ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Θέμα:

"Μελέτη και ανάπτυξη εφαρμογής για λήψη ειδησεογραφικού υλικού. (FeedUp)"

Σπουδαστές:

Σοφία Καρκάνη
Δημήτρης Πεντεδέκας

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστούμε τους καθηγητές μας που μας ενέπνευσαν και μας διδάξαν κατά την διάρκεια των σπουδών μας. Ιδιαίτερες ευχαριστίες στον επιβλέποντα καθηγητή κύριο Μιχάλη Ιορδανάκη, για τις πολύτιμες συμβουλές του και για τον χρόνο που αφιέρωσε, καθώς και στον κύριο Γεώργιο Διλιντά για τις πολύτιμες γνώσεις που μας έδωσε αλλά και για το έναυσμα να ασχοληθούμε με το συγκεκριμένο γνωστικό στοιχειό.

Ευχαριστούμε επιπλέον τους φίλους μας και αποφοίτους του Τμήματος Η.Υ.Σ. Ελένη Χουρουζίδου και Γιάννη Δεληγιάννη για την βοήθεια τους.

Πίνακας περιεχομένων

Εισαγωγή

Πρόλογος

Με την ραγδαία ανάπτυξη της τεχνολογίας και πιο συγκεκριμένα των εφαρμογών κινητών τηλεφώνων καθώς και των εφαρμογών μέσω διαδικτύου είναι όλο και πιο συχνό φαινόμενο ο συνδυασμός των δυο αυτών τομέων για την πληρέστερη υλοποίηση μιας εφαρμογής. Ο χρήστης έχει την δυνατότητα, είτε μέσω του υπολογιστή του είτε μέσω του κινητού τηλέφωνου, να κάνει χρήση της εφαρμογής και να έχει πρόσβαση στην πληροφόρα και τις υπηρεσίες της.

Η εφαρμογή που υλοποιήσαμε σχετίζεται και με το κομμάτι του διαδικτύου αλλά και με το κομμάτι της κινητής τηλεφωνίας και εντάσσεται στο πεδίο της ανάπτυξης λογισμικού. Οι τεχνολογίες που χρησιμοποιήσαμε είναι αρκετά σύγχρονες και καινοτόμες ενώ ιδιαίτερη έμφαση δόθηκε στην ευκολία χρήσης της εφαρμογής καθώς και στο περιβάλλον του χρηστή. Στο κομμάτι του κινητού δόθηκε επιπλέον έμφαση στην ομαλή λειτουργιά της εφαρμογής έτσι ώστε να μην προκύψουν προβλήματα με την συσκευή του κινητού τηλέφωνου.

Σκοπός Πτυχιακής Εργασίας

Σκοπός της πτυχιακής εργασίας είναι η ανάπτυξη εφαρμογής για την λήψη δεδομένων τύπου Rss από κινητό τηλέφωνο με λειτουργικό σύστημα Android και εφαρμογής διαδικτύου σε γλώσσα Ruby on Rails. Ο χρήστης θα μπορεί να έχει πρόσβαση σε βασικές κατηγορίες ειδήσεων όπως αθλητικά και πολιτικά μέσω του υπολογιστή του στην ιστοσελίδα που έχει ο server. Η ιδία πληροφορία είναι προσβάσιμη και από το κινητό τηλέφωνο. Εφόσον επιθυμεί ο χρήστης μπορεί να δημιουργήσει έναν προσωπικό λογαριασμό είτε απο τον υπολογιστή του είτε απο το κινητό του τηλέφωνο και μέσω αυτού να προσθέτει τις δίκες του κατηγορίες που σχετίζονται με τα ενδιαφέροντά του.

Δομή πτυχιακής εργασίας

Στο πρώτο μέρος αναφέρεται και αναλύεται το κομμάτι του server όπου χωρίζεται σε επιμέρους κεφάλαια ανάλογα με τις τεχνολογίες και τις μεθόδους που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής του διαδικτύου.

Στο δεύτερο μέρος αναφέρεται και αναλύεται το κομμάτι της εφαρμογής του κινητού τηλέφωνου όπου χωρίζεται σε επιμέρους κεφάλαια ανάλογα με τις τεχνολογίες και τις μεθόδους που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής του διαδικτύου.

Μέρος Α': Ruby on Rails

Κεφάλαιο 1: Η γλώσσα Ruby

Η γλώσσα Ruby δημιουργήθηκε το 1993 από τον Ιάπωνα Yukihiro Matsumoto και δημοσιεύτηκε στο ευρύ κοινό το 1995. Σκοπός του Yokihiro ήταν η δημιουργία μιας γλώσσας προγραμματισμού που θα διευκόλυνε τον προγραμματιστή να επικεντρώνεται κυρίως στη συγγραφή κώδικα και οχι στην επίλυση παράπλευρων προβλημάτων που συχνά προέκυπταν από την ίδια τη φύση της γλώσσας. Έτσι δανείστηκε τα δυνατά γνωρίσματα άλλων γλωσσών όπως π.χ. Python, Perl, Smalltalk, Eiffel και Lisp.

Στην Ruby τα πάντα αντιμετωπίζονται ως αντικείμενα και έχουν τις δικές τους ιδιότητες και μεθόδους. Λόγω του απλό τρόπου σύνταξης της σε συνδυασμό με τη δυνατότητα που μας δίνουν τα αντικείμενα, μπορεί αν φτιαχτεί μια πολύπλοκη εφαρμογή σε μικρό χρονικό διάστημα. Γι' αυτό το λόγο έγινε γρήγορα διάσημη στην Ιαπωνία και τελικά από το 2000 αναγνωρίστηκε και από τον υπόλοιπο προγραμματιστικό κόσμο. Ένα στοιχείο που έκανε τη γλώσσα Ruby τόσο διάσημη είναι το framework της Ruby on Rails.

1.1 Γενικά Χαρακτηριστικά

1. Τα πάντα είναι αντικείμενα
2. Είναι cross-platform (Linux, Windows, Mac OS). Δε χρειάζεται κάποιο σύμβολο στο τέλος των εντολών, ούτε εξαρτάται από τον αριθμό των κενών όπως είναι στην python. Οι συναρτήσεις που δε δέχονται ορίσματα μπορούν να κληθούν και χωρίς παρενθέσεις.
3. Οι κλάσεις είναι ανοικτές. Ο κώδικας μπορεί να αλλαχτεί οποιαδήποτε στιγμή.
4. Blocks. Τα blocks στη ruby είναι το πιο σημαντικό γνώρισμα. Είναι κομμάτια κώδικα που μπορούν να περαστούν σαν παράμετροι σε μια συνάρτηση ή να κληθούν με την εντολή yield.
5. Τα πάντα έχουν μια τιμή. Δεν υπάρχει διαφορά μεταξύ μιας έκφρασης και μιας δήλωσης. Όλα έχουν μια τιμή, έστω κι αν αυτή είναι nil.
6. Τα σύμβολα μπορούν να περιγραφούν σαν ταυτότητες. Η διαφορά τους με τα strings φαίνεται στο παρακάτω παράδειγμα.

```
:foo.object_id==:foo.object_id  =>επιστρέφει true
"foo".object_id=="foo".object_id  =>επιστρέφει false
```

Σύμβολα με ίδιους χαρακτήρες αναφέρονται στο ίδιο αντικείμενο στην κύρια μνήμη. Αντίθετα τα strings δεν αναφέρονται στο ίδιο αντικείμενο. Η Ruby δεσμεύει χώρο στη μνήμη κάθε φορά που δημιουργείται ένα νέο string.

7. Στη ruby τα πάντα εκτός από nil και το false θεωρούνται true. Το 0 που σε άλλες γλώσσες θεωρείται false στη ruby παίρνει την τιμή true.
8. Υποστηρίζει παράλληλα μετάθεση. Μπορούμε να αλλάξουμε πολλαπλές μεταβλητές με μια ανάθεση. Για παράδειγμα a , b = b, a
9. Το return είναι προαιρετικό.
10. Οι τελεστές ++ και - - δεν υποστηρίζονται. Αυτό οφείλεται στο ότι στη Ruby όλα είναι αντικείμενα, οπότε τα σύμβολα +,- [] κτλ. είναι μέθοδοι. Επομένως, η γραφή ++ και – δε συμβαδίζει με αυτή τη λογική.
11. Υποστηρίζει singleton μεθόδους. Οι συναρτήσεις αυτές είναι διαθέσιμες για το αντικείμενο που ορίστηκαν. Για παράδειγμα:

```
obj = Object.new
def obj.talk
puts "Hi!"
end
obj.talk
```

1.2 Γιατί Ruby

Είναι μια καθαρή γλώσσα προγραμματισμού. Αυτό καθιστά την εκμάθηση της να είναι ευκολότερη σε σχέση με άλλες γλώσσες. Οι εντολές είναι απλές και συγκεκριμένες. Για παράδειγμα

```
777.times do { puts "Ruby" }
```

Έχει αυτόματη διαχείριση μνήμης και δε υπάρχει ανάγκη για τη δήλωση μεταβλητών. Επίσης είναι μια σημαντική γλώσσα γενικού σκοπού. Μπορεί να χρησιμοποιηθεί για να γράψουμε scripts με τον ίδιο τρόπο που θα χρησιμοποιούσε κάποιος Perl ,και επίσης μπορεί να χρησιμοποιηθεί και για την δημιουργία ολοκληρωμένων, μεγάλης κλίμακας, αυτόνομων GUI- βασιζόμενων εφαρμογών. Επιπλέον η Ruby είναι εξαιρετική για υποστήριξη ιστοσελίδων, δημιουργώντας περιεχόμενο δυναμικών ιστοσελίδων και προσφέροντας διαδικασίες προσπέλασης βάσεων δεδομένων. Επίσης είναι και εύκολα επεκτάσιμη, καθώς μπορούμε να ενσωματώσουμε συναρτήσεις, ή ακόμα και βιβλιοθήκες. Κλείνοντας φυσικά να αναφέρουμε και τη μεταφερσιμότητα που έχει ως διερμηνευόμενη γλώσσα. Οποιαδήποτε εφαρμογή που έχει αναπτυχθεί σε Ruby μπορεί να εκτελεστεί ισότιμα σε οποιαδήποτε πλατφόρμα την υποστηρίζει.

1.3 Εγκατάσταση της Ruby

1. Κατέβασμα του πηγαίου κώδικα από την επίσημη ιστοσελίδα της Ruby http://www.ruby-lang.org/en
2. Για την εγκατάσταση της γλώσσας στο linux πρέπει να μεταγλωτιστεί ο πηγαίος κώδικας με την εντολή ./ install, αφού πρώτα πάμε στο φάκελο που βρίσκεται ο πηγαίος κώδικας.

Υπάρχουν τρεις διαφορετικοί τρόποι για να τρέξουμε ενα πρόγραμμα στη ruby:

- Από αρχείο

Δημιουργούμε ένα αρχείο, γράφουμε τον κώδικα που θέλουμε σε ruby και στη συνέχεια το σώζουμε με κατάληξη .rb (Για παράδειγμα foo.rb). Έπειτα το τρέχουμε με την εντολή ruby. Για παράδειγμα ruby foo.rb

- από irb (είναι ένα ruby shell με αλληλεπίδραση)

Με την εγκατάσταση της ruby έχουμε και το πρόγραμμα irb.Το irb παίρνει την εντολή που θέλουμε να εκτελέσουμε και εμφανίζει το αποτέλεσμα της εκτέλεσης στην οθόνη.

- ως εκτελέσιμο

Δημιουργούμε ένα αρχείο, γράφουμε τον κώδικα που θέλουμε και στη συνέχεια το σώζουμε με κατάληξη .rb. Αλλάζουμε τα δικαιώματα πρόσβασης έτσι ώστε να μπορεί να εκτελεστεί, και τέλος το τρέχουμε σαν εκτελέσιμο.

1.4 Ruby Gems

Είναι οι βιβλιοθήκες της Ruby. Για να δούμε ποιες βιβλιοθήκες έχουμε εγκατεστημένες πατάμε στο τερματικό: gem -v

Η σελίδα που μπορούμε να βρούμε gems είναι η http://rubygems.org/. Ο τρόπος εγκατάστασης και η διαχείρησή τους θυμίζει linux. Π.χ. Ανοίγουμε το «Start Command Prompt with Ruby» (σε unix λειτουργικά δε θα χρειαστεί) γράφοντας «gem install the_gem» εγκαθιστάται το gem με όνομα the_gem. Με την εντολή «gem list» μπορούμε να δούμε ποια gems υπάρχουν αυτή τη στιγμή στο μηχάνημά που δουλεύουμε. Μάλιστα υπάρχει η δυνατότητα να έχουμε και πολλαπλές εκδόσεις του ίδιου gem.Η ενσωμάτωση των gem στον κώδικά γίνεται με τις εξής 2 εντολές στην αρχή του αντίστοιχου αρχείου:

```
require 'rubygems'
require 'the_gem'
```

Η πρώτη εντολή λέει ότι θα χρειαστούμε gem και η δεύτερη ποιο ακριβώς. Το " rubygems'" είναι ένα gem που αναλαμβάνει τη διαχείριση των gems!

Ακόμη, έχουμε τη δυνατότητα να θέσουμε κριτήρια για την έκδοση. Π.χ.

```
require 'rubygems'
gem 'activerecord', '>= 1.4.0'
gem 'actionpack', '<= 1.2.0'
gem 'actionmailer', '= 0.5.0'
gem 'rails', '= 0.9.3'
```

Έτσι, η Ruby θα ελέγξει εάν το σύστημα στο οποίο τρέχει ο κώδικας έχει όχι μόνο τα συγκεκριμένα gems αλλά και κάποια έκδοση που να ικανοποιεί αυτά τα κριτήρια. Έτσι, εάν για κάποιο λόγο χρησιμοποιούμε συγκεκριμένη έκδοση gem ενώ π.χ. η επόμενη δημιουργεί λάθη στο πρόγραμμα (λόγω π.χ. αλλαγών στο gem) ορίζουμε την συγκεκριμένη έκδοση που θέλουμε.

Κεφάλαιο 2: Frameworks – Ruby on Rails

Τα τελευταία χρόνια παρατηρούμε μια έκρηξη τόσο στη χρήση όσο και στην παραγωγή των web frameworks. Τι είναι όμως το framework; Το framework είναι μια συλλογή από βιβλιοθήκες και εργαλεία που προορίζονται να διευκολύνουν την ανάπτυξη εφαρμογών. Σχεδιασμένο με την παραγωγή στο μυαλό, ένα καλό framework παρέχει μια βασική αλλά πλήρη υποδομή πάνω στην οποία μπορεί να οικοδομηθεί μια εφαρμογή.

Έχοντας ένα καλό framework, ένα βασικό κομμάτι της εφαρμογής είναι έτοιμο γραμμένο. Αντί ο προγραμματιστής να ξεκινήσει να γράφει κώδικα από το μηδέν, ξεκινάει με μια ήδη υπάρχουσα βάση. Ένα καλό framework μπορεί να περιγραφεί ώς εξης:

- Full stack(πλήρης στοίβα): Ότι χρειάζεται για τη δημιουργία ολοκληρωμένων εφαρμογών θα πρέπει να περιλαμβάνεται στο πακέτο. Μπορεί να εγκαταταστήσει διαφορετικές βιβλιοθήκες και να ρυθμίσει ο προγραμματιστής τα components που θα χρησιμοποιήσει. Τα διαφορετικά στρώματα πρέπει να ταιριάζουν αρμονικά μεταξύ τους.
- Open Source (ανοικτού κώδικα): Ένα framework θα πρέπει να είναι ανοικτού κώδικα.
- Cross-platform: Ένα καλό framework είναι ανεξάρτητο από την πλατφόρμα(λειτουργικό σύστημα) που το τρέχει.

Ένα καλό framework παρέχει τα ακόλουθα:

- Μια θέση για τα πάντα: Η δομή είναι πολύ σημαντική για ένα framework. Τα πάντα θα πρέπει να έχουν μια κατάλληλη θέση μέσα στο σύστημα. Αυτό αυξάνει την παραγωγικότητα.
- Database abstraction layer: Ένα καλό framework φροντίζει για το μεγαλύτερο μέρος της εργασίας με τη βάση και λειτουργεί με σχεδόν οποιαδήποτε βάση δεδομένων.

Το πιο διάσημο framework της Ruby είναι η Ruby on Rails ή για συντομία Rails. Είναι πλήρης, open-source, και cross-platform. Παρέχει ένα ισχυρό database abstraction layer που ονομάζεται Active Record, το οποίο συνεργάζεται με τα πιο δημοφιλή συστήματα βάσεων δεδομένων. Αρχικά δημιουργήθηκε από τον David Heinemeier Hansson στο Basecamp, ένα εργαλείο διαχείρισης project από την εταιρεία 37signals. Στόχος της σαν framework είναι η επίλυση του 80% των προβλημάτων που παρουσιάζονται στην ανάπτυξη ιστοσελίδων, με στόχο ότι το υπόλοιπο 20% είναι μοναδικό για την κάθε εφαρμογή. Η rails έχει συγκεκριμένη δομή καταλόγου, ονομασίες αρχείων,data structures, method arguments.Όταν γράφει κάποιος μια εφαρμογή σε rails, αναμένεται να τηρηθουν οι συμβάσεις που έχουν τεθεί ήδη.

Αντι να επικεντρωνεται ο προγραμματιστης στις λεπτομέρειες της ανάπτυξης της εφαρμογής, θα επικεντρωθεί στο 20% που έχει πραγματικά σημασία.

2.1 Η RAILS είναι η RUBY

Πριν εμφανιστεί η rails, δεν υπήρχαν πολλά άτομα τα οποία ήξεραν να γράφουν εφαρμογές σε ruby. Άλλες γλώσσες όπως η PHP και η ASP ήταν οι βασικές γλώσσες για την ανάπτυξη web εφαρμογών.Το γεγονός οτι η rails είναι γραμμένη σε ruby είναι σημαντικό, γιατί η ruby θεωρείται πιο δυνατή σε σχέση με την php και την asp, με βάση της δυνατοτητές της σαν γλώσσα προγραμματισμού.

2.1.1 Χαρακτηριστικά της Rails:

1. Μην επαναλαμβάνεσαι(Don't Repeat Yourself, DRY): Για να διατηρηθεί ο κώδικας "καθαρός", η rails ακολουθεί την ιδέα του DRY. Η ιδέα που βρίσκεται από πίσω είναι απλή. Είναι προτιμότερη η επαναχρησιμοποίηση του κώδικα που ήδη υπάρχει, και όχι η άσκοπη επανάληψη παρόμοιου κώδικα σε πολλαπλές θέσεις. Αυτό μειώνει τα λάθη, οι αλλαγές στον κώδικα γίνονται πιο εύκολα.

    - Η αρχή αυτή γενικεύεται και στα εξής:
    - Στα σχήματα της βάσης δεδομένων (database schema)
    - Στα πλάνα δοκιμών (test plans)
    - Στην τεκμηρίωση (documentation)

    Χρησιμοποιώντας την τεχνική αυτή επιτυχώς, σημαίνει ότι η αλλαγή ενός συγκεκριμένου στοιχείου δεν επηρεάζει τα υπόλοιπα, λογικά ασύνδετα στοιχεία του συστήματος. Τέλος, στοιχεία που συνδέονται λογικά, αλλάζουν ομοιογενώς είναι απολύτως προβλέψιμα και έτσι είναι συγχρονισμένα.

2. Η rails χρησιμοποιεί την αρχιτεκτονική model-view-controller (MVC), για να οργανώσει τον προγραμματισμό των εφαρμογών της.

- Model(μοντέλο): Είναι υπεύθυνο για την επικοινωνία με τη βάση της εφαρμογής. Αν και λέμε όλο το layer model, οι rails εφαρμογές περιέχουν συνήθως πολλά ανεξάρτητα models, το καθένα από τα οποία "δείχνει" σε ένα πίνακα στη βάση δεδομένων.
- View(όψεις): Αφορά το επίπεδο παρουσίασης. Πως θα φαίνονται τα αποτελέσματα / δεδομένα. Είναι ουσιαστικά templates που, τις περισσότερες φορές, περιέχουν κώδικα HTML. Γενικά, τα views, έχουν την ευθύνη για τη

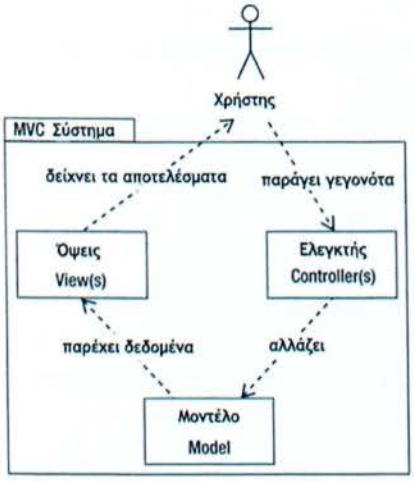
μορφοποίηση και για το πως θα φαίνονται στην οθόνη του υπολογιστη, τα δεδομένα του μοντέλου.Το view είναι το πιο σημαντικό σημείο, διότι είναι αυτό που βλέπει ο τελικός χρήστης στην οθόνη του. Η MVC λογική βοηθάει με το να αφήνει την προγραμματιστική λογική έξω από το view.Με αυτό το σκεπτικό οι προγραμματιστές ασχολούνται με τον κώδικα, και οι σχεδιαστές με την html.

