ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

ΑΝΑΠΤΥΞΗ WEB ΕΦΑΡΜΟΓΗΣ ΓΙΑ ON-LINE ΚΡΑΤΗΣΕΙΣ ΔΩΜΑΤΙΩΝ ΜΕ ΧΡΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ RUBY ΚΑΙ ΤΟΥ WEB FRAMEWORK RUBY ON RAILS

Ο ΣΠΟΥΔΑΣΤΗΣ:

Δρακόντης Παναγιώτης Τ-1515

ΕΠΙΒΛΕΠΩΝ

Χρήστος Σωμαράς

ΛΑΡΙΣΑ 2010

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Λάρισα …/…../2010

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

ΕΥΧΑΡΙΣΤΙΕΣ

Ξεκινώντας θα ήθελα να ευχαριστήσω θερμά τον κο. Χρήστο Σωμαρά που εκτός από την καθοδήγηση, την υπομονή του και την εμπιστοσύνη που μου έδειξε αναθέτοντας μου την πτυχιακή, μου έδειξε τον τρόπο για την σωστή ανάπτυξη μιας εφαρμογής. Χωρίς την πολύτιμη βοήθειά του το τελικό αποτέλεσμα της εφαρμογής θα ήταν ελλιπές και χαμηλότερου επιπέδου. Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την συμπαράσταση και την κατανόηση που έδειξαν κατά την διάρκεια της ενασχόλησης μου με την πτυχιακή εργασία.

Με εκτίμηση Δρακόντης Παναγιώτης

Αφιερωμένη στους γονείς μου για την υπομονή,
την στήριξη και την εμπιστοσύνη που μου έδειξαν,
καθ' όλη την διάρκεια της φοίτησής μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Ο ταχύτατα αναπτυσσόμενος κόσμος της πληροφορικής και των τηλεπικοινωνιών σε συνδυασμό με τις νέες πιο ευέλικτες και "ανώδυνες" τεχνολογίες ανάπτυξης λογισμικού, καθιστούν απαραίτητη την ηλεκτρονική υπόσταση κάθε επιχείρησης, μιας και το διαδίκτυο έχει γίνει αναπόσπαστο κομμάτι του τρόπου ζωής μας.

Ο αναγνώστης του παρόντος συγγράμματος θα είναι σε θέση να κατανοήσει πλήρως της αρχές και την φιλοσοφία της αντικειμενοστραφούς γλώσσας προγραμματισμού Ruby και του Web Framework Ruby On Rails (RoR), επίσης θα γίνει επεξήγηση του κώδικα HTML και JavaScript που γράφηκε για την εφαρμογή και του συστήματος διαχείρισης βάσεων δεδομένων MySQL που χρησιμοποιήθηκε για την δημιουργία και αποθήκευση της βάσης δεδομένων. Επί πλέον θα γίνει εκτενής αναφορά στη λογική, την λειτουργία και στην απαραίτητη δομή που θα πρέπει να έχει μια on-line εφαρμογή ηλεκτρονικής κράτησης δωματίων τόσο από την πλευρά του διαχειριστή (ιδιοκτήτης) όσο και από την πλευρά του πλευρά του χρήστη (πελάτης).

ΕΙΣΑΓΩΓΗ

Η πορεία για την ολοκλήρωση της πτυχιακής εργασίας έγινε σταδιακά. Το πρώτο βήμα το οποίο έπρεπε να γίνει, το οποίο μάλλον ήταν και το πιο χρονοβόρο, ήταν η εκμάθηση και η κατανόηση της γλώσσας προγραμματισμού Ruby, μιας και δεν υπήρχε προηγούμενη επαφή με αυτή την γλώσσα, ώστε όταν θα ξεκινούσε η συγγραφή του κώδικα να αποφεύγονταν προβλήματα που αφορούσαν την ίδια την γλώσσα (σύνταξη, τύποι δεδομένων, ενσωματωμένες συναρτήσεις κτλ). Το επόμενο βήμα ήταν η αναζήτηση πληροφοριών στο διαδίκτυο για τον τρόπο λειτουργίας των on-line συστημάτων κρατήσεων δωματίων, μιας και ένα τέτοιο σύστημα θα έπρεπε να δημιουργηθεί. Έχοντας κάνει ένα προσχέδιο και γνωρίζοντας την Ruby σε αρκετά καλό επίπεδο, ξεκίνησε η συγγραφή του κώδικα για την Web εφαρμογή.

Στο 1ο κεφάλαιο θα γίνει ανάλυση του θεωρητικού μέρους ενός on-line καταστήματος και συγκεκριμένα ενός on-line ξενοδοχείου. Επίσης θα γίνει μια προσπάθεια θεωρητικής προσέγγισης της έννοιας του ηλεκτρονικού εμπορίου. Επιπλέον θα παρουσιαστούν τα πλεονεκτήματα και τα μειονεκτήματα που θα αποκομίσει μια επιχείρηση αποκτώντας ηλεκτρονική υπόσταση. Έτσι ώστε ο αναγνώστης να αποκτήσει μια γενική εικόνα για την παρουσίαση της εφαρμογής που θα ακολουθήσει στα επόμενα κεφάλαια.

Στα κεφάλαια 2 και 3 θα αφιερωθούν στην γλώσσα προγραμματισμού Ruby και του Web framework Ruby on Rails (RoR), όπου και θα παρουσιαστεί η φιλοσοφία, η δομή και οι λειτουργίες της γλώσσας και του framework. Επίσης θα γίνει και μια μικρή αναφορά της open source φιλοσοφίας που επικρατεί στους κύκλους της κοινότητας των προγραμματιστών της Ruby.

Το σύστημα διαχείρισης που χρησιμοποιήθηκε για την δημιουργία και αποθήκευση της βάσης που χρησιμοποιεί η εφαρμογή είναι η MySQL. Το 4ο κεφάλαιο θα αφιερωθεί στην παρουσίαση της MySQL και των γραφικών διασυνδέσεων χρήστη (Graphical User Interface – GUI) που χρησιμοποιήθηκαν για τον χειρισμό και την διαχείριση της βάσης δεδομένων.

Η γλώσσα που χρησιμοποιήθηκε κατά κόρον για την υλοποίηση της εφαρμογής είναι η Ruby, για το εικαστικό μέρος και για την δημιουργία του Web Interface χρησιμοποιήθηκαν οι

γλώσσες JavaScript και HTML. Στο 5ο κεφάλαιο θα γίνει παρουσίαση του κώδικα και της εφαρμογής, όπου για κάθε σημείο της εφαρμογής που θα παρουσιάζεται, θα παρατίθεται και θα αναλύεται ο κώδικας που "τρέχει" για το συγκεκριμένο σημείο. Επίσης θα παρουσιαστεί και η δημιουργία και η σύνδεση της βάσης δεδομένων με την εφαρμογή.

ΚΕΦΑΛΑΙΟ 1ο : ΗΛΕΚΤΡΟΝΙΚΟ ΕΜΠΟΡΙΟ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΟ ΚΑΤΑΣΤΗΜΑ

Ως ηλεκτρονικό εμπόριο ορίζεται το εμπόριο που πραγματοποιείται με ηλεκτρονικά μέσα, αποτελεί δηλαδή μια ολοκληρωμένη συναλλαγή που πραγματοποιείται μέσω διαδικτύου - internet χωρίς να είναι απαραίτητη η φυσική παρουσία των συμβαλλομένων μερών (δηλαδή του πωλητή και του αγοραστή).

