεκριμένο τύπο περιεχομένου που ταιριάζει περισσότερο σε αυτές, πρέπει να δομηθεί μια υπηρεσία που χρησιμοποιεί built-in HTTP Accept header , όπου η τιμή του header είναι τύπου MIME. Μερικά συνηθισμένα είδη MIME που χρησιμοποιούνται στις RESTful υπηρεσίες φαίνονται παρακάτω.

| **Τύπος MIME** | **Τύπος περιεχομένου** |
|---|---|
| **JSON** | application/json |
| **XML** | application/xml |
| **XHTML** | application/xhtml+xml |

**Πίνακας 1: Συνήθεις τύποι MIME που χρησιμοποιούνται στις RESTful υπηρεσίες**

Αυτό επιτρέπει στην υπηρεσία να χρησιμοποιηθεί από μια ποικιλία από πελάτες γραμμένη σε διαφορετικές γλώσσες που τρέχουν σε διαφορετικές πλατφόρμες και συσκευές. Χρησιμοποιώντας τύπους MIME και HTTP Accept header είναι ένας μηχανισμός γνωστός ως *content negotiation,* που επιτρέπει στους πελάτες να διαλέξουν ποια μορφή είναι σωστή γι' αυτούς και να ελαχιστοποιήσουν τη ζεύξη δεδομένων ανάμεσα στην υπηρεσία και την εφαρμογή που την χρησιμοποιεί.

## *2.4.1 JSON*

JSON (JavaScript Object Notation) είναι ένα format ανοιχτού προτύπου που χρησιμοποιεί κείμενο αναγνωρίσιμο από τον άνθρωπο για να μεταδώσει αντικείμενα δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικών-τιμών . Χρησιμοποιείται κυρίως για τη μετάδοση δεδομένων μεταξύ εξυπηρετητή και διαδικτυακής εφαρμογής, ως εναλλακτική λύση για την XML .

Παρά το γεγονός ότι προέρχεται αρχικά από την JavaScript scripting γλώσσα, η JSON είναι μια γλώσσα ανεξάρτητη μορφής δεδομένων. Κώδικας για την ανάλυση και την παραγωγή δεδομένων JSON είναι εύκολα διαθέσιμος σε πολλές γλώσσες προγραμματισμού.

Η μορφή JSON αρχικά είχε προταθεί από τον Daglas Crockford . Αυτή τη στιγμή περιγράφεται από δύο ανταγωνιστικά πρότυπα, RFC 7159 και ECMA-404. Το πρότυπο ECMA είναι ελάχιστο, περιγράφοντας μόνο την επιτρεπτή γραμματική σύνταξη, ενώ η RFC παρέχει επίσης ορισμένες σημασιολογικές και εκτιμήσεις ασφαλείας. Το Internet Media Type για JSON είναι application/json . Η επέκταση ονόματος αρχείου JSON είναι .json .

**Βασικοί τύποι JSON είναι:**

- Number – ένας δεκαδικός αριθμός με πρόσημο που μπορεί να περιέχει ένα κλασματικό μέρος και μπορεί να χρησιμοποιήσει εκθετική Ε σημειογραφία. Η JSON δεν επιτρέπει μη-αριθμούς, όπως NaN , ούτε κάνει καμία διάκριση μεταξύ ακεραίου και floating-point. (Ακόμα κι αν η JavaScript χρησιμοποιεί μια Μορφή διπλής ακρίβειας κινητής υποδιαστολής για όλες τις αριθμητικές τιμές, άλλες γλώσσες της εφαρμογής του JSON μπορούν να κωδικοποιούν αριθμούς διαφορετικά).

- String - μια ακολουθία από μηδέν ή περισσότερους Unicode χαρακτήρες, αν και οι χαρακτήρες έξω από το BMP πρέπει να εκπροσωπούνται ως υποκατάστατο ζευγάρι . Τα String οριοθετούνται με διπλά εισαγωγικά και να υποστηρίζουν μια ανάστροφη κάθετο σύνταξη διαφυγής.

- Boolean - οι τιμές true ή false.

- Array - μια ταξινομημένη λίστα από μηδέν ή περισσότερες τιμές, καθεμία από τις οποίες μπορεί να είναι οποιουδήποτε τύπου. Οι πίνακες χρησιμοποιούν σημειογραφία αγκυλών και τα στοιχεία είναι διαχωρισμένα με κόμματα.
- Αντικείμενο -ένα μη διατεταγμένο συσχετισένο array (ζευγάρια ονομάτων/τιμής). Τα αντικείμενα οριοθετούνται με αγκύλες και χρησιμοποιούν κόμματα για να διαχωρίσουν κάθε ζευγάρι, ενώ μέσα σε κάθε ζεύγος ο »:« χαρακτήρας διαχωρίζει το κλειδί ή το όνομα από την τιμή του. Όλα τα κλειδιά πρέπει να είναι strings και πρέπει να διακρίνονται από κάποιο άλλο μέσα στο αντικείμενο.
- null - μια κενή τιμή, χρησιμοποιώντας τη λέξη null

Η JSON αγνοεί γενικά οποιοδήποτε κενό γύρω ή μεταξύ συντακτικών αντικειμένων (τιμές και σημεία στίξης, αλλά όχι μέσα σε ένα string). Ωστόσο η JSON αναγνωρίζει μόνο τέσσερις συγκεκριμένους χαρακτήρες κενού: το whitespace, οριζόντιο tab, line feed, και carriage return. Η JSON δεν παρέχει ή να επιτρέπει κανενός είδους συντακτικού σχολίου.

Οι παλαιότερες εκδόσεις της JSON (όπως καθορίζονται από το RFC 4627 ) απαιτούσαν ότι ένα έγκυρο "έγγραφο" JSON πρέπει να αποτελείται μόνο από ένα αντικείμενο ή ένα τύπο array παρ 'όλο που θα μπορούσαν να περιέχουν άλλους τύπους στο εσωτερικό τους. Ο περιορισμός αυτός έγινε πιο χαλαρός αρχίζοντας από το RFC 7158 , έτσι ώστε ένα έγγραφο JSON μπορεί να αποτελείται εξ ολοκλήρου από κάθε πιθανή JSON τιμή.

Το ακόλουθο παράδειγμα δείχνει μια δυνατή απεικόνιση JSON που περιγράφει ένα άτομο.

```
{
  "Name": "John",
  "Surname": "Smith",
  "IsAlive": true,
  «Age»: 25,
  "Height_cm": 167,6,
  "Address": {
    "StreetAddress": "21 2nd Street»,
```

```
    "City": "New York",
    "Position: "New York",
    "PostalCode": "10021-3100"
  }
  "Telephone ": [
    {
      "Type": "home",
      "Number": "212 555-1234"
    }
    {
      "Type": "office",
      "Number": "646 555-4567"
    }
  ],
  "Children": [],
  "Partner": null
}
```

### *2.4.2 XML*

Η **XML** ( **Extensible Markup Language**) είναι μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων.

Ορίζεται, κυρίως, στην προδιαγραφή XML 1.0 (XML 1.0 Specification), που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων.

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο Διαδίκτυο. Είναι μία μορφοποίηση δεδομένων κειμένου, με ισχυρή υποστήριξη Unicode για όλες τις γλώσσες του κόσμου. Αν και η σχεδίαση της XML εστιάζει στα κείμενα, χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων, που προκύπτουν για παράδειγμα στις υπηρεσίες ιστού.

Υπάρχει μία ποικιλία διεπαφών προγραμματισμού εφαρμογών, που μπορούν να

χρησιμοποιούν οι προγραμματιστές, για να προσπελαύνουν δεδομένα XML, αλλά και διάφορα συστήματα σχημάτων XML, τα οποία είναι σχεδιασμένα για να βοηθούν στον ορισμό γλωσσών, που προκύπτουν από την XML.

Βασικά χαρακτηριστικά:

- Χαρακτήρας Unicode

  Εξ ορισμού, ένα κείμενο XML είναι μία ακολουθία χαρακτήρων. Σχεδόν κάθε χαρακτήρας Unicode μπορεί να εμφανίζεται σε ένα κείμενο XML.

- Επεξεργαστής και Εφαρμογή

  Είναι το λογισμικό που επεξεργάζεται ένα κείμενο XML. Είναι αναμενόμενο, ότι ένας επεξεργαστής δουλεύει για μία εφαρμογή. Υπάρχουν μερικές πολύ συγκεκριμένες απαιτήσεις, σχετικά με το τι μπορεί και τι δεν μπορεί να κάνει ένας επεξεργαστής XML, αλλά καμία, όσον αφορά στη συμπεριφορά της εφαρμογής. Ο επεξεργαστής (όπως ονοματίζεται από την προδιαγραφή), αναφέρεται συχνά, με τον αγγλικό όρο *XML parser*.

- Σήμανση και Περιεχόμενο

  Οι χαρακτήρες που απαρτίζουν ένα κείμενο XML, αποτελούν είτε τη *σήμανση* είτε το *περιεχόμενό* του. Η σήμανση και το περιεχόμενο, μπορούν να επισημανθούν και να διακριθούν, ύστερα από την εφαρμογή κάποιων απλών συντακτικών κανόνων. Όλα τα αλφαριθμητικά που συνιστούν τη σήμανση, είτε ξεκινούν με το χαρακτήρα "<" και καταλήγουν στο χαρακτήρα ">", είτε ξεκινούν με το χαρακτήρα "&" και καταλήγουν στο χαρακτήρα ";". Ακολουθίες χαρακτήρων που δε συνιστούν τη σήμανση, αποτελούν το περιεχόμενο ενός κειμένου XML.

- Ετικέτα

  Ένα στοιχείο σήμανσης που ξεκινά με το χαρακτήρα "<" και καταλήγει στο χαρακτήρα ">". Υπάρχουν τρία είδη ετικέτας:*ετικέτες-αρχής*,για παράδειγμα <section>, *ετικέτες-τέλους*, για παράδειγμα </section>, και *ετικέτες-χωρίς-περιεχόμενο*, για παράδειγμα <line-break/>.

- Στοιχείο

  Ένα λογικό απόσπασμα ενός κειμένου, που είτε ξεκινά με μία ετικέτα-αρχής και καταλήγει σε μία ετικέτα-τέλους, είτε αποτελείται μόνο από μία ετικέτα-χωρίς-περιεχόμενο. Οι χαρακτήρες που υπάρχουν, αν υπάρχουν, μεταξύ μιας ετικέτας-αρχής και μιας ετικέτας-τέλους, συνιστούν το *περιεχόμενο* του

στοιχείου, το οποίο μπορεί να περιέχει σήμανση, συμπεριλαμβανομένων και άλλων στοιχείων, που ονομάζονται *στοιχεία-παιδιά*. Ένα παράδειγμα ενός στοιχείου είναι το <Greeting>Hello, world.</Greeting>. Ένα άλλο είναι το <line-break/>.

- Χαρακτηριστικό

  Ένα στοιχείο σήμανσης που αποτελείται από ένα ζευγάρι *όνομα/τιμή*, το οποίο υπάρχει μέσα σε μία ετικέτα-αρχής ή σε μία ετικέτα-χωρίς-περιεχόμενο. Στο παράδειγμα παρακάτω, το στοιχείο *img* έχει δύο χαρακτηριστικά, τα *src* και *alt*: <img src="madonna.jpg" alt='by Raphael'/>. Ένα άλλο παράδειγμα θα ήταν το <step number="3">Connect A to B.</step>, όπου το όνομα του χαρακτηριστικού είναι "number" και η τιμή του είναι "3".

- Δήλωση XML

  Τα κείμενα XML μπορούν να αρχίζουν, με τη δήλωση κάποιων πληροφοριών σχετικών με αυτά, όπως στο ακόλουθο παράδειγμα:

  <?xml version="1.0" encoding="UTF-8"?>

***Παράδειγμα***

Το παρακάτω είναι ένα μικρό, αλλά πλήρες κείμενο XML, που κάνει χρήση όλων των παραπάνω εννοιών και στοιχείων.

```
<?xml version="1.0" encoding='UTF-8'?>
<painting>
        <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
        <caption>This is Raphael's "Foligno" Madonna, painted in
        <date>1511</date>-<date>1512</date>.</caption>
</painting>
```

Υπάρχουν πέντε στοιχεία σε αυτό το κείμενο του παραδείγματος: τα painting, img, caption, και δύο date. Τα στοιχεία date, είναι παιδιά του στοιχείου caption, το οποίο είναι παιδί του στοιχείου-ρίζας painting. Το στοιχείο img έχει δύο χαρακτηριστικά, τα src και alt.

### *2.4.3 Extensible Hypertext Markup Language (XHTML)*

Η **XHTML (Extensible Hypertext Markup Language )** είναι μια οικογένεια XML γλωσσών σήμανσης που αντανακλούν ή επεκτείνουν τις εκδόσεις του ευρέως χρησιμοποιούμενου HTML (Hypertext Markup Language), η γλώσσα στην οποία οι ιστοσελίδες έχουν διατυπωθεί.

Ενώ η HTML, πριν από την HTML5 , ορίστηκε ως μια εφαρμογή της Standard Generalized Markup Language (SGML), ένα ευέλικτο πλαίσιο γλώσσας σήμανσης, η XHTML είναι μια εφαρμογή της XML , ένα πιο περιοριστικό υποσύνολο της SGML. Τα XHTML κείμενα είναι καλοσχηματισμένα και μπορούν, επομένως, να αναλυθούν χρησιμοποιώντας πρότυπο ανάλυσης XML, σε αντίθεση με την HTML, η οποία απαιτεί ένα πιο επιεικές, ειδικό πρότυπο ανάλυσης HTML.

XHTML 1.0 έγινε μια Σύσταση World Wide Web Consortium (W3C) στις 26 Ιανουαρίου, 2000.Η XHTML 1.1 έγινε μια Σύσταση του W3C στις 31 Μαΐου, 2001. Το πρότυπο που είναι γνωστό ως XHTML5 αναπτύσσεται ως μια προσαρμογή της XML της προδιαγραφής HTML5.

# 3 Βασικά εργαλεία του WebHookIt

## 3.1 NodeJS

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον Javascript. Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων μία διεργασία *node* δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου.

### 3.1.1 Ιστορία.

Το Node.js δημιουργήθηκε από τον Ryan Dahl το 2009. Η δημιουργία και η συντήρηση του έργου χορηγήθηκε από την εταιρία Joyent. Η ιδέα για την ανάπτυξη του *node* προήλθε από την ανάγκη του Ryan Dahl να βρεί τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Επίσης επηρεάστηκε από το Mongrel του Zed Shaw. Επιπροσθέτως μετά από αποτυχημένα έργα σε C, Lua, Haskell η κυκλοφορία της μηχανής V8 (V8 JavaScript Engine) της Google τον ώθησε να ασχοληθεί με την Javascript.

### *3.1.2 Χαρακτηριστικά*

Το *Node* χαρακτηρίζεται από την έμφαση στην ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων. Αυτό επιτυγχάνεται με την χρήση συμβάντων (*events*) που προσφέρει η Javascript και ονομάζονται callbacks. Για παράδειγμα όταν ένας περιηγητής ιστού φορτώσει πλήρως ένα αρχείο, ένας χρήστης πατάει κάποιο κουμπί, ολοκληρώνεται ένα αίτημα AJAX, τα συμβάντα αυτά πυροδοτούν ένα συγκεκριμένο *callback*. Αυτό με την σειρά του επιτρέπει την ροή του κώδικα χωρίς να αφήνει ανενεργό τον επεξεργαστή προκειμένου να εκτελεστεί μια λειτουργία, όπως μια επιτυχής ανάγνωση αρχείου από τον δίσκο.

### *3.1.3 Παραδείγματα*

#### *HTTP Server (εξυπηρετητή)*

Ένα χαρακτηριστικό παράδειγμα *node* για ένα απλό HTTP εξυπηρετητή είναι ο παρακάτω κώδικας

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Ο κώδικας αυτός δημιουργεί έναν εξυπηρετητή ο οποίος τρέχει τοπικά στην θύρα "1337". Ο χρήστης μπορεί στην μπάρα του φυλλομετρητή του να γράψει την διεύθυνση url "http://127.0.0.1:1337/" και να του εμφανιστεί μια σελίδα με το "Hello World" σαν κείμενο. Για την εκτέλεση του παραπάνω προγράμματος απαιτείται η αποθήκευση του παραπάνω κώδικα σε ένα αρχείο (έστω my_basic_http_server.js) με την εντολή:

```
$ node my_basic_http_server.js
```

στην κονσόλα του συστήματος που χρησιμοποιεί.

Ο χρήστης μπορεί στην μπάρα του φυλλομετρητή του να γράψει την διεύθυνση url "http://127.0.0.1:1337/" και να του εμφανιστεί μια σελίδα με το "Hello World" σαν

κείμενο. Για την εκτέλεση του παραπάνω προγράμματος απαιτείται η αποθήκευση του παραπάνω κώδικα σε ένα αρχείο (έστω my_basic_http_server.js) με την εντολή:
$ node my_basic_http_server.js
στην κονσόλα του συστήματος που χρησιμοποιεί.