- Controller(ελεγκτής): Είναι υπεύθυνο για την διεπαφή με το χρήστη και γενικά με τη λειτουργικότητα της εφαρμογής. Δέχεται τα αιτήματα από τον "έξω" κόσμο, κάνει τις απαραίτητες διεργασίες και μετά δίνει τον έλεγχο στο view επίπεδο για να εμφανίσει τα αποτελέσματα. Είναι δουλειά του controller να επεξεργαστεί τα web requests, όπως να επεξεργαστεί μεταβλητές του server και δεδομένα από φόρμες επικοινωνίας, να ζητά από το μοντέλο πληροφορίες, και να στέλνει πίσω τα αποτελέσματα για να αποθηκευτούν από το μοντέλο. Ο controller ουσιαστικά εκτελεί ενέργειες από τον χρήστη για να δημιουργήσει(create), να διαβάσει(read), να ανανεώσει(update) και να διαγράψει(destroy) ενα αντικείμενο του μοντέλου.Αυτές οι ενέργειες περιγράφονται στη rails σαν CRUD. Για να απαντήσει σε ένα αίτημα, ο controller τυπικά εκτελεί το CRUD στο μοντέλο, θέτει μεταβλητές για να χρησιμοποιηθούν από το view, και στη συνέχεια ανακατευθύνει σε μια άλλη ενέργεια αφού ολοκληρωθεί η διαδικασία.

Ο κύκλος του MVC:

Αν και το MVC έχει διάφορες μορφές, η γενική λογική είναι η εξής(εικόνα 1):

- Ο χρήστης αλληλεπιδρά με τον γραφικό περιβάλλον και ενεργοποιεί ένα event (για παράδειγμα,αποστέλει στοιχεία μέσα από μια φόρμα επικοινωνίας)
- Ο controller δέχεται τα δεδομένα από το γραφικό περιβάλλον(για παράδειγμα, τα στοιχεία από τη φόρμα επικοινωνίας)
- Ο controller αποκτά πρόσβαση στο model, και πολύ συχνά το ανανεώνει με κάποιο τρόπο.(για παράδειγμα, δημιουργώντας ένα νέο χρήστη)
- Ο controller επικαλείται το view, που κατευθύνει ένα ανανεωμένο γραφικό περιβάλλον(για παράδειγμα, μια welcome σελίδα)
- Το περιβάλλον περιμένει για περαιτέρω αλληλεπίδραση με τον χρήστη, και ο κύκλος επαναλαμβάνεται.



Εικόνα 1

3. Σύμβαση έναντι Ρύθμισης (Convention over Configuration, CoC): Ο προγραμματισμός έχει να κάνει με αποφάσεις.Αν πρόκειτε να γράψει ο προγραμματιστής μια εφαρμογή από την αρχή, χωρίς την μορφοποίηση της rails,πρέπει να παρθούν πολλές αποφάσεις.Πως θα οργανωθούν τα αρχεία, πως θα ονομαστούν,πως θα χειριστεί την βάση δεδομένων κτλ. Με τη rails ξεκινάς να γράφεις κώδικα, μιας και όλα τα παραπάνω τα παρέχει έτοιμα. Αυτό έχει σαν αποτέλεσμα να αφοσιώνεται ο προγραμματιστής στην ανάπτυξη της εφαρμογής και να την ξεκινάει σχεδόν αμέσως. Τα ονόματα των κλάσεων και των μεθόδων ονοματίζονται με βάση του Convention over Configuration. Αν ένας πίνακας ονομάζετε "entries" (πληθυντικός και με πεζή γραφή), τότε ονομάζετε μια κλάση "Entry" (ενικός και αρχικό γράμμα κεφαλαίο). Υπάρχει παρόλα αυτά η δυνατότητα, υπεκφυγής από την εξ ορισμού συμπεριφορά, π.χ. όταν γίνετε χρήση μιας ήδη υπαρκτής βάσης δεδομένων ή όταν δεν μπορεί ή δεν επιτρέπετε να γίνει η αλλαγή ονόματος ενός πίνακα. Ο προγραμματιστής χρειάζεται να ορίσει μόνο τις παραμέτρους που δεν είναι δυνατόν να «προτυποποιηθούν». Έτσι, οι «συμβάσεις» που χρησιμοποιούνται από το Rails μπορούν να μειώσουν αισθητά τον κώδικα που χρειάζεται να γραφεί.

4. Agile Manifesto:
    - Η ευχαρίστηση του πελάτη με άμεση και συνεχή παράδοση χρήσιμου λογισμικού.
    - Το λειτουργικό λογισμικό παραδίδεται συχνά (εβδομάδες αντί για μήνες).
    - Το λειτουργικό λογισμικό είναι το πρωταρχικό κριτήριο για την μέτρηση της προόδου.
    - Ακόμα και αλλαγές που ανακύπτουν αργότερα είναι ευπρόσδεκτες.
    - Στενή, καθημερινή συνεργασία μεταξύ επιχειρηματιών και προγραμματιστών.
    - Συζήτηση πρόσωπο-με-πρόσωπο είναι η καλύτερη μορφή επικοινωνίας.
    - Συνεχής προσοχή στην τεχνική αρτιότητα και τον καλό σχεδιασμό.
    - Απλότητα.
    - Αυτό-διαχειριζόμενες ομάδες.
    - Συχνή προσαρμογή στις αλλαγές περιστάσεων.

2.1.2 Οι βιβλιοθήκες της Rails

Η rails είναι ένα σύνολο βιβλιοθηκών, όπου η κάθε μια ασχολείται με ένα συγκεκριμένο έργο. Μαζεύοντας τες μαζί, αυτές οι βιβλιοθήκες δημιουργούν το rails framework. Από όλες τις βιβλιοθήκες 3 είναι αυτές που αντικατοπτρίζουν απευθείας το MVC pattern:

- Active Record: Μια βιβλιοθήκη που χειρίζεται την αλληλεπίδραση με τη βάση.
- Action View: Ένα σύστημα με templates που παράγει τα HTML έντυπα που ο χρήστης "παίρνει" πίσω σαν αποτέλεσμα από ένα αίτημα σε μια rails εφαρμογή.
- Action Controller: Είναι μια βιβλιοθήκη που χειρίζεται και τη ροή της εφαρμογής και τα δεδομένα που έρχονται από τη βάση, για να εμφανιστούν στο view.

Κεφάλαιο 3: Η βάση δεδομένων – MongoDB

Η mongodb είναι ένα open-source document-oriented database system που αναπτύχτηκε από την εταιρία 10gen. Είναι κομμάτι των No-sql βάσεων δεδομένων. Αντί να αποθηκεύει τα δεδομένα σε πίνακες όπως κάνουν οι "κλασσικές" σχεσιακές βάσεις δεδομένων, η mongodb αποθηκεύει τα δεδομένα σαν Json-like δεδομένα με δυναμικά schemas και κάνει την προσπέλαση και επεξεργασία των συγκεκριμένων τύπων εφαρμογών εύκολη και γρήγορη.

Η 10gen ξεκίνησε την ανάπτυξη της mongodb τον Οκτώβριο του 2007, όταν η εταιρία εφτιαχνε μια πλάτφορα σαν υπηρεσία όπως το Google app engine. Το 2009 η mongodb έγινε open-source σαν stand-alone προιόν με την άδεια AGPL. Το Μάρτιο του 2010, από την έκδοση 1.4, είναι έτοιμη για την παραγωγή. Η τελευταία σταθερή έκδοση βγήκε τον Μάρτιο του 2013.Την βάση αυτή τη χρησιμοποιουν μεγάλες εταιρίες όπως το Foursquare.

Η mongodb προσφέρει μεγάλη απόδοση, μεγάλη διαθεσιμότητα, και τέλος εύκολη επεκτασιμότητα.

Έγγραφο της βάσης δεδομένων (Document Database)

- Τα έγγραφα (αντικείμενα) καθορίζονται εύκολα στους τύπους των δεδομένων στις γλώσσες προγραμματισμού.
- Τα ενσωματωμένα έγγραφα και οι πίνακες μειώνουν την ανάγκη για ενώσεις (joins)
- Το δυναμικό σχήμα καθιστά ευκολότερο τον πολυμορφισμό.
- Υποστηρίζει αναζήτηση με βάση το πεδίο, range queries και regular expression αναζητήσεις. Τα queries μπορούν να επιστρέψουν συγκεκριμενα πεδία από τα δεδομένα καθώς επίσης περιλαμβάνουν user-defined javascript συναρτήσεις.

Υψηλή Απόδοση

- Τα ενσωμάτωμένα έγγραφα διαβάζονται και γράφονται γρηγορότερα.
- Οποιοδήποτε πεδίο σε ένα mongodb αρχείο μπορεί να αρχειοθετηθεί.
- Οι δείκτες μπορούν να περιλάμβάνουν τα κλειδιά από ενσωματωμένα έγγραφα και πίνακες.
- File storage: η mongodb μπορεί να χρησιμοποιηθεί και σαν file system, αξιοποιώντας την εξισορρόπηση του φορτίου και τα χαρακτηριστικά αντιγραφής δεδομένων σε πολλαπλά μηχανήματα σαν storing αρχεια.

- Aggregation: Το MapReduce μπορεί να χρησιμοποιηθεί ως σύνολο επεξεργασίας δεδομένων και ομαδοποίηση εργασίων.
- Server-side Javascript execution: Η javascript μπορεί να χρησιμοποιηθεί σε queries στην ομαδοποίηση δεδομένων, στέλνοντάς τα απευθείας στη βάση για να εκτελεστούν.

Εύκολη επεκτασιμότητα

- Αυτόματο sharding που διανέμει τη συλλογή δεδομένων σε διαφορετικές μηχανές.Ο προγραμματιστής διαλέγει το shard key, που υποστηρίζει πως τα δεδομένα σε μια collection(συλλογή) θα είναι κατανεμημένα. Τα δεδομένα είναι χωρισμένα σε ranges (βασισμένα στο shard key) και κατανεμημένα σε πολλαπλά shard.(Ένα shard είναι ενας master με ένα ή περισσότερα slaves). Η mongodb μπορεί να τρέξει σε πολλαπλούς servers, εξισορροπώντας το φορτίο ή/και αντιγράφοντας τα δεδομένα για να κρατήσει το σύστημα ενεργό και να τρέχει ακόμα και σε περιπτώσεις που το hardware αποτύχει.Αυτόματες διαμορφώσεις είναι πιο εύκολες για την ανάπτυξη και νεά μηχανήματα μπορουν να προστεθούν στην ήδη υπάρχουσα βάση δεδομένων που "τρέχει".
- Replication: Η mongodb υποστηρίζει master-slave replication. Ο master μπορεί να διαβάζει και να γράφει. Ο slave αντιγράφει τα δεδομένα από τον master και μπορεί μόνο να διάβαζει και να κρατάει backup, αλλά όχι να γράψει. Τα slaves έχουν την δυνατότητα να διαλέξουν νέο master αν ο συγκεκριμένος δεν είναι διαθέσιμος.

3.1 Διαχείριση Δεδομένων: συλλογές(collections) και αρχεία(documents)

Η mongodb αποθηκεύει τα δεδομένα σαν json αντικείμενα, χρησιμοποιώντας δυναμικά schemas, που λέγονται bson. Στην mongodb το στοιχείο δεδομένων λέγεται αρχείο(documents) και αποθηκεύεται σε συλλογές(collections).Μια συλλογή μπορεί να έχει πολλά αρχεία.

Η διευθέτηση των δεδομένων στη mongodb είναι καινοτόμος σε σχέση με τις κλασσικές σχεσιακές βάσεις δεδομένων(RDBMS, 'relational database management system'). Ένα από τα βασικά χαρακτηριστικά μιας RDBMS βάσης, είναι οτι σε κάθε πίνακα, κάθε εγγραφή έχει τα ίδια πεδία με την άλλη και καταχωρούνται με τη ίδια σειρά.

Last Name	First Name	Date of Birth
DUMONT	Jean	01-22-1963
PELLERIN	Franck	09-19-1983
GANNON	Dustin	11-12-1982

Στη mongodb μπορούμε να πούμε οτι οι συλλογές(collections) είναι σαν πίνακες και τα αρχεια(documents) σαν εγγραφές(records).Υπάρχει όμως μια μέγαλη διαφορά: κάθε αρχείο σε μια collection μπορεί να έχει εντελώς διαφορετικά πεδία από τα άλλα documents. Το μόνο schema που απαιτείται στη mongodb είναι ένα '_id' πεδίο με μονάδικη τιμή(non-array).

```
{
   "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
   "Last Name": "DUMONT",
   "First Name": "Jean",
   "Date of Birth": "01-22-1963"
},
{
   "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
   "Last Name": "PELLERIN",
   "First Name": "Franck",
   "Date of Birth": "09-19-1983",
   "Address": "1 chemin des Loges",
   "City": "VERSAILLES"
}
```

Σε ένα document μπορούν να προστεθούν νεα πεδία να μετανομαστούν ή να αλλάξουν τα ήδη υπάρχοντα οποιαδήποτε στιγμή. Δεν υπάρχει προκαθορισμένο schema. Η δομή του document είναι απλή: ακολουθεί το json format και αποτελείται από μια σειρα key-value pairs, ετσι ώστε το document να είναι ισοδύναμο σε δυνατότητες και ονομάζεται σε διάφορες γλώσσες προγραμματισμού associative arrays. Το κλειδί μιας key-value pair είναι το όνομα του πεδίου, η τιμή της key-value pair είναι το περιεχόμενο ενός πεδίου. Το κλειδί και η τιμή χωρίζονται με ":".Η τιμή μπορεί να είναι αριθμός, string, boolean, binary data όπως εικόνα, πινακας με τιμές – που η κάθε τιμή είναι διαφορετικός τύπος- ή ενα υποδεεστερο document. Για παράδειγμα:

```
{
   "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
   "Last Name": "PELLERIN",
   "First Name": "Franck",
   "Date of Birth": "09-19-1983",
   "phoneNumber": [
      {
         "type": "home",
         "number": "212 555-1234"
      },
      {
         "type": "fax",
         "number": "646 555-4567",
         "verified": false
      }
   ],
   "Address": {
      "Street": "1 chemin des Loges",
      "City": "VERSAILLES"
   },
   "Months at Present Address": 37
}
```

3.2 Εγκατάσταση της mongodb.

Για να εγκαταστήσουμε τη mongo στον υπολογιστή μας αρχικά θα πρέπει να κατεβάσουμε από την σελίδα της http://www.mongodb.org/downloads κάποια αρχεία.

Στο terminal του ubuntu πληκτρολογούμε

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
```

Δημιουργούμε ένα **/etc/apt/sources.list.d/10gen.list αρχείο** και προσθέτουμε την παρακάτω γραμμή για το 10gen repository.

deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen

Τώρα τρέχουμε την εντολή:

```
sudo apt-get update
```

Για να εγκαταστήσουμε την τελευταία σταθερή έκδοση της mongodb γράφουμε στο terminal.

```
sudo apt-get install mongodb-10gen
```

Όταν ολοκληρωθεί η διαδικασία έχει εγκατασταθεί με επιτυχία η mongo στον υπολογιστή μας.

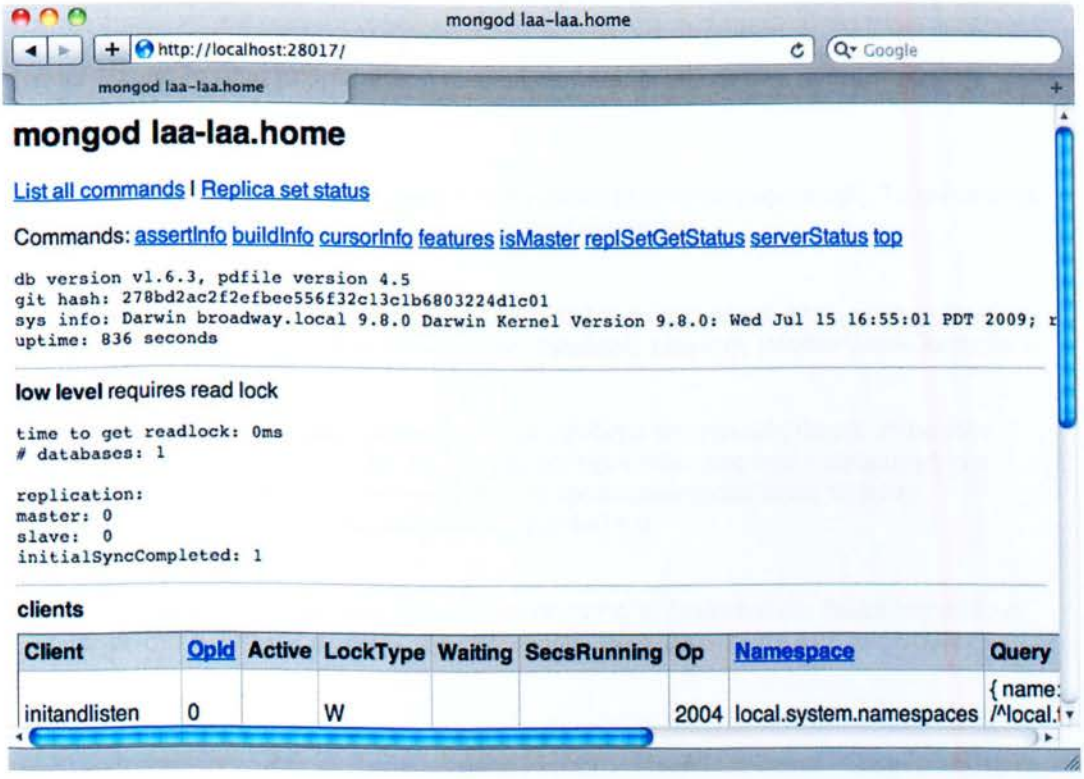
Για να ξεκινήσει τη λειτουργία της πληκτρολογούμε στο terminal:

```
sudo service mongodb start
```

Αντίστοιχα για να σταματήσουμε τη λειτουργία της και να την επανακινησουμε γράφουμε:

