ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αποστολή χαρτών και γεωγραφικών πληροφοριών σε κινητές συσκευές

Κυριακή Ν. Ανδρή

Γεωργία Λ. Χαλμούκη

Επιβλέποντες: Ευστάθιος Χατζηευθυμιάδης, Επικούρος Καθηγητής ΕΚΠΑ
Αναστάσιος Ιωαννίδης, Επιστημονικός Συνεργάτης ΕΚΠΑ

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2009

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αποστολή χαρτών και γεωγραφικών πληροφοριών σε κινητές συσκευές

Κυριακή Ν. Ανδρή

Α.Μ.: 1115200200207

Γεωργία Λ. Χαλμούκη

Α.Μ.: 1115200300152

ΕΠΙΒΛΕΠΟΝΤΕΣ:

Ευστάθιος Χατζηευθυμιάδης, Επικούρος Καθηγητής ΕΚΠΑ
Αναστάσιος Ιωαννίδης, Επιστημονικός Συνεργάτης ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται την αποστολή χαρτών και γεωγραφικών πληροφοριών σε κινητές συσκευές. Ειδικότερα, στόχος είναι η συλλογή και επεξεργασία πληροφοριών με βάση τα δεδομένα εισόδου και τις επιλογές του χρήστη, η προσωρινή αποθήκευση τους και η οργάνωση τους σε ένα μήνυμα SMS (Short Message Service) με σκοπό την αποστολή τους σε κινητή συσκευή μέσω πύλης δικτύου (gateway).

Στα πλαίσια της εργασίας δημιουργήθηκε μια διαδικτυακή εφαρμογή. Η εφαρμογή αυτή χρησιμοποιεί τις διεπαφές προγραμματισμού εφαρμογών της Google, Google Static Maps API και Google Maps API, οι οποίες παρέχουν έναν αριθμό εργαλείων για την προβολή και το χειρισμό χαρτών. Έτσι, ο χρήστης μπορεί να επιλέξει με visual τρόπο την περιοχή του χάρτη που επιθυμεί να αποστείλει σε κινητή συσκευή. Επιπλέον χρησιμοποιείται η υπηρεσία Google App Engine, που προσφέρει η Google, για αποθήκευση των στοιχείων μηνύματος του χρήστη έτσι ώστε να μπορούν να ανακτηθούν μετέπειτα. Τέλος γίνεται η αποστολή του μηνύματος με τη βοήθεια ενός παροχέα υπηρεσιών.

Επίσης εκτελέστηκε σενάριο αποστολής μηνύματος σε κινητή συσκευή για την απόδειξη ορθής λειτουργίας.

Στο τελευταίο κεφάλαιο παρατίθεται ο πηγαίος κώδικας που αναπτύχθηκε για την εκπόνηση της εργασίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνολογίες διαδικτύου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: διαδικτυακή εφαρμογή, χάρτες google, google app engine, πρωτόκολλο μεταφοράς υπερκειμένου, wap push μήνυμα

ABSTRACT

The current BSc Thesis deals with sending maps and geographical information to mobile devices. Specifically, our objective is the collection and processing of information that is exported from the user's data of entry and selections, the temporary storage of them and the creation of an SMS (Short Message Service) in order to send it to a mobile appliance via gateway.

In the context of work a web application was created. This application uses the application programming interfaces (APIs) of Google, Google Maps API and Google Static Maps API, that provide a number of utilities for displaying and manipulating maps. Thus, the user is allowd to select in a visual way the area of the map that he wants to send to a mobile device. In addition, the service Google App Engine of Google is used for storaging the elements of user's message in order to retrieve them later. Finally, the message is sent through a service provider.

Also, an example of sending a message to a mobile device is presented as proof of concept.

In last chapter, the source code of the implementation is demonstrated.

SUBJECT AREA: Web technologies

KEYWORDS: web application, google maps, google app engine,
HTTP protocol, wap push message

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να ευχαριστήσουμε θερμά τον καθηγητή μας, κύριο Ευστάθιο Χατζηευθυμιάδη, για την εμπιστοσύνη που μας έδειξε αναθέτοντας μας την παρούσα πτυχιακή εργασία καθώς και για την καθοδήγηση και την πολύτιμη βοήθεια του καθ' όλη τη διάρκεια της εκπόνησής της. Επίσης, θα θέλαμε να ευχαριστήσουμε ιδιαιτέρως τον κύριο Αναστάσιο Ιωαννίδη, ο οποίος με τις γνώσεις και την εμπειρία του πάνω στο αντικείμενο αλλά και με τη συνεχή βοήθεια που μας παρείχε, συνέβαλε καθοριστικά στην εξέλιξη και στην ολοκλήρωση της εργασίας μας.

Κυριακή Ν. Ανδρή
Γεωργία Λ. Χαλμούκη

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια του Προπτυχιακού Προγράμματος Σπουδών του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών το ακαδημαϊκό έτος 2008-2009. Αντικείμενο μελέτης της είναι η αποστολή χαρτών και γεωγραφικών πληροφοριών σε κινητές συσκευές μέσω πύλης δικτύου.

Το Διαδίκτυο και οι εφαρμογές πάνω σε αυτό είναι ένας τομέας της Πληροφορικής με μεγάλο εύρος, ενδιαφέρον και συνεχώς εξελισσόμενος. Αυτά τα χαρακτηριστικά σε συνδυασμό με την επιθυμία μας για την ενασχόληση με αυτούς τους τομείς αποτέλεσαν καθοριστικούς παράγοντες για την επιλογή αυτής της πτυχιακής.

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

Η ταχύτατα αναπτυσσόμενη τεχνολογία της κινητής επικοινωνίας και των ασύρματων δικτύων στις μέρες μας, παρέχει τη δυνατότητα μετάδοσης της πληροφορίας οπουδήποτε και οποτεδήποτε με ελάχιστο κόστος. Διάφορες συσκευές ανεξαρτήτου μεγέθους, όπως τα κινητά τηλέφωνα, διαθέτοντας ασύρματη σύνδεση μπορούν να επικοινωνούν με δίκτυα με μεγάλη ευκολία και παράλληλα έχουν το πλεονέκτημα της κινητικότητας και μεταφοράς σε διάφορα σημεία. Αυτή είναι και η έννοια του διάχυτου και κινητού υπολογισμού, που σύμφωνα με τους Tomasz Imieliński και Henry F. Korth [1], ο τελευταίος ανοίγει μια νέα κλάση εφαρμογών οι οποίες περιλαμβάνουν την αρμονική συνεργασία desktop υπολογιστών και κινητών συσκευών προσφέροντας ευκολία χρήσης και ενοποίηση των προσφερόμενων υπηρεσιών.

Επίσης, η τεράστια ανάπτυξη των κινητών συσκευών που επιτρέπουν τη διαχείριση δεδομένων ανοίγει νέους ορίζοντες στις υπηρεσίες ώθησης πληροφορίας (push). Οι νέες ιδιότητες των κινητών τηλεφώνων παρέχουν τη δυνατότητα στους χρήστες να επικοινωνούν με νέους τρόπους και να έχουν πρόσβαση σε νέες εφαρμογές και σε πληροφορία οπουδήποτε και οποτεδήποτε. Η επιτυχία της Υπηρεσίας Σύντομου Μηνύματος (Short Message Service – SMS) επεκτείνεται στην επικοινωνία χρησιμοποιώντας push μηνύματα, με τα οποία ένας χρήστης στέλνει έναν υπερσύνδεσμο προς πολυμεσικό περιεχόμενο μέσω SMS [2]. Αυτός είναι ένας εναλλακτικός τρόπος δημοσίευσης περιεχομένου, εκτός από την Υπηρεσία Πολυμεσικών Μηνυμάτων (Multimedia Messaging Service – MMS), ο οποίος απαιτεί πρόσβαση στο διαδίκτυο.

Η παρούσα πτυχιακή εργασία έχει να κάνει με την ανάπτυξη διαδικτυακής εφαρμογής σε συνδυασμό με ανταλλαγή δεδομένων με κινητές συσκευές. Μια διαδικτυακή εφαρμογή (web application) είναι μια εφαρμογή που προσπελαύνεται μέσω του φυλλομετρητή ιστοσελίδων (web browser) πάνω σε ένα δίκτυο, είτε στο διαδίκτυο (Internet) είτε σε ενδοδίκτυο (intranet). Οι διαδικτυακές εφαρμογές είναι δημοφιλείς [3] εξ' αιτίας της πανταχού

παρουσίας των φυλλομετρητών και της ευκολίας χρήσης τους ως πελάτες (clients). Η ικανότητα της αναβάθμισης και συντήρησης των διαδικτυακών εφαρμογών χωρίς την εγκατάσταση προγραμμάτων σε πιθανόν χιλιάδες υπολογιστές-πελάτες είναι ο βασικότερος λόγος της εξάπλωσής τους. Τα πλεονεκτήματά τους είναι οι μικρές απαιτήσεις στα χαρακτηριστικά του πελάτη, όπως σχεδόν καθόλου χώρος στο σκληρό δίσκο, η αυτόματη αναβάθμισή τους και η εύκολη ενσωμάτωση τους σε διαδικτυακές διαδικασίες. Επιπλέον παρέχουν συμβατότητα με τα περισσότερα λειτουργικά συστήματα. Στα αρνητικά χαρακτηριστικά των διαδικτυακών εφαρμογών συγκαταλέγονται τα προβλήματα συμβατότητας που δημιουργούνται κατά την μεταφορά αρχείων εξ' αιτίας τους ποικίλλους δημιουργούς εγγράφων. Επίσης καθώς οι διαδικτυακές εφαρμογές βασίζονται στην πρόσβαση των αρχείων που βρίσκονται σε απομακρυσμένους υπολογιστές - εξυπηρετητές στο διαδίκτο, είναι πιθανόν να μην είναι διαθέσιμες όταν η σύνδεση διακόπτεται ή αντιμετωπίζει προβλήματα.

Στο Κεφάλαιο 2 της Πτυχιακής Εργασίας παρουσιάζονται οι διεπαφές προγραμματισμού εφαρμογων Google Maps API και Google Static Maps API της Google, οι οποίες παρέχουν τη δυνατότητα ενσωμάτωσης χαρτών και γεωγραφικών πληροφοριών στις ιστοσελίδες.

Στο Κεφάλαιο 3 γίνεται ανάλυση της υπηρεσίας Google App Engine, η οποία επιτρέπει την φιλοξενία διαδικτυακών εφαρμογών.

Στο Κεφάλαιο 4 γίνεται μια εκτενής αναφορά στην τεχνολογία wap push, η οποία είναι ένας εναλλακτικός τρόπος δημοσίευσης περιεχομένου.

Στο Κεφάλαιο 5 εξηγείται η αρχιτεκτονική της εφαρμογής και παρουσιάζονται τα βήματα που ακολουθήθηκαν.

Στο Κεφάλαιο 6 αναλύονται τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2

ΧΡΗΣΗ ΤΟΥ GOOGLE MAPS API ΣΕ ΔΙΑΔΙΚΤΥΑΚΕΣ ΕΦΑΡΜΟΓΕΣ

2.1 *Διεπαφή Προγραμματισμού Εφαρμογών Google Maps*

Το Google Maps API [4] είναι μια διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface – API) της Google το οποίο δίνει τη δυνατότητα ενσωμάτωσης ενός δυναμικού χάρτη Google σε ιστοσελίδες (web σελίδες) με τη χρήση της javascript. Το API παρέχει έναν αριθμό εργαλείων για το χειρισμό χαρτών και την προσθήκη περιεχομένου σε αυτούς μέσω διαφόρων υπηρεσιών, όπως και στην επίσημη ιστοσελίδα του Google Maps (http://maps.google.com), επιτρέποντας έτσι τη δημιουργία εύρωστων εφαρμογών που εκμεταλλεύονται γεωγραφικές πληροφορίες.

Το Google Maps API είναι μια δωρεάν υπηρεσία, διαθέσιμη για οποιοδήποτε ιστότοπο (web site) που είναι ελεύθερος για το κοινό. Το μόνο που χρειάζεται είναι ένα κλειδί API (API key) που μπορεί να ζητηθεί εφόσον υπάρχει λογαριασμός στη Google και αυτομάτως συνδέεται με το λογαριασμό. Κάθε κλειδί είναι έγκυρο μόνο για ένα συγκεκριμένο όνομα τομέα.

Βασικό στοιχείο οποιασδήποτε Google Maps API εφαρμογής είναι ο ίδιος ο χάρτης, ο οποίος φορτώνεται, αρχικοποιείται με συγκεκριμένες διαστάσεις (ύψος και πλάτος), κέντρο(γεωγραφικό πλάτος, γεωγραφικό μήκος), επίπεδο εστίασης και τύπο (κανονικό, δορυφορικό, υβριδικό). Στη συνέχεια είναι δυνατή η αλληλεπίδραση με το χάρτη μέσω διαφόρων μηχανισμών όπως είναι τα events, τα controls, τα overlays αλλά και συναρτήσεις επεξεργασίας των χαρακτηριστικών του, τα οποία αναλύονται παρακάτω:

2.1.1 Γεγονότα (Events)

Το Google Maps API εμπεριέχει δηλώσεις γεγονότων (events) [5] τα οποία εμφανίζονται μέσω των φυλλομετρητών ιστοσελίδων (web browsers) πάνω σε αντικείμενα του Google Maps. Ας σημειωθεί ωστόσο ότι τα γεγονότα αυτά είναι διαφορετικά από τα standard γεγονότα πάνω στο

Μοντέλο Αντικειμένου Εγγράφου (Document Object Model – DOM), το οποίο είναι μια προγραμματιστική διεπαφή που επιτρέπει σε προγράμματα και εκτελέσιμα σενάρια δράσης (scripts) την ενημέρωση του περιεχομένου και του στυλ όλου του εγγράφου με δυναμικό τρόπο. Έτσι το Google Maps API δημιουργεί και διαχειρίζεται δικά του γεγονότα ενώ το Μοντέλο Αντικειμένου Εγγράφου δημιουργεί άλλα γεγονότα και τα επεξεργάζεται σύμφωνα με το μοντέλο αντικειμένου που χρησιμοποιεί ο συγκεκριμένος φυλλομετρητής. Επιπλέον, το Google Maps API έχει το πλεονέκτημα ότι προσφέρει μηχανισμούς που ακούνε (listen) και απαντάνε (respond) στα DOM γεγονότα ανεξάρτητα από τις ιδιαιτερότητες του κάθε φυλλομετρητή. Παραδείγματα γεγονότων του Google Maps API είναι το κλικ του ποντικιού, το διπλό κλικ, η κίνηση του ποντικιού τα οποία σηματοδοτούν την έναρξη κάποιων διαδικασιών που περιγράφονται στον κώδικα javascript. Εξάλλου η javascript είναι μια scripting γλώσσα που «καθοδηγεί γεγονότα» (“event driven”) εξελισσόμενα στους φυλλομετρητές.

2.1.2 Ρυθμιστές (Controls)

Όπως μπορεί να διαπιστώσει κανείς και στη σελίδα του Google Maps (http://maps.google.com) υπάρχουν διάφορα στοιχεία στη διεπιφάνεια με το χρήστη (user interface) που επιτρέπουν την αλληλεπίδρασή του χρήστη με το χάρτη. Τα στοιχεία αυτά είναι γνωστά ως ρυθμιστές (controls) και μπορούν να περιληφθούν στις Google Maps εφαρμογές σε οποιαδήποτε σημείο πάνω στο χάρτη. Τα κυριότερα είναι:

- GLargeMapControl3D: Τοποθετεί ένα μεγάλο τρισδιάστατο κουμπί ελέγχου εστίασης και κίνησης του χάρτη, όπως χρησιμοποιείται τωρα στη σελίδα του Google Maps. Εμφανίζεται στην πάνω αριστερή γωνία του χάρτη αυτόματα.
- GLargeMapControl: Τοποθετεί ένα μεγάλο απλό κουμπί ελέγχου εστίασης και κίνησης του χάρτη. Εμφανίζεται στην πάνω αριστερή γωνία του χάρτη αυτόματα.

- GSmallMapControl : Τοποθετεί ένα μικρό κουμπί ελέγχου εστίασης και κίνησης του χάρτη. Εμφανίζεται στην πάνω αριστερή γωνία του χάρτη αυτόματα.
- GSmallZoomControl3D: Τοποθετεί ένα μικρό τρισδιάστατο κουμπί ελέγχου εστίασης (χωρίς έλεγχο κίνησης) του χάρτη. Εμφανίζεται στην πάνω αριστερή γωνία του χάρτη αυτόματα.
- GSmallZoomControl: Τοποθετεί ένα μικρό κουμπί ελέγχου εστίασης (χωρίς έλεγχο κίνησης) του χάρτη που χρησιμοποιείται στα μικρά παράθυρα που εμφανίζονται στο χάρτη Google για επεξήγηση των οδικών οδηγιών.
- GScaleControl: Τοποθετεί μια κλίμακα με στάδια μεγέθυνσης του χάρτη.
- GMapTypeControl: Τοποθετεί κουμπιά που επιτρέπουν στο χρήστη να διαλέγει μεταξύ τύπων του χάρτη (όπως δορυφορικός, εδαφικός κ.ά.)
- GHierarchicalMapTypeControl: Τοποθετεί κουμπιά επιλογών (checkboxes) για ιεραρχική προτίμηση των τύπων του χάρτη.
- GOverviewMapControl: Τοποθετεί μια πτυσσόμενη επισκόπηση του χάρτη στη γωνία της οθόνης.
- GNavLabelControl: Τοποθετεί μια δυναμική ετικέτα που επισημαίνει τη «διεύθυνση» της συγκεκριμένης περιοχής χάρτη ανάλογα με το επίπεδο εστίασης.

