ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΚΑΙ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σχεδίαση Και Υλοποίηση Εφαρμογής Για Διαλειτουργική Χρήση Δεδομένων Τοποθεσίας Σε Κοινωνικά Δίκτυα.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΟΙΚΟΝΟΜΙΔΗ ΙΩΑΝΝΗ

Επιβλέπων : Ιωάννης Ψαρράς
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2012

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

Σχεδίαση Και Υλοποίηση Εφαρμογής Για Διαλειτουργική Χρήση Δεδομένων Τοποθεσίας Σε Κοινωνικά Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΟΙΚΟΝΟΜΙΔΗ ΙΩΑΝΝΗ

Επιβλέπων : Ιωάννης Ψαρράς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Φεβρουαρίου 1999.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Μέντζας Γεώργιος	Ψαρράς Ιωάννης	Ασκούνης Δημήτριος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2012

(Υπογραφή)

.....

ΟΙΚΟΝΟΜΙΔΗΣ ΙΩΑΝΝΗΣ

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

Περίληψη

Τα μέσα κοινωνικής δικτύωσης (social media) και τα κοινωνικά δίκτυα (social networks) που αναπτύσσονται μέσω του internet κατακτούν με εκπληκτικά αυξανόμενους ρυθμούς ολοένα και περισσότερους χρήστες, κατακλύζοντας έτσι τον παγκόσμιο ιστό. Παράλληλα, οι δυνατότητες των φορητών συσκευών, όπως κινητά τηλέφωνα, PDA's, tablets, laptops και smartphones αυξάνονται συνεχώς. Δημιουργείται λοιπόν η ανάγκη για διευκόλυνση του χρήστη να χρησιμοποιήσει και να συγχρονίσει όλες αυτές τις πληροφορίες, σε όλες τις συσκευές που έχει στην κατοχή του χωρίς ιδιαίτερο κόπο.

Στόχος της παρούσας διπλωματικής εργασίας είναι η υλοποίηση εφαρμογής ανεξάρτητης από την πλατφόρμα, μέσω της οποίας ο χρήστης θα μπορεί να επιλέξει ένα ή και περισσότερα online κοινωνικά δίκτυα εκ των Facebook, Foursquare και Google +, για να δημοσιεύσει και να κοινοποιήσει την τρέχουσα τοποθεσία του. Με τη βοήθεια σύγχρονων προγραμματιστικών εργαλείων, μεθόδων και τεχνολογιών, πραγματοποιείται η σχεδίαση της εφαρμογής και επιτυγχάνεται η υλοποίησή της.

Αρχικά, επιχειρείται η συνοπτική παρουσίαση των κοινωνικών δικτύων που χρησιμοποιήθηκαν από την εφαρμογή. Στη συνέχεια, γίνεται μία αναφορά για τις mobile web εφαρμογές και το Project Management.

Επόμενο βήμα, είναι η εκτεταμένη περιγραφή και παρουσίαση της εφαρμογής συνοδευόμενη από την προδιαγραφή σχεδίασης και απαιτήσεων της και περιγραφή της υλοποίησής της.Τέλος, παρατίθενται ο έλεγχος που εφαρμόστηκε στην εφαρμογή, τα συμπεράσματα που διεξήχθησαν και στόχοι και προοπτικές για μελλοντική εργασία στο συγκεκριμένο θέμα. Στο παράρτημα παρατίθενται εικόνες συσκευών με εγκατεστημένη την εφαρμογή όπως επίσης και το διάγραμμα με όλες τις οθόνες της εφαρμογής.

Λέξεις Κλειδιά: <<Κοινωνικά δίκτυα, εφαρμογή, ανεξάρτητη πλατφόρμα, φορητές συσκευές, δεδομένα τοποθεσίας χρήστη, προδιαγραφές σχεδίασης, προδιαγραφές απαιτήσεων. >>

Abstract

Social media and social networks that are developed through the internet conquer with surprisingly increasing ratio more and more users, thus flushing the web. Furthermore, the possibilities of mobile devices like mobile phones, PDA's, tablets, laptops and smartphones are steadily increasing. Therefore comes, the need for the user to synchronize all this information to all the devices in his possession without much effort.

The aim of this thesis is to implement a platform independent application, through which the user will be able to publish and share his/her current location to Facebook and-or Foursquare and-or Google +. The design and the implementation of this application are achieved using modern developing methods, tools and technologies.

Initially, this analysis attempts to briefly present the social networks that are used in the application. Then we discuss about mobile web applications and Project Management.

In addition, we present extensive description of the implemented application, followed by its software requirements specification (SRS) and its software design documentation (SDD), and the description of its implementation. Finally we discuss about how we managed to test the application, the conclusions we ended up and objectives and prospects for future work on this issue. In the appendix you can find all the code used to develop this application.

Keywords: <<Social networks, application, platform independent, portable devices, user location data, software design documentation, software requirements specification.>>

Ευχαριστίες

Κατ' αρχάς θα ήθελα να ευχαριστήσω τον υπεύθυνο κατά την εκπόνηση της διπλωματικής Καθηγητή της διπλωματικής μου κ. Ι. Ψαρρά για την ανάθεση αυτής και τη δυνατότητα που μου δόθηκε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Επιπροσθέτως θα επιθυμούσα να εκφράσω τις θερμές μου ευχαριστίες στον Αναπληρωτή Καθηγητή κ. Δ. Ασκούνη για την πολύτιμη συμβολή και βοήθειά του στην ολοκλήρωση και στην εκπόνηση της παρούσας εργασίας.

Τέλος θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής Διδάκτορα Γ. Γκιώνη και ερευνητή Ι. Αλβέρτη, για την υποστήριξη και την καθοδήγηση που μου παρείχαν κατά τη συγγραφή της εργασίας.

Πίνακας περιεχομένων

1

Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μία ιστορική αναφορά στην εξέλιξη του WEB από τη δημιουργία του μέχρι σήμερα. Έπειτα παρουσιάζεται το αντικείμενο της διπλωματικής εργασίας, ο στόχος της και η αρχιτεκτονική της.

1.1 Αναδρομή στην εξέλιξη του WEB

Οι εξελίξεις στην τεχνολογία επιτρέπουν πλέον στους ανθρώπους να επικοινωνούν χωρίς περιορισμούς, όπως αρχικά συνέβαινε με την έκδοση εφημερίδων.

Όλα ξεκίνησαν στα τέλη της δεκαετίας του 60', όταν ο οργανισμός ARPA (Advanced Research Projects Agency) στις ΗΠΑ, προσανατολισμένος σε ερευνητικά προγράμματα υψηλής τεχνολογίας, ξεκίνησε μια ερευνητική δραστηριότητα σχετικά με τα δίκτυα μεταγωγής δεδομένων, τα λεγόμενα Packet Switched Networks. Το 1970 καταγράφηκε για πρώτη φορά η λέξη Internet, όταν αναπτύχθηκε και το πρωτόκολλο επικοινωνίας TCP/IP το οποίο έγινε το αποκλειστικό πρωτόκολλο επικοινωνίας του ARPAnet.

Το 1971 έγινε η αποστολή του πρώτου email και ταυτόχρονα θεωρείται το πρώτο δείγμα κοινωνικής δικτύωσης. Το 1983 το ARPAnet χωρίσθηκε σε δύο κομμάτια, το στρατιωτικό MILNET με αυστηρά ελεγχόμενη πρόσβαση και το ακαδημαϊκό ARPAnet με πρόσβαση από ένα μεγαλύτερο κοινό. Στις αρχές της δεκαετίας του 90' ο Tim Berners Lee ανέπτυξε τη γλώσσα HTML, η οποία βασίζεται στο hypertext και σχεδίασε τον παγκόσμιο ιστό (World Wide Web) στο Ερευνητικό Κέντρο Φυσικής CERN, έξω από τη Γενεύη. Πρόκειται για ένα σύστημα διασύνδεσης πληροφοριών σε μορφή πολυμέσων (multimedia) που βρίσκονται αποθηκευμένες σε χιλιάδες υπολογιστές του Internet σε ολόκληρο τον κόσμο και παρουσίασής τους σε ηλεκτρονικές σελίδες, στις οποίες μπορεί να περιηγηθεί κανείς

χρησιμοποιώντας το ποντίκι. Παρόλα αυτά, μέχρι το 1991 το Internet ήταν περιορισμένο σε ερευνητική, εκπαιδευτική και κυβερνητική χρήση έως ότου η άρση του περιορισμού στην εμπορική χρήση του δικτύου έκανε το Internet να αναπτυχθεί με εκθετικούς ρυθμούς.

Ιδιαίτερο ενδιαφέρον όμως, αποτελεί και η άνθιση των κινητών συσκευών από το 1990 μέχρι σήμερα. Η πρώτη στροφή από τα κινητά τηλέφωνα στα smartphones, παρουσιάστηκε τη δεκαετία του 90' και αποδείχθηκε ότι οι κινητές συσκευές θα μπορούσαν να έχουν δυνατότητες περισσότερες από ένα απλό τηλέφωνο. Αφορμή για αυτήν τη στροφή ήταν ο ερχομός των PDA's (personal digital assistants), των QWERTY πληκτρολογίων, της διαχείρισης των επαφών (contact manager) και των εφαρμογών ημερολογίων (calendar applications) στον πολλά υποσχόμενο κόσμο των κινητών συσκευών. Από το 2000 έως σήμερα, η έγχρωμη εικόνα, η λήψη φωτογραφιών και βίντεο, οι υπηρεσίες που προσφέρονται από το Παγκόσμιο Σύστημα Τοποθεσίας (Global Positioning System-GPS) και η πρόσβαση στο Internet μέσω WiFi έχουν γίνει απαραίτητες προϋποθέσεις για αγορά κινητού, ενώ η εμφάνιση των tablets, συσκευές που στεγάζουν όλα τα στοιχεία του υλικού τους (hardware) σε επίπεδες οθόνες αφής, προκάλεσε σημαντικές μεταβολές στο σύνολο της αγοράς των υπολογιστών, σύμφωνα με έρευνα αγοράς της Gartner Inc και της IDC. Αξίζει να αναφερθεί ότι τα επικρατέστερα και πιο σύγχρονα λειτουργικά συστήματα για κινητές συσκευές είναι τα iPhone και Android.

Πλέον, το μεγαλύτερο μέρος της ανθρωπότητας ζει σε χώρες που είναι συνδεδεμένες στο Internet. Είναι προφανές ότι πλέον το Internet δεν αποτελεί ένα δίκτυο των φοιτητών και των ερευνητών, αλλά ότι επεκτείνεται και επιδρά στις καθημερινές πρακτικές όλων μας. Η ανάγκη του ανθρώπου για κοινωνικότητα και διάδραση με τους κοντινούς και μη ανθρώπους τον έκανε να αναζητεί τρόπους επικοινωνίας. Το Internet λοιπόν δεν μπορούσε παρά να αποτελέσει το πιο διαδεδομένο κοινωνικό μέσο στην εποχή μας.

Σήμερα, η επικοινωνία μέσω διαδικτύου δεν έχει μείνει στα emails. Τα online κοινωνικά δίκτυα παρουσιάζουν ακμάζουσα δημοτικότητα και ο ερχομός τους υπήρξε ένα από τα πιο συναρπαστικά γεγονότα της τελευταίας δεκαετίας, καθώς έχουν καταφέρει να προσελκύσουν εκατομμύρια χρήστες. Πολλά τέτοια ανεπίσημα και μη δίκτυα εμφανίστηκαν μέσα σε λίγα χρόνια. Κάποια από αυτά είναι τα Xanga, Bebo, Friendster, Tagged, Imbee, LiveJournal, MySpace, Foursquare και τα γνωστά στην πλειοψηφία της ανθρωπότητας Twitter, Google + και Facebook.

1.2 Αντικείμενο διπλωματικής

Η εργασία αυτή έχει σκοπό να δείξει πώς θα μπορούσαμε να εκμεταλλευτούμε τα παραπάνω επιτεύγματα της τεχνολογίας και κατά κάποιον τρόπο να τα συνδυάσουμε, υλοποιώντας μία

απλή εφαρμογή. Μία εφαρμογή ανεξάρτητης πλατφόρμας, που θα ενοποιεί κάποιες κοινές λειτουργίες ορισμένων κοινωνικών δικτύων προσφέροντας μία ξεχωριστή εμπειρία χρήσης.

1.3 Οργάνωση κειμένου

Το υπόλοιπο της εργασίας είναι δομημένο ως εξής: Στο 2ο Κεφάλαιο αναπτύσσεται το θεωρητικό υπόβαθρο που χρησιμοποιήθηκε για την ολοκλήρωση της εργασίας. Στο 3ο Κεφάλαιο, γίνεται μια παρουσίαση των προδιαγραφών απαιτήσεων της εφαρμογής. Στο 4ο Κεφάλαιο παρουσιάζονται αναλυτικά η αρχιτεκτονική του συστήματος και οι κύριες οθόνες της εφαρμογής. Τα κομμάτια με το μεγαλύτερο τεχνικό και αλγοριθμικό ενδιαφέρον της υλοποίησης της εφαρμογής παρατίθενται στο κεφάλαιο 5. Στο Κεφάλαιο 6 γίνεται μία αναφορά σχετικά με τον έλεγχο της εφαρμογής και τον τρόπο που διεξήχθησαν στατιστικά χρήσης της. Τα συμπεράσματα της εργασίας και οι προοπτικές για περαιτέρω εργασία στο θέμα παρατίθενται στο 7ο Κεφάλαιο. Τέλος, στο Παράρτημα παρουσιάζονται εικόνες από συσκευές με εγκατεστημένη την εφαρμογή αλλά και το mockup της εφαρμογής με όλες τις οθόνες της.

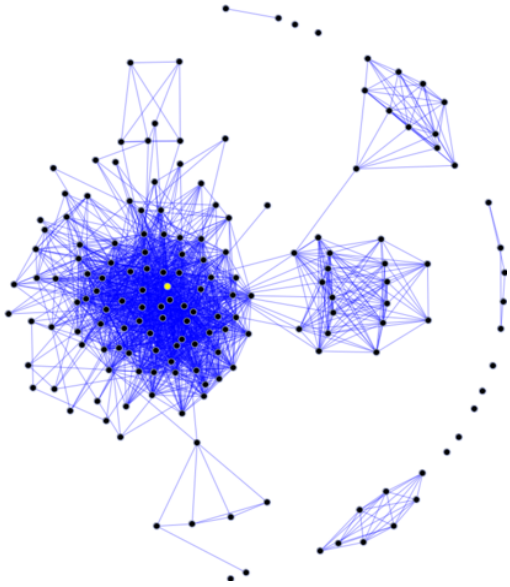
2

Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο γίνεται μία εισαγωγή στα κοινωνικά δίκτυα και αναλύονται περαιτέρω αυτά που χρησιμοποιήθηκαν για την εργασία. Ακόμη γίνεται μια αναφορά για τις mobile web εφαρμογές και τα πλεονεκτήματά τους. Συζητούνται επίσης οι τεχνικές διοίκησης έργων λογισμικού και περιγράφεται αναλυτικά η μεθοδολογία που χρησιμοποιήθηκε στην εργασία.

2.1 Κοινωνικά Δίκτυα

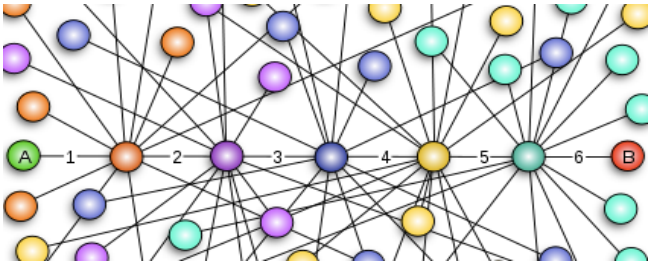
Ένα κοινωνικό δίκτυο είναι μία κοινωνική δομή που συμπεριλαμβάνει κόμβους οι οποίοι συνδέονται με έναν ή περισσότερους τύπους αλληλεξάρτησης, όπως αξίες, οράματα, στόχοι, ιδέες, οικονομικές αλλαγές, φιλία. Οι κόμβοι είναι τα άτομα που απαρτίζουν ένα κοινωνικό δίκτυο και οι μεταξύ τους σχέσεις αναπαρίστανται με γραμμές (εικόνα 2.1).



Εικόνα 2.1: 'Social Network Analysis, Wikipedia'

Τα άτομα αυτά λοιπόν μπορεί να συνδέονται με άλλα άτομα άμεσα ή έμμεσα, αλλά και να επικοινωνούν με άτομα που δε γνωρίζουν αλλά έχουν κοινά ενδιαφέροντα. Σύμφωνα μάλιστα με μία ενδιαφέρουσα έρευνα, υποστηρίζεται ότι οποιοσδήποτε άνθρωπος στον κόσμο απέχει από οποιονδήποτε άλλον κατά μέσο όρο έξι μόλις βήματα.

Το φαινόμενο αυτό διατυπώθηκε από τον Milgram τη δεκαετία του 60’ ως «6 degrees of separation» και αναφέρεται και ως «small world phenomenon» (εικόνα 2.2).



Εικόνα 2.2: ‘Six Degrees of Separation, Wikipedia’

Τέτοια κοινωνικά δίκτυα συναντάμε στο διαδίκτυο αλλά και σε κινητές συσκευές. Πρόκειται για διαδικτυακές υπηρεσίες (Social Media) που επιτρέπουν στους χρήστες να δημιουργήσουν ένα Profile[1] καθώς και μία λίστα από άλλους χρήστες με τους οποίους έχουν κάποια σχέση. Σύμφωνα με το Wikipedia, ως Social Media ορίζονται τα μέσα που έχουν σχεδιαστεί ώστε να διαχέονται μέσω της κοινωνικής αλληλεπίδρασης, με υψηλή προσβασιμότητα και με τεχνικές δημοσίευσης που μπορεί να μεταβάλλονται. Χρησιμοποιούν το Internet και τεχνολογίες στηριζόμενες στο web ώστε να ενισχύσουν τη δημοκρατικοποίηση της γνώσης και της πληροφορίας, να δώσουν την ευκαιρία στο χρήστη όχι μόνο να «καταναλώνει» αλλά και να «παράγει» περιεχόμενα. Στην εικόνα 2.3, παρουσιάζεται μία κατηγοριοποίηση των social media με βάση τις υπηρεσίες που προσφέρουν.

Εικόνα 2.3: ‘Social Media Landscape, www.minabard.com’

[1] Με τον όρο Profile στα κοινωνικά δίκτυα, εννοούμε τη δυνατότητα του χρήστη να συλλέξει και να περιγράψει τα προσωπικά του δεδομένα, όπως ονοματεπώνυμο, ηλικία, φωτογραφία, ημερομηνία γέννησης κ.ά.