***Δημιουργία αρχείου***

```
var fs = require('fs')
fs.writeFile("D:\myfile.txt", "Hello World", function(err, data) {
   if (err) throw err;
   console.log(data);
})
```

Ο παραπάνω κώδικας χρησιμοποιεί την έτοιμη βιβλιοθήκη του node για την επεξεργασία αρχείων με την εντολή "require([library_name])" και δημιουργεί ή αντικαθιστά στην περίπτωση που υπάρχει στο δίσκο D το αρχείο "myfile.txt" με περιεχόμενο "hello world". Όταν ολοκληρωθεί η λειτουργία αυτή της εγγραφής, πυροδοτείται το callback το οποίο ορίστηκε, και αυτό με την σειρά του είτε στέλνει ένα μήνυμα προβλήματος στην περίπτωση αποτυχίας, είτε στέλνει τα δεδομένα του αρχείου στην κονσόλα του χρήστη.
Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα από βιβλιοθήκες που προορίζονται ή είναι συμβατές με το *node*. Ανάμεσά τους εργαλεία που ξεχώρισαν όπως το node-mysql, το Mongodb και το Express παίζουν σημαντικό ρόλο υποστηρίζοντας την ασύγχρονη διάδραση με τις παραδοσιακές και NoSQL μεθόδους βάσεων δεδομένων. Αυτό επιτυγχάνεται με την χρήση του node package manager το οποίο επιτρέπει την εγκατάσταση των παραπάνω βιβλιοθηκών. Χρησιμοποιείται συνήθως σε εφαρμογές Chat, Proxy, Http Server καθώς και για παρακολούθηση εφαρμογών και του συστήματος (monitoring).

Όταν ολοκληρωθεί η λειτουργία αυτή της εγγραφής, πυροδοτείται το callback το οποίο ορίστηκε, και αυτό με την σειρά του είτε στέλνει ένα μήνυμα προβλήματος στην περίπτωση αποτυχίας, είτε στέλνει τα δεδομένα του αρχείου στην κονσόλα του χρήστη.

```
var fs = require('fs')

fs.writeFile("D:\myfile.txt", "Hello World", function(err, data) {
   if (err) throw err;
   console.log(data);
})
```

Ο παραπάνω κώδικας χρησιμοποιεί την έτοιμη βιβλιοθήκη του node για την επεξεργασία αρχείων με την εντολή "require([library_name])" και δημιουργεί ή αντικαθιστά στην περίπτωση που υπάρχει στο δίσκο D το αρχείο "myfile.txt" με περιεχόμενο "hello world". Όταν ολοκληρωθεί η λειτουργία αυτή της εγγραφής, πυροδοτείται το callback το οποίο ορίστηκε, και αυτό με την σειρά του είτε στέλνει ένα μήνυμα προβλήματος στην περίπτωση αποτυχίας, είτε στέλνει τα δεδομένα του αρχείου στην κονσόλα του χρήστη.
Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα από βιβλιοθήκες που προορίζονται ή είναι συμβατές με το *node*. Ανάμεσά τους εργαλεία που ξεχώρισαν όπως το node-mysql, το Mongodb και το Express παίζουν σημαντικό ρόλο υποστηρίζοντας την ασύγχρονη διάδραση με τις παραδοσιακές και NoSQL μεθόδους βάσεων δεδομένων. Αυτό επιτυγχάνεται με την χρήση του node package manager το οποίο επιτρέπει την εγκατάσταση των παραπάνω βιβλιοθηκών. Χρησιμοποιείται συνήθως σε εφαρμογές Chat, Proxy, Http Server καθώς και για παρακολούθηση εφαρμογών και του συστήματος (monitoring).

Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα από βιβλιοθήκες που προορίζονται ή είναι συμβατές με το *node*.

## *3.2 MongoDB*

Η MongoDB είναι μια βάση δεδομένων εγγράφων που παρέχει υψηλή απόδοση, υψηλή διαθεσιμότητα και εύκολη επεκτασιμότητα.

- Βάση δεδομένων εγγράφων
  - Έγγραφα (αντικείμενα) αντιστοιχίζονται καλά με τους τύπους δεδομένων γλωσσών προγραμματισμού.
  - Τα ενσωματωμένα έγγραφα και οι πίνακες(arrays) μειώνουν την ανάγκη για ζεύξεις.
    - Το δυναμικό σχήμα καθιστά ευκολότερο τον πολυμορφισμό.
- Υψηλή απόδοση
  - Η ενσωμάτωση κάνει το διάβασμα και την εγγραφή γρήγορη.

- Οι δείκτες μπορούν να περιλαμβάνουν κλειδιά από ενσωματωμένα έγγραφα και πίνακες.
- Προαιρετικές εγγραφές ροής (όχι αναγνωρίσεις).

- Υψηλή διαθεσιμότητα

  Επαναλαμβανόμενοι εξυπηρετητές με αυτόματο master failover.
- Εύκολη επεκτασιμότητα
  - Το αυτόματη sharding διανέμει συλλογικά δεδομένα σε όλες τις μηχανές.
  - Συνεπή διαβάσματα μπορούν να κατανεμηθούν σε επαναλαμβανόμενους εξυπηρετητές.
- Προηγμένες λειτουργίες

  Με την υπηρεσία MongoDB Management Service(MMS) η MongoDB υποστηρίζει μια ολοκληρωμένη λύση δημιουργίας αντιγράφων ασφαλείας και την πλήρη παρακολούθηση της ανάπτυξης.

### *3.2.1 MongoDB Μοντέλο Δεδομένων*

Μια ανάπτυξη MongoDB φιλοξενεί μια σειρά από βάσεις δεδομένων. Ένα *manual:database περιέχει* ένα σύνολο συλλογών. Ένα *manual:collection* περιέχει μια σειρά από έγγραφα. Ένα *manual:document* είναι ένα σύνολο από ζεύγη κλειδιού-τιμής. Τα έγγραφα έχουν δυναμικό σχήμα. Δυναμικό σχήμα σημαίνει ότι τα έγγραφα στην ίδια συλλογή δεν χρειάζεται να έχουν το ίδιο σύνολο πεδίων ή δομή, και κοινά πεδία στα έγγραφα μιας συλλογής μπορούν να έχουν διαφορετικούς τύπους δεδομένων.

### *3.2.2 MongoDB Αιτήματα( Queries)*

Τα αιτήματα MongoDB παρέχουν ένα σύνολο φορέων για να καθορίσουν πώς η find() μέθοδος επιλέγει έγγραφα από μια συλλογή βασισμένη σε ένα έγγραφο προδιαγραφών αιτημάτων που χρησιμοποιεί ένα συνδυασμό από ακριβώς ίσες αντιστοιχίες και συνθήκες που χρησιμοποιούν έναν φορέα αιτημάτων.

### *3.2.3 Ανάπτυξη Αρχιτεκτονικών*

Παρ' όλο που η MongoDB υποστηρίζει μια «αυτόνομη» ή μια λειτουργία ενός στιγμιότυπου, η ανάπτυξη της MongoDB διανέμεται προεπιλεγμένα. Σύνολα αντιγράφων παρέχουν υψηλής απόδοσης αντιγραφή με αυτόματο failover, ενώ sharded clusters επιτρέπουν να διανεμηθούν μεγάλα σύνολα δεδομένων σε πολλά μηχανήματα με διαφάνεια προς τους χρήστες. Οι χρήστες της MongoDB συνδυάζουν σύνολα αντιγράφων και sharded clusters για να παρέχουν υψηλά επίπεδα απολύσεων για μεγάλα σύνολα δεδομένων με διαφάνεια για τις εφαρμογές.

### *3.2.4 MongoDB φιλοσοφία σχεδίασης*

Οι νέες τεχνολογίες βάσεων δεδομένων απαιτούνται για να διευκολυνθεί η οριζόντια κλιμάκωση του στρώματος δεδομένων, ευκολότερη ανάπτυξη, καθώς και την ικανότητα να αποθηκεύει εντολές μεγέθους περισσότερων δεδομένων απ' ο,τι είχε χρησιμοποιηθεί στο παρελθόν.

Μια μη-σχεσιακή προσέγγιση είναι ο καλύτερος δρόμος σε λύσεις βάσεων δεδομένων που κλιμακώνονται οριζόντια σε πολλές μηχανές.

Είναι απαράδεκτο εάν αυτές οι νέες τεχνολογίες καθιστούν πιο δύσκολες τις εφαρμογές εγγραφής. Γράφοντας κώδικα θα πρέπει να είναι ταχύτερο, ευκολότερο και πιο ευέλικτο.

Το μοντέλο δεδομένων του εγγράφου (JSON / BSON ) είναι εύκολο να κωδικοποιηθεί, εύκολο να διαχειριστεί, και παρέχει εξαιρετική απόδοση με την ομαδοποίηση σχετικών στοιχείων μεταξύ τους εσωτερικά.

Είναι σημαντικό να διατηρηθεί η βαθιά λειτουργικότητα για να κρατήσει τον προγραμματισμό γρήγορο και απλό. Ενώ κάποια πράγματα πρέπει να μείνουν έξω, κρατάει όσο το δυνατόν περισσότερο - για παράδειγμα δευτερεύοντα ευρετήρια, περιορισμούς μοναδικού κλειδιού, ατομικές πράξεις, ενημερώσεις πολλαπλών εγγράφων.

Η τεχνολογία βάσης δεδομένων θα πρέπει να τρέχει οπουδήποτε, παραμένοντας διαθέσιμη τόσο για τη λειτουργία στους εξυπηρετητές ή σε εικονικές μηχανές, και επίσης ως μια cloud υπηρεσία pay-for-what-you-use.

### *3.2.5 Βασικά Χαρακτηριστικά της MongoDB*

Η MongoDB επικεντρώνεται στην ευελιξία, τη δύναμη, την ταχύτητα και την ευκολία στη χρήση:

**Ευελιξία**

Η MongoDB αποθηκεύει δεδομένα σε έγγραφα JSON (τα οποία σειριοποιούνται σε BSON ). Το JSON παρέχει ένα πλούσιο μοντέλο δεδομένων που αντιστοιχεί απρόσκοπτα σε τοπικές γλώσσες προγραμματισμού, καθώς και το δυναμικό σχήμα καθιστά ευκολότερο να εξελιχθεί μοντέλο δεδομένων απ' ο,τι με ένα σύστημα με επιβαλλόμενα σχήματα όπως ένα RDBMS.

**Δύναμη**

Η MongoDB παρέχει πολλά από τα παραδοσιακά χαρακτηριστικά της RDBMS, όπως δευτερεύοντα ευρετήρια, δυναμική αιτήματα, διαλογή, πλούσιες ενημερώσεις, upserts (ενημέρωση εάν υπάρχει έγγραφο, εισαγωγή αν δεν υπάρχει), και εύκολη συνάθροιση. Αυτό δίνει το εύρος των λειτουργιών που χρησιμοποιούνται και από τη RDBMS, με την ευελιξία και την ικανότητας κλιμάκωσης που επιτρέπει ένα μη-σχεσιακό μοντέλο.

**Ταχύτητα / Κλιμάκωση**

Με τη διατήρηση συναφών δεδομένων μαζί σε έγγραφα, τα αιτήματα μπορούν να είναι πολύ πιο γρήγορα απ' ο τι σε μια σχεσιακή βάση δεδομένων, όπου τα συσχετισμένα δεδομένα είναι χωρισμένα σε πολλαπλούς πίνακες και στη συνέχεια θα πρέπει να ενωθούν αργότερα. Η MongoDB καθιστά επίσης εύκολο την αναβάθμιση της βάσης δεδομένων. Το Autosharding επιτρέπει την κλιμάκωση του cluster γραμμικά προσθέτοντας περισσότερες μηχανές. Είναι δυνατόν να αυξηθεί η χωρητικότητα χωρίς κανένα downtime, που είναι πολύ σημαντικό στο διαδίκτυο, όταν το φορτίο μπορεί να αυξηθεί ξαφνικά.

**Ευκολία στη χρήση**

Η MongoDB είναι αρκετά εύκολη στην εγκατάσταση,τη ρύθμιση,τη συντήρηση και τη χρήση. Για το σκοπό αυτό, η MongoDB παρέχει λίγες επιλογές διαμόρφωσης, και αντί αυτού προσπαθεί να κάνει αυτόματα το "σωστό" όποτε είναι δυνατόν.

**Λειτουργίες**

Η MongoDB είναι μια διαδικασία εξυπηρετητή που τρέχει σε Linux, Windows και OS X. Μπορεί να τρέξει τόσο ως μια εφαρμογή 32 ή 64-bit. Είναι προτιμότερο να εκτελούνται σε λειτουργία 64-bit, αφού η MongoDB περιορίζεται σε ένα συνολικό μέγεθος δεδομένων γύρω στα 2GB για όλες τις βάσεις δεδομένων σε λειτουργία 32-bit.

Η διαδικασία MongoDB ακούει στη θύρα 27017 από προεπιλογή.

Οι πελάτες συνδέονται στην διαδικασία MongoDB, προαιρετικά επικυρώνουν τους εαυτούς τους αν η ασφάλεια είναι ενεργοποιημένη, και εκτελούν μια σειρά από δράσεις, όπως εισαγωγές, αιτήματα και ενημερώσεις.

Η MongoDB αποθηκεύει τα δεδομένα του σε αρχεία (προεπιλεγμένη θέση είναι /data/db/), και χρησιμοποιεί αντιστοιχισμένα αρχεία μνήμης για τη διαχείριση δεδομένων για αποτελεσματικότητα.

Η MongoDB μπορεί επίσης να ρυθμιστεί για αντιγραφή δεδομένων .

Επιπλέον, η εφαρμογή MongoDB Management Service (MMS) για τη διαχείριση MongoDB clusters χρησιμοποιώντας μια απλή διεπαφή χρήστη.Η MMS προσφέρει δημιουργία αντιγράφων ασφαλείας και παρακολούθησης.Η MMS είναι διαθέσιμη σε όλους τους χρήστες στο cloud και σε χώρους ως μέρος των MongoDB Standard and Enterprise Subscritions.

### *3.2.6 Επίσημα εργαλεία της MongoDB*

Σε μια εγκατάσταση MongoDB οι ακόλουθες εντολές είναι διαθέσιμες:

**mongo**

Η MongoDB προσφέρει ένα διαδραστικό shell που ονομάζεται Mongo, η οποία επιτρέπει στους προγραμματιστές να δουν, να τοποθετήσουν, να αφαιρέσουν, και να ενημερώσουν τα δεδομένα στις βάσεις τους, καθώς και να πάρουν πληροφορίες αντιγραφής, που έχει συσταθεί sharding, να κλείσουν τους εξυπηρετητες, να εκτελέσουν JavaScript, και περισσότερα.

Οι χρήστες επίσης μπορούν να έχουν πρόσβαση στις πληροφορίες διαχείρισης μέσω μιας διαδικτυακής διαπροσωπείας, μια απλή ιστοσελίδα που προσφέρει πληροφορίες

σχετικά με την τρέχουσα κατάσταση του διακομιστή. Από προεπιλογή, αυτή η διασύνδεση είναι 1000 θύρες πάνω από τη θύρα βάσης δεδομένων (28017).

**mongostat**

Το mongostat είναι ένα εργαλείο γραμμής εντολών που εμφανίζει μια συνοπτική λίστα με τα στατιστικά στοιχεία για το τρέχων στιγμιότυπο της MongoDB: πόσες εισαγωγές, ενημερώσεις, αφαιρέσεις, αιτήματα και εντολές εκτελέσθηκαν, καθώς και ποιο είναι το ποσοστό του χρόνου που η βάση δεδομένων ήταν κλειδωμένη και πόση μνήμη χρησιμοποιεί. Αυτό το εργαλείο είναι παρόμοιο με το UNIX / Linux vmstat utility.

**mongotop**

Το mongotop είναι ένα εργαλείο γραμμής εντολών που παρέχει μια μέθοδο για την παρακολούθηση του χρόνου που ξοδεύει ένα στιγμιότυπο της MongoDB για την ανάγνωση και την εγγραφή δεδομένων. Το mongotop παρέχει στατιστικά στοιχεία στο επίπεδο συλλογής. Από προεπιλογή, το mongotop επιστρέφει τιμές ανά δευτερόλεπτο. Αυτό το εργαλείο είναι παρόμοιο με το UNIX / Linux top utility.

**mongosniff**

Το mongosniff είναι ένα εργαλείο γραμμής εντολών που παρέχει μια ανίχνευση χαμηλού επιπέδου στις δραστηριότητες της βάσης δεδομένων από την παρακολούθηση (ή sniffing) της κυκλοφορίας του δικτύου προς και από τη MongoDB. Το mongosniff απαιτεί την libpcap βιβλιοθήκη του δικτύου και είναι διαθέσιμη μόνο για Unix συστήματα. Μια εναλλακτική λύση cross-platform είναι ο ανοιχτού κώδικα αναλυτής πακέτων Wireshark που έχει την πλήρη υποστήριξη για το πρωτόκολλο καλωδιώσεων της MongoDB.

**mongooplog**

Το mongooplog είναι ένα απλό εργαλείο που ανιχνεύει λειτουργίες από το oplog αντίγραφο ενός απομακρυσμένου εξυπηρετητή, και τις εφαρμόζει στον τοπικό εξυπηρετητή. Αυτή η δυνατότητα υποστηρίζει συγκεκριμένες κατηγορίες μετάβασης σε πραγματικό χρόνο που απαιτούν ο εξυπηρετητής προέλευσης να παραμείνει σε απευθείας σύνδεση και σε λειτουργία καθ 'όλη τη διαδικασία μετάβασης.