Επιπλέον υπάρχει η δυνατότητα δημιουργίας ειδικών ρυθμιστικών στοιχείων από τον προγραμματιστή.

2.1.3 Overlays

Τα overlays είναι αντικείμενα πάνω στο χάρτη που αντιστοιχούν σε συγκεκριμένες συντεταγμένες και γι' αυτό μετακινούνται ανάλογα όταν μετακινείται ή αλλάζει εστίαση ο χάρτης. Χρησιμοποιούνται για το σχεδιασμό

και την επίδειξη σημείων, γραμμών ή ολόκληρων περιοχών. Το Google Maps API διαθέτει πολλά είδη overlays, τα οποία αναλύονται παρακάτω:

- Σημεία

Αναπαριστώνται με τη χρήση δεικτών (markers) πάνω στο χάρτη και συχνά προβάλλουν μια σχετική εικόνα στο σημείο που επιδεικνύουν. Οι markers είναι σχεδιασμένοι να είναι διαδραστικοί, δηλαδή επάνω τους πραγματοποιούνται γεγονότα, όπως ένα κλικ ή δεξί κλικ, και συχνά προβάλεται παράλληλα με αυτούς ένα παράθυρο πληροφοριών για το σημείο. Μια επιπλέον ιδιότητα τους είναι ότι μπορεί να είναι συρόμενοι (draggable) από ένα σημείο σε άλλο στο χάρτη. Επίσης δυνατή είναι η επεξεργασία και αλλαγή της εικόνας (icon) που αντιστοιχεί σε κάθε δείκτη. Η εικόνα αποτελείται από μια εικόνα στο προσκήνιο και μια άλλη ως σκια τοποθετημένες σε συγκεκριμένες θέσεις μεταξύ τους. Παρότι οι δείκτες έχουν μεγάλη χρησιμότητα, η προσθήκη πολλών από αυτούς σε ένα χάρτη Google μπορεί να έχει αρνητική επίδραση στην ταχύτητα απεικόνισης του χάρτη (map rendering) ειδικά σε συγκεκριμένα επίπεδα εστίασης. Λύση σε αυτό δίνει ο Διαχειριστής Δεικτών (Marker Manager), ο οποίος έχει την ικανότητα να παρακολουθεί ποιοί δείκτες είναι ορατοί σε ποιά επίπεδα εστίασης και να στέλνει μόνο τους απαραίτητους στο χάρτη για σχεδιαστικούς σκοπούς. Δηλαδή προσθέτει ή αφαιρεί δυναμικά δείκτες από το χάρτη με αποτέλεσμα την αύξηση της ταχύτητας απεικόνισης του χάρτη και της μείωσης των άχρηστων οπτικών στοιχείων.

- Γραμμές

Αναπαριστώνται με τη χρήση των polylines (που ουσιαστικά είναι μια ακολουθία συνδεδεμένων σημείων) πάνω στο χάρτη, για τις οποίες μπορεί να καθοριστεί χρώμα, πάχος και επίπεδο διαφάνειας. Χωρίζονται σε τρεις κατηγορίες: τις γραμμικές (drawing), τις γεωδεσικές (geodesic) και τις κωδικοποιημένες (encoded).

Οι γραμμικές polylines εμφανίζονται ως ευθύγραμμα τμήματα στο χάρτη και για την απεικόνιση τους το Google Maps API εκμεταλλεύεται πιθανές σχεδιαστικές ικανότητες του φυλλομετρητή, αλλιώς γίνεται αίτηση για μια

εικόνα polyline από τους εξυπηρετητές (servers) της Google. Οι γεωδεσικές polylines εμφανίζονται ως καμπυλωτές γραμμές λαμβάνοντας υπόψιν τους την καμπυλότητα της γης. Οι κωδικοποιημένες polylines είναι συνήθως μεγαλύτερες και πιο περίπλοκες από τις υπόλοιπες και αποτελούνται από πολλά ευθύγραμμα τμήματα που συνδέονται με τυχαίο τρόπο μεταξύ τους. Γι' αυτό το λόγο απαιτούν περισσότερη μνήμη και χρόνο για να απεικονισθούν. Έχουν όμως το πλεονέκτημα ότι μπορούν να καθορίζουν πόσο λεπτομερής θα είναι μια polyline σε ένα δοσμένο επίπεδο εστίασης καθώς δίνουν τη δυνατότητα να αγνοούνται τμήματα της γραμμής σε συγκεκριμένες ομάδες επιπέδων εστίασης.

- Περιοχές

Αναπαριστώνται με τη χρήση είτε πολυγώνων (polygons) πάνω στο χάρτη, αν η περιοχή έχει τη μορφή αυθαίρετου σχήματος, είτε με τη χρήση ground overlays, αν η περιοχή έχει τη μορφή ορθογώνιου σχήματος. Τα πολύγωνα είναι παρόμοια με τις polylines αφού και αυτά είναι μια ακολουθία σημείων, προσθέτωντας ότι σχηματίζουν κύκλο και μπορούν να πάρουν οποιαδήποτε μορφή. Τα ground overlays είναι συχνά χρήσιμα για περιοχές που τα άκρα τους στοχεύουν κάθετα σε σημεία στο χάρτη.

- Ο ίδιος ο χάρτης

Αναπαρίσταται με τη χρήση του tile overlay, κάτι το οποίο μπορεί να τροποποιηθεί μέσω του Google Maps API. Τα διαθέσιμα tiles δεν καλύπτουν όλες τις περιοχές σε όλα τα επίπεδα εστίασης. Για παράδειγμα πολλές περιοχές στον Ειρηνικό Ωκεανό δεν εμφανίζονται σε υψηλά επίπεδα εστίασης, ενώ το Μανχάταν προσφέρει πολύ λεπτομερείς εικόνες στον δορυφορικό τύπο χάρτη.

Αξίζει να σημειωθεί ότι στο μικρότερο επίπεδο εστίασης (επίπεδο 0), ένα tile αναπαριστά ολόκληρη τη γη. Αν N είναι το επίπεδο εστίασης, τότε το Google Maps διαιρεί το tile ολόκληρης της γης σε 4^N tiles και προβάλει το κατάλληλο.

- Το παράθυρο πληροφοριών (info window)

Αποτελεί ένα ειδικό overlay. Προστίθεται στο χάρτη αυτόματα και σε κάθε χάρτη αντιστοιχεί ένα μόνο αντικείμενο παραθύρου πληροφοριών.

2.1.4 Προσφερόμενες υπηρεσίες του Google Maps API

Πρόσθετες υπηρεσίες τις οποίες προσφέρει το Google Maps API είναι η μετατροπή πραγματικών διευθύνσεων ανά τον πλανήτη σε γεωγραφικές συντεταγμένες (geocoding), κάτι το οποίο είναι χρήσιμο στην τοποθέτηση δεικτών, αλλά και το αντίστροφο (reverse geocoding). Επιπλέον το Google Maps API είναι εξοπλισμένο με κρύφη μνήμη (cache memory) από τη μεριά του πελάτη (client), η οποία αποθηκεύει διάφορες απαντήσεις σε μετατροπές διευθύνσεων, επιτρέποντας ταχύτερη εξυπηρέτηση σε περίπτωση που ζητηθεί η ίδια μετατροπή διεύθυνσης στο μέλλον.

Μια άλλη υπηρεσία είναι η άποψη από δρόμο (street view), η οποία παρέχει πανοραμική θέα 360^{o} περιοχών για τις οποίες υπάρχουν διαδραστικές εικόνες. Κάθε οπτική της εικόνας χαρακτηρίζεται από την τοποθεσία και τον προσανατολισμό και είναι μοναδική.

Επιπλέον υπηρεσία είναι η προσθήκη του Google Earth στις Maps API εφαρμογές, το οποίο προβάλει τον χάρτη Google τρισδιάστατα.

Η μπάρα αναζήτησης Google είναι άλλο ένα χρήσιμο εργαλείο με το οποίο αναζητούνται πόλεις, πανεπιστημια, επιχειρήσεις κ.ά. Η μπάρα αναζήτησης είναι φιλική προς το χρήστη και συμπεριλαμβάνει στα αποτελέσματα της διαφημίσεις με σκοπό το οικονομικό κέρδος αν προστεθεί στις Google Maps εφαρμογές.

Δίνεται η δυνατότητα επιπλέον να περιληφθεί στην εφαρμογή ο υπολογισμός οδηγιών για τη μετάβαση από ένα μέρος σε ένα άλλο, χρησιμοποιώντας μια ποικιλία μέσων μεταφοράς).

2.2 *Διεπαφή Προγραμματισμού Εφαρμογών Google Static Maps*

Το Google Static Maps API [6] είναι μια διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface – API) της Google το οποίο δίνει τη δυνατότητα ενσωμάτωσης μιας εικόνας χάρτη Google σε ιστοσελίδα (webpage) χωρίς τη χρήση javascript ή οποιασδήποτε άλλης δυναμικής γλώσσας. Η υπηρεσία Google Static Map δημιουργεί ένα Google χάρτη βασιζόμενη στις παραμέτρους του URL που στέλνονται με HTTP κλήση (HTTP request) και επιστρέφει το χάρτη ως εικόνα (μορφή gif, png ή jpeg) έτοιμη να απεικονισθεί στην ιστοσελίδα. Για κάθε request πρέπει να δηλώνονται μέσω του Google Static Maps API URL ορισμένες παράμετροι, κάποιες από τις οποίες είναι υποχρεωτικές ενώ άλλες είναι προαιρετικές, και οι οποίες περιγράφονται παρακάτω:

- center (απαραίτητο αν δεν υπάρχουν markers): Καθορίζει το κέντρο του χάρτη, ισαπέχον σημείο από όλες τις γωνίες του χάρτη, και αποτελείται από ένα ζευγάρι {latitude, longitude} ({γεωγραφικό πλάτος, γεωγραφικό μήκος}) χωρισμένο με κόμμα, το οποίο και αντιπροσωπεύει μια μοναδική τοποθεσία σε όλη τη γη.
- zoom (απαραίτητο αν δεν υπάρχουν markers): Καθορίζει το zoom level του χάρτη και παίρνει τιμές από 0-19
- size (απαραίτητο): Καθορίζει τις ορθογώνιες διαστάσεις της εικόνας του χάρτη και παίρνει τιμές της μορφής valuexvalue, όπου η πρώτη τιμή δηλώνει τα οριζόντια pixels ενώ η δεύτερη τα κάθετα pixels. Το λογότυπο “Powered by Google” αυξομειώνεται αυτόματα.
- format (προαιρετικό): Καθορίζει το format της εικόνας του χάρτη. Εξ’ ορισμού το Static Maps API δημιουργεί εικόνες gif, αλλά υποστηρίζονται και άλλοι τύποι, όπως jpeg, png.
- maptype (προαιρετικό): Καθορίζει το είδος χάρτη που θα εισαχθεί στη σελίδα ανάμεσα σε satellite, terrain, hybrid και mobile.
- markers (προαιρετικό): Καθορίζει έναν ή περισσότερους markers που τοποθετούνται στο χάρτη σε συγκεκριμένες θέσεις. Οι δηλώσεις των

markers διαχωρίζονται μεταξύ τους με τον χαρακτήρα pipe (|) και κάθε marker μπορεί να αναπαρασταθεί με χαρακτηριστικά όπως:

- latitude (απαραίτητο): Καθορίζει το γεωγραφικό πλάτος του marker
- longitude (απαραίτητο): Καθορίζει το γεωγραφικό μήκος του marker
- size (προαιρετικό): Καθορίζει το μέγεθος του marker και οι τιμές μπορεί να είναι tiny, mid, small, normal
- color (προαιρετικό): Καθορίζει το χρώμα του marker και οι τιμές μπορεί να είναι black, brown, green, purple, yellow, blue, gray, orange, red, white
- alphanumeric-character (προαιρετικό): Καθορίζει ένα αλφαριθμητικό χαρακτήρα από το σύνολο {a-z, 0-9} για το δείκτη

- path (προαιρετικό): Καθορίζει ένα μονοπάτι ανάμεσα σε δυο ή περισσότερα σημεία στο χάρτη διαχωρισμένα μεταξύ τους με τον χαρακτήρα pipe (|).
- span (προαιρετικό): Καθορίζει ένα viewport (από ζευγάρια {γεωγραφικό πλάτος, γεωγραφικό μήκος}) και η υπηρεσία static map ρυθμίζει το επίπεδο εστίασης (zoom level) ανάλογα για να περιλαμβάνει την span τιμή από το κέντρο του χάρτη. Αν έχει καθοριστεί το zoom, το span αγνοείται.
- frame (προαιρετικό): Προσθέτει ένα καθορισμένο μπλε πλαίσιο στην εικόνα του χάρτη και παίρνει τιμές true ή false.
- hl (προαιρετικό): Καθορίζει τη γλώσσα των ετικετών στο χάρτη.
- key (απαραίτητο): Καθορίζει το κλειδί για το όνομα τομέα (domain name) πάνω στο οποίο γίνεται η κλήση του URL. Κλειδιά δίνονται δωρεάν σε όλους τους εγγεγραμένους χρήστες στο Google.
- sensor (απαραίτητο): Καθορίζει αν η εφαρμογή που ζητά να ενσωματώσει το χάρτη, χρησιμοποιεί σένσορα για να εντοπίσει τη θέση του χρήστη και παίρνει τιμές true ή false.

Ένα παράδειγμα που συνοψίζει τα παραπάνω δείχνει ένα URL και την αντίστοιχη static map εικόνα σε πραγματικό μέγεθος:

http://maps.google.com/staticmap?center=40.714728,-73.998672&zoom=14&size=248x248&maptype=mobile\&markers=40.702147,-74.015794,blues|40.711614,-74.012318,greeng|40.718217,-73.998284,redc\&key=MAPS_API_KEY&sensor=false

Εικόνα 1: Google Static Map URL και η αντίστοιχη εικόνα χάρτη που επιστρέφεται μετά το http request

ΚΕΦΑΛΑΙΟ 3

ΦΙΛΟΞΕΝΙΑ ΔΙΑΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ ΣΤΟ GOOGLE APP ENGINE

3.1 *Τι είναι το Google App Engine*

Το Google App Engine [7] είναι μία πλατφόρμα για την ανάπτυξη και τη φιλοξενία (hosting) διαδικτυακών εφαρμογών (web applications) σε κέντρα δεδομένων διοικούμενα από τη Google. Αρχικά εκδόθηκε σε δοκιμαστική μορφή (beta version) τον Απρίλη του 2008.

Το Google App Engine ανήκει στην τεχνολογία “Cloud Computing” [8]. Πρόκειται για ένα είδος παράλληλου και κατανεμημένου συστήματος το οποίο αποτελείται από μία συλλογή εικονικών υπολογιστών, συνδεδεμένων μεταξύ τους, οι οποίοι μπορούν να τροφοδοτούνται δυναμικά και να παρουσιάζονται ως ένας ή περισσότεροι ενοποιημένοι υπολογιστικοί πόροι. Οι παρεχόμενες υπηρεσίες καθορίζονται από μία συμφωνία σε επίπεδο υπηρεσιών (Service-Level Agreement - SLA) μεταξύ του παρόχου και του πελάτη. Σκοπός είναι ο χρήστης να μπορεί να αξιοποιεί όσο το δυνατόν περισσότερους υπολογιστικούς πόρους, χωρίς να χρειάζεται να γνωρίζει ή να διαχειρίζεται τις πολύπλοκες πληροφορίες που διέπουν αυτή την πλατφόρμα. Άλλες cloud-based πλατφόρμες περιλαμβάνουν υπηρεσίες όπως οι Amazon Web Services και η Azure Services Platform της Microsoft.

Το Google App Engine [9] είναι δωρεάν μέχρι ένα ορισμένο επίπεδο χρησιμοποιούμενων πόρων. Συνδρομή χρεώνεται για τον πρόσθετο αποθηκευτικό χώρο, το εύρος ζώνης, ή τους κύκλους ΚΜΕ (CPU) που απαιτούνται από την εφαρμογή.

Ο χρήστης μπορεί να θέσει σε διάθεση την εφαρμογή του από το δικό του όνομα διαδικτύου (domain name), όπως π.χ. http://www.example.com/, με το Google Apps. Ή, μπορείτε να τη διαθέσει χρησιμοποιώντας ένα ελεύθερο όνομα στον appspot.com τομέα. Μπορεί να μοιραστεί την εφαρμογή του με τον κόσμο, ή να περιορίσει την πρόσβαση στα μέλη του οργανισμού του.