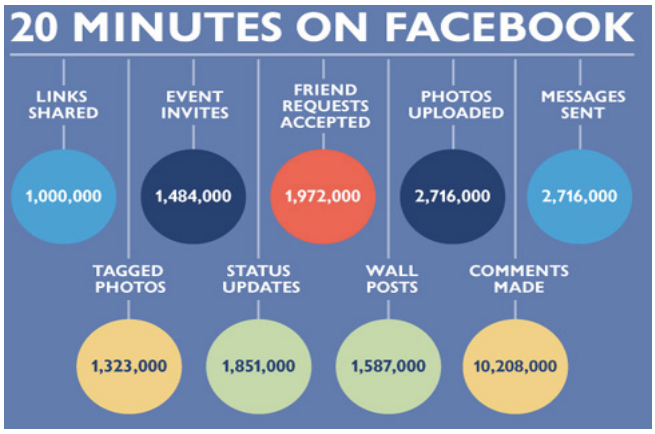
Τα κριτήρια που έκαναν τα διαδικτυακά κοινωνικά μέσα να ξεχωρίσουν από τα παραδοσιακά μέσα ενημέρωσης, είναι η εύκολη και οικονομική προσβασιμότητα από όλους, η δυνατότητα όλων να κοινοποιήσουν μια πληροφορία, καθώς επίσης και η επικοινωνία και ενημέρωση πραγματικού χρόνου. Παρακάτω θα παρουσιάσουμε τα κοινωνικά δίκτυα που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

2.1.1 Facebook

Το Facebook είναι ιστοχώρος κοινωνικής δικτύωσης που ξεκίνησε στις 4 Φεβρουαρίου του 2004. Οι χρήστες μπορούν να επικοινωνούν μέσω μηνυμάτων με τις επαφές τους και να τους ειδοποιούν όταν ανανεώνουν τις προσωπικές πληροφορίες τους. Όλοι έχουν ελεύθερη πρόσβαση στο να συμμετάσχουν σε δίκτυα που σχετίζονται μέσω πανεπιστημίου, θέσεων απασχόλησης ή γεωγραφικών περιοχών. Έχει σημειώσει τεράστια οικονομική επιτυχία κάνοντας το δημιουργό του, έναν από τους πιο πλούσιους developers παγκοσμίως στα δύο πρώτα χρόνια της λειτουργίας του.

Ο Mark Zuckerberg ίδρυσε το Facebook ως μέλος του πανεπιστημίου του Harvard. Αρχικά δικαίωμα συμμετοχής είχαν μόνο οι φοιτητές του Harvard. Το όνομα της ιστοσελίδας προέρχεται από τα έγγραφα παρουσίασης των μελών πανεπιστημιακών κοινοτήτων μερικών Αμερικάνικων κολεγίων και προπαρασκευαστικών σχολείων που χρησιμοποιούσαν οι νεοεισερχόμενοι σπουδαστές για να γνωριστούν μεταξύ τους. Το 2005 το δικαίωμα πρόσβασης επεκτάθηκε σε μαθητές συγκεκριμένων λυκείων και μέλη ορισμένων μαθητικών κοινοτήτων, ενώ το 2006 η υπηρεσία έγινε προσβάσιμη σε κάθε άνθρωπο στο πλανήτη που η ηλικία του ξεπερνούσε τα 13 χρόνια.

Το Facebook σήμερα αριθμεί περισσότερους από 800 εκατομμύρια ενεργούς χρήστες και κατατάσσεται ως ένα από τα δημοφιλέστερα web sites του πλανήτη σύμφωνα με τη λίστα του www.alexa.com ($2^{ο}$ μετά το Google). Επίσης, το Facebook είναι από τα δημοφιλέστερα sites για ανέβασμα φωτογραφιών με πάνω από 2.5 εκατομμύρια φωτογραφίες ανά είκοσι λεπτά (εικόνα 1.4). Με αφορμή τη δημοτικότητά του, το Facebook έχει υποστεί κριτική και κατηγορηθεί σε θέματα που αφορούν τα προσωπικά δεδομένα και τις πολιτικές απόψεις των ιδρυτών του. Ωστόσο η συγκεκριμένη ιστοσελίδα παραμένει η πιο διάσημη κοινωνική περιοχή δικτύωσης σε πολλές αγγλόφωνες χώρες.



Εικόνα 2.4: 'Facebook Stats, http://www.digitalbuzzblog.com/'

2.1.2 Google Plus

Η Google είναι μια από τις μεγαλύτερες εταιρείες διαδικτυακών υπηρεσιών. Η λειτουργία της ξεκίνησε στις 27 Σεπτεμβρίου του 1998. Η ονομασία της προήλθε από αναγραμματισμό της λέξης Googol, η οποία εκφράζει μαθηματικό όρο και σημαίνει το «1 ακολουθούμενο από 100 μηδενικά». Το 1996 οι Larry Page και Sergey Brin έκαναν μία κολεγιακή εργασία στο Stanford University για μια μηχανή αναζήτησης, η οποία τελικά κατέληξε να κυριαρχήσει στο Internet. Πλέον η Google είναι η πιο δημοφιλής μηχανή αναζήτησης και μάλιστα αυτό επιβεβαιώνεται καθώς η λέξη google έχει εισχωρήσει ως ρήμα, στην καθομιλουμένη όχι μόνο των Αμερικανών αλλά και των Ευρωπαίων.

Το 2011 η Google ανακοίνωσε τη νέα πλατφόρμα κοινωνικής δικτύωσης που ονομάζεται Google Plus. Αν και συγκριτικά με το Facebook, η Google είχε αργήσει, τον Ιανουάριο του 2012, η Google Plus είχε ήδη 90 εκατομμύρια ενεργούς χρήστες. Μάλιστα, η Google Plus, έκανε «ρεκόρ ταχύτητας» στον «αγώνα δρόμου» για τους 50 εκατομμύρια χρήστες[2]. Ο χρόνος της ήταν μόλις 88 μέρες (εικόνα 2.5).

[2] http://www.oneclickcustomers.com/infographics/google-vs-facebook-infographic.html

Εικόνα 2. 5: 'Αγώνας Δρόμου, http://www.oneclickcustomers.com'

Το Google Plus για να κερδίσει τον ανταγωνισμό, θα έπρεπε όχι μόνο να προσφέρει στους χρήστες τις ήδη γνωστές από το Facebook δυνατότητες, αλλά να συμπεριλάβει και κάτι καινοτόμο. Έτσι λοιπόν, ενσωμάτωσε τις επαφές σε κύκλους, που ορίζει ο ίδιος ο χρήστης. Με αυτόν τον τρόπο, ο χρήστης μοιράζεται πληροφορίες με συγκεκριμένους κύκλους της επιλογής του και έτσι ο βασικός δομικός λίθος της πλατφόρμας, υλοποιεί αυτό που πολλοί έλεγαν ότι λείπει από το Facebook (εικόνα 2.6).

Εικόνα 2.6: 'Google Plus Circles, http://www.scribbal.com'

Τέλος, μία ακόμη καινοτομία της Google που έκανε την πλατφόρμα της να ξεχωρίσει, είναι η ιδέα των Hangouts. Τα Hangouts είναι η απάντηση στο μαζικό Video Conferencing. Ο χρήστης μπορεί να αρχίσει ένα Hangout, και να επιτρέψει σε κάποιους άλλους χρήστες να συμμετέχουν σε αυτό, οπότε ενεργοποιείται αυτόματα το μικρόφωνο και η κάμερα του υπολογιστή. Αργότερα, ανά πάσα στιγμή και όσο ο χρήστης συμμετέχει σε αυτό το Hangout, οι προσκεκλημένοι μπορούν να συμμετάσχουν.

Ιδιαίτερο ενδιαφέρον θα έχει η εξέλιξη της 'μάχης' αυτής η οποία είναι πολύ νωρίς ακόμα για να πει κάποιος ότι έχει τελειώσει.

2.1.3 Foursquare

Το Foursquare είναι ένας κοινωνικός ιστότοπος, ο οποίος βασίζεται στην τρέχουσα τοποθεσία του χρήστη (location-based social networking site) και χρησιμοποιείται σε κινητές συσκευές. Ο χρήστης μπορεί να κοινοποιήσει στους υπόλοιπους χρήστες με τους οποίους είναι συνδεδεμένος την τρέχουσα τοποθεσία του κάνοντας «check-in» στα μέρη που επισκέπτεται, είτε μέσα από φυλλομετρητή ή εφαρμογή κινητής συσκευής, είτε μέσω sms. Μπορεί να ενημερώσει τους υπόλοιπους χρήστες για ένα μέρος επισυνάπτοντας σχόλια σε αυτό, όπως για παράδειγμα μία κριτική σε ένα εστιατόριο, και να φτιάχνει λίστες με τα μέρη που θέλει να επισκεφτεί. Ο χρήστης κερδίζει πόντους για τα check-ins που πραγματοποιεί και κάποιες φορές και βραβεία ή αλλιώς badges.

Η υπηρεσία αυτή δημιουργήθηκε από τους Dennis Crowley και Naveen Selvadurai το 2009. Ο Crowley, είχε ήδη ασχοληθεί με παρόμοιο project (Dodgeball) στη διπλωματική του εργασία στο New York University, το οποίο και αγοράστηκε από την Google το 2005 και το 2009 αντικαταστάθηκε από την υπηρεσία Google Latitude. Το Foursquare λοιπόν αποτέλεσε μία δεύτερη προσπάθεια για υλοποίηση της ίδιας ιδέας. Μέχρι τον Ιούνιο του 2011, η υπηρεσία αριθμούσε 10 εκατομμύρια ενεργούς χρήστες. Τα check-ins σε καθημερινό επίπεδο φτάνουν τα 3 εκατομμύρια κατά μέσο όρο, ενώ συνολικά υπολογίζεται ότι ο αριθμός τους ξεπερνά τα 750 εκατομμύρια[3].

Μπορεί το Foursquare, να μην είναι τόσο δημοφιλές όσο τα υπόλοιπα, αλλά αποτέλεσε τη βάση της ιδέας για τη διπλωματική αυτή εργασία.

Εικόνα 2. 7: 'Το λογότυπο του Foursquare, www.foursquare.com'

[3] http://en.wikipedia.org/wiki/Foursquare_(website)

2.2 Mobile Web Applications

Με τον όρο Mobile Web περιγράφονται υπηρεσίες όπως το World Wide Web (σύστημα υπερσυνδέσμων στο οποίο επιτυγχάνεται πρόσβαση μέσω διαδικτύου) και WAP (συσκευή η οποία επιτρέπει ασύρματη επικοινωνία) οι οποίες είναι βασισμένες σε κάποιον web browser και οι οποίες χρησιμοποιούν κάποια φορητή συσκευή μικρής οθόνης όπως είναι τα κινητά τηλέφωνα, τα PDA's ή οποιαδήποτε άλλη φορητή συσκευή που είναι συνδεδεμένη σε δημόσιο δίκτυο. Κατά συνέπεια, ο χρήστης αποκτά πρόσβαση στο διαδίκτυο από οποιοδήποτε σημείο κι αν βρίσκεται, χωρίς να απαιτείται υπολογιστής γραφείου (desktop computer) ή προκαθορισμένη σύνδεση με καλώδια.

Αποκτά συνεπώς ο χρήστης το πλεονέκτημα της άμεσης πρόσβασης στο διαδίκτυο από το κινητό του τηλέφωνο με αποτέλεσμα να αυξάνεται η διαλειτουργικότητα και η χρηστικότητα. Ωστόσο, ο χρήστης του Mobile Web δεν έχει συνήθως τα ίδια ενδιαφέροντα και τις ίδιες ανάγκες με το χρήστη του κλασικού Web. Ενώ ο χρήστης του κλασικού Web μπορεί να ενδιαφέρεται να περιηγηθεί στο διαδίκτυο, να αναζητήσει μακροσκελείς πληροφορίες και να ανατρέξει σε διαφημιστικά παράθυρα, ο χρήστης του Mobile Web προτιμά να προσπελαύνει ιστοσελίδες που του παρέχουν χρήσιμες πληροφορίες όπως τα δρομολόγια των λεωφορείων, την πρόγνωση του καιρού ή την ανταλλαγή e-mail μηνυμάτων. Αυτό δε σημαίνει φυσικά πως στο Mobile Web εκλείπουν τα διαφημιστικά παράθυρα. Απεναντίας οι διαφημιστές θεωρούν το κινητό τηλέφωνο ως ένα ιδανικό μέσο για την προώθηση των προϊόντων τους αφού ο αριθμός των χρηστών του είναι τεράστιος.

Το σημαντικότερο πλεονέκτημα των web εφαρμογών είναι ότι ο προγραμματιστής δεν χρειάζεται κάποιο συγκεκριμένο SDK και μπορεί να χρησιμοποιήσει web τεχνολογίες όπως HTML, CSS και JavaScript. Ακόμη, ένα web app μπορεί να καλύψει την πλειοψηφία των συσκευών και των φυλλομετρητών ανεξαρτήτως κατασκευαστή και λειτουργικού συστήματος. Τέλος, ακόμα ένα πλεονέκτημα μπορεί να θεωρηθεί η ανεξάρτητη διανομή, αφού η εφαρμογή «ζει» στο web και ο χρήστης συνδέεται με αυτήν χρησιμοποιώντας το φυλλομετρητή του. Το σημαντικότερο μειονέκτημα έναντι των native εφαρμογών είναι ότι συνήθως καθυστερούν περισσότερο λόγω της υποχρεωτικής «εξάρτησης» που έχουν με το web.

2.3 Project Management

Τα έργα από την αρχή του πολιτισμού, υπήρξαν μέρος της ανθρώπινης καθημερινότητας. Ωστόσο η πρακτική διοίκησή τους απέκτησε την αρμόζουσα σημασία, μόλις τις δύο τελευταίες δεκαετίες, και αυτό γιατί σχετίζεται με τη διαχείριση, ανθρώπινων και μη, πόρων.

Ο ανταγωνισμός του σήμερα, έχει αναγκάσει τα έργα να εξαρτώνται άμεσα από την έγκαιρη και εντός των δαπανών του προϋπολογισμού πραγματοποίηση των στόχων.

Ο τομέας λοιπόν που πραγματεύεται τεχνικές, μεθοδολογίες, πρακτικές και εργαλεία για την συστηματική και μεθοδική προδιαγραφή, σχεδίαση, υλοποίηση, έλεγχο και συντήρηση συστημάτων λογισμικού υψηλής ποιότητας και εντός δεδομένου προϋπολογισμού και χρόνου εκτέλεσης ονομάζεται τεχνολογία λογισμικού (Software Engineering - IEEE Standard 610.12). Οι φάσεις από τις οποίες διέρχεται μια εφαρμογή λογισμικού από τη σύλληψή της μέχρι και την απόσυρσή της, ονομάζεται κύκλος ζωής λογισμικού (Software Life Cycle). Οι βασικές φάσεις κύκλου ζωής είναι οι ακόλουθες[4]:

- Ανάλυση Απαιτήσεων (Requirements)

  Η ανάλυση απαιτήσεων απαντά σε βασικές ερωτήσεις, με την αναλυτική διερεύνηση του προβλήματος. Καταγράφονται οι περιορισμοί που δημιουργούνται από τα πλαίσια του προϋπολογισμού όπως και το τι χρειάζεται και τι θέλει ο πελάτης. Με αυτόν τον τρόπο διασφαλίζεται η ποιότητα του λογισμικού. Η διαδικασία θεωρείται εύλογη οικονομικά.

- Καθορισμός Προδιαγραφών (Specification)

  Ο καθορισμός προδιαγραφών απαιτεί συμπλήρωση εγγράφων που έχουν νομική υπόσταση και καθορίζουν επακριβώς το πρόβλημα. Οι προδιαγραφές δεν πρέπει να είναι διφορούμενες, ημιτελείς ή αντιφατικές. Μετά τη συμπλήρωση των σχετικών εγγράφων γίνεται αναλυτικός σχεδιασμός παραγωγής και υπολογισμός του κόστους του έργου. Με αυτόν τον τρόπο δημιουργείται ένα πλάνο (Software Product Management Plan – SPMP) με τα παραδοτέα και τα βασικά χρονικά σημεία των παραδοτέων (deliverables).

- Σχεδίαση (Design)

  Η βασική διαφορά στον καθορισμό προδιαγραφών και στο σχεδιασμό είναι ότι στην πρώτη περίπτωση απαντάμε στο ερώτημα «τι πρόκειται να υλοποιηθεί», ενώ στη δεύτερη περίπτωση στο «πώς θα υλοποιηθεί». Επιλέγονται έτσι οι αλγόριθμοι, οι δομές δεδομένων και οι δομικές ενότητες (modules) του συστήματος και καταγράφονται οι σχετικές αποφάσεις.

- Υλοποίηση (Implementation)

  Η υλοποίηση του έργου λογισμικού, δεν περιλαμβάνει μόνο τη συγγραφή κώδικα. Ο έλεγχος της φάσης αυτής με test cases, με ανασκόπηση κώδικα (code review) κ.ά. κρίνεται απαραίτητος.

[4] http://www.mech.upatras.gr/~nikos/mis-ii/notes/notes-01.pdf

- Ενοποίηση και Εγκατάσταση (Integration)

  Στη ενοποίηση και εγκατάσταση του κώδικα, έχουμε συνένωση των δομικών ενοτήτων του λογισμικού και ελέγχεται η σωστή λειτουργία του προϊόντος ως ενιαία οντότητα. Επίσης καταμετράται η ευρωστία και η ευχρηστία του συστήματος και διορθώνεται με ανατροφοδότηση.

- Συντήρηση (Maintenance)

  Αυτή η φάση περιλαμβάνει οποιαδήποτε αλλαγή στο σύστημα αφότου ο πελάτης έχει παραλάβει το προϊόν λογισμικού. Αν και κοστίζει περισσότερο από το άθροισμα των υπόλοιπων φάσεων, πρέπει πάντα να θεωρείται αναπόσπαστο τμήμα της διαδικασίας ανάπτυξης λογισμικού.

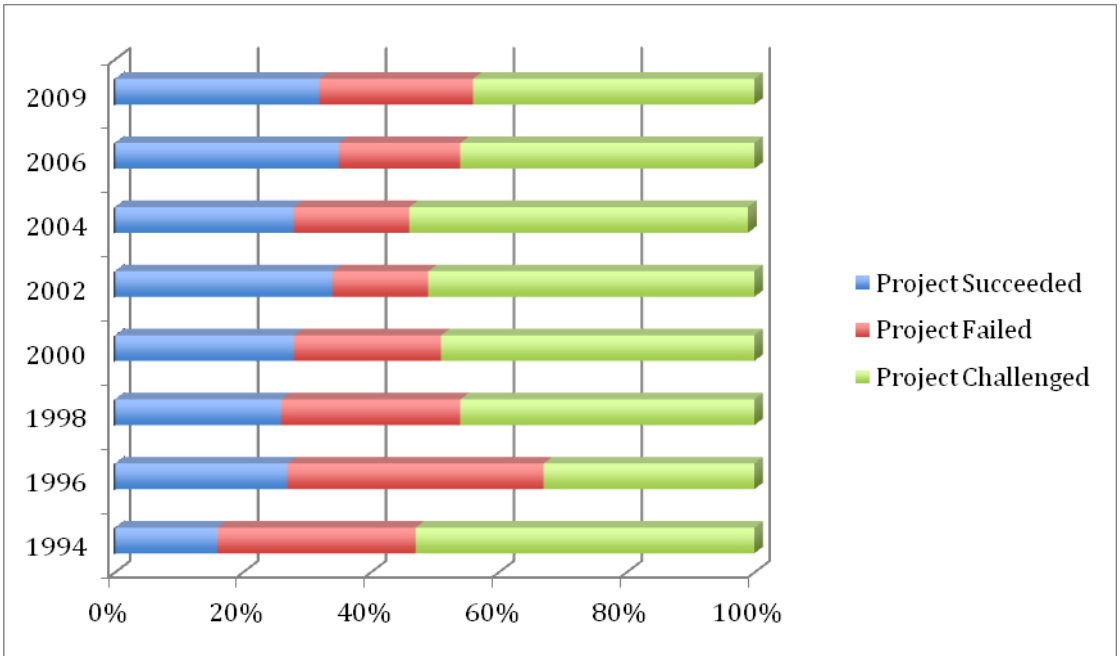
- Αποχώρηση (Retirement)

  Στην αποχώρηση, οδηγούνται συνήθως λογισμικά όταν έχουν ανύπαρκτη, ελλιπή ή ανακριβή τεκμηρίωση, ή όταν απαιτούνται δραστικές αλλαγές, ασύμφορες ή αδύνατες.