1.1: Το ηλεκτρονικό εμπόριο

Ως ηλεκτρονικό εμπόριο ορίζεται το εμπόριο που πραγματοποιείται με ηλεκτρονικά μέσα βασίζεται δηλαδή στην ηλεκτρονική μετάδοση δεδομένων. Το ηλεκτρονικό εμπόριο αποτελεί έκφανση των λεγόμενων υπηρεσιών εξ αποστάσεως.(el.science.wikia.com)

Τα τελευταία χρόνια η γρήγορη ανάπτυξη της πληροφορικής έχει ανοίξει νέους ορίζοντες σε πολλούς τομείς της ζωής μας. Ένας από αυτούς είναι και το εμπόριο, που με την ανάπτυξη της πληροφορικής αποκτά μια νέα μορφή, την ηλεκτρονική.

Ένα μεγάλο μέρος επιχειρήσεων που τείνει να μετατραπεί σε πλειοψηφία έχουν αποκτήσει ηλεκτρονική υπόσταση στο διαδίκτυο. Επίσης οι περισσότερες τράπεζες δίνουν την δυνατότητα στους πελάτες της να διαχειρίζονται τον τραπεζικό τους λογαριασμό μέσω του διαδικτύου. Αν λάβουμε υπόψιν μας και την ανάπτυξη και εξάπλωση της πληροφορικής που είναι το κύριο μέσο για την επικοινωνία επιχείρησης – πελάτη, συμπεραίνουμε ότι ικανοποιείται η βασική αρχή του εμπορίου που είναι η προσφορά και η ζήτηση.

Ένας πολύ σημαντικός συντελεστής για την ανάπτυξη του ηλεκτρονικού εμπορίου είναι οι ηλεκτρονικές πληρωμές με όλες τις παραμέτρους που περιλαμβάνουν. Με τον όρο ηλεκτρονικές πληρωμές εννοούνται όλες τις διαδικασίες που ξεκινούν από την στιγμή που ο αγοραστής θα αποφασίσει ότι θέλει να αγοράσει κάποιο προϊόν ηλεκτρονικά και δίνει την εντολή για την έναρξη της διαδικασίας μέχρι την παραλαβή του προϊόντος που έχει παραγγείλει και την εξόφλησή του.

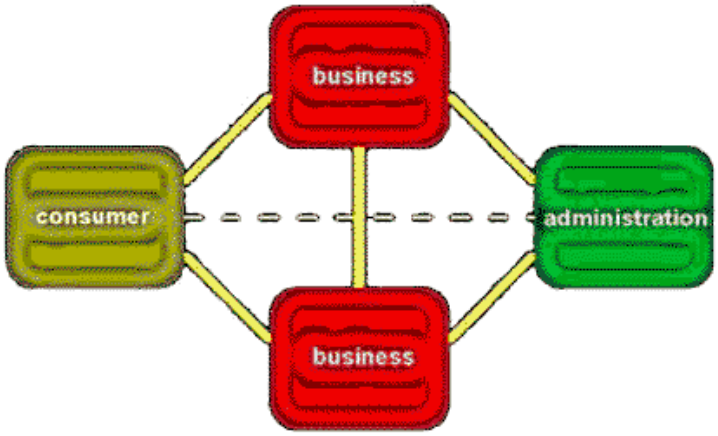
Ο όρος ηλεκτρονικό εμπόριο χρησιμοποιείται για να περιγράψει την χρήση

τηλεπικοινωνιακών μέσων για κάθε είδους εμπορικές συναλλαγές ή επιχειρηματικές δραστηρίοτητες μεταξύ επιχειρήσεων και ιδιωτών. Με άλλα λόγια, κάθε εμπορική δραστηριότητα που πριν από μερικά χρόνια ήταν δυνατή, μόνο χάρη στην φυσική παρουσία και μεσολάβηση ανθρώπων ή υλικών μέσων, σήμερα μπορεί να πραγματοποιηθεί αυτόματα, ηλεκτρονικά εξ' αποστάσεώς.

Τύποι ηλεκτρονικού εμπορίου:

Τα είδη του ηλεκτρονικού εμπορίου είναι τέσσερα και είναι τα εξής :

- **επιχείρηση - επιχείρηση**
- **επιχείρηση - καταναλωτής**
- **επιχείρηση - δημόσια διοίκηση**
- **καταναλωτής - δημόσια διοίκηση**



Εικόνα 1.1 - Το μοντέλο του ηλεκτρονικού εμπορίου

επιχείρηση – επιχείρηση

Είναι μια επιχείρηση που χρησιμοποιεί ένα δίκτυο για τις παραγγελίες της από προμηθευτές, που λαμβάνει τιμολόγια και κάνει πληρωμές. Αυτή η κατηγορία έχει κατοχυρωθεί αρκετά χρόνια, ειδικά με την χρησιμοποίηση του EDI σε κλειστά ή διεθνή δίκτυα.

επιχείρηση – καταναλωτής

Εξομοιώνεται με την ηλεκτρονική λιανική πώληση. Αυτή η κατηγορία έχει αναπτυχθεί με την εκτόξευση του World Wide Web. Οι καταναλωτές μαθαίνουν για τα προϊόντα μέσα από ηλεκτρονικές εκδόσεις, αγοράζουν προϊόντα με "ψηφιακό" χρήμα και άλλα ασφαλή συστήματα πληρωμής. Υπάρχουν τώρα "καταστήματα" σε όλο το Internet, που προσφέρουν κάθε είδος προϊόντων.

επιχείρηση - δημόσια διοίκηση

Καλύπτει όλες τις συναλλαγές μεταξύ επιχειρήσεων και δημόσιων οργανισμών. Προς το παρόν, αυτή η κατηγορία είναι σε νηπιακό στάδιο, αλλά μπορεί να αναπτυχθεί ραγδαία όσο οι κυβερνήσεις χρησιμοποιούν τις δικές τους λειτουργίες για να προωθήσουν την αντίληψη τους για το Ηλεκτρονικό Εμπόριο. Επιπροσθέτως, οι διοικήσεις πρέπει να παρέχουν την ευκαιρία ηλεκτρονικών συναλλαγών για καταστάσεις όπως επιστροφές ΦΠΑ και δασμών.

πελάτης - δημόσια διοίκηση

Δεν έχει ακόμα ενεργοποιηθεί. Στον βωμό της ανάπτυξης των 2 προηγούμενων κατηγοριών, οι επιχειρήσεις πρέπει να αναπτύξουν τις ηλεκτρονικές συναλλαγές σε περιοχές όπως πληρωμές κοινωνικής πρόνοιας και ιδιωτικών φόρων.

Πλεονεκτήματα και Μειονεκτήματα του Ηλεκτρονικού Εμπορίου

Πλεονεκτήματα του ηλεκτρονικού εμπορίου για τον καταναλωτή

- Τα ηλεκτρονικά καταστήματα είναι ανοιχτά 24 ώρες το 24ωρο. Με άλλα λόγια οποιαδήποτε στιγμή το επιθυμούμε, μπορούμε να αγοράσουμε ένα CD, ένα αεροπορικό εισιτήριο, να κάνουμε μια κράτηση σε ένα ξενοδοχείο κτλ.

- Το κόστος των προϊόντων που πωλούνται μέσω Internet είναι κατά γενικό κανόνα πολύ χαμηλότερο από τις τιμές του εμπορίου, αφού ένα ηλεκτρονικό κατάστημα είναι απαλλαγμένο από μεγάλο μέρος του λειτουργικού κόστους ενός πραγματικού καταστήματος (ενοικίαση χώρου και «αέρα», ηλεκτρικό, νερό κλπ) και γενικά απαιτεί πολύ λιγότερο υπαλληλικό προσωπικό.

- Η αγορά είναι πραγματικά παγκόσμια. Με άλλα λόγια, μπορούμε μέσω του υπολογιστή μας να αγοράσουμε ακόμα και κάτι το οποίο δεν κυκλοφορεί στην Ελλάδα, χωρίς να πρέπει πια να περιμένουμε πότε κάποιος φίλος μας θα ταξιδέψει στο εξωτερικό για να μας το φέρει.