**mongofiles**

Το mongofiles utility επιτρέπει να χειριστείτε αρχεία που είναι αποθηκευμένα στο στιγμιότυπο MongoDB σε GridFS αντικείμενα από τη γραμμή εντολών. Είναι

ιδιαίτερα χρήσιμο, καθώς παρέχει μια διεπαφή μεταξύ των αποθηκευμένων αντικειμένων στο σύστημα αρχείων και στο GridFS.

mongoimport, mongoexport

Το mongoimport είναι ένα βοηθητικό πρόγραμμα γραμμής εντολών για την εισαγωγή περιεχομένου από μια JSON, CSV, ή TSV εξαγωγή που δημιουργήθηκε από ένα mongoexport ή ενδεχομένως άλλες εξαγωγές δεδομένων.

mongodump, mongorestore

Το mongodump είναι ένα βοηθητικό πρόγραμμα γραμμής εντολών για τη δημιουργία μιας δυαδικής εξαγωγής του περιεχομένου μιας βάσης δεδομένων Mongo. Το mongorestore μπορεί να χρησιμοποιηθεί για να ξαναφορτώσει στοιχεία της βάσης δεδομένων που είχαν απομακρυνθεί.

## *3.3 WebHookIt*

Το WebHookIt παρέχει ένα παράδειγμα οπτικού προγραμματισμού για τη δημιουργία απλών υπηρεσιών του παγκόσμιου ιστού , την σύνθεση των υπηρεσιών αυτών, mash-ups , την ενσωμάτωση των διαδικτυακών εφαρμογών και του διαδικτυακού προγραμματισμού καθοδηγούμενου από συμβάντα(event driven). Είναι ένα project ανοιχτού λογισμικού , που διατίθεται βάσει άδειας του MIT , και η ιδέα προήλθε από τα Yahoo Pipes . Το WebHookIt δεν λειτουργεί ως Eclipse plugin όπως και τα προηγούμενα εμπορικά εργαλεία ( JOpera και BPEL Designer) , αλλά ως ένα αυτόνομο εργαλείο στο διαδίκτυο, με βάση την Javascript και συνδέεται με μια τοπική βάση δεδομένων ( MongoDB ) .

### *3.3.1 Βασικά χαρακτηριστικά*

- Visual Editor
- Εκτελέσεις βασισμένες σε cron

- RESTful API
  - Εκτέλεση με μια απλή αίτηση HTTP
  - Κλήση καλωδιώσεις μέσα από μια ιστοσελίδα με μια φόρμα για να εισάγονται οι παράμετροι.
  - Δημόσια εκτέλεση των καλωδιώσεων μέσω κλήσεων HTTP
  - Εύκολη ενσωμάτωση
  - Εκτέλεση από ένα email μέσω Mailhooks
- Επεκτάσιμο
  - Προσθήκη προσαρμοσμένων μονάδων
  - Σύνδεση σε οποιαδήποτε διαδικτυακή υπηρεσία
- Ασύγχρονο
- Ελαφρύ
- 100% Javascript

### *3.3.2 Περιπτώσεις χρήσης*

- Απλές διαδικτυακές υπηρεσίες
- Widget για μια ιστοσελίδα
- Δημιουργία Mashups
- Ακροατές Webhooks
  - Ειδοποιήσεις όπως ακριβώς τις χρειάζονται.
  - Συγχρονισμός των δεδομένων έτσι ώστε οι διαδικτυακές εφαρμογές να κρατούν επικαλυπτόμενα δεδομένα σε συγχρονισμό.
  - Αλυσιδωτή σύνδεση έτσι ώστε μια εφαρμογή να κάνει κάτι, όταν μια άλλη εφαρμογή χρησιμοποιείται.
  - Τροποποίηση έτσι ώστε ο χρήστης να έχει τη δυνατότητα να κάνει την εφαρμογή ελαφρώς διαφορετική.
  - Plugins ώστε άλλοι χρήστες να μπορούν να επεκτείνουν κατασκευάσουν plugins για την εφαρμογή κάποιου άλλου χρήστη.

### *3.3.3 Διαφορά από τα Yahoo! Pipes;*

Σίγουρα, η διαπροσωπεία ήταν η έμπνευση, αλλά το πεδίο εφαρμογής του WebHookIt είναι πολύ ευρύτερο. Ο κύριος λόγος αυτού του έργου είναι να έχουν καλύτερο έλεγχο πάνω σε κάθε μέρος του συστήματος.

Το WebHookIt βρίσκεται στη μέση των Yahoo! Pipes και υπηρεσιών όπως οι Scriptlets , Notify.io , ή Tarpipe.

- Δεν υπάρχουν περιορισμοί στην εκτέλεση.
- Έλεγχος στις πολιτικές της cache,
- Ενσωματωμένο σύστημα cron να καλούνται τα pipes που ζητούνται.
- Καλύτερη RESTful ενσωμάτωση.
- Προσαρμόσιμη.
- Ενσωμάτωση με ιδιωτικά συστήματα.
- Αλληλεπίδραση με το δίκτυο των αντικειμένων
- Τρέχει εντολές συστήματος.
- Ο κώδικας των μονάδων μπορεί να είναι σε Ruby, Python, κτλ
- Πρόσβαση στο σύστημα αρχείων

Φυσικά, χάνουμε κάποια χαρακτηριστικά:

- Δεν είναι μια υπηρεσία που φιλοξενείται: αλλά είναι αρκετά εύκολο να εγκατασταθεί ελεύθερα σε Joyent.
- Από τη σκοπιά της κοινότητας: Σχεδιάζεται να παρέχεται ένας packet manager της κοινότητας για να μοιράζονται τις καλωδιώσεις στο μέλλον.

Υπάρχουν δύο είδη μονάδων στο WebHookIt:

- **Βασικές μονάδες**: Αποτελούνται από απλές ασύγχρονες συναρτήσεις γραμμένες σε ένα αρχείο Node.js και από τους ορισμούς διεπαφής της μονάδας.

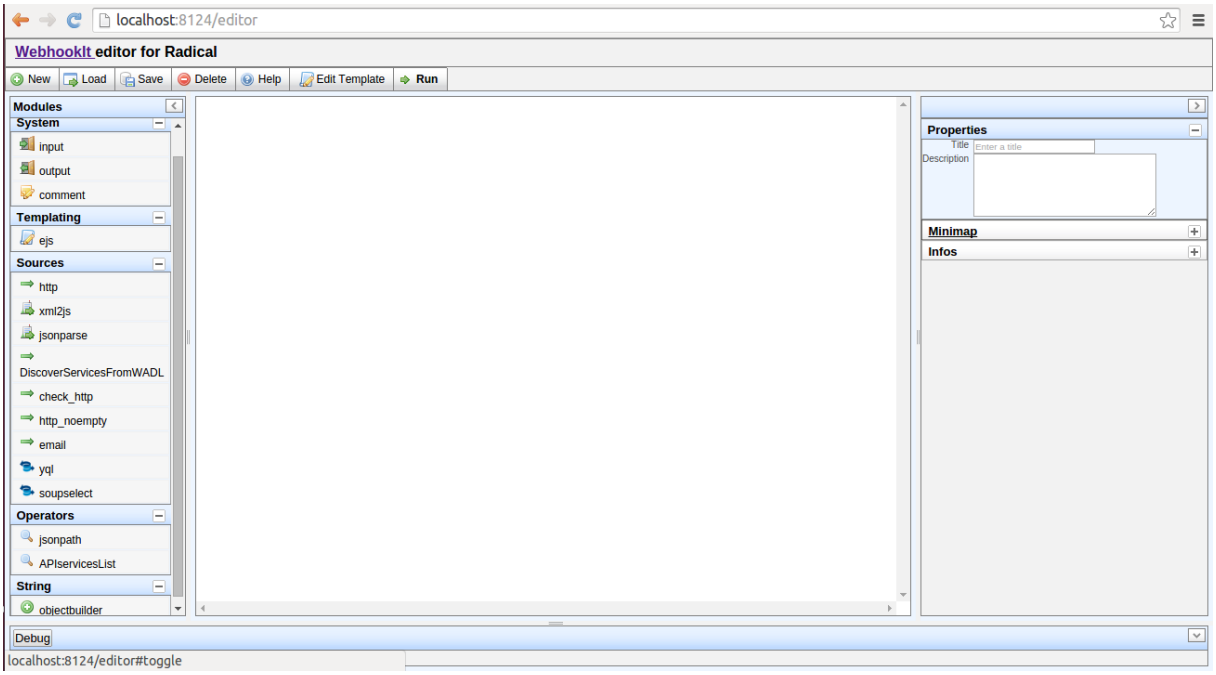
- **Μονάδες καλωδίωσης** : Δημιουργούνται μέσω του visual editor και είναι η σύνθεση διάφορων βασικών μονάδων.

Το WebHookIt υποστηρίζει τη σύνθεση των ροών εργασίας των υπηρεσιών REST σε Μονάδες καλωδίωσης μέσω ενός χρηστικού και απλού web GUI. Ο visual editor χρησιμοποιείται για να συνδυάσει απλές λειτουργίες με τη βοήθεια των "κουτιών" και των "καλωδίων". Αυτό είναι το ελεγκτικό μέρος της εφαρμογής. Το WebHookIt υποστηρίζει επίσης πολλαπλές επεκτάσεις με την προσθήκη νέων προσαρμοσμένων μονάδων, όπως η υποστήριξη ενός αποθετηρίου υπηρεσίας YQL. Επιπλέον, ένα πρόγραμμα αποσφαλμάτωσης συμπεριλαμβάνεται, για να αφήσει στον τελικό χρήστη να επιθεωρήσει τις τιμές επιστροφής της κάθε υπό-ενότητας, πατώντας σε διάφορες μονάδες.

Σε αντίθεση με το Yahoo Pipes που απαιτεί τοπικές εγκαταστάσεις, μια κεντρική εγκατάσταση μπορεί να γίνει στην κορυφή κάποιας πλατφόρμας, παρέχοντας πολλαπλές διεπαφές πελάτη. Σε κάθε περίπτωση, οι προγραμματιστές υπηρεσιών θα είναι σε θέση να συνδεθούν μέσω ενός απλού web browser, να ανακτήσουν όλες τις διαθέσιμες υπηρεσίες μέσω του YQL με δυνατότητα αποθήκευσης και να σχηματίσουν νέες εφαρμογές αποθήκευσης, αποθηκεύοντας τες σε τοπικό επίπεδο για να μπορούν να τις ξανακαλέσουν για οποιαδήποτε απαιτούμενη μελλοντική εκτέλεση
.

Φυσικά, θα χρειαστούν προσαρμοσμένες μονάδες, για να ρυθμιστεί η μονάδα YQL και URL υπηρεσιών που χρειάζονται οι καλωδιώσεις σε κλήσεις REST. Επιπλέον, οι επεκτάσεις πιθανόν θα χρειαστούν για να υλοποιηθεί ο χειρισμός όλων των run-time εξαιρέσεων που συμβαίνουν ή τη δημιουργία υποκαταστημάτων για εκτελέσεις REST και τα χρονόμετρα για την περιοδική εκτέλεση εφαρμογών εργαλείο.

### *3.3.4 Visual Editor*




**Εικόνα 5: Visual Editor**

Ένας visual editor που συνδυάζει απλές λειτουργίες με τη βοήθεια "κουτιών" και "καλωδιώσεων". Αυτό είναι το σημείο ελέγχου της εφαρμογής.

### Μονάδα εισόδου

Η μονάδα εισόδου προσθέτει μια νέα καταχώρηση στις παραμέτρους καλωδίωσης. Κάθε μονάδα εισόδου ορίζει ένα πεδίο φόρμας χρησιμοποιώντας τη βιβλιοθήκη inputEx.

### Μονάδα εξόδου

Οι μονάδες εξόδου αποτελούν τις επιστρεφόμενες τιμές των καλωδιώσεων. Μία καλωδίωση μπορεί να έχει πολλαπλές μονάδες εξόδου.

### Λειτουργία εντοπισμού σφαλμάτων

Χρειάζεται να πατηθεί το κουμπί "debug" για να εκτελεστεί η καλωδίωση.


**Εικόνα 6: Debug**

Συμπεριλαμβάνεται επίσης ο εντοπισμός σφαλμάτων.

Πατώντας στις διαφορετικές μονάδες επιτρέπει στους χρήστες να επιθεωρούν τις τιμές επιστροφής της κάθε υπομονάδας.

**Σύνθεση**

Μία σωστή καλωδίωση περιέχει μονάδες εισόδου και εξόδου.

Η φόρμα έχει δημιουργηθεί αυτόματα με την συνάθροιση των τομέων μονάδων εισόδου σε μια ενιαία μορφή.

Οι ακροδέκτες εξόδου δημιουργούνται σύμφωνα με τις μονάδες εξόδου στην αρχική καλωδίωση.

### *3.3.5 HTTP API*

**URL μοτίβο**

Το γενικό μοτίβο διεύθυνσης URL για την εκτέλεση μιας καλωδίωσης είναι ο εξής:

/wirings/:id/run[.(html|json|xml)]?param1=value1&param2=value2

**Φόρμα εκτέλεσης**

Αν πληκτρολογήσουμε στον browser την παρακάτω διεύθυνση URL, θα έχουμε μια φόρμα HTML για να επικαλεστούμε την καλωδίωση:

GET /wirings/:id

Η φόρμα χτίζεται αυτόματα από τις μονάδες εισόδου της καλωδίωσης.

Μπορούμε να προρυθμίσουμε διάφορες παραμέτρους σε αυτή τη μορφή:

GET /wirings/:id?param1=value1&param2=value2

**Πρότυπο Rendering**

Εάν χρησιμοποιείται κάποιο πρότυπο EJS.

### *3.3.6 Δραστηριότητες*

Στο /activities μπορούν να δουν οι χρήστες τις τελευταίες εκτελέσεις των καλωδιώσεων τους.

Μπορούν να επιθεωρήσουν:

- το κάλεσμα των παραμέτρων
- τη τιμή εξόδου
- το χρόνο ολοκλήρωσης της εκτέλεσης
- τη μορφή απάντησης

### *3.3.7 Εργασίες Cron*

Οι καλωδιώσεις μπορούν να προγραμματιστούν χρησιμοποιώντας ένα μοτίβο Cron.

**Χρησιμότητα**

Το μοτίβο cron περιέχει 6 πεδία χωρισμένα με κενά. Η μορφή είναι η εξής:

[δευτερόλεπτο] [λεπτό] [ώρα] [ημέρα του μήνα] [μήνας] [ημέρα της εβδομάδας]

Οι επιτρεπτές τιμές για τα πεδία του μοτίβου cron περιγράφονται στους παρακάτω πίνακες:

| Πεδία | Επιτρεπτές τιμές |
|---|---|
| **Δευτερόλεπτο** | 0-59 |

| | |
|---|---|
| **Λεπτό** | 0-59 |
| **Ώρα** | 0-23 |
| **Ημέρα του μήνα** | 1-31 |
| **Μήνας** | 0-11(0=Ιανουάριος) |
| **Ημέρα της εβδομάδας** | 1-7(1=Κυριακή) |

**Πίνακας 2: Επιτρεπτές τιμές cron**

Για να διαμορφωθεί κάθε πεδίο, μπορείτε να χρησιμοποιήσετε μία από τις παρακάτω επιλογές:

● Έναν αστερίσκο για να τρέχει κάθε στιγμιότυπο της τιμής του πεδίου.

Είναι ισοδύναμο με το εύρος [πρώτο-τελευταίο]. Για παράδειγμα, ένας αστερίσκος στο πεδίο Μήνα τρέχει την εργασία κάθε μήνα.

● Ένα εύρος.

Το συγκεκριμένο εύρος είναι χωρίς αποκλεισμούς. Για παράδειγμα, για να καθοριστεί ένας χρόνος λειτουργίας μεταξύ των ωρών 8:00 και 11:00 το εύρος του πεδίου Ώρα θα είναι [8-11], το οποίο ισοδυναμεί με [8,9,10,11].

● Ένα σύνολο αριθμών ή πεδίων διαχωρισμένα με κόμμα.

Για παράδειγμα, αν εισάγουμε [1,8-12,15] στο πεδίο Ημέρα του Μήνα, η εργασία εκτελείται στην πρώτη, όγδοη, ένατη, δέκατη, ενδέκατη, δωδέκατη και δέκατη πέμπτη ημέρα του μήνα.

● Ένα βήμα, χρησιμοποιείται με τα εύρη.

Πληκτρολογούμε "/" μετά από ένα εύρος για να καθορίσουμε ποιες αξίες να χρησιμοποιήσουμε στην περιοχή. Για παράδειγμα, [0-23 / 2] μπορεί να χρησιμοποιηθεί στον τομέα της ώρας για να διευκρινιστεί ότι η εργασία εκτελείται κάθε άλλη ώρα. Αυτό είναι ισοδύναμο με το να διευκρινίσουμε [0,2,4,6,8,10,12,14,16,18,20,22] στο πεδίο Ώρα. Τα βήματα είναι επίσης επιτρεπτά μετά τον αστερίσκο, οπότε αν θέλουμε να εκτελέσουμε μια εργασία κάθε δύο ώρες, θα μπορούσαμε να χρησιμοποιήσουμε [* / 2].