Οι φάσεις του κύκλου ζωής ενός λογισμικού μπορούν να οριστούν με μεγάλη σαφήνεια και φαίνεται να καθορίζουν πλήρως τον τρόπο με τον οποίο ένα έργο λογισμικού μπορεί να διοικηθεί και να αναπτυχθεί. Στην πραγματικότητα όμως το ποσοστό των επιτυχημένων έργων λογισμικού, επομένως και η διοικητική και αναπτυξιακή μεθοδολογία, έχουν ακόμα σημαντικά περιθώρια βελτίωσης. Σύμφωνα με το CHAOS Study και το CHAOS Summary, (έρευνες που διεξάγονται από το Standish Group International) υπάρχουν τρεις κατηγορίες έργων λογισμικού:

- Project Succeeded (Επιτυχημένα έργα-On Budget και On Time και On Specs)
- Project Challenged (Not On Budget και/ή Not On time)
- Project Failed ή Impaired (Αποτυχημένα έργα-ποτέ δεν τελείωσαν, ακυρώθηκαν)

Τα αποτελέσματα των ερευνών, φαίνονται στο παρακάτω χρονοδιάγραμμα σε δείγμα περισσότερων από 70.000 έργων από το 1992.



Εικόνα 2. 8: 'Χρονοδιάγραμμα έργων, CHAOS Study and CHAOS Summary'[5]

2.4 *Lean Software Development*

Το παραδοσιακό Project Management βασίζεται κυρίως στο μοντέλο καταρράκτη (Waterfall) ή σε αυτό που ονομάζουμε Predictive Project Management. Γνωρίζοντας όλες τις απαιτήσεις του έργου προγραμματίζουμε πολύ νωρίς στην αρχή και μετά εκτελούμε το πλάνο μας κάνοντας τις απαραίτητες διορθώσεις. Ο προϋπολογισμός και το φυσικό αντικείμενο του έργου είναι γνωστά εξαρχής. Η μέθοδος αυτή αποτελεί για πολλούς την κυριότερη αιτία για την αποτυχία ενός έργου.

Στην απέναντι πλευρά έχουμε έργα που ξεκινούν με άγνωστο φυσικό αντικείμενο χωρίς ουσιαστικά να γνωρίζουμε το τελικό προϊόν του έργου, όπως για παράδειγμα τα ερευνητικά έργα. Σε αυτήν την περίπτωση η δημιουργία αρχικού πλάνου καθίσταται αδύνατη και ο προϋπολογισμός του έργου εγκρίνεται σταδιακά. Υπάρχει λίστα προτεραιοτήτων και το προϊόν ενδέχεται να έχει πολλές εκδόσεις. Στην προσπάθεια τους λοιπόν να δώσουν μια απάντηση στο ερώτημα γιατί τα έργα λογισμικού αποτυγχάνουν, το 2003 οι Mary και Tom Poppendieck δημοσίευσαν το βιβλίο τους, 'Lean Software Development: An Agile Toolkit for Software Development Managers'. Η ευέλικτη προσέγγιση που παρουσίασαν, εστιάζει στη δημιουργία λογισμικού με υψηλό βαθμό ανοχής σε αλλαγές και οι 7 αρχές της είναι οι ακόλουθες:

[5] http://www.stpsoft.co.uk/stpsoft_general2.html

- Εξάλειψη των αχρείαστων (Eliminate Waste)
- Δημιουργία ποιοτικών κατασκευών (Build Quality In)
- Δημιουργία γνώσης (Create Knowledge)
- Αναβολή δέσμευσης (Defer Commitment)
- Γρήγορη παράδοση (Deliver Fast)
- Σεβασμός στους ανθρώπους (Respect People)
- Ολική βελτιστοποίηση (Optimize the Whole)

Στην ανάπτυξη και υλοποίηση του έργου που δημιουργήθηκε στην παρούσα διπλωματική εργασία χρησιμοποιήθηκαν οι αρχές του Lean Software Development και παρακάτω θα γίνει μια σύντομη περιγραφή τους[6].

- Εξάλειψη των αχρείαστων

Η κατηγορία αυτή έχει να κάνει με οτιδήποτε καταναλώνει χρόνο χωρίς να προσθέτει κάποια αξία στο έργο.

- Οι βλάβες-ελαττώματα (Defects)

Η διαφορά με το παραδοσιακό Project Management και το Lean, έγκειται στο ότι δεν περιμένουμε να βγουν οι βλάβες στην επιφάνεια για να τις λύσουμε, αλλά τις προλαμβάνουμε ακόμα και αν αυτό μας στοιχίσει περισσότερο χρόνο. Επίσης, όταν βρεθεί μία βλάβη πρέπει να γίνουν οι κατάλληλες ενέργειες που θα εντοπίσουν την πηγή της βλάβης και θα εξασφαλίσουν ότι η συγκεκριμένη βλάβη δε θα ξαναεμφανιστεί από τη συγκεκριμένη πηγή. Αυτό, στο Software Development μπορεί να εφαρμοσθεί με τη δημιουργία αυτοματοποιημένων tests, τα οποία δημιουργούνται από τους προγραμματιστές για να τεστάρουν ένα συγκεκριμένο κομμάτι κώδικα και μπορούν να τρέξουν αυτόματα με χρήση μιας εντολής. Με αυτόν τον τρόπο αν κάποιος αλλάξει τον κώδικα χωρίς να έχει γνώση ότι κάτι τέτοιο θα προκαλέσει βλάβη, όταν έρθει η ώρα να τρέξουν τα tests, η αποτυχία τους θα δείξει όχι μόνο την ύπαρξη βλάβης, αλλά και το σημείο εκείνο του κώδικα που τη δημιούργησε, ή αλλιώς την πηγή.

- Υπερπαραγωγή-Επιπλέον λειτουργικότητες (Overproduction-Extra Features)

Κάθε γραμμή κώδικα κοστίζει λεφτά και χρόνο, και για αυτό δε χρειάζεται να υλοποιούμε λειτουργικότητες πέρα από αυτές που πραγματικά χρειάζεται ο πελάτης. Ο κανόνας 80/20 που λέει ότι το 80% των πραγματικών αναγκών του πελάτη ικανοποιείται από το 20% των λειτουργικοτήτων ενός έργου λογισμικού, εφαρμόζεται στα περισσότερα προϊόντα λογισμικού. Αξίζει να δώσουμε σε αυτό το σημείο κάποια στατιστικά στοιχεία, που αποδεικνύουν ότι σε 400 έργα λογισμικού που εξετάστηκαν σε περίοδο 15 χρόνων,

[6] Curt Hibbs, Steve Jewett, Mike Sullivan: 'The Art Of Lean Software Development'

λιγότερο από το 5% του κώδικα ήταν τελικά χρήσιμο[7] και το 45% των λειτουργικοτήτων δε χρησιμοποιείται ποτέ[8].

- Μεταφορές-Μεταβιβάσεις (Transportation-Handoffs)

Ένα τυπικό σφάλμα που συμβαίνει στο μοντέλο καταρράκτη, είναι οι πολύ συχνές μεταβιβάσεις που γίνονται. Για παράδειγμα, οι αναλυτές αρχικά δίνουν στους αρχιτέκτονες όλες τις απαιτήσεις του έργου, οι οποίοι με τη σειρά τους το σχεδιάζουν και μεταβιβάζουν τη δουλειά τους στους προγραμματιστές. Η γνώση που χάνεται σε κάθε μεταφορά υπολογίζεται στο 50% και για αυτόν το λόγο σύμφωνα με το Lean οι μεταβιβάσεις πρέπει να αποφεύγονται όποτε αυτό είναι εφικτό.

- Αναμονή-Καθυστερήσεις (Waiting-Delays)

Οι καθυστερήσεις εμφανίζονται τη στιγμή που πρέπει να παρθεί μια απόφαση και ο προγραμματιστής πρέπει να ρωτήσει ή να συμβουλευτεί είτε τον πελάτη είτε συναδέλφους του οι οποίοι δεν είναι διαθέσιμοι.

- Αποθέματα-Μερικώς ολοκληρωμένα έργα (Inventory-Partially Completed Work)

Οτιδήποτε δηλαδή έχει αρχίσει χωρίς να έχει τελειώσει. Όπως για παράδειγμα, προδιαγραφές απαιτήσεων που δεν έχουν κωδικοποιηθεί, κώδικας που δεν έχει τεσταριστεί, τεκμηριωθεί και δημοσιοποιηθεί, και σφάλματα που δεν έχουν διορθωθεί. Αυτές είναι και οι προϋποθέσεις για το Lean Software Development, που χαρακτηρίζουν ένα έργο ολοκληρωμένο.

- Κίνηση-Εναλλαγή δραστηριοτήτων (Motion-Task Switching)

Η εναλλαγή δραστηριοτήτων σκοτώνει την παραγωγικότητα. Ο developer καταναλώνει αρκετό χρόνο να συγκεντρωθεί σε μία δραστηριότητα ώσπου να κατανοήσει το πρόβλημα και να αρχίσει τη διαδικασία επίλυσής του.

- Υπερεπεξεργασία-Περιττές διαδικασίες (Overprocessing-Unneeded Processes)

Σε αυτήν την κατηγορία περιλαμβάνονται χειρονακτικές εργασίες που θα μπορούσαν να αυτοματοποιηθούν, διαδικασίες που δεν πετυχαίνουν τίποτα και έγγραφα που δε διαβάζονται ποτέ.

- Δημιουργία ποιοτικών κατασκευών

---

[7] IEEE 26th Software Engineering Workshop, 'Improving Software Investments through Requirements Validation', 2001

[8] XP 2002 Conference, Standish Group

Ο έλεγχος ποιότητας του προϊόντος δεν πρέπει να αφήνεται για το τέλος της γραμμής παραγωγής γιατί τότε θα είναι αργά για βελτιώσεις. Στα μεγάλα έργα λογισμικού η ποιότητα ελέγχεται με την ανάπτυξη μιας σουίτας tests που αυτόματα τρέχουν κάθε φορά που αλλάζει ο κώδικας του έργου και έτσι οι εξαρτήσεις που δημιουργούνται μεταξύ διαφορετικών σημείων στον κώδικα δε δυσκολεύουν πολύ τη διόρθωση-βελτίωση του κώδικα. Η μέθοδος Test Driven Development ή αλλιώς TDD προσφέρει μεγαλύτερη αυτοπεποίθηση στον προγραμματιστή για τον κώδικα που γράφει και κάνει ευκολότερη και ασφαλέστερη τη διαδικασία διόρθωσης του κώδικα, εξασφαλίζοντας έτσι στο προϊόν περισσότερη ποιότητα. Πιο συγκεκριμένα, η μέθοδος αυτή στηρίζει την άποψη ότι για κάθε μικρό κομμάτι κώδικα που γράφεται στο έργο, θα πρέπει να έχει πρώτα δημιουργηθεί ένα test από τον ίδιο τον προγραμματιστή, που θα κάνει κάποιες προσδοκίες-υποθέσεις (expectations-assertions). Το κομμάτι κώδικα αυτό, θα ολοκληρωθεί μόνο όταν περάσει το test.

- Δημιουργία γνώσης

Σύμφωνα με το Lean Development μία ομάδα προγραμματιστών που δουλεύουν σε ένα έργο λογισμικού θα πρέπει να μοιράζονται τα προβλήματα-εμπόδια που αντιμετωπίζουν έτσι ώστε την επόμενη φορά που το ίδιο πρόβλημα εμφανιστεί, ο χρόνος αντιμετώπισής του να είναι σημαντικά μικρότερος. Ο σχολιασμός σε ένα 'δύσκολο' κομμάτι κώδικα, η συγγραφή documentation και τα stand-ups είναι οι επικρατέστερες μορφές για την επικοινωνία αυτή των προγραμματιστών.

- Αναβολή δέσμευσης

Οι καλύτερες αποφάσεις παίρνονται τη στιγμή που υπάρχει η περισσότερη πληροφορία διαθέσιμη και για αυτόν το λόγο, το Lean προτείνει την αναμονή μέχρι την τελευταία στιγμή για μία μη αναστρέψιμη απόφαση. Για παράδειγμα, εάν υπάρχουν δύο διαφορετικοί τρόποι να αναπτυχθεί μία λειτουργικότητα χωρίς κάποιος από αυτούς να έχει σαφές πλεονέκτημα, τότε το καλύτερο θα είναι (αν βέβαια αυτό είναι οικονομικά και χρονικά εφικτό) να αναπτυχθούν και οι δύο παράλληλα και στο τέλος να επιλεχθεί αυτός που ταιριάζει καλύτερα στο έργο λογισμικού. Αυτή τη προσέγγιση είχε χρησιμοποιήσει και η εταιρία Toyota, όταν έβγαλε στην κυκλοφορία το μοντέλο Toyota Prius. Οι απαιτήσεις για το αυτοκίνητο αυτό δεν καθόριζαν έναν υβριδικό κινητήρα και έτσι σχεδιάστηκαν ταυτόχρονα Prius με διαφορετικούς κινητήρες. Η απόφαση για τον υβριδικό κινητήρα πάρθηκε την τελευταία στιγμή, που αρκούσε για να φέρει η Toyota το Prius στην παραγωγή.

- Γρήγορη παράδοση

Η ανάπτυξη μιας λειτουργικότητας σε ένα έργο λογισμικού μπορεί να διασπαστεί σε μικρές παρτίδες που παραδίδονται σύντομα το ένα μετά το άλλο στον πελάτη. Αυτό σημαίνει ότι ο πελάτης μπορεί να έχει γρήγορα μία πρώτη εικόνα και εγκαίρως να αλλάξει τις απαιτήσεις του έργου πριν αυτές υλοποιηθούν, αν αυτό είναι αναγκαίο. Κάθε παραδοτέο, δίνει την ευκαιρία να επανατοποθετηθούν οι προτεραιότητες στις απαιτήσεις, μέσα από

πραγματική χρήση και ανατροφοδότηση. Το τελικό αποτέλεσμα είναι ένα προϊόν που ικανοποιεί περισσότερο τις πραγματικές ανάγκες του πελάτη.

- Σεβασμός στους ανθρώπους

Το Lean Software Development υποστηρίζει την εμπιστοσύνη απέναντι στους εργαζόμενους, τη συμμετοχή τους στην έκθεση ελλείψεων στην τρέχουσα διαδικασία, την ενθάρρυνσή τους να βρουν τρόπους να βελτιωθούν, τη δυνατότητα να συμβουλεύουν και την αναγνώριση της δουλειάς τους. Άλλωστε, όπως είπαν και οι Mary και Tom Poppendieck, το πλέον αειφόρο ανταγωνιστικό πλεονέκτημα είναι οι σκεπτόμενοι άνθρωποι.

- Ολική βελτιστοποίηση

Αυτό είναι ένα σημαντικό μέρος της Lean μεθοδολογίας, ανεξάρτητα από το πού εφαρμόζεται. Η συνεχής βελτιστοποίηση μιας μόνο διαδικασίας στην ανάπτυξη ενός έργου γίνεται σε βάρος των υπολοίπων διαδικασιών και για αυτόν το λόγο θα πρέπει να συμπεριλαμβάνονται όσες περισσότερες διαδικασίες γίνεται όταν έρχεται η ώρα για βελτιστοποίηση.

3

Προδιαγραφές Απαιτήσεων

Στο πρώτο μέρος του κεφαλαίου αυτού παρουσιάζουμε τα σενάρια χρήσης της εφαρμογής. Στο δεύτερο μέρος του μελετάμε τα διαγράμματα χρήσης, τα ακολουθιακά και συνεργατικά διαγράμματα και το σχεσιακό διάγραμμα της βάσης δεδομένων. Οι προδιαγραφές απαιτήσεων θα μας βοηθήσουν αρκετά να κατλάβουμε τον τρόπο και το τι ακριβώς θέλουμε να υλοποιήσουμε.

3.1 Περιγραφή χρήσης

3.1.1 Σενάριο 1. Simple Checkin

Ο χρήστης κάνει log in σε κάποιο social network, επιλέγει κάποιο μέρος από αυτό και κάνει checkin δημοσιεύοντας την τρέχουσα τοποθεσία του στο social network που επέλεξε.

- Ο χρήστης εισέρχεται στην εφαρμογή.
- Ο χρήστης κάνει log in σε ένα social network.
- Ο χρήστης μεταφέρεται σε μία σελίδα στην οποία του παρουσιάζονται μέρη από το social network που επέλεξε, σε ακτίνα 500 μέτρων.
- Ο χρήστης επιλέγει κάποιο μέρος και μεταφέρεται σε μια σελίδα με περισσότερες πληροφορίες για το μέρος που επέλεξε.
- Σε αυτήν τη σελίδα ο χρήστης μπορεί να δει το μέρος που επέλεξε και την τρέχουσα τοποθεσία του στο χάρτη.

- Ο χρήστης πατάει το checkin button και επιτρέπει τη δημοσίευση της τρέχουσας τοποθεσίας του στο social network στο οποίο «ανήκει» το μέρος που έχει επιλέξει.

3.1.2 Σενάριο 2. Mixed Checkin

Ο χρήστης κάνει log in σε κάποια social networks και επιλέγει κάποιο μέρος από το mixed tab. Σε αυτό το tab εμφανίζονται τα κοινά μέρη των διαφορετικών social networks που έχει συνδεθεί ο χρήστης. Ο χρήστης μπορεί να κάνει checkin στο μέρος που επέλεξε και να δημοσιεύσει την τρέχουσα τοποθεσία του σε όλα τα social networks που το μέρος ανήκει.

- Ο χρήστης εισέρχεται στην εφαρμογή.
- Ο χρήστης κάνει log in σε τουλάχιστον 2 social networks.
- Ο χρήστης μεταφέρεται σε μία σελίδα με ένα ειδικό tab στο οποίο υπάρχουν κοινά μέρη από τα social networks που επέλεξε.
- Ο χρήστης επιλέγει κάποιο μέρος από αυτό το tab και μεταφέρεται σε μια σελίδα με περισσότερες πληροφορίες για το μέρος που επέλεξε.
- Σε αυτήν τη σελίδα ο χρήστης μπορεί να δει το μέρος που επέλεξε και την τρέχουσα τοποθεσία του στο χάρτη.
- Ο χρήστης πατάει το checkin button και επιτρέπει τη δημοσίευση της τρέχουσας τοποθεσίας του στα social networks στα οποία «ανήκει» το κοινό μέρος που έχει επιλέξει.

3.1.3 Σενάριο 3. Multiple Checkin

Ο χρήστης κάνει log in σε κάποια social networks και επιλέγει ένα μέρος από κάθε social network. Ο χρήστης μπορεί να κάνει checkin στα μέρη που επέλεξε και να πραγματοποιήσει πολλαπλά checkins σε διαφορετικά networks και σε διαφορετικά μέρη.