- Η συναλλαγή είναι γρήγορη και άμεση. Με άλλα λόγια, από τη στιγμή που ολοκληρώνετε η παραγγελία μας, μέσα λίγες ημέρες την έχουμε λάβει, ακόμα και αν εκείνη τη στιγμή το προϊόν βρίσκεται στην άλλη άκρη του πλανήτη.

- Αλλά το πιο πρακτικό και πιο σημαντικό όφελος για τον καταναλωτή από το ηλεκτρονικό εμπόριο είναι το ότι ο καθένας βρίσκει αυτό που θέλει, όποτε το θέλει χωρίς κόπο και χωρίς καμία σπατάλη χρόνου. Με άλλα λόγια απλά και εύκολα ψώνια από το σπίτι ή το γραφείο.

Πλεονεκτήματα του ηλεκτρονικού εμπορίου για την εταιρία

- Όπως προαναφέραμε, κάθε εταιρία που έχει ηλεκτρονική παρουσία μπορεί να διευρύνει τον κύκλο εργασιών της επεκτείνοντας τα γεωγραφικά όρια των συναλλαγών της. Αυτό σημαίνει πως κάθε επιχείρηση που διαθέτει τα προϊόντα της online μπορεί και αποκτά

πελάτες σε περιοχές που βρίσκονται μακριά από την έδρα της, ακόμα και στο εξωτερικό. Με άλλα λόγια, κάθε επιχείρηση που έχει ένα ηλεκτρονικό κατάστημα, είναι σαν να έχει υποκαταστήματα σε πολλές περιοχές και μάλιστα με ελάχιστο λειτουργικό κόστος.

- Κάθε εταιρία που χρησιμοποιεί τις νέες τεχνολογίες, όπως το Internet, γίνεται εξ ορισμού πιο ανταγωνιστική, αφού μπορεί να ενημερώνεται πιο εύκολα για τις τρέχουσες εξελίξεις στο χώρο της. Με άλλα λόγια και με δεδομένο το ότι σε λίγα χρόνια όλες οι εμπορικές δραστηριότητες θα γίνονται μέσω Internet, το ηλεκτρονικό εμπόριο είναι η νέα μεγάλη πρόκληση για κάθε εταιρία που θέλει να είναι ανταγωνιστική.

- Οι ηλεκτρονικές συναλλαγές επιτρέπουν την αμφίδρομη σχέση μεταξύ επιχείρησης και καταναλωτή (interaction). Αυτό σημαίνει πως κάθε εταιρία μέσω των ηλεκτρονικών συναλλαγών μπορεί να συλλέξει πολλά στοιχεία για τις συνήθειες, τις ανάγκες και τα γούστα των καταναλωτών και σύμφωνα με αυτά να αναπροσαρμόσει την πολιτική της προς το θετικότερο.

- Τέλος, γνωρίζοντας τις συγκεκριμένες ανάγκες των πελατών τους, οι εταιρίες μπορούν να προχωρήσουν στη δημιουργία συγκεκριμένων προϊόντων είτε ανταποκρινόμενων σε έναν καταναλωτή, είτε σε μια ομάδα καταναλωτών που χρειάζονται ένα νέο προϊόν το οποίο δεν υπάρχει ακόμα στην αγορά.

Μειονεκτήματα του ηλεκτρονικού εμπορίου

- Προβλήματα ασφαλείας

Το διαδίκτυο είναι ένα μέσο που δεν παρέχει το επιθυμητό επίπεδο ασφάλειας στις συναλλαγές, με αποτέλεσμα και οι συναλλαγές να μην ασφαλείς. Βέβαια σε αυτόν τον τομέα γίνεται εκτεταμένη έρευνα έτσι ώστε οι συναλλαγές να γίνονται με όσο το δυνατόν μεγαλύτερη ασφάλεια. Βέβαια για να μην είμαστε υπερβολικοί, τα ηλεκτρονικά συστήματα πληρωμών που εφαρμόζονται, έχουν λύσει τα μεγαλύτερα προβλήματα ασφάλειας και μπορεί κανείς να πει ότι είναι εξίσου, αν όχι περισσότερο, ασφαλή και ευέλικτα από τις παραδοσιακές μεθόδους πληρωμών.

- Έλλειψη επαφής πωλητή - πελάτη

Το φαινόμενο αυτό δημιουργεί δυσπιστία στον καταναλωτή αφού δεν βλέπει το προϊόν και τον πωλητή. Δεν είναι σίγουρος αν αυτό που βλέπει στην οθόνη είναι όντος αυτό που θα παραλάβει, ή αν αυτά που ισχυρίζεται η εταιρία για το προϊόν είναι όντως αληθινά.

- Δυσκολία της χρήσης πολύπλοκων ηλεκτρονικών συστημάτων πληροφορικής

Η εκθετική αύξηση της ποσότητας πληροφοριών που είναι διαθέσιμες μέσα από τη ψηφιακή υποδομή, κάνει διαρκώς δυσκολότερο το διαχωρισμό και την ανεύρεση συγκεκριμένων πληροφοριών. Οι χρήστες επιθυμούν να μπορούν να βρουν πληροφορίες με την ελάχιστη δυνατή προσπάθεια, αλλά συχνά δεν διαθέτουν τα εργαλεία και τις γνώσεις που απαιτούνται για μια αποτελεσματική αναζήτηση.

1.2: Το ηλεκτρονικό κατάστημα

Η Επανάσταση της Πληροφορικής άλλαξε σημαντικά τον τρόπο ζωής των πολιτών, επιφέροντας μια σειρά αλλαγών, που επηρεάζουν και τις εμπορικές επιχειρήσεις. Αυτές οι αλλαγές μπορούν να αποτελέσουν σημαντικό όπλο στα χέρια των επιχειρήσεων που θέλουν να ανταποκριθούν στις απαιτήσεις ενός νέου περιβάλλοντος που χαρακτηρίζεται από συνεχώς μεταβαλλόμενες συνθήκες, διεθνοποίηση και εντατικοποίηση του ανταγωνισμού, κ.α. Οι επιχειρήσεις που θα "επιβιώσουν" στον ανταγωνισμό είναι αυτές που στον παρόντα χρόνο θα κάνουν τις στρατηγικές επιλογές για την ενσωμάτωση των νέων τεχνολογιών στις πρακτικές τους. Παλαιότερα, η ενσωμάτωση αυτή περιλάμβανε μόνο την ηλεκτρονική παρουσίαση των καταστημάτων, όχι όμως και όλων των ειδών τους και, πολύ περισσότερο, δεν υπήρχε δυνατότητα άμεσης παραγγελίας κάποιου είδους.

Το ηλεκτρονικό επιχειρείν (E-Business) αναφέρεται στην πραγματοποίηση επιχειρηματικών συναλλαγών μέσω του Internet και είναι η προσαρμογή του κλασικού επιχειρηματικού μοντέλου στην νέα ηλεκτρονική πραγματικότητα ή την ανάπτυξη νέου επιχειρηματικού μοντέλου με αντικείμενο μόνο το Διαδίκτυο. Η ανάπτυξη της Ελληνικής αγοράς μέσω διαδικτύου χρονολογείται από τα μέσα της προηγούμενης δεκαετίας και, αρχικά,

παρουσίαζε διαφορά φάσης περίπου πέντε ετών από τις αναπτυγμένες διεθνώς αγορές. Σύμφωνα με την Κλαδική Μελέτη «Υπηρεσίες Internet» για την Ελλάδα, (ICAP, Νοέμβριος 2004): *Το ηλεκτρονικό επιχειρείν και εμπόριο βρίσκεται ακόμα σε νηπιακό στάδιο και σε τούτο συμβάλλει και η έλλειψη παράδοσης στις συναλλαγές εξ αποστάσεως. Μόνο το 23% των επιχειρήσεων έχει παρουσία στο Internet το 2004*. Η κατάσταση αυτή άλλαξε από τότε και συνεχίζει να αλλάζει, όχι μόνον στην Ελλάδα, αλλά και σε διεθνές επίπεδο.

