Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων,
Μηχανικών Πληροφορικής

Πτυχιακή Εργασία
Τίτλος:

Δημιουργία δυναμικής σέλιδας (webservice) γραφείου ενοικίασης επαγγελματικών χώρων και οικημάτων με γεωγραφική αναπαράσταση αυτών με τη βοηθεια των google maps και της γλώσσας Ruby.

Λάμπρου Κωνσταντίνος (Α.Μ.:1615)

Επιβλέπων Καθηγητής: Νίκος Παπαδάκης

Ευχαριστίες

Ευχαριστώ την Δήμητρα Γαλανάκη για τη στήριξη της και τον καλό μου φίλο Καπενεκάκη Κώστα, για τα προγραμματιστικά ερεθίσματα και το brainstorming που κάναμε.

Abstract

The project you are about to see in the next pages it has to do with a webservice that helps a rental office to represent every available space for rent in a map. This specific paper is specialized in this direction and all the technologies required for something like this to be build.

Web app includes a simple admin panel based on basic authentication that can easily insert new spaces and manage them.

The project includes various technologies in order to build the application, as the famous Ruby language in combination with an elegant minimal framework called Sinatra.
Other technologies like a new object-oriented database called MongoDB (NoSQL) with specific analysis and design to feet in our purposes.

After that, follows an extensively analysis for the rest technologies included like: Html,5, CSS3, Git as a version control and deployment tool, the cloud based heroku / mongoLab platform which allows easily to host and deploy our application and its database

The projects makes several references on the open source technologies used. Several other technologies have been used in order for this webpage to be able to work without
problems and to be user friendly.

Περίληψη

Το παρακάτω project περιγράφει τη λειτουργία ενός webservice (μιας εφαρμογής - πλατφόρμας), το οποίο βοηθά ένα γραφείο ενοικίασης επαγγελματικών χώρων και οικιών να αναπαριστήσει γεωγραφικά και να διαχειριστεί όλους τους ανοίκιαστους χώρους με την χρήση των google maps / markers.

Η συγκεκριμένη αναφορά γίνεται προς την κατεύθυνση της δημιουργίας της εφαρμογής, όπως και στις απαιτούμενες τεχνολογίες που θα χρησιμοποιηθούν.

Όταν αναφερόμαστε στον όρο "γεωγραφική απεικόνιση" εννοούμε τη χρήση των χαρτών της google (google maps) μέσω του API που παρέχει και την εισαγωγή των επαγγελματικών χώρων ως markers στον χάρτη. Με αυτόν τον τρόπο μπορεί να γίνει εύκολη αναζήτηση εικονικά.

Η δημιουργία της εφαρμογή περιλαμβάνει ένα μεγάλο εύρος τεχνολογιών οπως αυτές παραθέτονται και αναλύονται παρακάτω.
Ενδεικτικά κάποιες από αυτές ειναι: η δημοφιλής αντικειμενοστραφής γλώσσα προγραμματισμού Ruby, σε συνδυασμό με το ευρέως διαδεδομένο web framework Sinatra (ένα υπερσύνολο βιβλιωθηκών που κάνει εύκολη τη δημιουργία διαδυκτιακών εφαρμογών), όπως και της εγγραφοστραφής βάσης δεδομένων mongoDB (NoSQL). Η εφαρμογή και η βάση φιλοξενούνται στο cloud. Στις γνωστές πλατφόρμες heroku και MongoLab (βαση).

Περισσότερες αναφορές στις υπόλοιπες open source και μη τεχνολογίες που θα λάβουν μέρος, όπως:
html5 / CCS3, Javascript, Thin webserver, ruby gems, Git, photoshop,

Τέλος, αναφορά στην τελική φάση του deployment/hosting στις γνωστή πλατφόρμα του heroku/mongoLab.

Παιρεταίρω ανάλυση για τις παραπάνω, αλλά και για επιπρόσθετες τεχνολογίες που κανούν την εφαρμογή φιλική προς τον χρήστη, θα ακολουθήσει στα παρακάτω κεφάλαια.

Πίνακας Περιεχομένων

1. Εισαγωγή

1.2 Κίνητρα για την διεξαγωγή της πτυχιακής

Τα κίνητρα που μας ώθησαν στο να ασχοληθούμε με ένα τέτοιο θέμα είχαν να κάνουν κυρίως με τις ανάγκες του σύγχρονου άνθρωπου αλλά και την όλο και αυξανομένη ζήτηση τέτοιον τεχνολογιών στη σύγχρονη κοινωνία.
Το χαμηλό κόστος λειτουργίας, η 24ωρη λήψη παραγγελιών καθώς και η άνεση του να παραγγέλλει κάποιος ένα προϊόν από τον καναπέ του σπιτιού του γλιτώνοντας έτσι τις πολλές ώρες αναμονής στις ουρές των καταστημάτων είναι κάποιοι από τους κύριους λόγους. Ένα επίσης προτέρημα ενός ηλεκτρονικού καταστήματος είναι η εύκολη διάθεση των προϊόντων του σε μέρη, χώρες ακόμα και ηπείρους πολύ μακριά από την βάση του καταστήματος μέσω υπηρεσιών ταχυδρομείου.

1.3 Δομή πτυχιακής εργασίας.

Η δομή της πτυχιακής μας εργασίας θα είναι η εξής:

Στο πρώτο μέρος θα μελετήσουμε την μεθοδολογία υλοποίησης προκειμένου να εξοικειωθούμε με όλες τις τεχνολογίες που έχουν χρησιμοποιηθεί. Στην συνέχεια θα προχωρήσουμε στο σχέδιο δράσης και την εκπόνηση της εργασίας μας. Τέλος θα πραγματοποιηθεί ανάλυση του προβλήματος και η επίλυση του.

2. Μεθοδολογία Υλοποίησης

Η μέθοδος που θα χρησιμοποιήσουμε θα είναι να σπάσουμε αυτό το γενικό και δύσκολο πρόβλημα σε μικρότερα υπό-προβλήματα και να αναλύσουμε καθένα από αυτά ξεχωριστά.

2.2 Θεωρία

2.2.1 Open Source

Στον χώρο της πληροφορικής και των ηλεκτρονικών υπολογιστών, με τον όρο λογισμικό ανοικτού κώδικα (αγγλ.: Open Source Software, OSS) εννοείται λογισμικό του οποίου ο πηγαίος κώδικας διατίθεται με κάποιον τρόπο ελεύθερα σε όσους ζητούν να τον εξετάσουν, ακόμα και να τον τροποποιήσουν ή αξιοποιήσουν σε άλλες εφαρμογές. Κατά καιρούς έχουν εμφανιστεί αρκετές διαφορετικές άδειες χρήσης σχεδιασμένες να συνοδεύουν λογισμικό ανοικτού κώδικα.

Το λογισμικό ανοικτού κώδικα δεν σημαίνει απαραιτήτως δωρεάν λογισμικό, ούτε ελεύθερο λογισμικό σύμφωνα με τον ευρύ ορισμό που δίνει στο ελεύθερο λογισμικό το Ίδρυμα Ελεύθερου Λογισμικού, αλλά αναφέρεται μόνο στο γεγονός πως επιτρέπεται σε κάθε χρήστη να εξετάσει και να χρησιμοποιήσει τη γνώση και τις δυνατότητες που προσφέρει ο παρεχόμενος πηγαίος κώδικας. Στην πράξη, τα περισσότερα προγράμματα ανοιχτού κώδικα παρέχονται δωρεάν και μπορούν να χαρακτηριστούν ελεύθερα.

2.2.2 Στατικές και δυναμικές Ιστοσελίδες

Γενικά, αυτό που προσφέρουν οι δυναμικές ιστοσελίδες, είναι μεγαλύτερη αλληλεπίδραση του χρήστη με την σελίδα π.χ. να προσθέτει τα σχόλια του στην σελίδα, αλλά και πολλές ευκολίες στον διαχειριστή του περιεχομένου της ιστοσελίδας π.χ. τον ιδιοκτήτη της σελίδας. Πάντως το γεγονός ότι μια δυναμική ιστοσελίδα προσφέρει περισσότερες δυνατότητες, δεν σημαίνει ότι αυτές είναι απαραίτητες σε όλους, δηλαδή σε αρκετές περιπτώσεις, μία στατική ιστοσελίδα μπορεί να καλύπτει πλήρως τις ανάγκες μιας συνοπτικής παρουσίασης.. Από πλευράς κόστους, η στατική ιστοσελίδα είθισται να είναι η φτηνή επιλογή, καθώς είναι πιο απλή η κατασκευή της ιστοσελίδας, ενώ οι δυναμικές ιστοσελίδες λόγω της πολυπλοκότητας τους κοστίζουν ακριβότερα και αυτό είναι λογικό τουλάχιστον όταν γίνονται κατά παραγγελία. Υπάρχει βέβαια και η περίπτωση υλοποίησης μιας δυναμικής ιστοσελίδας με την χρήση κάποιας open source εφαρμογής (CMS) η οποία διατίθεται δωρεάν μέσω του internet και σε αυτήν την περίπτωση η κατασκευή της ιστοσελίδας μπορεί να έχει μηδενικό κόστος (αν γίνει self-service) ή να υπάρξει κάποια χρέωση (η τελική τιμή μιας ιστοσελίδας είναι υποκειμενική υπόθεση) αν η εγκατάσταση και η τυχόν παραμετροποίηση της δωρεάν εφαρμογής ανατεθεί σε κάποια εταιρία.

2.2.3 Web Frameworks

Η ανάπτυξη ενός webservice είναι μια web εφαρμογή που αντλεί και αποθηκεύει πληροφορίες σε μια βάση δεδομένων. Εκτελείται σε ένα web server (με οποιοδήποτε λειτουργικό, συνηθως Linux) για να είναι διαθέσιμη στο διαδίκτυο. Απαιτείται λοιπόν ανάπτυξη βάσης δεδομένων και παράλληλα ανάπτυξη λογισμικού για την άντληση πληροφοριών από την βάση καθώς και την αποθήκευση πληροφοριών στη βάση.

Παραδείγματα γνωστών web frameworks:

Ruby
- Ruby on Rails
- Sintra
- Merb

Python
- Django
- CherryPy

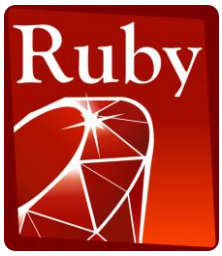
PHP

- CakePHP
- CodeIgniter
- Cohana

Scala

2.3 Τεχνολογίες

2.3.1 Ruby



Η **Ruby** είναι μια δυναμική, ανακλαστική, αντικειμενοστρεφής γλώσσα προγραμματισμού γενικής χρήσης που συνδυάζει μια σύνταξη επηρεασμένη από την Perl με χαρακτηριστικά από τη Smalltalk. Η Ruby προήλθε από την Ιαπωνία στα μέσα της δεκαετίας του 1990 και αρχικά σχεδιάστηκε και αναπτύχθηκε από τον Yukihiro "Matz" Matsumoto. Βασικές της επιρροές είναι η Perl, η Smalltalk, η Eiffel και η Lisp. Η Ruby υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού όπως ο συναρτησιακός προγραμματισμός, ο αντικειμενοστρεφής προγραμματισμός, ο προστακτικός προγραμματισμός και οανακλαστικός (reflective) προγραμματισμός. Έχει σύστημα δυναμικών τύπων και αυτόματη διαχείριση μνήμης, επομένως μοιάζει σε κάποια χαρακτηριστικά της με την Python, την Perl, τη Lisp, τηDylan, την Pike και τη CLU.

Η πρότυπη υλοποίηση 1.8.7 της Ruby είναι γραμμένη σε C, σαν μια διερμηνευόμενη γλώσσα ενός περάσματος. Προς το παρόν δεν υπάρχει κάποιο επίσημο πρότυπο αναφοράς για τη γλώσσα Ruby, επομένως η αρχική υλοποίηση θεωρείται το ντε φάκτο σημείο αναφοράς. Υπάρχουν αρκετές (ολοκληρωμένες ή σε ανάπτυξη) εναλλακτικές υλοποιήσεις της γλώσσας, συμπεριλαμβανομένων τωνYARV, JRuby, Rubinius, IronRuby, MacRuby και HotRuby, κάθε μια από τις οποίες και έχει διαφορετική προσέγγιση, με τις IronRuby, JRuby και MacRuby να προσφέρουν just-in-time compilation και τη MacRuby να προσφέρει επιπλέον ahead-of-time compilation. Ο κώδικας της επίσημης έκδοσης 1.9 χρησιμοποιεί τη YARV, όπως και αυτός της έκδοσης 2.0 (σε ανάπτυξη), η οποία και θα αντικαταστήσει την πιο αργή Ruby MRI.

2.3.2 Gems

Στη Ruby οι βιβλιοθήκες λέγονται **gems** και ο τρόπος διαχείρισής τους θυμίζει τον τρόπο εγκατάστασης προγραμμάτων των unix συστημάτων.

Τα RubyGems είναι ένας package manager για τη γλώσσα Ruby που παρέχουν βασικές βιβλιοθήκες (σφραγισμένες και αυτόνομες σε ανεξάρτητα πακέτα που ονομάζονται "gems"). Ένα gem είναι ενα εργαλείο σχεδιασμένο να κάνει μια δουλειά. Ο αντίστοιχος manager στην python είναι το Easy Install.

Δημοφιλή Gems:

1.RMagick
2.Cancan
3.CarrierWave
4.Kaminari
5.HAML
6.factory_girl
7.SASS κτλ..