```
sudo service mongodb stop
sudo service mongodb restart
```

Για να δούμε αν δουλεύει πληκτρολούμε στον browser μας: http://localhost:28017



Κεφάλαιο 4: Δομή μιας ruby on rails εφαρμογής

Κάθε εφαρμογή Ruby on Rails από τη δημιουργία της είναι οργανωμένη σε φακέλους. Οι εικόνες, το σχήμα της βάσης δεδομένων, ο κώδικας HTML, ο κώδικας CSS και κάθε άλλο κομμάτι της εφαρμογής, έχουν τη δική τους θέση. Το πλεονέκτημα αυτό συμβάλει στην εύκολη συντήρηση και επέκταση του κώδικα. Στην παρακάτω εικόνα φαίνεται η δομή των φακέλων και ακολουθεί αναλυτικότερη παρουσίαση για κάθε φάκελο.

app/ Αυτός είναι ο φάκελος που χρησιμοποιείται περισσότερο από τον προγραμματιστή. Περιέχει τον κυρίως κώδικα της εφαρμογής (μοντέλο-προβολή-ελεγκτής) και θα αναλυθεί εκτενέστερα στην επόμενη παράγραφό με τίτλο "Μοντέλο-Προβολή-Ελεγκτής".

config/ Η Ruby on Rails, όπως αναφέρθηκε προηγούμενως, χρησιμοποιεί την αρχή της σύμβασης αντί παραμετροποίησης. Οι επιπλέον ρυθμίσεις που απαιτούνται για την κάλυψη των αναγκών της εφαρμογής ή οι ρυθμίσεις που παρακάμπτουν τις συμβάσεις αποθηκεύονται στον φάκελο config.

config.ru Αρχείο για την παραμετροποίηση της διεπαφής με τον Rack εξυπηρετητή.

db/ Σχήμα βάσης δεδομένων και πληροφορίες για τα migrations.

doc/ Σ' αυτό το φάκελο βρίσκεται η τεκμηρίωση που δημιουργείται αυτόματα με την εντολή doc.

Gemfile Εδώ καθορίζονται τα gems από τα οποία εξαρτάται η εφαρμογή. Τα gems είναι πακέτα που περιέχουν προγράμματα ruby και βιβλιοθήκες.

lib/ Ο φάκελος αυτός φιλοξενεί κώδικα που είτε δεν ανήκει σε κανένα ή χρησιμοποιείται από περισσότερα από ένα εκ των μοντέλο, προβολή, ελεγκτής (Model-View-Controller).

log/ Όσο "τρέχει" μια Rails εφαρμογή δημιουργούνται καταγραφές (logs). Υπάρχουν τρεις φάκελοι για την ανάπτυξη, τις δοκιμές και την κατάσταση που η εφαρμογή έχει δημοσιευτεί. Καταγράφονται στοιχεία που αφορούν ερωτήματα προς τη βάση δεδομένων, δεδομένα της κρυφής μνήμης (cache) κ.α.

public/ Ο φάκελος αυτός είναι προσπελάσιμος από το διαδίκτυο και θεωρείται από τον εξυπηρετητή ο βασικός φάκελος της εφαρμογής. Εδώ, βρίσκονται στατικές σελίδες.

Rakefile Εδώ ορίζονται εργασίες που μπορεί να εκτελούν δοκιμές (tests), να δημιουργούν τεκμηρίωση, να εκτυπώσουν το σχήμα της βάσης κ.α.

README Οδηγίες εγκατάστασης και χρήσης.

script/ Εδώ βρίσκονται τα σενάρια (scripts) της Rails, τα οποία "τρέχουν" αν στη γραμμή εντολών γράψουμε την εντολή rails και το όνομα της συνάρτηση που καλούμε, δηλαδή *rails console*, *rails generate*, *rails new* κ.λ.π. Επίσης, σ' αυτό το φάκελο αποθηκεύονται και τα σενάρια που γράφει ο προγραμματιστής.

test/ Η Rails προσφέρει ευρύ φάσμα εργαλείων για την δημιουργία και την εκτέλεση δοκιμών (tests). Σ' αυτό το φάκελο βρίσκεται ό,τι σχετίζεται με δοκιμές, όπως δοκιμές ενσωμάτωσης (integration tests), δοκιμές λειτουργικότητας (functional tests), δοκιμές επανάληψης (iteration tests).

tmp/ Ένας φάκελος για τα προσωρινά αρχεία, όπως τα περιεχόμενα της κρυφής μνήμης.

vendor/ Όλες σχεδόν οι εφαρμογές χρησιμοποιούν έτοιμο κώδικα ο οποίος προσθέτει λειτουργικότητα. Ο πιο διαδεδομένος τρόπος στην Rails είναι τα gems. Παρόλ' αυτά αν κρίνεται σκόπιμο ο κώδικας αυτός μπορεί να αποθηκευτεί στο φάκελο vendor/plugin.

Κεφάλαιο 5: Η Διαδικτυακή εφαρμογή

Η εφαρμογή που δημιουργήθηκε με τη χρήση της Ruby on Rails είναι ένας Rss Reader. Ενώ υπάρχουν στο διαδίκτυο πολλές παρόμοιες εφαρμογές, η συγκεκριμένη δίνει τη δυνατότητα στον χρήστη να μπορεί να προσθέτει τους δικούς του Rss υπερσυνδέσμους.

Η λειτουργία πίσω από την εφαρμογή είναι πολύ απλή. Ο χρήστης μόλις μπει στην ιστοσελίδα, δημιουργεί ένα λογαριασμό βάζοντας ένα όνομα χρήστη, το email του, και έναν κωδικό πρόσβασης. Με την εγγραφή του στην ιστοσελίδα, θα του αποσταλεί ενα email που θα περιέχει έναν σύνδεσμο για την ενεργοποίηση του λογαριασμού του, που ο χρήστης θα πρέπει να πατήσει για να γίνει δεκτή η εγγραφή του.

Στη συνέχεια, από το μενού μπορεί να διαλέξει την επιλογή Channels, όπου θα τον πάει σε μια σελίδα που θα μπορεί να δει όλες τις κατηγορίες των Rss υπερσυνδέσμων που έχει προσθέσει. Την πρώτη φορά που θα την επισκεφτεί θα είναι άδεια, περιμένοντας να βάλει τα δικά του Rss νέα.

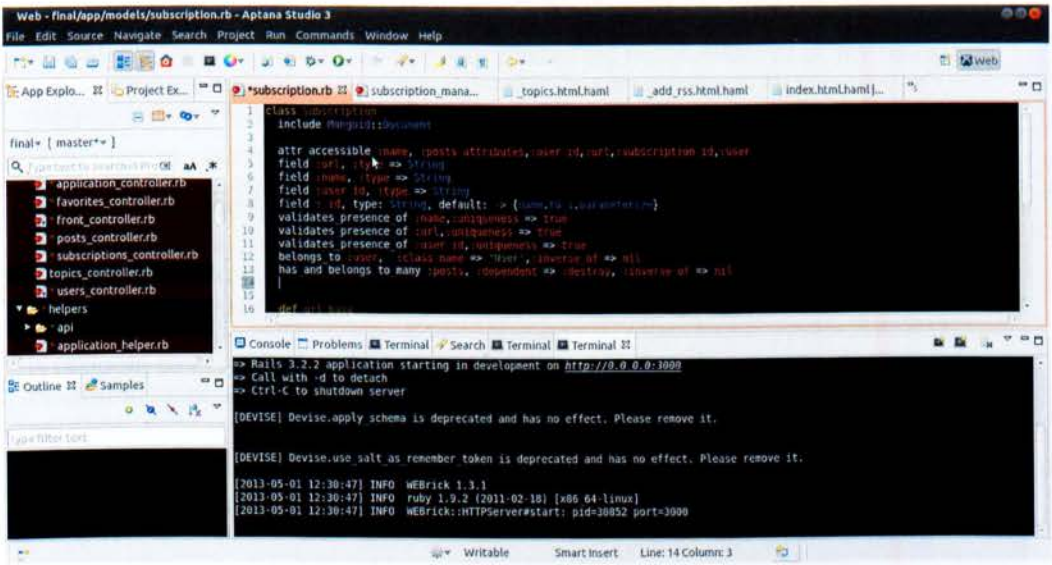
Για να προσθέσει ένα νέο Rss Feed, ο χρήστης θα πρέπει να επικολλήσει τον υπερσύνδεσμο από την ιστοσελίδα που επιθυμεί, στην φόρμα που υπάρχει αριστερά από την σελίδα. Αφού τον επικολλήσει, πατάει το κουμπί Add, και εκείνη τη στιγμή γίνεται η επεξεργασία του υπερσυνδέσμου. Αν είναι λάθος, βγαίνει ένα μήνυμα λάθους που ειδοποιεί τον χρήστη να κοιτάξει αν ο υπερσύνδεσμος είναι σωστά γραμμένος. Στην περίπτωση που όλα πάνε καλά και δεν υπάρχουν λάθη, καλείται ο δαίμονας που κάνει την επεξεργασία, τραβάει τα δεδομένα από το site με βάση τη μορφή που έχει η Rss σελίδα, και τα προσθέτει στη βάση δεδομένων, δημιουργεί την κατηγορία όπου θα εμφανίζονται τα νέα, και βγάζει μήνυμα επιτυχίας στον χρήστη. Αν ο χρήστης πάτησει και πάλι από το μενού το κουμπί Channels και πάει στη σελίδα, θα δει τον τίτλο από το Rss νέο που έχει προσθέσει.

Από τη στιγμή που δημιουργηθεί μια ετικέτα, πέρα από τη HTML μορφή που ο χρήστης βλέπει στον φυλλομετρητή του, δημιουργείται και η αντίστοιχη json μορφή, όπου ο χρήστης μπορεί να κανεί από την εφαρμογή του κινητού android, αφού την κάνει εγκατάσταση αρχικά.

Κεφάλαιο 6: Εργαλεία και Μέθοδοι

6.1 Περιβάλλον ανάπτυξης – Εργαλεία

6.1.1 Aptana



Η εφαρμογή αναπτύχθηκε στο περιβάλλον ανάπτυξης Aptana. Το Aptana αποτελεί μια ειδική έκδοση του Eclipse και είναι προσαρμοσμένο στις ανάγκες εφαρμογών διαδικτύου με την Ruby on Rails. Υποστηρίζει αυτόματη συμπλήρωση κώδικα (intellisense) για πολλές διαδεδομένες γλώσσες προγραμματισμού. Μερικές από αυτές είναι οι Javascript, CSS, Ruby/Rails, PHP, HTML, SASS. Ένα από τα δυνατά του σημεία είναι η υποστήριξη του git στην προβολή του έργου, η χρήση των bundles (αυτοματοποίηση κώδικα) προσφέροντας γρήγορο deploy σε διάφορες πλατφόρμες όπως το heroku και η Engine Yard ή deploy με τη χρήση του capistrano.

6.1.2 Phusion Passenger

Ο Phusion Passenger είναι ένας εξυπηρετητής ειδικά σχεδιασμένος για να υποστηρίζει Ruby on Rails εφαρμογές. Συνεργάζεται με τον apache και τον nginx και προσφέρει

γρήγορο και χωρίς προβλήματα deploy, με έμφαση στην απόδοση και στην ταχύτητα εξυπηρέτησης των αιτήσεων.

6.1.3 RVM

Το RVM (Ruby Version Manager) δημιουργήθηκε από τον Seguin Waynee με σκοπό την ταυτόχρονη εγκατάσταση πολλών εκδόσεων της γλώσσας Ruby. Επίσης υποστηρίζει την καλύτερη οργάνωση των βιβλιοθηκών σε gemsets. Κάθε gemset αποτελείται από ένα σύνολο βιβλιοθηκών (gems) της Ruby. Ανάλογα με την εφαρμογή χρησιμοποιείται η κατάλληλη έκδοση βιβλιοθηκών και οργανώνονται καλύτερα οι αλληλεξαρτήσεις των βιβλιοθηκών.

Μπορούμε να εγκαταστήσουμε το rvm με την εντολή:

```
curl -L get.rvm.io | bash -s stable –auto
```

Η εγκατάσταση του RVM και της Ruby γίνεται στο home directory μας. Στη συνέχεια πρέπει να φορτώσουμε το ~/.bash_profile αρχείο με την εντολή

```
~/.bash_profile
```

Η επόμενη εντολή που θα τρέξουμε θα μας πει τι άλλα πακέτα χρειάζεται να εγκαταστήσουμε για να κάνουμε τη Ruby να δουλέψει:

```
rvm requirements

...

# For Ruby / Ruby HEAD (MRI, Rubinius, & REE), install the following

ruby: /usr/bin/apt-get install build-essential openssl libreadline6 libreadline6-dev curl git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt-dev autoconf libc6-dev ncurses-dev automake libtool bison subversion pkg-config
```

Στη συνέχεια για να τα εγκαταστήσουμε γράφουμε στο terminal του ubuntu:

```
sudo apt-get install build-essential openssl libreadline6 libreadline6-dev curl git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt-dev autoconf libc6-dev ncurses-dev automake libtool bison subversion pkg-config
```

Με το RVM και τα παραπάνω πακέτα μπορούμε να εγκαταστήσουμε τη ruby.

```
rvm install 1.9.3
```

Με το τέλος της εκτέλεσης της εντολής, η Ruby 1.9.3 έχει εγκατασταθεί με επιτυχία.Για να ξεκινήσουμε να τη χρησιμοποιούμε γράφουμε στο terminal:

```
rvm use 1.9.3
```

Για να δουμε οτι όντως έχει εγκατασταθεί η συγκεκριμένη έκδοση της ruby γράφουμε στο terminal:

```
ruby -v
ruby 1.9.3p327 (2012-04-20 revision 35410) [x86_64-linux]
```

Για να ορίσουμε τη default έκδοση γράφουμε:

```
rvm --default use 1.9.3-p327
```

Τώρα για να εγκαταστήσουμε τη rails γράφουμε:

```
gem install rails -v 3.2.9
```

Η εντολή αυτή θα εγκαταστήσει αρκετά gem συμπεριλαμβανομένου και του bundler.

6.1.4 Bundler

Ο Bundler αναπτύχθηκε από έναν από τους πρωτοπόρους της βασικής ομάδας της Rails και της jQuery, τον Yehuda Katz. Ελέγχει τις αλληλεξαρτήσεις μεταξύ των βιβλιοθηκών της Ruby και φροντίζει ώστε να γίνεται με αυτοματοποιημένο τρόπο η μεταφόρτωση των βιβλιοθηκών μιας εφαρμογής σε διαφορετικά συστήματα.

6.1.5 Devise

Το devise είναι μια βιβλιοθήκη για την ταυτοποίηση των χρηστών. Αναλαμβάνει ένα πλήθος λειτουργιών, είναι πλήρως παραμετροποιήσιμο και μπορεί να προστεθεί σε οποιαδήποτε Rails εφαρμογή. Οι λειτουργίες που υποστηρίζει είναι οι εξής:

- Εγγραφή / Ταυτοποίηση χρηστών
- Δυνατότητα εγγραφής με ταυτότητα Omniauth (υπηρεσία που προσφέρει κοινή ταυτότητα για χρήση σε πολλές ιστοσελίδες, όπως το facebook και το twitter).
- Επιβεβαίωση στοιχείων λογαριασμού χρηστών
- Υπενθύμιση και επαναφορά κωδικού χρήστη
- Συλλογή στοιχείων όπως τελευταίας σύνδεσης και πλήθος συνδέσεων
- Δυνατότητα αποσύνδεσης λογαριασμού μετά από καθορισμένο χρόνο
- Έλεγχος εγκυρότητας της μορφής της διεύθυνσης ηλεκτρονικού ταχυδρομείου και του κωδικού πρόσβασης
- Κλείδωμα λογαριασμού με συγκεκριμένα κριτήρια, σε περίπτωση υποκλοπής του μετά από πολλές αποτυχημένες συνδέσεις

6.1.6 Mongoid

Το mongoid είναι μια αντικειμενο-σχεσιακή χαρτογράφηση (Object-Document-Mapper - ODM) για τη βάση δεδομένων mongodb που είναι γραμμένο σε ruby. Η ODM είναι μια τεχνική προγραμματισμού για τη μετατροπή δεδομένων μεταξύ ασύμβατων συστημάτων τύπου σε αντικειμενοστραφή γλώσσα προγραμματισμού. Στην πραγματικότητα δημιουργεί μια εικονική βάση δεδομένων αντικειμένων(virtual object database) που μπορεί να χρησιμοποιηθεί από το εσωτερικό της γλώσσας προγραμματισμού. Με αυτόν το τρόπο είναι πολύ εύκολη η δήλωση και οι συνεχείς ανανέωση των αντικειμένων στη βάση. Μια απλή δομή μιας καταχώρησης στη βάση με το mongoid είναι η παρακάτω:

```
class Person
  include Mongoid::Document

  field :first_name, type: String
  field :middle_name, type: String
  field :last_name, type: String
end
```

6.1.7 Haml

Είναι μια πιο μικρή markup γλώσσα που χρησιμοποείται για να περιγράψει XHTML για οποιοδήποτε web αρχείο. Έχει σχεδιαστεί για να αντιμετωπίσει τις ελλείψεις σε παραδοσιακές μηχανές templating(templating engines), καθώς επίσης και για την πιο όμορφη σήμανση του κώδικα.

6.1.8 Sass-rails

Είναι μια CSS metalanguage. Ουσιαστικά είναι μια scripting γλώσσα που μεταφράζεται σε Css. Η sass παρέχει μια σειρά από χαρακτηριστικά για να κάνει τη διαχείριση των φύλλων στυλ (stylesheet) μιας εφαρμογής ευκολότερη συμπεριλαμβανομενου του εμφωλιασμού(nesting), των μεταβλητών και πολλά άλλα χαρακτηριστικά που δεν έρχονται σαν βασικά στοιχεία του CSS. Ένα παράδειγμα του Sass είναι το παρακάτω:

CSS	SASS
#header { background-color: #03507B; color: #FFF; padding: 4px 100px;	#header { background-color: #03507B; color: #FFF; padding: 4px 100px;

```
  border-bottom: solid 5px #00395A;
}

#header h1 {
  font-size: 30px;
}
```

```
border-bottom: solid 5px #00395A;

    h1 {
        font-size: 30px;
    }
}
```

6.1.9 delayed_job

Το delayed_job είναι μια βιβλιοθήκη που χειρίζεται εργασίες στο παρασκήνιο. Το gem προέρχεται από το Shopify όπου ο πίνακας εργασίων είναι υπεύθυνος για ένα πλήθος βασικών εργασίων. Κάποιες από τις βασικές εργασίες είναι:

- Μαζική αποστολή ενημερωτικών δελτίων (newsletters)
- Αλλαγή μεγέθους εικόνας
- Http downloads

6.1.10 will_paginate

Είναι μια βιβλιοθήκη που προσφέρει ένα απλό api για την σελιδοποίηση των δεδομένων

6.1.11 feedzirra

Το feedzirra είναι μια βιβλιοθήκη που έχει σχεδιαστεί για να παίρνει και να ανανεώνει πολλά rss feeds όσο πιο γρήγορα γίνεται. Μόλις το feedzirra πάρει τα feed,αυτά μπορούν να ανανεωθούν με τη χρήση των feed αντικειμένων. Το feedzirra αυτόματα εισάγει etag και πότε η πληροφορία έχει ανανεωθεί από τα http headers για μικρότερη χρήση του bandwidth, και εξουδετερώνει άχρηστα parsing και γενικά κάνει τη διαδικασία πιο γρήγορη.

Ένα άλλο χαρακτηριστικό που παρουσιάζεται στο feedzirra είναι η δυνατότητα που έχει για να δημιουργεί αντικειμενοστραφείς συναρτήσεις οι οποίες καλούνται όταν επιτυχημένα ή αποτυχημένα "έρθει" το feed. Αυτό κάνει πιο εύκολα τα πράγματα, ειδικά για τα log errors ή για την ανανέωση των δεδομένων.

Η λογική για να φέρει και να αναλύσει τα δεδομένα είναι η αποσύνθεσή τους ως σύνολο, έτσι ώστε να μπορεί το κάθε στοιχείο να χρησιμοποιηθεί μόνο του, σε περίπτωση που κάποιος αποφασίσει να μη χρησιμοποιήσει όλα τα στοιχεία που προσφέρει το gem.

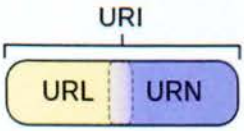
Το τελευταία στοιχείο του feedzirra είναι η δυνατότητα που δίνει σχετικά με τον προσαρμοσμένο τρόπο προσπέλασης των δεδομένων. Στην πραγματικότητα, το feedzirra μπορεί να χρησιμοποιηθεί για να προσπελάσει μέγαλο αριθμό δεδομένων.

URI (Uniform Resource Identifier)

Στον κλάδο της πληροφορικής, ένα ενιαίο αναγνωριστικό πόρου (URI) είναι μια σειρά από χαρακτήρες που χρησιμοποιούνται για τον εντοπισμό ενός ονόματος(name) ή ένα web resource. Αυτή την ταυτοποίηση επιτρέπει την αλληλεπίδραση με αναπαραστάσεις του διαδικτυακού πόρου μέσω δικτύου (συνήθως το world wide web) με τη χρήση ειδικών πρωτοκόλλων. Τα σχέδια (schemes) καθορίζουν μια συγκρεκριμένη σύνταξη και παρόμοια πρωτόκολλα ορίζουν κάθε URI.

Η σχέση μεταξύ τη διεύθυνση URI και URN

Στην εικόνα 2 φαίνεται ότι ένα ενιαίο αναγνωριστικό πόρου (URI) είναι είτε ένας εναιός εντοπιστής πόρου (URL: uniform resource locator), ή ένα ενιαίο όνομα πόρου (URN: Uniform resource name) ή και τα δύο.



Εικόνα 2

Το URN λειτουργεί όπως για παράδειγμα το όνομα ενός ατόμου, ενώ παράλληλα το URL μοιάζει με τη διεύθυνση αυτού του ατόμου. Με άλλα λόγια, το URL προσδιορίζει την ταυτότητα ενός αντικειμένου, ενώ η διεύθυνση URL παρέχει μια μέθοδο για την εύρεση του.

Από τεχνικής άποψης, ένα URL είναι ένα URI, που εκτός από τον εντοπισμό ένος διαδικτυακού πόρου

6.2 Μέθοδοι - Κλάσεις

Στην εφαρμογή περιγράφονται οι εξής οντότητες: Οι χρήστες(users), οι εγγραφές(subscriptions) και τα νέα (posts). Κάθε μοντέλο αντιστοιχεί σε μια collection στη βάση δεδομένων σύμφωνα με τη δομή της mongodb.

Οι ρυθμίσεις για τη βάση δεδομένων περιγράφονται στο αρχείο config/mongoid.yml

Η Ruby on Rails υποστηρίζει τρια διαφορετικα περιβάλλοντα εργασίας, ανάπτυξης (development), ελέγχων (test), παραγωγής (production). Κάθε περιβάλλον χρείαζεται

διαφορετική βάση για την αποφυγή λαθών, διαγραφή δεδομένων κτλ. Για τη σύνδεση της εφαρμογής με τη βάση δεδομένων χρησιμοποείται το gem mongoid.

Ένα παράδειγμα κώδικα του άρχειου mongoid.yml είναι:

```
development:
  host: localhost
  database: feedal_development


test:
  host: localhost
  database: feedal_test


production:
  uri: <%= ENV['MONGOHQ_URL'] %>
```

6.2.1 Μοντέλο user

Στο μοντέλο χρήστης ορίζεται η σχέση ένα με πολλά μεταξύ των πινάκων των εγγραφών. Αν διαγραφεί κάποιος χρήστης, θα διαγραφουν και οι εγγραφές που έχει κάνει.