- Ο χρήστης εισέρχεται στην εφαρμογή.
- Ο χρήστης κάνει log in σε τουλάχιστον 2 social networks.
- Ο χρήστης μεταφέρεται σε μία σελίδα στην οποία του παρουσιάζονται μέρη από τα social networks που επέλεξε, σε ακτίνα 500 μέτρων σε διαφορετικά tabs.
- Ο χρήστης επιλέγει ένα μέρος από κάθε tab και με ένα κουμπί μπορεί να προχωρήσει στο επόμενο βήμα και να επιτρέψει τη δημοσίευση της τοποθεσίας του στα αντίστοιχα social networks.

3.1.4 Σενάριο 4. Help the Community

Ο χρήστης κάνει log in σε κάποια social networks και επιλέγει ένα μέρος από κάθε social network. Ο χρήστης έχει τη δυνατότητα να «βοηθήσει» την κοινότητα υποδεικνύοντάς της ότι τα μέρη που έχει επιλέξει είναι στην πραγματικότητα το ίδιο μέρος. Πατώντας ένα κουμπί αποστέλλει αυτήν την πληροφορία στους Servers της εφαρμογής, οι οποίοι με τη σειρά τους «σερβίρουν» το μέρος αυτό στο mixed tab.

- Ο χρήστης εισέρχεται στην εφαρμογή.
- Ο χρήστης κάνει log in σε τουλάχιστον 2 social networks.
- Ο χρήστης μεταφέρεται σε μία σελίδα στην οποία του παρουσιάζονται μέρη από τα social networks που επέλεξε, σε ακτίνα 500 μέτρων σε διαφορετικά tabs.
- Ο χρήστης επιλέγει ένα μέρος από κάθε tab και επιλέγει να «βοηθήσει» την κοινότητα.
- Η ενέργεια αυτή του χρήστη θα ενοποιήσει τα μέρη από τα διαφορετικά social networks και θα τα παρουσιάζει σαν ένα στο mixed tab που αναφέρθηκε στο σενάριο 2.

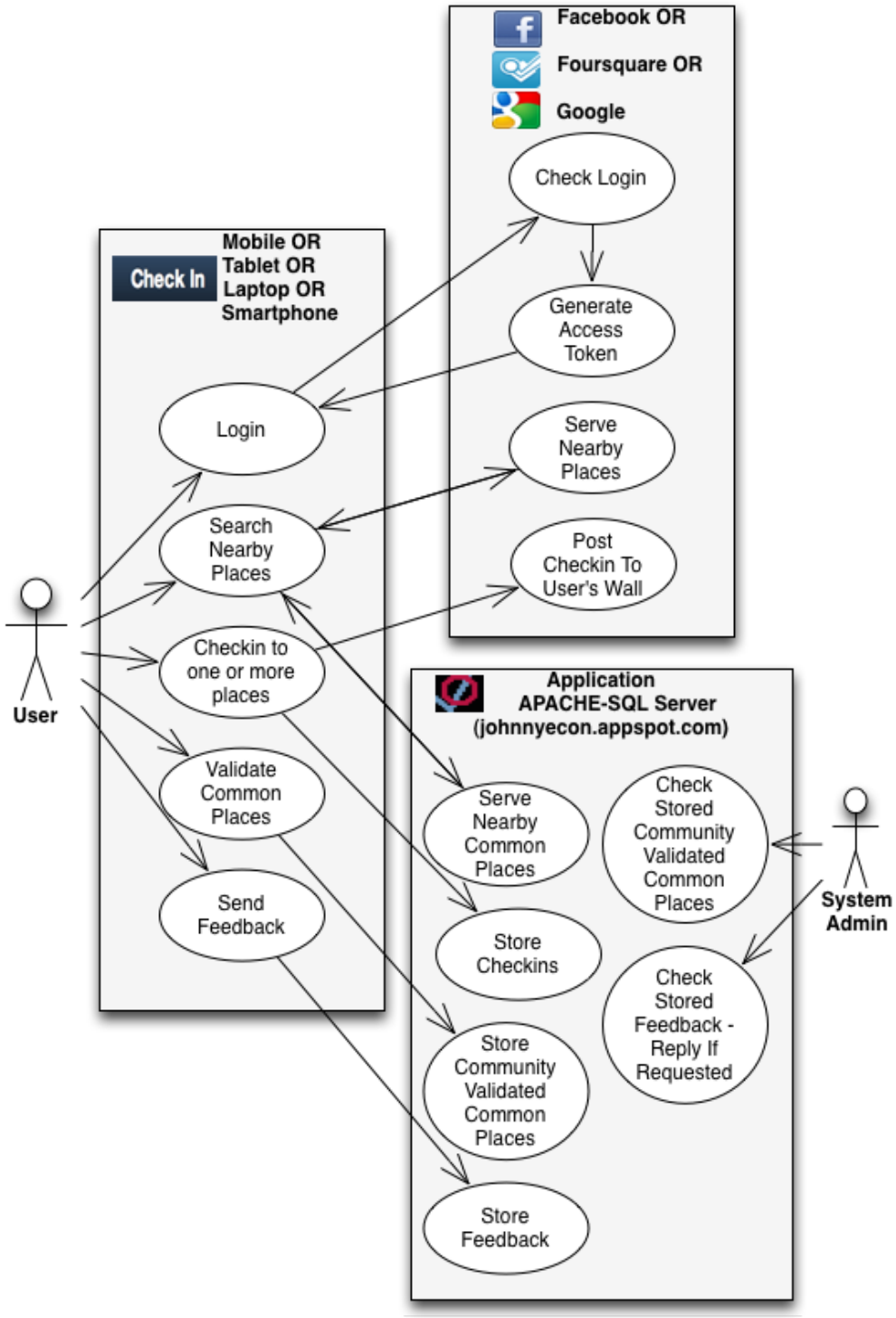
3.1.5 Σενάριο 5. Feedback

Ο χρήστης κάνει log in σε τουλάχιστον ένα social network και επιλέγει να στείλει ένα μήνυμα στους διαχειριστές της εφαρμογής, συμπληρώνοντας μία φόρμα με το email του, τον τίτλο και το περιεχόμενο του μηνύματος αυτού.

- Ο χρήστης εισέρχεται στην εφαρμογή.
- Ο χρήστης κάνει log in σε τουλάχιστον 1 social network.
- Ο χρήστης μεταφέρεται σε μία σελίδα στην οποία του παρουσιάζονται μέρη από το social network που επέλεξε, σε ακτίνα 500 μέτρων.
- Ο χρήστης επιλέγει να στείλε ένα feedback στους διαχειριστές της εφαρμογής με ένα απλό κουμπί και τη συμπλήρωση μιας φόρμας (τίτλος μηνύματος, μήνυμα, email επικοινωνίας).

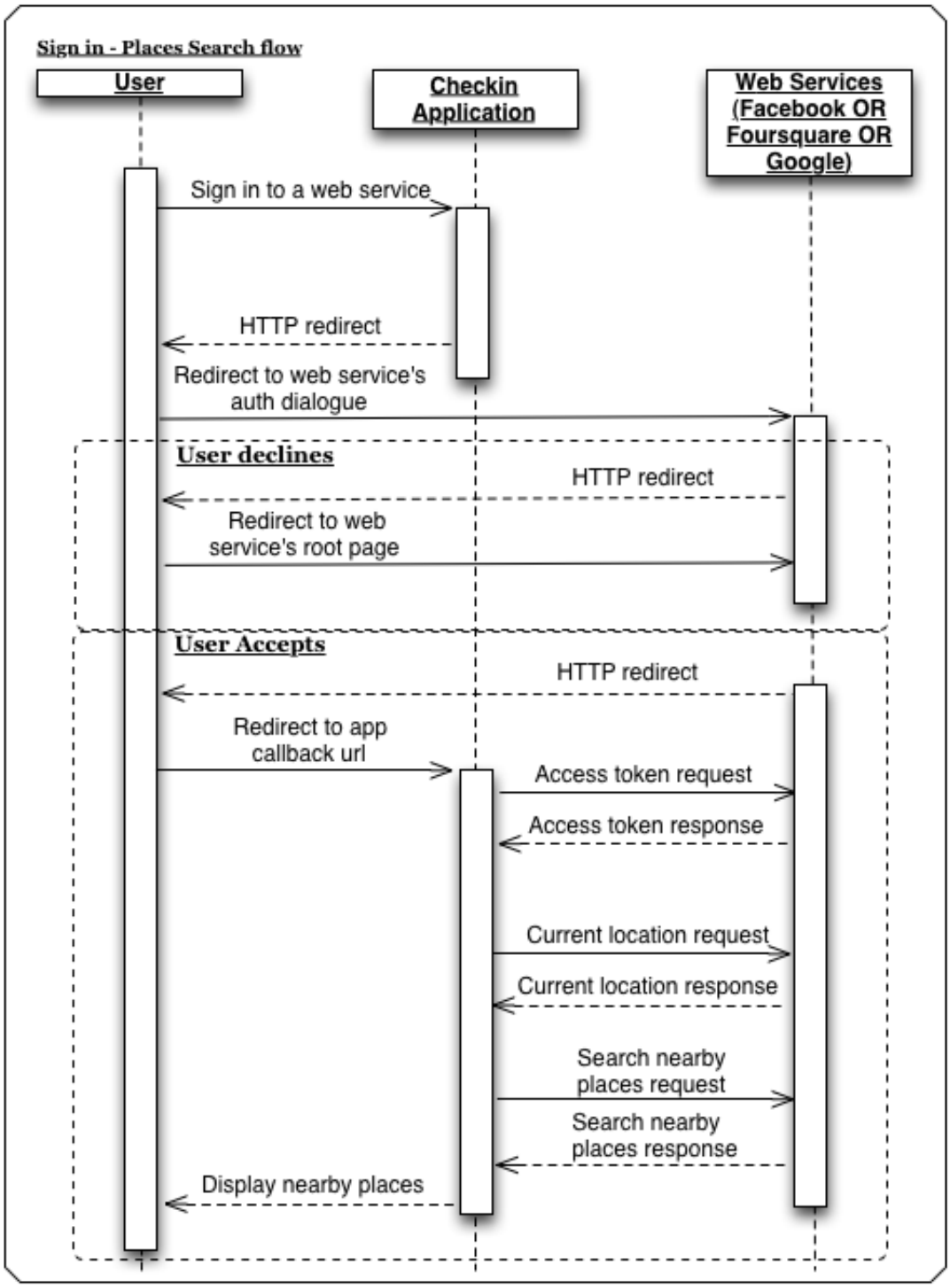
3.2 Διαγράμματα

3.2.1 Use Case Diagram

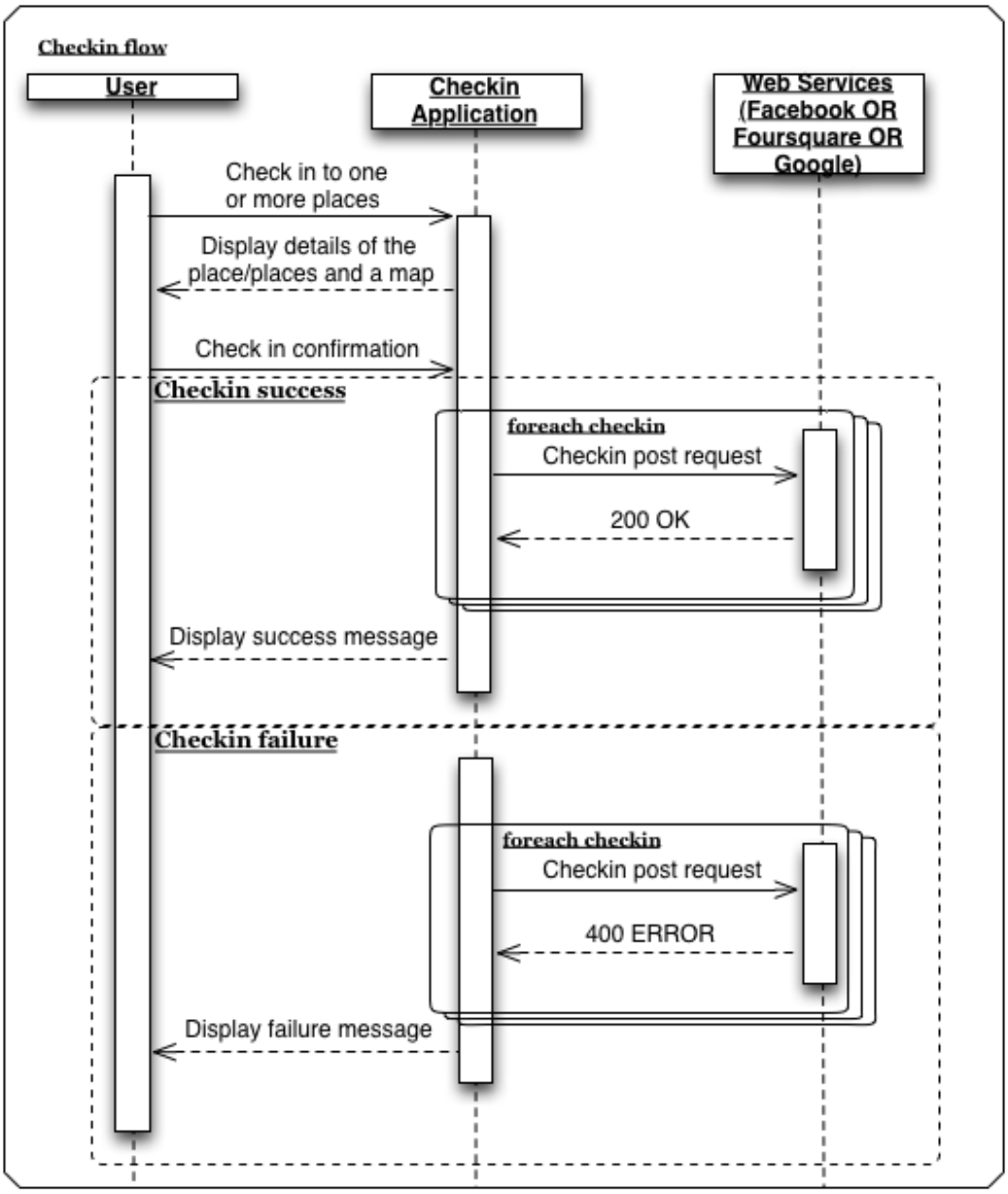


Εικόνα 3.1: Διάγραμμα χρήσης, Use Case Diagram

3.2.2 Sequencing Diagrams

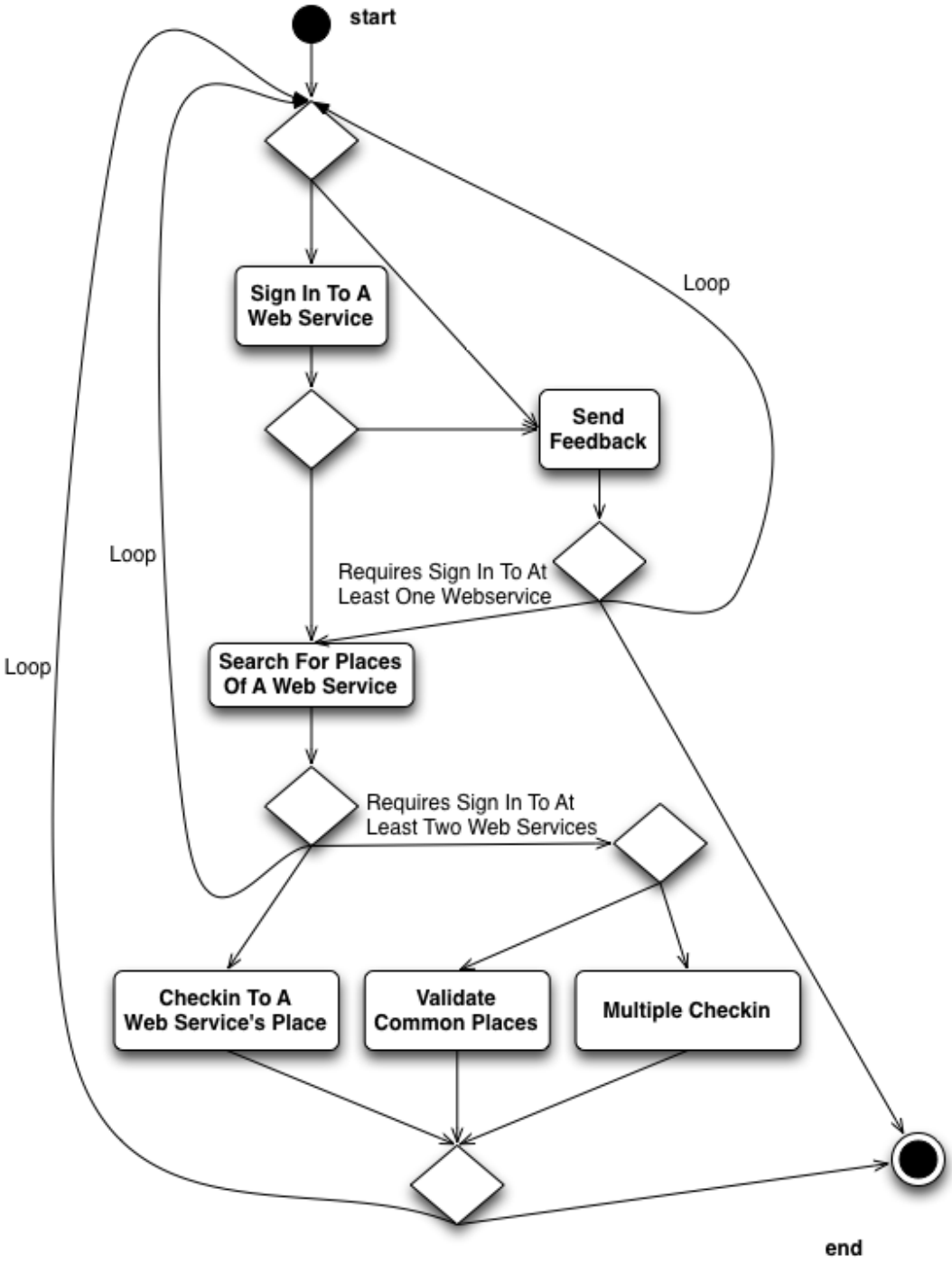


Εικόνα 3.2: Ακολουθιακό Διάγραμμα, Εγγραφή - Αναζήτηση



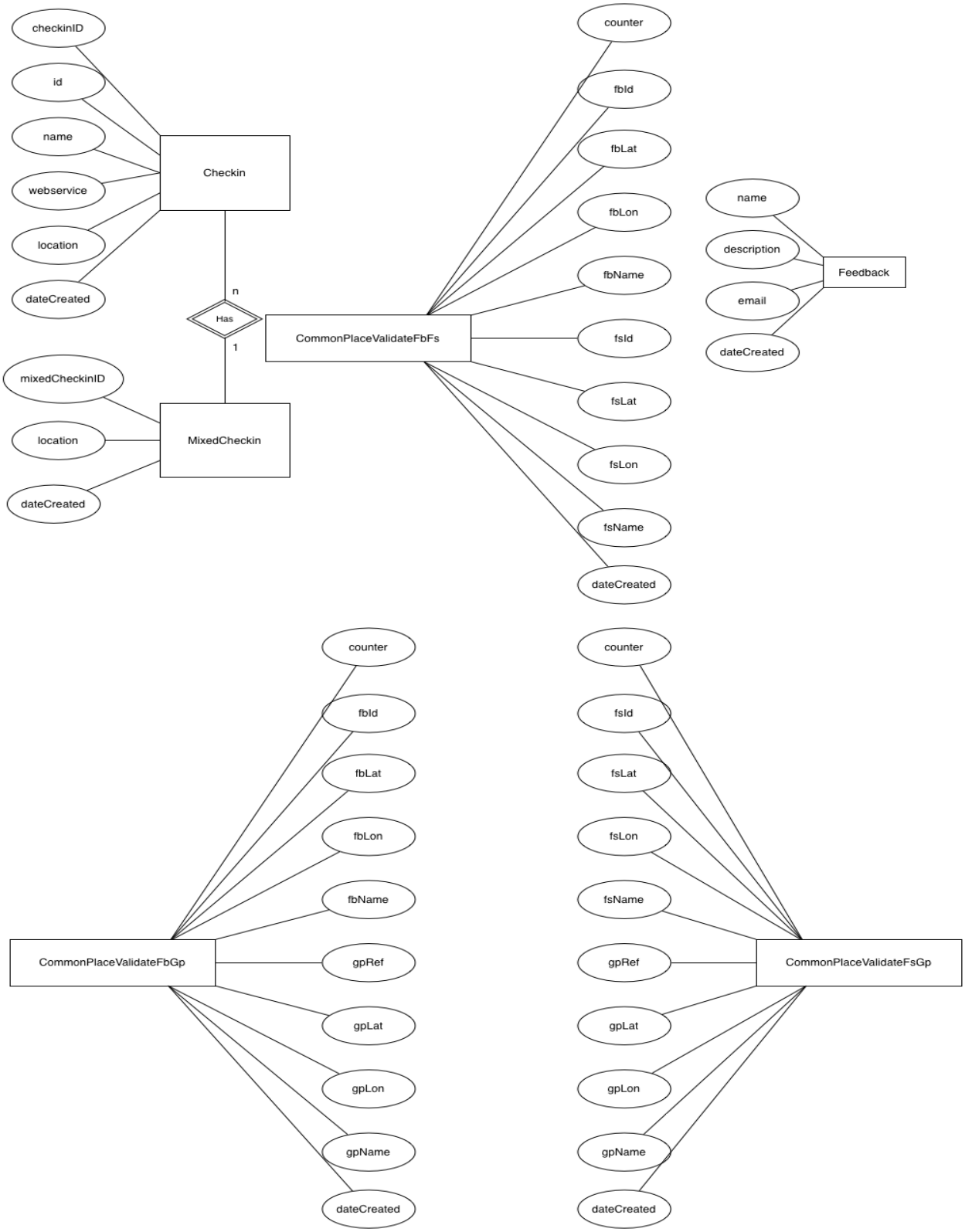
Εικόνα 3. 3: Ακολουθιακό διάγραμμα, Checkin