Το Google App Engine υποστηρίζει εφαρμογές γραμμένες σε διάφορες γλώσσες προγραμματισμού. Με το περιβάλλον χρόνου εκτέλεσης της Java

(Java Runtime Environment), ο χρήστης μπορεί να στήσει την εφαρμογή του (app) χρησιμοποιοντας τυποποιημένες Java τεχνολογίες, συμπεριλαμβανομένης της JVM, τους Java servlets, καθώς και τη γλώσσα προγραμματισμού της Java - ή οποιαδήποτε άλλη γλώσσα, όπως οι JavaScript και η Ruby, χρησιμοποιώντας ένα βασισμένο σε JVM (JVM-based) διερμηνευτή ή μεταγλωττιστή. Το App Engine διαθέτει επίσης ένα ειδικό περιβάλλον χρόνου εκτέλεσης της Python (Python runtime environment), το οποίο περιλαμβάνει έναν γρήγορο διερμηνευτή Python και την πρότυπη βιβλιοθήκη της Python. Τα περιβάλλοντα χρόνου εκτέλεσης της Java και της Python έχουν κατασκευαστεί για να διασφαλιστεί ότι η εφαρμογή του χρηστη τρέχει γρήγορα, με ασφάλεια και χωρίς παρεμβολές από άλλες εφαρμογές στο σύστημα.

3.2 *Το περιβάλλον της εφαρμογής*

Με το Google App Engine γίνεται πολύ εύκολη η δημιουργία μιας εφαρμογής που λειτουργεί με αξιοπιστία, ακόμη και υπό βαρύ φορτίο και με μεγάλες ποσότητες δεδομένων. Το App Engine περιλαμβάνει τα ακόλουθα στοιχεία:

- δυναμική εξυπηρέτηση (dynamic web serving), με πλήρη υποστήριξη για κοινές διαδικτυακές τεχνολογίες (web technologies)
- αποθηκευτικός χωρος με ερωτήσεις ανάκτησης (queries), ταξινόμηση και συναλλαγές (database transactions)
- δυνατότητα αυτόματης κλιμάκωσης και εξισορρόπησης φορτίου
- APIs για τον έλεγχο ταυτότητας των χρηστών και αποστολή email χρησιμοποιώντας το Google Accounts
- ένα πλήρως διατιθέμενο τοπικό περιβάλλον ανάπτυξης που προσομοιώνει το Google App Engine στον υπολογιστή του χρήστη
- προγραμματισμένες εργασίες για γεγονότα που λειτουργούν σε συγκεκριμένες χρονικές στιγμές και τακτά χρονικά διαστήματα

Η εφαρμογή μπορεί λοιπόν να λειτουργεί σε ένα από τα δύο περιβάλλοντα χρόνου εκτέλεσης (runtime environments): το περιβάλλον Java και το

περιβάλλον Python. Κάθε περιβάλλον παρέχει τυποποιημένα πρωτόκολλα και κοινές τεχνολογίες για την ανάπτυξη διαδικτυακών εφαρμογών.

3.2.1 Το Sandbox

Οι εφαρμογές τρέχουν σε ένα ασφαλές περιβάλλον που παρέχει περιορισμένη πρόσβαση στο υποκείμενο λειτουργικό σύστημα. Οι περιορισμοί αυτοί επιτρέπουν στο App Engine να διανέμει διαδικτυακές αιτήσεις (web requests) για την εφαρμογή σε πολλούς διακομιστές, και να ξεκινάει και να σταματάει τους διακομιστές έτσι ώστε να να πληρούν τις απαιτήσεις της κυκλοφορίας. Το SandBox απομονώνει την εφαρμογή του χρήστη στο δικό της ασφαλές, αξιόπιστο περιβάλλον, το οποίο είναι ανεξάρτητο από το υλικό (hardware), το λειτουργικό σύστημα και τη φυσική τοποθεσία του διακομιστή (web server).

Παραδείγματα των περιορισμών του ασφαλούς περιβάλλοντος SandBox περιλαμβάνουν:

- Μία εφαρμογή μπορεί να έχει πρόσβαση σε άλλους υπολογιστές στο Internet μόνο μέσω του URL που παρέχεται και των υπηρεσιών email. Οι άλλοι υπολογιστές μπορούν να συνδεθούν με την εφαρμογή μόνο κάνοντας HTTP (ή HTTPS) αιτήσεις (HTTP requests) στις προκαθορισμένες θύρες (standard ports).
- Μία εφαρμογή δεν μπορεί να γράψει στο σύστημα αρχείων. Μία εφαρμογή μπορεί να διαβάσει αρχεία, αλλά μόνο τα αρχεία που φορτώνονται με τον κώδικα της εφαρμογής. Η εφαρμογή πρέπει να χρησιμοποιεί τον αποθηκευτικό χώρο του App Engine (datastore), την memcache ή άλλες υπηρεσίες για όλα τα δεδομένα που εξακολουθουν να υφίστανται μεταξύ των αιτήσεων.
- Ο κώδικας της εφαρμογής εκτελείται μόνο ως απάντηση σε μία διαδικτυακή αίτηση (web request) ή σε ένα cron job, και πρέπει να επιστρέψει απάντηση δεδομένων εντός 30 δευτερολέπτων σε κάθε περίπτωση. Ένας χειριστής αιτήσεων (request handler) δεν μπορεί να

γεννήσει μια επιμέρους διεργασία (sub-process) ή να εκτελέσει κώδικα αφού έχει αποσταλεί η απάντηση (response).

3.2.2 Το περιβάλλον χρόνου εκτέλεσης Java (Java Runtime Environment)

Ο χρήστης μπορεί να αναπτύξει την εφαρμογή του για το Java Runtime Environment χρησιμοποιώντας κοινά διαδικτυακά εργαλεία ανάπτυξης και πρότυπα διεπαφών προγραμματισμού εφαρμογών (API). Η εφαρμογή αλληλεπιδρά με το περιβάλλον, χρησιμοποιώντας το πρότυπο Java Servlet, και μπορεί να χρησιμοποιήσει τεχνολογίες κοινών διαδικτυακών εφαρμογών, όπως οι JavaServer Pages (JSPs).

Το Java Runtime Environment χρησιμοποιεί Java 6. Το App Engine Java SDK υποστηρίζει τις αναπτυσσόμενες εφαρμογές χρησιμοποιώντας είτε Java 5 ή 6.

Το περιβάλλον περιλαμβάνει την πλατφόρμα Java SE Runtime Environment (JRE) 6 και τις βιβλιοθήκες. Οι περιορισμοί του περιβάλλοντος sandbox υλοποιούνται στο JVM. Μια εφαρμογή μπορεί να χρησιμοποιήσει οποιοδήποτε JVM bytecode ή χαρακτηριστικό βιβλιοθήκης, εφόσον αυτό δεν υπερβαίνει τους περιορισμούς του SandBox. Για παράδειγμα, κώδικας bytecode που προσπαθεί να ανοίξει μια υποδοχή (socket) ή να γράψει σε ένα αρχείο θα προκαλέσει εξαίρεση χρόνου εκτέλεσης (runtime exception).

Η εφαρμογή έχει πρόσβαση στις περισσότερες υπηρεσίες του App Engine με τη χρήση προτύπων APIs της Java. Για το χώρο αποθήκευσης του App Engine (datastore), το Java SDK περιλαμβάνει υλοποιήσεις των Java Data Objects (JDO) και Java Persistence API (JPA) διεπαφών. Η εφαρμογή μπορεί να χρησιμοποιήσει το JavaMail API για να στείλει μηνύματα ηλεκτρονικού ταχυδρομείου με την υπηρεσία App Engine Mail. Τα java.net HTTP APIs έχουν πρόσβαση στην υπηρεσία ανακτησης URL (URL fetch) του App Engine. Το App Engine περιλαμβάνει επίσης APIs χαμηλού επιπέδου έτσι ώστε οι υπηρεσίες του να εφαρμόζουν πρόσθετους προσαρμογείς, ή να χρησιμοποιούν άμεσα από την εφαρμογή.

Συνήθως, οι προγραμματιστές της γλώσσας Java χρησιμοποιούν τη γλώσσα προγραμματισμού Java και τα APIs για την υλοποίηση διαδικτυακών εφαρμογών για το JVM. Με τη χρήση συμβατών με το JVM συμβολομεταφραστών ή διερμηνευτών, μπορούν επίσης να χρησιμοποιηθούν άλλες γλώσσες για την ανάπτυξη διαδικτυακών εφαρμογών, όπως η JavaScript, η Ruby, ή η Scala.

3.2.3 Το περιβάλλον χρόνου εκτέλεσης Python (Python Runtime Environment)

Με το περιβάλλον χρόνου εκτέλεσης Python του App Engine, ο χρήστης μπορεί να υλοποιησει την εφαρμογή του χρησιμοποιώντας τη γλώσσα προγραμματισμού Python και να την τρέχει σε έναν βελτιστοποιημένο διερμηνέα της Python. Το App Engine περιλαμβάνει πλούσια APIs και εργαλεία για την ανάπτυξη διαδικτυακών εφαρμογών Python, συμπεριλαμβανομένου ενός API χαρακτηριστικά πλούσιου σε μοντελοποίηση δεδομένων, ένός εύκολου στη χρήση πλαισίου (framework) διαδικτυακών εφαρμογών, καθώς και εργαλεία για τη διαχείριση και την πρόσβαση στα δεδομένα της εφαρμογής. Ο χρήστης μπορεί επίσης να επωφεληθεί από μια μεγάλη ποικιλία από βιβλιοθήκες και πλαίσια για την ανάπτυξη διαδικτυακών εφαρμογών σε Python, όπως το Django.

Το περιβάλλον χρόνου εκτέλεσης Python χρησιμοποιεί την έκδοση 2.5.2 της Python. Πρόσθετη υποστήριξη για την Python 3 εξετάζεται για μια μελλοντική έκδοση.

Το περιβάλλον χρόνου εκτέλεσης Python περιλαμβάνει την τυποποιημένη βιβλιοθήκη της Python (Python standard library). Φυσικά, δεν είναι δυνατόν όλα τα χαρακτηριστικά της βιβλιοθήκης να μπορούν να τρέξουν στο περιβάλλον του sandbox. Για παράδειγμα, η κλήση μιας μεθόδου που επιχειρεί να ανοίξει μια υποδοχή (socket) ή να γράψει σε ένα αρχείο θα προκαλέσει μια εξαίρεση. Για λόγους ευκολίας, πολλές ενότητες στην πρότυπη βιβλιοθήκη, των οποίων τα βασικά χαρακτηριστικά δεν υποστηρίζονται από το περιβάλλον χρόνου εκτέλεσης, έχουν απενεργοποιηθεί, και κώδικας που τα εισάγει θα προκαλέσει σφάλμα.

Κώδικας εφαρμογής γραμμένος για το περιβάλλον της Python πρέπει να είναι γραμμένος, αποκλειστικά σε Python. Επεκτάσεις γραμμένες στην γλώσσα C δεν υποστηρίζονται.
Το περιβάλλον παρέχει πλούσια Python APIs για το χώρο αποθήκευσης (datastore), για λογαριασμούς Google (Google Accounts), για την ανάκτηση URL (URL fetch) και για υπηρεσίες email. Το App Engine παρέχει επίσης ένα απλό πλαίσιο διαδικτυακών εφαρμογών (web application framework) σε Python που ονομάζεται webapp για να καταστήσει εύκολο το ξεκίνημα δημιουργίας εφαρμογών.
Ο χρήστης μπορεί να μεταφορτώσει και βιβλιοθήκες τρίτων με την εφαρμογή του, εφόσον είναι υλοποιημένες σε καθαρή Python και δεν απαιτούν καποια μη-υποστηριζόμενη ενότητα πρότυπης βιβλιοθήκης.

3.2.4 Το Datastore

Το App Engine παρέχει μία ισχυρά κατανεμημένη υπηρεσία αποθήκευσης δεδομένων (data storage service) που διαθέτει επίσης μηχανή αναζήτησης και συναλλαγών. Ακριβώς όπως ο κατανεμημένος διακομιστής (web server) μεγαλώνει με την κυκλοφορία, έτσι και το διανεμόμενο datastore μεγαλώνει με τα δεδομένα.

Το datastore του App Engine δεν είναι σαν μία παραδοσιακή σχεσιακή βάση δεδομένων. Τα αντικείμενα δεδομένων, ή αλλιώς "οντότητες" (entities), έχουν ένα είδος (kind) και ένα σύνολο ιδιοτήτων (properties). Τα επερωτήματα (queries) μπορούν να ανακτήσουν οντότητες ενός συγκεκριμένου είδους φιλτράρισμένου και ταξινομημένου με βάση τις τιμές των ιδιοτήτων. Οι τιμές των ιδιοτήτων μπορούν να είναι οποιουδήποτε από τους υποστηριζόμενους τύπους τιμών των ιδιοτήτων.
Οι οντότητες του datastore είναι "schemaless." Η δομή των οντοτήτων δεδομένων παρέχεται και ενισχύεται από τον κώδικα της εφαρμογής. Οι Java JDO/JPA διεπαφές και η διεπαφή του datastore της Python περιλαμβάνουν χαρακτηριστικά για την επιβολή και την ενίσχυση της δομής μέσα στην εφαρμογή. Η εφαρμογή μπορεί επίσης να έχει πρόσβαση στο datastore άμεσα έτσι ώστε να εφαρμόσει λιγότερη ή περισσότερη δομή όπου χρειάζεται.

Το datastore είναι συνεπές και χρησιμοποιεί τη μέθοδο του βέλτιστου ελέγχου συγχρονισμού (optimistic concurrency control - OCC). Μια ενημερωμένη έκδοση μιας οντότητας παρουσιάζεται σε μια συναλλαγή που επαναπροσπαθείται σταθερό αριθμό φορών αν άλλες διεργασίες που προσπαθούν να ενημερώσουν την ίδια οντότητα ταυτόχρονα. Η εφαρμογή μπορεί να εκτελέσει πολλαπλές λειτουργίες στο datastore σε μία μοναδική συναλλαγή που είτε όλες επιτύγχάνουν ή όλες αποτυγχάνουν, διασφαλίζοντας την ακεραιότητα των δεδομένων.
Το datastore εκτελεί συναλλαγές κατα μήκος του κατανεμημένου δικτύου του χρησιμοποιώντας «ομάδες οντοτήτων». Μια συναλλαγή χειρίζεται οντότητες που ανήκουν στην ίδια ομάδα. Οντότητες της ίδιας ομάδας αποθηκεύονται μαζί για την αποτελεσματική εκτέλεση των συναλλαγών. Η εφαρμογή μπορεί να εκχωρήσει οντότητες σε ομάδες κατά τη δημιουργία των οντοτήτων.

3.2.5 Υπηρεσίες του App Engine

Το App Engine παρέχει ένα φάσμα υπηρεσιών που επιτρέπουν στο χρήστη να εκτελεί συνηθισμένες λειτουργίες κατά τη διαχείριση της εφαρμογής του. Οι ακόλουθες διεπαφες προγραμματισμού εφαρμογών παρέχονται για την πρόσβαση στις υπηρεσίες αυτές:

- URL fetch

Οι εφαρμογές μπορούν να έχουν πρόσβαση σε πόρους του Διαδικτύου, όπως σε διαδικτυακές υπηρεσίες ή άλλα δεδομένα, χρησιμοποιώντας την υπηρεσία URL fetch του App Engine. Η υπηρεσία URL fetch ανακτά διαδικτυακούς πόρους χρησιμοποιόντας την ίδια Google υποδομή υψηλής ταχύτητας που ανακτά τις σελίδες για πολλά άλλα προϊόντα της Google.

- Mail

Οι εφαρμογές μπορούν να στείλουν μηνύματα ηλεκτρονικού ταχυδρομείου με τη χρήση της υπηρεσίας ηλεκτρονικού ταχυδρομείου του App Engine. Η υπηρεσία χρησιμοποιεί την υποδομή του Google για να στείλει μηνύματα ηλεκτρονικού ταχυδρομείου.

- Memcache

Η υπηρεσία Memcache παρέχει στην εφαρμογή υψηλές επιδόσεις σε κρυφή μνήμη (cache), η οποία είναι προσβάσιμη από πολλά στιγμιότυπα της εφαρμογής. Η Memcache είναι χρήσιμη για προσωρινά δεδομένα ή δεδομένα τα οποία τα αντιγράφονται από το datastore στη μνήμη cache για υψηλής ταχύτητας πρόσβαση.

- Image Manipulation

Η υπηρεσία επιτρέπει στην εφαρμογή τη διαχείριση εικόνων. Με αυτό το API, υπάρχει η δυνατότητα αλλαγής του μεγέθους, αποκοπής και περιστροφής σε εικόνες σε μορφή JPEG και PNG formats.

- Scheduled Tasks

Η υπηρεσία Cron επιτρέπει στο χρήστη τον προγραμματισμό εργασιών να εκτελούνται σε τακτά χρονικά διαστήματα.

3.3 *Γιατί Google App Engine;*

Το Google App Engine επιτρέπει το χτίσιμο διαδικτυακών εφαρμογών πάνω στα ίδια κλιμακωτά συστήματα που τροφοδοτούν τις εφαρμογές του Google. Οι εφαρμογές του App Engine είναι εύκολες στην κατασκευή, τη διατήρηση και την κλιμάκωση της ανάπτυξής τους, όσο η κυκλοφορία και η ανάγκη αποθηκευτικού χώρου για τα δεδομένα (data storage) αυξάνεται. Με το App Engine, δεν υπάρχουν κεντρικοί εξυπηρέτες προς συντήρηση: Απλά φορτώνεται η εφαρμογή και είναι έτοιμη να εξυπηρετήσει τους χρήστες.