```
has_many :subscriptions, :class_name => 'Subscription'
```

Από το gem devise θα χρησιμοποιηθουν οι παρακάτω λειτουργίες:

- εγγραφή και ταυτοποίηση των χρηστών
- αποστολή μηνυμάτων στο ηλεκτρονικο ταχυδρομείο των χρηστών, για την επιβεβαίωση του λογαριασμού και για την περίπτωση αλλαγής του κωδικού πρόσβασης
- αποθήκευση αρχείου cookie για να μη χρειάζεται ο χρήστης να προσθετει συνέχεια τα στοιχειά του για να μπαίνει στην εφαρμογή

```
devise :database_authenticatable, :registerable,:token_authenticatable, :recoverable,
:rememberable, :trackable, :validatable6.2.2 Μοντελο Subscription
```

Κάθε εγγραφή συνδέεται με έναν χρήστη

```
belongs_to :user,  :class_name => 'User',:inverse_of => nil
```

Η κάθε εγγραφή περιέχει τα εξης πεδία: Όνομα(name), url, user_id. Τα πεδία αυτά δε μπορουν να είναι κενά. Σε περίπτωση που είναι η εφαρμογή βγάζει την ένδειξη λάθους.

Για να δηλώσουμε αυτό στο μοντέλο μας γράφουμε:

```
validates_presence_of :name,:uniqueness => true
validates_presence_of :url,:uniqueness => true
validates_presence_of :user_id,:uniqueness => true
```

Στο μοντέλο εγγραφή ορίζεται η σχέση πολλά με πολλά, καθώς μια εγγραφή μπορεί να έχει πολλά και να ανηκει σε πολλά νέα.

```
has_and_belongs_to_many :posts, :dependent => :destroy, :inverse_of => nil
```

Μέσα στο μοντέλο είναι δηλωμένες και τρεις συναρτήσεις που βοηθούν στην συγκέντρωση και κατανομή των νέων με βάση το id τους. Για να μπορέσει να προσπελαστεί το url για να μπορέσει να "τραβήξει" τα δεδομένα χρησιμοποιούμε τη συνάρτηση URI(Uniform resource identifier), που ουσιαστικά είναι μια κλάση που μπορεί να χειριστεί τα url.

6.2.3 Μοντέλο "posts"

Στο μοντέλο νέο ορίζεται η σχέση πολλά με πολλά.Συγκεκριμένα το νέο έχει και ανηκει σε πολλές εγγραφές. Επίσης, κάθε νέο μπορεί να έχει και να ανήκει σε πολλά θέματα.

Τα πεδία που υπάρχουν στο μοντέλο νέα είναι τα: Τίτλος, περιεχόμενο, συγγραφέας, περιληψη, url, αναγνωριστικό του νέου,ημερομηνία δημιουργίας και ημερομηνία επεξεργασίας.Αυτή τη μορφή έχουν τα νέα μέσα στη βάση δεδομένων. Ανάλογα με τη μορφή που έχει ένα νέο στη json μορφή του, μπορεί κάποιο από αυτά να είναι κενό.

```
field :title, :type => String
field :body, :type => String
```

```
field :author, :type => String
field :post_identifier, :type => String
field :updated_at, :type => DateTime
field :published_at, :type => DateTime
field :summary, :type => String
field :url, :type => String
```

6.2.4 Μοντέλο Topic

Το μοντέλο χρησιμοποείται στην περίπτωση που το rss feed που έχουμε τραβήξει είναι κατηγοριοποιημένο σε θέματα(topic).

```
has_and_belongs_to_many :subscription
has_and_belongs_to_many :topics
```

6.2.5 Ελεγκτής εφαρμογής (application_controller.rb)

Η εντολή protect_from_forgery, στον κώδικα του ελεγκτή εφαρμογής που ακολουθεί, προσφέρει προστασία από CSRF (Cross-Site Request Forgery) επιθέσεις. Οι επιθέσεις αυτές εκτελούν κακόβουλο κώδικα εκτός της ιστοσελίδας "τρέχοντας" σενάρια (scripts). Η συγκεκριμένη εντολή προστατεύει από όλες τις HTTP αιτήσεις την εφαρμογή, με εξαίρεση των GET. Ο λόγος που επιτρέπονται οι GET αιτήσεις από άλλες ιστοσελίδες, είναι η συχνή και απαραίτητη χρήση τους από την ίδια την εφαρμογή (π.χ. Η χρήση του CDN της Google για τη φόρτωση της βιβλιοθήκης jQuery mobile).

Ο τρόπος λειτουργίας της προστασίας επιτυγχάνεται ως εξής: Η Rails δημιουργεί για κάθε φόρμα μία ακολουθία χαρακτήρων (token). Η ακολουθία αυτή πρέπει να ταιριάζει με την αποθηκευμένη ακολουθία για να γίνει αποδεκτή η υποβολή HTTP αίτησης.

```
<!-- κώδικας που παράγεται για κάθε φόρμα -->
</meta content="authenticity_token" name="csrf-param" />
<meta content="1nzF4kk3ltA/smYVqeKLq2FLTVILOUS0ID2BOfKbERM=" name="csrf-token" />


# κώδικα ελεγκτή εφαρμογής για την προστασία από xss επιθέσεις
class ApplicationController < ActionController::Base
```

```
protect_from_forgery
end
```

6.2.6 Ελεγκτής αρχικής σελίδας (front_controller.rb)

Ο ελεγκτής αρχικής σελίδας ορίζει το περιεχόμενο που θα εμφανίζεται στην αρχική σελίδα. Έχουν δημιουργηθεί 3 κενές κλάσεις home, contact, help που κάθε φορά που τις καλει ο χρήστης ανακατευθύνονται στα στατικα template που υπάρχουν στο view.

6.2.7 Ελεγκτής Εγγραφής (subscriptions_controller.rb)

Πριν οποιαδήποτε λειτουργία που αφορά τις εγγραφές πρέπει να έχει γίνει ταυτοποίηση του χρήστη. Αυτός ο έλεγχος επιτυγχάνεται με την εντολή:

```
before_filter :authenticate_user!
```

Συνοπτικά, ο ελεγκτής διαδρομής περιλαμβάνει τις εξής ενέργειες (actions):

Ενέργεια	Περιγραφή
index	Εμφάνιση όλων των εγγραφών του συνδεδεμένου χρήστη. Αν δεν είναι συνδεμένος ο χρήστης εμφανίζει τις προκαθορισμένες εγγραφές.
show	Εμφάνιση εγγραφής με συγκεκριμένο id.
create	Έλεγχος έγκυρων παραμέτρων όπως έχουν οριστεί στο μοντέλο και δημιουργία αντικειμένου στη βάση με POST.

6.2.8 Ελεγκτής Νέων (posts_controller.rb)

Πριν οποιαδήποτε λειτουργία που αφορά τις εγγραφές πρέπει να έχει γίνει ταυτοποίηση του χρήστη. Αυτός ο έλεγχος επιτυγχάνεται με την εντολή:

```
before_filter :authenticate_user!
```

Συνοπτικά, ο ελεγκτής διαδρομής περιλαμβάνει τις εξής ενέργειες (actions):

Ενέργεια	Περιγραφή
index	Εμφάνιση όλων των νέων του συνδεδεμένου χρήστη.
show	Εμφάνιση νέου με συγκεκριμένο id.

6.2.9 Ελεκτής θεμάτων(topics_controller.rb)

Ενέργεια	Περιγραφή
create	Δημιουργία θέματος.
show	Εμφάνιση θέματος με συγκεκριμένο id και εμφάνιση όλων των νέων με αυτό το θέμα.

6.3 Δημιουργία του API

Η βασική ιδέα πίσω από τη δημιουργία του api για την εφαρμογή είναι να επιτρέπει στους χρήστες που έχουν την εφαρμογή στο κινητό τους να μπορούν να κάνουν σύνδεση και αποσύνδεση μέσω json αιτημάτων. Αφού συνδεθεί επιτυχώς στο σύστημα, ο χρήστης θα δέχεται ένα authentication token(κλειδι πιστοποιησης) που θα μπορεί να χρησιμοποιηθεί στα api αιτήματα για να εγκρίνει το χρήστη και να εξασφαλίζει την σωστή πρόσβαση στα αρχεία της εφαρμογής.

Τα αρχεία του API βρίσκονται στον φάκελο app/controller/api/v1.

Ο sessions controller έχει δυο ενέργειες (actions): create για την σύνδεση και destroy για την αποσύνδεση. Η πρώτη ενέργεια δέχεται έναν json αντικείμενο του χρήστη ως POST δεδομένα, με παραμέτρους το email και τον κωδικό του χρήστη και επιστρέφει ένα auth_token μαζί με ένα μήνυμα επιτυχιας, αν ο χρήστης υπάρχει στη βάση δεδομένων και ο κωδικός πρόσβασης είναι σωστός. Για την αποσύνδεση περιμένει ένα auth_token σαν παράμετρο στο url του.

```
class Api::V1::SessionsController < Devise::SessionsController
  skip_before_filter :verify_authenticity_token,
                     :if => Proc.new { |c| c.request.format == 'application/json' }

  respond_to :json

  def create
```

```
  warden.authenticate!(:scope => resource_name, :recall =>
"#{controller_path}#failure")
  render :status => 200,
        :json => { :success => true,
                   :current_user => current_user.id,
                   :data => { :auth_token => current_user.authentication_token } }
 end

 def destroy
  warden.authenticate!(:scope => resource_name, :recall =>
"#{controller_path}#failure")
  current_user.update_column(:authentication_token, nil)
  render :status => 200,
        :json => { :success => true,
                   :info => "Logged out",
                   :data => {} }
 end

 def failure
  render :status => 401,
        :json => { :success => false,
                   :info => "Login Failed",
                   :data => {} }
 end
end
```

Για να μπορέσει να τρέξει το api πρέπει να αλλάξουμε και τις διαδρομές απόκρισης της εφαρμογής:

```
namespace :api do
  namespace :v1 do
    devise_scope :user do
      post 'sessions' => 'sessions#create', :as => 'login'
```