### *3.3.8 Δημόσια εκτέλεση των καλωδιώσεων*

Το WebHookIt δίνει πρόσβαση σε δημόσιους χρήστες να καλέσουν από το εξωτερικό

/wirings/public/:publicname

/wirings/public/:publicname.json

#### *3.3.9 Πρότυπα*

Κάθε μονάδα μπορεί να έχει το δικό της πρότυπο. Όταν τρέχει με HTML έξοδο, το πρότυπο θα χρησιμοποιείται αντί της προεπιλεγμένης σελίδας λειτουργίας.

/ καλωδιώσεις / ID / edit-pattern

Τα πρότυπα χρησιμοποιούν EJS.

Μπορούμε επίσης να ρυθμίσουμε το περιεχόμενο / τύπο.

Στο μέλλον, σχεδιάζεται να επιλέγονται προ-διαμορφωμένα πρότυπα (Google Maps, Exhibit, κλπ ...) πράγμα το οποίο είναι ιδανικό για Mashups.

### *3.3.10 Προσαρμοσμένες μονάδες διαπροσωπείας χρήστη*

Τα περισσότερα από τα στοιχεία UI του WebHookIt Editor παρέχονται από τη WireIt Javascript βιβλιοθήκη. Πιο συγκεκριμένα, τα WireIt "κουτιά" καλούνται **Containers.**

Κατά το σχεδιασμό μιας μονάδας WebHookIt, θα πρέπει να οριστεί ποιο Container πρέπει να χρησιμοποιηθεί, καθώς και κάποια ρύθμιση γι 'αυτό. Μπορούμε:

- Να προσαρμόσουμε ένα ήδη υπάρχον Container
- Να δημιουργήσουμε ένα ειδικό Container

Ο ευκολότερος τρόπος για να , είναι να χρησιμοποιήσουμε ένα υπάρχον Container. Το WireIt προ-καθορίζει κάποια είδη Container:

- FormContainer: Εμφανίζει μια φόρμα μέσα στο κουτί.
- InOutContainer: Είσοδοι / έξοδοι με όνομα.
- ImageContainer: Απλό Container εικόνας.
- TextareaContainer: Απλή ζώνη κειμένου.

Κατά τη δημιουργία μιας μονάδας από μια προηγούμενα καλωδίωση, το WebHookIt χρησιμοποιεί το ComposedContainer, που κληρονομεί το FormContainer. Το ComposedContainer δημιουργεί τη φόρμα από τις διάφορες μονάδες εισόδου που υπάρχουν στη καλωδίωση.

Η δημιουργία ενός ειδικού WireIt Container μπορεί να είναι δύσκολο, ανάλογα με το τι θέλουμε να κάνουμε. Θα χρειαστεί καλή εμπειρία και γνώση της Javascript,της Βιβλιοθήκης YUI, και του WireIt framework.

**Γενική μορφή-exports.definition**

Τα WireIt χαρακτηριστικά του container που αναπαριστούν τη μονάδα έχουν δομή JSON, που μοιάζει κάπως έτσι;

```
exports.definition = {
  "name": "yql",
  "category": "Sources",
  "container": {
    "xtype": "WireIt.TextareaContainer",
    "icon": "/images/module_icons/yql.png",
    "title": "yql"
  }
};
```

name

category

container

xtype

icon

title

Ανάλογα με το container xtype, το αντικείμενο container μπορεί να πάρει πρόσθετες παραμέτρους.

**Διαμόρφωση των τερματικών**

Όλα WireIt Container πρέπει να κληρονομούν τη βασική κλάση WireIt.Container.

Η κλάση αυτή καθορίζει τις ιδιότητες των τερματικών, η οποία είναι ένας πίνακας από ρυθμίσεις των τερματικών.

**YQL παράδειγμα**

Ας δούμε τη μονάδα YQL. Κατ 'αρχάς, όπως θα εμφανιστεί στον editor, ο ορισμός της μονάδας:

```
exports.definition = {
  "name": "yql",
    "category": "Sources",
  "container": {
        "icon": "/images/module_icons/yql.png",
     "xtype": "WireIt.TextareaContainer",
        "title": "yql",
        "fields": [
            {"type": "text", "name": "query", "wirable": true }
        ],
        "terminals": [
           {
            "name": "out",
            "direction": [0,1],
            "offsetPosition": {"left": 86, "bottom": -15},
            "ddConfig": { "type": "output", "allowedTypes": ["input"] }
       }
      ]
  }
};
```

**Μορφοποίηση με CSS**

Το WireIt αναθέτει δύο CSS κλάσεις στα containers:

- Το όνομα του container xtype
- Το όνομα του container

Η μονάδα YQL θα έχει τις ακόλουθες κλάσεις : WireIt-TextareaContainer και WiringEditor-module-YQL.

### *3.3.11 Εκτέλεση προσαρμοσμένων μονάδων*

Οι προσαρμοσμένες μονάδες χρησιμοποιούν Node.js και επιτρέπει να κάνουμε οτιδήποτε επιθυμούμε.

**Γενική μορφή**

Η προσαρμοσμένη μονάδα πρέπει να εξάγει μια συνάρτηση run. Αυτή η μέθοδος παίρνει δύο παραμέτρους:

**params**: ένας συσχετισμένος πίνακας (hash) που περιέχει τις παραμέτρους.

**CB**: η μέθοδος επανάκλησης(callback), που πρέπει να κληθεί όταν η μονάδα έχει τελειώσει.

```
exports.run = function(params, cb) {
cb({"out": response});
};
```

Οι μέθοδοι επανάκλησης πρέπει να κληθούν με αποτέλεσμα παραμέτρους αντικειμένων. Κάθε κλειδί του αντικειμένου πρέπει να συνδέεται με ένα τερματικό με το ίδιο όνομα.

**Παράδειγμα χρησιμοποιώντας το YQL**

Η μονάδα YQL είναι ένα τυπικό σενάριο, όπου υπάρχει ήδη μια βιβλιοθήκη Node.js για τη διαδικτυακή υπηρεσία. Η μέθοδος εκτέλεσης(run) γίνεται ένα πραγματικά απλό περιτύλιγμα γύρω από την κλήση της βιβλιοθήκης:

```
var YQL = require('yql');
exports.run = function(params, cb) {
   new YQL.exec(params.query, function(response) {
      cb({"out": response});
   }, {}, {"https": true});
};
```

### *3.3.12 Πακέτα προσαρμοσμένων μονάδων*

**NPM**

Οι βασικές WebHookIt μονάδες διανέμονται μέσω του NPM (http://npmjs.org/) διαχειριστή πακέτων, που σημαίνει ότι οποιοσδήποτε μπορεί να συνεισφέρει με WebHookIt μονάδες δημοσιεύοντας πακέτα για τον κατάλογο NPM που ονομάζεται "WebHookIt-mymodule".

Βασικές ενότητες είναι διαθέσιμες σε αυτό το αποθετήριο: https://github.com/neyric/WebHookIt-packages

Οι μονάδες WebHookIt μπορούν να εγκατασταθούν / ενημερωθούν / απεγκατασταθούν από τη σελίδα / πακέτα. Αυτή η σελίδα απαριθμεί αυτόματα όλα τα πακέτα ονομαζόμενα "WebHookIt-mymodule» από τον κατάλογο NPM.

Δημιουργώντας ένα πακέτο

Δημιουργήστε μια δομή καταλόγου όπως αυτό:

mymodule /

package.json

lib /

WebHookIt-mymodule.js

Το αρχείο πρέπει να περιέχει ένα index.js

Στη συνέχεια, ρυθμίζουμε το αρχείο package.json:

```
{
   "name": "WebHookIt-yql",
   "version": "0.0.1",
   "description": "YQL module for WebHookIt",
   "author": "Eric Abouaf <eric.abouaf@gmail.com>",
   "bugs": { "web" : "http://github.com/neyric/WebHookIt-packages/issues" },
   "os": ["darwin", "linux"],
   "contributors": [
      "Eric Abouaf <eric.abouaf@gmail.com> (http://neyric.com)"
   ],
   "engines": {
      "node" : ">=0.4.0"
   },
   "directories": {
      "lib" : "./lib"
   },
   "main": "./lib/WebHookIt-yql",
   "dependencies": {
      "yql": ">=0.2.0"
   },
   "licenses":[  {"type" : "MIT"} ],
   "repository": {
```

```
        "type":"git",
        "url":"http://github.com/neyric/WebHookIt-packages.git"
    }
}
```

Τελικά,δημοσιεύουμε το πακέτο στον κατάλογο:

```
cd mymodule/
npm publish .
```

Στη συνέχεια, πηγαίνουμε στη σελίδα / packages στην εγκατάσταση του WebHookIt και πατάμε στο κουμπί Install Package.

# 4

# *Δημιουργία ροών δεδομένων με την κατάλληλη χρήση και επέκταση του εργαλείου WebHookIt*

## *4.1 WebHookIt βασικές μονάδες*

Παρακάτω θα παρουσιάσουμε και θα αναλύσουμε τις βασικές μονάδες του WebHookIt καθώς και τις μονάδες που διαμορφώθηκαν. Στις διαμορφωμένες μονάδες παραθέτουμε και τον κώδικα για περεταίρο κατανόηση.

### *4.1.1 input*

Το input είναι η μονάδα εισόδου δεδομένων. Αποτελείται από ένα InOutContainer. Η μονάδα αυτή μπορεί να πάρει διάφορους τύπους δεδομένων που καθορίζονται στις ρυθμίσεις του. Οι τύποι αυτοί είναι:

string,group, combine, text, select, integer, number, email, url, list, boolean,inplaceedit, color, type.

Ανάλογα με τη μονάδα εισόδου που θα διαλέξει ο χρήστης μπορούν προαιρετικά να καθοριστούν και άλλες παράμετροι ανάλογα με τις ανάγκες του.


**Εικόνα 7: μονάδα input**

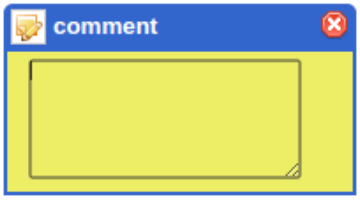
### *4.1.2 output*

Το output είναι η μονάδα εξόδου. Αποτελείται επίσης από ένα InOutContainer. Σε αυτή μπορούμε να καθορίσουμε το όνομα του αντικειμένου εξόδου μιας καλωδίωσης.


**Εικόνα 8: μονάδα output**

### *4.1.3 comment*

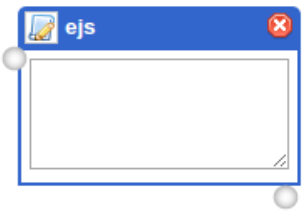
Όπως δηλώνει και η λέξη η μονάδα comment μας δίνει τη δυνατότητα να προσθέσουμε σχόλια στις καλωδιώσεις μας έτσι ώστε να είναι πιο κατανοητά.




**Εικόνα 9: μονάδα comment**

### 4.1.4 ejs

Η μονάδα ejs (Embedded Javascript Templates) χρησιμοποιείται για να λαμβάνει είσοδο ejs, στη συνέχεια εκτελεί τον κώδικα javascript και παράγει κώδικα html.

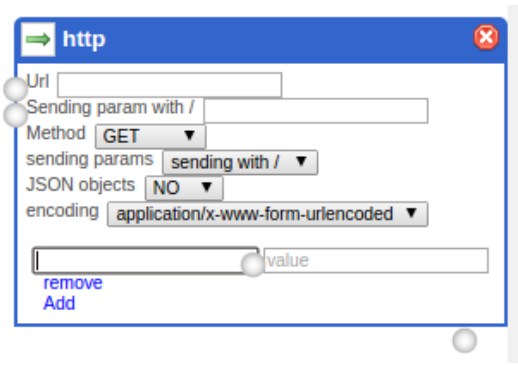



**Εικόνα 10: μονάδα ejs**

### 4.1.5 http

Αυτή η μονάδα είναι βασική για τις Restful διαδικτυακές υπηρεσίες. Αποτελείται από ένα FormContainer και έχει επεκταθεί σε σχέση με την αρχική μονάδα http του WebHookIt. Χρησιμοποιώντας το URL που μας δίνεται από τη διαδικτυακή υπηρεσία μπορούμε να εκτελέσουμε τις μεθόδους GET, POST, PUT, DELETE, HEAD θέτοντας κάθε φορά και τις κατάλληλες παραμέτρους. Τις παραμέτρους αυτές μπορούμε να τις στείλουμε είτε με τη μορφή URL/value είτε με τη μορφή URL?key=value. Επίσης πρέπει να καθορίσουμε αν οι παράμετροι εισόδου είναι κάποιο JSON αντικείμενο.

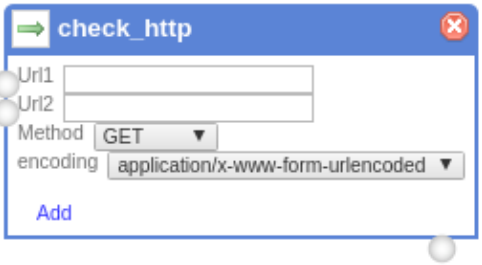
Τέλος,μας δίνεται η δυνατότητα να καθορίσουμε τη μορφή του αποτελέσματος διαλέγοντας το κατάλληλο encoding application/x-www-form-urlencoded, XML ή JSON.




**Εικόνα 11: μονάδα http**

### 4.1.6 *check_http*

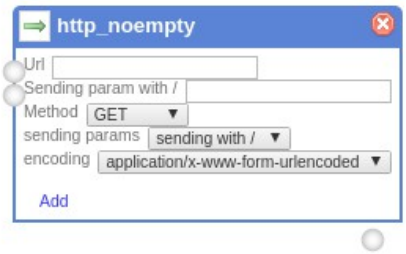
Αυτή η μονάδα είναι παρόμοια με την αρχική http με τη διαφορά ότι παίρνει δύο URLεισόδους και ελέγχει αν αυτές είναι ίσες ή όχι. Αν δεν είναι ίσες τότε βγάζει μήνυμα λάθους αλλιώς συνεχίζει κανονικά την εκτέλεση της.




**Εικόνα 12: μονάδα check_http**

### 4.1.7 *http_noempty*

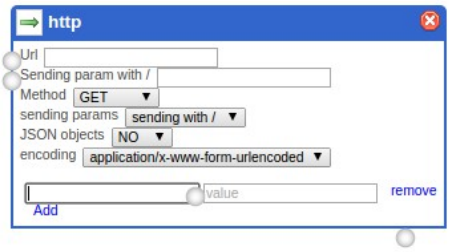
Και αυτή η μονάδα είναι παραπλήσια της αρχικής http με τη διαφορά ότι ελέγχει πριν την εκτέλεσή της αν η τιμή εισόδου δεν είναι κενή.




**Εικόνα 13: μονάδα http_noempty**

### *4.1.8 xml2js*

Η μονάδα αυτή χρησιμοποιεί ένα TextareaContainer και παίρνει ως είσοδο έναν κείμενο XML το οποίο μετατρέπει σε JSON.




**Εικόνα 14: μονάδα xml2js**

### *4.1.9 jsonparse*

Αυτή η μονάδα έχει παρόμοια λειτουργία με τη προηγούμενη με τη διαφορά ότι σε αυτή έχουμε ένα κώδικά JSON τον οποίο μετά την εκτέλεση μετατρέπουμε σε JSON αντικείμενα.