2.3.4 Sinatra

Το framework sinatra είναι μια βιβλιοθήκη γραμμένη με στη γλώσσα Ruby με σκοπό την δημιουργία web εφαρμογών. Μοιάζει πολύ με της άλλες βιβλιοθήκες, όπως Ruby on Rails, Merb, Nitro και Camping. Είναι βασισμένο στον Rack web server. Το framework sinatra αναπτύχθηκε και σχεδιάστηκε το 2007 από τον Blake Mizerany με την προοπτική να είναι μικρό και ευέλικτο εργαλείο που θα προσφέρει γρήγορη και εύκολη ανάπτυξη εφαρμογών. Δεν ακολουθεί το κλασσικό MVC (model-view-controller) pattern για τον λόγο ότι δίνει έμφαση στην δημιουργία εφαρμογών με τον λίγοτερο δυνατόν κόπο. Πολλές μεγάλες εταιρείες συμπεριλαμβανομένων των Apple, Github, RedHat, BBC έχουν χρησιμοποιήσει το framework για τις εφαρμογές τους.

Παράδειγμα κώδικα:

```
#!/usr/bin/env ruby
require 'sinatra'

get '/' do
  redirect to('/hello/World')
end

get '/hello/:name' do
  "Hello #{params[:name]}!"
end
```

2.3.4 Γλώσσα Προγραμματισμού HTML

Η HTML (ακρωνύμιο του αγγλικού HyperText Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες, οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης (CSS) για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού.

Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

```
<body onload="init()">
    <div class="snap-drawers">
        <div class="snap-drawer snap-drawer-left">
            <div>
                <ul>
                    <li><a id="feed" href="#"><span class="menu-icon feed"></span> Feed<img src="img/arrow-righ
                    </li>
                    <li><a id="promoted" href="#"><span class="menu-icon promoted"></span> Promoted<img src="i
                    arrow-menu"></a></li>
                    <li><a id="popular" href="#"><span class="menu-icon popular"></span> Popular<img src="img/a
                    menu"></a></li>
                    <li><a href="#"><span class="menu-icon nearby"></span> Nearby<img src="img/arrow-right_@2x.
                    <li><a href="#"><span class="itsme"><img src="http://images.ak.instagram.com/profiles/profi
                    </span>
                    <span class="user-name"> dew</span>
                    <span class="settings-gear">&#xe001;</span></a></li>
                    <li class="stripe-menu stripe-words">Photos By Me</li>
                </ul>
                <div>
                    <div id="photos_by_me" class="clearfix"></div>
                </div>
            </div>
        </div>
        <div class="snap-drawer snap-drawer-right"></div>
    </div>
```

2.3.5 Γλώσσα μορφοποίησης CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης (html). Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στη γλώσσα HTML, δηλαδή για τον έλεγχο της εμφάνισης ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώμματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

Η αλληλουχία εφαρμογής των φύλλων στύλ

Για ένα έγγραφο πχ xhtml θα υπάρχουν παραπάνω από ένα φύλλα στυλ τα οποία περιέχουν δηλώσεις για την εμφάνιση ενός συγκεκριμένου στοιχείου. Το Φύλλο στύλ που εφαρμόζεται σε ένα έγγραφο μπορεί να προέρχεται από :

- το συγγραφέα μιας ιστοσελίδας
- το χρήστη του πλοηγού
- τον ίδιο τον πλοηγό, αν έχει το δικό του φύλλο στυλ .

Συνεπώς για ένα html tag θα υπάρχουν παραπάνω από μια δηλώσεις που πιθανόν να είναι συγκρουόμενες.Το πρότυπο css για να επιλύσει παρόμοιες συγκρούσεις έχει καθορίσει μια αλληλουχία-σειρά στην οποία θα μπούν αυτές οι δηλώσεις και με βάση την οποία θα επιλεγεί πχ η δήλωση που είναι πρώτη στη σειρά.

Ο αλγόριθμος δημιουργίας αυτής της σειράς-αλληλουχίας είναι ο ακόλουθος:

1. Βρες όλες τις δηλώσεις που εφαρμόζονται στο στοιχείο που μας ενδιαφέρει. Οι δηλώσεις εφαρμόζονται στο στοιχείο αν ο επιλογέας του το επιλέξει (ταιριάζει με αυτό).
2. Ταξινόμησε με βάση τη σημασία (κανονική ή σημαντική) και προέλευση (συγγραφέας , χρήστη ή πλοηγός χρήστη). Με αύξουσα σειρά προτεραιότητας:
    1. Δηλώσεις πλοηγού χρήστη
    2. Κανονικές δηλώσεις χρήστη
    3. Κανονικές δηλώσεις συγγραφέα
    4. Σημαντικές δηλώσεις συγγραφέα
    5. Σημαντικές δηλώσεις χρήστη
3. Ταξινόμησε τις δηλώσεις ίδιας σημασίας και προέλευσης με κριτήριο την εξειδίκευση του επιλογέα: οι πιο εξειδικευμένοι επιλογείς υπερισχύουν των πιο γενικών. Τα ψευδό-στοιχεία και οι ψευδο-κλάσεις λογαριάζονται σαν κανονικά στοιχεία και κλάσεις αντίστοιχα.
4. Τέλος ταξινόμησε ανάλογα με τη σειρά καθορισμού: αν δύο δηλώσεις έχουν το ίδιο βάρος , προέλευση και εξειδίκευση , αυτή που προσδιορίστηκε τελευταία επικρατεί. Οι δηλώσεις σε εισαγώμενα φύλλα στυλ θεωρούνται ότι δηλώνονται πριν από τις δηλώσεις στο ίδιο το φύλλο στυλ .

Αφού λοιπόν προκύψει μια σειρά-αλληλουχία κανόνων εμφάνισης που αφορούν το ίδιο στοιχείο θα επιλεγεί προς εφαρμογή (για την αποφυγή συγκρούσεων) η δήλωση που θα είναι τελευταία στην σειρά που αναλύθηκε πιο πάνω.

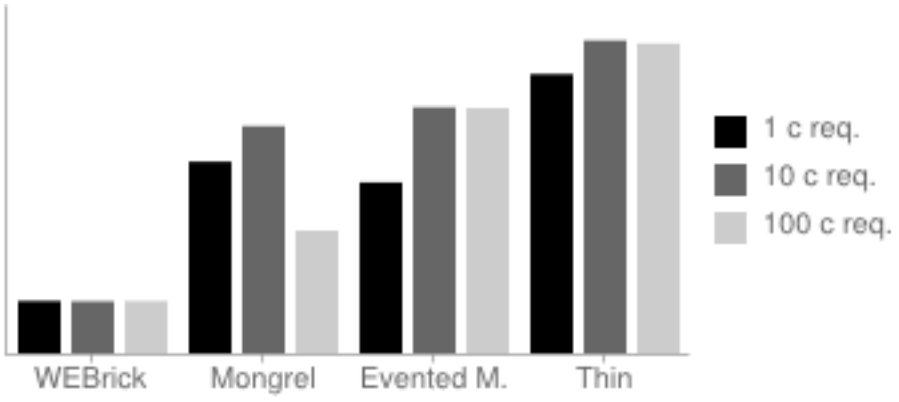
```
.feature {
    position: relative;
    margin: 0 auto;
    top: 150px;
    max-width: 954px;
    height: 620px;
    padding: 0px 5px;
}

.animation{-webkit-animation: voila 1s ease-out;}
.second-fade{
}

.right { float: right; }
.left { float: left; }

.words {
    color: white;
    text-shadow: 0px 1px 1px black;
    display: table;
    text-align: left;
}

.title { font-size: 38px; margin-top: 80px; }
.explain {
    font-size: 24px; margin-top: 40px;
}
```

2.3.6 Ο Thin Webserver

Ο thin web server ειναι ένας εξαιρετικά γρήγορος εξυπυρετητής ιδανικός για Ruby εφαρμογές, ο οποίος συνδυάζει τις 3 καλύτερες βιβλιοθήκες που γράφτηκαν ποτέ με την γλώσσα, για τη γλώσσα.
Και αυτες είναι:
Mongrel Parser (Mongrel server root),
EventMachine (πρέκειται για ένα network I/O library με εξαιρετικές αντοχές, επιδόσεις, και σταθερότητα.),
Rack (ενα library με το ελάχιστο απαιτούμενο interface για τη διασύνδεση μεταξύ server και Ruby frameworks.)

Όλα τα παραπάνω τον καθιστούν ως έναν απο τους πιο ασφαλείς, σταθερούς, γρήγορους και επεκτάσιμους Ruby web servers.

Request / seconds comparison



c req. = Concurrency Level (number of simultaneous requests)

2.3.7 NoSQL και MongoDB

Αρχίζοντας την αναφορά μας στα **NoSQL** (διαβάζεται Not Only SQL ή No SQL) συστήματα και βάσεις δεδομένων θα λέγαμε πως πρόκειται για μια ευρεία ομάδα συστημάτων διαχείρισης βάσεων δεδομένων (database management system) που το κύριο χαρακτηριστικό του είναι η μη τήρηση του μοντέλου RDBMS (Relational Database Management System), το οποίο και χρησιμοποιείται «τυφλά» στην συντριπτική πλειοψηφία των περιπτώσεων. Ίσως επίσης παρατηρήσατε πως δεν γίνεται καμία αναφορά στο «τι είναι NoSQL βάσεις». Και αυτό γιατί δεν μπορούμε να δώσουμε έναν σαφή ορισμό να πούμε τί είναι ή τι δεν είναι ένα σύστημα NoSQL, απλά λέμε πως δεν είναι RDBMS και πως χρησιμοποιεί εντελώς διαφορετικό από τον κλασικό τρόπο για την διαχείριση των δεδομένων μέσα στην βάση (data manipulation). Οι NoSQL βάσεις δεδομένων γενικώς δεν χρησιμοποιούν κάποιο δομημένο σύστημα για τα στοιχεία που περιλαμβάνουν, όπως πχ πίνακες, ούτε χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων, αλλά χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων.

Τα NoSQL συστήματα κατά κύριο λόγο είναι βελτιστοποιημένα (optimized) ώστε να ανακτούν και να επισυνάπτουν δεδομένα. Η μειωμένη ευελιξία του χρόνου εκτέλεσης σε σύγκριση με συστήματα SQL (δηλαδή με τα RDBMS) αντισταθμίζεται από την σημαντική αύξηση στην απόδοση (και την επεκτασιμότητα) για ορισμένα μοντέλα δεδομένων. Τα δεδομένα θα μπορούσαν να είναι δομημένα, αλλά αυτό είναι στην πραγματικότητα ασήμαντο καθώς αν κάτι έχει ουσιαστική σημασία στα NoSQL συστήματα αυτό είναι η ικανότητα να αποθηκεύουν και να ανακτούν μεγάλες ποσότητες δεδομένων, «αδιαφορώντας» για τις σχέσεις μεταξύ των στοιχείων αυτών.

Τα βασικά χαρακτηριστικά των NoSQL database systems είναι:

1) Η μη χρήση της SQL ως query language,

2) δεν μπορεί να εγγυηθεί ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα και 3) έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας.

Αναλυτικότερα, τα NoSQL συστήματα αναπτύχθηκαν και εξελίχθηκαν παράλληλα με τις μεγαλύτερες εταιρίες πληροφορικής στον κόσμο, όπως η Google ή η Amazon, που είχαν ανάγκη να διαχειρίζονται έναν τεράστιο όγκο δεδομένων, κάτι στο οποίο δεν βόλευαν τα παραδοσιακά μέχρι τότε συστήματα RDBMS. Τα NoSQL συστήματα έχουν φτιαχτεί έτσι ώστε να μπορούν να διαχειριστούν μεγάλες ποσότητες δεδομένων χωρίς κατ'ανάγκη να διατηρούν μία συγκεκριμένη δομή (schema). Επίσης, οι μέθοδοι υλοποίησης και εφαρμογής τους αξιοποιούν μια αρχιτεκτονική που επιτρέπει (και ίσως διευκολύνει) την κατανεμημένη λειτουργία του συστήματος. Έτσι με αυτόν τον τρόπο το σύστημα μπορεί θεωρητικά να «απογειωθεί» σε επιδόσεις, από την στιγμή που μπορούν να προστεθούν θεωρητικά άπειροι servers όπου κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος. Ακόμη, η ενδεχόμενη αδυναμία λειτουργίας ενός server του συστήματος μπορεί να αντιμετωπισθεί εύκολα. Αυτός ο τύπος βάσεων δεδομένων αναπτύσσεται διαρκώς οριζόντια, καθώς αυξάνονται οι ανάγκες για storage δεδομένων, που συνεχώς πληθαίνουν (data growth), ενώ είναι πιο σημαντικές οι επιδόσεις προσπέλασης των δεδομένων από την συνοχή που παρουσιάζουν.