Όπως χαρακτηριστικά έχει πει ο Μπιλ Γκέιτς: "Το Διαδίκτυο δεν είναι απλώς άλλο ένα κανάλι πωλήσεων. Θα μετασχηματίσει την επιχείρησή σας. Η μελλοντική επιχείρηση θα λειτουργεί με ένα ψηφιακό νευρικό σύστημα". Με λίγα λόγια: Πωλήσεις σε οποιονδήποτε, οπουδήποτε, οποτεδήποτε.

Έτσι, αναπτύσσονται διεθνώς, αλλά και στην Ελλάδα, τα ηλεκτρονικά καταστήματα, που προσφέρουν ημερησίως χιλιάδες προϊόντα που υπόσχονται χαμηλότερες τιμές. Ανάλογα με τα προσφερόμενα είδη, ο μελλοντικός πελάτης μπορεί να αναζητήσει ανάμεσα σε πολλά ομοειδή το συγκεκριμένο είδος που επιθυμεί, να μάθει την τιμή και τον χρόνο αποστολής (εάν το παραγγείλει), να το δει σε εικόνες (ορισμένες φορές και σε βίντεο) και να κάνει και σχετικές συγκρίσεις τιμών. Οι τιμές στα ηλεκτρονικά καταστήματα είναι φθηνότερες, γιατί ένα τέτοιο κατάστημα δεν διατηρεί σημεία πώλησης με υψηλό ενοίκιο, δεν απασχολεί αριθμητικά το ίδιο προσωπικό με ένα συμβατικό και παραμένει "ανοικτό" σε 24ωρη βάση και για 365 μέρες ετησίως. Ο πελάτης μπορεί ακόμη να βρει και να παραγγείλει είδη που δεν υπάρχουν στα συμβατικά καταστήματα της πόλεως ή της χώρας του και μπορεί να πληρώσει μέσω της πιστωτικής του κάρτας ή με την χρήση της αντικαταβολής ή paypal.

1.2.1 Οι on-line κρατήσεις δωματίων

Ένα ξενοδοχείο, όπως οποιαδήποτε άλλη επιχείρηση, αποκομίζει όλα τα πλεονεκτήματα που προσφέρει το ηλεκτρονικό εμπόριο. Πλεονεκτήματα όμως δεν αποκομίζει μόνο το ξενοδοχείο αλλά και ο πελάτης που θα επιλέξει να κάνει μια κράτηση μέσω του on-line συστήματος. Αναλυτικότερα τα πλεονεκτήματα μιας on-line κράτησης τόσο από την πλευρά του ξενοδοχείου όσο και από την πλευρά του πελάτη είναι:

- Παγκόσμια προβολή
- Το ξενοδοχείο γίνεται ανταγωνιστικότερο.
- Άμεση και γρήγορη συναλλαγή
- 24ωρη λειτουργία και 24ωρη δυνατότητα πραγματοποίησης κρατήσεων
- Δυνατότητα σύγκρισης και επιλογής ξενοδοχείου (που βρίσκεται σε οποιοδήποτε μέρος του κόσμου) από το σπίτι.
- Χαμηλότερες τιμές.

Βασικές λειτουργίες ενός συστήματος on-line κρατήσεων δωματίων στην πλευρά του διαχειριστή:

Ο διαχειριστής – ιδιοκτήτης ενός ξενοδοχείου που χρησιμοποιεί ένα on-line σύστημα κρατήσεων δωματίων, θα πρέπει να έχει την δυνατότητα να διαχειρίζεται πλήρως την εφαρμογή, έχοντας έτσι την ευελιξία που προσφέρει η διαχείριση της επιχείρησης αν δεν είχε ηλεκτρονική υπόσταση. Οι κυριότερες λειτουργίες που ικανοποιούν την παραπάνω συνθήκη είναι:

- Δυνατότητα προσθήκης και αφαίρεσης δωματίων προς κράτηση στο on-line σύστημα.
- Δυνατότητα ενημέρωσης του διαχειριστή για τις κρατήσεις δωματίων.
- Δυνατότητα διαγραφής – ακύρωσης κράτησης από την λίστα των κρατήσεων.
- Δυνατότητα προσθήκης και αφαίρεσης πακέτων διακοπών.
- Δυνατότητα ενημέρωσης του διαχειριστή για τις κρατήσεις πακέτων διακοπών.
- Δυνατότητα διαγραφής – ακύρωσης κράτησης από την λίστα κρατήσεων των πακέτων διακοπών.
- Δυνατότητα προσθήκης - αφαίρεσης επιπλέον διαχειριστών στο σύστημα.

- Δυνατότητα αφαίρεσης διαχειριστών στο σύστημα.

- Δυνατότητα προσθήκης πληροφοριών στο on-line σύστημα.

Βασικές λειτουργίες ενός συστήματος on-line κρατήσεων δωματίων στην πλευρά του διαχειριστή:

Ο πελάτης που θα επιλέξει να κάνει μια κράτηση μέσω ενός συστήματος on-line κρατήσεων, θα πρέπει να έχει την δυνατότητα να υποβάλει ερωτήματα στο σύστημα και να παίρνει τις πληροφορίες που θα έπαιρνε, εάν αντί για το σύστημα απευθυνόταν στον υπάλληλο που βρίσκεται στην ρεσεψιόν του ξενοδοχείου. Οι κυριότερες λειτουργίες που ικανοποιούν την παραπάνω συνθήκη είναι:

- Δυνατότητα αναζήτησης ελεύθερων δωματίων προς κράτηση με βάση την επιθυμητή ημερομηνία κράτησης.

- Δυνατότητα προσθήκης στο καλάθι των επιθυμητών δωματίων προς κράτηση.

- Δυνατότητα ολοκλήρωσης κράτησης και επιλογή του τρόπου πληρωμής.

- Δυνατότητα εμφάνισης των διαθέσιμων πακέτων διακοπών.

- Δυνατότητα κράτησης πακέτου διακοπών.

- Δυνατότητα εμφάνισης πληροφοριών για το ξενοδοχείο.

Η υλοποίηση των παραπάνω λειτουργιών, τόσο από την πλευρά του διαχειριστή όσο και από την πλευρά του πελάτη, καθώς και μερικών επιπλέον θα παρουσιαστεί στο κεφάλαιο 5.

ΚΕΦΑΛΛΑΙΟ 2ο: Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ RUBY

Ήθελα να μειώσω όσο το δυνατόν περισσότερο την απογοήτευσή μου όταν προγραμμάτιζα, ήθελα δηλαδή να μειώσω την δυσκολία μιας γλώσσας. Αυτός ήταν ο αρχικός μου στόχος όταν σχεδίαζα την Ruby. Ήθελα ο χρόνος που αφιέρωνα όταν προγραμμάτιζα να είναι στιγμές διασκέδασης.

- Yukihiro "Matz" Matsumoto, δημιουργός της Ruby

Η Ruby είναι μια δυναμική, γενικής χρήσης, αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία συνδυάζει σύνταξη εμπνευσμένη από την perl και χαρακτηριστικά της smalltalk. Η ruby δημιουργήθηκε στην Ιαπωνία στα μέσα της δεκαετίας του 90'. Αρχικά αναπτύχθηκε και σχεδιάστηκε από τον Yukihiro "Matz" Matsumoto για προσωπική του χρήση. Ήταν αρχικά επηρεασμένη από της γλώσσες Perl, Smalltalk, Eiffel και Lisp.