**Εικόνα 15: μονάδα jsonparse**

### *4.1.10 DiscoverServicesFromWADL*

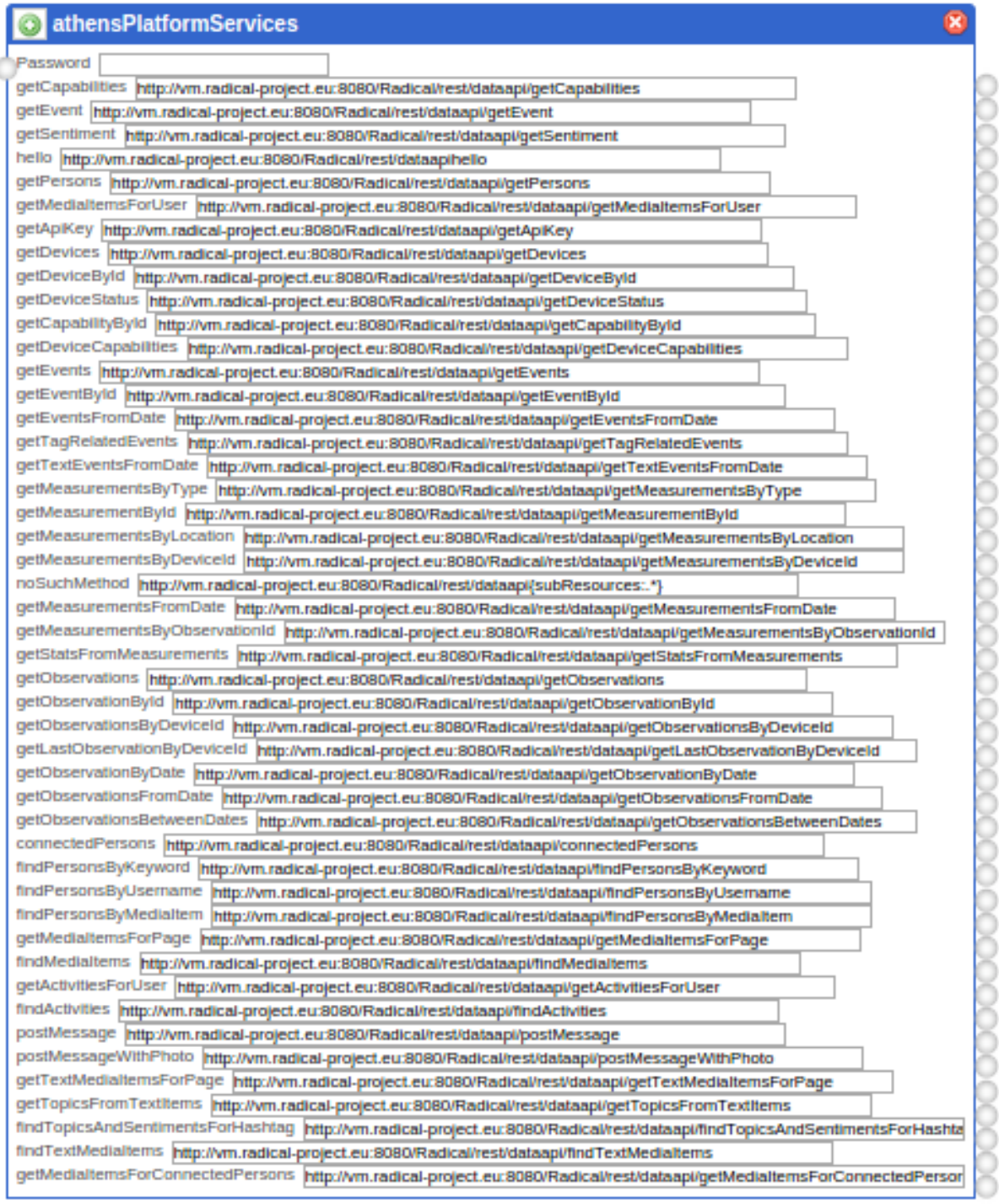
Η μονάδα αυτή παίρνει ως είσοδο ένα WADL URL και μέσω αυτού λαμβάνει τους συνδέσμους(links) που έχει. Αυτοί οι σύνδεσμοι εμφανίζονται σε μια νέα διαμορφωμένη μονάδα τη <city>PlatformServices η οποία περιέχει μια λίστα με τις υπηρεσίες και την οποία θα αναλύσουμε στη συνέχεια. Η καινούργια μονάδα εμφανίζεται όταν ανανεώσουμε τη σελίδα. Η συγκεκριμένη μονάδα μαζί με την επόμενη αποτελούν σημαντικά συστατικά του project Radical που θα αναλύσουμε στη συνέχεια.


**Εικόνα 16: μονάδα DiscoverServicesFromWADL**

### *4.1.11 <city>PlatformServices*

Αφού έχουμε τρέξει την μονάδα DiscoverServicesFromWADL και ανάλογα με την πόλη την οποία έχει δηλώσει ο χρήστης καλείται η αντίστοιχη μονάδα.

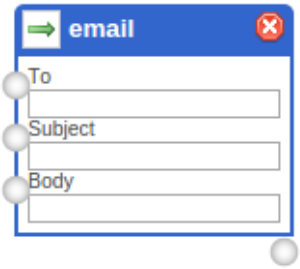
Αυτή περιέχει τα URL των διαδικτυακών υπηρεσιών και για να τα συνδέσουμε με κάποια άλλη μονάδα και για να εκτελεστεί πρέπει να εισάγουμε τον κωδικό που έχει η πόλη.




**Εικόνα 17: μονάδα <city>PlatformServices**

### *4.1.12 email*

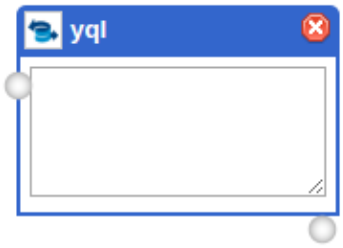
Αυτή η μονάδα στέλνει κάποιο e-mail ανάλογα με τις παραμέτρους που της δίνει ο χρήστης.




**Εικόνα 18: μονάδα email**

### *4.1.13 yql*

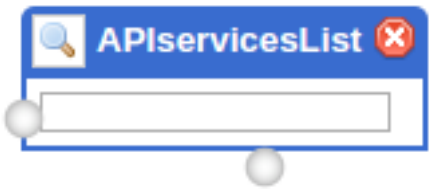
Η μονάδα αυτή αποτελείται από ένα TextareaContainer. Χρησιμοποιείται για την εκτέλεση αιτημάτων της γλώσσας Yahoo (Yahoo Query Language).




**Εικόνα 19: μονάδα yql**

### *4.1.14 APIservicesList*

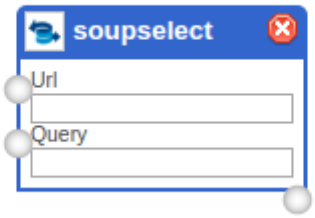
Επιστρέφει τη λίστα με τα διαθέσιμα DATA API που λαμβάνει από τον κατάλογο YQL ως αντικείμενα JSON και δημιουργεί μια μονάδα ServicesURLs που δίνει τις υπηρεσίες αυτές ως εισόδους.




**Εικόνα 20: μονάδα APIservicesList**

### *4.1.15 soupselect*

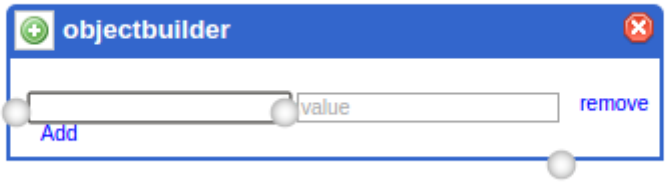
Η μονάδα αυτή λαμβάνει τον html κώδικα ενός URL και επιστρέφει τα tags που είναι καθορισμένα από το αίτημα (query).




**Εικόνα 21: μονάδα soupselect**

### *4.1.16 objectbuilder*

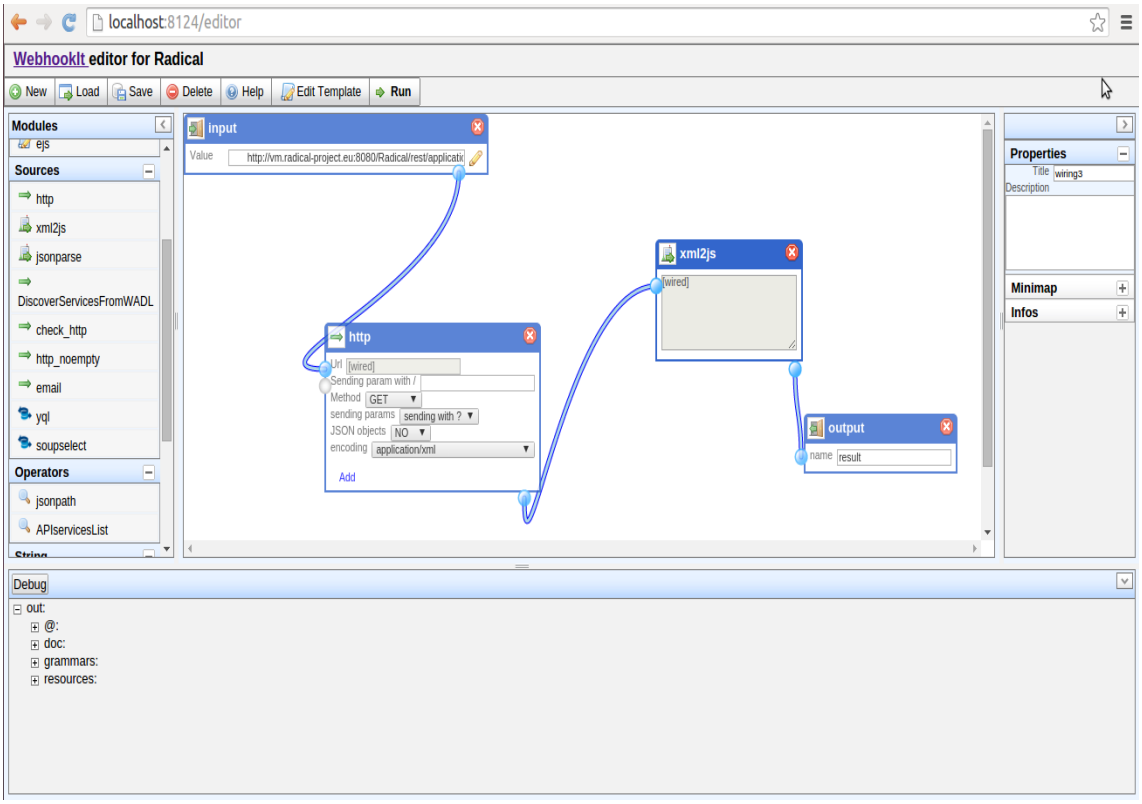
Και αυτή η μονάδα αποτελείται από ένα FormContainer. Η λειτουργία της είναι να δημιουργεί JSON αντικείμενα ανάλογα με τα κλειδί και τις τιμές που θέτει ο χρήστης.




**Εικόνα 22: μονάδα objectbuilder**

### *4.1.17 Παράδειγμα*

Με αυτό το απλό παράδειγμα μπορεί να κατανοήσει κάποιος τη σημασία του WebHookIt καθώς και τον τρόπο που μπορεί να συνδυάσει τις διάφορες μονάδες. Σε αυτό το παράδειγμα βλέπουμε μια καλωδίωση στην οποία δίνουμε σαν είσοδο (input) ένα URL. Όταν η μονάδα HTTP εκτελείται φέρνει τα δεδομένα σε μια μορφή XML. Στη συνέχεια χρησιμοποιείται η μονάδα xml2js η οποία μετατρέπει τα δεδομένα από XML σε JSON αντικείμενα. Τέλος χρησιμοποιούμε μια μονάδα εξόδου στην οποία δίνουμε το όνομα result και η οποία φέρει τα αποτελέσματα ως JSON αντικείμενα. Αυτή η καλωδίωση μπορεί να αποθηκευτεί για να χρησιμοποιηθεί και μελλοντικά και να αποτελεί από μόνη της μια μονάδα.




**Εικόνα 23: Παράδειγμα WebHookIt ροής εργασιών**

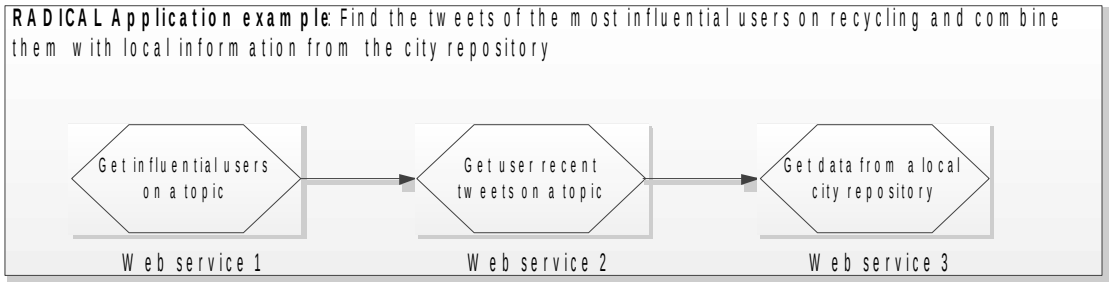
## *4.2 WebHookIt for RADICAL*

Αφορμή και σκοπός για την επέκταση του WebHookIt ήταν η ανάγκη ενός εργαλείου κατάλληλου για τη δημιουργία ροών εργασιών (workflows) κοινωνικών δικτύων και Internet of Things REST διαδικτυακών υπηρεσιών ώστε να χρησιμοποιηθεί στην πλατφόρμα RADICAL.

Η πλατφόρμα αυτή επιτρέπει την ανάπτυξη και εξάπλωση των διαλειτουργικών διάχυτων πολύ-αισθητηριακών και κοινωνικά ενημερωμένων υπηρεσιών, με τη χρήση του Internet of Things και των τεχνολογιών των κοινωνικών δικτύων. Οι υπηρεσίες της πλατφόρμας RADICAL ακολουθούν μια προσέγγιση SOA, διασυνδέοντας πολλαπλά στοιχεία και παρέχοντας τα API τους στη μορφή μιας δικτυακής υπηρεσίας. Επίσης, η πλατφόρμα RADICAL διευκολύνει την έξυπνη διακυβέρνηση και την ευέλικτη αναπαραγωγή των υπηρεσιών στις πόλεις και τις περιφέρειες, μέσω μηχανισμών που αξιολογούν τις ιδιαιτερότητες των πόλεων από την άποψη τόσο των τεχνικών υποδομών τους και των κοινωνικό-οικονομικών χαρακτηριστικών και των νομικών διοικητικών περιβαλλόντων.

Αυτές οι απαιτήσεις οδηγούν σε μια προσέγγιση προσανατολισμένη στις υπηρεσίες για την παροχή και την ολοκλήρωση των υπηρεσιών RADICAL. Γι' αυτό αποφασίστηκε ότι όλες οι λειτουργίες του RADICAL DATA API πρέπει να παρέχονται ως ένας συνδυασμός διαδικτυακών μεθόδων, με στόχο να παρέχουν στους προγραμματιστές των υπηρεσιών (Service Developers) διάφορα εργαλεία για να ανακαλύψουν τις βασικές υπηρεσίες του RADICAL (RADICAL Core Services) και να δημιουργήσουν νέες εφαρμογές συνδυάζοντας αυτές τις υπηρεσίες σε ροές εργασιών. Πιο αναλυτικά οι προγραμματιστές θα πρέπει να είναι σε θέση να χρησιμοποιούν τις βασικές υπηρεσίες για να σχηματίσουν ροές, με την ακολουθιακή εκτέλεση των υπηρεσιών και αντιστοιχώντας το αποτέλεσμα μιας υπηρεσίας στις παραμέτρους εισόδου κάποιας άλλης. Αυτό έχει ως αποτέλεσμα μια σύνθετη ροή εργασιών των υπηρεσιών, δεδομένου που οι είσοδοι που παρέχονται σε κάθε υπηρεσία μπορούν να καταναλωθούν με έναν ουσιαστικό τρόπο, για να παράγουν πολλαπλά αποτελέσματα και να τα συνδυάσουν αυτά σε ένα σύνολο αποτελεσμάτων ενός συνόλου υπηρεσιών.

Η διαδικασία που περιγράφεται παραπάνω, πρακτικά οδηγεί στην εκτέλεση ενός συνδυασμού υπηρεσιών, Ένα παράδειγμα μιας εφαρμογής RADICAL απεικονίζεται παρακάτω:




**Εικόνα 24: Παράδειγμα εφαρμογής RADICAL**

Σε αυτό το παράδειγμα, ο Service Developer ανακαλύπτει και χρησιμοποιεί τρεις διαδικτυακές υπηρεσίες που θα μπορούσαν να παρέχονται από το RADICAL Data API, προκείμενου να δημιουργηθεί μια συγκεκριμένη RADICAL εφαρμογή. Η εφαρμογή ανακτά τις θέσεις σχετικά με την ανακύκλωση από τους πλέον σημαντικούς χρήστες του Twitter και συνδυάζει αυτά τα τοπικά δεδομένα της πόλης για την ανακύκλωση, τα οποία λήφθηκαν από το αποθετήριο της πόλης. Η εφαρμογή χρησιμοποιεί τρεις διακριτές βασικές υπηρεσίες που το RADICAL παρουσιάζει ως διαδικτυακές υπηρεσίες REST:

Διαδικτυακή υπηρεσία 1: Μια υπηρεσία που παρέχει ο σημαντικός χρήστης του Twitter σε ένα θέμα(ανακύκλωση), που είναι κατά κάποιο τρόπο εγγραφεί στην πλατφόρμα RADICAL η στην τοπική εφαρμογή.

Διαδικτυακή υπηρεσία 2: Μια υπηρεσία που ανιχνεύει όλες τις πρόσφατες δημοσιεύσεις Twitter (tweets) από αυτούς τους χρήστες σχετικά με το θέμα αυτό.

Διαδικτυακή υπηρεσία 3: Μια υπηρεσία που παρέχει δεδομένα σχετικά με αυτό το θέμα από το τοπικό αποθετήριο της πόλης.

Οι εφαρμογές ροής εργασίας που δημιουργούνται με αυτόν τον τρόπο, θα πρέπει να είναι επίσης οδηγούμενες από συμβάντα(event-driven), πράγμα που σημαίνει ότι ένα γεγονός (όπως μια δημοσίευση ενός χρήστη ή ένα νέο θέμα της πόλης ανιχνεύεται σε κοινωνικά δίκτυα ή εφαρμογές smartphone, ή ακόμη και μια μέτρηση του αισθητήρα πάνω από ένα ορισμένο όριο) μπορεί να ενεργοποιούν συγκεκριμένες κλήσεις υπηρεσιών (όπως υπηρεσίες ειδοποίησης). Αυτό μπορεί να υλοποιηθεί με τη χρήση

χρονομέτρου, τα οποία καλούν σε τακτά χρονικά διαστήματα την εφαρμογή ροής εργασίας να ελέγξει την αξία ορισμένων μετρήσεων ή ανακτηθέντων στοιχείων πολυμέσων.