3.2.3 Collaboration Diagram



Εικόνα 3.4: Διάγραμμα Συνεργασίας, Collaboration Diagram

3.2.4 ER Diagram



Εικόνα 3. 5: Διάγραμμα οντοτήτων - συσχετίσεων, ER Diagram

4

Σχεδίαση Συστήματος

Στο πρώτο μέρος αυτού του κεφαλαίου παρουσιάζεται η αρχιτεκτονική του συστήματος με τη βοήθεια του οποίου υλοποιήθηκε η εφαρμογή. Στο δεύτερο μέρος του αναλύονται οι κυριότερες οθόνες της εφαρμογής.

4.1 Αρχιτεκτονική

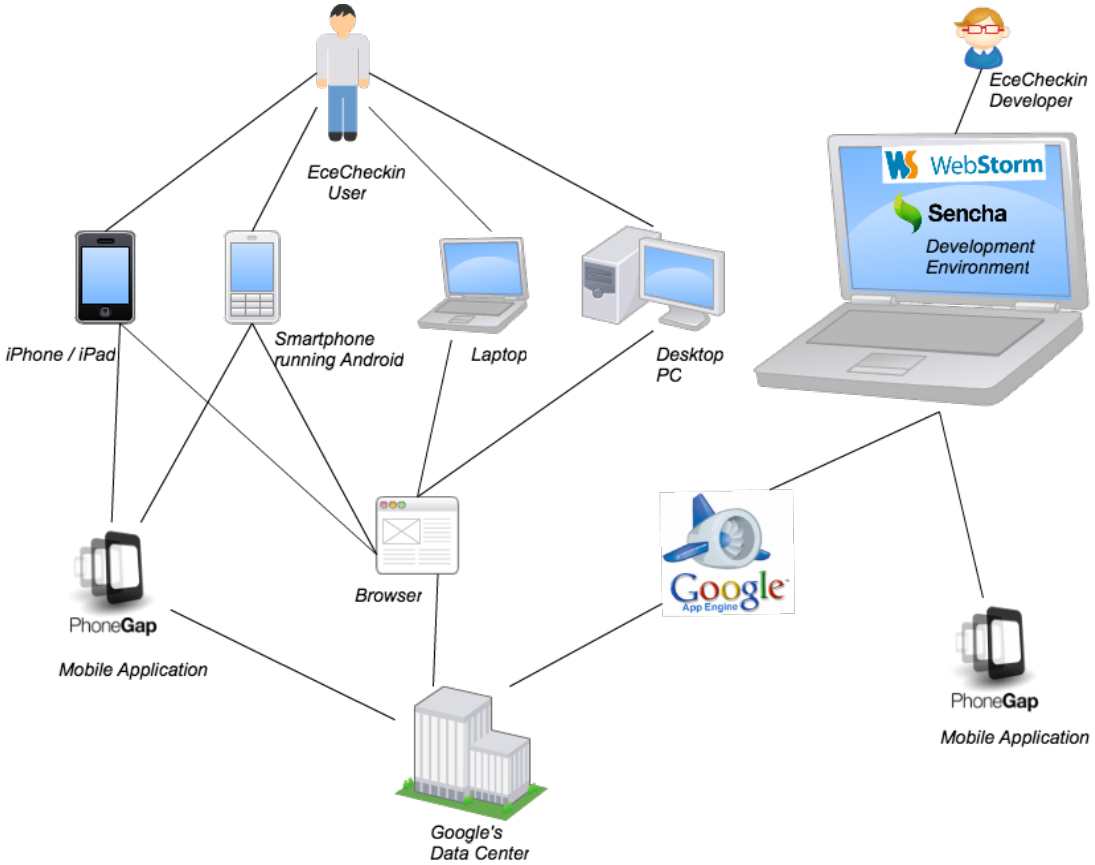
Η εφαρμογή αυτή θα είναι ανεξάρτητη πλατφόρμας, αφού θα μπορεί να «τρέχει» σε συσκευές iPhone και Android, καθώς επίσης και σε οποιαδήποτε συσκευή έχει δυνατότητες πλοήγησης στο web μέσω ενός φυλλομετρητή (browser), όπως PCs, laptops, smartphones και tablets. Για αυτόν το λόγο χρησιμοποιήθηκε ένα Open Source Mobile Framework, που λέγεται Phonegap, το οποίο δίνει τη δυνατότητα στους developers να φτιάξουν εφαρμογές για κινητά τηλέφωνα και tablets χωρίς να μπουν στη διαδικασία να γράφουν native εφαρμογές, αλλά φτιάχνοντας στην ουσία ένα web application με HTML 5, JavaScript και CSS.

Ακόμα, στην προσπάθεια αναζήτησης ενός Mobile Web Framework με στοιχεία για τη διεπαφή χρήστη (user interface) και ποικιλία εργαλείων για τη διαχείριση των δεδομένων, βρέθηκαν διάφορα, όπως jQTouch, jQuery Mobile, Sencha Touch, Titanium Mobile κ.ά.. Επιλέχθηκε το Sencha Touch μιας και βασίζεται σε αντικείμενα JavaScript και όχι σε προϋπάρχουσα HTML. Για να τεστάρουμε τον κώδικά μας χρησιμοποιήθηκε το Jasmine framework αφού ένα μεγάλο μέρος της εφαρμογής είναι γραμμένο σε JavaScript.

Θα χρειαστούμε έναν Apache server για να μπορούμε να απαντάμε στα HTTP requests των browsers των διαφόρων συσκευών και μία βάση δεδομένων για να μπορούμε να αποθηκεύουμε χρήσιμες πληροφορίες για την εφαρμογή. Με μία γρήγορη αναζήτηση στο

web, καταλήξαμε στο προϊόν της Google που ονομάζεται Google App Engine. Πρόκειται για μία cloud-computing υπηρεσία-πλατφόρμα (Platform as a Service-PaaS) που προσφέρεται για την ανάπτυξη και φιλοξενία διαδικτυακών εφαρμογών στα κέντρα δεδομένων της Google. Οι γλώσσες που υποστηρίζονται είναι οι Java, Python και Go. Η υπηρεσία προσφέρει στον προγραμματιστή ένα εργαλείο (App Engine Launcher) με το οποίο μπορεί να «σηκώνει» τοπικά την εφαρμογή του για να την «τεστάρει» αλλά και να την «ανεβάζει» στο cloud με ένα μόνο κουμπί (deploy). Ένα ακόμα σημαντικό πλεονέκτημα, είναι τα datastores. Η βάση δεδομένων αυτή δεν είναι σχεσιακή και βασίζεται πάνω στην τεχνολογία BigTable, ένα ευρείας κλίμακας κατανεμημένο σύστημα αποθήκευσης και διαχείρισης δομημένων δεδομένων. Χρησιμοποιώντας αυτή τη βάση μας παρέχεται το GeoModel, ένα open-source έργο με στόχο μια γενικευμένη λύση για την αναζήτηση και ανάκτηση γεωχωρικών δεδομένων. Όλο το πακέτο αυτό προσφέρεται δωρεάν από την Google, τουλάχιστον όσο οι πόροι που καταναλώνονται από την εφαρμογή κινούνται σε χαμηλά επίπεδα.

Τέλος, για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment – IDE) για JavaScript, CSS και HTML JetBrains WebStorm. Το εργαλείο αυτό προσφέρει επεξεργαστή πηγαίου κώδικα (source code editor), αυτόματη παραγωγή κώδικα (autocompletion), και αποσφαλματωτή (debugger) JavaScript.



Εικόνα 4.1: 'Αρχιτεκτονική του Συστήματος'

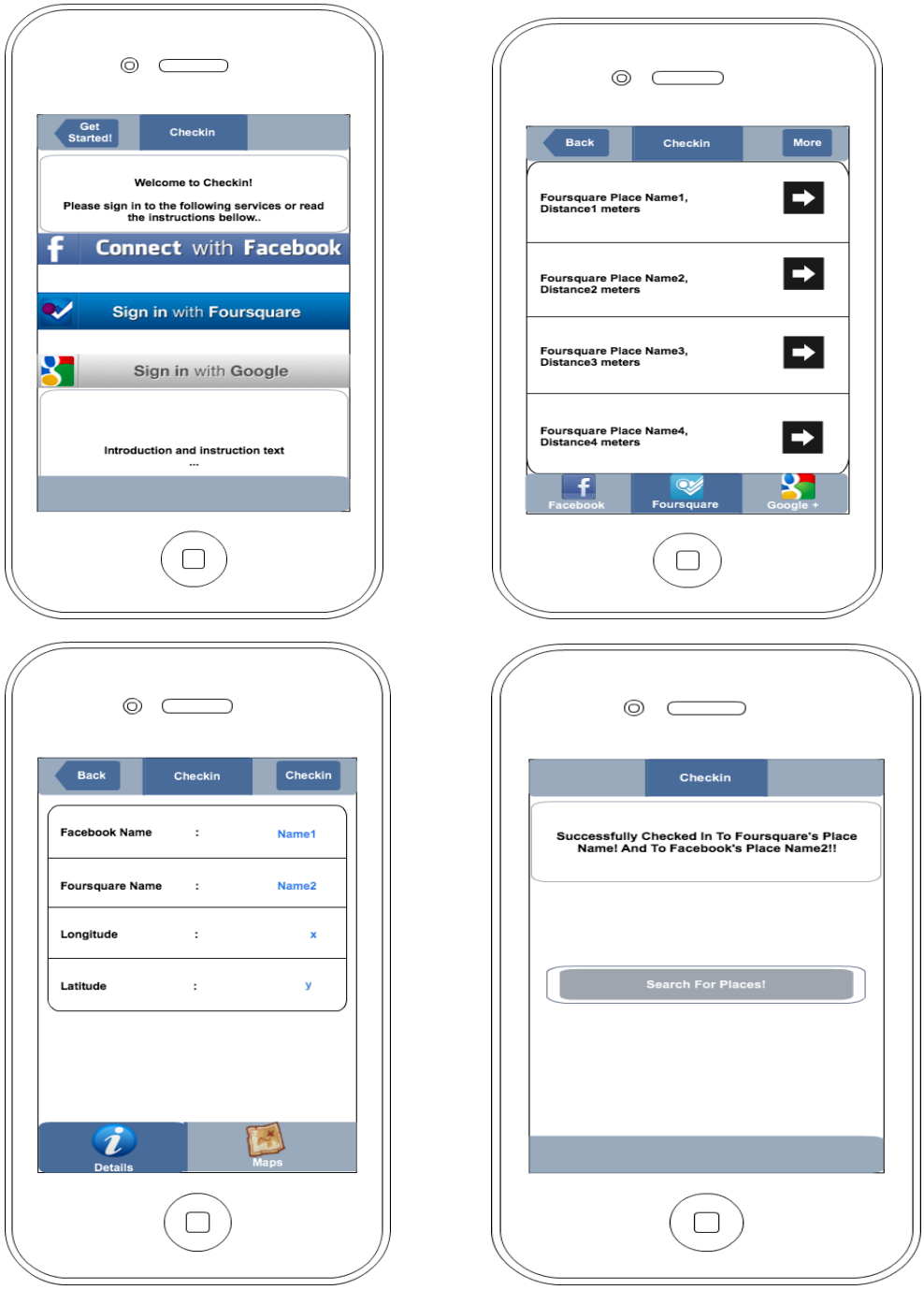
4.2 Κύριες Οθόνες

Παρακάτω θα παρουσιάσουμε τις κυριότερες οθόνες από τη σκοπιά του τελικού χρήστη.

Στο παράρτημα παρουσιάζονται το landing page σε 3 διαφορετικές συσκευές (Εικόνες 9.1-9.3) αλλά και το αναλυτικό διάγραμμα με όλες οι οθόνες της εφαρμογής (Εικόνα 9.4).

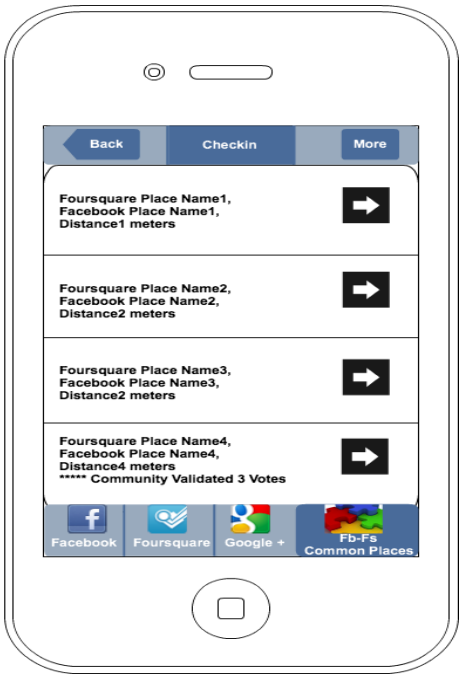
4.2.1 Simple Checkin

Ο χρήστης κάνει log in σε κάποιο social network, επιλέγει κάποιο μέρος από αυτό, βλέπει κάποιες παραπάνω λεπτομέρειες για το μέρος που επέλεξε και κάνει checkin.



4.2.2 Mixed Checkin

Ο χρήστης κάνει log in σε κάποια social networks και επιλέγει κάποιο μέρος από το mixed tab. Σε αυτό το tab εμφανίζονται τα κοινά μέρη των διαφορετικών social networks που έχει συνδεθεί ο χρήστης. Η διαδικασία προχωράει όπως προηγουμένως (Simple Checkin) και η μόνη οθόνη που αξίζει να παρουσιάσουμε είναι η παρακάτω.

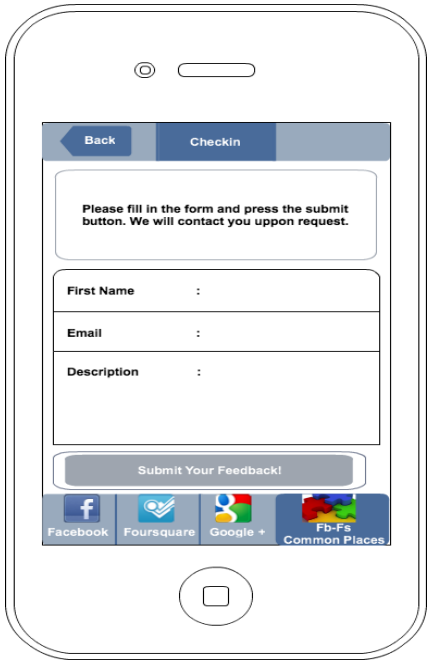


4.2.3 More button

Ο χρήστης στην πρώτη οθόνη μετά το log in, έχει τη δυνατότητα να πατήσει το More button που βρίσκεται επάνω αριστερά, ώστε να του «φανερωθούν» οι επιπλέον λειτουργίες της εφαρμογής.

4.2.4 Feedback

Ο χρήστης αφού πατήσει το More button, επιλέγει να στείλει ένα μήνυμα στους διαχειριστές της εφαρμογής, συμπληρώνοντας μία φόρμα με το email του, τον τίτλο και το περιεχόμενο του μηνύματος αυτού.



5

Υλοποίηση

Παρακάτω γίνεται μια αναφορά σε θέματα που έχουν τεχνικό ή αλγοριθμικό ενδιαφέρον όσον αφορά στην υλοποίηση της εφαρμογής. Επίσης παρατίθενται οι οδηγίες εγκατάστασης της εφαρμογής σε συσκευές με λειτουργικό iOS.

5.1 Λεπτομέρειες υλοποίησης

5.1.1 Git

Το git είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων κώδικα (distributed version control system) και δίνει την δυνατότητα στους προγραμματιστές να δουλεύουν πάνω σε ένα έργο έχοντας τον πλήρη έλεγχο του κώδικα που αναπτύσσουν χωρίς να είναι συνδεδεμένοι στο διαδίκτυο. Η γενική ιδέα είναι ότι υπάρχει σε ένα κεντρικό αποθετήριο (repository) η επίσημη έκδοση του κώδικα και με απλή εγκατάσταση του git ένας προγραμματιστής να μπορεί να την «τραβήξει» στον υπολογιστή του, βλέποντας ποιος, πότε, πώς και γιατί άλλαξε κάποιο σημείο του κώδικα.

Παρακάτω γίνεται μία αναφορά σε ένα σενάριο χρήσης του εργαλείου αυτού στην εργασία.