```
    delete 'sessions' => 'sessions#destroy', :as => 'logout'
  end
  resources :tokens,:only => [:create, :destroy]
 end
end
```

6.3.1 Διαδρομές απόκρισης της εφαρμογής (routes.rb)

Οι διαδρομές (paths) της εφαρμογής ρυθμίζονται στο αρχείο routes.rb. Συγκεκριμένα, οι διαδρομές συνδέονται με τα κατάλληλα HTTP ρήματα και προωθούνται στους ελεγκτές. Οι παραγόμενες διαδρομές αποτυπώνονται στην παρακάτω εικόνα. Παρατηρούμε, ότι στην πρώτη στήλη εμφανίζονται οι "μεταβλητές" κάθε διαδρομής. Με αυτόν τον τρόπο μας δίνεται η δυνατότητα χρήσης τους στην εφαρμογή και σε περίπτωση που γίνει αλλαγή της ίδιας της διαδρομής στο μέλλον (π.χ. αλλάξει το όνομα κάποιας οντότητας), να διατηρηθεί η λειτουργικότητα. Στη δεύτερη στήλη εμφανίζεται το HTTP ρήμα απόκρισης της εφαρμογής. Η τρίτη στήλη δείχνει τη διαδρομή και τις πιθανές παραμέτρους της και τις αποδεκτές μορφές που μπορεί να έχει. Η τέταρτη και τελευταία στήλη δηλώνει τους ελεγκτές και τις ενέργειες που καλούνται ανάλογα με τη διαδρομή.

Μέρος Β': Android

Γλωσσάρι ενότητας Android

debug - διόρθωση λογικών λαθών

web browser - πρόγραμμα περιήγησης στο διαδίκτυο

cpu - κεντρική μονάδα επεξεργασίας

e-books - βιβλία σε ηλεκτρονική μορφή

video-chat - τηλεδιάσκεψη

codex - κωδικοποιητές

android runtime - χρόνος εκτέλεσης του Android

Dalvik virtual machine - εικονική μηχανή Dalvik

Package Manager - διαχειριστής πακέτων

Activity Manager - διαχειριστής των δραστηριοτήτων

Content Providers - διαχειριστές περιεχομένου

activity - δραστηριότητα / κλάση του Android

layout - σχέδιο

ram - προσωρινή μνήμη

widget - γραφικό στοιχειό

social media - μέσα κοινωνικής δικτύωσης

values - τιμές

server - εξυπηρετητής

client - πελάτης

Κεφάλαιο 7: Η ιστορία των κινητών τηλεφώνων

Η ιστορία των κινητών τηλεφώνων ξεκίνησε μετά τον Β' Παγκόσμιο Πόλεμο έχοντας ως κύριους ανταγωνιστές για την κατασκευή τους, τους Σουηδούς, τους Αμερικανούς και τους Φιλανδούς. Η ακριβής ημερομηνία της "γέννησης" του πρώτου κινητού τηλεφώνου όμως θεωρείται η 3η Απριλίου του 1973 και ο "τόπος γέννησης" η Νέα Υόρκη. Εκεί ο δόκτωρ Μάρτιν Κούπερ της Motorola ήταν αυτός που έκανε την πρώτη κλήση απο το πρώτο κινητό τηλέφωνο το οποίο έμοιαζε με φορητό ασύρματο και είχε ύψος 25 εκατοστά και βάρος 900 γραμμάρια. Το όνομα που αποφάσισε να του δώσει η Motorola ήταν το MotorolaDynaTAC . Η πρώτη κλήση του MotorolaDynaTAC έγινε στον ανταγωνιστή του Μάρτιν Κούπερ, τον Τζόελ Ένγκελ, που δούλευε για λογαριασμό της Bell Labs. Η Bell Labs παρόλα αυτά ήταν η εταιρία που κατασκεύασε το πρώτο δοκιμαστικό δίκτυο κινητής τηλεφωνίας, που ήταν αναγκαίο για την εξέλιξη και την εμπορική εκμετάλλευση του κινητού.

Μέχρι τα τέλη της δεκαετίας του '80 τα κινητά τηλέφωνα ήταν ογκώδη για να μεταφέρονται στην τσέπη κι έτσι ήταν εγκατεστημένα κυρίως σε αυτοκίνητα. Το πρώτο κινητό που έλαβε άδεια έγκρισης ήταν το μοντέλο της *Motorola* DynaTAC8000X και υπήρξε η ναυαρχίδα των λεγόμενων κινητών πρώτης γενιάς (1G).

Motorola DynaTAC 8000X

Στην αρχή της δεκαετίας του '90 άρχισε η απογείωση των κινητών τηλεφώνων, με την ψηφιοποίηση δικτύων (GSM) και συσκευών. Τα κινητά έγιναν μικρότερα και ελαφρύτερα (100-200 γραμμάρια), χωρούσαν στην παλάμη και έμπαιναν έστω και με δυσκολία στην τσέπη του χρήστη τους. Έτσι έγινε το πέρασμα στα κινητά της δεύτερης γενιάς (2G), που παρείχαν και άλλες ευκολίες, όπως την αποστολή σύντομων γραπτών μηνυμάτων (SMS) και τη λήψη φωτογραφιών.

Στις αρχές του 21ου αιώνα έκαναν την εμφάνιση τους τα κινητά τρίτης γενιάς (3G), με τις απεριόριστες δυνατότητες των πολυμέσων. Μελέτες που έγιναν το 2011 έδειξαν ότι με τον πληθυσμό της γης να ξεπερνά τα 7 δισεκατομμύρια τα κινητά τηλέφωνα είναι πάνω απο 5.6 δισεκατομμύρια ποσοστό που αγγίζει το 80% του πληθυσμού της γης. Αυτό δείχνει ότι τα κινητά τηλέφωνα έχουν γίνει αναπόσπαστο κομμάτι της ζωής μας και έχουν φτάσει ακόμα και σε χώρες της Αφρικής όπου κατά κοινή ομολογία η τεχνολογία

υστερεί. Αυτό έχει γίνει λόγο των πολύ χαμηλών τιμών που έχουν στις μέρες μας αρκετές συσκευές αλλά και στις απίστευτες ευκολίες που μπορούν να παρέχουν ακόμα και οι πιο φτηνές συσκευές.

- 1947 - Γεννιέται η ιδέα του κινητού τηλεφώνου, όταν οι επιστήμονες των AT&T(Cingular) συνειδητοποιούν ότι ένας πομπός μικρής εμβέλειας μπορεί να μεταμορφωθεί σε πομπό μεγάλης εμβέλειας συνδέοντας πολλές <<κυψέλες>> ενός τοπικού δικτύου.
- 1950s - Διαδίδονται τα τηλέφωνα στο αυτοκίνητο. Το 1954 ο επιχειρηματίας με όνομα Larrabee πραγματοποιεί μια κλήση από το τηλέφωνο της λιμουζίνας του.
- 1970s - Οι Rich&famous στη Βρετανία εμφανίζονται όλο και περισσότερο να χρησιμοποιούν τηλέφωνο στο αυτοκίνητο.
- 1973 - Ο Dr. Martin Cooper της Motorola κάνει το πρώτο τηλεφώνημα στον ανταγωνιστή του Joel Engel(AT&T Bell Labs) περπατώντας στους δρόμους της Νέας Υόρκης χρησιμοποιώντας το Motorola DynaTAC.
- 1979 - Αρχίζει η λειτουργία του πρώτου εμπορικού δικτύου κινητής τηλεφωνίας στο Τόκιο.
- 1983 - Ο Dr. Martin Cooper παρουσιάζει το DynaTAC 8000X που στοιχίζει 2,500$.
- 1984 - Παρά της τσιμπημένες τιμές των κινητών τηλεφώνων περίπου 300.000 άνθρωποι στον κόσμο έχουν στην κατοχή τους ένα.
- 1989 - Η Motorola παρουσιάζει το MicroTac. Έκπληκτοι αφελείς τρέχουν να αποκτήσουν το τηλεφωνικό θαύμα που δεν είναι δύσκολο στην χρήση και κυρίως στην μεταφορά.
- 1990 - Έρχεται η δεύτερη γενιά κινητής τηλεφωνίας (2G) συμπεριλαμβανομένου του GSM και στις ΗΠΑ πραγματοποιείται το πρώτο ψηφιακό τηλεφώνημα από κινητό.
- 1991 - Η Ευρώπη ακολουθεί το παράδειγμα των ΗΠΑ καθώς ανοίγει το πρώτο GSM ευρωπαϊκό δίκτυο.
- 1992 - Κυκλοφορεί το περίφημο Νοκια 101 και κλέβει τις εντυπώσεις.
- 1996 - Λανσάρεται στην αγορά το Motorola StarTAC το πρώτο Clamshell κινητό και το μικρότερο της εποχής του. Λίγο αργότερα, Βραβεύεται ως ένα από τα 50 σημαντικότερα gadgets του δεύτερου μισού του 20ου αιώνα.
- 1999 - Το Matrix κάνει διάσημα τα κινητά μπανάνα 7110 και 8110. Ποτέ άλλοτε τα κινητά της Nokia δεν ήταν τόσο δημοφιλή.
- 2000 - Ξεκινά η λειτουργία των 3G δικτύων τα οποία επιτρέπουν στους συνδρομητές να μεταφέρουν τεράστιες ποσότητες data ασύρματα και να κάνουν βιντεοκλήσεις. Ωστόσο κανένας δεν κάνει το τελευταίο.
- 2001 - Κυκλοφορεί το SE T68 το πρώτο μαζικό έγχρωμο κινητό.
- 2004 - Οι πωλήσεις των ringtones ξεπερνούν τα 2,5 δις.$.
- 2007 - Περίπου 1,3 δις άνθρωποι έχουν κινητό δηλαδή το 1/5 του πληθυσμού της Γής.
- 2008 - Πρωτοεμφανίζονται τα Smartphones
- 2011 - Το 80% των ανθρώπων του πλανήτη έχουν κινητό

Κεφάλαιο 8: Ιστορικά για το λειτουργικό σύστημα Android

Το λειτουργικό σύστημα Android δημιουργήθηκε απο την εταιρία Android και αναπτύχτηκε από τον Andy Rubin, τον Rich Miner, τον Nick Sears και τον Chris White το 2003 με στόχο όπως είχε δηλώσει ο Andy Rubin να υπάρξουν έξυπνα κινητά τηλέφωνα. Στις 17 Αύγουστου του 2005 και μη έχοντας πάρει μεγάλες διαστάσεις μέχρι τότε το συγκεκριμένο project η Google αγόρασε την εταιρία Android διατηρώντας το μέχρι τότε έμψυχο δυναμικό της προσθέτοντας εργαζόμενους απο την ίδια την Google. Με αυτή την αγορά η Google ήθελε να κάνει ένα αποφασιστικό βήμα στην αγορά των κινητών τηλεφώνων το όποιο όπως αποδείχτηκε στο μέλλον της εξασφάλισε τεράστια κέρδη.

Στις 5 Νοεμβρίου του 2007 η Google ιδρύει την Open Handset Alliance έναν όμιλο εταίρων όπου συμμετείχαν εταιρίες όπως Google, HTC, Intel, LG, Nvidia, Samsung κ.α έχοντας ως στόχο την ανάπτυξη ενός ανοιχτού λογισμικού για κινητά τηλέφωνα. Την ίδια μέρα η Open Handset Alliance παρουσίασε το πρώτο της προϊόν. Ένα λειτουργικό σύστημα βασισμένο στον πυρήνα του Linux 2,6. Το Android. Παρά τις ομοιότητες που έχουν οι δυο πυρήνες του Android και του Linux 2,6 το Android δεν υποστηρίζει το πλήρες σετ βιβλιοθηκών του GNU καθώς και δεν υποστηρίζει το X Window System (πρωτόκολλο απεικόνισης για τη δημιουργία "παραθύρων" στην οθόνη ενός υπολογιστή). Άρα δεν είναι εύκολο να τρέξουμε εφαρμογές του Linux στο Android.

Απο τις 5 Νοεμβρίου του 2007 κυκλοφορεί η δοκιμαστική έκδοση του Android ενώ το λογισμικό ανάπτυξης (SDK) κυκλοφόρησε στις 12 Νοεμβρίου του 2007. Απο τότε και μέχρι σήμερα έχουν κυκλοφορήσει στην αγορά 9 βασικές εκδόσεις του Android προσθέτοντας η κάθε μια ακόμα πιο πολλά χαρακτηριστικά και δυνατότητες στις μέχρι τότε προϋπάρχουσες εκδώσεις. Η Google υπήρξε αρκετά πρωτότυπη στην ονομασία που έδωσε στις επιμέρους βασικές εκδώσεις του Android μιας και επέλεξε να χρησιμοποιήσει ονόματα απο την ζαχαροπλαστική και πιο συγκεκριμένα απο γλυκά που είναι αγαπητά σε όλους.

Στις 30 Απριλίου του 2009 παρουσιαστικά η πρώτη έκδοση με όνομα 1.5 ή αλλιώς "Cupcake" και βασισμένη στον πυρήνα του Linux 2.6.27. Υποστήριζε καινούργιες λειτουργιές για την κάμερα της συσκευής μιας και μέχρι τότε ο χρήστης μπορούσε απλά να τραβήξει φωτογραφίες χωρίς να μπορεί να παρέμβει στις ρυθμίσεις της κάμερας όπως αλλαγή ανάλυσης, φωτεινότητας κλπ. Ο χρήστης πια μπορούσε να τραβήξει και βίντεο καθώς να και να τα ανεβάσει στο YouTube ή στο Picasa(αντίστοιχα για τις φωτογραφίες) απευθείας απο το κινητό του. Το πληκτρολόγιο διέθετε αυτόματη συμπλήρωση και πρόβλεψη κειμένου ενώ το Bluetooth της συσκευές υποστήριζε πρωτοκολλά A2DP και AVRCP. Στο αισθητικό κομμάτι Google προσέθεσε εφέ στην αρχική οθόνη με κινούμενες μεταβάσεις είτε αριστερά είτε δεξιά.

Η επόμενη έκδοση που έβγαλε η Google λεγόταν 1.6 ή αλλιώς " Donut" και παρουσιάστηκε στις 15 Σεπτεμβρίου του 2009 και ο πυρήνας που είχε ήταν ο 2.6.27. Ήταν ταχύτερη σε συνολική απόκριση απο την προηγούμενη έκδοση ενώ τόσο η εφαρμογή για την προβολή εικόνων όσο και η φωτογραφική μηχανή είχαν ανανεωθεί και σε σχεδιαστικό περιβάλλον αλλά και σε ταχύτητα απόκρισης και ευκολία χρήσης. Η νέα έκδοση περιελάμβανε δυνατότητα επιλογής πολλαπλών αρχείων και διαχείρισης τους αλλά και βελτιωμένη εφαρμογή Google Market όπου ο χρήστης μπορούσε πια να δει μερικά στιγμιότυπα απο την εφαρμογή που επρόκειτο να κατεβάσει. Στον τομέα τον επαφών και των κλήσεων προστέθηκε η δυνατότητα της φωνητικής κλήσης αλλά και αναζήτησης επαφών. Τέλος το Android Donut υποστήριζε και αναλύσεις οθονών WVGA δίνοντας έτσι την δυνατότητα σε περισσότερες συσκευές να το έχουν.

Η έκδοση 2.0 ή αλλιώς "Éclair" παρουσιάστηκε στις 26 Οκτώβριου του 2009 και ήταν και αυτή βασισμένη στον ίδιο πυρήνα με την 1.6 και τον Ιανουάριο του 2010 επανακυκλοφόρησε με το αλλάζοντας απο 2.0 σε 2.1 κρατώντας όμως την ονομασία "Éclair" ίδια. Η ανανεωμένη έκδοση συνέχισε να βελτιώνεται στο κομμάτι της απόδοσης εκμεταλλευόμενη καλλίτερα το υλικό και επιπλέων αυξήθηκαν οι υποστηριζόμενες απο αυτήν οθόνες και αναλύσεις. Ο web browser υποστήριζε πια HTML5, το περιβάλλον του χρήστη είχε αλλάξει ριζικά και είχε ανανεωθεί και η εφαρμογή Google maps. Στο κομμάτι της κάμερας το Android υποστήριζε φλας καθώς και ψηφιακό zoom. Στον κομμάτι της οθόνης είχαν αναβαθμιστεί τα κομμάτια του κώδικα τα όποια πλέων επέτρεπαν τα πολλαπλά σημεία αφής στην οθόνη (multitouching) .

Η έκδοση 2.2 ή αλλιώς "Froyo" βασίστηκε πια τον πυρήνα 2.6.32 και κυκλοφόρησε στις 20 Μαΐου του 2010. Οι βελτιώσεις στην ταχύτητα συνεχιστικά και σε αυτή την έκδοση έγινε ακόμα καλλίτερη διαχείριση της μνήμης και της cpu. Η ενσωμάτωση του μηχανισμού JavaScript απο τον Google Chrome στον web browser ήταν ένα απο τα σημαντικά στοιχειά αυτής της έκδοσης. Ένα άλλο ήταν ότι πια υποστηριζόταν και ο Adobe Flash 10.1. Συνεχίζοντας η Google έδωσε την δυνατότητα στον χρήστη να αποθηκεύει της εφαρμογές του όχι μόνο στο τηλέφωνο αλλά και στην εξωτερική κάρτα μνήμης καθώς και να μπορεί να ενεργοποιεί και να απενεργοποιεί την χρήση των δικτύων κινητής τηλεφωνίας για μεταγωγή πακέτων δεδομένων(3g). Τέλος το κινητό μπορούσε πια να λειτούργει και σαν Wi-Fi hotspot

Η έκδοση που κατέχει το μεγαλύτερο μερίδιο της αγοράς και είναι και μέχρι σήμερα η πιο διαδεδομένη σε όλα τα κινητά είναι η έκδοση 2.3 ή αλλιώς "Gingerbread" η όποια βασίζεται στον πυρήνα 2.6.35.7 και κυκλοφόρησε για πρώτη φορά στις 6 Δεκεμβρίου του 2010 ενώ επανακυκλοφόρησε βελτιωμένη και με τον κωδικό 2.3.3 τον Φεβρουάριο του επόμενου έτους. Η ταχύτητα και η εκμετάλλευση των πόρων της συσκευής συνέχισαν να βελτιώνονται, όπως και σε κάθε έκδοση άλλωστε, ενώ το ίδιο συνέβη και με τις οθόνες που πια υποστηρίζονταν οθόνες υψηλής ανάλυσης και μεγάλων μεγεθών. Το περιβάλλον του χρήστη έγινε πιο απλό και πιο γρήγορο, η ποιότητα του ήχου καλύτερη καθώς και ο τρόπος απεικόνισης των παιχνιδιών που πια υποστηρίζουν υψηλής ποιότητας γραφικά. Το Android 2.3.3 υποστηρίζει την λειτουργία παραπάνω απο μιας κάμερας. Τέλος έγινε και μετάβαση στο σύστημα αρχείων απο YAFFS σε ext4.

Η Google δεν σταμάτησε εκεί και με την έκδοση 3.0 ή αλλιώς "Honeycomb" επεκτάθηκε και στα tablets. Η έκδοση αυτή βασίστηκε στον πυρήνα 2.6.36 και παρουσιάστηκε στις 9 Μαΐου του 2011. Ο προσανατολισμός της έκδοσης αυτής ήταν να εκμεταλλευτεί στο έπακρο τις υψηλές επιδόσεις που είχαν τα tablets για αυτό υποστήριζε διπύρηνους και τετραπύρηνους επεξεργαστές ενώ δημιουργήθηκε ένα τελείως διαφορετικό περιβόλων χρήστη ιδικά για τα tablets. Λόγο του μεγέθους του tablet το Android 3.0 υποστήριζε την προβολή e-books καθώς και την δυνατότητα πραγματοποίησης video chat μέσω της εφαρμογής Google Talk. Το multitasking βελτιώθηκε σημαντικά για την πλήρη αξιοποίηση του tablet.

Η τελευταία έκδοση που έχει βγει έχει μπει αρκετά δυναμικά στην αγορά και ονομάζεται "Ice Cream Sandwich" και έχει κωδικό 4.0.1. Για άλλη μια φορά η διαχείριση των πόρων βελτιώθηκε και το περιβάλλον του χρήστη άλλαξε κατά πολύ σε σχέση με την προηγούμενη έκδοση. Έγινε μεγάλη βελτίωση στην ασφάλεια καθώς υποστηρίζεται πλέον αναγνώριση προσώπου για το ξεκλείδωμα της συσκευής αλλά και στο κομμάτι του web browser μιας και είναι πολύ πιο γρήγορος και μπορεί να ανοίξει μέχρι και 16 καρτέλες. Με την ύπαρξη του Wi-Fi Direct πλέον επιτρέπεται η σύνδεση συσκευών μεταξύ τους χωρίς να μεσολαβεί κάποιο access point. Τέλος το Android 4.0.1 υποστηρίζει την εγγραφή και αναπαραγωγή βίντεο 1080p.

Κεφάλαιο 9: Hardware

Το hardware σε κάθε συσκευή Android μπορεί να χωριστεί στις εξής κατηγορίες:

- Την οθόνη
- Τα κουμπιά
- Το πληκτρολόγιο
- Τους αισθητήρες

9.1 Οθόνες

Ποιο αναλυτικά οι οθόνες μπορούν να χωριστούν ανάλογα με τον τρόπο απεικόνισης.

- TFT
- OLED
- AMOLED

Οι οθόνες TFT (Thin Film Diode) είναι ένας απο τους καλυτέρους τύπους οθονών υγρών κρυστάλλων όσο αφορά την ποιότητα απεικόνισης. Παρόλα αυτά καταναλώνουν μεγάλη ενεργεία και είναι ακριβές στην κατασκευή τους. Χρησιμοποιούν την τεχνολογία active-matrix που σημαίνει πως διπλά σε κάθε pixel της οθόνης υπάρχει ένας πομποδέκτης που επιτρέπει στο pixel να λειτουργει αυτόνομα. Έτσι επιτυχαίνεται καλλίτερη αντίθεση και απόκριση.

Οι οθόνες OLED (Organic Light-Emitting Diode) χρησιμοποιούν μια τεχνολογία απεικόνισης που αποτελείται απο μικρές κουκίδες απο οργανικό πολυμερές που εκπέμπει φως όταν φορτίζεται ηλεκτρικά. Τα πλεονεκτήματα των οθονών oled απο τις lcd είναι ότι οι oled είναι φτηνότερες στην κατασκευή τους, είναι λεπτότερες, φωτεινότερες, χρειάζονται λιγότερη ενεργεία, έχουν καλλίτερη αντίθεση και καλλίτερο χρόνο απόκρισης. Το μόνο μειονέκτημα είναι ότι οι οθόνες lcd παρέχουν καλλίτερη οπτική σε φωτεινά περιβάλλοντα.

Οι οθόνες AMOLED (Active-matrix Light-Emitting Diode) έχουν βελτιωμένη απόδοση σε σχέση με τις οθόνες OLED αλλά δραματικά χαμηλότερη κατανάλωση μπαταριάς όποτε προτιμούνται σε κινητά υψηλής τεχνολογίας.

Στον πίνακα που ακλουθεί βλέπουμε την ταξινόμηση των οθονών με βάση τις διαστάσεις αλλά και την πυκνότητά τους.

Τύπος Οθόνης	Χαμηλής πυκνότητας (~ 120 pixel/ίντσα) ldpi	Μεσαίας πυκνότητας (~160 pixel/ίντσα) mdpi	Υψηλής πυκνότητας (~240 pixel/ίντσα) hdpi
Μικρή οθόνη	QVGA (240x320) 2.6 έως 3.0 ίντσες	-	-
Μεσαία οθόνη	WQVGA (240x400) 3.2 έως 3.5 ίντσες & FWQVGA (240x432) 3.2 έως 3.5 ίντσες	HVGA (320x480) 3.0 έως 3.5 ίντσες	WVGA (480x800) 3.3 έως 4.0 ίντσες & FWVGA (480x854) 3.5 έως 4.0 ίντσες
Μεγάλη οθόνη	-	WVGA (480x800) 4.8 έως 5.5 ίντσες & FWVGA (480x854) 5.0 έως 5.8 ίντσες	-

Όπως βλέπουμε οι οθόνες χωρίζονται σε μικρές, μεσαίες και μεγάλες ανάλογα με τις ίντσες τους αλλά και σε χαμηλής, μεσαίας και υψηλής ευκρίνειας. Το μεγαλύτερο ποσοστό των συσκευών που υποστηρίζουν Android σήμερα είναι στην κατηγορία normal screen και medium density και πάνω. Πέραν αυτού υπάρχουν και δύο κατηγορίες τεχνολογιών για τον τρόπο που αντιλαμβάνοντα το άγγιγμα οι οθόνες αφής.

- Resistive
- Capacitive

Η Resistive ή αλλιώς αντιστασιακή οθόνη αφής αποτελείται απο πολλά και λεπτά στρώματα. Τα πιο σημαντικά είναι δύο λεπτές μεταλλικές ηλεκτραγώγιμες και αντιστάσιμες επιφάνειες, που χωρίζονται από λεπτό διάστημα. Όταν κάποιο αντικείμενο αγγίζει αυτό το είδος της επιφάνειας αφής, τα στρώματα συνδέονται σε ένα ορισμένο σημείο, και στη συνέχεια η επιφάνεια αντιδρά ηλεκτρικά παρόμοια με δύο διαιρέτες τάσης που συνδέονται με τις εκροές τους. Αυτό προκαλεί μια αλλαγή στο ηλεκτρικό ρεύμα και εκλαμβάνεται ως νέο γεγονός αφής που πια διαχειρίζεται ο Surface Manager, όπως αναφέρθηκε σε προηγούμενη ενότητα. Οι συγκεκριμένες οθόνες είναι λιγότερο ευαίσθητες και πια δεν χρησιμοποιούνται τόσο πολύ όσο τα πρώτα χρόνια της κυκλοφορίας των κινητών με οθόνες αφής. Τις περισσότερες φορές η συσκευή είναι εφοδιασμένη με μια γραφίδα στην όποια η οθόνη 'αντιδρά" καλυτέρα απο ότι στο ανθρωπινό δάχτυλο.

Η Capacitive ή αλλιώς χωρητική οθόνη αφής λειτούργει χρησιμοποιώντας το φαινόμενο Capacitive coupling. Είναι το φαινόμενο κατά το οποίο αλλάζει το ηλεκτρικό φορτίο ενός σώματος όταν ένα άλλο φορτισμένο σώμα πλησιάσει. Ως φορτισμένο σώμα θεωρείται ο άνθρωπος. Έτσι μετακινώντας το δάχτυλο στην οθόνη αλλάζει το ηλεκτρικό φορτίου ενός πλέγματος που βρίσκεται στο κάτω μέρος της οθόνης και στην συνέχεια ένας ελεγκτής είναι εύκολο να εντοπίσει το σημείο όπου ακούμπησε ο χρήστης.