Αφού είδαμε πιο συγκεκριμένα κάποια στοιχεία σχετικά με την λειτουργία των NoSQL συστημάτων, τώρα θα αναφερθούμε πιο ειδικά στα πλεονεκτήματα τους, σε πρακτικό πια επίπεδο. Αρχικά, παρουσιάζουν μεγάλη «ελαστικότητα» στην βελτίωση της επίδοσης τους. Μάλιστα από την στιγμή που η υλοποίησή τους τις περισσότερες φορές γίνεται σε υποδομές cloud ή σε virtualized environments, συμφέρουν πολύ από οικονομική άποψη, σε αντίθεση με τα κλασικά RDBMS συστήματα. Ειδικότερα, ενώ στα RDBMS συστήματα, όταν θέλουμε να τα βελτιώσουμε, προσθέτουμε περισσότερη μνήμη RAM ή καλύτερους επεξεργαστές στις υποδομές μας, αντίθετα στα NoSQL συστήματα απλά προσθέτουμε κόμβους ώστε να επεξεργάζονται ακόμη περισσότερα δεδομένα ταυτόχρονα. Επίσης, η συντήρηση ενός high-end συστήματος RDBMS που διαχειρίζεται τεράστιο όγκο δεδομένων απαιτεί εξίσου μεγάλο κόστος (κυρίως Database Administrators, που αρχίζουν πια να χάνουν την αξία τους). Αντίθετα, τα NoSQL συστήματα εξ'ορισμού είναι σχεδιασμένα ώστε να απαιτούν τη λιγότερη δυνατή διαχείριση

από τον ανθρώπινο παράγοντα, χρησιμοποιώντας μεθόδους όπως automatic repair, data distribution και πιο απλά μοντέλα δεδομένων. Όλα αυτά οδηγούν σε σαφώς μικρότερες ανάγκες για διαχείριση ή βελτιστοποίηση του συστήματος. Επιπλέον, οποιαδήποτε αλλαγή στο μοντέλο δεδομένων ενός συστήματος RDBMS είναι μία διαδικασία που απαιτεί ένα σεβαστό χρονικό διάστημα που η εφαρμογή που στηρίζεται στην συγκεκριμένη βάση δεδομένων θα είναι «κατεβασμένη» (downtime) ή θα μειωθούν σε μεγάλο βαθμό τα επίπεδα λειτουργίας της υπηρεσίας, ενώ ένα NoSQL σύστημα έχει πολύ λιγότερους (έως μηδαμινούς) περιορισμούς σε αυτό το ζήτημα.

Ας δούμε όμως λίγο πιο προσεκτικά ορισμένες λεπτομέρειες για τα NoSQL συστήματα. Εδώ θα πρέπει να αναφερθεί ότι υπάρχουν ορισμένες κατηγορίες για τις NoSQL βάσεις:

1) οι Key-values Stores,

2) οι Column Family Stores ,

3) οι Document Databases και

4) οι Graph Databases.

Η πρώτη κατηγορία, οι key-value stores, δημιουργήθηκαν με κυρίως ιδέα την ύπαρξη ενός hash table όπου υπάρχει ένα μοναδικό κλειδί και ένας δείκτης στοχεύοντας σε ένα συγκεκριμένο στοιχείο. Αυτό το είδος mapping συνήθως συνοδεύεται από μηχανισμους cache, για την καλύτερη απόδοση του συστήματος.

Οι Column family stores βάσεις σχεδιάστηκαν έτσι ώστε να διαχειρίζονται πολύ μεγάλα ποσά δεδομένων που είναι κατανεμημένα σε διάφορους servers. Όπως και στην προηγούμενη κατηγορία, έτσι κι εδώ υπάρχουν συγκεκριμένα κλειδιά (keys) που στοχεύουν όμως σε περισσότερα από ένα στοιχεία. Οι rows εδώ αναγνωρίζονται από ένα μοναδικό row key ενώ οι στήλες είναι οργανωμένες σε column families («οικογένειες στηλών»).

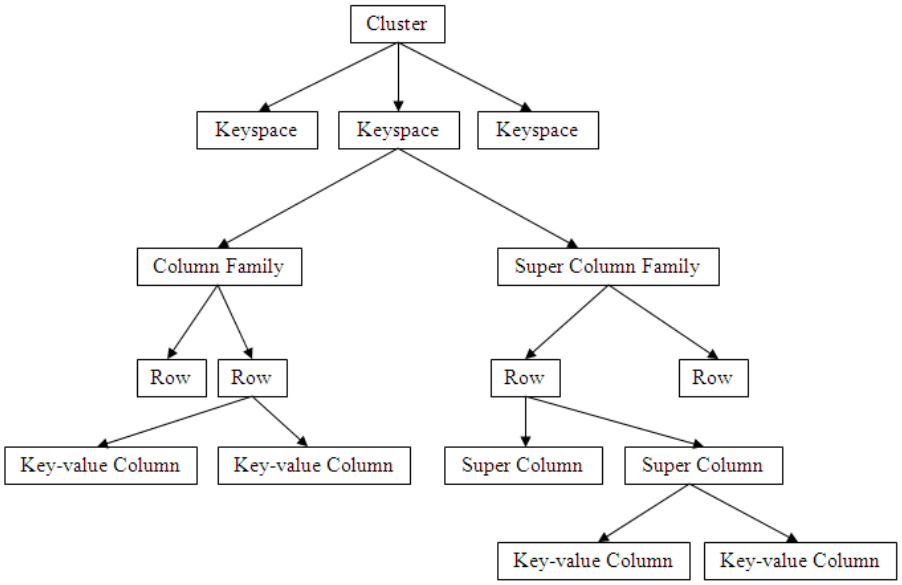
Περισσότερα:
http://www.linuxinside.gr/forum/8873/eisagogi-stis-nosql-baseis-dedomenon-xrisi-toy-cassandra#ixzz33n6pNEX0

MongoDB

Η mongoDB είναι μία document oriented database, αυτό σημαίνει ότι η αποθήκευση των εγγραφών σε ένα πίνακα της βάσης μπορεί να έχει τη μορφή δεδομένων json , δεν είναι απαραίτητο ο πίνακας να έχει συγκεκριμένο αριθμό πεδίων όπως στις σχεσιακές βάσεις, όπου εγγραφές συχνά περιέχουν πολλά κενά εξαιτίας του σταθερού αριθμού πεδίων του πίνακα. Με λίγα λόγια το Schema της βάσης δεν είναι απαραίτητα σταθερό.

Επιπλέον τα ερωτήματα που γίνονται για τη συλλογή των δεδομένων από τη βάση είναι δυναμικά και δεν απαιτούν συγκεκριμένους δείκτες, γράφονται σύμφωνα με την MongoDB's document-based query language. Αποφεύγονται τα ερωτήματα join που είναι και η αιτία που δεν μπορεί να γίνουν scale οι σχεσιακές βάσεις. Η mongoDB είναι προτιμότερο να χρησιμοποιείται σε εφαρμογές όπου τα δεδομένα αλλάζουν πολύ γρήγορα και χρειαζόμαστε ταχύτητα, μία άλλη βάση τέτοιου τύπου είναι couchDB όπου είναι προτιμότερο να χρησιμοποιείται σε δεδομένα που δεν άλλαζουν γρήγορα και τα ερωτήματα είναι προκαθορισμένα π.χ σε CMS και CRM.

Η επόμενη κατηγορία, των **Document databases**, είναι όμοιες με τις key-value stores. Τα δεδομένα σε αυτή την περίπτωση είναι οργανωμένα από «συλλογές» key-valued «συλλογών» δεδομένων.



Η τελευταία κατηγορία, αυτή των Graph databases, είναι βασισμένη σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graph model) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους).

Αφού αναφερθήκαμε και στις κατηγορίες των NoSQL συστημάτων , θα κάνουμε και μια αναφορά στις πιο γνωστές και διαδεδομένες βάσεις δεδομένων NoSQL.

1) **Cassandra**: Η Cassandra είχε δημιουργηθεί από την Facebook και ανήκει πια στα Apache projects. Είναι μια column oriented NoSQL database και ανοικτού κώδικα.

2) **Dynamo** και **SimpleDB**: Αποτελούν δημιουργίες της Amazon και η πρώτη είναι η πιο γνωστή Key-Value NoSQL βάση δεδομένων, ενώ η δεύτερη είναι μέρος των Amazon Cloud Services EC2 και S3.

3) **BigTable**: η BigTable είναι δημιούργημα της Google, και η χρήση της περιορίζεται μόνο μέσω του Google App Engine. Τεχνικά, είναι μια column oriented database κλειστού κώδικα.

4) **Neo4J**: αποτελεί μια τύπου graph **NoSQL** database ελεύθερου κώδικα.

5) **CouchDB** και **MongoDB**: αυτές οι δύο αποτελούν τις πιο γνωστές open source document oriented NoSQL βάσεις.

Οι περισσότεροι developers αναρωτιούνται πώς μπορούν να συντάξουν ένα query για μία NoSQL βάση καθώς το να υπάρχει αποθηκευμένο σε μία βάση δεν λέει τίποτα από μόνο του αν δεν μπορούμε να το εμφανίσουμε στους end users μέσω μίας πχ web application. Αντίθετα με τις RDBMS βάσεις, εδώ δεν υπάρχει κάποια συγκεκριμένη ξεκάθαρη query language όπως η SQL για την διαχείριση των δεδομένων αλλά αντίθετα η διαχείριση και η εξέταση αυτών των βάσεων είναι “εξαρτημένη” από μοντέλο δεδομένων που χρησιμοποιείται.

Κάποιες NoSQL πλατφόρμες επιτρέπουν την χρήση RESTful μοντέλων διεπαφής με τα δεδομένα ενώ άλλες παρέχουν query APIs. Υπάρχουν και μερικά (δύο για την ακρίβεια) εργαλεία που έχουν αναπτυχθεί με σκοπό την διαχείριση πολλαπλών NoSQL βάσεων δεδομένων, τα οποία προσπαθούν κυρίως να διαχειριστούν NoSQL βάσεις της ίδιας κατηγορίας.

Το ένα από αυτά (και πιο γνωστό) είναι το SPARQL, που απευθύνεται σε graph databases. Παρακάτω σε δύο παραδείγματα θα δούμε πως μπορούμε να

ανακτήσουμε το link ενός συγκεκριμένου blogger μέσα σε μία βάση με αποθηκευμένα στοιχεία για blogs και να κάνουμε μία αναζήτηση όλων των ονομάτων που αναφέρονται στο FOAF file του Tim Berners Lee.

FOAF file είναι ο τύπος των αρχείων μέσα στα οποία περιγράφονται οι άνθρωποι και οι σχέσεις τους. Αποτελεί ένα standard RDF λεξιλόγιο , αναγνωρίσιμο δηλαδή τύπο αρχείων από τις NoSQL βάσεις και το SPARQL.

Περισσότερα:
http://www.linuxinside.gr/forum/8873/eisagogi-stis-nosql-baseis-dedomenon-xrisi-toy-cassandra#ixzz33n8WOCsp

2.3.8 Javascript

Η **JavaScript** (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές.[1] Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.[1]
Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.[2] Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζονταςαντικειμενοστρεφές,[3] προστακτικό και συναρτησιακό[4][5] στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript

Παράδειγμα:

```
<script>
```

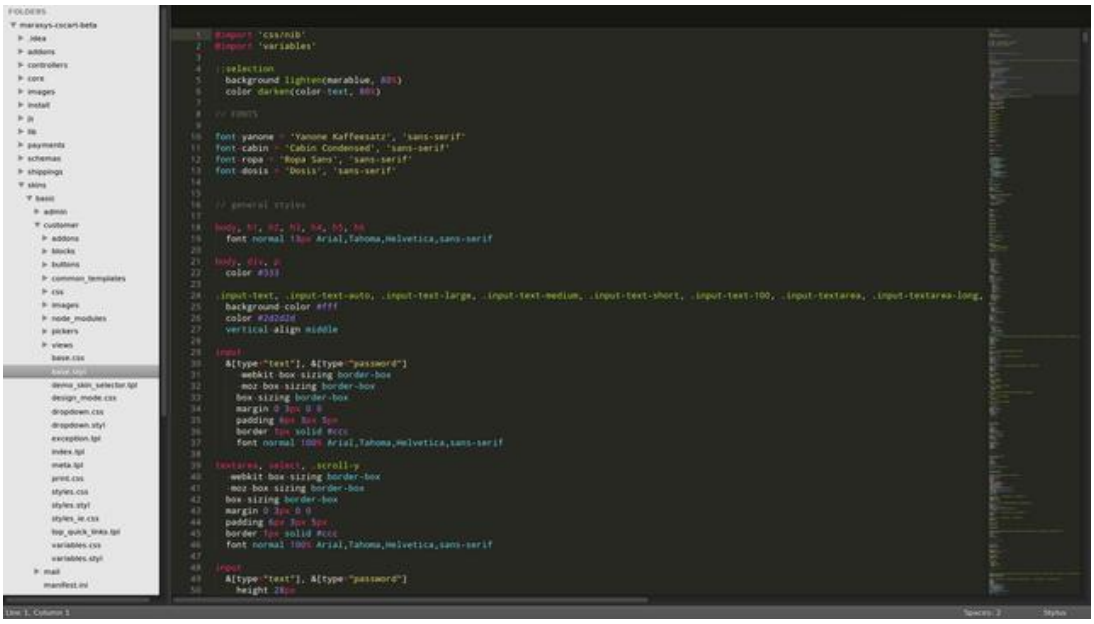
```
var FIRSTvariable = window.prompt("PLEASE FILL IN YOUR
NAME")
alert("Your name is " + FIRSTvariable + ".")
</script>
```