- Τοποθέτηση προβλήματος

  Για κάθε μικρό πρόβλημα ή στόχο της υλοποίησης της εφαρμογής δημιουργούμε ένα καινούριο issue στο bitbucket, μια υπηρεσία η οποία προσφέρει δωρεάν «φιλοξενία» repository στους χρήστες της. Όσοι παρακολουθούν το repository αυτό ενημερώνονται με email για το issue, το οποίο έχει status new, που σημαίνει ότι δεν

έχει λυθεί ακόμα. Το issue αντιστοιχεί σε ένα μοναδικό αριθμό και έναν τίτλο. Στην περίπτωση που εξετάζουμε, πρόκειται για το issue που αντιστοιχεί στο νούμερο 17 και έχει τίτλο ‘display img for fb place under checkin/html’.

- Αλλαγή κώδικα και ενημέρωση του repository

  Ο προγραμματιστής ενημερώνεται με email για το καινούριο issue, και αρχίζει τη δουλειά τρέχοντας το git στον υπολογιστή του (εικόνα 5.1).

  Στη συνέχεια εκτελεί τις απαιτούμενες αλλαγές που το issue του υποδεικνύει.

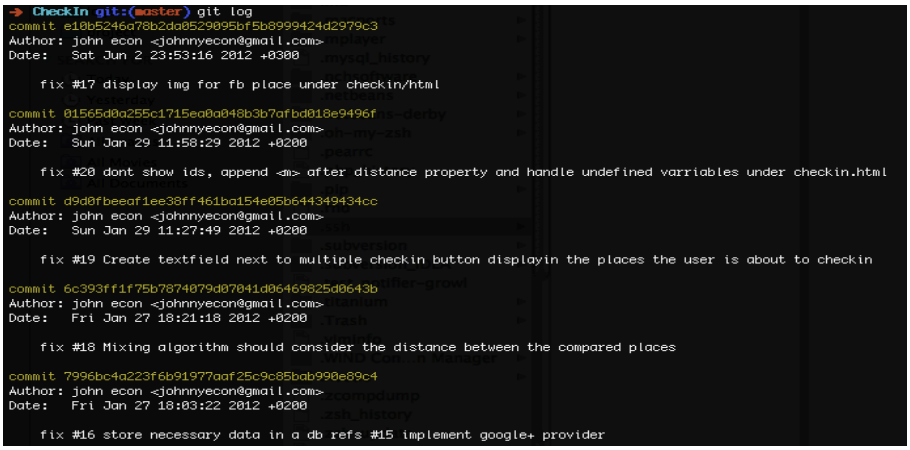
  Βλέπει τις αλλαγές του για να βεβαιωθεί για τον κώδικα που έγραψε (εικόνα 5.2) και όντας βέβαιος πια, κάνει commit την αλλαγή του στο repository (εικόνα 5.3).

  Επαληθεύει ότι το commit που έκανε είναι το τελευταίο στην αλυσίδα των commits (εικόνα 5.4), και έπειτα κάνει push την αλλαγή του (εικόνα 5.5).

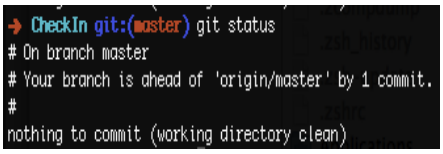
  Το status του issue έχει γίνει resolved και ο supervisor της διπλωματικής ενημερώνεται για την πρόοδο της εργασίας μέσω email (εικόνα 5.6).

Εικόνα 5.1: git status

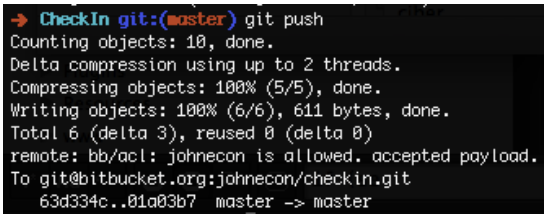
Εικόνα 5.2: git diff



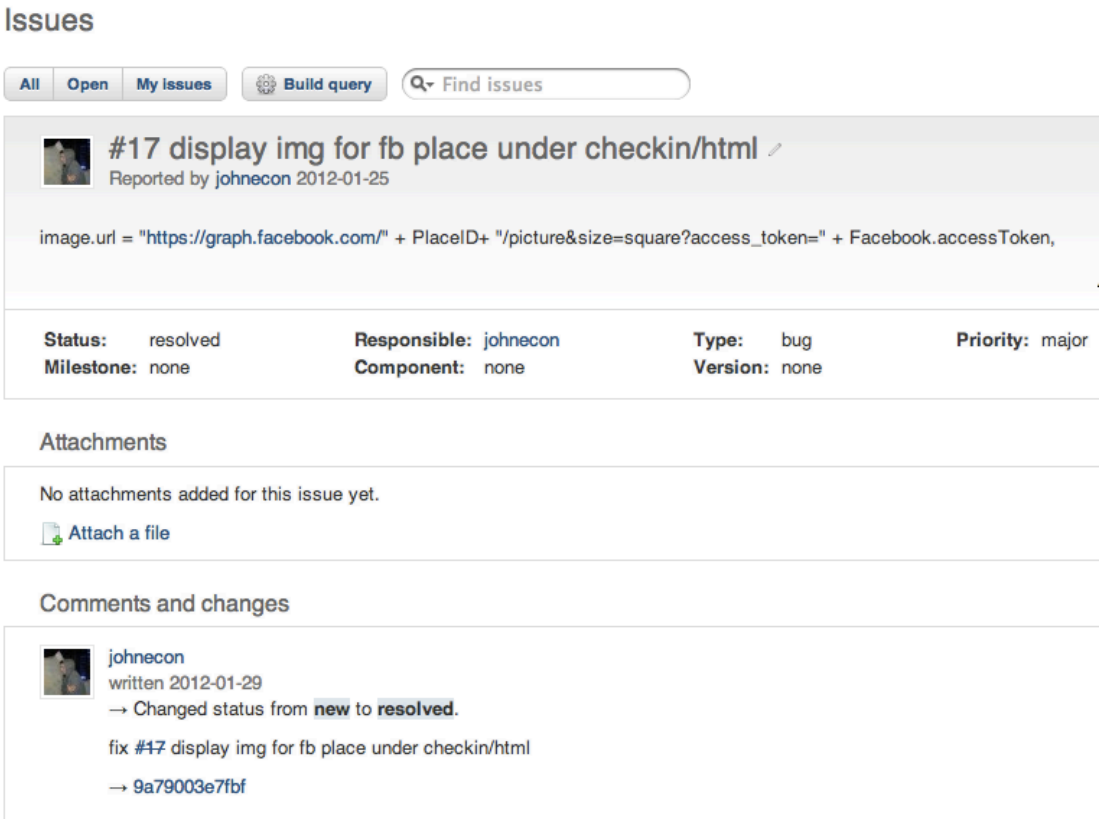
Εικόνα 5.3: git log



Εικόνα 5.4: git status



Εικόνα 5.5: git push



Εικόνα 5.6: Bitbucket issue resolved

Είναι φανερό ότι το εργαλείο αυτό αποτελεί μία πηγή γνώσης καθώς ανά πάσα στιγμή είναι προσβάσιμος ο τρόπος με τον οποίο λύθηκαν όλοι οι επιμέρους στόχοι στους οποίους

διασπάστηκε η αρχική εφαρμογή, προσφέροντας ένα αξιόπιστο αντίγραφο ασφαλείας (backup) και ενισχύοντας τις αρχές του Lean Software Development.

5.1.2 Google App Engine

Με τον όρο cloud computing αναφερόμαστε στην παροχή υπολογιστικής ισχύος, χώρου αποθήκευσης, δικτυακού εξοπλισμού και όλων των μέσων όπου μπορούν επιχειρήσεις ή και ομάδες ανθρώπων να χρησιμοποιήσουν χωρίς να χρειάζεται να κατέχουν ανάλογο εξοπλισμό. Με αυτόν τον τρόπο οι cloud εφαρμογές γίνονται πιο ελαστικές, καθώς ανάλογα με τη ζήτηση που έχουν από τους χρήστες υπάρχει η δυνατότητα να χρησιμοποιούν περισσότερους πόρους από το υπολογιστικό σύστημα στο οποίο φιλοξενούνται.

Για την παρούσα εργασία χρησιμοποιήθηκε η Cloud υπηρεσία της Google, το Google App Engine. Η πλατφόρμα αυτή αναπτύχθηκε έτσι ώστε να μπορεί να εξυπηρετήσει εφαρμογές πραγματικού χρόνου με πολλούς χρήστες χωρίς να χάνεται η απόδοση του συστήματος. Οι εφαρμογές που αναπτύσσονται με το Google App Engine μπορούν να είναι άμεσα διαθέσιμες σε όλον τον κόσμο. Ο προγραμματιστής αποστέλλει την εφαρμογή του (deploy) μέσω του Google App Engine Launcher και το σύστημα αναλαμβάνει να αντιγράψει την εφαρμογή σε όλα τα data center της Google ανά τον κόσμο ώστε να ελαχιστοποιούνται οι χρόνοι χρήσης της υπηρεσίας, ανεξάρτητα από τη γεωγραφική τοποθεσία του τελικού χρήστη.

Επίσης, προσφέρονται στον system administrator διαγράμματα (errors/sec, requests/sec κτλ) και στατιστικά στοιχεία με τους πόρους που καταναλώνει η εφαρμογή. Για δοκιμαστικές εφαρμογές που δεν έχουν αρκετούς χρήστες και συνεπώς δεν καταναλώνουν αρκετούς πόρους, η υπηρεσία είναι δωρεάν.

5.1.3 Database

Το Google App Engine υποστηρίζει Java, Python και Go. Για την παρούσα εργασία χρησιμοποιήθηκε η Python και σε αυτό το σημείο θα γίνει μια μικρή αναφορά στον τρόπο που γίνονται τα reads και writes από την database.

Κάθε φορά που θέλουμε να διαβάσουμε ή να γράψουμε στη βάση μας, αποστέλλεται ένα JSONP request από την εφαρμογή στο cloud. Ας υποθέσουμε ότι κάποιος χρήστης θέλει να κάνει ένα ‘Mixed’ Checkin, δηλαδή να επιλέξει ένα κοινό μέρος δύο διαφορετικών social networks από το Common Places Tab και να κάνει Checkin σε αυτό. Ο Javascript κώδικας της εφαρμογής σε συνεργασία με το Sencha Touch framework αναλαμβάνει την αποστολή του request. Αυτό γίνεται ως εξής:

```
Ext.util.JSONP.request({
        url: 'http://johnnyecon.appspot.com/db/',
        callbackKey: 'callback',
        params: {
                service: service,
                transaction: 'write',
                fbId: place.fbId,
                fbName: place.fbName,
                fbLat: place.fbLat,
                fbLon: place.fbLon,
                gpId: place.gpId,
                gpName: place.gpName,
                gpLat: place.gpLat,
                gpLon: place.gpLon
        },
```

Το request φτάνει στο cloud και κάποιος server θα αναλάβει να το εξυπηρετήσει. Το Google App Engine δίνει τη δυνατότητα στον προγραμματιστή να διαμορφώσει όπως εκείνος θέλει το αρχείο app.yaml το οποίο είναι υπεύθυνο για τη δρομολογηση τα requests που φτάνουν στο cloud. Συγκεκριμένα στην εφαρμογή, το app.yaml αρχείο περιέχει τις παρακάτω εγγραφές:

```
handlers:
- url: /db/.*
  script: db/transaction.py
```

Με αυτόν τον τρόπο ορίζουμε ότι όλα τα requests στο url johnnyecon.appspot.com/db/* θα εξυπηρετούνται από το αρχείο transaction.py που βρίσκεται κάτω από το φάκελο db και μάλιστα ότι αυτό το αρχείο θα περιέχει κώδικα python. Στο συγκεκριμένο αρχείο λοιπόν, αρχικά κάνουμε import τις βιβλιοθήκες που θα χρειαστούμε, και στη συνέχεια φτιάχνουμε τα μοντέλα Checkin και Mixed Checkin που στην ουσία αποτελούν αναπαράσταση των πινάκων στη βάση δεδομένων:

```
import cgi
import datetime
import simplejson
import urllib
import urllib2
from google.appengine.ext import db
from geo.geomodel import GeoModel

class Checkin(GeoModel):
    id = db.StringProperty(required=True)
    name = db.StringProperty(required=True)
    dateCreated = db.DateTimeProperty()
    webservice = db.StringProperty(required=True)


class MixedCheckin(GeoModel):
    dateCreated = db.DateTimeProperty()
```

Στη συνέχεια ορίζουμε τη συνάρτηση getUrlParameter() και παίρνουμε από το request τις παραμέτρους που χρειαζόμαστε. Με τη βοήθεια των Geomodels μπορούμε να αποθηκεύσουμε γεωγραφικά δεδομένα (location) που στηρίζονται στα γεωγραφικό μήκος και γεωγραφικό πλάτος[9]. Αυτό αργότερα θα μας βοηθήσει πολύ, καθώς η αναζήτηση θα είναι εντυπωσιακά γρηγορότερη συγκριτικά με μια παρόμοια υλοποίηση σε μια σχεσιακή βάση δεδομένων όπως η MySql. Τέλος επιστρέφουμε στο callback της javascript, εκτυπώνοντας το κατάλληλο response.

```
def getUrlParameter(arg):
    form = cgi.FieldStorage()
    for i in form.keys():
        if i == arg:
            return form[i].value

transaction = getUrlParameter('transaction')
service = getUrlParameter('service')
currentLat = getUrlParameter('currentLat')
currentLon = getUrlParameter('currentLon')

if (transaction == 'write'):

```

[9] http://pypi.python.org/pypi/geomodel, http://code.google.com/p/geomodel/wiki/Usage

```
    if (service == 'MixedFbGp'):
            mixedCheckin = MixedCheckin(location=db.GeoPt(float(getUrlParameter('fbLat')),
float(getUrlParameter('fbLon'))))
            mixedCheckin.dateCreated = datetime.datetime.now()
            mixedCheckin.update_location()
            mixedCheckin.put()
            gpCheckin = Checkin(location=db.GeoPt(float(getUrlParameter('gpLat')),
float(getUrlParameter('gpLon'))),
                 id=getUrlParameter('gpId'),
                 name=getUrlParameter('gpName').decode('utf-8'),
                 webservice='GooglePlus',
                 parent=fbGpMixedCheckin)
            gpCheckin.dateCreated = datetime.datetime.now()
            gpCheckin.update_location()
            gpCheckin.put()
            fbCheckin = Checkin(location=db.GeoPt(float(getUrlParameter('fbLat')),
float(getUrlParameter('fbLon'))),
                 id=getUrlParameter('fbId'),
                 name=getUrlParameter('fbName').decode('utf-8'),
                 webservice='Facebook',
                 parent=fbGpMixedCheckin)
            fbCheckin.dateCreated = datetime.datetime.now()
            fbCheckin.update_location()
            fbCheckin.put()
            print 'Content-Type: application/json\n'
            print 'Ext.util.JSONP.callback({})'
```

5.1.4 Javascript

Η εφαρμογή πρέπει να θυμάται κατά κάποιον τρόπο αν ο χρήστης έχει κάνει log in σε κάποιο Social Network πρόσφατα. Για αυτόν το λόγο αποθηκεύουμε ένα cookie για μία ώρα στο φυλλομετρητή κάθε φορά που ο χρήστης κάνει log in σε κάποιο Social Network[10]. Αυτό γίνεται με τον παρακάτω κώδικα:

[10] http://www.w3schools.com/js/js_cookies.asp

```
function setCookie(c_name,value,expireseconds)
{
    var exdate=new Date();
    exdate.setTime(exdate.getTime()+(expireseconds*1000))
    var c_value=escape(value) + ((expireseconds==null) ? "" : "; expires="+exdate.toUTCString());
    document.cookie=c_name + "=" + c_value;
}

var accessToken = getUrlParameter('access_token');
setCookie('fbAccessToken', accessToken, 3600);
```

Μπορούμε να πάρουμε την τιμή ενός cookie με την παρακάτω συνάρτηση:

```
function getCookie(c_name)
{
    var i,x,y,ARRcookies=document.cookie.split(";");
    for (i=0;i<ARRcookies.length;i++)
    {
        x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
        y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
        x=x.replace(/^\s+|\s+$/g,"");
        if (x==c_name)
        {
            return unescape(y);
        }
    }
}
```

Για τη σύγκριση δύο Places από διαφορετικά Social Networks δημιουργήθηκε μια συνάρτηση που επιστρέφει true ή false, ανάλογα με το αν αυτά είναι ίδια ή αν όχι, σύμφωνα με τον αλγόριθμο που περιγράφεται παρακάτω. Για να γυρίσει true αυτή η συνάρτηση θα πρέπει τα ορίσματα που θα πάρει, να ικανοποιούν και το distanceCriterion() και το nameCriterion().

```
function compareItems(fbItem, fsItem) {
    return (distanceCriterion(fbItem, fsItem) && nameCriterion(fbItem, fsItem));
}

function nameCriterion(fbItem, fsItem) {
    return ((fbItem.data.name == fsItem.data.name) ||
            (removeMarks(translitToUpperLatin(fbItem.data.name))==
             removeMarks(translitToUpperLatin(fsItem.data.name))));
}

function distanceCriterion(fbItem, fsItem) {
    return (Math.abs(fbItem.data.distance-fsItem.data.distance) < 60);
}
```

Το distanceCriterion() ικανοποιείται μόνο όταν τα δυο Places απέχουν λιγότερο από 60 μέτρα μεταξύ τους. Το nameCriterion() απαιτεί είτε τα ονόματα των δύο Places να είναι ακριβώς ίδια, είτε να γίνονται ίδια όταν μεταφραστούν σε λατινικά κεφαλαία και έπειτα αφαιρεθούν τα σημεία στίξης. Παρακάτω παρατίθεται η συνάρτηση translitToUpperLatin() και η removeMarks()[11].