Άρα λοιπόν από την στιγμή που χρειάζεται η αλληλεπίδραση δύο φορτισμένων σωμάτων αυτός είναι και ο λόγος που σε αυτές τις οθόνες δεν μπορούμε να χρησιμοποιήσουμε γραφίδα ή να φορέσουμε κάποιο γάντι κατά την χρήση τους.

9.2 Κουμπιά

Συνεχίζοντας με το hardware έχουμε τα κουμπιά τα όποια συνήθως είναι 3 ή 4. Σε λίγα μοντέλα υπάρχει άλλο ένα κουμπί που μοιάζει με joystick ή με ένα είδος ποντικιού σαν αυτό που υπάρχει στους προσωπικούς υπολογιστές.

- Back button (κουμπί επιστροφής) - το συγκεκριμένο κουμπί είναι υπεύθυνο για να πηγαίνει στην προηγούμενη οθόνης απο αυτήν που είναι ο χρήστης ή για να ακυρώνει μια εργασία, εάν αυτό είναι δυνατό, να κλείσει μια εφαρμογή και τέλος επιτελεί και το κουμπί back που έχει ο web browser
- Home button (κουμπί μετάβασης στην αρχική οθόνη) - το κουμπί αυτό μεταφέρει τον χρηστή στην αρχική οθόνη βγαίνοντας απο οποιοδήποτε εφαρμογή χωρίς όμως να την τερματίσει. Η εφαρμογή συνεχίζει να εκτελείται στο παρασκήνιο
- Options button (κουμπί επίλογων) - το options button χρησιμοποιείται για την εμφάνιση ενός μικρού μενού στο κάτω μέρος της οθόνης με επιλογές που σχετίζονται με την εφαρμογή που βλέπει ο χρήστης εκείνη την στιγμή.
- Search button (κουμπί εύρεσης) - το κουμπιά αυτό ανοίγει ένα παράθυρο όπου ο χρήστης πληκτρολογεί κάτι που θέλει να βρει στο κινητό ή στην μηχανή αναζήτησης της Google
- trackball - είναι συνήθως μια μικρή μπίλια η όποια επιτρέπει στον χρηστή να επιλέξει διάφορα αντικείμενα στην οθόνη. Έχει δυνατότητα περιστροφής 360 μοίρες

Τα κουμπιά μπορεί να είναι είτε hard είτε touch. Με τον ορο hard ενοούμε οτι έχουν φυσικύ υπύσταση δηλαδύ μύκος, πλύτος, ύψος και είναι το αντίθετο του touch όπου απλά είναι αφής χωρίς αν εξέχουν από την συσκευή ήπως τα hard. Μερικές φορές μένο το home είναι hard και όλα τα άλλα touch.

Χαρακτηριστικό μοντέλο κινητού το HTC Nexus One με 4 hard buttons και joystick αλλά και το Samsung Galaxy S που δεν έχει search button αλλά ούτε trackball και hard home button.

9.3 Πληκτρολογιο

Η επόμενη κατηγορία του hardware είναι το πληκτρολόγιο. Σε αυτήν την κατηγορία συναντάμε δυο ειδών κινητά. Αυτά που έχουν ενσωματωμένο hard keyboard ή αυτά που το πληκτρολόγιο εμφανίζεται στην οθόνη αφής όταν υπάρχει δυνατότητα πληκτρολόγησης. Το hard keyboard είναι ακριβώς όπως το πληκτρολόγιο του προσωπικού μας υπολογιστή και βγαίνει συρταρωτά απο το κάτω μέρος του κινητού. Χαρακτηριστικό παράδειγμα είναι το Sony Ericson Xperia Mini Pro.

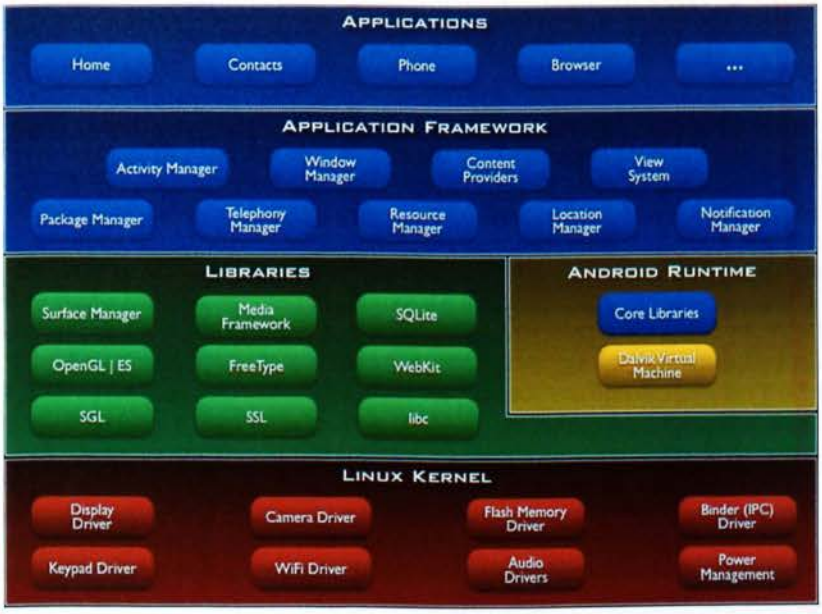
9.4 Αισθητήρες

Η τελευταία κατηγορία αλλά εξίσου σημαντική με τις υπόλοιπες είναι οι αισθητήρες που είναι εφοδιασμένα όλα τα κινητά. Με τις υψηλές επιδόσεις και τις μεγάλες δυνατότητες τους είναι απαραίτητοι κάποιοι αισθητήρες για την λειτουργιά ορισμένων εφαρμογών. Οι βασικές κατηγορίες των αισθητήρων που έχουν τα κινητά είναι οι εξής:

- Αισθητήρας μέτρηση της επιτάχυνσης της βαρύτητας
- Μαγνητόμετρο τριών αξόνων για την μέτρηση του μαγνητικού πεδίου
- Αισθητήρα θερμοκρασίας

Κεφάλαιο 10: Αρχιτεκτονική του Android

Το Android δεν είναι απλά ένα λειτουργικό σύστημα αλλά μια "στοίβα" λογισμικού που αποτελείται από το λειτουργικό σύστημα, το middleware και τις κυρίως εφαρμογές.



Με μια πρώτη ματιά στο διάγραμμα που δίνει η Google βλέπουμε το Android χωρίζεται στα έξης στρώματα:

- Τον πυρήνα (Linux Kernel)
- Τις βιβλιοθήκες (Libraries)
- Το Android Runtime που περιέχει τον Dalvik καθώς και τις βασικές βιβλιοθήκες
- Πλαίσιο Εφαρμογών (Application Framework)
- Τις εφαρμογές (Applications)

Ξεκινώντας από το τελευταίο στρώμα βλέπουμε ότι εκεί είναι ο πυρήνας του Linux που χρησιμοποιεί το Android. Εκεί είναι εγκατεστημένοι οι drivers που είναι υπεύθυνοι για τις λειτουργιές των επιμέρους στοιχείων του τηλεφώνου όπως το Wi-Fi, η οθόνη, η κάμερα κλπ.

Πιο πάνω υπάρχουν οι βιβλιοθήκες που χρησιμοποίει το Android που είναι γραμμένες σε C και C++. Ο Surface Manager είναι υπεύθυνος για την σωστή απεικόνιση και λειτουργιά της οθόνης. Για το πως δηλαδή κάθε διαδικασία θα εμφανίσει κάτι στην οθόνη την ώρα που πρέπει χωρίς να δημιουργεί πρόβλημα στις υπόλοιπες καθώς και για τον τρόπο που θα μετακινούνται τα διάφορα παράθυρα στην οθόνη

ανάλογα με τις επιλογές του χρήστη. Το OpenGL και το SGL είναι οι πλατφόρμες για τα 3D και τα 2D γραφικά που χρησιμοποιούν οι εφαρμογές. Η βιβλιοθήκη Media Framework περιέχει όλα τα codex που χρειάζονται τα πολυμέσα που έχει η συσκευή όπως αναπαραγωγή βίντεο ή ήχου. Όλες οι γραμματοσειρές που υποστηρίζει το Android βρίσκονται στην βιβλιοθήκη Free Type ενώ η βιβλιοθήκη WebKit περιέχει τον πυρήνα του web browser όπου είναι ανοιχτού λογισμικού και μια πιο ελαφριά εκδοχή Safari που χρησιμοποίει η Apple.

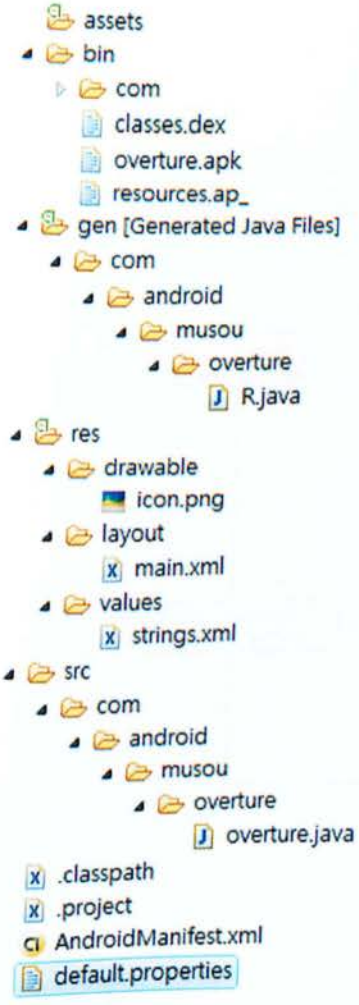
Το επόμενο στρώμα είναι το Android Runtime του οποίου πολύ σημαντικό κομμάτι είναι ο Dalvik Virtual Machine ο οποίος είναι σχεδιασμένος στα πρότυπα του ανοιχτού λογισμικού και έχει την ίδια λειτουργία που έχει και ο Java Virtual Machine στην Java . Το Runtime σχεδιάστηκε αποκλειστικά για το Android και είναι προσαρμοσμένο πάνω στις ανάγκες ενός κινητού τηλεφώνου, δηλαδή περιορισμένη μνήμη, περιορισμένη cpu και περιορισμένη μπαταρία. Ο Dalvik τρέχει αρχεία .dex τα οποία προέρχονται από την μεταγλώττιση αρχείων .class και .jar και τα συγκεκριμένα αρχεία είναι πολύ αποτελεσματικά σε συσκευές όπως τα κινητά τηλέφωνα μιας και απαιτείται πολύ μικρό ποσοστό μνήμης καθώς και πολύ μικρό ποσοστό της cpu για να εκτελεστούν. Επιπλέον ο Dalvik δίνει την δυνατότητα να έχουμε πολλαπλά στιγμιότυπά του, ενώ καθένα από αυτά μπορεί να τρέχει και μια διαφορετική διαδικασία πράγμα που μας δίνει την δυνατότητα του multitasking. Αυτό σε συνδυασμό με τα όσα αναφέρθησαν για τα αρχεία dex κάνουν το Android ένα πολύ φιλικό προς την συσκευή λειτουργικό σύστημα.

Ακριβώς πιο πάνω υπάρχει το Application Framework. Είναι ένα σύνολο εργαλείων γραμμένο αποκλειστικά σε Java και το χρησιμοποιούν τόσο οι εφαρμογές που υπάρχουν ήδη εγκατεστημένες στο τηλέφωνο, όπως για παράδειγμα η εφαρμογή για την πραγματοποίηση τηλεφωνικών κλήσεων, όσο και η εφαρμογές που μπορεί να κατεβάσει κανείς από το Android Market αλλά και να γράψει και ο ίδιος. Πιο αναλυτικά ο Activity Manager είναι υπεύθυνος για την διατήρηση του κύκλου ζωής της εφαρμογής καθώς και για τον σορό της κάθε εφαρμογής έτσι ώστε αυτές που εκτελούνται σε διαφορετικές διεργασίες να τρέχουν ομαλά και να μην καθυστερούν τον χρηστή στις επιλογές του. Το επόμενο που αξίζει να δούμε είναι ο Package Manager που είναι υπεύθυνος να καταγράφει τις εφαρμογές που είναι εγκατεστημένες στην συσκευή. Οι Content Providers είναι αποκλειστικά σχεδιασμένοι για το Android και είναι τα εργαλεία εκείνα που επιτρέπουν στις εφαρμογές να μοιράζονται τα δεδομένα τους με άλλες εφαρμογές. Η χρησιμότητα του συγκεκριμένου εργαλείου είναι πολύ χαρακτηριστική στον κατάλογο επαφών. Εκεί ο χρήστης πέρα απο τηλέφωνα μπορεί να αποθηκεύσει φωτογραφίες, διευθύνσεις, emails, να βάλει προσωπικούς ήχους κλήσης και με τους Content Providers αυτές οι πληροφορίες μπορούν να διαχειριστούν απο άλλες εφαρμογές όπως την εφαρμογή για τα emails, ή την εφαρμογή Google Maps. Ο Resource Manager είναι υπεύθυνος για την αποθήκευση μικρού όγκου δεδομένων (όχι βάση δεδομένων) όπως μερικές γραμμές κειμένου ή πιθανόν μερικές εικόνες που μπορεί να χρειάζεται κάποια εφαρμογή.

Τέλος στο τελευταίο επίπεδο είναι όλες οι εφαρμογές του κινητού. Ο web browser, η εφαρμογή για τις κλήσεις τηλέφωνου, η εφαρμογή για τα μηνύματα κλπ

Κεφάλαιο 11: Δομή μιας εφαρμογής Android

Κάθε εφαρμογή του Android είναι ένα σύνολο απο αρχεία και φακέλους τα όποια αφού γίνουν compiled παράγεται ένα αρχείο με κατάληξη .apk το οποίο στην συνέχεια εγκαθίσταται στο κινητό και περιέχει την εφαρμογή κωδικοποιημένη. Κάθε εφαρμογή ή αλλιώς Android Project έχει ένα μοναδικό όνομα και μπορεί να περιέχει πολλούς υποφακελους και πολλά αρχεία ανάλογα με την πολυπλοκότητα της εφαρμογής.



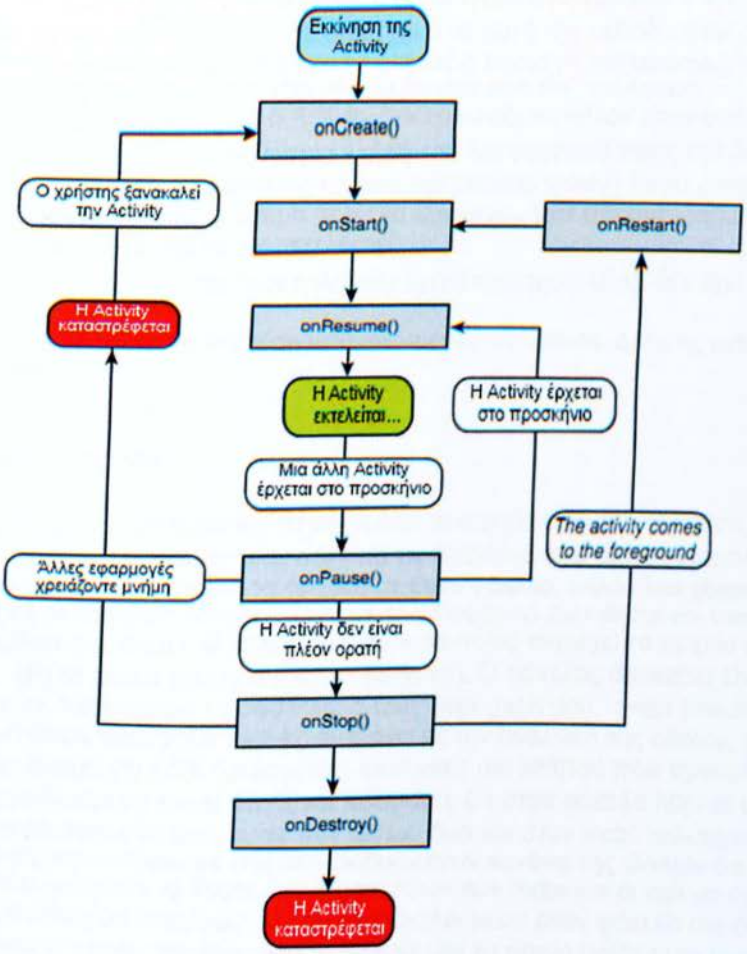
Σε κάθε project υπάρχει ένα αρχείο όπου περιέχει βασικές πληροφορίες που αφορούν την εφαρμογή. Το αρχείο αυτό ονομάζεται Application Manifest.xml και είναι δουλεία του προγραμματιστή να καταχωρίσει τις πληροφορίες με τέτοιο τρόπο ώστε να είναι χρήσιμες για το λειτουργικό σύστημα. Μιας και το Manifest είναι ένα xml αρχείο υπακούει στον τρόπο σύνταξης που έχουν όλα τα xml αρχεία. Μερικές απο τις πληροφορίες που έχει είναι η εξής:

- Το όνομα του πακέτου της εφαρμογής
- Το όνομα της εφαρμογής που εμφανίζεται στον χρηστή
- Την έκδοση των βιβλιοθηκών που χρησιμοποιούνται
- Τον αριθμό έκδοσης της εφαρμογής
- Τις άδειες που χρειάζεται η εφαρμογή απο το λειτουργικό σύστημα για να λειτουργήσει σωστά

Πέραν του Manifest που, όπως διαπιστώνουμε είναι πολύ σημαντικό για την εφαρμογή, υπάρχουν και αλλά δομικά μέρη στης εφαρμογής που αξίζει να σημειωθούν.

11.1 Ο φάκελος src

Ο φάκελος src (source) περιέχει αρχεία Java τα όποια είναι δομημένα μέσα σε ένα η περισσότερα πακέτα ανάλογα με το project. Ο συγκεκριμένος φάκελος είναι ουσιαστικά το μοναδικό σημείο της εφαρμογής όπου υπάρχει κώδικας. Κάποια απο τα αρχεία που βρίσκονται σε αυτόν τον φάκελο κάνουν extend μια κλάση που ονομάζεται activity, παίρνουν όλες τις ιδιότητες της και τα αποκαλούμε εν συντομία και αυτά activities ή αλλιώς δραστηριότητες. Η activity ουσιαστικά είναι ο κώδικας που τρέχει πίσω απο κάθε διαφορετική οθόνη στο κινητό. Η κάθε activity έχει έναν "κύκλο ζωής" που φαίνεται στο παρακάτω διάγραμμα.



- onCreate() - Η συγκεκριμένη μέθοδος καλείται μόλις δημιουργείται η Activity. Σε αυτή την μέθοδο γίνονται όλες οι αρχικοποιήσεις μεταβλητών που θα χρησιμοποιηθούν στην Activity, και δημιουργείται η οθόνη που θα προβληθεί στον χρηστή
- onStart() - Καλείται πριν γίνει ορατή η Activity στον χρηστή
- onResume() - Καλείται αφού πρώτα έχει προηγηθεί η onPause() και εκεί συνήθως ανακτώνται δεδομένα που αποθηκευτήκαν στην προσωρινή μνήμη.
- onPause() - Καλείται όταν το λειτουργικό πρόκειται να εκτελέσει την onResume() κάποιας άλλης Activity. Σε αυτή την μέθοδο συνήθως υπάρχει ο κώδικας για την αποθήκευση των δεδομένων προκειμένου να αξιοποιηθούν αργότερα με την χρήση της onResume(). Επιπλέον γίνεται παύση διεργασιών που καταναλώνουν πόρους απο την CPU καθώς και κάποια κινούμενα γραφικά στην οθόνη ή η χρήση του δεκτή GPS. Πολύ σημαντικό σε αυτή την μέθοδο είναι ότι πρέπει να ολοκληρωθεί πολύ γρήγορα γιατί αλλιώς η επομένη onResume() που είναι προς εκτέλεση θα αργήσει και έτσι όλο το λειτουργικό θα "κολλήσει".
- onStop() - Καλείται όταν η Activity δεν εμφανίζεται πλέον στην οθόνη.
- onDestroy() - Είναι η τελευταία κλήση του λειτουργικού προς την Activity. Αυτό είτε γιατί ο προγραμματιστής κάλεσε την μέθοδο finish() όπου καταστρέφει την Activity είτε γιατί το σύστημα θέλει να εξοικονομήσει πόρους για κάποια άλλη εφαρμογή που πρέπει να εκτελεστεί.
- onRestart() - Καλείται όταν η Activity έχει σταματήσει αλλά δεν έχει καταστραφεί.

Ο προγραμματιστής όμως δεν είναι υποχρεωμένος να καλέσει όλες τις μεθόδους πλην της onCreate().

11.2 Ο φάκελος res

Ο φάκελος res (resources) περιέχει όλα τα αρχεία εικόνας, κειμένου, xml layout, κλπ τα οποία χρησιμοποιούνται από τις Activities που βρίσκονται στον φάκελο src. Φυσικά δεν βρίσκονται όλα τα αρχεία, σε έναν φάκελο, αλλά είναι χωρισμένα και ταξινομημένα σε υποφακέλους ανάλογα με το είδος τους. Συνηθισμένοι υποφάκελοι του κύριου φακέλου res, είναι ο φάκελος drawable ο οποίος περιέχει τα αρχεία εικόνας (.png, .jpg, .gif) τα οποία χρησιμοποιεί η εφαρμογή. Ο φάκελος drawable είναι χωρισμένος σε 3 επιμέρους υποφακελους τους hdpi (high dpi), mdpi (medium dpi) και ldpi (low dpi) όπου περιέχουν εικόνες ανάλογα με την ανάλυση της οθόνης (οι οποίες έχουν το ίδιο όνομα για κάθε διαφορετική ανάλυση) του κινητού που προορίζονται όπως είδαμε σε προηγούμενο κεφάλαιο. Είναι προφανές ότι στον φάκελο hdpi οι εικόνες έχουν καλύτερη ανάλυση από αυτές των άλλων δυο και στον mdpi καλύτερη απο τον ldpi. Οι εικόνες που υπάρχουν εκεί ακολουθούν έναν κανόνα της Google ότι οι hdpi εικόνες πρέπει να έχουν το 150% των διαστάσεων των mdpi και οι ldpi να έχουν το 75% των mdpi. Εν συνέχεια υπάρχουν και άλλοι φάκελοι μέσα στον φάκελο res όπως ο φάκελος layout ο οποίος περιέχει όλα τα αρχεία xml τα οποία ορίζουν τα layouts που υπάρχουν στην εφαρμογή (θα αναλυθούν λεπτομερέστερα σε επόμενη ενότητα), και τέλος ο φάκελος values στον οποίο αποθηκεύονται τα διάφορα κείμενα , κωδικοί χρωμάτων κλπ τα όποια χρησιμοποίει η εφαρμογή.

11.3 Βασικά κομμάτια μιας εφαρμογής

Περά από τις activities υπάρχουν και αλλά βασικά κομμάτια μιας εφαρμογής.

- Προθέσεις (Intents) – Οι Activities επικοινωνούν και εναλλάσσουν την λειτουργία τους μέσω των Intents. Ουσιαστικά τα Intents εξασφαλίζουν την μετάβαση από την μία Activity σε μια άλλη και επίσης χρησιμοποιούνται για ανταλλαγή δεδομένων. Η ανταλλαγή δεδομένων, μπορεί να γίνει είτε μεταξύ των Activities μιας εφαρμογής, είτε από τη μία εφαρμογή στην άλλη. Παραδείγματος χάρη μπορούμε μέσω ενός Intent να εκκινήσουμε έναν web browser ώστε να μας ανοίξει απευθείας ένα url το οποίο έχουμε παρέχει εμείς μέσω ενός Intent.
- Υπηρεσίες (Services) – Πρόκειται για λειτουργίες της εφαρμογής οι οποίες είναι σχεδιασμένες να τρέχουν στο παρασκήνιο και να επιστρέφουν αποτελέσματά ακόμη και όταν η εφαρμογή δεν είναι στο προσκήνιο. Πχ μια εφαρμογή media player μπορεί μέσω μιας υπηρεσίας να συνεχίσει να παίζει μουσική ακόμη και όταν το κύριο παράθυρο της εφαρμογής δεν βρίσκεται στο προσκήνιο.
- Πάροχος Περιεχόμενου (Content Provider) - Η ανταλλαγή δεδομένων από μια εφαρμογή στην άλλη όπως είπαμε παραπάνω μπορεί να γίνει μέσω ενός Intent, ένας πάροχος περιεχομένου όμως έχει πιο σύνθετη λειτουργία. Οι content providers μιας εφαρμογής διαχειρίζονται συγκεκριμένα δεδομένα της εφαρμογής τα οποία έχει ορίσει ο προγραμματιστής κατά την κατασκευή του προγράμματος. Συνηθισμένα δεδομένα τα οποία μοιράζονται μέσω Content Providers, είναι οι βάσεις δεδομένων SQLite μιας εφαρμογής, και οι επαφές του χρήστη.
- Δέκτες Μετάδοσης (Broadcast Receivers) – Πρόκειται για ένα είδος υπηρεσίας η οποία αντιλαμβάνεται κάποια γεγονότα του συστήματος και αναλαμβάνει να ενημερώσει το σύστημα ή τις υπόλοιπες εφαρμογές. Ο σκοπός τους είναι διπλός καθότι μπορούν και να ενημερωθούν για κάποιο συμβάν από άλλες εφαρμογές, αλλά και να ειδοποιήσουν τις υπόλοιπες εφαρμογές και το σύστημα για κάποιο συμβάν που τις ενεργοποίησε. Δεν έχουν γραφικό περιβάλλον αλλά μπορούν να προβάλουν ειδοποίηση στον χρήστη μέσω της μπάρας ειδοποιήσεων. Συνήθως χρησιμοποιούνται ως διαμεσολαβητές μεταξύ των Activities και των Services μιας εφαρμογής.

Κεφάλαιο 12: Περιβάλλον Χρήστη

Ένα βασικό κομμάτι του Android είναι σίγουρα το περιβάλλον του χρηστή ή αλλιώς User Interface (UI). Είναι ουσιαστικά αυτό που βλέπει ο χρήστης την ώρα που εκτελείται η εφαρμογή και με το οποίο αλληλεπιδρά. Είναι σίγουρα ένας βασικός παράγοντας που συντελεί στην επιτυχία μιας εφαρμογής μαζί άλλωστε με την βασική ιδέα της εφαρμογής αλλά και τον τρόπο που έχει υλοποιηθεί. Το UI του Android απαρτίζεται από δυο βασικά στοιχειά.

- Το Layout
- Τα Widgets

Το UI μπορεί να σχεδιαστεί με δυο τρόπους. Ο πιο εύκολος είναι από μια παλέτα που βρίσκεται στα αριστερά της οθόνης σχεδίασης και ο προγραμματιστής μπορεί να επιλέξει και να εισάγει στον κατάλληλα διαμορφωμένο χώρο τα αντικείμενα που θέλει να έχει το UI. Όπως προαναφέρθηκε το κάθε Layout με τα Widgets που περιέχει είναι ουσιαστικά ένα αρχείο xml άρα κατά την σχεδίαση και προσθήκη οποιουδήποτε στοιχείου το αρχείο xml ανανεώνεται ανάλογα με τις αλλαγές που κάνει ο χρήστης (δημιουργείται ο κώδικας xml). Ο δεύτερος τρόπος σχεδίασης και πολύ πιο αργός είναι ο χρήστης να γράψει κατευθείαν των κώδικα xml που αντιστοιχεί σε κάθε στοιχείο του UI. Αυτός ο τρόπος προτιμάται μόνο για παραμετροποίηση των στοιχείων και όχι για τον βασικό σχεδιασμό.

12.1 Layout

Το Layout είναι ένα νοητό παραλληλόγραμμο το όποιο έχει τις διαστάσεις της οθόνης στην όποια αναφέρεται. Εκτός των ορίων του παραλληλογράμμου ο προγραμματιστής δεν μπορεί να σχεδιάσει τίποτα. Ένα Layout μπορεί να περιέχει και άλλα Layouts στο εσωτερικό του σε διάφορες διαστάσεις χωρίς όμως να ξεφεύγει από το βασικό το όποιο ονομάζεται και Parent Layout.

Υπάρχουν πολλών ειδών διαφορετικά Layouts τα όποια διαφέρουν στην διάταξη των αντικείμενων που θα έχουν στο εσωτερικό τους. Το γραμμικό ή αλλιώς Linear ορίζει τα αντικείμενα είτε σε μια νοητή κάθετη ή οριζόντια γραμμή. Το σχετικό ή αλλιώς Relative ορίζει τα αντικείμενα συσχετίζοντας τις διαστάσεις τους με άλλα αντικείμενα που υπάρχουν ή με τα όρια του Parent Layout. Το Table Layout έχει νοητές γραμμές και στήλες, όπως ακριβώς ένας πινάκας, και εκεί τοποθετούνται τα διάφορα στοιχειά. Είναι αρκετά εύχρηστο σε εφαρμογές οπού υπάρχουν πολλά στοιχειά τα οποία πρέπει

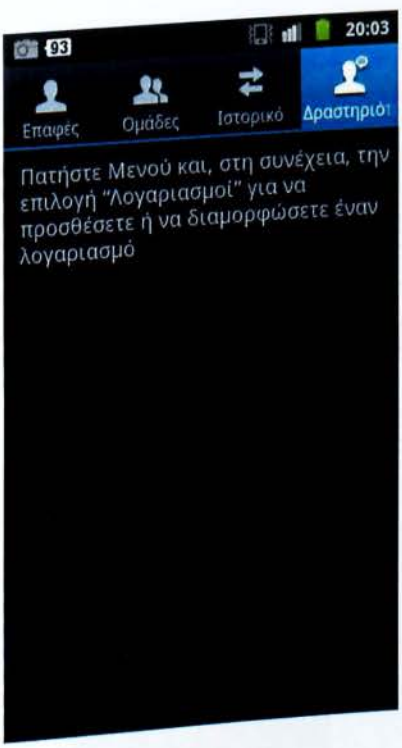
να στοιχηθούν οριζόντια και κάθετα όπως το πληκτρολόγιο από μια αριθμομηχανή. Στο παράδειγμα της εικόνας που ακολουθεί παρουσιάζεται είναι αρκετά σύνθετο που περιέχει πολλά Widgets τα όποια διαθέτει το Android.

12.2 Widgets

Τα Widgets είναι τα επιμέρους αντικείμενα που βρίσκονται μέσα στο Layout και χρησιμοποιούνται για να αλληλεπιδρά ο χρήστης με την εφαρμογή και καθένα από τα αυτά έχει και διαφορετικές ιδιότητες. Χωρίζονται στις εξής κατηγορίες

12.2.1 Text View

- TextView – Είναι ο πιο απλός τρόπος για εμφάνιση κειμένου το οποίο είναι απλώς για ανάγνωση (read only) από τον χρήστη .

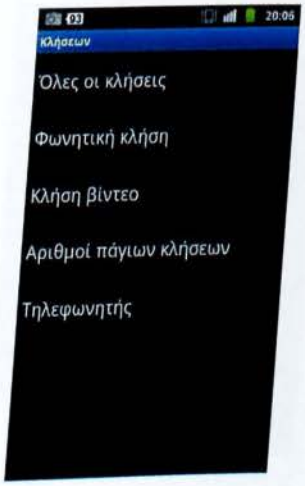


12.2.2 Edit Text

- Edit Text – Είναι ο χώρος στον οποίο ο χρήστης μπορεί να εισάγει κείμενο ή αριθμούς.

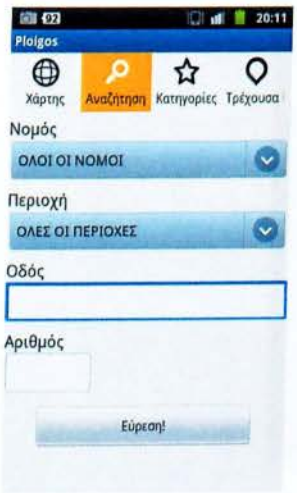
12.2.3 ListView