2.1 Η ιστορία της Ruby

Η Ruby θεωρείται νέα γλώσσα στον κόσμο τον γλωσσών προγραμματισμού. Άρχισε να πρωτοαναπτύσσεται το 1993, έχοντας έτσι περίπου την ίδια ηλικία με την Perl και την Python.

Η ruby ξεκίνησε στην Ιαπωνία ως δημιούργημα του Yukihiro Matsumoto, γνωστού και ως "Matz". Σε αντίθεση με τους περισσότερους προγραμματιστές, το κίνητρο του Matz, για την δημιουργία της γλώσσας, ήταν η αρχή της "αποφυγής δυσάρεστων εκπλήξεων", επίσης ήθελε να κάνει την συγγραφή κώδικα πιο ευχάριστη και όσο το δυνατόν πιο "ανώδυνη", ώστε να βελτιωθεί η παραγωγικότητα των προγραμματιστών. Αφού δοκίμασε πολλές γλώσσες, δεν κατάφερε να βρει μια που να του "ταιριάζει" απόλυτα, έτσι λοιπόν άρχισε να αναπτύσσει την δική του άποψη για το

πως πρέπει να λειτουργεί μια γλώσσα προγραμματισμού και έτσι δημιουργήθηκε η Ruby.

Το όνομα “Ruby” αποφασίστηκε κατά την διάρκεια μιας on-line συνομιλίας με τον Keitzu Ishitsuka στης 24 Φεβρουαρίου του 1993, πριν ακόμη ξεκινήσει να γράφεται ο οποιοσδήποτε κώδικας για την ανάπτυξη της Ruby. Αρχικά προτάθηκαν δύο ονόματα, το Coral και το Ruby. Το όνομα Coral είχε ήδη μεγάλη ιστορία ως όνομα μιας γλώσσας προγραμματισμού που δημιουργήθηκε το 1964 στο Royal Radar Establishment στο Ηνωμένο Βασίλειο. Έτσι τελικά ο “Matz” επέλεξε το “Ruby”. Αργότερα ο “Matz” δήλωσε πως ο κύριος παράγοντας για την επιλογή του ονόματος Ruby ήταν λόγω του ότι ένα ρουμπίνι ήταν το birthstone (το birthstone είναι ένας πολύτιμος λίθος που δίνεται σαν δώρο και συμβολίζει στο Γρηγοριανό ημερολόγιο τον μήνα που γεννήθηκε κάποιος) από κάποιον συνάδελφό του.

Μετά από μακρόχρονη εμπειρία και έρευνα στον αντικειμενοστραφή προγραμματισμό, ο "Matz" πίστευε πως ήταν το ιδανικό μοντέλο που έπρεπε να υιοθετήσει για την νέα γλώσσα που επρόκειτο να δημιουργήσει, αλλά σε αντίθεση με άλλες γλώσσες, όπως η Perl, η αντικειμενοστρέφεια δεν θα ήταν η δευτερογενής σκέψη, αλλά το θεμέλιο για ολόκληρη την γλώσσα. Τα πάντα έπρεπε να είναι αντικείμενα και οι μέθοδοι (methods) έπρεπε να αντικαταστήσουν τον ρόλο τον διαδικασιών (procedures), που χρησιμοποιούσαν οι προγραμματιστές που προγραμμάτιζαν σε διαδικαστικές γλώσσες (procedural languages). Όπως είπε και ο "Matz" σε μια συνέντευξη το 2001 "Ήθελα μια γλώσσα που να ήταν πιο δυνατή από την Perl και πιο αντικειμενοστραφής από την Python, γιαυτό αποφάσισα να φτιάξω την δικιά μου γλώσσα".

Τον Δεκέμβριο του 1995, ο “Matz” δημοσίευσε την πρώτη δημόσια άλφα έκδοση (public alpha version) της Ruby, και σύντομα μια κοινότητα άρχισε να σχηματίζεται στην Ιαπωνία. Ωστόσο, αν και στην Ιαπωνία η Ruby έγινε σχετικά γρήγορα δημοφιλής, χρειάστηκε να “παλέψει” αρκετά για να αποκτήσει μια βάση στον υπόλοιπο κόσμο.

Το 1996, ο ανάπτυξη εφαρμογών σε Ruby άρχισε σιγά σιγά να αποκτάει περισσότερους “οπαδούς” και μια μικρή ομάδα από προγραμματιστές του πυρήνα (core developers) της Ruby και μερικούς λάτρεις της γλώσσας άρχισαν να συγκροτούν μια περισσότερο γενική κοινότητα προγραμματιστών της Ruby. Η έκδοση 1.0 της ruby δημοσιεύτηκε στις 25 Δεκεμβρίου 1996. Οι προγραμματιστές πυρήνα (core develpers) της Ruby βοηθούσαν τον “Matz” στην ανάπτυξη της

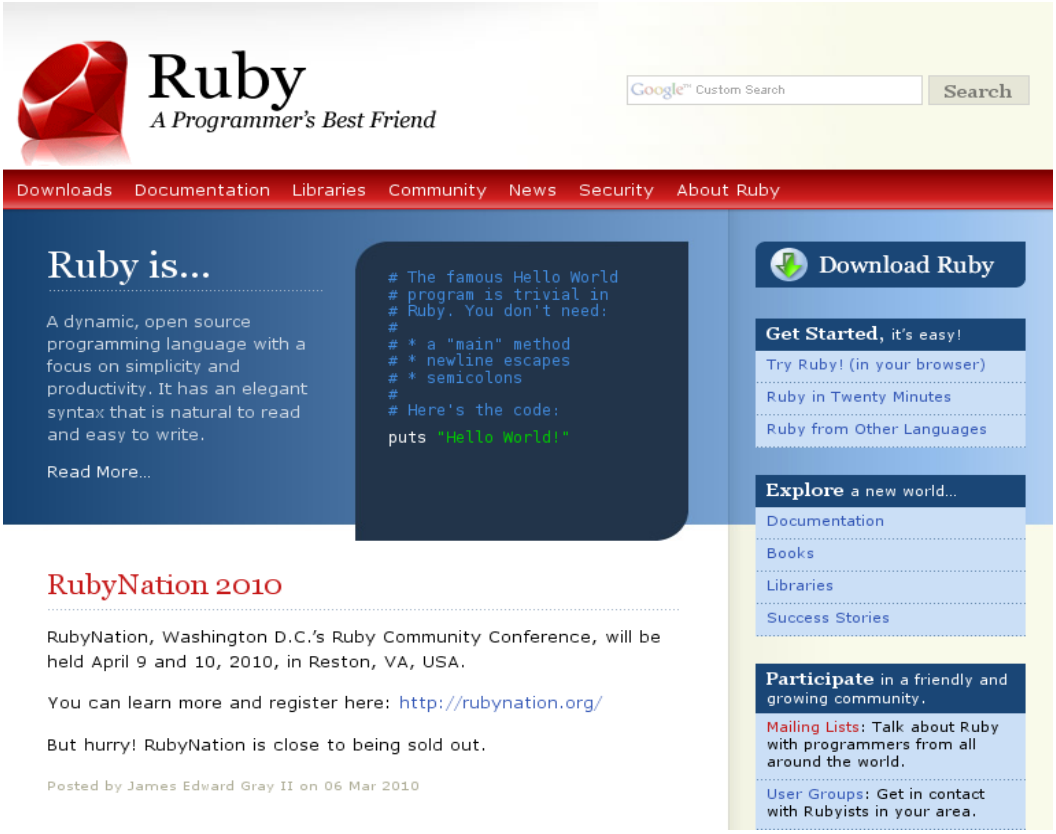
γλώσσας, δημοσίευαν patches με διορθώσεις και του παρείχαν ιδέες. Ο “Matz” συνέχισε να ενεργεί ως “καλοκάγαθος δικτάτορας” που τελικά έλεγχε τις “κατευθύνσεις της γλώσσας, παρ' όλη την συνεχώς διευρυνόμενη προσέλευση προγραμματιστών.