Επιπλέον, μετά την ανακάλυψη υπηρεσιών ο Service Developer πρέπει να βλέπει μόνο τις υπηρεσίες και τα δεδομένα που αυτός είναι εξουσιοδοτημένος να δει, ή που έχουν νόημα με βάση την τοποθεσία του κ.λπ. Αυτό σημαίνει ότι περιορισμοί απαιτούνται για να διασφαλιστεί ότι όχι κάθε συνδυασμός υπηρεσίας είναι διαθέσιμος σε κάθε είδος Service Developer.

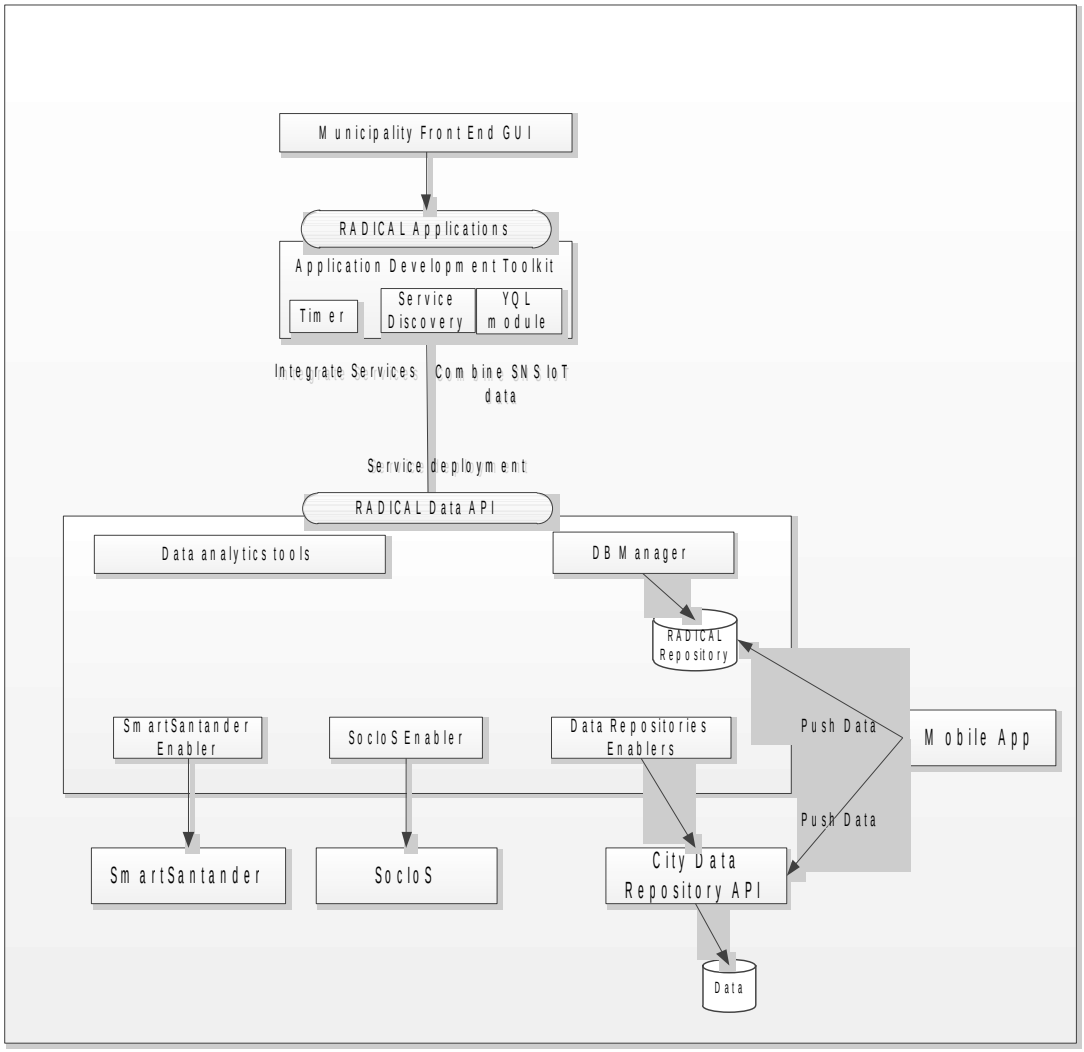
### *4.2.1 Ανάλυση του περιβάλλοντος ανάπτυξης εφαρμογών RADICAL.*

#### *4.2.1.1 Επισκόπηση εργαλείου.*

Το περιβάλλον ανάπτυξης του RADICAL θα είναι το εργαλείο υπεύθυνο για το συνδυασμό των βασικών υπηρεσιών σε ροές εργασιών και την εκτέλεση αυτών των ροών εργασίας. Η λειτουργικότητα του στοιχείου παρουσιάζεται στους Service Developers μέσω ενός χρηστικού GUI όπου μπορούν να δημιουργήσουν και να εκτελέσουν RADICAL εφαρμογές, χρησιμοποιώντας απλές drag and drop ενέργειες.

Το πακέτο ανάπτυξης εφαρμογών θα περιλαμβάνει λειτουργίες χρονοδιακόπτη, για την περιοδική εκτέλεση της εφαρμογής, καθώς και μια μονάδα Service Discovery, το οποίο θα χρησιμοποιεί μια μονάδα Yahoo! Query Language για την ανάκτηση των υπηρεσιών και τη σύνθεση τους. Το Σχήμα 9 παρουσιάζει το πακέτο στο πάνω μέρος της RADICAL αρχιτεκτονικής.

Όπως φαίνεται παρακάτω, οι Service Developers θα είναι σε θέση να έχουν πρόσβαση στο περιβάλλον ανάπτυξης, μέσω του δήμου GUI, και πιο συγκεκριμένα μέσω ενός web browser.




**Εικόνα 25: επισκόπηση εργαλείου**

### *4.2.1.2 Απαιτήσεις*

Για τον ορισμό των λειτουργικών και μη λειτουργικών προδιαγραφών του Πακέτου Ανάπτυξης Εφαρμογών χρειαζόταν η ύπαρξη ενός εργαλείου που να πληρεί όλες τις απαιτήσεις των τελικών χρηστών του περιβάλλοντος και των Application Developers.

Οι απαιτήσεις αυτές ήταν οι εξής:

- Γραφική διεπαφή χρήστη (GUI)

Για την εύκολη διαχείριση των παραμέτρων και των συνδέσεων των βασικών υπηρεσιών(Core Modules), θα πρέπει να παρέχεται ένα γραφικό περιβάλλον. Τα Core Services θα πρέπει να παρουσιάζονται ιδανικά ως απλά κουτιά με εισόδους και εξόδους, με συνδέσμους που επιτρέπουν τις υπηρεσίες μιας ροής εργασίας να συνδεθούν, και να ορίσουν κάποιες από τις εξόδους μιας υπηρεσίας ως εισροές σε μια άλλη υπηρεσία. Ο προγραμματιστής μπορεί να είναι σε θέση να παράσχει εισροές για

τις υπηρεσίες που δεν παίρνει τίποτα από την προηγούμενη υπηρεσία ροής εργασίας, καθώς και τα τελικά αποτελέσματα εφαρμογής θα πρέπει να συνδυάζουν τις εξόδους των τελευταίων υπηρεσιών με ενδιάμεσες εξόδους υπηρεσιών.

- Αποθήκευση δημιουργηθέντων εφαρμογών

Μια βασική απαίτηση για τον ορισμό μιας εφαρμογής είναι ο συνδυασμός βασικών υπηρεσιών σε μια συγκεκριμένη υπηρεσία ροής εργασίας και να υπάρχει η δυνατότητα αποθήκευσης αυτής της ροής εργασίας ως μια RADICAL εφαρμογή. Ο προγραμματιστής θα πρέπει να είναι σε θέση να φορτώσει την αποθηκευμένη εφαρμογή κάθε φορά που μπαίνει στο Περιβάλλον Ανάπτυξης Εφαρμογών.

- Περιοδική εκτέλεση της ροής εργασίας

Ορισμένες εφαρμογές ροής εργασίας μπορεί να είναι χρήσιμο να εκτελούνται περιοδικά (π.χ. feeds κοινωνικής δικτύωσης και λειτουργιών) ή προγραμματισμένες για συγκεκριμένο χρονικό διάστημα. Έτσι, ένα χρονόμετρο θα ήταν χρήσιμο για την προγραμματισμένη εκτέλεση.

- Διαχείριση εξαιρέσεων

Ένα γραφικό σύστημα χειρισμού εξαίρεσης απαιτείται, σε περίπτωση που συμβαίνουν εξαιρέσεις κατά την εκτέλεση της αίτησης. Αντί απλά μηνύματα λάθους, το περιβάλλον πρέπει να επισημάνει τις ανακρίβειες μετά από μια αποτυχία.

- Περιορισμοί ροών εργασίας

Όχι όλες οι πιθανές ροές εργασίας μπορούν να είναι διαθέσιμες, συνδυάζοντας διαφορετικά Core Services. Μπορεί να υπάρχουν ορισμένοι περιορισμοί για τον καθορισμό μιας ροής εργασίας, δηλαδή λόγω των νομικών ζητημάτων μια βασική υπηρεσία που συγκεντρώνει ορισμένα στοιχεία κοινωνικής δικτύωσης δεν μπορεί να συνδυαστεί με μια συγκεκριμένη υπηρεσία ανάλυσης.

- Τελική διαθεσιμότητα εφαρμογής

Η τελική παραγόμενη εφαρμογή , όπως η ροή εργασίας των διαφόρων βασικών υπηρεσιών, θα πρέπει να είναι διαθέσιμη ως μια διαδικτυακή υπηρεσία ή μια διεπαφή που θα χρησιμοποιηθεί από έναν δήμο ως web front-end ή τοπική εφαρμογή.

Όλες αυτές τις προϋποθέσεις τις πληρεί το WebHookIt και ο editor του και γι' αυτό είναι και το πλέον καταλληλότερο εργαλείο για το Περιβάλλον Ανάπτυξης Εφαρμογών.

### *4.2.2 Το WebHookIt ως περιβάλλον ανάπτυξης εφαρμογών RADICAL.*

Ο WebHookIt editor for Radical είναι η κεντρική μονάδα, υπεύθυνη για τη διαμόρφωση και εκτέλεση των εφαρμογών ροής εργασίας που δημιουργούνται μέσω ενός GUI και αποθηκεύονται ή φορτώνονται από τον τελικό χρήστη, χρησιμοποιώντας ένα μοναδικό όνομα. Είναι προσβάσιμο μέσω ενός web browser, και παρέχει τη δυνατότητα να συνδέσουμε REST http κλήσεις σε μια ροή καλωδιωμένων εργασιών που συνδέουν τις εξόδους μιας υπηρεσίας στις εισόδους μιας άλλης υπηρεσίας. Επιπλέον, παρέχει ένα χρονοδιάγραμμα cronjob που ορίζει την περιοδική ή προγραμματισμένη εκτέλεση μιας καλωδίωσης ή μεμονωμένης κλήσης http στο μέλλον, όπως ζητήθηκε στον κατάλογο των προϋποθέσεων που απαιτούνται από τους Service Developers.

**Λειτουργίες**

- Wire: function (terminal1, terminal2, parentEl, options)

Καλωδιώνει τη μονάδα πόρου terminal1 (έξοδος υπηρεσίας ή τιμή εισόδου) σε μια μονάδα στόχου Terminal 2 (εισόδου ή εξόδου υπηρεσία) για να δημιουργήσει μια απλή διπλού αντικειμένου συνδεδεμένη ροή εργασίας. Το parentElvariable είναι η κλάση Container της ετικέτας CAMVAS και οι παράμετροι των επιλογών καθορίζουν τις ρυθμίσεις των καλωδιώσεων.

- Save: function(wiring, docs)

Αποθηκεύει μια δημιουργηθείσα εφαρμογή-καλωδίωση(wiring). Η μεταβλητή καλωδίωσης περιέχει τα ονόματα των μεθόδων και τις συνδέσεις των υπηρεσιών, ενώ τα docs περιέχουν τα έγγραφα της καλωδίωσης.

- Load: function (name)

Φορτώνει την καλωδίωση με το συγκεκριμένο όνομα.

- Run: function()

Διαδοχικά εκτελεί μονάδες (υπηρεσίες) της ροής εργασίας που έχει φορτωθεί. Εναλλακτικά, ο συντάκτης του τελικού χρήστη μπορεί να δοκιμάσει το "debug", η

οποία παρέχει την ίδια λειτουργικότητα, την εκτύπωση όλων των εξόδων της μονάδας στο GUI terminal εξόδου.

- CronJob: function (cronTime, onTick, onComplete, start, timeZone, context)

Αυτή η μέθοδος καθορίζει την περιοδική ή προγραμματισμένη εκτέλεση μιας καλωδίωσης, χρησιμοποιώντας ένα χρονόμετρο. Οι παράμετροι εισόδου παρέχουν τις διαστάσεις του χρόνου, όπως το χρόνο εκτέλεσης και τη ζώνη ώρας που χρησιμοποιείται, καθώς και τις συνθήκες εκτέλεσης και το πλαίσιο καλωδίωση.

Όπως περιγράψαμε αναλυτικά προηγουμένως όλες οι απαιτήσεις συμπεριλαμβάνονται στο WebHookIt γι' αυτό και φαίνεται να αποτελεί την ιδανική λύση για τη δόμηση της βάσης του Περιβάλλοντος Ανάπτυξης Εφαρμογών για το RADICAL. Δεδομένου ότι απαιτεί τοπικές εγκαταστάσεις, μια κεντρική εγκατάσταση έγινε στην κορυφή της πλατφόρμας RADICAL, παρέχοντας πολλαπλές διεπαφές πελάτη, ή, εναλλακτικά,η μια ανάπτυξη ανά πόλη θα μπορούσε να βοηθήσει τους σκοπούς του έργου. Επομένως οι Service Developers είναι σε θέση να συνδεθούν μέσω ενός απλού web browser, να ανακτήσουν όλες τις διαθέσιμες υπηρεσίες και να σχηματίσουν νέες βασικές εφαρμογές, αποθηκεύοντας τες σε τοπικό επίπεδο για να είναι σε θέση να τις επαναφορτώσουν για κάθε μελλοντική εκτέλεση.

Φυσικά, χρειάζονταν προσαρμοσμένες μονάδες, για να διαμορφωθεί η μονάδα YQL και να καλωδιωθούν τα URL των επιθυμητών υπηρεσιών στις κλήσεις REST.

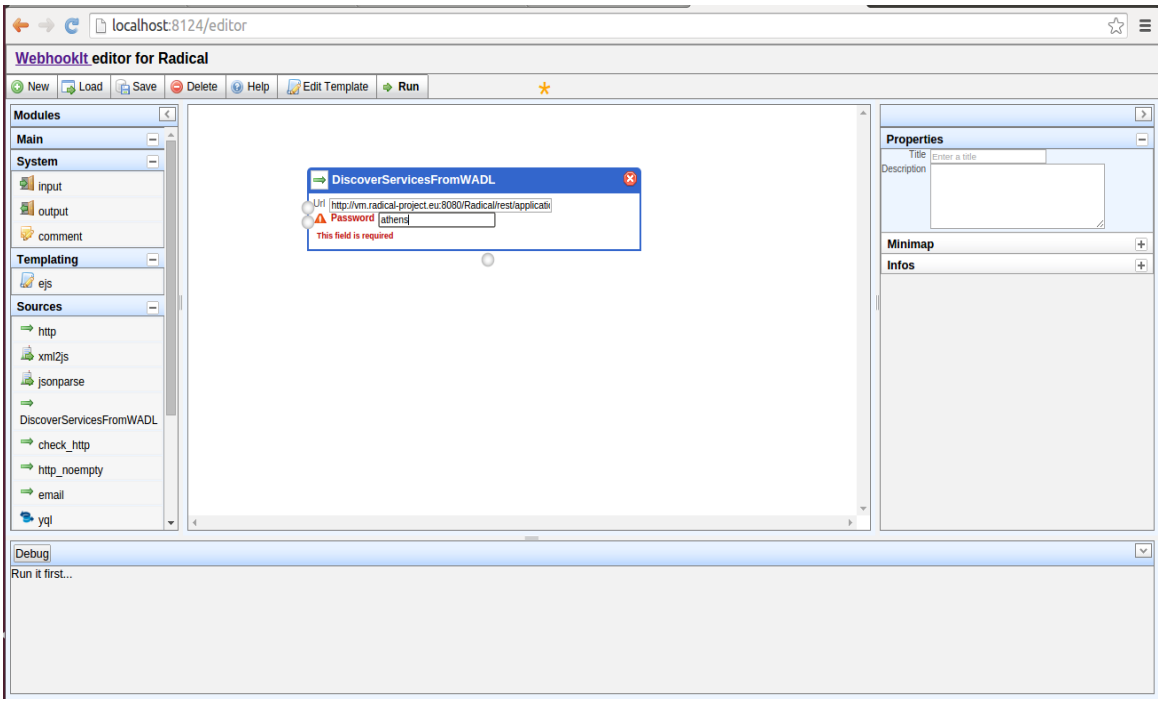
### *4.2.3 Service Discovery*

Το Service Discovery είναι ένα συστατικό καλωδίωσης του WebHookIt για την ανάκτηση ή την επικαιροποίηση του καταλόγου όλων των διαθέσιμων υπηρεσιών του RADICAL. Χειρίζεται την επικοινωνία μεταξύ του εργαλείου και του αποθετηρίου YQL. Κατά την πορεία αυτή παρέχει τη λίστα με τις διαθέσιμες υπηρεσίες από το YQL αποθετήριο ως ServicesURLs που ο χρήστης μπορεί να χρησιμοποιήσει ως εισόδους σε άλλες εφαρμογές.