2.3.9 Google Maps API

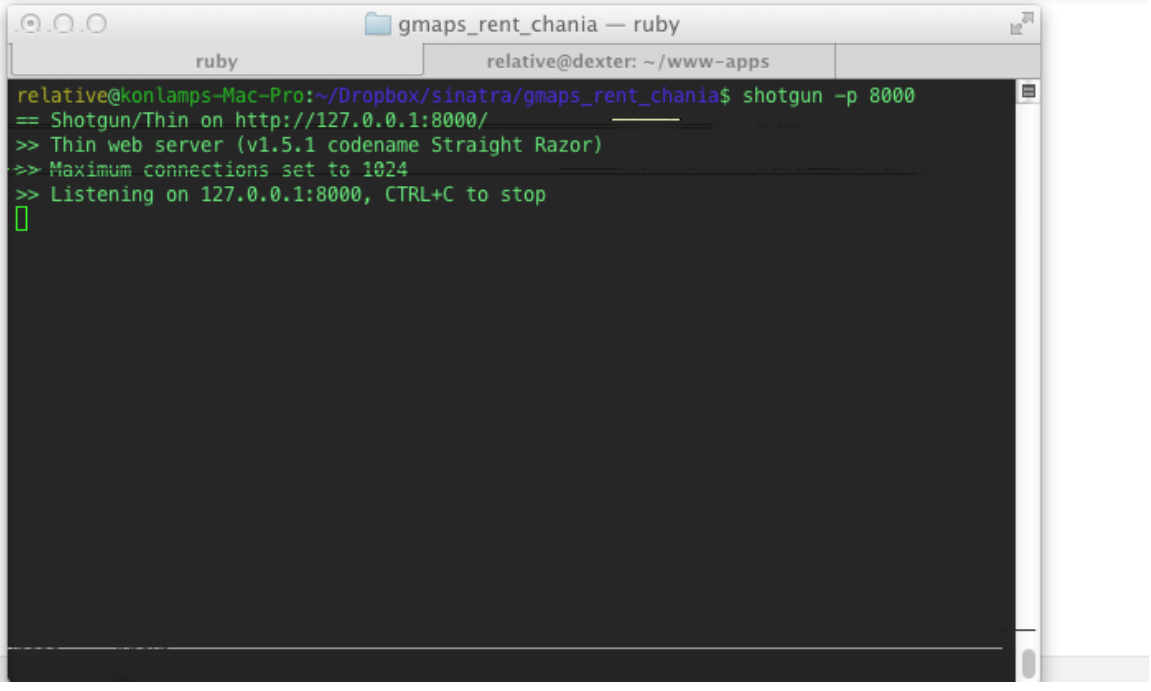
Οι **Χάρτες Google** (Google Maps) είναι υπηρεσία χαρτογράφησης στο Διαδίκτυο. Η εφαρμογή και η τεχνολογία της υπηρεσίας παρέχεται από την Google και υποστηρίζει πολλές υπηρεσίες που βασίζονται σε χάρτες, συμπεριλαμβανομένου της ιστοσελίδας "Χάρτες Google". Προσφέρει χάρτες δρόμων και σχεδιαστή διαδρομών για μεταφορές με τα πόδια, αυτοκίνητο, ποδήλατο (beta) ή μέσα μαζικής μεταφοράς. Περιλαμβάνει επίσης εντοπισμό των επιχειρήσεων που βρίσκονται σε πόλεις σε πολλές χώρες σε όλο τον κόσμο. Οι δορυφορικές εικόνες των Χαρτών Google δεν ανανεώνονται σε πραγματικό χρόνο, ωστόσο η Google προσθέτει δεδομένα στη Κύρια Βάση δεδομένων της σε τακτική βάση και οι περισσότερες από τις εικόνες δεν είναι πάνω από τριών ετών.

2.4 Εργαλεία και βοηθητικές εφαρμογές

2.4.1 Sublime editor



2.4.2 Console



2.4.3 Photoshop

Το Adobe Photoshop, ή απλά Photoshop, είναι ένα πρόγραμμα επεξεργασίας γραφικών που αναπτύχθηκε και κυκλοφόρησε από την Adobe Systems. Αυτή τη στιγμή αποτελεί ηγέτη της αγοράς (market leader) των προγραμμάτων επεξεργασίας εικόνων, και είναι το προϊόν - σήμα κατατεθέν της Adobe Systems. Χαρακτηρίζεται ως "απαραίτητο εργαλείο για τους επαγγελματίες γραφίστες" και θεωρείται πως προώθησε τις αγορές των Macintosh, και στη συνέχεια των Windows.

Η πιο πρόσφατη έκδοση του Adobe Photoshop είναι η Adobe Photoshop CS6 (13.0), που κυκλοφόρησε τον Μάιο του 2012. Διατίθεται στις εκδόσεις Standard και Extended. Η τελευταία διαφοροποιείται έναντι της απλής έκδοσης χάρη στα εργαλεία επεξεργασίας τρισδιάστατων αντικειμένων και ανάλυσης ποσοτικών δεδομένων εικόνας.

Ιστορία

Το 1987 ο Τόμας Κνολ, ένας φοιτητής του Πανεπιστημίου του Μίσιγκαν, ανέπτυξε ένα πρόγραμμα που εμφάνιζε εικόνες σε αποχρώσεις του γκρίζου (grayscale) σε μονοχρωματικό περιβάλλον. Αυτό το πρόγραμμα, το οποίο ονόμασε Display, τράβηξε την προσοχή του αδερφού του Τζον Κνολ, ο οποίος πρότεινε στον Τόμας να αναπτύξει ένα πλήρες πρόγραμμα επεξεργασίας εικόνας. Ο Τόμας έκανε διάλειμμα έξι μηνών από τις σπουδές του το 1988 και, σε συνεργασία με τον αδερφό του, ανέπτυξε το πρόγραμμα, το οποίο ονόμασαν ImagePro. Αργότερα το ίδιο έτος, ο Τόμας μετονόμασε το πρόγραμμα του σε Photoshop και έπειτα από συμφωνία με την κατασκευάστρια εταιρία σαρωτών Barneyscan, το πρόγραμμα διανεμήθηκε μαζί με μερικούς σαρωτές. Συνολικά διανεμήθηκαν 200 αντίγραφα του προγράμματος.

Εν τω μεταξύ, ο Τζον ταξίδεψε στο Σίλικον Βάλεϊ και παρουσίασε το πρόγραμμα του στους μηχανικούς της Apple και στην Adobe. Και οι δύο παρουσιάσεις ήταν επιτυχείς, καθώς η Adobe αποφάσισε να αγοράσει την άδεια να διανείμει το πρόγραμμα τον Σεπτέμβριο του 1988. Η επόμενη έκδοση του προγράμματος, η Photoshop 1.0, κυκλοφόρησε το 1990 αποκλειστικά για συστήματα Macintosh και είχε μέγεθος 1.44 MB.

3. Σχέδιο δράσης πτυχιακής

Σε αυτό το σημείο θα αναλύσουμε τη τεχνολογία αιχμής που χρησιμοποιήσαμε για την επίτευξη αυτής της πτυχιακής εργασίας καθώς και το σχέδιο δράσης το οποίο αρχικά αναπτύξαμε προκειμένου να έχουμε τα επιθυμητά αποτελέσματα.

3.1 State of the art

Όταν αναφερόμαστε στο state of art εννοούμε πολύ απλά τις τεχνολογίες αιχμής που έχουν χρησιμοποιηθεί για την διεκπεραίωση της εργασίας μας.
Για την κατασκευή μιας web εφαρμογης καθένας κατασκευαστής μπορεί να επιλέξει από μια μεγάλη γκάμα από προγράμματα για να δουλέψει. Σε κάθε κομβικό σημείο της παραγωγής μιας ιστοσελίδας οι επιλογές σε προγράμματα και τεχνολογίες είναι άπειρες.

Στην δικία μας περίπτωση οι επιλογές μας σε συγκεκριμένα προγράμματα είχαν να κάνουν με την εμπειρία μας πάνω σε αυτά καθώς και με την κρίση μας ότι τα συγκεκριμένα θα ήταν ιδανικά για μια λειτουργική και εύχρηστη ιστοσελίδα.

Το πρόγραμμα που χρησιμοποιήσαμε για να γράψουμε τον κώδικα μας ήταν όπως προαναφέραμε ο Sublime editor.

Τέτοια προγράμματα συγγραφής κώδικα υπάρχουν δεκάδες και όλα μπορούμε να πούμε ότι είναι ιδιαίτερα αξιόλογα. Τι ήταν όμως αυτό που μας ώθησε στην επιλογή του συγκεκριμένου?

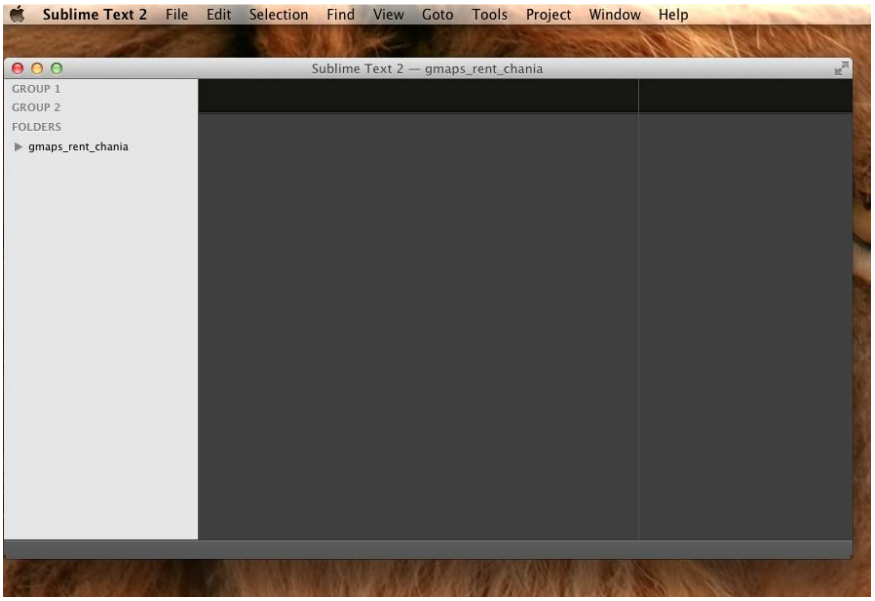
Η μεγάλη ευκολία που παρουσιάζει κατά το development και το γεγονός ότι είναι γρήγορο και light weight. Επιπλέον η προσθήκη extra plugins του επιτρέπουν να επεκτείνει τισ δυνατότητες του.

4. Κύριο Μέρος Της Πτυχιακής

4.1 Εγκατάσταση του sublime

Για την εγκατάσταση του sublime θα πρέπει να επισκεφτούμε την επίσημη σελίδα του http://www.sublimetext.com/ και να κατεβάσουμε την τελευταία έκδοση για το λειτουργικό σύστημα που μας ενδιαφέρει.

Στην συνέχεια κάνουμε την εγκατάσταση. Το λειτουργικό συστημα στο οπόιο δουλεύουμε στην προκειμένη είναι το Mac OS X 10.7.3 (Lion).



4.2 Εγκατάσταση της γλώσσας και των gems

Η εγκατάσταση της γώσσας (Ruby) στο Mac OS X, θα γίνει με τη βοήθεια του RVM (http://rvm.io/rvm/install). Το RVM ειναι ενα command-line εργαλείο που επιτρέπει την εύκολη

εγκατάσταση και διαχείρηση πολλών εκδόσεων της Ruby στο ίδιο υπολογιστή. Αυτό θα γίνει με την εντολή:

user$\curl -sSL https://get.rvm.io | bash -s stable

Στη συνέχεια εκτελούμε:
user$ rvm list known

Και βλέπουμε τις διαθέσιμες εκδόσεις της γλώσσας για να διαλέξουμε.

```
# MRI Rubies

[ruby-]1.9.3[-p545]

[ruby-]2.0.0-p353

[ruby-]2.0.0[-p451]

[ruby-]2.1[.1]

[ruby-]2.1-head

ruby-head
```

Διαλέγουμε την τελευταία πιο πρόσφατη έκδοση και εκτελούμε:

user$ rvm install 2.1

```
Checking requirements for opensuse.

Requirements installation successful.

Installing Ruby from source to: /home/mpapis/.rvm/rubies/ruby-2.1.1,

this may take a while depending on your cpu(s)...

Install of ruby-2.1.1 - #complete
```

Πλέον έχουμε εγκαταστήσει επιτυχώς την Ruby 2.1 με το gem set tool.

user$ ruby -v

```
ruby 2.1.1p76 (2014-02-24 revision 45161) [x86_64-darwin11.3.0]
```

user$ gem -v

```
2.3.0
```

4.2.1 Εγκατάσταση των gems που θα χρησιμοποιήσουμε

Τα gem όπως αναφέραμε είναι μικρές βιβλιοθήκες που μας βοηθούν στην δημιουργία της εφαρμογής μας. Μπορούμε να τα κάνουμε εγκατάσταση είτε ένα ένα εκτελώντας κάθε φορά την εντολή:

```
user$ gem install {name-of-the-gem}
```

Είτε γράφοντας όλα τα gems που θέλουμε μέσα σε ένα αρχείο (απλό text file) με όνομα Gemfile με την εξής μορφή:

```
source :rubygems

gem 'sinatra'
gem 'thin'
gem 'rack_csrf'
gem 'mongoid', "2.4.9"
gem 'bson_ext'

group :development do
  gem 'shotgun'
end
```

Και στη συνέχεια εκτελούμε την εντολή

```
user$ bundle install
```

για να εγκατασταθούν όλα τα gems όπως τα έχουμε ορίσει.