- Εύκολο Ξεκίνημα

Το App Engine είναι μία πλήρης στοίβα επιπέδων ανάπτυξης, η οποία χρησιμοποιεί γνωστές τεχνολογίες για την κατασκευή και φιλοξενία διαδικτυακών εφαρμογών. Με το App Engine παρέχεται η δυνατότητα στο χρήστη, αφού τελειώσει τη συγγραφή του κώδικα της εφαρμογής, να τον ελέγξει στο τοπικό του μηχάνημα και να τον φορτώσει στο Google με μία πολύ

απλή διαδικασία. Μόλις φορτωθεί η αίτηση, η εφαρμογή πλέον φιλοξενείται και ενημερώνεται εκεί. Το App Engine φροντίζει για όλες τις διαδικασίες συντήρησης, όπως τη διαχείριση του συστήματος, τη φόρτωση νέων στιγμιοτύπων της εφαρμογής και τη διαχείριση της βάσης δεδομένων, έτσι ώστε ο ιδιοκτήτης να μπορεί να επικεντρωθεί στα χαρακτηριστικά που προσφέρει η εφαρμογή του στους χρήστες.

- Αυτόματη επεκτασιμότητα

Οι εφαρμογές μπορούν να εκμεταλλευτούν τις ίδιες εξελικτικές τεχνολογίες στις οποίες στηρίζονται οι εφαρμογές του Google, όπως οι BigTable και GFS. Η αυτόματη επεκτασιμότητα είναι μία από τις ιδιότητες του App Engine, έτσι το μόνο που έχει να κάνει ο χρήστης ειναι να γράψει τον κώδικα της εφαρμογής. Ανεξάρτητα από το πόσους χρήστες έχει η εφαρμογή ή πόσα δεδομένα αποθηκεύει, με τη βοήθεια του App Engine μπορεί να προσαρμοστεί έτσι ώστε να ικανοποιήσει τις ανάγκες του χρήστη.

- Η αξιοπιστία, η απόδοση και η ασφάλεια της υποδομής του Google

Το Google έχει μια φήμη για ιδιαίτερα αξιόπιστη, υψηλής επίδοσης υποδομή. Με το App Engine ο χρήστης μπορεί να εκμεταλλευτεί τα 10 έτη γνώσης του Google στο να τρέχει μαζικά εξελικτικά συστηματα, οδηγούμενα με στόχο την υψηλή απόδοση. Οι ίδιες πολιτικές ασφάλειας, μυστικότητας και προστασίας δεδομένων που υπάρχουν για τις εφαρμογές του Google ισχύουν και για όλες τις εφαρμογές του App Engine. Το Google αντιμετωπίζει την ασφάλεια πολύ σοβαρά και λαμβάνει τα κατάλληλα μέτρα έτσι ώστε να προστατεύσει τον κώδικα και τα δεδομένα της εφαρμογής του χρήστη.

- Αποδοτικό κόστος φιλοξενίας

Το App Engine θα είναι πάντα δωρεάν για να ξεκινήσει κάποιος, και επιπλέον η Google σχεδιάζει να δώσει τη δυνατότητα στους χρήστες να μπορούν να αγοράσουν περισσότερους υπολογιστικούς πόρους στο εγγύς μέλλον.

- Δοκιμαστική περίοδος ελεύθερη κινδύνου

Όχι μόνο η δημιουργία μιας εφαρμογής του App Engine είναι πολύ εύκολη, αλλά είναι και δωρεάν. Ο χρήστης μπορεί να δημιουργήσει έναν λογαριασμό (account) και να δημοσιεύσει μια εφαρμογή, η οποία μπορεί να χρησιμοποιηθεί αμέσως, χωρίς καμία δαπάνη, και χωρίς καμία δέσμευση. Μια αποδοτική εφαρμογή σε έναν δωρεάν λογαριασμό μπορεί να χρησιμοποιήσει μέχρι 500MB αποθήκευσης και μέχρι 5 εκατομμύρια αναγνώσεις της ιστοσελίδας σε έναν μήνα. Όταν ο χρήστης είναι έτοιμος για περισσότερα, μπορεί να ενεργοποιήσει την τιμολόγηση, να θέσει έναν μέγιστο ημερήσιο προϋπολογισμό, και να διαθέσει τον προϋπολογισμό του για κάθε πόρο σύμφωνα με τις ανάγκες του.

ΚΕΦΑΛΑΙΟ 4
ΔΗΜΟΣΙΕΥΣΗ ΠΕΡΙΕΧΟΜΕΝΟΥ

4.1 *Γενικά*

Το WAP Push [10] εξακολουθεί να είναι το πιο κοινό μέσο για την παράδοση πλούσιων υπηρεσιών πολυμέσων από επιχειρήσεις σε ομότιμους χρήστες (peers). Ένα WAP Push είναι ένα απλό SMS εντός της κεφαλίδας του οποίου περιλαμβάνεται ένας υπερσύνδεσμος σε μία διεύθυνση WAP. Με τη λήψη ενός WAP Push, η συμβατή συσκευή αυτόματα θα δώσει στο χρήστη την επιλογή να αποκτήσει πρόσβαση στο περιεχόμενο WAP. Με τον τρόπο αυτό το WAP Push κατευθύνει τον τελικό χρήστη σε μια διεύθυνση WAP όπου συγκεκριμένο περιεχόμενό μπορεί να έχει αποθηκευτεί και να είναι έτοιμο για προβολή ή λήψη στη συσκευή. Η διεύθυνση θα μπορούσε να είναι μια απλή σελίδα ή ένα ολόκληρο WAP site.

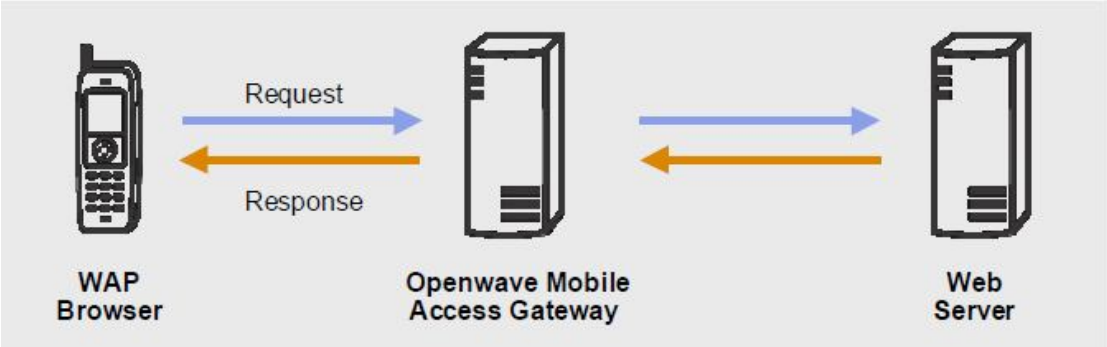
Η οντότητα δικτύου [11] που επεξεργάζεται τα WAP Push μηνύματα και τα παραδίδει πάνω σε έναν φορέα (bearer) IP ή SMS είναι γνωστή ως πύλη δικτύου Push Proxy (Push Proxy Gateway). Η πύλη δικτύου Push Proxy είναι ένα στοιχείο των πυλών δικτύου WAP που ωθεί URL ειδοποιήσεις προς κινητά τηλέφωνα.
Ένα σημαντικό όφελος του WAP Push είναι η δυνατότητα αποστολής των δεδομένων σε ένα συνδρομητή κινητής τηλεφωνίας χωρίς ρητό ad hoc αίτημα, από τον τελικό χρήστη (σε αντίθεση με μια διαδικτυακή διεύθυνση URL, όπου ο χρήστης πρέπει να "τραβήξει" περιεχόμενο).

4.2 *Τι είναι το WAP Push*

Το WAP push είναι [2] ένα ανοιχτό πρότυπο που αναπτύχθηκε από το WAP Φόρουμ (WAP Forum) για την παροχή δυνατοτήτων ώθησης, οι οποίες επιτρέπουν τη δημιουργία νέων εφαρμογων, καθώς και δραματικές βελτιώσεις στις ήδη υπάρχουσες υπηρεσίες ώθησης.

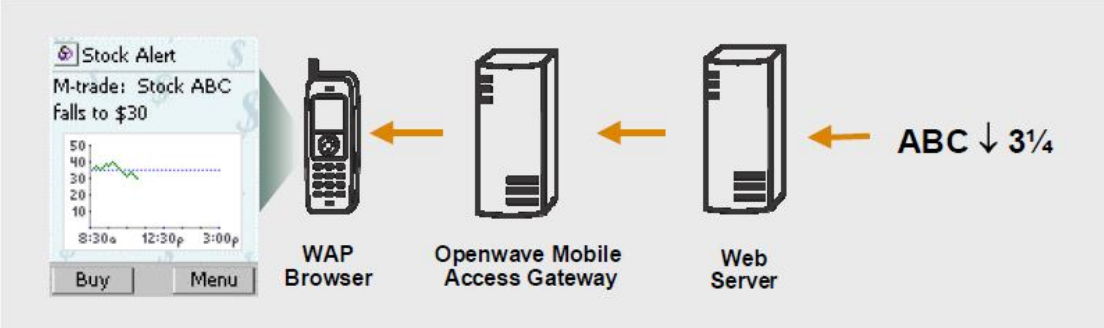
Το WAP push παρέχει ένα πρότυπο μέσο για την αποστολή δεδομένων σε ένα κινητό συνδρομητή χωρίς ρητό αίτημα του συνδρομητή τη στιγμή που

παραδίδονται τα δεδομένα. Στο κανονικό μοντέλο πελάτη/εξυπηρετητή (client/server model), ο πελάτης κάνει αίτηση για μια υπηρεσία ή για πληροφορίες από έναν διακομιστή, ο οποίος στη συνέχεια ανταποκρίνεται με τη μετάδοση πληροφοριών στον πελάτη. Αυτό είναι γνωστό ως τεχνολογία "pull". Ο πελάτης τραβάει (pulls) πληροφορίες από το διακομιστή. Ο Παγκόσμιος Ιστός (World Wide Web), είναι το καλύτερο παράδειγμα τεχνολογίας "pull", όπου ένας χρήστης εισάγει μια διεύθυνση URL (την αίτηση), η οποία αποστέλλεται σε ένα διακομιστή και ο διακομιστής απαντά με την αποστολή μιας ιστοσελίδας (η απάντηση) προς το χρήστη (Εικόνα 2) .



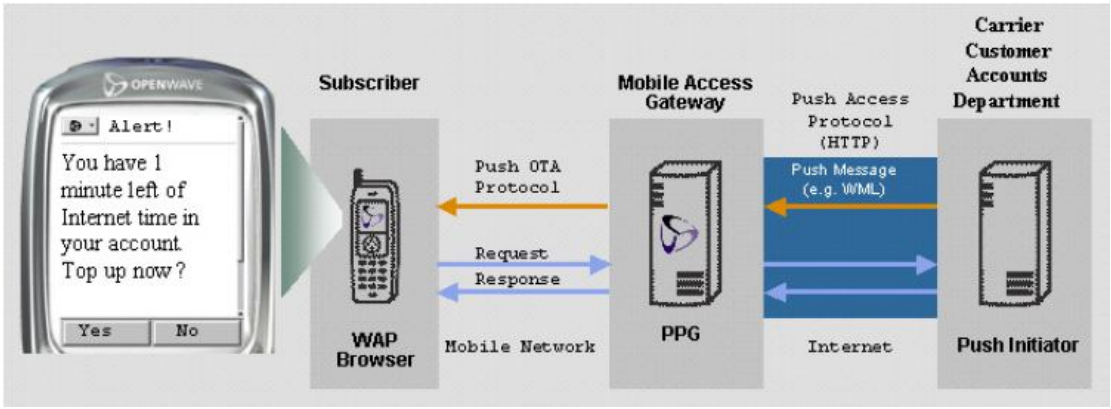
Εικόνα 2: Παραδοσιακό "Pull" μοντέλο αίτησης-απάντησης

Αν και το WAP push βασίζεται επίσης στο μοντέλο πελάτη/εξυπηρετητή, παρέχει ένα μέσο για την αποστολή δεδομένων προς το συνδρομητή, χωρίς ρητό αίτημα του πελάτη, τη στιγμή της παράδοσης (Εικόνα 3). Σημείωστε στην Εικόνα 3 ότι ο χρήστης λαμβάνει ένα διάγραμμα μετοχών (stock chart) και μπορεί αμέσως να αγοράσει τη μετοχή (το σύμβολο ABC) με ένα μόνο κλικ. Αυτό δεν είναι δυνατόν με τις εναλλακτικές λύσεις για το WAP push, όπως το πρότυπο SMS.



Εικόνα 3: WAP Push ειδοποίηση

Το σύστημα WAP Push μεταφέρει δεδομένα σε μία συμβατή με WAP συσκευή χωρίς την παρέμβαση του χρήστη. Στις κανονικές επικοινωνίες μέσω Διαδικτύου, ένας πελάτης ζητάει στοιχεία ή υπηρεσίες από μια διαδικτυακή εφαρμογή και η απόκριση παραδίδεται στον αιτούν πελάτη μέσω μιας συμβατικής WAP Pull πύλης. Χρησιμοποιώντας τεχνολογία push, οι διαδικτυακές εφαρμογές μπορούν να μεταφέρουν δεδομένα σε μια συσκευή χωρίς ρητή αίτηση του χρήστη κατά τη στιγμή της παράδοσης. Μία λειτουργία WAP push προκύπτει όταν η οντότητα που θέλει να στείλει την πληροφορία στον τομέα του Διαδικτύου (Push Initiator), μεταφέρει περιεχόμενο σε έναν πελάτη στον τομέα του WAP μέσω μιας πύλης δικτύου Push Proxy (Push Proxy Gateway).



Εικόνα 4: Παράδειγμα WAP Push συστήματος

Στην Εικόνα 4, στο λογαριασμό πρόσβασης στο Διαδίκτυο του κινητού συνδρομητή απομένουν μόνο λίγα λεπτά. Το σύστημα λογαριασμών των πελατών του φορέα της κινητής στέλνει ένα push μήνυμα στο συνδρομητή για να τον ενημερώνει για το χαμηλό υπόλοιπο του λογαριασμού. Με ένα κλικ ο συνδρομητής μπορεί να τροφοδοτήσει το λογαριασμό με τη χρήση των πληροφοριών της πιστωτικής του κάρτας, οι οποίες είναι αποθηκευμένες στο σύστημα λογαριασμώντου φορέα κινητής. Τα βέλη στο επάνω μέρος της εικόνας απεικονίζουν το WAP Push μήνυμα. Τα βέλη αίτησης/απάντησης αντιπροσωπεύουν την επόμενη σύνοδο WAP που εγκαθιδρύεται όταν ο παραλήπτης του μηνύματος αποφασίζει να ανανεώσει το υπόλοιπό του

λογαριασμού του μετά την παραλαβή του WAP Push μηνύματος που έγινε με ένα κλικ.

4.3 *Τα βασικά στοιχεία του συστήματος*

Τα βασικά στοιχεία του συστήματος WAP Push περιγράφονται παρακάτω:

4.3.1 Push Initiator

Ο Push Initiator είναι η οντότητα (π.χ. μία εφαρμογή, όπως μια ιστοσελίδα ειδήσεων) που δημιουργεί push περιεχόμενο και το υποβαλλει με τη μορφή μιας push αίτησης (με τη χρήση του πρωτοκόλλου Push Access - PAP) στην πύλη δικτύου Push Proxy για παράδοση στην κινητή συσκευή. Κάθε υποβολή έχει ένα μοναδικό αναγνωριστικό. Ο Push Initiator μπορεί να ζητήσει το αποτέλεσμα της υποβολής (αν παραδόθηκε ή οχι η αίτηση), να ελέγξει τις δυνατότητες μιας συγκεκριμένης συσκευής πελάτη και να ελέγξει την κατάσταση μιας προηγούμενης υποβολής ή την ακύρωση μιας υποβολής.

4.3.2 Push Proxy Gateway (PPG)

Η πύλη δικτύου Push Proxy είναι το σημείο πρόσβασης για τα μηνύματα push από το Διαδίκτυο στους πελάτες. Η πύλη δικτύου Push Proxy, μπορεί να εφαρμόσει πολιτικές που περιλαμβάνουν: έλεγχο πρόσβασης του Push Initiator, εντοπισμό σφαλμάτων στο push περιεχόμενο, επίλυση της διεύθυνσης πελάτη, μετάφραση του περιεχομένου και μετατροπή πρωτοκόλλων (HTTP σε WSP, HTTP σε UP.Notify). Από την μετάφρασης, το push μήνυμα παραδίδεται στη συσκευή χρησιμοποιώντας το Push Over-The-Air (OTA) πρωτόκολλο.