### *4.2.4 DiscoverServicesFromWADL*

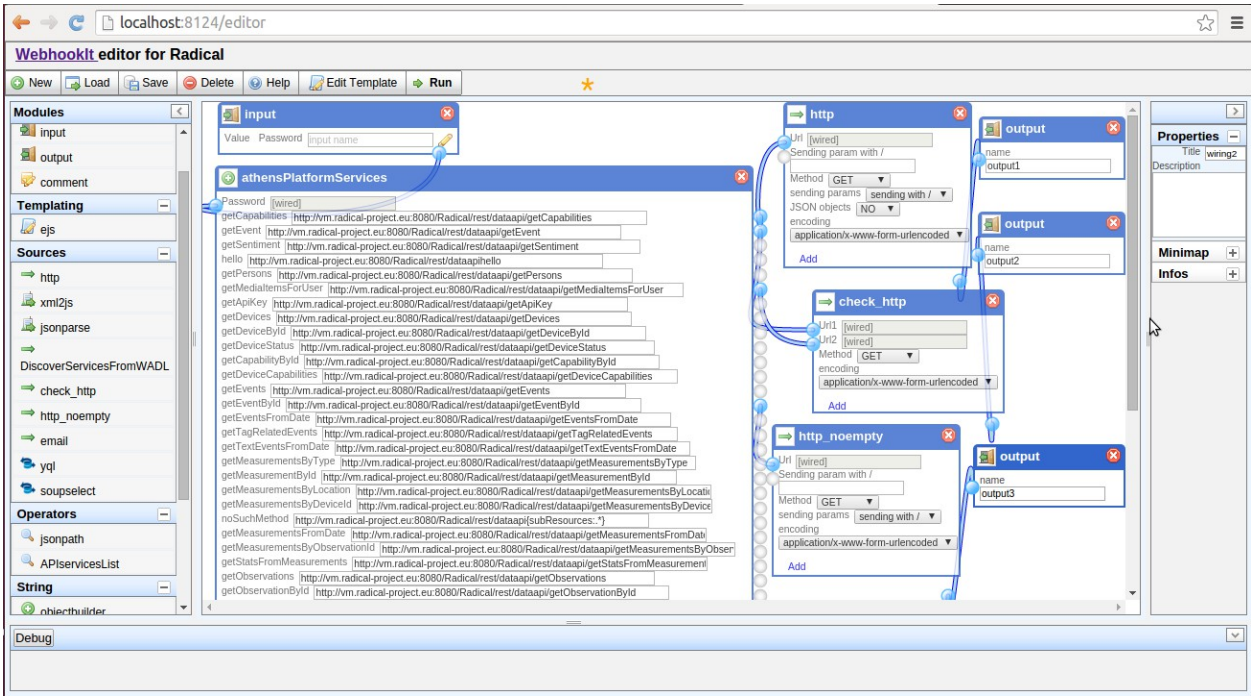
Η μονάδα **DiscoverServicesFromWADL** η οποία όπως περιγράψαμε είναι μια μονάδα που καλωδιώνει όλες τις μονάδες WebHookIt για τη λήψη των διαθέσιμων υπηρεσιών του RADICAL Data API μέσω ενός αρχείου WADL. Κατά την πορεία αυτή παρέχει τη λίστα με τις διαθέσιμες υπηρεσίες σε μια μονάδα εισόδου **<city>PlatformServices** που μπορεί να συνδεθεί με καλωδίωση, με την έννοια ότι ο τελικός χρήστης μπορεί να συνδέσει άμεσα τις διαθέσιμες διευθύνσεις URL υπηρεσιών στις διευθύνσεις URL των REST http κλήσεων http που σκοπεύει να κάνει.

Δίνοντας ως είσοδο ένα WADL και τον κατάλληλο κωδικό της πόλης στην οποία είναι ο χρήστης, όταν η μονάδα εκτελείται μας δίνει όλη τη λίστα με τα URLs των REST διαδικτυακών υπηρεσιών. Στη συγκεκριμένη περίπτωση ο χρήστης είναι από την Αθήνα γι' αυτό και η μονάδα που θα δημιουργηθεί αφού τρέξει και ανανεώσει τη σελίδα θα λέγεται athensPlatformServices.




**Εικόνα 26: Παράδειγμα DiscoverServicesFromWADL**

Εδώ βλέπουμε τη μονάδα που δημιουργήθηκε και πως μπορεί να χρησιμοποιηθεί περαιτέρω σαν μια μονάδα εισόδων για διάφορες άλλες μονάδες αφού πρώτα εισάγει πάλι τον κωδικό της πόλης που βρίσκεται. Όπως βλέπουμε στην εικόνα μπορεί να χρησιμοποιηθεί με τις μονάδες http, http_noempty, check_http και δίνοντας τις κατάλληλες παραμέτρους και ρυθμίσεις να δώσει τα αναμενώμενα αποτελέσματα.




**Εικόνα 27: Παράδειγμα WebHookIt ροής εργασιών με την νέα δημιουργηθείσα μονάδα athensPlatformServices**

## *4.3 Ολοκληρωμένο παράδειγμα σεναρίου χρήσηςRADICAL με τη χρήση του WebHookIt*

Εδώ θα περιγράψουμε με λεπτομέρειες ένα πλήρες σενάριο χρήσης του Περιβάλλοντος Ανάπτυξης. Η υπόθεση που εξετάζεται εδώ είναι για έναν τελικό χρήστη, π.χ. ένας προγραμματιστή της πόλης έχει πρόσβαση στο εργαλείο για να δημιουργήσει μια νέα RADICAL εφαρμογή συνδυάζοντας τη κοινωνική δικτύωση και άλλες βασικές υπηρεσίες (Core Services), χρησιμοποιώντας τις υπηρεσίες του SocIoS API και δεδομένα από τα τοπικά αποθετήρια. Η επιθυμητή λειτουργικότητα της εφαρμογής είναι να πάρει τις πρόσφατες δημοσιεύσεις από τους πιο σημαίνοντες χρήστες του Twitter σε ένα θέμα (π.χ. ανακύκλωση - οικολογική συνείδηση) και να συνδυάσει αυτές τις δημοσιεύσεις με τις υπάρχουσες πληροφορίες για την πόλη (π.χ. Σημεία ανακύκλωσης ) .Σε ο,τι ακολουθεί, θα εξηγήσουμε λεπτομερώς την αλληλουχία των αλληλεπιδράσεων μεταξύ του χρήστη και των διαφόρων στοιχείων που λαμβάνουν χώρα σε αυτό το σενάριο.

Για να δημιουργήσει μια νέα εφαρμογή του τελικού χρήστη (προγραμματιστή της πόλης) θα συνδεθεί με το διαδικτυακό GUI εργαλείο, όπως είναι η διεπαφή του περιβάλλοντος Ανάπτυξης Εφαρμογών WebHookIt. Στη συνέχεια, ο τελικός χρήστης θα πρέπει να βρει τις κατάλληλες υπηρεσίες,των οποίων τις λειτουργίες θα πρέπει να συνδυάσει για να δημιουργήσει την επιθυμητή εφαρμογή. Το παράδειγμα που περιγράφεται παρακάτω μπορεί να αναλυθεί στις ακόλουθες δράσεις / υπηρεσίες:

• Να βασίζεται σε ένα συγκεκριμένο θέμα (π.χ. ανακύκλωση της πόλης), να πάρει τους k πιο σημαίνοντες χρήστες του Twitter για αυτό το θέμα.

• Για κάθε χρήστη, να ανακτήσει τα τελευταία tweets για το συγκεκριμένο θέμα

• Να πάρετε πληροφορίες για την πόλη (π.χ. σημεία ανακύκλωσης από ένα τοπικό αποθετήριο) και να τις συγκεντρώσει μαζί με τις θέσεις.

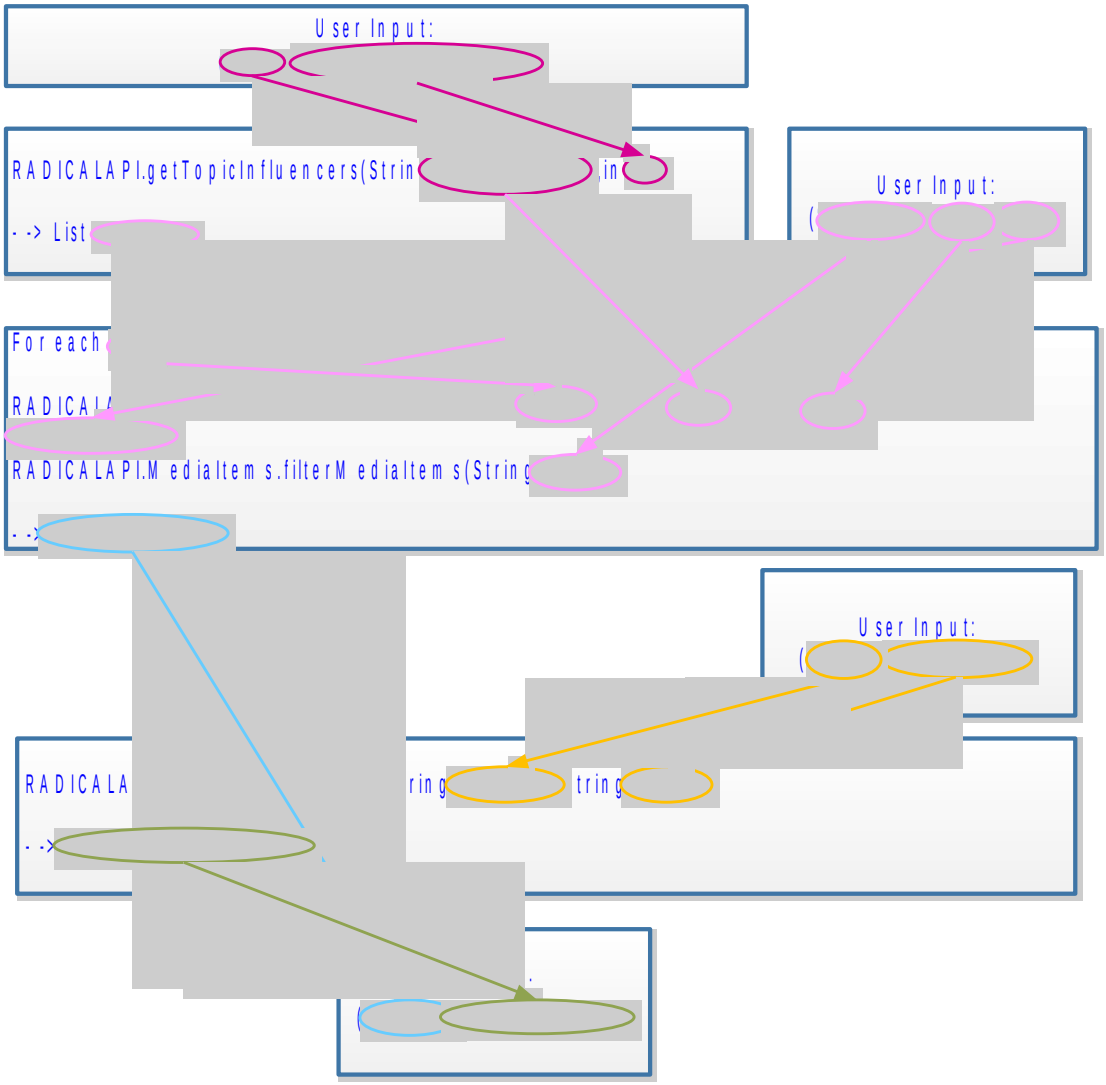
Οι δύο πρώτες δράσεις αφορούν την επεξεργασία SocIoS που σχετίζονται με τα δεδομένα, ενώ η τρίτη δράση ανακτά δεδομένα από ένα εξωτερικό Data Repository Enabler. Όλα θα εκτελεστούν μέσω του RADICAL Data API, όπως το εργαλείο

παρουσιάζει τις διαθέσιμες υπηρεσίες και τις μεθόδους τους για τον τελικό χρήστη με ομοιόμορφο τρόπο, μέσω του μηχανισμού ανακάλυψης υπηρεσιών του. Ο τελικός χρήστης επιλέγει τις υπηρεσίες και τις μεθόδους που θέλει να χρησιμοποιήσει και τις συνδυάζει σε μία ροή εργασιών-καλωδίωση (workflow), σύροντας τις υπηρεσίες στις καλωδιώσεις και αντιστοιχόντας παραμέτρους εισόδου και εξόδου.

Στο προαναφερθέν παράδειγμα, οι μέθοδοι που θα χρησιμοποιηθούν για τις αντίστοιχες δράσεις είναι οι ακόλουθες:

- *RADICALAPI.getTopicInfluencers*
- *RADICALAPI.MediaItems.getMediaItems*, *RADICALAPI.MediaItems.filterMediaItems*
- *RADICALAPI.getLocalInformation*

Στο παρακάτω σχήμα, η αντιστοίχηση των παραμέτρων εισόδου και εξόδου μεταξύ των μεθόδων που ο χρήστης πρέπει να καθορίσει απεικονίζεται:

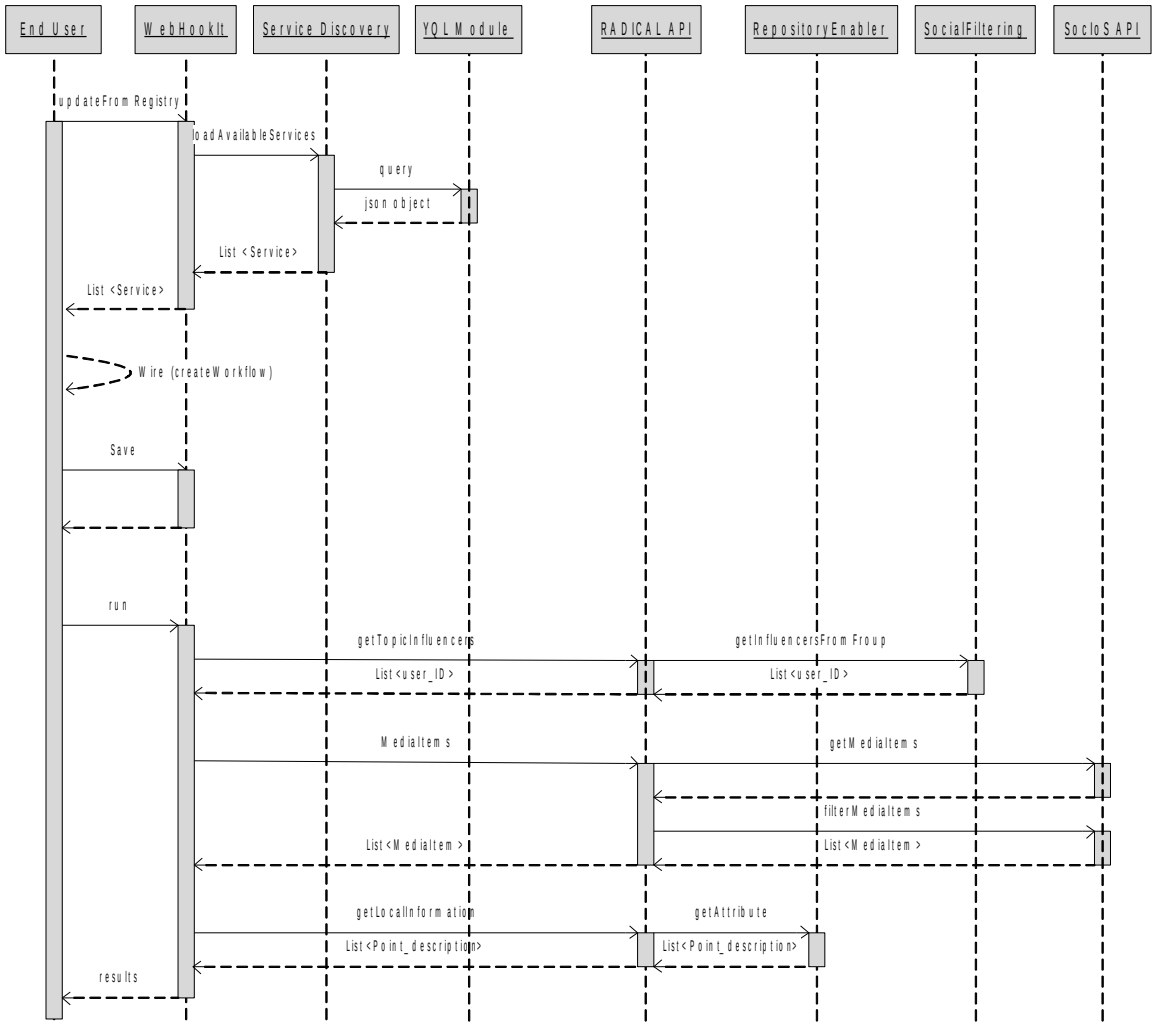



**Εικόνα 28: Αντιστοίχηση παραμέτρων εισόδου και εξόδου**

Αφού ο χρήστης έχει ορίσει την καλωδίωση, μπορεί να επιλέξει να την αποθηκεύσει ως μια νέα μονάδα. Στη συνέχεια, μπορεί να επιλέξει να την εκτελέσει κάνοντας κλικ στο κουμπί Run και παρέχοντας τις απαιτούμενες παραμέτρους (σε αυτή την περίπτωση το θέμα, numberOfInfluencers, text, date, value και RepositoryID). Οι αντίστοιχες υπηρεσίες θα ενεργοποιηθούν διαδοχικά από το εργαλείο και τα αποτελέσματα θα παρουσιαστούν συγκεντρωτικά στον τελικό χρήστη.