Η Ruby αρχικά δημιουργήθηκε από τον “Matz” για προσωπική του χρήση στην Ιαπωνία, αυτό είχε σαν αποτέλεσμα, όλο το το εγχειρίδιο χρήσης (documentation) της γλώσσας να είναι γραμμένο στα Ιαπωνικά, μη δίνοντας έτσι την δυνατότητα στους μη-Ιάπωνες προγραμματιστές να γράψουν κώδικα σε Ruby. Αν και συνήθως οι λέξεις κλειδιά στης περισσότερες γλώσσες είναι στα αγγλικά (όπως get, set, print, if, for κτλ), αυτό δεν συνέβαινε στην Ruby μέχρι και το 1997 όπου και άρχισε να μεταφράζετε το εγχειρίδιο χρήσης στα αγγλικά.

Ο ”Matz” ξεκίνησε επίσημα να προωθεί την γλώσσα του στα αγγλικά στα τέλη του 1998 δημιουργώντας την λίστα ταχυδρομείου (mailing list) ruby-talk, όπου και παραμένει μέχρι και σήμερα ένα από τα καλύτερα “μέρη” για να συζητήσεις σχετικά με την Ruby, καθώς επίσης και ένα πολύ καλό εργαλείο μιας και πάρα πολλοί Ruby προγραμματιστές θα σε βοηθήσουν να λύσεις κάποιο πρόβλημά σου. Ένα χρόνο μετά στα τέλη του 1999 δημιουργήθηκε και η πρώτη επίσημη αγγλική ιστοσελίδα της Ruby (http://www.ruby-lang.org/) που παραμένει μέχρι και σήμερα.

Η Ruby απέτυχε να προσελκύσει πολύ κόσμο παρά μόνο ελάχιστους “σκληροπυρηνικούς” προγραμματιστές, έτσι μέχρι το 2001 η κοινότητα που μιλούσε αγγλικά ήταν υπερβολικά μικρή (μιας και το κύριο newsgroup της ruby “comp.lang.ruby” δημιουργήθηκε στα τέλη του 2000). Μετά το 2001 όμως η κοινότητα άρχισε να μεγαλώνει προς μεγάλη έκπληξη του “Matz” που έβλεπε πως υπήρχαν άνθρωποι που έβρισκαν την γλώσσα πολύ χρήσιμη.

Ωστόσο η προώθηση της Ruby στο ευρύτερο κοινό των προγραμματιστών συνέχιζε να παραμένει χαμηλή, μέχρι που η IBM δημοσίευσε ένα άρθρο με μια σύντομη επισκόπηση της γλώσσας και με μια συνέντευξη του “Matz” στα τέλη του 2000. Έπειτα ο Dr.Bobb's Journal δημοσίευσε ένα άρθρο των Dave Thomas και Andy Hunt με μια παρόμοια εισαγωγή στην γλώσσα με αυτήν της IBM.



Εικόνα 2.1 - Η σημερινή μορφή της σελίδας www.ruby-lang.org

Παρά την προφανή δύναμη της Ruby, τα φαβορί για να κερδίσουν την "μάχη" για το ποια θα είναι η επόμενη perl και η γενικότερη scripting και WEB γλώσσα, ήταν η python και η PHP. Αυτό μέχρι το 2004, που όλα άλλαξαν όταν ο νέος σε ηλικία τότε Dane εξέδωσε το Web Framework Ruby On Rails (RoR), όπου γρήγορα άλλαξε την αντίληψη που είχε η παγκόσμια κοινότητα προγραμματιστών. Στο Web Framework Ruby on Rails θα αφιερωθεί το κεφάλαιο 3.

2.2 Οι επιρροές της Ruby

Κατά την ανάπτυξη της γλώσσας ο "Matz" ήταν επηρεασμένος από γλώσσες προγραμματισμού με τις οποίες είχε ήδη εξοικειωθεί αρκετά. Ο "Matz" θεωρούσε τον δημιουργό της perl Larry Wall ήρωά του, και το βασικό αξίωμα της perl "Υπάρχουν περισσότεροι απο έναν τρόποι για να κάνεις κάτι (There's more than one way to do it) εμφανίζεται συνεχώς στην Ruby.

Πολλές γλώσσες, όπως η Python, παρέχουν ως επί το πλείστον άκαμπτες δομές και "τυποποιημένες" μεθόδους, αφήνοντας έτσι στον προγραμματιστή λιγότερες επιλογές για την ανάπτυξη μιας εφαρμογής. Η Ruby αντιθέτως δίνει την ελευθερία στον προγραμματιστή να λύσει με πολύ περισσότερους από έναν τρόπους κάποιο πρόβλημα. Αυτή η ιδιότητα παρέχει μεγάλη ευελιξία στην γλώσσα και σε συνδυασμό με τη απόλυτα αντικειμενοστραφή φύση της γλώσσας, κάνει την Ruby μια πλήρως παραμετροποιήσιμη γλώσσα.

Όσον αφορά την αντικειμενοστραφή της φύση, η Ruby είναι πάρα πολύ επηρεασμένη από την Smalltalk, μια αντικειμενοστραφή γλώσσα που αναπτύχθηκε την δεκαετία του 1970. Όπως και στην Smalltalk τα πάντα στην Ruby είναι αντικείμενα, και η Ruby δίνει την δυνατότητα στους προγραμματιστές της να αλλάζουν πολλές λεπτομέρειες από τις λειτουργίες της γλώσσας πολύ εύκολα.

Εν κατακλείδι, η Python, η LISP, η ADA και η C++, έχουν επηρεάσει την Ruby. Οι επιρροές αυτές δείχνουν πως η Ruby δεν "φοβάται" να ενστερνιστεί της καλύτερες ιδέες από άλλες γλώσσες προγραμματισμού και σε αυτό οφείλει την δύναμη και την ευελιξία της. Τα χαρακτηριστικά αυτά κάνουν την μετάβαση από μια άλλη γλώσσα στην Ruby αρκετά εύκολη. Οπότε μαθαίνοντας Ruby, σε μεγάλο βαθμό, μαθαίνεις και τα καλύτερα χαρακτηριστικά άλλων γλωσσών.

2.3 Η Open Source κουλτούρα

Όταν η Ruby πρωτοξεκίνησε να αναπτύσσεται, "Matz" δεν είχε στο μυαλό του κάποια συγκεκριμένη κουλτούρα που θα ακολουθούσε η Ruby, απλά είχε φτιάξει μια γλώσσα για προσωπική του χρήση, που ταίριαζε με τον δικό του τρόπο σκέψης και τα πρώτα χρόνια κράτησε την γλώσσα για τον ίδιο. Το μεγαλύτερο μέρος της κουλτούρας που υιοθετήθηκε για το πως να προγραμματίσεις σε Ruby εξελίχθηκε τα τελευταία χρόνια.

Open source (ανοιχτός κώδικας) σημαίνει ότι ο πηγαίος κώδικας μιας εφαρμογής είναι ελεύθερα διαθέσιμος σε τρίτους για να τον μελετήσουν, να τον χρησιμοποιήσουν, ακόμα και για να τον βελτιώσουν. Όπως λοιπόν το Linux έτσι και η Ruby μαζί με όλες τις βιβλιοθήκες της, έχει εκδοθεί κάτω από μια open source άδεια που στην ουσία είναι ένας συνδυασμός της GPL με

κάποιους επιπλέον όρους που πρόσθεσε ο “Matz”.