[11] http://www.translatum.gr/converter/greek-transliteration.htm

```
function toUpperLatinChar(char) {
    return ((char == 'α') || (char == 'ά') || (char == 'A') || (char == 'Ά') ||
        (char == 'a') || (char == 'à') || (char == 'á') || (char == 'ä') || (char == 'â')) ? 'A' :
    ((char == 'β') || (char == 'B')) || (char == 'b') ? 'B' :
    ((char == 'γ') || (char == 'Γ') || (char == 'g')) ? 'G' :
    ((char == 'δ') || (char == 'Δ') || (char == 'd')) ? 'D' :
    ((char == 'ε') || (char == 'έ') || ((char == 'E')) || (char == 'Έ') ||
        (char == 'e') || (char == 'ë') || (char == 'ê') || (char == 'é') || (char == 'è')) ? 'E' :
    ((char == 'ζ') || (char == 'Z')) ? 'Z' :
    ((char == 'η') || (char == 'ή') || (char == 'H') ||
        (char == 'Ή')) ? 'I' :
    ((char == 'θ') || (char == 'Θ')) ? 'TH' :
    ((char == 'ι') || (char == 'ϊ') || (char == 'ί') || (char == 'I') || (char == 'Ί') ||
        (char == 'i') || (char == 'ì') || (char == 'í') || (char == 'ï') || (char == 'î')) ? 'I' :
    ((char == 'κ') || (char == 'K') || (char == 'k')) ? 'K' :
    ((char == 'λ') || (char == 'Λ') || (char == 'l')) ? 'L' :
    ((char == 'μ') || (char == 'M') || (char == 'm')) ? 'M' :
    ((char == 'ν') || (char == 'N') || (char == 'n')) ? 'N' :
    ((char == 'ξ') || (char == 'Ξ')) ? 'X' :
    ((char == 'ο') || (char == 'ό') || (char == 'O') || (char == 'Ό') ||
        (char == 'o') || (char == 'ó') || (char == 'ò') || (char == 'ö') || (char == 'ô')) ? 'O' :
    ((char == 'π') || (char == 'Π') || (char == 'p')) ? 'P' :
    ((char == 'ρ') || (char == 'P') || (char == 'r')) ? 'R' :
    ((char == 'σ') || (char == 'ς') || (char == 'Σ') || (char == 's')) ? 'S' :
    ((char == 'τ') || (char == 'T') || (char == 't')) ? 'T' :
    ((char == 'υ') || (char == 'ύ') || (char == 'Y') || (char == 'Ύ')) ? 'I' :
    ((char == 'φ') || (char == 'Φ')) ? 'F' :
    ((char == 'χ') || (char == 'X')) ? 'CH' :
    ((char == 'ψ') || (char == 'Ψ')) ? 'PS' :
    ((char == 'ω') || (char == 'ώ') || (char == 'Ω') || (char == 'Ώ')) ? 'O' :
    (char == 'c') ? 'C' :
    (char == 'f') ? 'F' :
    (char == 'h') ? 'H' :
    (char == 'j') ? 'J' :
    (char == 'q') ? 'Q' :
    ((char == 'u') || (char == 'ù') || (char == 'ú') || (char == 'ü') || (char == 'û'))? 'U' :
    (char == 'v') ? 'V' :
    (char == 'w') ? 'W' :
    (char == 'x') ? 'X' :
    (char == 'y') ? 'Y' :
    (char == 'z') ? 'Z' : char;
}

```

```
function translitToUpperLatin(string) {
    string = string.toLowerCase();
    var translitString = '';
    for (var i=0; i < string.length; i++) {
        var char = string[i];
        char = toUpperLatinChar(char);
        translitString += char;
    }
    return translitString;
}

function removeMarks(string) {
    var stringWithoutMarks = '';
    for (var i=0; i < string.length; i++) {
        var char = string[i];
        if (!((char == "!") || (char == '-') || (char == ' '))) {
            stringWithoutMarks += char;
        }
    }
    return removeQuoteMarks(stringWithoutMarks);
}
```

5.1.5 Memcache

Στα σύγχρονα πληροφοριακά συστήματα απαιτούνται μονάδες αποθήκευσης ώστε τα δεδομένα να παραμένουν σε συνεπή μορφή ακόμα και αν προκύψει κάποιο πρόβλημα υλικού. Το μέσο που επιλέγεται είναι οι σκληροί δίσκοι, όπου μαγνητικοί δίσκοι περιστρέφονται με μεγάλες ταχύτητες και κεφαλές γράφουν δεδομένα πάνω τους. Η εγγραφή ή η ανάγνωση στοιχείων από τους δίσκους είναι ακριβή, όσον αφορά το χρόνο ως προς τη συνολική απόκριση της εφαρμογής.

Οι web εφαρμογές υψηλών επιδόσεων χρησιμοποιούν μέσα αποθήκευσης όπως η Ram των μηχανών, καθώς επιτυγχάνονται καλύτερες ταχύτητες από αυτές των σκληρών δίσκων. Το μειονέκτημα με αυτές τις υλοποιήσεις είναι ότι δεν αποτελούν λειτουργική επιλογή για αποθήκευση μόνιμων δεδομένων μιας και σε πιθανή διακοπή ρεύματος τα δεδομένα χάνονται μέσα από τη Ram. Πολλά από τα δεδομένα που δημιουργούνται από την εφαρμογή δε χρειάζονται να αποθηκεύονται μόνιμα. Η Memcache αναλαμβάνει να τα αποθηκεύσει σε προσωρινές μνήμες για κάποια ορισμένα χρονικά διαστήματα. Είναι μια υπηρεσία που προσφέρεται στο Google App Engine και δίνει τη δυνατότητα να εκχωρούνται πληροφορίες σε αντικείμενα, που με τη σειρά τους αποθηκεύονται στη μνήμη Ram των εξυπηρετητών, και

η πρόσβαση σε αυτά γίνεται με το αντίστοιχο κλειδί του αντικειμένου. Σχεδιάστηκε ώστε τα ερωτήματα να εκτελούνται ταχύτατα, με μικρούς χρόνους απόκρισης στην αποθήκευση και ανάκτησή τους. Κάποιοι από τους βασικούς κανόνες της Memcache παρουσιάζονται παρακάτω:

- Υπάρχει μία μνήμη (cache memory) που διαμοιράζεται ανάμεσα σε όλες τις οντότητες της εφαρμογής. Εάν αποθηκευτεί ή διαγραφεί κάτι από τη μνήμη, σε μία οντότητα, τότε οι αλλαγές εφαρμόζονται άμεσα και στις υπόλοιπες.
- Η Memcache λειτουργεί σα μία μεγάλη μεταβλητή, τύπου dictionary στην Python, όπου κάθε αντικείμενο μέσα στο dictionary έχει ένα μοναδικό κλειδί.
- Κάθε αντικείμενο έχει κάποιο ορισμένο χρόνο ζωής που μπορεί να οριστεί από μερικά δευτερόλεπτα μέχρι και ένα μήνα.
- Τα αντικείμενα που αποθηκεύονται θα πρέπει να μην έχουν μεγάλο μέγεθος.
- Οι πληροφορίες που πρόκειται να αποθηκευτούν θα πρέπει να έχουν ένα μεγάλο ποσοστό να γίνει ανάκλησή τους.

Παρακάτω παρουσιάζεται ένα παράδειγμα χρήσης της Memcache. Έστω ότι φτάνει ένα request στο cloud στο οποίο πρέπει να απαντήσουμε με τα κοινά μέρη του Facebook και του Foursquare που οι χρήστες της κοινότητας έχουν επαληθεύσει. Αρχικά επιχειρούμε να διαβάσουμε αυτές τις εγγραφές από την cache memory. Αν τις βρούμε εκεί (cache hit) τότε τα επιστρέφουμε αμέσως. Αλλιώς (cache miss) τα διαβάζουμε από τη βάση δεδομένων και πριν τα επιστρέψουμε τα γράφουμε στην cache memory, ώστε την επόμενη φορά που το cloud δεχτεί το ίδιο request, να έχει σημαντικά μικρότερο χρόνο απόκρισης.

```
1 from google.appengine.api import memcache
2 from google.appengine.ext import db
3 from geo.geomodel import GeoModel
4
5 service = getUrlParameter('service')
6 currentLat = getUrlParameter('currentLat')
7 currentLon = getUrlParameter('currentLon')
8
9
10 class CommunityValidatedCommonPlaceFbFs(GeoModel):
11
12
13 def readFromModel(model):
14     if (model == 'FbFs'):
15         return CommunityValidatedCommonPlaceFbFs
16
17
```

```
def fetchRecordsFromDb(service):
    model = readFromModel(service)
    return model.proximity_fetch(
        model.all(),
        db.GeoPt(float(currentLat), float(currentLon)),
        max_results=10,
        max_distance=500)  # Within 500 m.


def getRecords(service):
    records = memcache.get(service + currentLat[0:7] + currentLon[0:7])
    if records is not None:
        return records
    else:
        records = fetchRecordsFromDb(service)
        memcache.add(service + currentLat[0:7] + currentLon[0:7], records, 3600)
        return records

if (transaction == 'readFbFs'):
    fbFsRecords = getRecords('FbFs')
```

Το Google App Engine προσφέρει και στατιστικά δεδομένα των cache hits και cache misses (εικόνα 5.7)

Memcache Viewer

Cache flushed, all keys dropped.

Statistics	
Hit count:	28
Miss count:	3
Hit ratio:	90%
Item count:	4 item(s)
Total cache size:	3691 byte(s)
Oldest item age:	10 second(s)
	Flush Cache

Εικόνα 5.7: Memcache hits and misses

5.1.6 Cron Job

Τα Cron Jobs μπορούν να αυτοματοποιήσουν κάποια διαδικασία ανά τακτά χρονικά διαστήματα που ορίζονται από τον προγραμματιστή και δημιουργήθηκαν για να διευκολύνουν

τους διαχειριστές ενός ιστότοπου. Στη δική μας περίπτωση γράψαμε ένα Cron Job που συλλέγει όλα τα checkins και όλα τα feedbacks της ημέρας και τα αποστέλλει με email στους διαχειριστές της εφαρμογής.

Για την υλοποίηση αυτής της λειτουργίας έπρεπε να επεξεργαστούμε το cron.yaml, το app.yaml και το dailyReport.py ως εξής:

```
cron:
- description: new daily summary job
  url: /report/daily
  schedule: every 24 hours
```

Snippet 1 Cron.yaml

```
handlers:
- url: /report/daily
  script: crons/dailyReport.py
```

Snippet 2 app.yaml

```
import datetime
from google.appengine.ext import db
from geo.geomodel import GeoModel
from google.appengine.api import mail


class Checkin(GeoModel):
    id = db.StringProperty(required=True)
    name = db.StringProperty(required=True)
    dateCreated = db.DateTimeProperty()
    webservice = db.StringProperty(required=True)


class Feedback(db.Model):
    name = db.StringProperty(required=True)
    email = db.StringProperty(required=True)
    description = db.StringProperty(required=True, multiline=True)
    dateCreated = db.DateTimeProperty()


def getNewCheckins():
    now = datetime.datetime.now()
    yesterday = now - datetime.timedelta(days=1)
    query = db.GqlQuery("SELECT * FROM Checkin " +
                "WHERE dateCreated > :1 AND dateCreated < :2 ",
                 yesterday, now)
    return query.fetch(100)
```

```
28
29
30 def getNewFeedbacks():
31     now = datetime.datetime.now()
32     yesterday = now - datetime.timedelta(days=1)
33     query = db.GqlQuery("SELECT * FROM Feedback " +
34                 "WHERE dateCreated > :1 AND dateCreated < :2 ",
35                  yesterday, now)
36     return query.fetch(100)
37
38
39 def getMessageBody():
40     checkins = getNewCheckins()
41     feedbacks = getNewFeedbacks()
42     body = 'Dear EceCheckin Developers!\n'
43     body += 'I am glad to announce you...\n'
44     if checkins is not None:
45         if (len(checkins) != 0):
46             body += '\n\nNew Checkins Found!\n'
47             for i in range(len(checkins)):
48                 body += '\nservice: ' + checkins[i].webservice
49                 body += '\nname: ' + checkins[i].name + '\n'
50         else:
51             body += '\n\nNo new Checkins Found..'
52     else:
53         body += '\n\nNo new Checkins Found..'
54     if feedbacks is not None:
55         if (len(feedbacks) != 0):
56             body += '\n\nNew Feedbacks Found!\n'
57             for i in range(len(feedbacks)):
58                 body += '\nname: ' + feedbacks[i].name
59                 body += '\nemail: ' + feedbacks[i].email + '\n'
60                 body += '\ndescription: ' + feedbacks[i].description + '\n'
61         else:
62             body += '\n\nNo new Feedbacks Found..'
63     else:
64         body += '\n\nNo new Feedbacks Found..'
65     return body
66
67
68 message = mail.EmailMessage(sender="<johnnyecon@gmail.com>",
69                             subject="Daily Report")
70
```

```
message.to = "<johnnyecon@gmail.com>"
message.body = getMessageBody()
message.send()
```
Snippet 3 dailyReport.py

5.1.7 Jasmine

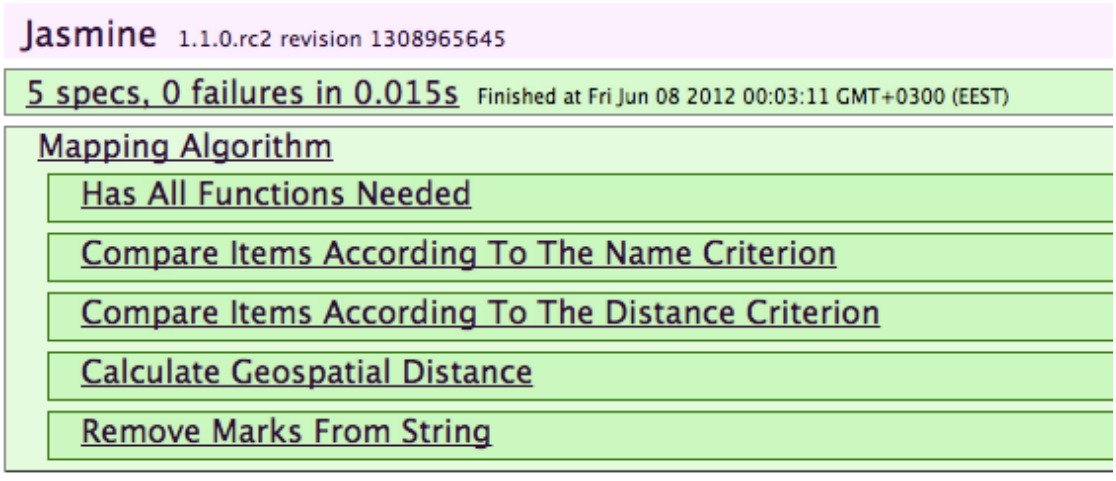
Όπως αναφέρθηκε και στο Κεφάλαιο 2, μία από τις βασικές αρχές του Lean Software Development είναι η συγγραφή των tests για την εφαρμογή που υλοποιείται. Για αυτόν το λόγο λοιπόν χρησιμοποιήθηκε το Jasmine Framework. Πρόκειται για ένα behaviour-driven development framework με απλή σύνταξη που επιτρέπει την εύκολη συγγραφή των tests. Μία σουίτα από tests αρχίζει με την συνάρτηση describe η οποία παίρνει σαν ορίσματα τον τίτλο της σουίτας και μία συνάρτηση. Η συνάρτηση αποτελείται από μία ή περισσότερες προδιαγραφές (specs). Κάθε προδιαγραφή ορίζεται με τη συνάρτηση it που πέρνει σαν ορίσματα τον τίτλο της προδιαγραφής και μία συνάρτηση, η οποία με τη σειρά της αποτελείται από μία ή περισσότερες υποθέσεις (assertions). Παρακάτω παρουσιάζονται τα tests που γράφτηκαν για τον αλγόριθμο αντιστοίχισης των Places που περιγράφηκε στο Κεφάλαιο 5.1.4 και τα αποτελέσματα των tests όπως αυτά φαίνονται στον φυλλομετρητή (εικόνα 5.8).

```
describe("Mapping Algorithm", function() {
    it("Has All Functions Needed", function() {
        expect(compareItems).toBeDefined();
        expect(nameCriterion).toBeDefined();
        expect(distanceCriterion).toBeDefined();
        expect(distance).toBeDefined();
        expect(removeMarks).toBeDefined();
    });

    it("Compare Items According To The Name Criterion", function() {
        testItem1 = {};
        testItem1.data = {};
        testItem2 = {};
        testItem2.data = {};
        testItem1.data.name = 'Foo!';
        testItem1.data.distance = '4';
        testItem2.data.name = 'Foo';
        testItem2.data.distance = '3';
        expect(compareItems(testItem1, testItem2)).toBeTruthy();
        testItem2.data.name = 'Fo';
        expect(compareItems(testItem1, testItem2)).toBeFalsy();
        testItem1.data.name = '!Foo - !';
        testItem1.data.distance = '4';
```

```
24          testItem2.data.name = '!- Foo';
25          testItem2.data.distance = '3';
26          expect(compareItems(testItem1, testItem2)).toBeTruthy();
27      });
28
29      it("Compare Items According To The Distance Criterion", function() {
30          testItem1 = {};
31          testItem1.data = {};
32          testItem2 = {};
33          testItem2.data = {};
34          testItem1.data.name = 'Foo!';
35          testItem1.data.distance = '59';
36          testItem2.data.name = 'Foo';
37          testItem2.data.distance = '0';
38          expect(compareItems(testItem1, testItem2)).toBeTruthy();
39          testItem1.data.distance = '60';
40          expect(compareItems(testItem1, testItem2)).toBeFalsy();
41      });
42
43      it("Calculate Geospatial Distance", function() {
44          expect(distance(10.112233, 10.332211, 10.332211, 10.112233)).toEqual(34317);
45          expect(distance(10.112233, 10.332211, 10.112233, 10.332211)).toEqual(0);
46      });
47
48      it("Remove Marks From String", function() {
49          expect(removeMarks("f!o-o !b'a" + '"' + 'r')).toEqual('foobar');
50      });
51  });
```

Snippet 4 testMappingAlgorithm.js

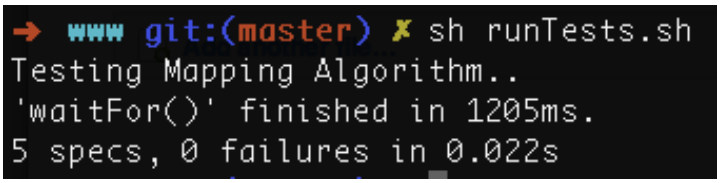


Εικόνα 5. 8 Jasmine browser output

Ο προγραμματιστής αφού έχει γράψει μία σουίτα από tests μπορεί στη συνέχεια να γράψει ένα script για να αυτοματοποιήσει τη διαδικασία. Ένα παράδειγμα με το αποτέλεσμα στο terminal (εικόνα 5.9) παρατίθεται πιο κάτω.

```
echo 'Testing Mapping Algorithm..'
phantomjs run-jasmine.js
http://localhost:8080/testMappingAlgorithm.html
```

Snippet 5 runTests.sh



Εικόνα 5. 9 Jasmine terminal output

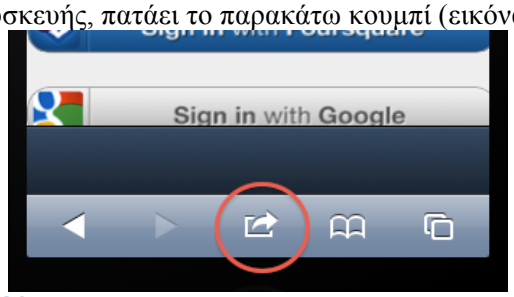
5.2 Οδηγίες εγκατάστασης της εφαρμογής

Όπως περιγράφηκε στο Κεφάλαιο 2.2 ένα σημαντικό προταίρημα των mobile web εφαρμογών είναι η πολύ εύκολη εγκατάσταση και χρήση τους τόσο σε σταθερούς υπολογιστές όσο και σε κινητές συσκευές.

Για τους υπολογιστές το μόνο που πρέπει να κάνει ο χρήστης για να χρησιμοποιήσει την εφαρμογή είναι να επισκευτεί τη διεύθυνση http://johnnyecon.appspot.com και να έχει ενεργοποιημένες τις επιλογές για την επίτρεψη της εκτέλεσης javascript και της αποθήκευσης cookies (προεπιλεγμένες ρυθμίσεις).