- ListView – Είναι ένα αντικείμενο το οποίο ταξινομεί έναν πίνακα από λέξεις την μια κάτω από την άλλη σε μορφή λίστας. Στην πιο απλή μορφή κάθε λέξη αναπαριστάται από ένα TextView το οποίο ο χρήστης μπορεί να επιλέξει και να μεταβεί σε μια άλλη οθόνη / Activity. Σε πιο σύνθετη μορφή μπορεί να σχεδιαστεί ένα καινούριο layout που μπορεί να έχει μια εικόνα, διπλά σε αυτή ένα text view και απο κάτω ένα edit text. Στην συνεχεία αυτό το layout μπορεί να αντικαταστήσει τα στοιχειά της λίστας.



12.2.4 Spinner

- Spinner – Είναι μια λίστα από TextView τα οποία εμφανίζονται σε μορφή drop down list την πρώτη φορά που ο χρήστης το επιλεγεί και μόλις ο χρήστης διαλέξει ένα στοιχειό της λίστας αυτή αποκρύπτεται και εμφανίζεται μόνο η επιλογή του χρηστή.

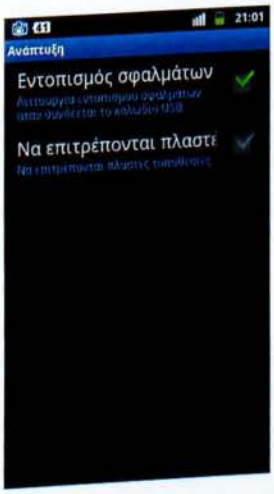


12.2.5 Button

- Button – Ένα απλό κουμπί.

12.2.6 CheckBox

- CheckBox – Ένα κουμπί δυο καταστάσεων το οποίο εμφανίζεται σαν ένα κουτί που περιέχει το σύμβολο του "τσεκ"

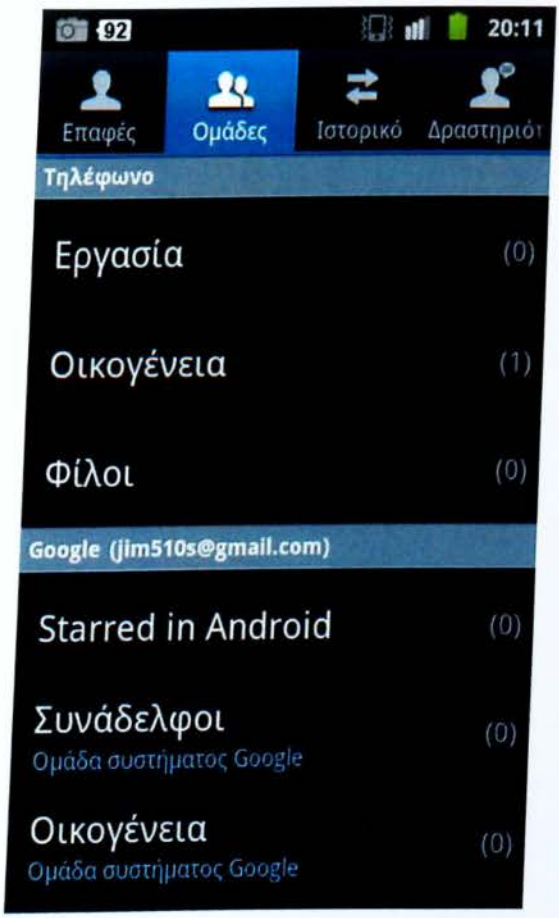


12.2.7 Radio Button

- Radio Button – Ένα κουμπί δυο καταστάσεων με στρογγυλό σχήμα που όταν είναι ενεργό ένας μικρότερος κύκλος στο εσωτερικό του αλλάζει χρώμα.

12.2.8 Tab Widget

- Tab Widget - Είναι ακριβώς όπως και στους web browsers που έχουμε στον υπολογιστή μας. Χρησιμοποιούνται για να χωρίσουμε σε κατηγορίες διάφορα Layouts. Κάθε καρτέλα μπορεί να είναι μια διαφορετική Activity με το δικό της Layout πράγμα που δίνει πληθώρα επίλογων στο πως θα δομηθεί η όλη εφαρμογή. Μπορεί να παραμετροποιηθεί σε μεγάλο βαθμό και δεν είναι λίγες οι περιπτώσεις που τα αποτελέσματα είναι πολύ καλά. Τις περισσότερες φορές επιλέγετε να μπει στο πάνω μέρος, όπως ακριβώς και τους web browsers, αν και υπάρχουν περιπτώσεις όπου βρίσκεται στο κάτω μέρος της οθόνης όπως στο phone. Μπορεί να έχει σαν τίτλο η κάθε καρτέλα ένα απλό κείμενο, ένα εικονίδιο ή και τα δυο μαζί.



Κάθε διαφορετική οθόνη μίας εφαρμογής (λέγεται και αυτή Layout για συντομία) είναι ένα σύνολο από τα παραπάνω. Όσο πολύπλοκη και αν φαίνεται στο μάτι μπορεί να χωριστεί στα βασικά προαναφερθέντα Layout ή Widgets. Προφανώς καθένα από αυτά μπορεί να παραμετροποιηθεί σε μεγάλο βαθμό ανάλογα με την φαντασία του προγραμματιστή.

Αφού ο προγραμματιστής κατασκευάσει με τα παραπάνω εργαλεία το Layout που θα αντιστοιχεί σε μια Activity πρέπει να το δηλώσει μέσα στον κώδικα της συγκεκριμένης Activity άλλα και να δηλώσει και τα Widgets τα οποία προορίζονται για την αλληλεπίδραση του χρηστή με την εφαρμογή. Κάθε Widget αλλά και Layout μέσα στο σύνολο των ιδιοτήτων του έχει και ένα πεδίο που ονομάζεται id. Όσο αναφορά το Layout απλά δηλώνεται ποιο αρχείο xml αντιπροσωπεύει την Activity. Τα Widgets από την άλλη λειτουργούν με διαφορετικό τρόπο. Για κάθε Widget υπάρχει μια κλάση που το αντιπροσωπεύει και για να γίνει η αντιστοίχιση ο προγραμματιστής δημιουργεί ένα αντικείμενο από κάθε διαφορετική κλάση ανάλογα με τα διαφορετικά Widgets που θέλει να δηλώσει. Έπειτα αντιστοιχεί κάθε μεταβλητή με το id του αντίστοιχου Widget.

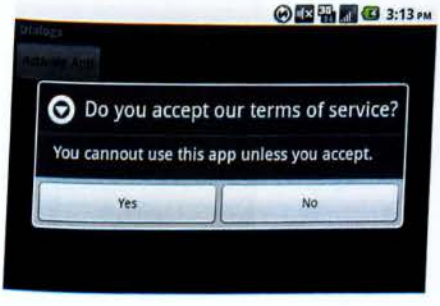
Κεφάλαιο 13: Options & Context menus, Toasts

Ένα βασικό στοιχειό του UI στο Android είναι τα Options Menus, τα Context Menus, τα Dialog Boxes και τα Toasts.

13.1 Options Menu

Το Options Menu είναι ένα μενού που εμφανίζεται στο κάτω μέρος της οθόνης πατώντας το Options Button που αναφέρθηκε σε προηγούμενη ενότητα. Εκεί ο προγραμματιστής μπορεί να βάλει διαφορές επιλογές οι οποίες κρίνει ότι δεν είναι τόσο σημαντικές για την βασική οθόνη και ενδέχεται να αποπροσανατολίζουν τον χρηστή από τον βασικό σκοπό της εφαρμογής. Το Options Menu δεν μπορεί να παραμετροποιηθεί όσο αναφορά την εμφάνιση του. Απλώς μπορούν να προστεθούν όσες επιλογές θέλει ο χρήστης. Καλό είναι πάντως να μην ξεπερνάνε οι επιλογές τις έξι γιατί μετά γίνεται δύσχρηστη η εφαρμογή. Ο τρόπος που κατασκευάζεται το Options Menu είναι μέσω ενός αρχείου xml όπου δηλώνονται οι επικεφαλίδες κάθε επιλογής και αν θα υπάρχει κάποιο εικονίδιο. Μετά μέσα από τον κώδικα της java και στην αντίστοιχη κλάση όπου θα εμφανίζεται το μενού καλούνται δυο μέθοδοι. Η μια αρχικοποιεί το μενού περνώντας σαν όρισμα το αρχείο xml που προαναφέρθηκε και η άλλη περιέχει τον κώδικα για τις διαφορετικές επιλογές που θα έχει το μενού. Για την υλοποίηση αυτή χρησιμοποιείται η "case" η όποια στο τέλος κάθε περίπτωσης πρέπει να έχει "break" για να μην εκτελούντα οι περιπτώσεις που ο κώδικας τους είναι μετά από αυτό που διάλεξε ο χρήστης.

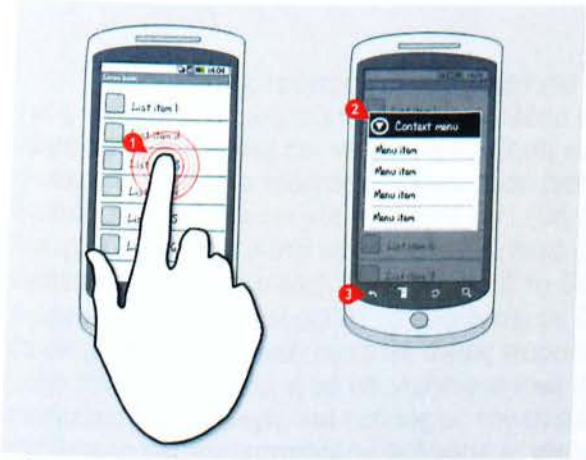
13.2 Dialog Box



Το Dialog Box ανήκει στην κατηγορία των Alerts όπως και το Toast. Το Dialog Box είναι ένα παράθυρο που εμφανίζεται με ένα σύντομο μήνυμα και συνήθως κάποιες επιλογές που δεν ξεπερνούν τις τέσσερις. Ένα χαρακτηριστικό παράδειγμα είναι στα παιχνίδια που εμφανίζεται το μήνυμα "Κάνατε σκορ 80/100. Θέλετε να επαναλάβετε το επίπεδο? " και έχει από κάτω δυο κουμπιά ναι ή όχι.

Για να εμφανιστεί καλείται η μέθοδος create και εκεί ορίζεται το τίτλος, ένα μικρό κείμενο, μια φωτογραφία και το ποσά κουμπιά θα έχει.
Εμφανίζεται στο κέντρο της οθόνης με το εφέ fade-in/fade-out και αν και μπορεί να παραμετροποιηθεί η θέση της εικόνας η γραμματοσειρά του κειμένου δεν προτιμάται.

13.3 Context Menu



Τα Context Menus ορίζονται με τον ίδιο τρόπο που ορίζονται και τα Option Menus μόνο που η συνάρτηση στην java που τα αρχικοποιεί είναι διαφορετική και ανάλογη για τα Contexts. Τα Context Menus μπορούν να παρομοιαστούν με το δεξί κλικ του ποντικιού στον υπολογιστή. Για να εμφανιστεί ο χρήστης απλώς κρατεί πατημένο ένα στοιχειό του οποίο στο Widget έχει οριστεί το Context Menu. Κυρίως χρησιμοποιούνται σε λίστες. Έτσι αν υπάρχει μια λίστα σε μια μεταβλητή list και θέσουμε εκεί ένα Context Menu τότε κρατώντας πατημένο κάθε στοιχειό της λίστας θα εμφανίζεται το μενού που έχουμε ορίσει.

13.4 Toast

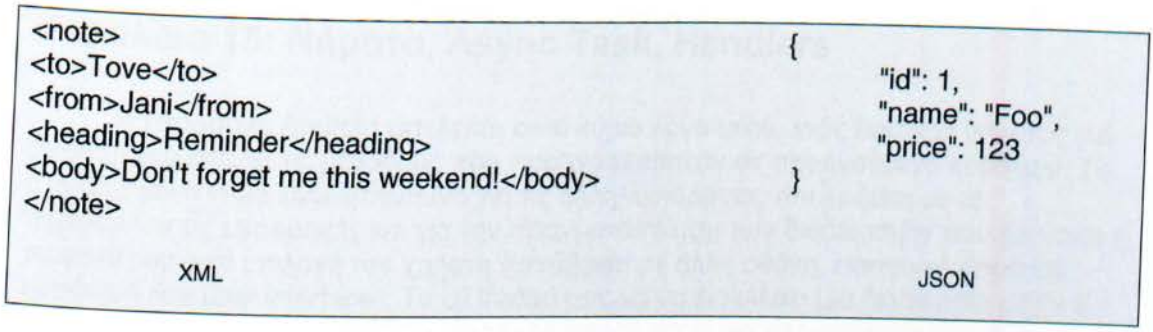
Τέλος τα Toasts είναι ο πιο συνηθισμένος τρόπος να εμφανίζει ο προγραμματιστής στον χρηστή μηνύματα περιορισμένης διάρκειας για να τον ενημερώσει αν ενεργοποιήθηκε το GPS ή το 3G ή αν κατέβηκε ένα αρχείο. Συνήθως εμφανίζεται κοντά στο κάτω μέρος της οθόνης και έχει διάρκεια από ένα μέχρι πέντε δευτερόλεπτα. Η εμφάνιση του είναι πολύ απλή αν και μπορεί να παραμετροποιηθεί πράγμα που δεν το προτιμάει κανείς μιας και δεν αξίζει τον κόπο αφού το μήνυμα είναι περιορισμένου χρόνου. Για την εμφάνιση του αρκεί μια γραμμή κώδικα πράγμα που το κάνει και πολύ καλό εργαλείο για debug.

Κεφάλαιο 14: Υπηρεσίες Δικτύου

Μια εφαρμογή Android μπορεί να γίνει πολύ χρήσιμη για τον χρήστη αν μπορεί να συνδεθεί στο internet και να ανταλλάξει δεδομένα με έναν server ή αν μπορεί να εμφανίσει το περιεχόμενο των σελίδων. Με την άνθηση των μέσων κοινωνικής δικτύωσης των υπηρεσιών για δωρεάν internet στο κινητό τηλέφωνων απο τους παροχους αλλά και τα όλο και περισσότερα μέρη με wi-fi όλο και περισσότερες είναι οι εφαρμογές που κάνουν χρήση του internet. Τις περισσότερες φορές αυτές οι εφαρμογές είναι που έχουν την μεγαλύτερη επισκεψημοτητα από τους χρηστές.

Ένας απλός τρόπος για να εμφανιστεί το περιεχόμενο μιας σελίδας είναι η χρήση ενός widget που ονομάζεται webview. Μέσα σε αυτό το widget μπορούν να εμφανιστούν σελίδες και να γίνει πλοήγηση κανονικά όπως και στον web browser της συσκευής. Αυτός ο τρόπος δεν είναι τόσο διαδεδομένος μιας και είναι ουσιαστικά ένας δεύτερος web browser και το κινητό έχει ήδη έναν. Δεν υπάρχει λοιπόν λόγος για την ύπαρξη δευτέρου. Αυτό που είναι σημαντικό είναι η λήψη δεδομένων όπως κείμενο, εικόνες, αρχεία ήχου κλπ. Έχοντας αυτά τα δεδομένα χωρίς όλη την σελίδα ο προγραμματιστής μπορεί να τα εμφανίσει σε ένα ωραίο UI και ο χρήστης να μπορεί να τα διαβάσει, να τα επεξεργαστεί, όπως παραδείγματος χάριν να αποθηκεύσει κάτι απο αυτά στο ημερολόγιο, ή να στείλει ένα e-mail και τέλος ακόμα και να "ανεβάσει", αν το επιτρέπει η εφαρμογή, και εκείνος με την σειρά του δεδομένα. Αυτές οι δυο διαδικασίες της λήψης και της αποστολής δεδομένων γίνονται με την βοήθεια δυο πρωτοκόλλων. Του http GET και του http POST. Με το http GET ή αλλιώς για συντομία GET η εφαρμογή λαμβάνει δεδομένα τύπου XML (eXtensible Markup Language) ή JSON (JavaScript Object Notation) ενώ με το http POST ή αλλιώς POST η εφαρμογή στέλνει δεδομένα σε έναν server.

Η XML είναι μια πολύ απλή στην ανάγνωση, τόσο από τον άνθρωπο όσο και από την μηχανή, κωδικοποίηση δεδομένων που χρησιμοποιείται πολύ συχνά στο internet. Υπάρχουν τα tags που στο παράδειγμα που ακολουθεί είναι οι λέξεις που βρίσκονται μέσα σε "<>" και τα values που είναι το κείμενο που είναι μεταξύ δυο tags. Κάθε tag πρέπει να σηματοδοτεί το τέλος του και αυτό γίνεται βάζοντας το σύμβολο "/". Το JSON είναι και αυτό μια κωδικοποίηση για την αποστολή και λήψη δεδομένων με σημαντική διαφορά τον χρόνο που χρειάζεται να γίνει η αποκωδικοποίηση των δεδομένων από τον client που στην προκειμένη περίπτωση είναι η εφαρμογή του κινητού. Πολλές φορές αυτό που καθορίζει αν τα δεδομένα που θα λάβει μια εφαρμογή ή που θα στείλει θα είναι σε XML ή JSON είναι σε τι μορφή υπάρχουν στον server από όπου θα τα πάρει η εφαρμογή. Έτσι λοιπόν το Android έχει εργαλεία και μεθόδους για την αποστολή και λήψη και για XML και για JSON.



Πολλές φορές για την καλύτερη ταξινόμηση των δεδομένων κατά την αποστολή του από τον server στον client δημιουργείται ένας πινάκας απο JSON Objects και ονομάζεται JSON Array. Για παράδειγμα κάθε αντικείμενο JSON μπορεί να περιέχει πληροφορίες για φοιτητές μιας τάξης και ο πινάκας να αναπαριστά όλη την τάξη. Αντίστοιχα πολλοί πινάκες μπορούν να αναπαραστήσουν ένα σχολείο. Γι' αυτό τον λόγο και το Android διαθέτει μεθόδους που διευκολύνουν τον προγραμματιστή στην αποθήκευση των δεδομένων που χρειάζεται, είτε αυτά είναι όλο το αντικείμενο JSON, είτε ορισμένα από τα attributes. Η αποθήκευση αυτή γίνεται με την χρήση του πρωτοκόλλου HTTP και από αυτό της μεθόδου GET. Η εφαρμογή "ζητάει" μέσο μιας διεύθυνσης από τον server ένα αρχείο (το όποιο είναι το JSON) και κατόπιν μέσο ενός stream reader επιστρέφει σε μια μεταβλητή τον πινάκα ή το αντικείμενο JSON. Από εκεί και πέρα τα δεδομένα είναι σε μια μεταβλητή και μπορούν εύκολα να διαχειριστούν περεταίρω.

Κεφάλαιο 15: Νήματα, Async Task, Handlers

Η εφαρμογή Android εκτελείται κατά κύριο λόγο μέσω ενός βασικού νήματος (UI thread) που εκτελεί τις μεθόδους που προαναφέρθηκαν σε προηγούμενο κεφάλαιο. Το συγκεκριμένο νήμα είναι υπεύθυνο για τις αλληλεπιδράσεις του χρήστη με το περιβάλλον τις εφαρμογής και για την πραγματοποίηση των διαδικασιών που ξεκινούν ή τερματίζουν από επιλογή του χρήστη (μετάβαση σε άλλη οθόνη, εισαγωγή κειμένου, εμφάνιση του user interface). Το UI thread μπορεί να εκτελέσει μια διαδικασία μέχρι 5 δευτερόλεπτα. Αν μια διαδικασία απασχολήσει το νήμα για περισσότερο χρόνο η εφαρμογή σταματήσει για να μην δημιουργηθεί πρόβλημα σε ολόκληρο το λειτουργικό σύστημα. Όπως όμως σε κάθε λειτουργικό σύστημα είναι δυνατόν να υπάρχουν ταυτόχρονα πάνω από ένα νήματα έτσι ακριβώς και στο Android. Η διάφορα είναι ότι στο Android οι διαδικασίες του παρασκηνίου μπορούν να εκτελεστούν με τρεις βασικούς τρόπους:

- Threads
- Handlers
- Async Task

Αρχικά τα νήματα , ή αλλιώς threads, που χρησιμοποίει το Android είναι ακριβώς τα ιδία που έχει και η Java. Χρησιμοποιούνται για την πραγματοποίηση διεργασιών στο παρασκήνιο που απαιτούν χρονικό διάστημα μεγαλύτερο από κάποια δευτερόλεπτα και το βασικό τους μειονέκτημα είναι ότι δεν μπορούν να αλληλεπιδράσουν με το UI thread και συνεπώς να εμφανίσουν κάποιο μήνυμα στον χρήστη της εφαρμογής.

Οι handlers και το Async Task χρησιμοποιούν νήματα για να πραγματοποιήσουν τις διεργασίες. Άρα λοιπόν ότι μπορεί να γίνει με handlers ή sync task μπορεί να γίνει και με νήματα. Η σημαντική διαφορά και η ευκολία που προσφέρουν αυτοί οι δυο τρόποι είναι η επικοινωνία μεταξύ του καλούντος νήματος και του νήματος που πραγματοποιεί την διεργασία. Βεβαία όπως είπαμε η διαδικασία αυτή μπορεί να γίνει και με άλλους τρόπους που όμως δεν είναι καθόλου ασφαλείς. Η διάφορα τώρα μεταξύ handlers και sync task είναι ότι με το sync task γίνεται σωστή και εύκολη χρήση του καλούντος νήματος. Η κλάση async task μέσω των μεθόδων της επιτρέπει την εκτέλεση διεργασιών από νήματα και την αποστολή των αποτελεσμάτων των διεργασιών στο νήμα από όπου έγινε η κλήση χωρίς να υπάρχει φόβος για συγχρονισμό ή αλλά προβλήματα ασφάλειας που μπορεί να προκύψουν. Κάποιες φορές το νήμα που κάνει την κλήση είναι το UI και άλλες όχι. Το async task χρησιμοποιείται κυρίως όταν το καλών νήμα δεν είναι το UI αλλά παρόλα αυτά πρέπει να επιστραφεί το αποτέλεσμα της διεργασίας ή μετά την εκτέλεση της διεργασίας για ξεκινήσει μια άλλη. Χαρακτηριστικό παράδειγμα είναι η λήψη δεδομένων μόλις τελειώσει με επιτυχία η αποθήκευση τους σε μια βάση δεδομένων ή η πραγματοποίηση πράξεων αν πρόκειται για αριθμούς .Αντιθέτως οι handlers χρησιμοποιούνται κυρίως για την ενημέρωση του UI Thread κατά την διάρκεια εκτέλεσης μιας διεργασίας στο παρασκήνιο. Τέτοιο παράδειγμα είναι η ενημέρωση μιας progress bar ή σε μια εφαρμογή ανταλλαγής μηνυμάτων μέσω διαδικτύου ένας handler μπορεί να διαχειρίζεται τα μηνύματα που εμφανίζονται στην

οθόνη χωρίς να επηρεάζεται η ταχύτητα αποστολής και λήψης. Έτσι αν η σύνδεση δεν είναι αρκετά γρήγορη και το μήνυμα καθυστερεί να φτάσει στον χρηστή ο handler θα το εμφανίσει μόλις αυτό φτάσει χωρίς να επηρεάσει την εφαρμογή.

Αυτό που γίνεται εύκολα αντιληπτό είναι ότι επειδή υπάρχουν πολλές ιδιομορφίες στον τρόπο που θα εκτελεστεί μια διεργασία από την εφαρμογή για αυτόν τον λόγο και υπάρχουν τριών ειδών διαφορετικές κατηγορίες από νήματα που όμως σε κάποια σημεία μοιάζουν μεταξύ τους. Αυτό που τα διαφοροποιεί είναι κυρίως οι μέθοδοι που μπορούν να εκτελέσουν και ανάλογα με την περίσταση χρησιμοποιείται και διαφορετική μέθοδος απο τον προγραμματιστή και άρα λοιπόν και διαφορετικού τύπου νήμα. Πάντως πολλές διαδικασίες μπορούν να εκτελεστούν και με τα τρία νήματα. Άλλοτε πιο εύκολα και άλλοτε πιο δύσκολα.

Κεφάλαιο 16: Η εφαρμογή

Η εφαρμογή που φτιάξαμε για το Android είναι ένας client που συνδέεται στο server μας και από εκεί παίρνει δεδομένα που προηγουμένως ο server έχει πάρει από διάφορες ιστοσελίδες που προϋπάρχουν ή που έχει εισάγει ο χρήστης. Το πλεονέκτημα της εφαρμογής είναι ότι ο χρήστης μπορεί να έχει πρόσβαση σε ειδήσεις και νέα από όλο τον κόσμο και αν συνδεθεί με τον προσωπικό λογαριασμό του μπορεί να προσθέσει επιπλέον πήγες κάτι που δεν υπάρχει σε άλλες αντίστοιχες εφαρμογές. Αξίζει να σημειωθεί ότι στην εφαρμογή χρησιμοποιήσαμε εξολοκλήρου παραμετροποιημένα γραφικά στοιχειά έτσι ώστε η εφαρμογή να είναι ευχάριστη στον χρηστή. Επιπλέον κάθε κλήση στον sever γίνεται μέσω threads και εμφανίζοντας ταυτόχρονα σχετικό μήνυμα έτσι ώστε ο χρήστης να ξέρει κάθε στιγμή ότι εκτελείται κάποια ενεργεία.

Τα βασικά χαρακτηριστικά της εφαρμογής είναι το UI όπου είναι απόλυτα παραμετροποιημένο στα χρώματα και το ύφος της εφαρμογής, η απλή και κατανοητή λειτουργία χωρίς να κουράζεται ο χρήστης με άσκοπες πληροφορίες, η ευκολία πρόσβασης στην πληροφορία με όσο το δυνατόν λιγότερες ενέργειες και τέλος η ομαλή λειτουργιά της εφαρμογής με την χρήση threads χωρίς να επιβαρύνεται η λειτούργει της συσκευής.

16.1 Αρχική Οθόνη

Κατά την εκκίνηση της εφαρμογής εμφανίζεται η οθόνη όπου ο χρήστης μπορεί να συνδεθεί με το προσωπικό του λογαριασμό. Επιπλέον έχει την δυνατότητα να μεταβεί στην σελίδα του server μέσω του web browser της συσκευής και να δημιουργήσει έναν λογαριασμό. Η τρίτη δυνατότητα που παρέχεται στον χρηστή είναι να παρακάμψει την οθόνη αυτή και να πάει κατευθείαν στις προϋπάρχουσες κατηγορίες.

16.2 Προϋπάρχουσες Κατηγορίες

Η οθόνη με τις προϋπάρχουσες κατηγορίες είναι μια λίστα με τους τίτλους των διαφόρων κατηγοριών. Αυτή η λίστα μεταβάλλεται δυναμικά μιας και κάθε φορά για την εμφάνιση της γίνεται λήψη των δεδομένων από τον server. Πατώντας σε μια κατηγορία, πάλι μέσω thread, γίνεται λήψη των δημοσιεύσεων που έχει αυτή κατηγορία και πιο συγκεκριμένα του τίτλους και μια περίληψη του άρθρου. Η μορφή των δεδομένων είναι και αυτή σε μορφή λίστας. Εάν ο χρήστης επιθυμεί μπορεί να πατήσει σε ένα στοιχειό της λίστας και αυτόματα μέσω του browser θα μεταβεί στο κανονικό άρθρο στο διαδίκτυο. Πατώντας το κουμπί menu στην συσκευή του και όντας στην οθόνη με τις προϋπάρχουσες κατηγορίες δίνεται στον χρηστή η επιλογή να μεταβεί στην αρχική οθόνη για να συνδεθεί στον προσωπικό του λογαριασμό.

Σε περίπτωση που ο χρήστης έχει έναν λογαριασμό και συνδεθεί πρώτα γίνεται η ταυτοποίηση με την βάση δεδομένων και στην συνεχεία γίνεται λήψη των κατηγοριών που έχει προσθέσει ο χρήστης. Η ταυτοποίηση πραγματοποιείται κάνοντας ένα post request στον server και η απάντηση είναι ένα json όπου μέσα υπάρχει το id του χρηστή του όποιο χρησιμοποιείται στην συνεχεία για την λήψη των δεδομένων.

16.3 Κατηγορίες Χρήστη

Η οθόνη που ακλουθεί είναι ιδία ουσιαστικά με την οθόνη των κατηγοριών που προϋπάρχουν μόνο που υπάρχουν διαφορετικές έγγραφες. Όπως και η επιλογή μιας κατηγορίας είναι ακριβώς ιδία με πριν. Το μόνο που αλλάζει είναι στο κουμπί menu όπου ο χρήστης μπορεί να αποσυνδεθεί, οπότε επανέρχεται η αρχική οθόνη, ή να δει τις προϋπάρχουσες κατηγορίες όντας συνδεδεμένος χωρίς να αποσυνδεθεί.

16.4 Χρήση του API

Όπως γίνεται αντιληπτό και είναι και προφανές μιας και μιλάμε για έναν client κυρίαρχο ρολό παίζει η επικοινωνία με τον server. Αυτή γίνεται με αρχεία JSON μιας και είναι πιο γρήγορα στην μεταφορά αλλά και στην ανάγνωση και αποστολή από την εφαρμογή σε σχέση με τα XML. Ο έλεγχος του ονόματος και του κωδικού χρήστη γίνεται με την βοήθεια ενός API στον server που αφού δεχτεί ένα JSON με τον κωδικό και το όνομα χρήστη που έχει πάρει από την φόρμα απατάει με ένα άλλο JSON όπου περιέχει πληροφορίες για το αν ολοκληρώθηκε με επιτυχία ο έλεγχος και με το id του χρήστη. Έχοντας το id γίνεται λήψη των δεδομένων του χρήστη.