4.3.3 WAP Push Client

Ένας WAP Push Client περιλαμβάνεται σε μια συσκευή (π.χ. κινητό τηλέφωνο) που μπορεί να δέχεται WAP push περιεχόμενο. Φιλοξενεί εφαρμογές, όπως ένα πρόγραμμα περιήγησης WAP, που μπορεί να λαμβάνει και να επεξεργάζεται το push περιεχόμενο. Επίσης φιλοξενεί το Session

Initiation πρωτόκολλο (SIP), που μπορεί να χρησιμοποιηθεί από την πύλη δικτύου για να ζητήσει από τον πελάτη να ξεκινήσει μια WAP σύνοδο. Μια λειτουργία push WAP προκύπτει όταν ένας Push Initiator στον τομέα του Διαδικτύου επικοινωνεί με μία πύλη δικτύου Push Proxy και παραδίδει το περιεχόμενο για έναν πελάτη/προορισμό χρησιμοποιώντας το πρωτόκολλο Push Access.

Η πύλη δικτύου Push Proxy τότε, ωθεί τα δεδομένα σε έναν πελάτη, μέσω του πρωτοκόλλου Push OTA. Και εκτός από την απλή παροχή υπηρεσιών πύλης δικτύου μεσολάβησης, η πύλη δικτύου Push Proxy παρέχει στον Push Initiator την δυνατότητα υπηρεσιών αναζήτησης των δυνατοτήτων των συσκευών, επιτρέποντας έτσι στον Push Initiator να επιλέξει το κατάλληλο περιεχόμενο ή να το μορφοποιήσει για την αποστολή σε μία συγκεκριμένη συσκευή. Η πύλη δικτύου Push Proxy, μπορεί επίσης να ειδοποιήσει τον Push Initiator σχετικά με την κατάσταση της λειτουργίας push.

4.4 *Τα βασικα οφέλη της τεχνολογίας WAP Push*

Το WAP Push προσφέρει άμεσα και απτά οφέλη για τους συνδρομητές, τους προγραμματιστές και τους φορείς εκμετάλλευσης κινητών δικτύων.

4.4.1 Οφέλη Συνδρομητών

Παρέχει μία απλή στη χρήση (ένα ή δύο κλικ) διαδικασία για τον τελικό χρήστη έτσι ώστε να αποκτησει πρόσβαση σε πλούσιο περιεχόμενο πολυμέσων (rich multimedia **content)**

- Περισσότερο περιεχόμενο: περισσότερες πληροφορίες, με ήχο και γραφικά
- Βελτίωση της χρηστικότητας: απρόσκοπτη ολοκλήρωση της εφαρμογης, του παραδοτέου περιεχομένου και του έργου του χρήστη με πολύ απλό τρόπο (π.χ. η λήψη της αναφοράς και η αγορά μετοχών με ένα κλικ).

- Νέες εφαρμογές: ενισχυμένες αλληλεπιδράσεις συνδρομητή με συνδρομητή, όπως μία δημοπρασία, ανταλλαγή άμεσων μηνυμάτων και παιχνίδια και εφαρμογές με πολλούς χρήστες.
- Ασφάλεια: το WTLS διασφαλίζει την προστασία των μεταδόσεων δεδομένων
- Το κόστος του WAP Push δεν υπερβαίνει την τιμή ενός SMS και είναι φθηνότερο από την αποστολή ενός μηνύματος MMS

4.4.2 Οφέλη προγραμματιστών

- Επιστροφή των συνδρομητών στη χρήση εφαρμογών ξανά και ξανά μόνο με ένα κλικ
- Δυνατότητα παράδοσης κειμένου, WML, γραφικών και ήχου
- Παράδοση περισσότερου περιεχομένου, νέες εφαρμογές, βελτιωμένη χρηστικότητα: μηδέν ή ένα κλικ για αλληλεπίδραση με την εφαρμογή
- Πρόσβαση σε εκατομμύρια από τις υπάρχουσες συσκευές χειρός
- Νέες εφαρμογές με νέες δυνατότητες:
    - Παράδοση του κατάλληλου περιεχομένου με την αναζήτηση και εύρεση των δυνατοτήτων της συσκευής
    - Παράδοση περιεχομένου συμφωνα με το πόσο επείγον είναι, με προτεραιότητα περιεχομένου
    - Επιβεβαίωση παράδοσης των μηνυμάτων
    - Προσκόμιση της στηριζόμενης στον εξυπηρετητή εφαρμογής απευθείας στο χρήστη με την αίτηση έναρξης συνόδου δεδομένων
    - Παράδοση σε ομάδες χρηστών με διευθυνσιοδότηση για πολλούς χρήστες

- Διασφάλιση των πιο νέων δεδομένων στη συσκευή με λειτουργίες κρυφής μνήμης
- Παράδοση δεδομένων σε συγκεκριμένες εφαρμογές στην πλευρά του πελάτη (client-side), όπως είναι το πρόγραμμα περιήγησης, MMS, WTA κ.α.
- Διασφάλιση ενημερωμένων πληροφοριών με ακύρωση/ αντικατάσταση, διερεύνηση κατάστασης και προσδιορισμός της ποιότητας των υπηρεσιών για τις πολλές μορφές των φορέων χρησιμοποιώντας ένα ανοιχτό πρότυπο πρωτόκολλο (open standard protocol)
- Υποστήριξη συσκευών που έχουν την δυνατότητα αποστολής/παραλαβής μόνο απλών SMS μηνυμάτων, με αυτόματη μετατροπή απλού κειμένου με βάση τις δυνατότητες συσκευής

4.4.3 Οφέλη των φορέων

- Ευρεία διείσδυση στην αγορά των συμβατών με WAP κινητών συσκευών
- Νέες ευκαιρίες για έσοδα: παροχή υπηρεσιών οι οποίες δεν ήταν δυνατές πιο πριν με ενισχυμένες δυνατότητες της εφαρμογής
- Αύξηση της χρήσης κινητού Internet: προσκόμιση των σχετικών υπηρεσιών και πληροφοριών στους συνδρομητές
- Αύξηση της χρήσης των υφιστάμενων υπηρεσιών: WAP, φωνή, μηνύματα, άλλες υπηρεσίες
- Ανοικτό πρότυπο: επιτρέπει στους παρόχους υπηρεσιών να δημιουργήσουν μια κοινότητα κλιμακούμενης ανάπτυξης με πλούσια SDKs
- Υποστηρίζει ένα ευρύτερο σύνολο συσκευών χειρός

- Ιδανικό για την εγγραφή σε υπηρεσίες. Ένα νέο WAP Push μπορεί να αποστέλλεται κάθε φορά που είναι απαραίτητο και δεν χρειάζεται να "τραβιέται" (pulled) από τον τελικό χρήστη

- Έλεγχος πρόσβασης: πρόληψη ανεπίκλητου περιεχομένου με έλεγχο της πρόσβασης στους συνδρομητές (λίστα ελέγχου πρόσβασης, τύπος περιεχομένου, κατά προτεραιότητα, μέγεθος)

- Μπορεί να συνδυαστεί με τη χρήση του SMS Billing για να επιτρέψει τη χρέωση για το περιεχόμενο ή την πρόσβαση σε WAP sites

- Χρέωση: είσοδος στην υπηρεσία συνδυασμένη με ένα γεγονός χρέωσης επιτρέπει ευέλικτα σχήματα χρεώσεων για τα wap push μηνύματα

4.5 *Παραδείγματα εφαρμογών WAP Push*

- Παροχή και διανομή περιεχομένου, όπως πολυφωνικά ringtones, screensavers, εικονομηνύματα, java παιχνίδια και εφαρμογές

- Ειδοποιήσεις με πληροφορίες και ενημερώσεις για πρόγνωση του καιρού και αλλαγές στις τιμές μετοχών

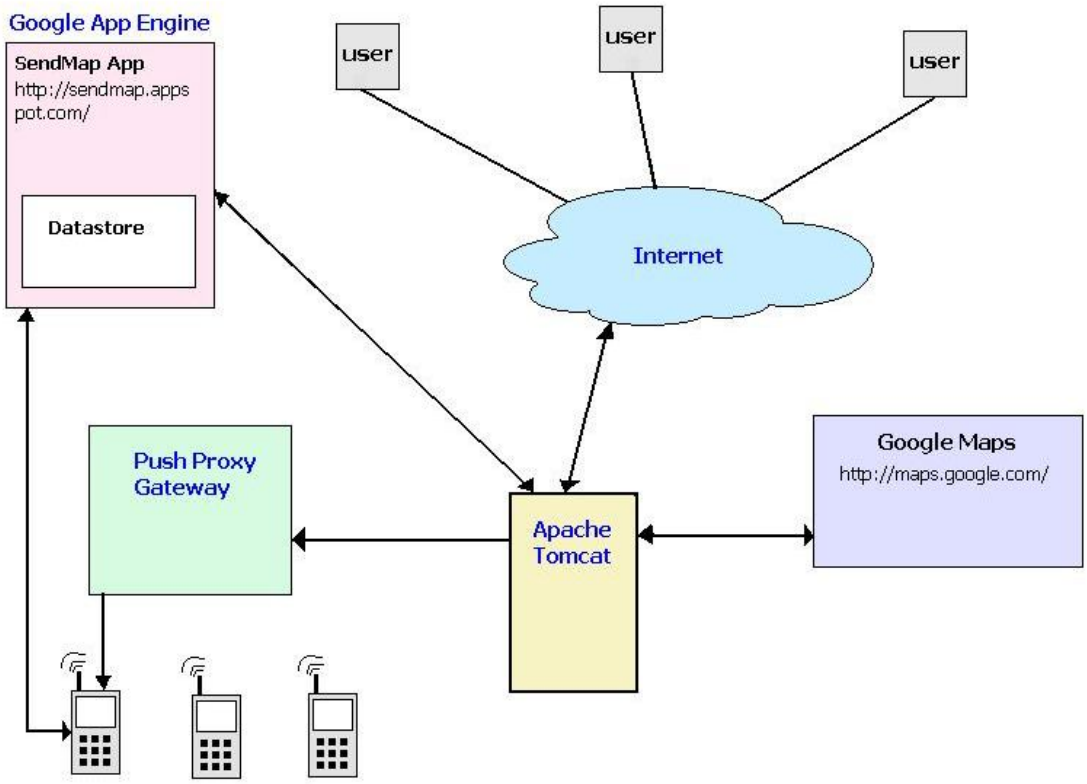
- Ασύρματο Email - ειδοποίηση για εισερχόμενα e-mail, και άμεση πρόσβαση στο γραμματοκιβώτιό του χρήστη

- Interactive Marketing – άμεση πρόσβαση των τελικών χρηστών να σε ειδικά σχεδιασμένα WAP sites που αποτελούν τμήμα συγκεκριμένων διαφημιστικών εκστρατειών

ΚΕΦΑΛΑΙΟ 5
ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΦΑΡΜΟΓΗΣ

5.1 Περιγραφή Συστήματος

Το παρακάτω σχήμα (Εικόνα 5) περιγράφει τη λειτουργία του συστήματος, παρουσιάζοντας τα βασικά δομικά στοιχεία της εφαρμογής καθώς και τις αλληλεπιδράσεις μεταξύ τους. Ακολούθως γίνεται μια σύντομη περιγραφή των βημάτων που ακολουθούνται από τη στιγμή που ένας αποστολέας-χρήστης αρχίζει να χρησιμοποιεί την εφαρμογή μέχρι την εμφάνιση του αποτελέσματος στον παραλήπτη-χρήστη.



Εικόνα 5 : Αρχιτεκτονική Συστήματος

- Όπως φαίνεται από το παραπάνω σχήμα (Εικόνα 5), κάθε χρήστης (user) του διαδικτύου (Internet) μπορεί να προσπελάσει την ιστοσελίδα

"Send Map Image via SMS", η οποία είναι ορατή μέσω του εξυπηρετητή Apache Tomcat.

- Αφού συμπληρώσει τα στοιχεία που είναι απαραίτητα για την αποστολή του μηνύματος, δηλαδή τον αριθμό κινητού του παραλήπτη, προαιρετικά ένα γραπτό μήνυμα και επιλέξει την επιθυμητή περιοχή του χάρτη, πατώντας αποστολή (Send), στέλνει όλες τις παραμέτρους στον εξυπηρετητή μέσω μιας Java Server σελίδας (JSP).

- Ακολούθως, ο Apache Tomcat στέλνει μια HTTP αίτηση GET (HTTP GET request) στον εξυπηρετητή του Google Maps και μέσω του Google Static Maps δημιουργείται η εικόνα του χάρτη, η οποία και στέλνεται ως απάντηση (HTTP response).

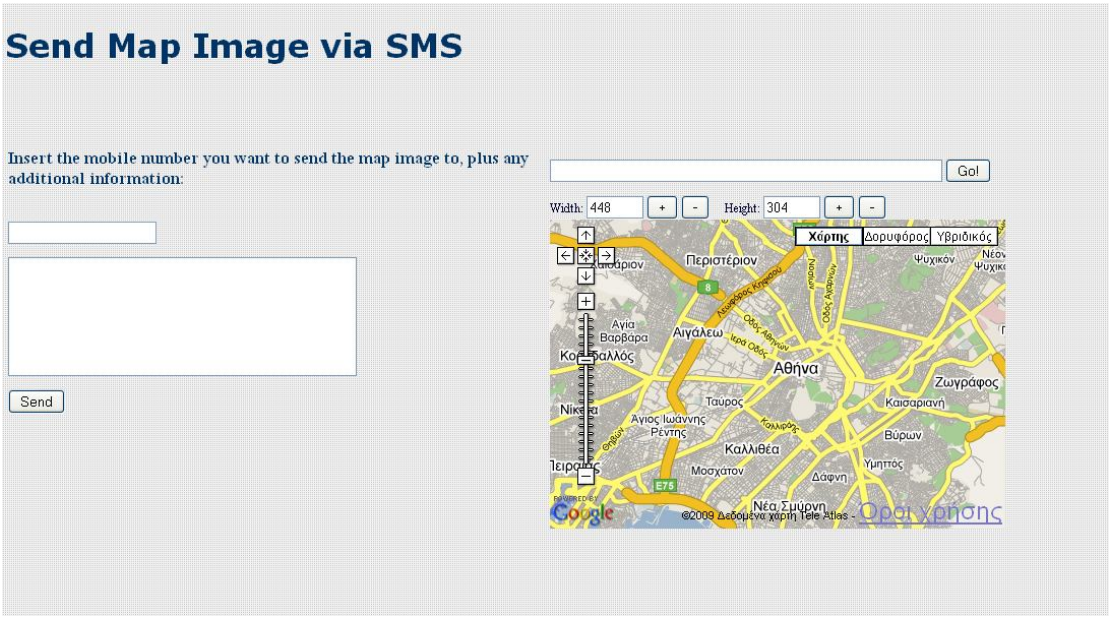
- Στη συνέχεια ο Apache Tomcat μέσω μιας HTTP αίτησης POST (HTTP POST request) αποστέλει στον εξυπηρετητή Google App Engine την εικόνα χάρτη μαζί με το γραπτό κείμενο, τα οποία και αποθηκεύονται στο χώρο αποθήκευσης του App Engine (Datastore). Τα δεδομένα είναι διαθέσιμα μέσω μιας σελίδας της εφαρμογής, το μοναδικό key της οποίας γνωστοποιείται στον Tomcat. Έτσι, δημιουργείται η URL διεύθυνση της σελίδας συνθέτοντας τη διεύθυνση της εφαρμογής (http://sendmap.appspot.com/) με το key. Ο λόγος που χρησιμοποιήθηκε το Google App Engine είναι ότι η σελίδα που περιέχει το αποτέλεσμα της εφαρμογής πρέπει να είναι ορατή δημόσια έτσι ώστε να μπορεί να προσπελαύνεται ανά πάσα στιγμή από το χρήστη, σε αντίθεση με τον Apache Tomcat, ο οποίος εκτελείται σε εσωτερικό δίκτυο.

- Έπειτα μέσω της Java Server σελίδας δημιουργείται το wap push μήνυμα, το οποίο περιέχει έναν υπερσύνδεσμο (link) προς τη σελίδα της εφαρμογής στο Google App Engine που περιέχει τα δεδομένα. Το wap push μήνυμα φτάνει στο κινητό του χρήστη μέσω μιας πύλης δικτύου (Push Proxy Gateway), η οποία ωθεί URL ειδοποιήσεις προς κινητά τηλέφωνα.

- Τέλος, ο χρήστης συνδεόμενος μέσω WAP από τη συσκευή μπορεί να κατεβάσει και να δει το περιεχόμενο που σχετίζεται με τη URL ειδοποίηση.