Το framework που θα χρησιμοποιήσουμε είναι και αυτό ένα gem (sinatra).
Ο server (thin) και η βάση (mongoDB) είναι επίσης gems.

4.2.2 Προετοιμασία σκελετού του web application

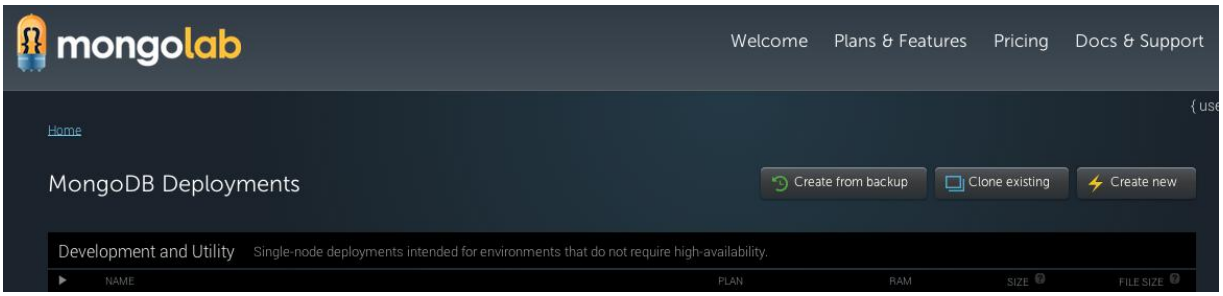
Το app έχει την εξής μορφή:

```
> app_gmaps_rent
        > public_dir
                > js_dir
                        app.js
                > css_dir
                        style.css
                > img_dir
                ....
        > views_dir
                index.erb
                ....
        app.rb
        config.ru
        Gemfile
        Gemfile.lock
        mongoid.yml
        Procfile
```

4.2.3 Δημιουργία της βάσης δεδομένων (mongolab.com)

Η βαση δεδομένων μας "κατοικεί" στο cloud και συγκεκριμένα στην πλατφόρμα mongolab. Το mongolab.com είναι μια υπηρεσία που προσφέρει τη δημιουργία και διαχείρηση της βάσης δεδομένων δίνοντάς μας τη δυνατότητα να συνδεθούμε σε αυτή remotely από το application μας.

Αφού δημιουργήσουμε τον λογαριασμό μας μπορούμε πολύ εύκολα από το panel να φτιάξουμε (δωρεάν) την βάση μας.
Αυτό γίνεται από το κουμπί ***Create new*** στα δεξιά όπως βλέπουμε στην παραπάνω εικόνα.



Δίνουμε στη βάση μας ένα όνομα, για τη συγκεκριμένη διαλέγουμε το ***testing_dat*** και όπως βλέπουμε παρακάτω η βάση είναι έτοιμη. Μπορούμε να συνδεθούμε σε αυτή με τα παρακάτω:

```
mongo ds039950.mongolab.com:39950/testing_dat -u <dbuser> -p <dbpassword>
mongodb://<dbuser>:<dbpassword>@ds039950.mongolab.com:39950/testing_dat
```

mongod 2.4.11

4.2.4 Τελευταιες ρυθμιίσεις

Αφού έχουμε λοιπόν την βάση έτοιμη δε μένει παρά να την ενώσουμε με τον σκελετό του application. Αυτό θα γίνει μέσα από το αρχείο mongoid.yml στο project μας.

```
development:
  host: ds031217.mongolab.com
  port: 33619
  username: tester_user
  password: pass123
  database: testing_dat

production:
  uri:
mongodb://testing_dat:pass123t@ds033617.mongolab.com:33619/testin
g_dat
```

Έχουμε δύο μπλοκ το development και το production. Και τα δυο περιέχουν τις οδηγίες για να συνδεθεί ηεφαρμογη στη βάση ανάλογα με το στάδιο στο οποίο βρίσκεται. Στην περίπτωση μας είναι η ίδια βάση.

Τα gems τα οποία χρειαζόμαστε για να την επίτευξη του στόχου μας είναι:

```
gem 'sinatra'
gem 'thin'
gem 'rack_csrf'
gem 'mongoid', "2.4.9"
gem 'bson_ext'

group :development do
  gem 'shotgun'
end
```

Και όλα τους γράφονται στο GEMFILE κατά αυτόν τον τρόπο.

Το gem sinatra, υπάρχει για να συμπεριληφθεί το βασικό web framework πάνω στο οποίο θα υλοποιηθεί η εφαρμογή μας.

Το thin gem είναι ο server ο οποίος θα σερβίρει το web app.

Το rack_csrf gem μας βοηθά να δημιουργούμε φόρμες που προστατεύονται από cross site request forgery. Δημιουργεί ένα unique token στο render της φόρμας και περιμένει αυτό το token κατα το submit της.

Τα gems mongoid και bson_ext χρειάζονται για την επικοινωνία της εφαρμογής και της βάσης δεδομένων.

Το shotgun gem μας βοηθά στο development. Συγκεκριμένα στις online αλλαγές τoy κώδικα (χωρίς να χρειάζεται refresh ο server).

Όλα τα gems που χρησιμοποιούμε τα κάνουμε require μέσα στο app.rb.

Τρέχουμε την εντολή ***bundle install*** στο root path του app μας και όλα τα απαραίτητα gems γίνονται εγκατάσταση. Είμαστε έτοιμοι για την ανάπτυξη του project μας.

4.3 Κατασκευαστικό μέρος

4.3.1 Backend - Κωδικας

Στο backend κομμάτι ο κώδικας μας έχει ως εξής:

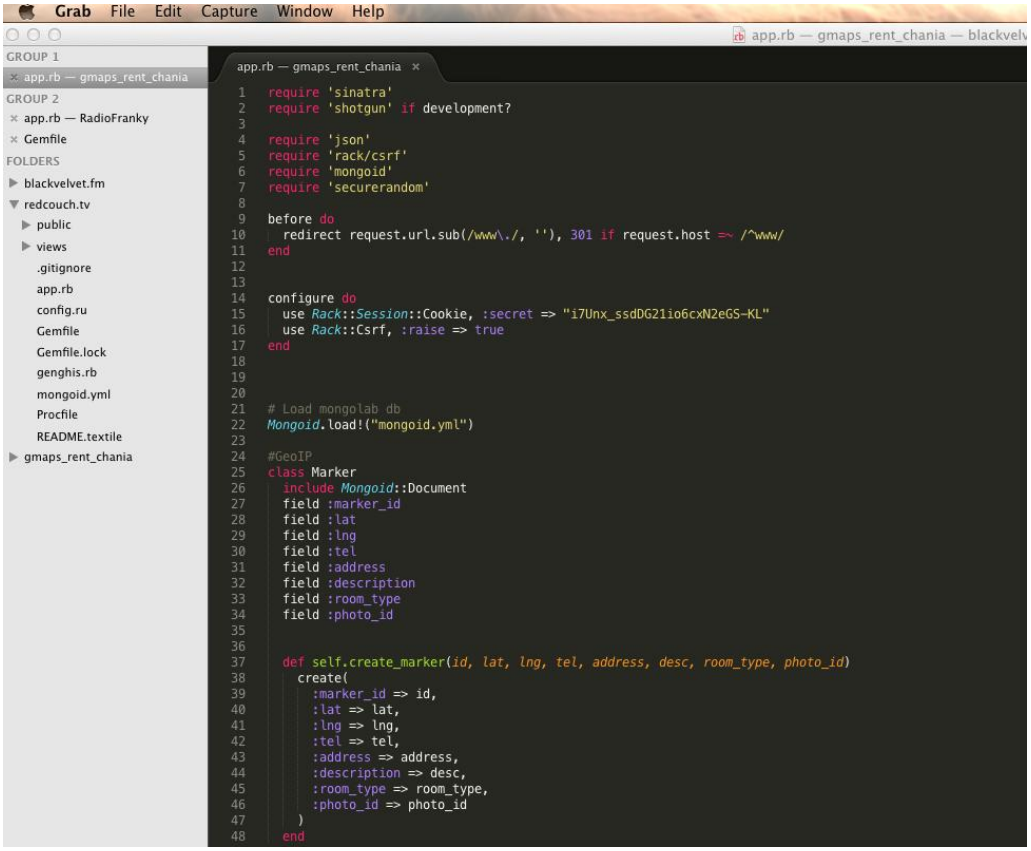
Ορίζουμε τα routes. Τις διαδρομές δηλαδή που αν πληκτολογηθούν θα τρέξει ο κατάλληλος κώδικας και θα γίνουν οι απαραίτητες ενέργειες. Τα routes μας είναι:

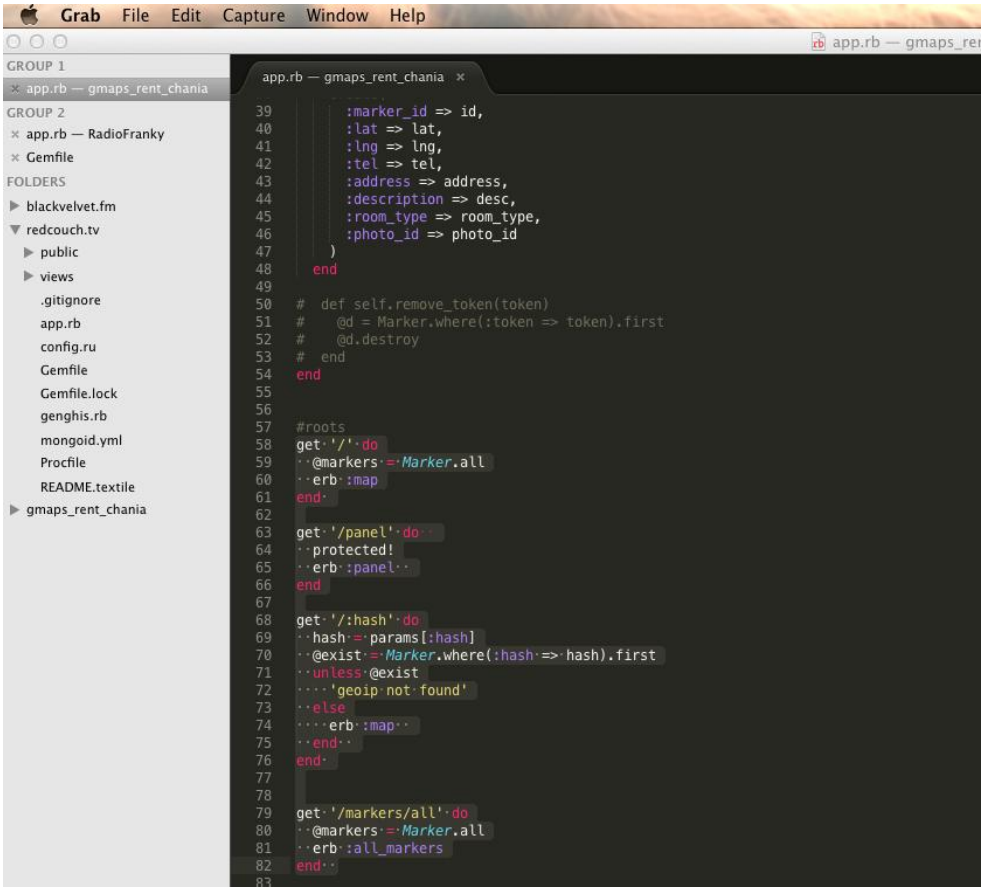
```
get '/' do
    ...
    @points = Points.all
    erb :index_map
end

get '/panel' do
    ...
    erb :panel
end

get '/markers/all' do
    ...
end
```

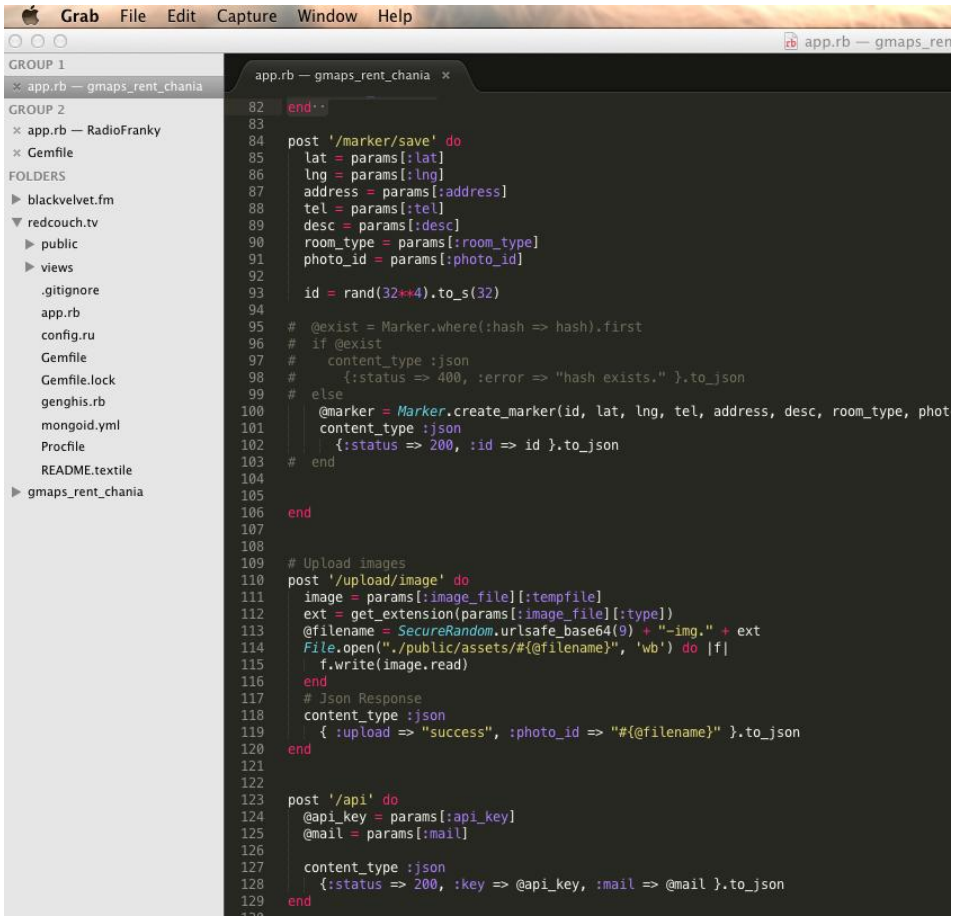
Παραπάνω έχουμε μεθόδους όπου ενεργοποιούνται οταν γίνονται GET requests ως προς τα συγκεκριμένα routes. Στο τέλος κάθε μεθόδου καλούμε το template το οποίο θα εμφανιστεί στον browser του χρήστη, μέσα στο οποίο περνούμε αυτόματα ως ορίσματα όποια objects έχουν προκύψει από την προγραμματιστική μας λογική.