```
case R.id.login:
dialog = ProgressDialog.show(SplashScreen.this, "", "Downloading personal categories");
new Thread(){
public void run(){
//validation
try {
HttpPost post = new HttpPost("http://192.168.0.2:3000/api/v1/sessions");
JSONObject holder = new JSONObject();
JSONObject userObj = new JSONObject();
String response = null;
JSONObject json;

userObj.put("email", un.getText().toString());
userObj.put("password", pass.getText().toString());
holder.put("user", userObj);
StringEntity se = new StringEntity(holder.toString());
post.setEntity(se);
post.setHeader("Accept", "application/json");
post.setHeader("Content-Type", "application/json");
ResponseHandler<String> responseHandler = new BasicResponseHandler();
response = NetworkManager.userclient.execute(post, responseHandler);
json = new JSONObject(response);

if(json.getBoolean("success")){
```

```
s=json.getString("current_user");

dm.setUserCategories(nm.getUserJson("http://192.168.0.2:3000/users/"+s+"/subscriptions.json"));
}

dialog.dismiss();

Intent intent = new Intent(SplashScreen.this, UserCategoryList.class);
startActivity(intent);

finish();

}catch (Exception e){
dialog.dismiss();
stopThread(this);
}
}//end of run
}.start();
break;
```

16.5 Κλάση Network Manager

Για την λήψη των δεδομένων δημιουργήσαμε μια κλάση που την ονομάσαμε Network Manager. Η μέθοδοι της δέχονται σαν όρισμα το url και επιστρέφουν έναν πινάκα με JSON Objects. Επιπλέον υπάρχει μια μέθοδος για την καταστροφή του session αλλά και μια μέθοδος ιδία ακριβώς με την πρώτη με την διαφορα οτι δημιουργήσαμε μια μεταβλητή τύπου http client η όποια είναι προσβάσιμη απο όλες τις κλάσεις και χρησιμοποιείται για την λήψη των δεδομένο του χρηστή και για να μένει το session ανοιχτό.

```
public ArrayList<JSONObject> getJson(String URL) throws ClientProtocolException, IOException, JSONException{

        StringBuilder url = new StringBuilder(URL);

        HttpGet get = new HttpGet(url.toString());
        HttpResponse r = client.execute(get);
        int status = r.getStatusLine().getStatusCode();
        if(status == 200){
        HttpEntity e = r.getEntity();
        String data = EntityUtils.toString(e);
        JSONArray array = new JSONArray(data);
        ArrayList<JSONObject> obj = new ArrayList<JSONObject>();
        obj.clear();
        for(int i=0;i<=array.length()-1;i++){
                obj.add(array.getJSONObject(i));
        }
        return obj;
        }
        else{
        return null;
        }
}

public ArrayList<JSONObject> getUserJson(String URL) throws ClientProtocolException, IOException,
JSONException{
```

```
        StringBuilder url = new StringBuilder(URL);

        HttpGet get = new HttpGet(url.toString());
        HttpResponse r = userclient.execute(get);
        int status = r.getStatusLine().getStatusCode();
        if(status == 200){
        HttpEntity e = r.getEntity();
        String data = EntityUtils.toString(e);
        JSONArray array = new JSONArray(data);
        ArrayList<JSONObject> obj = new ArrayList<JSONObject>();
        obj.clear();
        for(int i=0;i<=array.length()-1;i++){
        obj.add(array.getJSONObject(i));
        }
        return obj;
        }
        else{
        return null;
        }
}

public void DestroySession(){
        try{
        HttpDelete del = new HttpDelete("http://192.168.0.2:3000/api/v1/sessions");
        del.addHeader("Content-type", "application/json");
        userclient.execute(del);
        }catch(Exception e){}
}
}
```

16.6 Κλάση Data Manager

Μια άλλη βασική κλάση που δημιουργήσαμε είναι η κλάση Data Manager. Εκεί όλες οι μέθοδοι είναι της μορφής set και get και κάθε φορά που γίνεται λήψη δεδομένων μέσω της κατάλληλης μεθόδου αποθηκεύονται σε λίστες οι πίνακες οι όποιοι εν συνεχεία δύνονται σαν όρισμα σε μια κλάση που είναι υπεύθυνη για την οπτική απεικόνιση της λίστας.

```
        public ArrayList<JSONObject> getCategories(){
        return categories;
        }

        public void setCategories(ArrayList<JSONObject> arrayList){
        categories=arrayList;
        }

        public JSONObject getCategory(int i){
        return categories.get(i);
        }

        public int ArrayLength(){
        return categories.size();
        }
```

```
public void setData(ArrayList<JSONObject> arrayList){
insideCategory=arrayList;
}

public int DetailedListLength(){
return insideCategory.size();
}

public JSONObject getData(int i){
return insideCategory.get(i);
}

public void setUserCategories(ArrayList<JSONObject> arrayList){
usercategories=arrayList;
}
```

```
public JSONObject getUserCategory(int i){
return usercategories.get(i);
}

public int UserArrayLength(){
return usercategories.size();
}

}
```

16.7 Νήματα

Ένα άλλο πολύ σημαντικό στοιχειό της εφαρμογής είναι τα threads. Ο σκοπός τους είναι να κάνουν την εμπειρία του χρήστη πιο "ομαλή". Ουσιαστικά όταν ο χρήστης θέλει να δει ορισμένα δεδομένα γίνεται λήψη αυτών. Εκείνη την στιγμή εμφανίζεται στην οθόνη ένα μήνυμα με ένα γραφικό στοιχειό (progress dialog) που τον ενημερώνει ότι γίνεται λήψη των δεδομένων. Επιπλέον κατά την έξοδο από την εφαρμογή, σε περίπτωση που προηγουμένως έχει συνδεθεί ο χρήστης, πρέπει να τερματιστεί το session για λογούς ασφαλείας και για να μην καταναλώνονται πόροι απο τον server και την εφαρμογή. Η καταστροφή του session γίνεται πάλι από ένα thread προκειμένου να μην "παγώνει" κατά το κλείσιμο.

Βιβλιογραφία

http://mongoid.org/en/mongoid/index.html

http://www.mongodb.org/

http://ruby.railstutorial.org/ruby-on-rails-tutorial-book

http://railsforzombies.org/

Html5 The Missing Manual

Rails 3 In Action

Advanced Rails Recipes – 84 New Ways to Build Stunning Rails Apps

Agile Web Development With Rails

Επίσημο κανάλι της Google στο Youtube
(http://www.youtube.com/watch?v=Mm6Ju0xhUW8)

Χριστόπουλος Τηλεπικοινωνίες Α.Ε
(http://www.christopoulos.com.gr/index.php?option=com_content&view=article&id=78:2011-02-13-16-31-11&catid=19:2011-02-13-15-24-23&Itemid=22)

Επίσημο εγχειρίδιο της Google για το Android
(http://developer.android.com/guide/components/activities.html)

James Steele and Nelson To
(The Android Developers Cookbook. Building Applications with the Android SDK)

Reto Meier
(Android Application Development)

Reto Meier
(Android Application Development 2)

Ed Burnette
(Hello, Android. Introducing Google's Mobile Development Platform - second edition)

Sayed Y. Hashimi, Satya Komatinemi, Dave MacLean
(Pro Android 2)