5.2 *Παράδειγμα Εκτέλεσης*

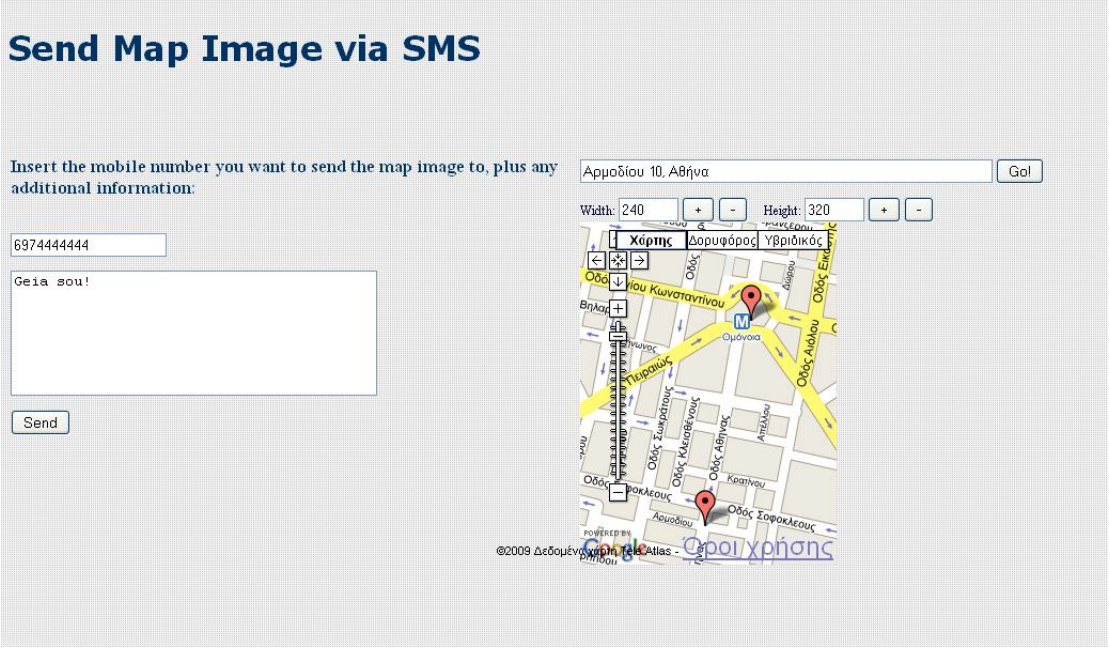
Για λόγους τεκμηρίωσης θεωρούμε ένα σενάριο στο οποίο υποθέτουμε ότι έχουμε δυο χρήστες: Α και Β. Ο Α, χρήστης ηλεκτρονικού υπολογιστή που έχει πρόσβαση στο ίντερνετ θέλει να δώσει ένα σημείο συνάντησης στο χρήστη Β που διαθέτει κινητή συσκευή με ανάλυση εικόνας 240Χ320 εικονοστοιχεία (pixels). Έστω ότι το σημείο συνάντησης είναι η διεύθυνση Αρμοδίου 10 στο κέντρο της Αθήνας, την οποία τυγχάνει ο χρήστης Β να μη γνωρίζει. Ο χρήστης Α μπορεί να χρησιμοποιήσει τη διαδικτυακή εφαρμογή Send Map Image via SMS (Εικόνα 6) και να στείλει μήνυμα στο κινητό του χρήστη Β επιδεικνύωντας του τη διεύθυνση και δίνοντας οδηγίες αν το επιθυμεί.



Εικόνα 6: Αρχική εμφάνιση ιστοσελίδας

Από τον ιστότοπο της εφαρμογής ο χρήστης Α χρησιμοποιεί τα πλήκτρα αυξομείωσης της ανάλυσης της εικόνας του χάρτη και επιλέγει τις επιθυμητές τιμές ανάλυσης της εικόνας. Στη συνέχεια πληκτρολογεί στη μπάρα

αναζήτησης Google “Αρμοδίου 10, Αθήνα” και η διεύθυνση επισημαίνεται με ένα δείκτη (marker). Επιπλεόν μπορεί να ανεβάσει το επίπεδο εστίασης της εικόνας (zoom in) για να είναι πιο ευδιάκριτα τα ονόματα στο χάρτη. Αυτό μπορεί να γίνει είτε πατώντας το πλήκτρο “+” στην κλίμακα μεγέθυνσης, είτε κάνοντας δεξί κλικ στο χάρτη και επιλέγοντας “Zoom In”. Ακόμη έχει τη δυνατότητα να προσθέσει ένα δείκτη (marker), κάνοντας δεξί κλικ και “Add Marker”, σε οποιοδήποτε σημείο του χάρτη. Ένα σημείο παραδείγματος χάριν μπορεί να είναι η πλατεία Ομονοίας ώστε να βοηθήσει περισσότερο το χρήστη Β να προσανατολιστεί. Αφού πληκτρολογήσει το κινητό τηλέφωνο του χρήστη Β και ένα μήνυμα κειμένου της επιλογής του, πατά Send (Εικόνα 7) και στέλνεται ένα wap push μήνυμα στην κινητή συσκευή του χρήστη Β. Αν όλα πήγαν καλά βλέπει στην οθόνη του ένα μήνυμα επιβεβαίωσης αποστολής μηνύματος (Εικόνα 8).



Εικόνα 7: Κατάσταση ιστοσελίδας λίγο πριν την αποστολή του μηνύματος

Μετά την αποστολή ο χρήστης Α βλέπει στην οθόνη του:

Εικόνα 8: Επιβεβαίωση αποστολής μηνύματος

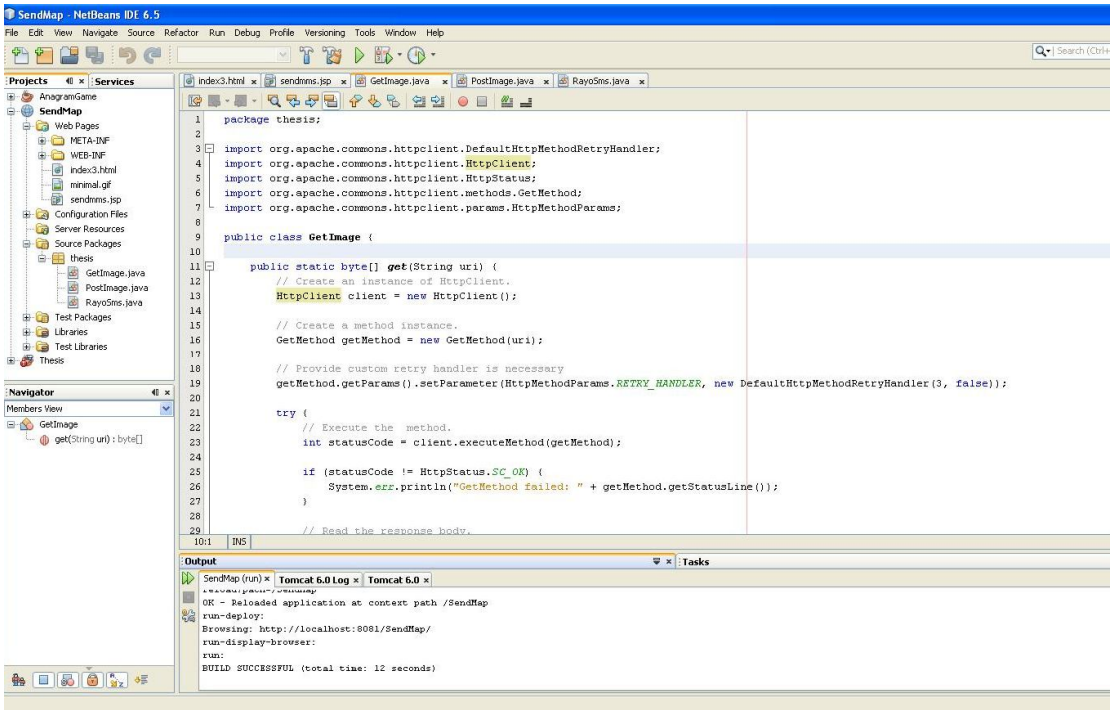
Ο χρήστης Β λαμβάνει ένα wap push μήνυμα στο κινητό του όπου του εμφανίζεται το κείμενο “Geia sou!” και μετά μπορεί να συνδεθεί απευθείας με το URL που θα έχει δεχθεί μέσω του φυλλομετρητή wap του κινητού του.

ΚΕΦΑΛΑΙΟ 6
ΛΕΠΤΟΜΕΡΕΙΕΣ ΥΛΟΠΟΙΗΣΗΣ

6.1 *Εργαλείο ανάπτυξης λογισμικού NetBeans*

Η υλοποίηση του λογισμικού πραγματοποιήθηκε στο περιβάλλον NetBeans (Εικόνα 9). Το NetBeans αποτελεί μία ανοιχτή πλατφόρμα ανάπτυξης λογισμικού σε γλώσσες Java, Javascript, PHP, Python, Ruby, Groovy, C και C++. Επίσης αποτελεί και ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) και προσφέρει εργαλεία εύκολα στη χρήση. Τα εργαλεία αυτά καλύπτουν όλα τα στάδια: από την υλοποίηση και τη μεταγλώττιση έως την εφαρμογή του λογισμικού που παράχθηκε.

Για την παρούσα εφαρμογή έγινε χρήση της έκδοσης NetBeans IDE 6.5



Εικόνα 9: Το περιβάλλον NetBeans IDE 6.5

6.2 *Εξυπηρετητής Apache Tomcat*

Ο Apache Tomcat [12] είναι ένας εξυπηρετητής δικτύου (web server) που εκπληρώνει τις προδιαγραφές της τεχνολογίας Java Server Pages και Java Servlets. Αναπτυγμένος από το Ίδρυμα Apache Λογισμικού, παρέχει ένα περιβάλλον εξυπηρετητή HTTP για java εφαρμογές.

Στα πλαίσια της εργασίας εγκαταστάθηκε και χρησιμοποιήθηκε ο Apache Tomcat 6.0.18

6.3 *Τεχνολογία JavaServer Pages (JSP)*

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε όλα τα λειτουργικά συστήματα χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Αυτό γίνεται με την ανάπτυξη της Εικονικής Μηχανής (Virtual Machine), πρόγραμμα το οποίο πρέπει να είναι εγκατεστημένο στο σύστημα, και το οποίο διαβάζει τα μεταγλωτισμένα αρχεία .class της java εφαρμογής και τα μεταφράζει σε κώδικα μηχανής (assembly) που υποστηρίζει το εκάστοτε λειτουργικό σύστημα και ο επεξεργαστής του συστήματος μας.

Εκμεταλλευόμενες τα πλεονεκτήματα της java, οι σελίδες java server (Java Server Pages – JSP) [13] είναι ιστοσελίδες που έχουν ενσωματωμένα δυναμικά έγγραφα java (jsp scriptlets). Ένα scriptlet είναι ένα μικρό κομμάτι εκτελέσιμου κώδικα που βρίσκεται σε μια σελίδα html. Οι σελίδες java server εκτελούνται δυναμικά στο διακομιστή (server), όπου βρίσκεται και η σχετική σελίδα, και ακολούθως ο πελάτης (client) λαμβάνει μια ιστοσελίδα στον φυλλομετρητή. Εκτελούνται δηλαδή από την πλευρά του διακομιστή (server side application), που σημαίνει ότι δέχονται μια αίτηση (request) και παράγουν μια απόκριση ή απάντηση (response). Οι σελίδες java server έχουν

πρόσβαση σε πηγές που βρίσκονται στην πλευρά του server (server-side resources), όπως είναι τα Servlets αλλά και οι βάσεις δεδομένων (databases).

6.4 *Πρωτόκολλο Μεταφοράς Υπερκειμένου (HTTP)*

Η επικοινωνία μέσω internet γίνεται χρησιμοποιώντας μια καθορισμένη ομάδα πρωτοκόλλων, μερικά από τα πιο γνωστά είναι το Πρωτόκολλο Μεταφοράς Αρχείων (File Transfer Protocol – FTP), το Πρωτόκολλο Μεταφοράς Απλών Μηνυμάτων (Simple Mail Transfer Protocol – SMTP), το Πρωτόκολλο Παραλαβής Ηλεκτρονικών Μηνυμάτων (Post Office Protocol - POP) και το Πρωτόκολλο Μεταφοράς Υπερκειμένου (Hypertext Transfer Protocol - HTTP).

Το πρωτόκολλο Μεταφοράς Υπερκειμένου [14] είναι ένα σύνολο κανόνων που καθορίζει τον τρόπο με τον οποίο θα γίνει η μεταφορά του υπερκειμένου (hypertext) μεταξύ του πελάτη (client) και του διακομιστή (server). Η ανάπτυξη του έγινε υπό την εποπτεία του World Wide Web Consortium και του Internet Engineering Task Force (IETF) και χρησιμοποιείται από τη συγκεκριμένη υπηρεσία δικτύου (WWW) από το 1990. Είναι το πιο συνηθισμένο στον ηλεκτρονικό χώρο του World Wide Web (WWW) καθώς οι αυξανόμενες υπηρεσίες και συσκευές δικτύου επεκτείνουν το ρόλο του HTTP πρωτοκόλλου πέραν από τους φυλλομετρητές ενώ αυξάνεται ο αριθμός των εφαρμογών που απαιτούν υποστήριξη του πρωτοκόλλου μεταφοράς υπερκειμένου.

6.4.1 Πακέτο Commons HTTPClient

Το Ίδρυμα Apache Λογισμικού (Apache Software Foundation - ASF) [12] είναι μια μη κερδοσκοπική εταιρεία παραγωγής ελεύθερου λογισμικού (open source software) η οποία έχει στόχο την υποστήριξη και δημιουργία των Apache πρότζεκτ. Η ιστορία της οργάνωσης αυτής ξεκινά το 1994 με την αρχή της ανάπτυξης του γνωστού Apache HTTP διακομιστή. Μια ομάδα οχτώ προγραμματιστών, με το όνομα Apache Group, προσπαθώντας αρχικά να βελτιώσουν τον NCSA HTTPd εξυπηρετητή, έναν από τους πρώτους εξυπηρετητές της εταιρείας National Center for Supercomputing Applications (NCSA), αποφάσισαν να ιδρύσουν το Ίδρυμα Apache Λογισμικού στις 13

Απριλίου του 1999. Έκτοτε έχουν αναπτυχθεί και αναγνωριστεί επίσημα πολλά πρότζεκτ τους, όπως ο διακομιστής HTTP, το Commons, ο Tomcat, το Jakarta, το HTTP components, σε ορισμένα από τα οποία γίνεται αναφορά παρακάτω.

Το Commons είναι ένα πρότζεκτ της ομάδας Apache που κύριος στόχος του είναι η δημιουργία και συντήρηση πακέτων σε γλώσσα Java των οποίων η χρήση είναι συχνή στις Java εφαρμογές. Αποτελείται από τρια μέρη: τα Commons Proper, Commons Sandbox και Commons Dormant.

Το Commons Proper αποτελεί την κύρια ομάδα πακέτων που είναι «δημοφιλή» στις java εφαρμογές. Βασικά χαρακτηριστικά των πακέτων αυτών είναι η ελάχιστη εξάρτηση τους από άλλες βιβλιοθήκες έτσι ώστε να μπορούν να φορτωθούν εύκολα και η σταθερότητα των διεπαφών τους που επιτρέπει στους χρήστες να τα εφαρμόζουν χωρίς να ανησυχούν για πιθανές μελλοντικές αλλαγές στην υλοποίησή τους. Σε αυτή την κατηγορία ανήκει το FileUpload component για ανάβασμα αρχείων σε ιστότοπο (web site).

Το Commons Sandbox είναι ένα περιβάλλον εργασίας όπου οι συνεργάτες του πρότζεκτ Commons πειραματίζονται πάνω σε νέα πακέτα που δεν έχουν συμπεριληφθεί ακόμα στο Commons Proper.

Το Commons Dormant είναι μια συλλογή πακέτων τα οποία έχουν χαρακτηριστεί ως «ανενεργά» εξ' αιτίας της μειωμένης ανάπτυξης τους πρόσφατα. Τα πακέτα αυτά μπορούν να χρησιμοποιηθούν αλλά θα πρέπει να μεταγλωττιστούν από τους χρήστες. Επίσης είναι σχεδόν σίγουρο ότι δε θα διατεθούν ολοκληρωμένα στο κοινό στο άμεσο μέλλον.

Άλλο ένα πρότζεκτ του Ιδρύματος Apache Λογισμικού είναι το λεγόμενο HTTPComponents, το οποίο παρέχει υποστήριξη για το βασικό πρωτόκολλο υπερκειμένου και χρησιμοποιείται σε εφαρμογές πελάτη και εξυπηρετητή, όπως φυλλομετρητές, http διακομιστές μεσολάβησης. Αποτελείται από δύο τμήματα: τα HttpComponents Core και HTTPComponents Client, όπου το δεύτερο προορίζεται για διάδοχος του Commons HTTPClient. Το πρότζεκτ HTTPComponents είναι υπεύθυνο για την ανάπτυξη και συντήρηση του πακέτου Commons HTTPClient που αποτελεί την τρέχουσα σταθερή βιβλιοθήκη επιλογής για τους περισσότερους χρήστες. Το HTTPClient είναι ουσιαστικά μια διεπαφή προγραμματισμού εφαρμογών που βοηθά τις java

εφαρμογές να επικοινωνούν μέσω του HTTP πρωτοκόλλου. Συμβάλλει στην ανάπτυξη κώδικα για τη μεριά του πελάτη (client side code) αλλά δε διαχειρίζεται αιτήσεις από τη μεριά του εξυπηρετητή.
Το πακέτο HTTPClient ξεκίνησε το 2001 σαν υποπρότζεκτ του Jakarta Commons αλλά αργότερα το 2007 το Commons έγινε ένα ανεξάρτητο πρότζεκτ υψηλού επιπέδου. Ομοίως και το πρότζεκτ HTTPComponents ανεξαρτητοποιήθηκε από το Jakarta και ανέλαβε την ευθύνη για τη διατήρηση του HTTPClient.