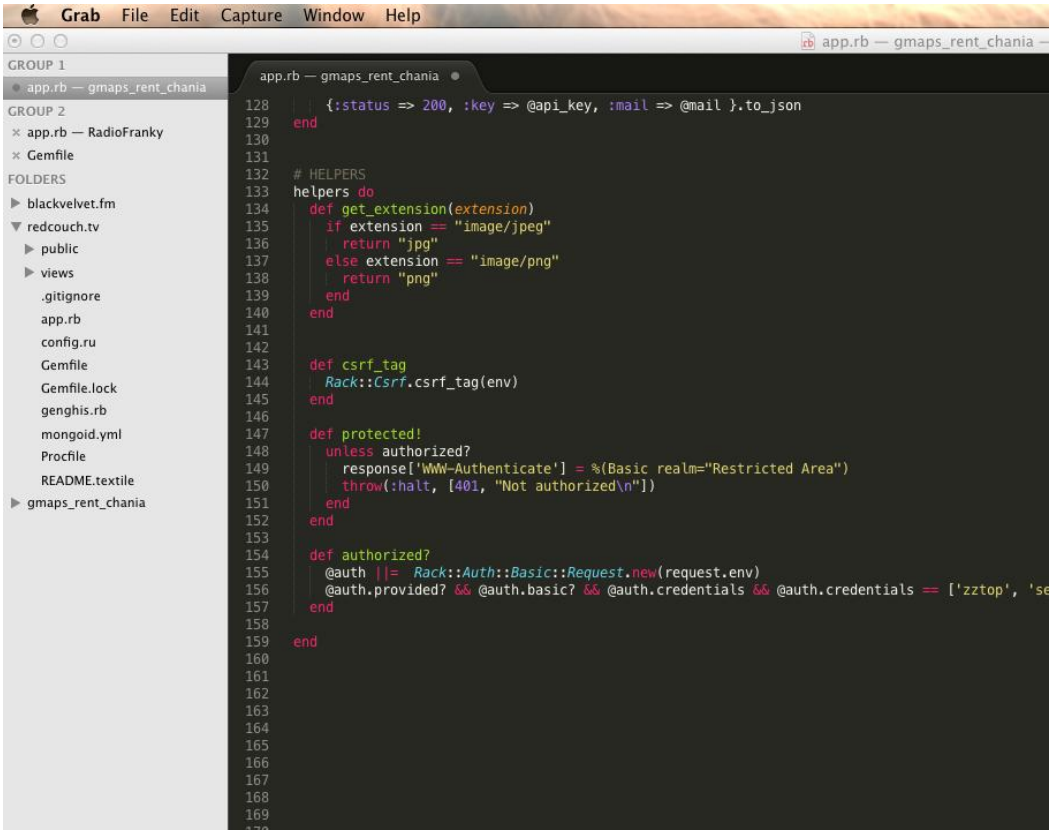
Όλη η λογική του framework είναι έτσι. Τα υπόλοιπα είναι βοηθητικές classes, helpers κτλ. Παρακάτω ακολουθούν εικόνες απο τον κώδικα. Φαίνεται η δομή του project και τα πεδία της βάσης τα οποία είναι:

marker_id
lat
lng
tel
address
description
room_type
photo_id

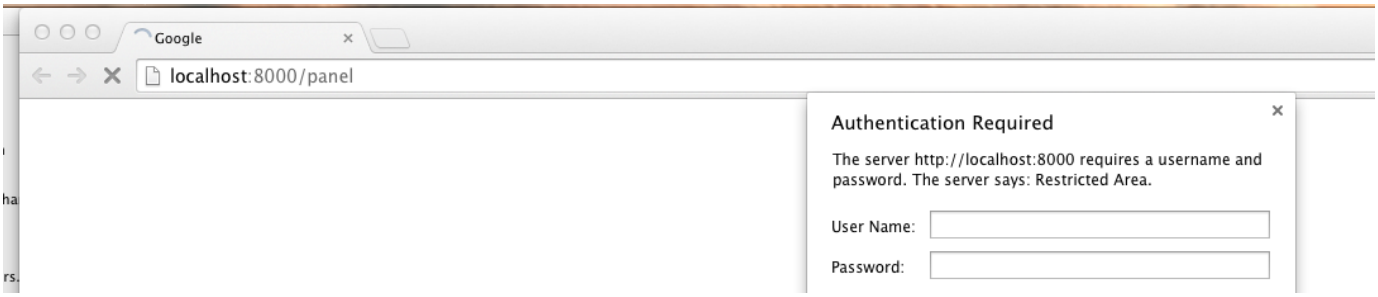



Grab File Edit Capture Window Help
app.rb — gmaps_rent_chania
GROUP 1
app.rb — gmaps_rent_chania
GROUP 2
app.rb — RadioFranky
Gemfile
FOLDERS
blackvelvet.fm
redcouch.tv
public
views
.gitignore
app.rb
config.ru
Gemfile
Gemfile.lock
genghis.rb
mongoid.yml
Procfile
README.textile
gmaps_rent_chania
app.rb — gmaps_rent_chania
:marker_id => id,
:lat => lat,
:lng => lng,
:tel => tel,
:address => address,
:description => desc,
:room_type => room_type,
:photo_id => photo_id
)
end
def self.remove_token(token)
@d = Marker.where(:token => token).first
@d.destroy
end
end
#roots
get '/' do
@markers = Marker.all
erb :map
end
get '/panel' do
protected!
erb :panel
end
get '/:hash' do
hash = params[:hash]
@exist = Marker.where(:hash => hash).first
unless @exist
'geoip not found'
else
erb :map
end
end
get '/markers/all' do
@markers = Marker.all
erb :all_markers
end


Grab File Edit Capture Window Help
app.rb — gmaps_rent_chania
GROUP 1
app.rb — gmaps_rent_chania
GROUP 2
app.rb — RadioFranky
Gemfile
FOLDERS
blackvelvet.fm
redcouch.tv
public
views
.gitignore
app.rb
config.ru
Gemfile
Gemfile.lock
genghis.rb
mongoid.yml
Procfile
README.textile
gmaps_rent_chania
app.rb — gmaps_rent_chania
end
post '/marker/save' do
lat = params[:lat]
lng = params[:lng]
address = params[:address]
tel = params[:tel]
desc = params[:desc]
room_type = params[:room_type]
photo_id = params[:photo_id]
id = rand(32**4).to_s(32)
# @exist = Marker.where(:hash => hash).first
# if @exist
# content_type :json
# {:status => 400, :error => "hash exists." }.to_json
# else
@marker = Marker.create_marker(id, lat, lng, tel, address, desc, room_type, phot
content_type :json
{:status => 200, :id => id }.to_json
end
end
Upload images
post '/upload/image' do
image = params[:image_file][:tempfile]
ext = get_extension(params[:image_file][:type])
@filename = SecureRandom.urlsafe_base64(9) + "-img." + ext
File.open("./public/assets/#{@filename}", 'wb') do |f|
f.write(image.read)
end
Json Response
content_type :json
{ :upload => "success", :photo_id => "#{@filename}" }.to_json
end
post '/api' do
@api_key = params[:api_key]
@mail = params[:mail]
content_type :json
{:status => 200, :key => @api_key, :mail => @mail }.to_json
end

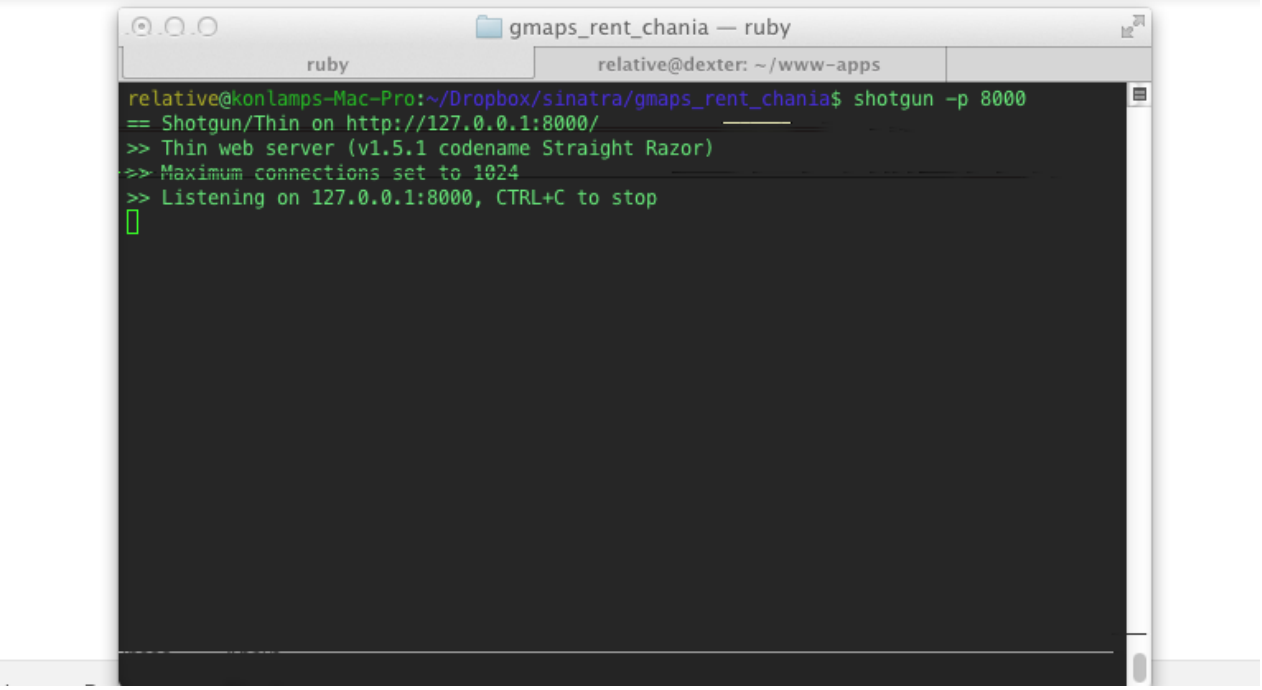


Τα routes μας είναι ουσιαστικά 3. Το κεντρικό index page ‘/’ , το panel page ‘/panel κατω από το οποίο, ο administrator μπορεί να εισάγει νέα δεδομένα και να τα διαχειριστεί. Το panel view είναι ‘προστατευμένο’ και ο administrator εισέρχεται με basic auth δίνοντας username και password.