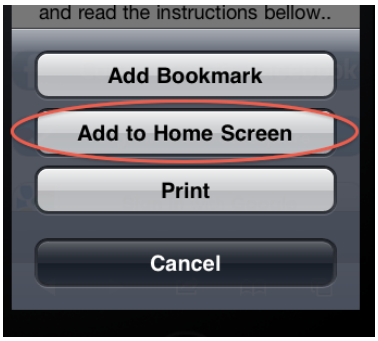
Για τις κινητές συσκευές, ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή από τον φυλλομετρητή της συσκευής, όπως και στους υπολογιστές. Συγκεκριμένα για τα iPhone και iPad υπάρχει η δυνατότητα αποθηκεύοντας της εφαρμογής στην επιφάνεια εργασίας όπως περιγράφεται παρακάτω:

- Αφού ο χρήστης επισκευτεί τη διεύθυνση http://johnnyecon.appspot.com από τον φυλλομετρητή της συσκευής, πατάει το παρακάτω κουμπί (εικόνα 5.10)



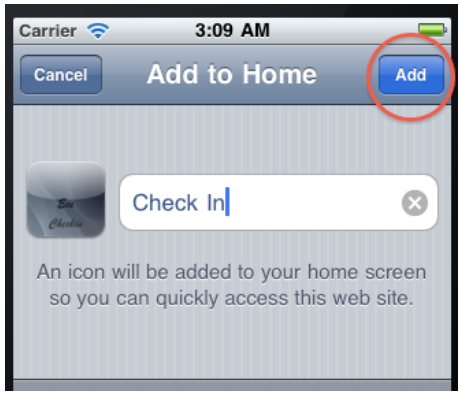
Εικόνα 5. 10 Οδηγίες εγκατάστασης εφαρμογής σε κινητή συσκευή

- Από το popup menu που θα εμφανιστεί ο χρήστης πατάει το κουμπί “Add to Home Screen” (εικόνα 5.11)



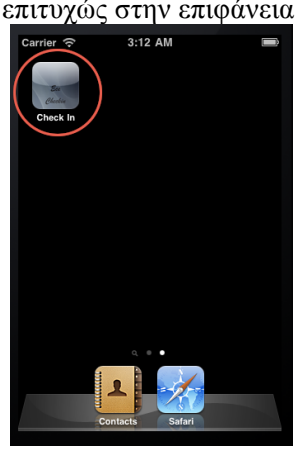
Εικόνα 5. 11 Οδηγίες εγκατάστασης εφαρμογής σε κινητή συσκευή, Add to Home Screen

- Ο χρήστης εισάγει τον τίτλο που θέλει να δώσει στην εφαρμογή και πατάει το κουμπί "Add" (εικόνα 5.12)



Εικόνα 5. 12 Οδηγίες εγκατάστασης εφαρμογής σε κινητή συσκευή, App Title

- Η εφαρμογή έχει αποθηκευτεί επιτυχώς στην επιφάνεια εργασίας (εικόνα 5.13)



Εικόνα 5. 13 Οδηγίες εγκατάστασης της εφαρμογής σε κινητή συσκευή, Ολοκλήρωση εγκατάστασης

6

Έλεγχος

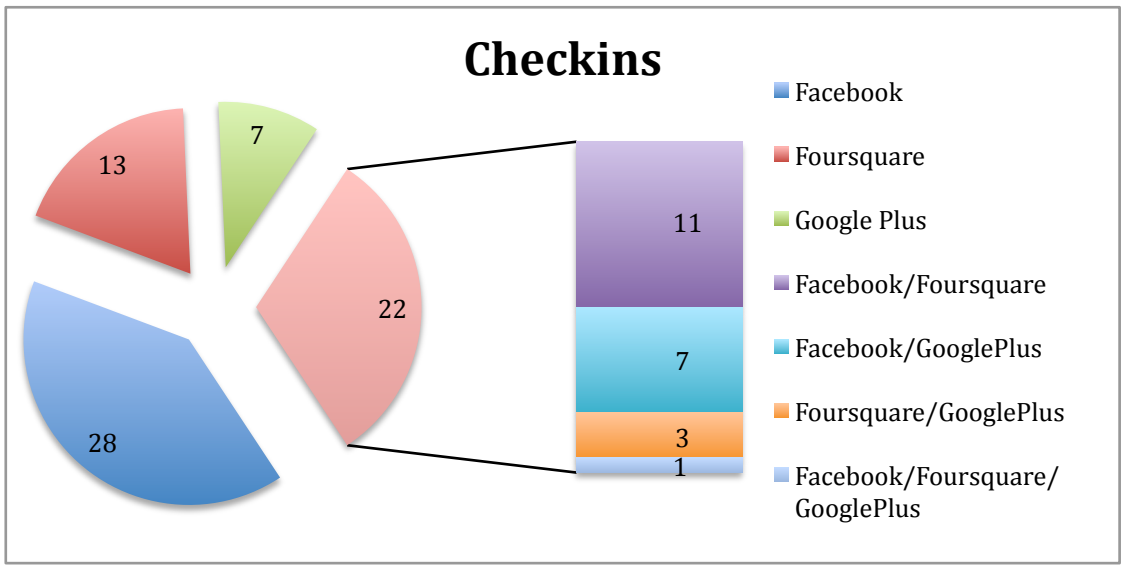
Στο κεφάλαιο αυτό παρουσιάζομαι τα στατιστικά που διεξήχθησαν μετά τη χρήση της εφαρμογής.

6.1 Μεθοδολογία ελέγχου

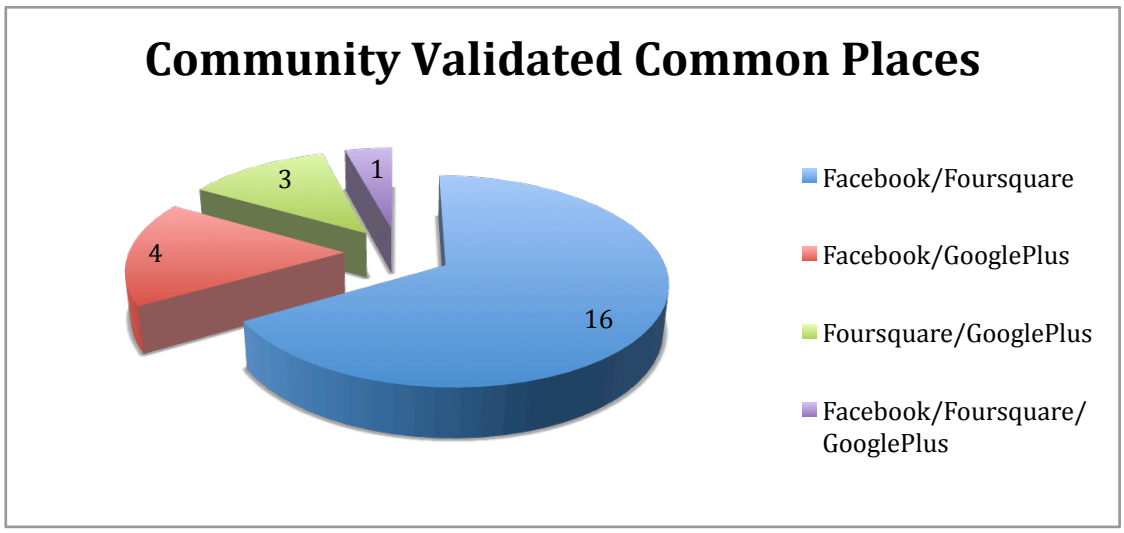
Η εφαρμογή μοιράστηκε σε είκοσι άτομα ηλικίας από είκοσι έως τριάντα ετών και χρησιμοποιήθηκε για έναν περίπου μήνα (10/5/2012 – 10/6/2012). Στη συνέχεια η εφαρμογή δοκιμάστηκε σε συνθήκες με υψηλή συχνότητα requests/second.

6.2 Παρουσίαση ελέγχου

Χρησιμοποιήθηκε η βάση δεδομένων που χρησιμοποιεί η εφαρμογή στο Google App Engine για την εξαγωγή ορισμένων στατιστικών (εικόνες 6.1 – 6.2), οι οποίες έδειξαν πως η προτίμησή των χρηστών στο Simple Checkin έναντι του Mixed Checkin ήταν εμφανής αλλά όχι ιδιαίτερα σημαντική (48 Simple Checkins - 22 Mixed Checkins). Το Facebook προτιμήθηκε από το Foursquare και το Google Plus για τη δημοσίευση της τοποθεσίας των χρηστών και η πλειοψηφία των "ψήφων" για κοινά μέρη μεταξύ δύο κοινωνικών δικτύων από τους χρήστες ανήκει στο συνδυασμό Facebook και Foursquare.

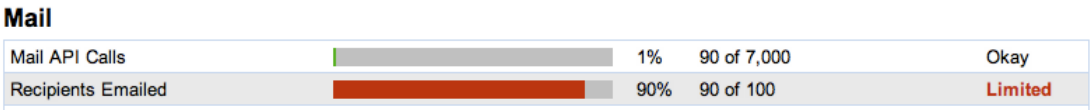


Εικόνα 6. 1 Κατανομή Checkins



Εικόνα 6. 2 Κατανομή Community Validated Common Places

Η δοκιμή σε συνθήκες με υψηλή συχνότητα requests που καταφθάνουν στο cluster έδειξε ότι το Google App Engine ανταποκρίνεται ικανοποιητικά αλλά μερικές υπηρεσίες που προσφέρει δωρεάν με συγκεκριμένο ημερίσιο όριο, όπως για παράδειγμα το Mail, μπορούν να φτάσουν στο τέλμα (εικόνα 6.3). Κάτι τέτοιο μπορεί να ξεπεραστεί με την πληρωμή κάποιου ποσού στην Google.



Εικόνα 6. 3 Google App Engine, Mail Service

7

Επίλογος

Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα που προέκυψαν από το Κεφάλαιο 6 και προτείνουμε επεκτάσεις και βελτιώσεις της εφαρμογής.

7.1 Σύνοψη και συμπεράσματα

Στη συγκεκριμένη διπλωματική εργασία αναπτύχθηκε μια web εφαρμογή για διαλειτουργική χρήση δεδομένων τοποθεσίας σε κοινωνικά δίκτυα. Ερευνήθηκαν οι μέθοδοι διοίκησης έργων λογισμικού και αναφέρθηκαν τα χαρακτηριστικά των επικρατέστερων. Επιλέχθηκε το Lean Software Development ως βάση για το έργο αυτό και υλοποιήθηκαν πολλές από τις αρχές του. Η υλοποίηση του έργου πραγματοποιήθηκε με ευέλικτες απαιτήσεις και περατώθηκε με επιτυχία εντός των αναμενόμενων χρονικών ορίων.

Τα συμπεράσματα που καταλήξαμε, σύμφωνα με τα διαγράμματα του προηγούμενου κεφαλαίου, είναι ότι οι χρήστες είναι πρόθυμοι όχι μόνο να χρησιμοποιήσουν κοινές λειτουργίες διαφορετικών κοινωνικών δικτύων αλλά και να βοηθήσουν στην ανάπτυξη της ιδέας αυτής.

7.2 Μελλοντικές επεκτάσεις

Στις ιδέες για μελλοντικές επεκτάσεις που ακολουθούν σημαντική ήταν η συνεισφορά των χρηστών που μέσω της εφαρμογής έστειλαν κάποιο Feedback:

- Ενσωμάτωση επιπλέον Κοινωνικών Δικτύων στην εφαρμογή (πχ Twitter).
- Υλοποίηση λειτουργικότητας σύνδεσης του χρήστη στην εφαρμογή και επεξεργασίας των προτιμήσεών του (πχ "Με ενδιαφέρει μόνο το Facebook και το Google").
- Εμφάνιση των πρόσφατων δημοσιεύσεων τοποθεσίας "φίλων" του χρήστη σε ξεχωριστή καρτέλα.
- Αλλαγή της οθόνης επιλογής τοποθεσιών, ώστε να περιέχει μόνο μία καρτέλα στην οποία ο χρήστης επιλέγει και αποεπιλέγει τα κοινωνικά δίκτυα που των ενδιαφέρουν.
- Ενεργοποίηση του Friend Tagging για όλα τα κοινωνικά δίκτυα που χρησιμοποιεί η εφαρμογή.
- Βελτίωση της βάσης δεδομένων, όσο αφορά τους πίνακες CommonPlaceValidate, με την προσθήκη των πινάκων CommunityVote, Place και ενός ενδιάμεσου CommunityVotePlace ώστε να έιναι εύκολα υλοποιήσημη και κατανοητή η σύνδεση many to many μεταξύ των CommunityVote και Place.

8

Βιβλιογραφία

http://ofps.oreilly.com/titles/9780596805784/ch03_id35816678.html

https://github.com/velesin/jasmine-jquery

http://www.ambysoft.com/surveys/

Curt Hibbs, Steve Jewett, Mike Sullivan: 'The Art Of Lean Software Development'

IEEE 26th Software Engineering Workshop, 'Improving Software Investments through Requirements Validation', 2001

XP 2002 Conference, Standish Group

http://www.oneclickcustomers.com/infographics/google-vs-facebook-infographic.html

http://en.wikipedia.org/wiki/Foursquare_(website)

http://www.mech.upatras.gr/~nikos/mis-ii/notes/notes-01.pdf

http://pypi.python.org/pypi/geomodel,

http://code.google.com/p/geomodel/wiki/Usage

http://www.translatum.gr/converter/greek-transliteration.htm

http://docs.sencha.com/touch/2-0/#!/api

https://developer.foursquare.com/overview/auth.html

http://www.onlinesolutionsdevelopment.com/blog/mobile-development/creating-a-sencha-touch-mvc-application-from-scratch-part-3/

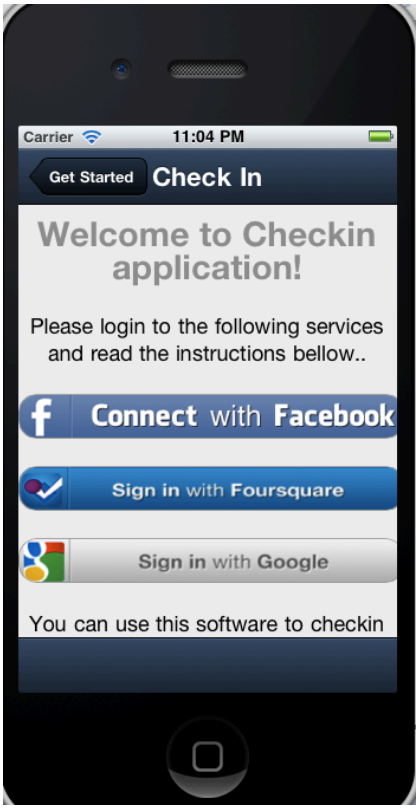
https://developers.facebook.com/docs/guides/mobile/web/#login

http://www.jeffbullas.com/2012/04/23/48-significant-social-media-facts-figures-and-statistics-plus-7-infographics/

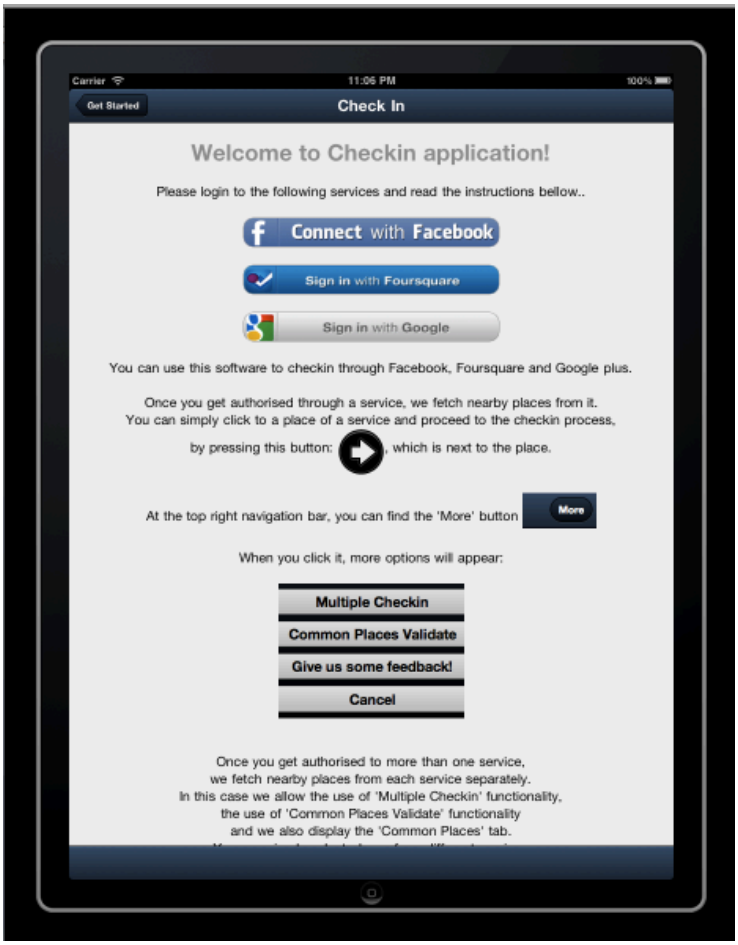
http://mobiledevelopertips.com/cocoa/launching-your-own-application-via-a-custom-url-scheme.html

9

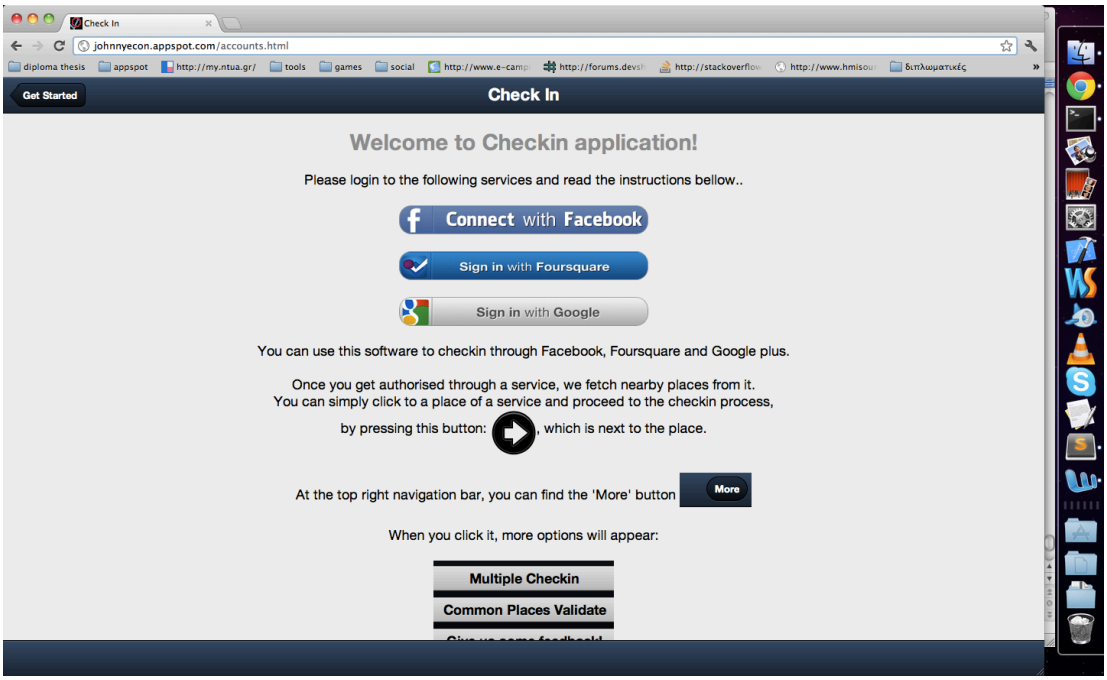
Παράρτημα



Εικόνα 9. 1 iPhone με εγκατεστημένη την εφαρμογή EceCheckin



Εικόνα 9. 2 iPad με εγκατεστημένη την εφαρμογή EceCheckin



Εικόνα 9. 3 Browser ενός υπολογιστή με την εφαρμογή EceCheckin

Εικόνα 9. 4 Mockup της εφαρμογής EceCheckin