6.4.1.1 Η μέθοδος GetMethod

Η μέθοδος GET χρησιμοποιείται κυρίως για ανάκτηση πόρων από τον εξυπηρετητή, όπως μια HTML (HyperText Markup Language - Γλώσσα Μορφοποίησης Υπερκειμένου*)* σελίδα, ένα αρχείο ήχου, ένα αρχείο εικόνας. Επιπλέον χρησιμοποιείται για αποστολή δεδομένων στον εξυπηρετητή αλλά ο συνολικός αριθμός χαρακτήρων που δέχεται είναι περιορισμένος και τα δεδομένα εμφανίζονται στο URL κάτι το οποίο δεν είναι ασφαλές αφού θα είναι ορατά δημόσια.

Η γενική διαδικασία χρήσης του HTTPClient και της GetMethod είναι:

- Δημιουργία ενός στιγμιοτύπου κλάσης HttpClient
- Δημιουργία ενός στιγμιοτύπου κλάσης Getmethod με το URL που θέλουμε να συνδεθούμε
- Εκτέλεση της μεθόδου από τον HttpClient
- Ανάγνωση της απάντησης
- Ελευθέρωση της σύνδεσης
- Επεξεργασία της απάντησης

6.4.1.2 Η μέθοδος MultiPart Post Method

Η μέθοδος MultiPart Post είναι μια διαφορετική μορφή σώματος αίτησης μιας POST μεθόδου. Η μέθοδος POST χρησιμοποιείται για λήψη και κυρίως αποστολή μεγάλου όγκου δεδομένων στον εξυπηρετητή, όπως ένα αρχείο εικόνας. Πλεονέκτημα της είναι η απόκρυψη των σταλθέντων δεδομένων

μέσω του URL και γι' αυτό η χρήση της ενδείκνυται σε περιπτώσεις διασφάλισης δεδομένων.

Η MultiPart Post μέθοδος χρησιμοποιείται κυριώς για λήψη των πληροφοριών που προκύπτουν από την αποστολή δεδομένων δυο ειδών: ενός αλφαριθμητικού μέρους (StringPart) και ενός μέρους αρχείου (FilePart). Το αλφαριθμητικό μέρος αντιπροσωπεύει αλφαριθμητικά στοιχεία, όπως ένα κείμενο, ενώ το μέρος αρχείου είναι ένα αρχείο οποιουδήποτε τύπου δεδομένων και μέσω της MultiPart Post τα περιεχόμενα του ανεβαίνουν στον εξυπηρετητή που προσδιορίζεται στο URL.

6.5 *Παροχέας Δημοσίευσης Περιεχομένου*

Ο παροχέας υπηρεσιών που χρησιμοποιείται για τη δημοσίευση περιεχομένου [13] υποστηρίζει τα πρωτόκολλα HTTP και HTTPS (Hypertext Transfer Protocol Secure). Το HTTPS πρωτόκολλο αναφέρεται στο συνδυασμό του απλού HTTP πρωτοκόλλου και των δυνατοτήτων κρυπτογράφησης που παρέχει το πρωτόκολλο Ασφαλούς Επιπέδου Υποδοχής (Secure Socket Layer - SSL). Η κρυπτογράφηση που χρησιμοποιείται διασφαλίζει ότι τα κρυπτογραφημένα δεδομένα δεν θα μπορούν να υποκλαπούν από άλλους κακόβουλους χρήστες ή από επιθέσεις. Έτσι η χρήση του HTTPS πρωτοκόλλου συνίσταται στην υπηρεσία δημοσίευσης περιεχομένου, διότι παρέχει μια ασφαλή HTTP σύνδεση στην οποία τα προσωπικά στοιχεία του αποστολέα (transmitter), όπως αλλά και του παραλήπτη (receiver) θα προστατεύονται.

Η υπηρεσία αποστολής μηνύματος είναι διαθέσιμη μόνο στους πιστοποιημένους πελάτες οι οποίοι είνια εξουσιοδοτημένοι να την χρησιμοποιούν. Πιστοποιημένοι είναι οι πελάτες που διαθέτουν ενεργή εγγραφή στην υπηρεσία, δηλαδή τους έχει παραχωρηθεί όνομα χρήστη (username) και κωδικός (password).

Ο παροχέας υπηρεσιών έχει στόχο να μεταδίδει μηνύματα ειδικού περιεχομένου, wap push μηνύματα, τα οποία περιέχουν οποιασδήποτε μορφής πολυμεσικό περιεχόμενο, στην προκειμένη εικόνα.

6.6 *Περιγραφή Λειτουργίας*

Σε αυτή την παράγραφο θα περιγράψουμε πιο αναλυτικά την υλοποίηση της διαδικτυακής εφαρμογής Send Map Image via SMS.

index.html

Αρχικά, ο χρήστης έχει πρόσβαση στην εφαρμογή μέσω της σελίδας index.html. Με τη γλώσσα σήμανσης ή μορφοποίησης HTML (HyperText Markup Language), η οποία είναι μια περιγραφική γλώσσα, δηλαδή ένας ειδικός τρόπος γραφής κειμένου για την αναπαράσταση ιστοσελίδων, ορίζονται τα μορφολογικά χαρακτηριστικά της ιστοσελίδας. Επιπλέον, με τη βοήθεια της γλώσσας προγραμματισμού JavaScript, η οποία έχει σαν σκοπό την παραγωγή δυναμικού περιεχομένου σε ιστοσελίδες, ενσωματώνονται και φορτώνονται οι χάρτες Google. Η δυνατότητα αυτή παρέχεται μέσω της διεπαφής προγραμματισμού εφαρμογών Google Maps, βασική κλάση της οποίας είναι η Gmap2. Η Gmap2 χρησιμοποιείται για τη δημιουργία αντικειμένου χάρτη, πάνω στον οποίο προστίθενται ειδικοί ρυθμιστές για αυξομείωση του επιπέδου εστίασης και εναλλαγή μεταξύ τύπων χαρτών. Επιπλέον, δίνεται η δυνατότητα προσθήκης ή διαγραφής δεικτών (markers) καθώς και εύρεσης τοποθεσίας στο χάρτη μέσω της εισαγωγής κάποιας διεύθυνσης στη μπάρα αναζήτησης Google Maps. Μια πρόσθετη ιδιότητα της εφαρμογής είναι η επιλογή του μεγέθους του χάρτη με τη χρήση κουμπιών αυξομείωσης ύψους και πλάτους. Τέλος με τη συνάρτηση doSubmit όλες οι απαραίτητες παράμετροι στέλνονται στη JavaServer σελίδα προς επεξεργασία.

GetImage.java

Η συνάρτηση get της κλάσης αυτής δέχεται σαν παράμετρο μία URL διεύθυνση και μέσω της GetMethod του HTTPClient, επιστρέφεται το περιεχόμενο του URL με τη μορφή ενός πίνακα bytes.

PostData.java

Η συνάρτηση post της κλάσης αυτής δέχεται σαν παραμέτρους το κείμενο και την εικόνα χάρτη σε μορφή πίνακα bytes που θέλει να στείλει ο χρήστης στο μήνυμά του. Μέσω της Multipart PostMethod του HTTPClient τα δεδομένα αυτά αποθηκεύονται στο datastore του Google App Engine και επιστρέφεται ως απάντηση ένα μοναδικό κλειδί που αντιστοιχεί στη σελίδα που έχουν ανέβει τα δεδομένα.

ContentPublishing.java

Η κλάση αυτή περιέχει τη συνάρτηση sendContentUrlToMobile με την οποία εγκαθίσταται μια HTTPS σύνδεση με σκοπό την αποστολή των απαραίτητων στοιχείων στον παροχέα δημοσίευσης περιεχομένου. Στη συνέχεια ο παροχέας θα δημιουργήσει το WAP Push μήνυμα, το οποίο θα αποστείλει στην κινητή συσκευή.

sendmap.jsp

Η sendmap.jsp αποτελεί την κύρια σελίδα της εφαρμογής, μέσα στην οποία καλούνται όλες οι υπόλοιπες συναρτήσεις. Αρχικά συνθέτει τη Google Static Maps API URL διεύθυνση που αντιστοιχεί στην εικόνα χάρτη που έχει επιλέξει ο χρήστης, χρησιμοποιώντας τις παραμέτρους που λαμβάνει από την index.html σελίδα. Στη συνέχεια με τη βοήθεια των κλάσεων GetImage και PostData, επιστρέφεται η εικόνα χάρτη και μαζί με το γραπτό κείμενο ανεβαίνουν στο Google App Engine. Τέλος, καλείται η συνάρτηση SendContentUrlToMobile με τις κατάλληλες παραμέτρους, συμπεριλαμβανομένης της URL διεύθυνσης που θα χρησιμοποιηθεί για τη δημοσίευση περιεχομένου.

Τα παρακάτω αρχεία, τα οποία είναι γραμμένα στη γλώσσα προγραμματισμού Python, χρησιμοποιούνται από το περιβάλλον χρόνου εκτέλεσης Python του App Engine:

writedata.py

Πραγματοποιεί την αποθήκευση της εικόνας χάρτη και του κειμένου στο datastore απο την πλευρά του εξυπηρετητή Google App Engine.

getimage.py

Ανακτά την εικόνα χάρτη που είναι αποθηκευμένη στο datastore.

getpage.py

Δημιουργεί την HTML σελίδα που περιλαμβάνει το μήνυμα κειμένου και την εικόνα χάρτη έτσι ώστε να είναι ορατή από το χρήστη κατά τη σύνδεση του στην εφαρμογή του Google App Engine μεσω WAP.

ΠΑΡΑΡΤΗΜΑ
ΚΩΔΙΚΑΣ ΥΛΟΠΟΙΗΣΗΣ

Παρατίθεται ο κώδικας υλοποίησης της διαδικτυακής εφαρμογής Send Map Image via SMS:

Π1 index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
      <title>Send Map Image via SMS</title>
      <style type="text/css">

                  body,td,th {
                     font-size: 12px;
                     color: #000033;
                  }
                  body {
                     background-color: #F3F3F3;
                     background-image: url(minimal.gif);
                  }
                  h1 {
                     font-size: 24px;
                     color: #003366;
                  }
                  h2 {
                     font-size: 14px;
                     color: #000000;
                  }
```

```
                .style1 {font-family: Verdana, Arial, Helvetica, sans-serif}
                h3 {
                    font-size: 36px;
                    color: #003366;
                }
                a {
                    font-size: 24px;
                    color: #3333CC;
                }
                a:visited {
                    color: #660066;
                }
                .style3 {font-family: Verdana, Arial, Helvetica, sans-serif; font-
size: 32px; }
                .style5 {font-size: 22px}
                .style6 {
                    color: #003366;
                    font-size: 17px;
                    font-weight: bold;
                }
                .context {
                    font-family:Arial, sans-serif;
                    text-decoration:none;
                    color:#4444ff;
                    font-size:small;
                }
                a:hover div {
                    background:#eee;
                }

        </style>

        <!-- Insert the Google Maps in the web page -->
```

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAA
zr2EBOXUKnm_jVnk0OJI7xSosDVG8KKPE1-m51RBrvYughuyMxQ-
i1QfUnH94QxWIa6N4U6MouMmBA" type="text/javascript">
</script>

<script type="text/javascript">
  // Global variables
  var map = null;
  var geocoder = null;
  var clickedPixel = null;
  var contextmenu = document.createElement("div");
  var contextmenumarker = document.createElement("div");
  var MarkersArray = [];
  var Counter = 0;

  // Global functions in order to be available to top-level calls
  function addMarker() {
    var PointsArray = [];
    // Convert pixel location to latitude and longitude
    var latlng = map.fromContainerPixelToLatLng(clickedPixel);
    var point = new GLatLng(latlng.lat(),latlng.lng());
    PointsArray[Counter] = point;
    var marker = new GMarker(PointsArray[Counter]);
    // Keep a reference to the actual GMarker object to an array
    MarkersArray[Counter] = marker;
    map.addOverlay(marker);
    Counter++;
    // Hide the context menu now that it has been used
    contextmenu.style.visibility="hidden";
  }

  function removeMarker() {
    // Remove the last marker
```

```
    map.removeOverlay(MarkersArray[Counter-1]);
    Counter--;
    // Hide the context menu now that it has been used
    contextmenumarker.style.visibility="hidden";
}

function zoomIn() {
    // Perform the requested operation
    map.zoomIn();
    // Hide the menu now that it has been used
    contextmenu.style.visibility="hidden";
    contextmenumarker.style.visibility="hidden";
}

function zoomOut() {
    // Perform the requested operation
    map.zoomOut();
    // Hide the menu now that it has been used
    contextmenu.style.visibility="hidden";
    contextmenumarker.style.visibility="hidden";
}

// Call this function when the page has been loaded
function initialize() {
    if (GBrowserIsCompatible()) {
        map = new GMap2(document.getElementById("map_canvas"));
        map.setCenter(new GLatLng(37.977492, 23.710556), 12);
        map.addControl(new GLargeMapControl());
        map.addControl(new GMapTypeControl());
        geocoder = new GClientGeocoder();

        // Customize the contextmenu div
        contextmenu.style.visibility="hidden";
        contextmenu.style.background="#ffffff";
```

```
            contextmenu.style.border="1px solid #8888FF";

            contextmenu.innerHTML = '<a
href="javascript:addMarker()"><div class="context">  Add
Marker  <\/div><\/a>'
                + '<a href="javascript:zoomIn()"><div
class="context">  Zoom in  <\/div><\/a>'
                + '<a href="javascript:zoomOut()"><div
class="context">  Zoom out  <\/div><\/a>';

            map.getContainer().appendChild(contextmenu);

            // Customize the contextmenumarker div
            contextmenumarker.style.visibility="hidden";
            contextmenumarker.style.background="#ffffff";
            contextmenumarker.style.border="1px solid #8888FF";

            contextmenumarker.innerHTML = '<a
href="javascript:removeMarker()"><div
class="context">  Remove Marker  <\/div><\/a>'
                + '<a href="javascript:zoomIn()"><div
class="context">  Zoom in  <\/div><\/a>'
                + '<a href="javascript:zoomOut()"><div
class="context">  Zoom out  <\/div><\/a>';

            map.getContainer().appendChild(contextmenumarker);

            // Listen for singlerightclick
            // Inside function:
            // "pixel" is the location of the clicked pixel (weather right-clicked
on map or on marker)
            // "url" is the tile url if right-clicked on map or javascript:void(0) if
right-clicked on a marker
```

```
            // "overlay" is the map container if right-clicked on map, or a
reference to the GMarker object if right-clicked on a marker
            GEvent.addListener(map, "singlerightclick", function(pixel, url,
overlay) {

              // Define position of the contextmenu and contextmenumarker
(same)
              clickedPixel = pixel;
              var x=pixel.x;
              var y=pixel.y;
              // Adjust the context menu position accordingly
              if (x > map.getSize().width - 120) { x = map.getSize().width -
120 }
              if (y > map.getSize().height - 100) { y = map.getSize().height -
100 }

              // Create a GControlPosition() for the position of the context
menu
              var pos = new GControlPosition(G_ANCHOR_TOP_LEFT,
new GSize(x,y));

              // If rightclick is on the marker
              if (overlay) {
                 pos.apply(contextmenumarker);
                 contextmenumarker.style.visibility = "visible";
                 map.getContainer().appendChild(contextmenumarker);
              }
              // If rightclick is on the map
              else {
                 pos.apply(contextmenu);
                 contextmenu.style.visibility = "visible";
                 map.getContainer().appendChild(contextmenu);
              }
            });
```

```
            // if the user clicks on the map, close the context menu
            GEvent.addListener(map, "click", function() {
               contextmenu.style.visibility="hidden";
               contextmenumarker.style.visibility="hidden"
            });
         }

         // display a warning if the browser was not compatible
         else {
            alert("Sorry, the Google Maps API is not compatible with this
browser");
         }
      }

      // Find the address specified by the user
      function showAddress(address) {
         if (geocoder) {
            geocoder.getLatLng(
            address,
            function(point) {
               if (!point) {
                  alert(address + " not found");
               } else {
                  map.setCenter(point, 13);
                  var marker = new GMarker(point);
                  map.addOverlay(marker);
                  marker.openInfoWindowHtml(address);
               }
            }
         );
         }
      }
```

```
        function doSubmit() {
            // Get parameteres of the user's map image choice
            var c = map.getCenter();
            var z = map.getZoom();
            var w = map.getSize().width;
            var h = map.getSize().height;

            // Submit the form with POST method
            document.testform.lat.value = c.lat();
            document.testform.longt.value = c.lng();
            document.testform.zoom.value = z;
            document.testform.width.value = w;
            document.testform.height.value = h;
            document.testform.submit();
        }

        // Resize height of the map image
        function changeHeight(t) {
            var el = document.getElementById("height");
            el.value = Number(el.value)+t;
            document.getElementById("map_canvas").style.height =
el.value+'px';
            map.checkResize();
        }

        // Resize width of the map image
        function changeWidth(t) {
            var el = document.getElementById("width");
            el.value = Number(el.value)+t;
            document.getElementById("map_canvas").style.width =
el.value+'px';
            map.checkResize();
        }
```

```
    </script>
  </head>

  <body onload="initialize()" onunload="GUnload()">
    <h3 align="left" class="style3">Send Map Image via SMS</h3>
    <table width="1022" height="473" border="0">
      <tr>
        <td width="529" height="469">
          <form name="testform" method="post" action="sendmap.jsp"
onsubmit="return checkForm()">
            <input type="hidden" name="lat" />
            <input type="hidden" name="longt" />
            <input type="hidden" name="zoom" />
            <input type="hidden" name="width" />
            <input type="hidden" name="height" />
            <h2 class="style6">Insert the mobile number you want to send
the map image to, plus any additional information:</h2>
            <h2><br />
              <input type="text" name="mobile" />
            </h2>
            <p>
              <label>
                <textarea name="textarea" cols="40"
rows="6"></textarea>
              </label>
            </p>
            <p>
              <input type="button" value="Send" onclick="doSubmit();" />
            </p>
          </form>

          <p> </p>
          <p> </p>
          <p> </p>
```

```
            <label></label>
          <p align="left">  </p></td>
          <td width="483">

          <form action="#" onsubmit="showAddress(this.address.value);
return false">
            <p>
              <input type="text" size="60" name="address">
              <input type="submit" value="Go!" />
            </p>
            Width: <input id="width" type="text" name="width" value="448"
size="5">
            <input type="button" value="+" onclick="changeWidth(16);">
            <input type="button" value="-" onclick="changeWidth(-16);">
               
            Height: <input id="height" type="text" name="height" value="304"
size="5">
            <input type="button" value="+" onclick="changeHeight(16);">
            <input type="button" value="-" onclick="changeHeight(-16);">
            <div id="map_canvas" style="width: 448px; height: 304px"></div>
        </form>  </tr>
      </table>
      <h2 align="left" class="style1"> </h2>
      <h2 align="left" class="style1">
        <label><br />
        </label>
      </h2>
      <h1 align="left" class="style1">  </h1>
   </body>

</html>
```