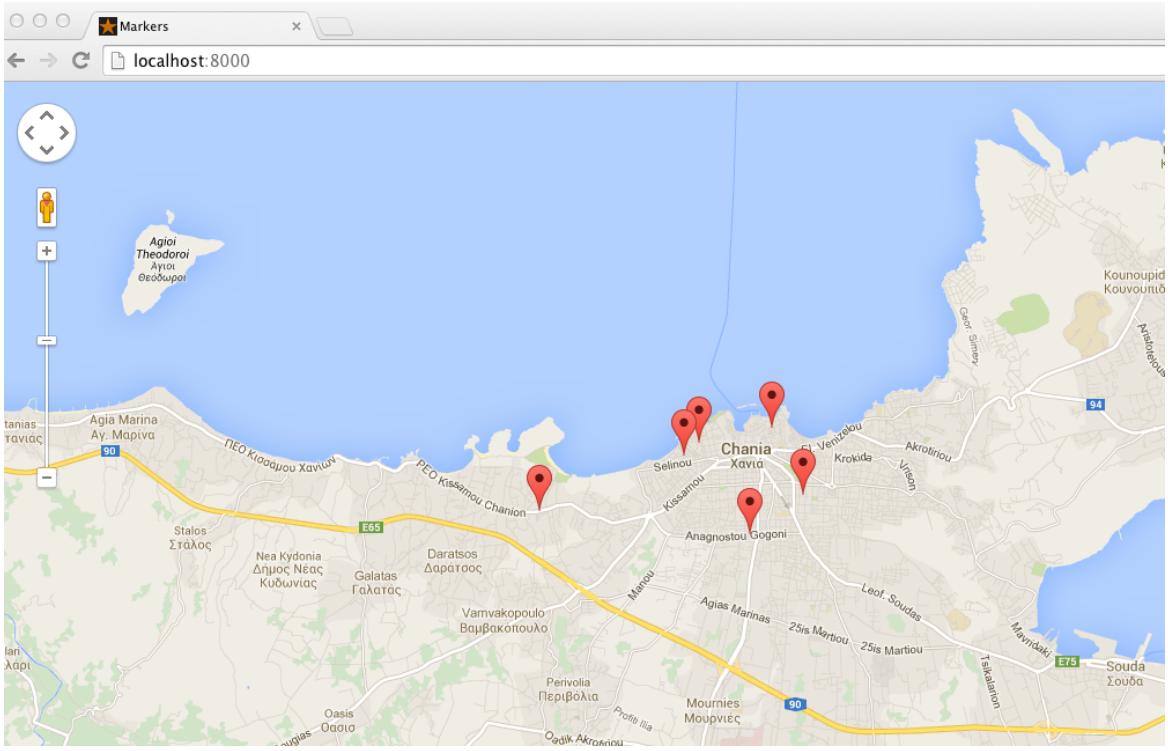
Ο server μας έχει ενεργοποιηθεί με την εντολή:

$ shotgun -p 8000

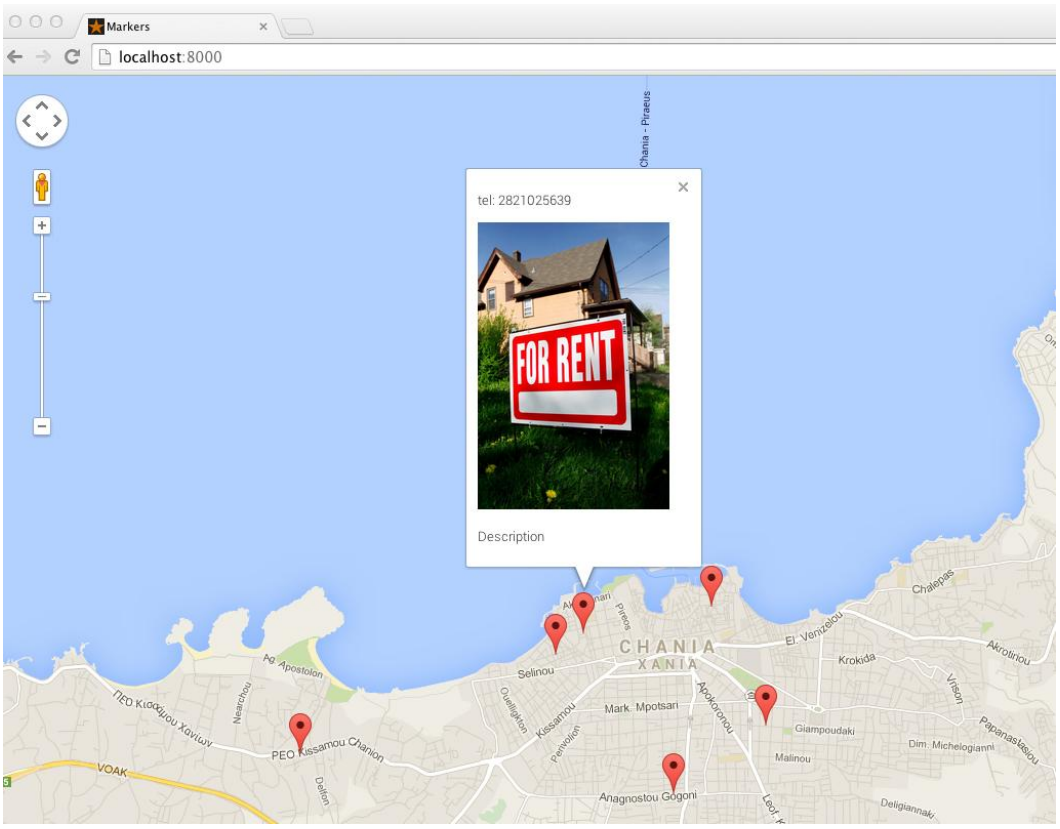


4.3.2 Front-end και εισαγωγή χάρτη

Στο frontend μας, στο index view, σε αυτό που ο χρήστης αντικρίζει όταν επισκεφτεί την διεύθυνση http://localhost:8000/, είναι ένα google map full screen με τους markers που έχουν ήδη εισαχθεί.



Κλικάροντας πάνω στον marker βλέπουμε τις πληροφορίες που έχουμε εισάγει. Και μια ενδεικτική φωτογραφία του σημείου.



Ο κώδικας για το frontend , το styling (CSS) και η εισαγωγή του χάρτη ακολουθούν στις επόμενες εικόνες:

map.erb — blackvelvet.fm, redcouch.tv, gmaps_rent_chania

map.erb

```
<div id="mapCanvas2" class="frontmap"></div>

<script type="text/javascript" charset="utf-8">

var locations = [
  <% @markers.each_with_index do |marker, index| %>
    [
      <% if marker.photo_id %>'<%= marker.photo_id %>',<% else %>'<%= "house-for-rent.jpg" %>',<% end %>
      <%= marker.lat %>,
      <%= marker.lng %>,
      <%= marker.tel %>
    ]<% unless index == @markers.size - 1 %>,<% end %>
  <% end %>
];

var marker, i;


function initialize() {
 var latLng = new google.maps.LatLng( 35.513114, 24.017792 );
 var map = new google.maps.Map(document.getElementById('mapCanvas2'), {
   zoom: 13,
   center: latLng,
   mapTypeId: google.maps.MapTypeId.ROADMAP
 });
 for (i = 0; i < locations.length; i++) {
  console.log(parseFloat(locations[i][1]) + " - " + parseFloat(locations[i][2]));
   marker = new google.maps.Marker({
     position: new google.maps.LatLng(parseFloat(locations[i][1]), parseFloat(locations[i][2])),
     map: map
   });
   (function(marker, i) {
     // add click event
     google.maps.event.addListener(marker, 'click', function() {
         infowindow = new google.maps.InfoWindow({
             content: '<p>tel: '+ locations[i][3] +'</p><img style="height: 280px" src="/assets/'+locations[i][0]+'" alt="img"><p>
               Description</p>'
         });
         infowindow.open(map, marker);
     });
   })(marker, i);
 }
}

google.maps.event.addDomListener(window, 'load', initialize);

</script>
```

Η εισαγώγή του χάρτη και ο κώδικας για την απεικόνιση των markers.

app.js — blac

map.erb

```
/* js */

var geocoder = new google.maps.Geocoder();
var map;
var marker;
var fields_ok = false;

function geocodePosition(pos) {
  geocoder.geocode({
    latLng: pos
  }, function(responses) {
    if (responses && responses.length > 0) {
      updateMarkerAddress(responses[0].formatted_address);
    } else {
      updateMarkerAddress('Cannot determine address at this location.');
    }
  });
}

function updateMarkerStatus(str) {
  // document.getElementById('markerStatus').innerHTML = str;
}

function updateMarkerPosition(latLng) {
  document.getElementById('hidden_lat').value = latLng.lat();
  document.getElementById('hidden_lng').value = latLng.lng();

//  document.getElementById('lat').innerHTML = latLng.lat();
//  document.getElementById('long').innerHTML = latLng.lng();
}

function updateMarkerAddress(str) {
  document.getElementById('closest-address').innerHTML = str;
}

function initialize() {
  var latLng = new google.maps.LatLng(35.513114, 24.017792);
  // Global var
  map = new google.maps.Map(document.getElementById('mapCanvas'), {
    zoom: 14,
    center: latLng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });
  // Global var
  marker = new google.maps.Marker({
    position: latLng,
    title: 'Dragg me',
    map: map,
    draggable: true
  });

  // Update current position info.
  updateMarkerPosition(latLng);
  geocodePosition(latLng);

  // Add dragging event listeners.
  google.maps.event.addListener(marker, 'dragstart', function() {
    updateMarkerAddress('Dragging...');
  });
```

Απόσπασμα από το css.