Οι όροι της Ruby δεν προϋποθέτουν πως οποιαδήποτε εφαρμογή δημιουργηθεί με την Ruby θα πρέπει απαραίτητα να είναι και open source. Η άδεια κάτω από την οποία διανέμεται η Ruby σου δίνει και την δυνατότητα να αναπτύξεις μια εφαρμογή που θα είναι κλειστού κώδικα, τον πηγαίο κώδικα της οποίας δεν θα μπορεί κανείς να τον δει ή να τον χρησιμοποιήσει.

Όταν η 37Signals, που συμμετείχαν στο project Basecamp, ανέπτυξαν το framework Ruby On Rails, το εξέδωσαν κάτω από open source άδεια. Αυτό είχα σαν αποτέλεσμα η εταιρία να αποκτήσει μεγάλη δημοσιότητα και να προσλάβει πολύ καλούς προγραμματιστές, οι οποίοι δούλευαν πάνω στο framework δωρεάν, προς κοινό όφελος όλων. Πολλά γνωστά προγράμματα όπως ο Apache server, η MySQL, ο Firefox κτλ, έχουν εκδοθεί κάτω από διάφορες open source άδειες.

Η open source κοινότητα, έχει ως σκοπό να προσφέρει την γνώση ελεύθερα και έχοντας τον κώδικα μιας εφαρμογής οποιοσδήποτε, η εφαρμογή αυτή έχει περισσότερες πιθανότητες βελτίωσης από μια εφαρμογή που τον πηγαίο κώδικά της τον έχουν λίγοι. Αν και πάντα οι κλειστού κώδικα εφαρμογές θα υπάρχουν, η ανοιχτού κώδικα κουλτούρα έχει αρχίσει να γίνεται μονόδρομος για τις γλώσσες προγραμματισμού και για τις βιβλιοθήκες τους.

Η κατανόηση της open source φιλοσοφία είναι το κλειδί για την κατανόηση της κοινότητας της Ruby. Αν και αρκετοί προγραμματιστές δεν κάνουν τις εφαρμογές τους ανοιχτού κώδικα, πολύ συχνά εκδίδουν δωρεάν εργαλεία και προγραμματιστικές τεχνικές για την ανάπτυξη κώδικα βοηθώντας έτσι την κοινότητα.

2.4 Η κοινότητα της Ruby

Η μελέτη ενός βιβλίου συνήθως καλύπτει ένα θέμα από μια πιο γενική πλευρά. Έτσι αν συναντήσεις κάποιο συγκεκριμένο πρόβλημα κατά την ανάπτυξη μιας εφαρμογής, το καλύτερο μέρος που μπορείς να απευθυνθείς για να βρεις την λύση είναι η κοινότητα της γλώσσας, όπου εκεί υπάρχουν περισσότεροι προγραμματιστές, οι οποίοι θα σε βοηθήσουν να βρεις την λύση για το πρόβλημα που αντιμετωπίζεις. Οι κοινότητα της Ruby αποτελείται από mailing lists, forums,

Newsgroups κτλ.

- Οι maling lists της Ruby

  Η Ruby έχει τρεις επίσημες mailling lists οι οποίες είναι:

  1. Η ruby-talk στην οποία υπάρχουν ερωτήσεις και θέματα που αφορούν την ίδια την γλώσσα.

  2. Η ruby-core στην οποία συζητούνται θέματα που αφορούν την ανάπτυξη της γλώσσας

  3. και η ruby-doc όπου τα θέματα που βρίσκονται εκεί έχουν να κάνουν με το εγχειρίδιο χρήσης της γλώσσας.

- Usenet Newsgroups

  Το κύριο Newsgroup της ruby είναι το comp.lang.ruby, όπου αν δεν έχουμε εγκατεστημένο λογισμικό για newsgroups, μπορεί να διαχειριστεί από την ιστοσελίδα http://groups.google.com/groups/comp.lang.ruby. Από το 2006 και μετά στο group γίνονται πολλές αναρτήσεις κάθε μέρα και αποτελείται από μεγάλο αριθμό Ruby προγραμματιστών.

- Το κανάλι στο IRC (Internet Relay Chat)

  Το IRC τελικά αποδείχτηκε το πιο σύνηθες μέρος συνάντησης των φίλων της Ruby. Στο κανάλι θα βρούμε 24ωρη υποστήριξη σε θέματα που αφορούν την Ruby. Το κανάλι της Ruby είναι: #ruby-lang στον server irc.freenode.net και του framework Ruby On Rails το #rubyonrails στον ίδιο server.

- Τα Forums

  Τα forums δεν θα μπορούσαν να λείπουν από την κοινότητα της Ruby. Τα πιο γνωστά είναι:

1. Το www.ruby-forum.com το οποίο δεν είναι ακριβώς forum, αλλά παρέχει σε μορφή forum, τις συζητήσεις που γίνονται στις τρεις mailing lists της ruby, που αναφέρθηκαν πιο πάνω.

2. Το www.rubyforums.com το οποίο αποτελείται από 15 subforums και ασχολείται με θέματα που αφορούν την Ruby και το framework Ruby On Rails.

3. Και το railsforum.com στο οποίο συζητούνται θέματα που αφορούν κατά κύριο λόγο το framework Ruby On Rails

2.5 Interactive Ruby Shell (Irb)

Το Irb είναι ένα πολύ ισχυρό εργαλείο που συνοδεύει την Ruby. Το Irb είναι ένα shell που αποτελείται από μια γραμμή εντολών. Οποιαδήποτε Ruby εντολή δοθεί εκεί εκτελείται αμέσως και βλέπουμε σε πραγματικό χρόνο τα αποτελέσματα. Αυτό βοηθάει πολύ στην εκμάθηση της γλώσσας, και ακόμη περισσότερο στην δοκιμή τμημάτων κώδικα, μιας και βλέπουμε το αποτέλεσμα κάθε εντολής.

```
irb(main):001:0> puts "hello world"
hello world
=> nil
irb(main):002:0> x = 5
=> 5
irb(main):003:0> y = 3
=> 3
irb(main):004:0> z = x + y
=> 8
irb(main):005:0> z.times do puts "hello world" end
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
=> 8
irb(main):006:0>
```

Εικόνα 2.2 - Παράδειγμα χρήσης του Irb

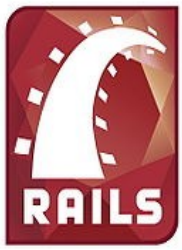
2.6 Η σύνταξη της ruby

Η σύνταξη της Ruby είναι σχεδόν ίδια με αυτήν της Perl και της Python. Ο ορισμός των κλάσεων και των μεταβλητών γίνεται με λέξεις κλειδιά. Σε αντίθεση όμως με την Perl, οι μεταβλητές δεν είναι υποχρεωτικό να ξεκινούν με κάποιον ειδικό χαρακτήρα (sigil) που να ορίζει τον τύπο δεδομένων της μεταβλητής. Τα sigils στην Ruby χρησιμοποιούνται για να δείξουν την εμβέλεια της μεταβλητής. Η πιο εντυπωσιακή διαφορά όμως σε σχέση με την C και την Perl είναι ότι χρησιμοποιούνται λέξεις κλειδιά για να ορίσουν την αρχή και το τέλος ενός μπλοκ κώδικα, χωρίς να χρειάζεται να χρησιμοποιηθούν αγκύλες. Αυτό κάνει τον κώδικα πιο ευανάγνωστο. Για πρακτικούς λόγους δεν υπάρχει διαφορά ανάμεσα σε εκφράσεις (expressions) και δηλώσεις (statements).