Π2 sendmap.jsp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Send Map Image via SMS</title>
    <style type="text/css">

      body {
        background-color: #F3F3F3;
        background-image: url(minimal.gif);
      }

      h3 {
        font-size: 36px;
        color: #003366;
      }

      h2 {
        font-size: 14px;
        color: #000000;
      }

      .style1 {
        font-size: 32px;
        color: #003366;
        font-family: Verdana, Arial, Helvetica, sans-serif;
      }

      .style2 {
```

```
            color: #003366;
            font-size: 17px;
            font-weight: bold;
        }

        .style3 {font-size: 14px}

        .style4 {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 32px; }

    </style>
</head>

<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="javax.imageio.ImageIO, thesis.*" %>

<%
    // Access the values of the requested parameteres
    String latitude = request.getParameter("lat");
    String longitude = request.getParameter("longt");
    String zoom = request.getParameter("zoom");
    String width1 = request.getParameter("width");
    String height1 = request.getParameter("height");
    String mobile = request.getParameter("mobile");
    String textarea = request.getParameter("textarea");

    // Create the url of the image of the map we want
    String url = String.format("http://maps.google.com/staticmap?center=%1$s,%2$s&zoom=%3$s&size=%4$sx%5$s&key=MAPS_API_KEY&sensor=false", latitude, longitude, zoom, width1, height1);

    // Get the image to an array of bytes
```

```
byte[] myImage = GetImage.get(url);

// Post the text and image to Google App Engine's datastore
String key = PostData.post(textarea, myImage);

// Get the final page from Google App Engine's datastore
String uri = String.format("http://sendmap.appspot.com/getpage?key=%1$s", key);

// Content publishing
ContentPublishing message = ContentPublishing.getInstance();

String serviceUrl = "https://sms.rayo.gr/api/sms.sms";
String username = "mobics";
String password = "q2mob9dics";
String fromLabel = "sms.rayo.t3";
String phone = mobile;
String cpNotificationText = "Click on the link below to download the GoogleMap and get directions!";
String cpUrl = uri;
String clientRequestId = "1";
String callbackWhenProcessedFlag = "no";
String simulateSendFlag = "yes";

String returnMsg = message.sendContentUrlToMobile(serviceUrl, username, password, fromLabel, phone, cpNotificationText, cpUrl, clientRequestId, callbackWhenProcessedFlag, simulateSendFlag);
%>

<body>
<h3 align="left" class="style4">Send Map Image via SMS</h3>
<% if (returnMsg == "OK") { %>
<h2 class="style2">The following message was sent to <%= mobile%>.</h2>
```

```
        <%
        } else {
        %>
        <h2 class="style2">The following message could not be sent to <%= mobile%>.</h2>
        <% } %>

        <table width="301" height="109" border="1">
          <tr>
            <td align="center" bgcolor="#CCCCCC"><p align="center" class="style1 style3"><%= textarea%></p> </td>
          </tr>
        </table>
    </body>
</html>
```

Π3 GetImage.java

```
package thesis;

import org.apache.commons.httpclient.DefaultHttpMethodRetryHandler;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.params.HttpMethodParams;

public class GetImage {

    public static byte[] get(String uri) {
        // Create an instance of HttpClient
```

```
        HttpClient client = new HttpClient();

        // Create a get method instance
        GetMethod getMethod = new GetMethod(uri);

        // Provide custom retry handler is necessary

getMethod.getParams().setParameter(HttpMethodParams.RETRY_HANDLE
R, new DefaultHttpMethodRetryHandler(3, false));

        try {
            // Execute the get method
            int statusCode = client.executeMethod(getMethod);

            if (statusCode != HttpStatus.SC_OK) {
                System.err.println("GetMethod failed: " +
getMethod.getStatusLine());
            }

            // Read the response body
            byte[] responseBody = getMethod.getResponseBody();

            return responseBody;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } finally {
            // Release the connection
            getMethod.releaseConnection();
        }
    }
}
```

Π4 PostData.java

```
package thesis;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.HttpStatus;
import
org.apache.commons.httpclient.methods.multipart.MultipartRequestEntity;
import org.apache.commons.httpclient.methods.multipart.Part;
import org.apache.commons.httpclient.methods.multipart.FilePart;
import
org.apache.commons.httpclient.methods.multipart.ByteArrayPartSource;
import org.apache.commons.httpclient.methods.multipart.StringPart;

public class PostData {

   public static String post(String text, byte[] image) {

      String url = "http://sendmap.appspot.com/writedata";
      String res = null;

      // Create an instance of HttpClient
      HttpClient client = new HttpClient();

      // Create a poost method instance
      PostMethod filePost = new PostMethod(url);

      try {
         // Define the parts of multipart post method
         Part[] parts = {
            new FilePart("image", new ByteArrayPartSource("image.gif",
image)),
```

```
            new StringPart("text", text)
        };
        filePost.setRequestEntity(new MultipartRequestEntity(parts,
filePost.getParams()));

client.getHttpConnectionManager().getParams().setConnectionTimeout(5000)
;

        // Execute the multipart post method
        int status = client.executeMethod(filePost);

        if (status != HttpStatus.SC_OK) {
            System.err.println("PostMethod failed: " + filePost.getStatusLine());
        }

        // Read the response body
        res = new String(filePost.getResponseBody());
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        // Release the connection
        filePost.releaseConnection();
    }
    return res;
  }
}
```

Π5 ContentPubllishing.java

```
package thesis;

import java.net.*;
import java.io.*;
import java.security.cert.*;
import javax.net.ssl.*;

public class ContentPublishing {

   private static ContentPublishing sms = null;

   private ContentPublishing() {
   }

   public static ContentPublishing getInstance() {

      if (sms == null) {
         sms = new ContentPublishing();
      }

      return sms;
   }

   public String sendContentUrlToMobile(String urlToSend, String usr, String
pwd,
         String from, String phone, String cpNotificationText,
         String cpUrl,
         String clientRequestId, String callbackWhenProcessedFlag,
         String simulateSendFlag)

         // Exception if one of the parameters is missing
```

```
        throws Exception {
    if (urlToSend == null || urlToSend.equals("") ||
            usr == null || usr.equals("") ||
            pwd == null || pwd.equals("") ||
            cpNotificationText == null || cpNotificationText.equals("") ||
            cpUrl == null || cpUrl.equals("") ||
            from == null || from.equals("") ||
            phone == null || phone.equals("")) {

        throw new Exception("sendToMobile:EMPTY PARAMETERS");
    }

    if (callbackWhenProcessedFlag == null ||
!callbackWhenProcessedFlag.equals("yes")) {
        callbackWhenProcessedFlag = "no";
    }

    if (simulateSendFlag == null || !simulateSendFlag.equals("yes")) {
        simulateSendFlag = "no";
    }

    HttpsURLConnection https = null;
    InputStream input_stream = null;
    OutputStreamWriter out = null;

    try {
        SSLContext sslctx = SSLContext.getInstance("SSL");
        sslctx.init(null, new X509TrustManager[]{new MyTrustManager()},
                null);


HttpsURLConnection.setDefaultSSLSocketFactory(sslctx.getSocketFactory())
;
```

```
        HttpsURLConnection.setDefaultHostnameVerifier(new
MyHostnameVerifier());
        Authenticator.setDefault(new MyAuthenticator(usr, pwd));
        URL url = new URL(urlToSend);
        https =
              (HttpsURLConnection) url.openConnection();

        String data = new String();
        String name = "action";
        String value = "asmsmt";
        data += name + "=" + value + "&";

        name = "username";
        value = usr;
        data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

        name = "password";
        value = pwd;
        data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

        name = "toMobileNumber";
        value = "";

        // Building the phone number
        if (phone.indexOf("69") == -1) {
           return "NOT_OK";
           //throw new Exception("sendToMobile:Wrong Number Format");
        }

        if (phone.indexOf("69") == 0) {
           phone = "+30" + phone;
        }

        if (phone.indexOf("30") == 0) {
```

```
    phone = "+" + phone;
}

if (phone.indexOf("00") == 0) {
    phone = "+" + phone.substring(2);
}

if (phone.length() != 13) {
    return "NOT_OK";
}

if (value.length() != 0) {
    value = value + ";";
}
value = value + phone;
data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

name = "fromLabel";
value = from;
data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

name = "cpNotificationText";
value = cpNotificationText;
data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

name = "cpUrl";
value = cpUrl;
data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

if (clientRequestId != null && clientRequestId.length() > 0) {
    name = "clientRequestId";
    value = clientRequestId;
    data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";
}
```

```
name = "simulateSendFlag";
value = simulateSendFlag;
data += name + "=" + URLEncoder.encode(value, "UTF-8") + "&";

name = "callbackWhenProcessedFlag";
value = callbackWhenProcessedFlag;
data += name + "=" + URLEncoder.encode(value, "UTF-8");

//-----------------
https.setRequestMethod("POST");
https.setDoOutput(true);
https.setRequestProperty("User-Agent", "MSIE 6.0");
https.setRequestProperty("Content-type",
      "application/x-www-form-urlencoded;charset=UTF-8");
https.setRequestProperty("Content-length",
      String.valueOf(data.length()));
https.setRequestProperty("Cache-Control", "no-cache");
https.setRequestProperty("Connection", "close");

https.connect();
out =
      new OutputStreamWriter(https.getOutputStream(), "UTF-8");

// Post the data
out.write(data);

out.flush();

// Get the answer
input_stream = https.getInputStream();
InputStreamReader input_stream_reader =
      new InputStreamReader(input_stream, "UTF-8");
BufferedReader buffered_reader =
```

```
                new BufferedReader(input_stream_reader);

            String line = null;
            while ((line = buffered_reader.readLine()) != null) {
                System.out.println(line);

            }

        } catch (Exception e) {

            e.printStackTrace();
            throw new Exception(e);

        } finally {

            if (out != null) {
                out.close();
            }

            if (input_stream != null) {
                input_stream.close();
            }

            if (https != null) {
                https.disconnect();
            }

        }

        return "OK";

    }

    class MyTrustManager implements X509TrustManager {
```

```
        public void checkClientTrusted(X509Certificate[] chain, String authType)
{
        }

        public void checkServerTrusted(X509Certificate[] chain, String authType)
{
        }

        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }
    }

    class MyHostnameVerifier implements HostnameVerifier {

        public boolean verify(String urlHostName, SSLSession session) {
            return true;
        }
    }

    class MyAuthenticator extends Authenticator {

        private String m_strUsername = "";
        private String m_strPassword = "";

        public MyAuthenticator(String i_strUsername, String i_strPassword) {
            super();
            m_strUsername = i_strUsername;
            m_strPassword = i_strPassword;
        }

        @Override
        protected PasswordAuthentication getPasswordAuthentication() {
```

```
            return new PasswordAuthentication(m_strUsername,
                    m_strPassword.toCharArray());
        }
    }
}
```

Π6 writedata.py

```
import cgi
from google.appengine.ext import db

class AreaMap(db.Model):
  text = db.StringProperty()
  image = db.BlobProperty()

form = cgi.FieldStorage()
aimage = form["image"].value
atext = form["text"].value
e = AreaMap(image=aimage, text=atext)
e.put()

print('Content-Type: text/plain')
print('')
print(str(e.key()))
```

Π7 getimage.py

```
import cgi
from google.appengine.ext import db

class AreaMap(db.Model):
  text = db.StringProperty()
  image = db.BlobProperty()

form = cgi.FieldStorage()
akey = form["key"].value
e = db.get(db.Key(akey))

print('Content-Type: image/gif')
print('')
print(str(e.image))
```

Π8 getpage.py

```
import cgi
from google.appengine.ext import db

baseurl = "http://sendmap.appspot.com/getimage"

class AreaMap(db.Model):
  text = db.StringProperty()
  image = db.Blob()

form = cgi.FieldStorage()
akey = form["key"].value
```

```
e = db.get(db.Key(akey))

print('Content-Type: text/html')
print("""
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <p>%s</p>
      <img src="%s?key=%s">
   </body>
</html>
""" % (e.text,baseurl,akey))
```

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Αγγλική Ορολογία	Ελληνική Ορολογία
Short Message Service (SMS)	Υπηρεσίας Σύντομου Μηνύματος
gateway	Πύλη δικτύου
Application Programming Interface (API)	διεπαφή προγραμματισμού εφαρμογών
Java Runtime Environment	περιβάλλον χρόνου εκτέλεσης Java
Python Runtime Environment	περιβάλλον χρόνου εκτέλεσης Python
Push Proxy Gateway (PPG)	Ενδιάμεσος εξυπηρετητής ώθησης
Multimedia Message Service (MMS)	Υπηρεσία Πολυμεσικών Μηνυμάτων
web application	διαδικτυακή εφαρμογή
web browser	φυλλομετρητής ιστοσελίδων
internet	διαδίκτυο
intranet	ενδοδίκτυο
client	πελάτης
server	εξυπηρετητής / διακομιστής
web page	ιστοσελίδα
web site	ιστότοπος
Document Object Model (DOM)	Μοντέλο Αντικειμένου Εγγράφου
user interface	διεπαφή χρήστη
checkbox	κουμπί επιλογής
marker	δείκτης
Marker Manager	διαχειριστής δεικτών
host	φιλοξενώ
dynamic web serving	δυναμική εξυπηρέτηση
datastore	χώρος αποθήκευσης
Hypertext Transfer Protocol (HTTP)	Πρωτόκολλο Μεταφοράς Υπερκειμένου
request	αίτηση
response	απάντηση
open source software	ελεύθερο λογισμικό
entity	οντότητα

multimedia content	περιεχόμενο πολυμέσων
link	υπερσύνδεσμος
hypertext	υπερκείμενο
socket	υποδοχή
map rendering	απεικόνιση του χάρτη
zoom level	επίπεδο εστίασης
pull	τραβώ / εξάγω

ΑΝΑΦΟΡΕΣ

[1] Tomasz Imieliński και Henry F. Korth, "*Mobile Computing*", Kluwer Academic Publishers, 1996.

[2] Openwave Systems, Άρθρο "*The value of WAP Push*", 2001.

[3] Woojong Suh, "Web Engineering: Principles and Techniques", Idea Group Inc.

[4] Google, Google Maps API, http://code.google.com/intl/el-GR/apis/maps/documentation/index.html

[5] David Flanagan, "*Javascript: The Definitive Guide", Fifth Edition*, O'Reilly, 2006.

[6] Google, Google Static Maps API, http://code.google.com/intl/el-GR/apis/maps/documentation/staticmaps/

[7] Wikipedia, Google App Engine, http://en.wikipedia.org/wiki/Google_App_Engine

[8] Ιωάννης Παπαρρίζος, "*Cloud Computing and Windows Azure*", Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Τμήμα Πληροφορικής.

[9] Google, Google App Engine, http://code.google.com/intl/el-GR/appengine/

[10] Wikipedia, Wireless Application Protocol, http://en.wikipedia.org/wiki/Wireless_Application_Protocol

[11] Wikipedia, Push Proxy Gateway, http://en.wikipedia.org/wiki/Push_Proxy_Gateway

[12] Apache Software Foundation, 1994, http://www.apache.org/

[13] Hans Bergsten, "*JavaServer Pages",* 3rd Edition, O'Reilly, 2003.

[14] Jason Hunter και William Crawford, "Java Servlet Programming", 2nd Edition, O'Reilly, 2001.