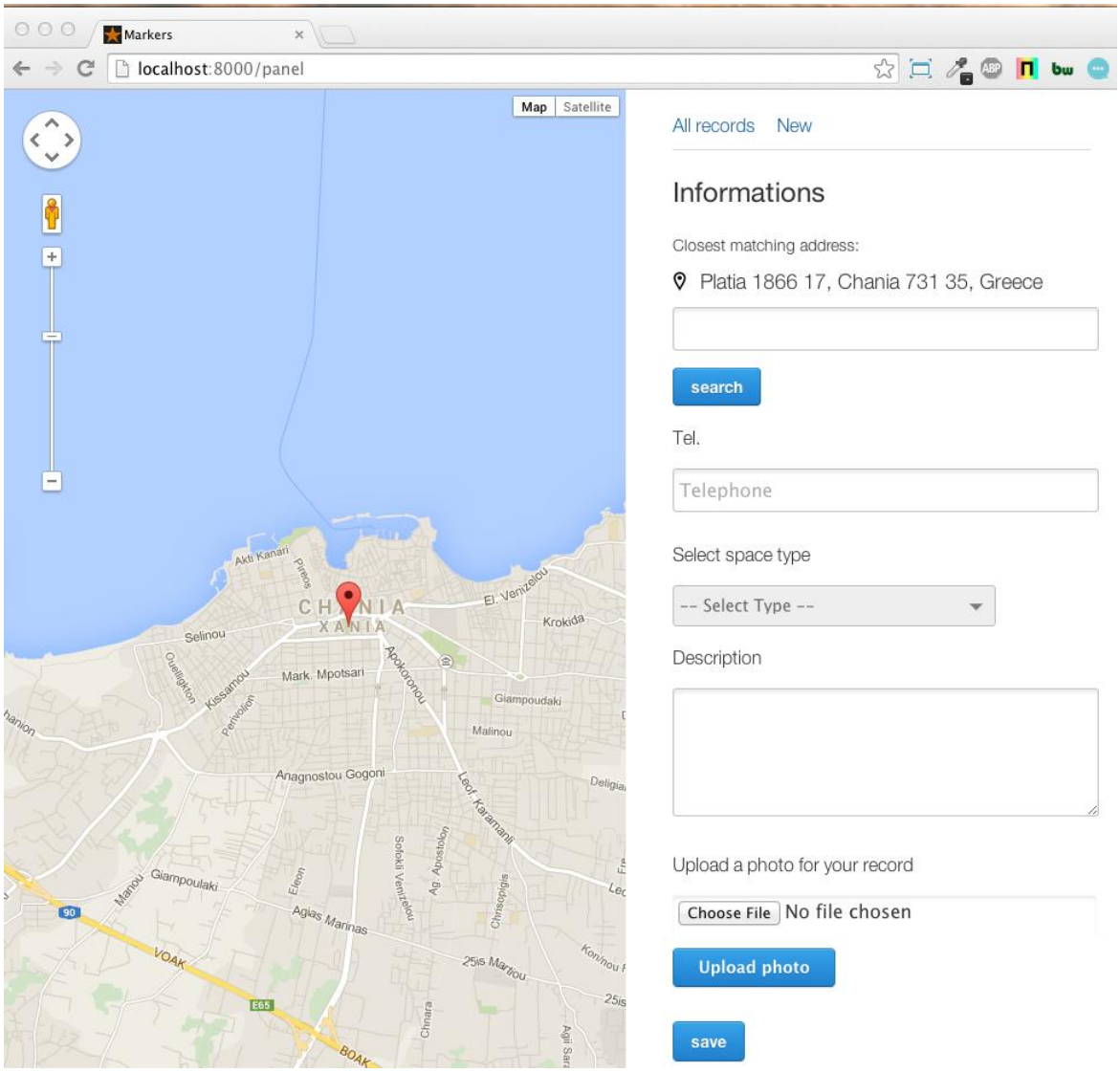
style.css — Di

map.erb

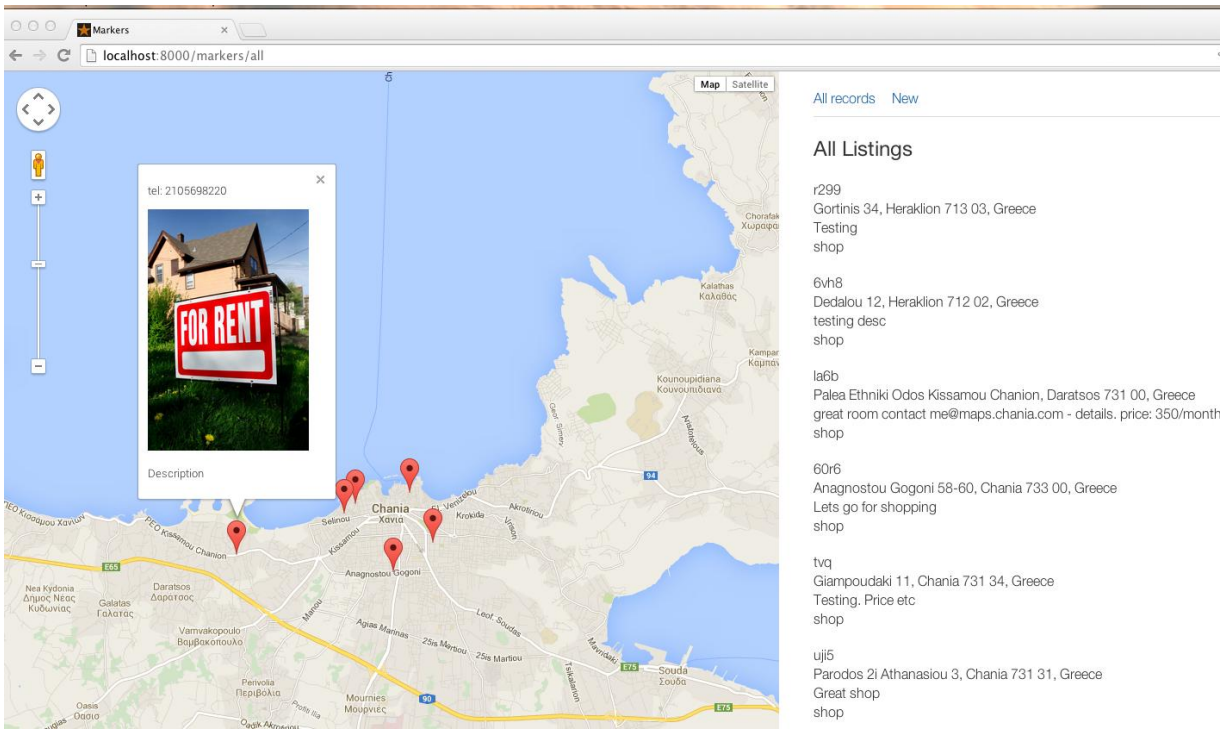
```
1   html, body { margin: 0 auto; }
2
3   body {
4       font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;
5       color: #333;
6       font-size: 16px;
7   }
8
9   a { text-decoration: none; color: #185fa5; }
10  /*a:hover { text-decoration: none; color: #185fa5; }*/
11
12  .pseudo-link { color: #185fa5; cursor: pointer; }
13
14  ul { padding: 0; margin: 0; }
15  li { list-style: none; }
16
17
18  h3 {
19      font-size: 24px;
20      font-weight: 200;
21      margin: 20px 0;
22      line-height: 1.2;
23      position: relative;
24  }
25
26  .frontmap {
27      position: absolute;
28      width: 100%;
29      height: 100%;
30  }
31
32  .panelmap{
33      position: absolute;
34  }
35
36  .mapCanvas {
37      width: 55%;
38      height: 100%;
39      float: left;
40      display: inline-block;
41  }
42
43  .container {
44      padding: 20px 40px;
45      margin-bottom: 60px;
46      display: table;
47      font-weight: 300;
48      line-height: 1.4;
49      float: right;
50      width: calc(45% - 80px);
51  }
52
53  .sectionTitle{
54      font-weight: 400;
55  }
56
57  .photos {
58      display: inline-block;
59  }
60
```

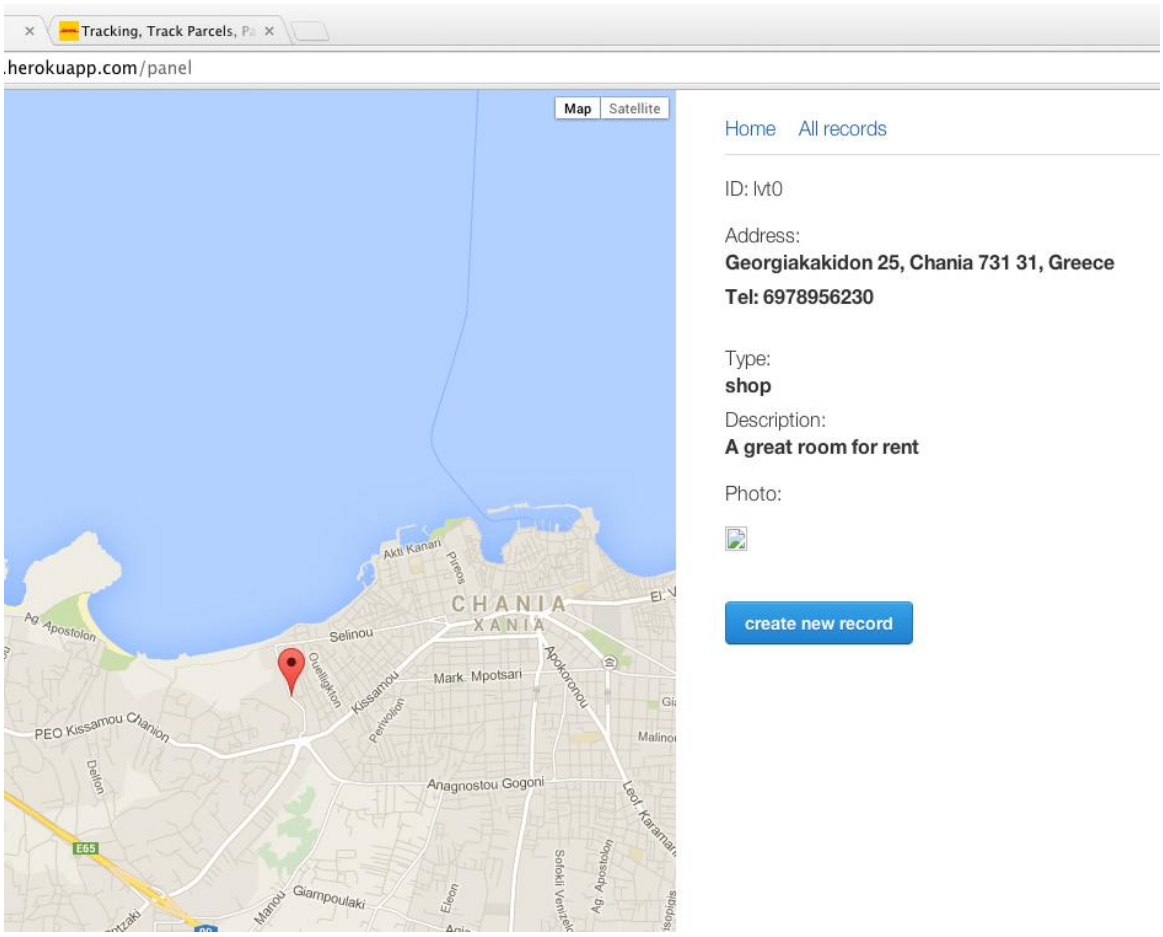
4.3.3 Μέρος διαχειριστή

Το panel στο εσωτερικό του. Ακολουθούν εικόνες:



Εδώ μπορούμε να δούμε όλες τις εγγραφές που έχουμε κάνει κατα χρονολογική σειρά.





Μια νέα εγγραφή μόλις πραγματοποιήθηκε ασύγχρονα.

4.3.4 Deployment (heroku.com)

Το κεφάλαιο αυτό περιγράφει την διαδικασία του deployment του application. Δηλαδή το που θα φιλοξενηθεί η εφαρμογή για το τελικό στάδιο της παραγωγής (production phase), ώστε να είναι ορατή σε οποιδήποτε θελήσει να την επισκεφτεί, αλλά και η διαδικασια μεχρι να φτάσουμε εκεί.

Για αυτό το σκοπό διαλέξαμε το heroku.com . Ένα platform as a service όπου κάποιος μπορεί να φιλοξενήσει το web page/app του (δωρεάν).

Το heroku μας δίνει την δυνατότητα να αυξομειώσουμε τα resources του node (virtual machine) μας on the fly (paid features). Γενικότερα είναι ικανό να μας γλιτώσει από πολύ χρονοβόρες διαδικασίες και κόπο. Διατίθεται για πολλά έιδη εφαρμογών - γλωσσών, μεταξυ αυτών και η ruby.

Αφού έχουμε κάνει τον απαραίτητο λογαριασμό μας εγκαθιστούμε το command line tool ***heroku*** και το ***GIT*** ώστε να στείλουμε την εφαρμογή μας στο repository. Τρέχουμε τα παρακάτω στο terminal μας:

```
$ heroku create -s cedar ptyxiaki-am1615
```

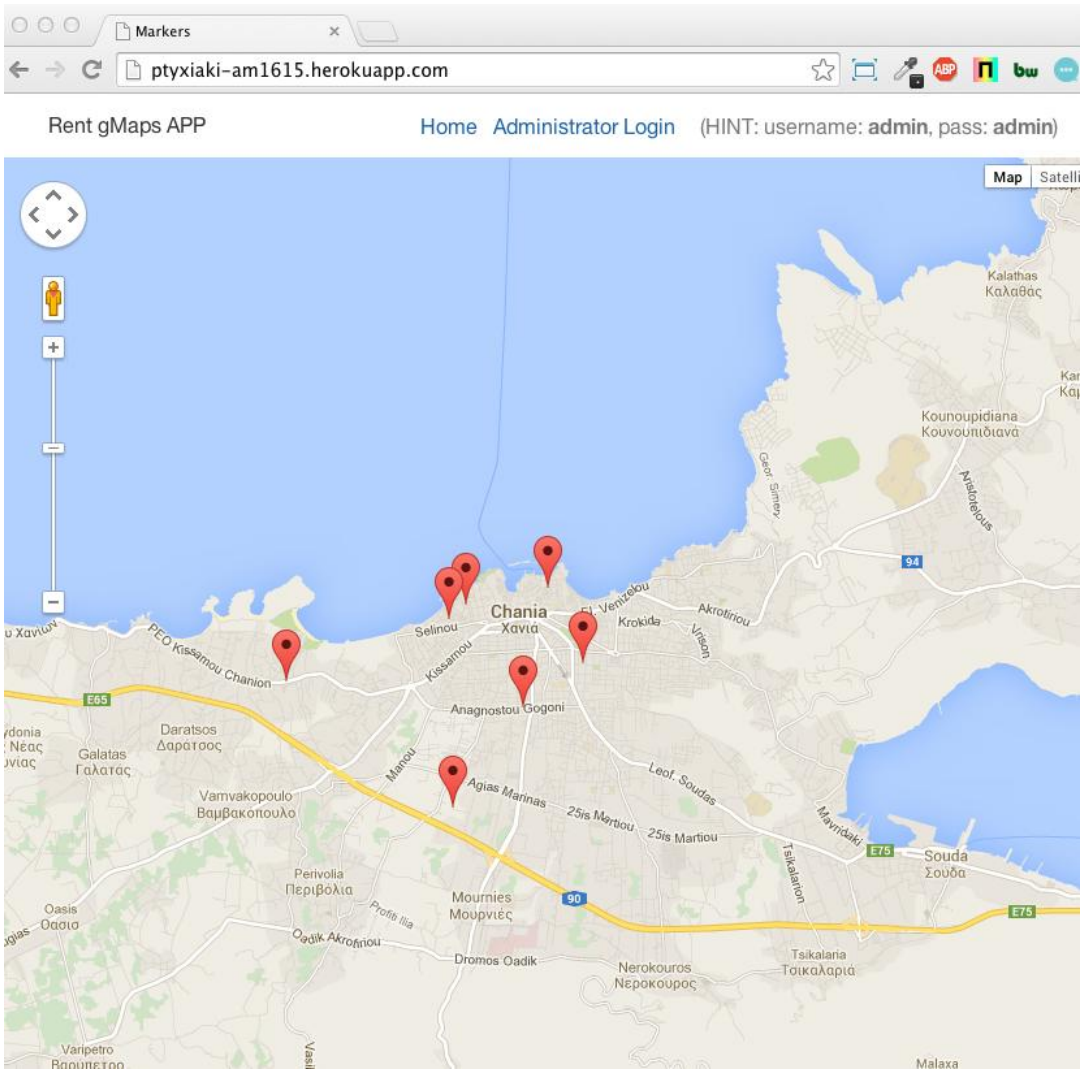
Το Heroku δημιουργεί ένα repo κάτω από την διεύθυνση:

Creating ptyxiaki-am1615... done, region is us
http://ptyxiaki-am1615.herokuapp.com/ | git@heroku.com:ptyxiaki-am1615.git
Git remote heroku added

```
$ git init
$ git add .
$ git commit -am "first commit"
$ git push heroku master
```

Πλέον η εφαρμογή είναι προσβάσιμη για τον οποιονδήποτε στην διεύθυνση

http://ptyxiaki-am1615.herokuapp.com/



5. Συμπεράσματα

Στα πλαίσια αυτής της πτυχιακής εργασίας, υλοποιήθηκε μια πλατφόρμα για αποθήκευση των προς ενοικίαση καταστημάτων / χώρων μιας πόλης και την απεικόνηση τους γεωγραφικά. Η εφαρμογή αυτή θα μπορούσε να έχει διάφορες χρήσης και σε άλλα σενάρια. Γενικά εξυπηρετεί την γεωγραφική απεικόνιση σημείων της πόλης και την διαχείρηση τους. Πιο συγκεκριμένα στόχος ήταν να δημιουργηθεί μια ιστοσελίδα φιλική προς τον διαχειριστή και εύκολη στην καταχώρηση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

http://www.mongodb.com/architecture

http://docs.mongolab.com/

https://devcenter.heroku.com/

http://rvm.io/rvm/install

http://smallbusiness.chron.com/create-company-logo-photoshop-1680.html

http://en.wikipedia.org/wiki/Ruby_(programming_language)

http://en.wikipedia.org/wiki/Sinatra_(software)

http://www.sinatrarb.com/

http://code.macournoyer.com/thin/

http://nefeli.lib.teicrete.gr/browse/stef/epp/2013/KritsotakiDimitra,MavrakiEleni/document-1369032474-486955-21724.tkl

http://stackoverflow.com/

http://en.wikipedia.org/wiki/JavaScript

http://en.wikipedia.org/wiki/Cascading_Style_Sheets

http://en.wikipedia.org/wiki/HTML