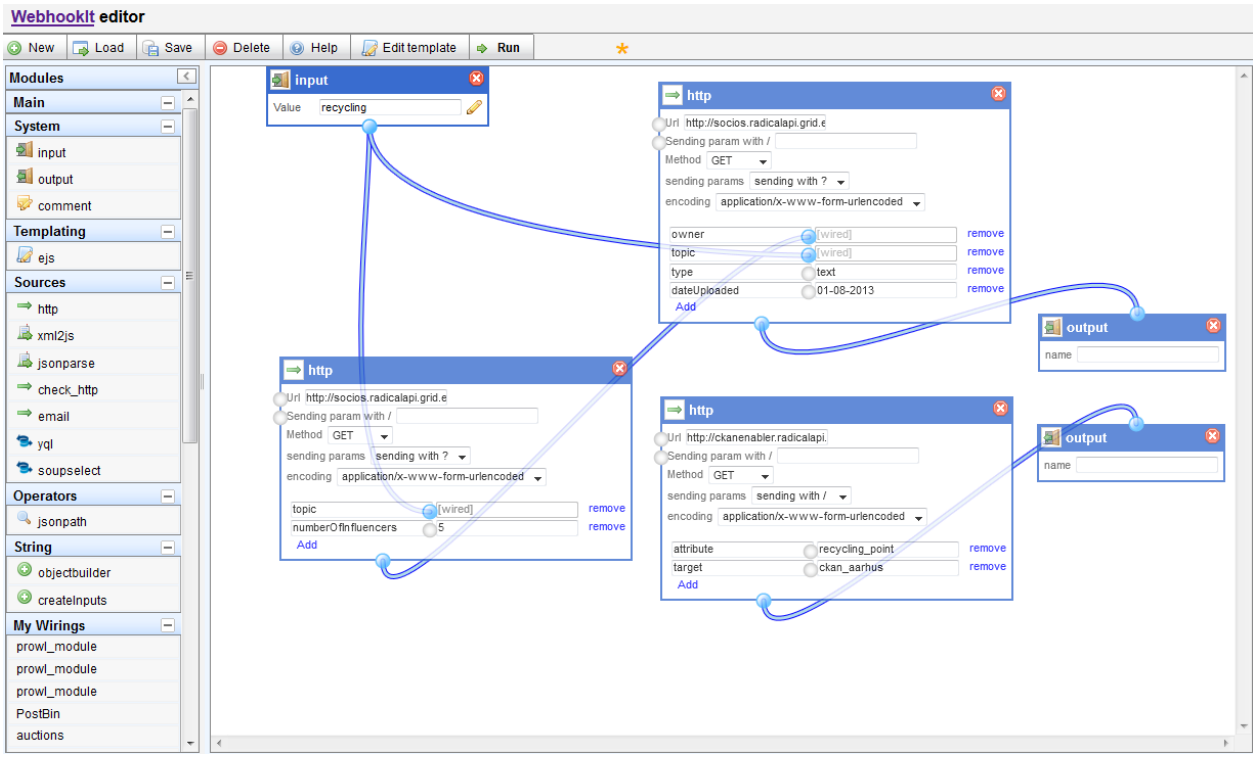
Το ακόλουθο Σχήμα 12 απεικονίζει την αλληλουχία των αλληλεπιδράσεων μεταξύ των μονάδων που συμμετέχουν στο παραπάνω παράδειγμα. Ο χρήστης έχει πρόσβαση στο περιβάλλον Ανάπτυξη Εφαρμογών, μέσω του GUI για να ανακαλύψει τις διαθέσιμες υπηρεσίες, προκειμένου να δημιουργηθεί μια εφαρμογή. Με τη χαρτογράφηση των παραμέτρων εισόδου και εξόδου, όπως εξηγήθηκε παραπάνω, ο

τελικός χρήστης ορίζει την καλωδίωση μιας υπηρεσίας η οποία στη συνέχεια αποθηκεύεται και εκτελείται από το WebHookIt.




**Εικόνα 29: Ακολουθιακό διάγραμμα**

Το κάτω μέρος του ακολουθιακού διαγράμματος λαμβάνει χώρα όταν ο τελικός χρήστης κάνει κλικ στο κουμπί Run , για να ξεκινήσει η εκτέλεση της εφαρμογής που δημιουργήθηκε. Το εργαλείο θα εκτελέσει έπειτα διαδοχικά τις αντίστοιχες μεθόδους και θα επιστρέψει το συνολικό κατάλογο των αποτελεσμάτων για τον τελικό χρήστη. Στον τελικό χρήστη, η δημιουργηθείσα εφαρμογή θα μοιάζει σαν καλωδίωση WebHookIt:




**Εικόνα 30: Τελική WebHookIt καλωδίωση ροής εργασιών**

## *4.4 Επέκταση της βιβλιοθήκης WebHookIt*

Εκτός από τις καινούργιες βασικές μονάδες χρειάστηκε επίσης να τροποποιήσουμε και να προσθέσουμε κώδικα στη βιβλιοθήκη του WebHookIt.

- **simpleflow.js**

Ο κώδικας αυτός δημιουργήθηκε για να τρέξει όλα τα πακέτα των μονάδων έτσι ώστε να φορτώσουν στον WebHookIt Editor. Μια από τις επεκτάσεις που υλοποιήθηκαν για το WebHookIt ήταν ότι κατά την εκτέλεση μια βασικής μονάδας (DiscoverServicesFromWADL)( Να βάλω και την άλλη μονάδα **APIservicesList**;) μπορεί να δημιουργηθεί μια καινούργια βασική μονάδα (<city>PlatformServices) δυνατότητα η οποία δεν υπήρχε προηγουμένως. Ο μόνος τρόπος να επιτευχθεί όμως αυτό ήταν να ξανατρέξει η simpleflow. js και να προστεθούν τότε τα χαρακτηριστικά της καινούργιας μονάδας. Γι' αυτό και για να δημιουργηθούν οι καινούργιες βασικές μονάδες χρειάζεται να ανανεωθεί η σελίδα του editor που ουσιαστικά σημαίνει ότι θα ξανατρέξει η simpleflow.js.

Ο κώδικας που προσθέσαμε παρουσιάζεται παρακάτω.

```
for (var i=0; i<modulesLength; i++) {
        if (config.modules[i].name=="APIservicesList") {
                var pkg = "WebHookIt-APIservicesList";
                var name = pkg.split("-");
                var m = require(pkg);
                definitions.push(m.definition);
                console.log("Definitions should be changed",
                JSON.stringify(m.definition));
                 }
        else if
        (config.modules[i].name=="DiscoverServicesFromWADL"){
        var pkg = "WebHookIt-DiscoverServicesFromWADL";
        var name = pkg.split("-");
```

```
 var m = require(pkg);
definitions.push(m.definition);
 console.log("Definitions should be changed",
JSON.stringify(m.definition));
        }
}
```

- **sessions.js**

Για τις ανάγκες του RADICAL όταν οι χρήστες του WebHookIt εγγράφονται πρέπει να ορίζουν και την πόλη που αντιπροσωπεύουν γιατί ανάλογα με αυτή έχουν ένα συγκεκριμένο Token. Ανάλογα με το Token της κάθε πόλης μπορούν να έχουν πρόσβαση στις διαδικτυακές υπηρεσίες της πόλης τους. Για να επιτευχθεί αυτό έπρεπε να προσθέσουμε στο αρχείο sessions.js τον αντίστοιχο κώδικα ώστε η εφαρμογή να ζητάει από τον χρήστη και την παράμετρο πόλη.

```
app.post('/sessions/signup', function(req, res) {
        User.create({
                name: req.param("username"),
                email: req.param("email"),
                password: req.param("pass"),
                password_confirmation: req.param("pass2"),
                city:req.param("city"),
                created_at: new Date()
        }, function(errors, user) {
                if(errors.length > 0) {
                        req.flash('error', errors);
                        res.redirect('/sessions/signup');
                }
```

```
                else {
                        // log me in !
                        req.session.user_id = user._id;
                        res.redirect('/dashboard');
                }
        }, function(err) {
                req.flash('error', err.message);
                res.redirect('/sessions/signin');
        });

});
```

- **user.js**

Επίσης για να εισάγει ο χρήστης την πόλη του έπρεπε να προσθέσουμε και το πεδίο πόλη "city" στον ορισμό του καινούργιου χρήστη (user).

```
User.fields = {
        _id: {},
        name: {},
        email: {},
        password: {},
        city: {},
        created_at: {},
        debug_runs: {},
        public_runs: {},

        // attribute accessors
        password_confirmation: {}
};
```

- **signup.ejs**

Αυτό το αρχείο παρουσιάζει τη μορφή της σελίδας όταν ένας νέος χρήστης εγγράφεται. Για να μπορεί ο χρήστης να εισάγει την πόλη την οποία αντιπροσωπεύει έπρεπε να προστεθεί και η αντίστοιχη επιλογή.

```
<select name="city">
        <option value="genoa">Genoa</option>
        <option value="athens">Athens</option>
        <option value="cantabria">Cantabria</option>
        <option value="aarhus">Aarhus</option>
        <option value="issy">Issy</option>
        <option value="ganziantep">Ganziantep</option>
      </select>
```

- **layout.ejs**

Σε αυτό το αρχείο προσθέσαμε τον κατάλληλο κώδικα ώστε στο header του WebHookIt να εμφανίζεται το token του κάθε χρήστη καθώς και αλλάξαμε τον τίτλο του WebHookIt σε WebHookIt for RADICAL.

```
<div id="hd">
  <h1>WebHookIt for RADICAL</h1>
<% if(typeof current_user != "undefined") { %>
<%%>
      <span id="username">Welcome, <%= current_user.name %> </br>Access Token:
  <%
  var TokenArray=[{"city":"genoa","token":"genoaToken"},
      {"city":"athens","token":"athensToken"},
      {"city":"cantabria","token":"cantabriaToken"},
      {"city":"aarhus","token":"aarhusToken"},
```

```
        {"city":"issy","token":"issyToken"},
        {"city":"ganziantep","token":"ganziantepToken"}];
    for(var i=0; i<TokenArray.length; i++){
     if (current_user.city==TokenArray[i].city) { %>
      <%=TokenArray[i].token%></span>
    <% }
     } %>
```

- **wirings.js**

Παρατηρήσαμε ότι στον WebHookIt Editor δεν μπορούσαμε να αφαιρέσουμε κάποια δημιουργηθείσα καλωδίωση. Γι' αυτό υπήρξε μια αλλαγή στον κώδικα του αρχείου controllers/wirings.js

```
app.del('/wirings/:id.json',app.require_login, get_wiring_collection,get_wiring, function(req, res){
        console.log("Deleting a wiring with id=",req.wiring._id);

        req.wiring_collection.remove({"_id":req.wiring._id}, function(error){
                if(error) { throw error; }
                res.send(200);
        });
});
```

- **editor.ejs**

Οι αλλαγές σε αυτό το αρχείο ήταν η αλλαγή του τίτλου σε WebHookIt for RADICAL Επίσης στο πεδίο Infos προστέθηκε το κατάλληλο κείμενο:

```
<h2>Infos</h2>
                <div>
                        <div style="padding: 10px;">
```

```
                <p style="font-weight: bold;">Welcome to RADICAL
                Application Development Toolkit! <br> Available
                        WADL APIs:</p>
                <br>
                <p> RADICAL Data API v0.5:
                <br>
            http://147.102.19.128:8080/Radical/rest/application.wadl
                </p>
            </div>
    </div>
```

# 5

# *Επίλογος*

## *5.1 Σύνοψη και συμπεράσματα*

Συνοψίζοντας, βλέπουμε ότι το WebHookIt με τις επεκτάσεις του αποτελεί ένα ιδανικό εργαλείο για την ανάπτυξη ροών εργασιών για RESTful διαδικτυακές υπηρεσίες. Οι RESTful διαδικτυακές υπηρεσίες αποτελούν το μέλλον στο διαδίκτυο και η ανάπτυξη του internet of things καθιστά αυτές καθώς και τον συνδυασμό αυτών σε ροές εργασιών απαραίτητες. Το WebHookIt θεωρείται ένα κατάλληλο εργαλείο για να ανταπεξέλθει σε αυτές τις απαιτήσεις. Είναι ένα εργαλείο ανοιχτού λογισμικού που μπορεί να επεκταθεί και να προσαρμοστεί στις εκάστοτε απαιτήσεις. Μέσω αυτού μπορούν να δημιουργηθούν άπειρες νέες βασικές μονάδες αλλά και από τις ήδη υπάρχουσες να σχεδιαστούν νέες πιο σύνθετες. Αυτό είναι το βασικό στοιχείο που δίνει μεγάλη ευελιξία στους προγραμματιστές. Το γραφικό περιβάλλον επίσης είναι αρκετά φιλικό και εύκολα κατανοητό ως προς τους προγραμματιστές. Τέλος, η δυνατότητα εκτέλεσης των εργασιών περιοδικά μας δίνει έναν παραπάνω λόγο ώστε να το επιλέξουμε ως η καλύτερη λύση για την ανάπτυξη ροών εργασιών.

## *5.2 Μελλοντικές επεκτάσεις*

Εξαιτίας της συνεχούς εξέλιξης και εδραίωσης των διαδικτιακών υπηρεσιών καθώς και λόγω του ανοιχτού κώδικα του το WebHookIt διατίθεται για περεταίρω επεκτάσεις.

Μια επέκταση θα μπορούσε να είναι η δημιουργία νέων βασικών μονάδων από τις ήδη υπάρχουσες μονάδες χωρίς την ανάγκη επαναφόρτωσης της σελίδας του WebHookIt editor. Μέχρι στιγμής ο κώδικας της καινούργιας μονάδας υπάρχει ήδη ως αρχείο και φορτώνεται μόνο όταν η βασική μονάδα που το γεννάει εκτελείται. Μια βελτιστοποίηση επομένων θα ήταν αυτή η βασική μονάδα να παράγεται απευθείας χωρίς να προϋπάρχει ο κώδικας της.

Επιπλέον, σχετικά με το project RADICAL, μέχρι στιγμής υπάρχει περιορισμένος αριθμός πόλεων που έχουν πρόσβαση σε αυτή την πλατφόρμα. Μια άλλη βελτιστοποίηση θα ήταν να μπορούν να εγγράφονται όσες πόλεις χρειάζονται και να δημιουργείται αυτόματα κάποιο token γι' αυτές.

# 6 *Βιβλιογραφία*


[1] Φωτης Α. Αίσωπος. "*Αξιολόγηση ποιότητας εφαρμογών σε υπηρεσιοστρεφείς υποδομές πραγματικού χρόνου*"

[2] Φωτης Α. Αίσωπος. "*Μελέτη, Ανάλυση και Υλοποίηση Γραφικών Διεπαφών για Υποβολή Εργασιών σε Περιβάλλον Πλέγματος*"

[3] Fotis Aisopos,Antonis Litke, Konstantinos Tserpes. *Service Development, Deployment and Integration Capabilities of the RADICAL Platform,* www.radical-project.eu

[4] Alex Rodriguez "*RESTful Web services*"

[5] Richardson, Leonard; Sam Ruby (2007), *RESTful web service*, O'Reilly Media

[6] WebHookIt Documentation, neyric.github.io/WebHookIt/docs

[7] WebHookIt first release, neyric.com/category/projects/webhookit

[8] WireIt, neyric.github.io/wireit/docs

[9] YUI , hyuilibrary.com

[10] inputEx: Field framework for web applications neyric.github.io/inputex/

[11] Bray, Tim. "*JSON Redux AKA RFC7159*" json.org

[12] "*XHTML 1.0 Specification, Section 1: What is XHTML?*". World Wide Web Consortium, www.w3.org/TR/xhtml1/

[13] "*SOAP Specifications*".W3.org, www.w3.org/TR/2007/REC-soap12-

part0-20070427/
[14] Web Services Description Language (WSDL) 1.1 ,www.w3.org/TR/wsdl
[15] Sun Microsystems (31 August 2009). "*Web Application Description Language: W3C Member Submission 31 August 2009*". World Wide Web Consortium, www.w3.org/Submission/wadl/
[16] OASIS UDDI Specifications TC - Committee Specifications
"*Relationship to the World Wide Web and REST Architectures*". *Web Services Architecture*. W3C. http://www.w3.org/TR/ws-arch/
[17] UDDI, uddi.xml.org
[18] mongoDB, www.mongodb.org
[19] NODE.JS, nodejs.org