ΚΕΦΑΛΑΙΟ 3ο: Το Web Framework Ruby On Rails

Το Ruby On Rails είναι ενα ανοιχτού κώδικα Web Framework για την γλώσσα προγραμματισμού Ruby, που κάνει ευκολότερο τον προγραμματισμό και την διαχείριση των Web εφαρμογών, συχνά συναντάται με την συντομογραφία RoR ή απλά Rails. Τους μήνες που ακολούθησαν από την πρώτη έκδοση του, το Ruby On Rails από ένα άγνωστο "παιχνίδι" κατέληξε να γίνει ένα παγκοσμίως γνωστό και ευρέος χρησιμοποιούμενο Framework και κέρδισε πολλά βραβεία. Το σημαντικότερο όμως είναι ότι έγινε ένα Framework για να ανάπτυξη web2 εφαρμογών, και πολλές εταιρίες το χρησιμοποιούν για την ανάπτυξη Web εφαρμογών.

3.1 Η ιστορία του Ruby on Rails



Το Ruby On Rails αρχικά ξεκίνησε ως μια εφαρμογή που ονομαζόταν Basecamp. Ήταν ένα framework που αναπτύχθηκε από τον Δανό Web προγραμματιστή David Heinemeier Hansson, για την εταιρία σχεδιασμού 37signals. Λόγω της μεγάλης επιτυχίας του Basecamp η εταιρία 37signals στράφηκε στην ανάπτυξη και παραγωγή εφαρμογών και ο David Heinemeier Hansson έγινε συνεταίρος στην εταιρία.

Το Rails αρχικά δεν δημιουργήθηκε σαν ένα αυτόνομο framework. Ήταν η εξέλιξη μιας ήδη υπάρχουσας εφαρμογής, και θα χρησιμοποιούνταν για να την παραγωγή άλλων εφαρμογών της 37signal. Ο Hansson ξεκίνησε με την προοπτική να κάνει την δουλειά του ευκολότερη προσθέτοντας περισσότερη λειτουργικότητα, όπως ο χειρισμός βάσεων δεδομένων και η παραγωγή templates. Έτσι γεννήθηκε η πρώτη έκδοση του Ruby on Rails.

Ο Hansson αποφάσισε να εκδώσει την εφαρμογή κάτω από μια open source άδεια. Η πρώτη beta έκδοση του Rails κυκλοφόρησε τον Ιούλιο του 2004, με τις εκδόσεις 1.0 και 2.0 να ακολουθούν στις 13 Δεκεμβρίου του 2005 και στις 7 Δεκεμβρίου 2007 αντίστοιχα. Πολλές χιλιάδες προγραμματιστών από όλο τον κόσμο κατέβασε την εφαρμογή από το Internet, και το

πλήθος τους συνεχώς αυξανόταν.

Το γεγονός ότι το framework Ruby On Rails προήλθε από το project Basecamp, θεωρείται από την κοινότητα των Rails προγραμματιστών, ότι είναι στοιχείο που έδωσε την μεγάλη δύναμη στο framework που ήταν οτι από την πρώτη έκδοσή του, έδινε λύσεις σε πραγματικά προβλήματα. Το Rails δεν δημιουργήθηκε από το μηδέν, αλλά ήταν η εξέλιξη μιας ήδη υπάρχουσας εφαρμογής και κατάφερε να αποδείξει ότι είναι ένα χρήσιμο, συνεκτικό και περιεκτικό framework.

Η απόφαση του Hansson να κυκλοφορήσει το framework ως open source, έδωσε την δυνατότητα στο Ruby On Rails να πάρει όλα τα θετικά στοιχεία που μπορεί να προσφέρει η open source φιλοσοφία. Συνεχώς προγραμματιστές που δουλεύουν με το rails εκδίδουν εργαλεία και διορθώνουν λάθη που βρίσκουν στον πηγαίο κώδικα στο repository (αποθήκη) του rails. Το repository ελέγχεται από τον πυρήνα του Rails που αποτελείται από μια ομάδα έξι προγραμματιστών της οποίας ηγείται ο Hansson.

3.2 Η αρχιτεκτονική MVC (models – views – controllers)

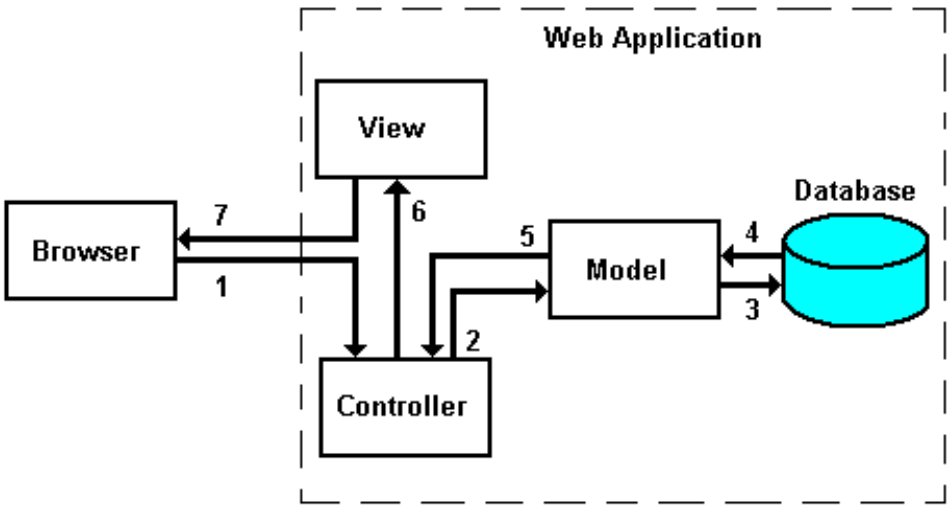
Το 1979 Trygve Reenskaug εμπνεύστηκε μια νέα αρχιτεκτονική για την ανάπτυξη διαδραστικών εφαρμογών. Σε αυτή την αρχιτεκτονική η εφαρμογή "σπάει" σε τρία μέρη: στα models, στα views και στους controllers. Το framework Ruby On Rails χρησιμοποιεί αυτή την αρχιτεκτονική.

Το μοντέλο (model) αντιστοιχίζεται στα δεδομένα της εφαρμογής. Τα δεδομένα αυτά μπορεί να είναι παροδικά, όταν υπάρχει μεγάλη αλληλεπίδραση με τον χρήστη, ή μπορεί να είναι σταθερά, και τότε συνήθως πρέπει να αποθηκευτούν κάπου εκτός της εφαρμογής, για παράδειγμα σε μια βάση δεδομένων.

Το μοντέλο δεν είναι μόνο τα δεδομένα της εφαρμογής. Είναι αυτό που επιβάλλει όλες τις ενέργειες και τους περιορισμούς που αφορούν τα δεδομένα της εφαρμογής. Αναπτύσσοντας λοιπόν αυτές τις ενέργειες και τους περιορισμούς μέσα στο ίδιο το μοντέλο, είμαστε σίγουροι πως στην εφαρμογή μας τίποτα, εκτός από το μοντέλο, δεν μπορεί να "πειράξει" τα δεδομένα μας.

Έτσι λοιπόν ένα μοντέλο είναι ταυτόχρονα διαχειριστής και φύλακας των δεδομένων της εφαρμογής μας.

Η όψη (view) είναι υπεύθυνη για την παραγωγή της διασύνδεσης χρήστη (user interface), που συνήθως βασίζεται στα δεδομένα του μοντέλου. Για παράδειγμα σε μια εφαρμογή που χειρίζεται ένα on-line κατάστημα, η λίστα με τα προϊόντα είναι προσπελάσιμη από το μοντέλο, αλλά η όψη είναι αυτή που μέσω του μοντέλου τα παρουσιάζει στον χρήστη την λίστα. Αν και η όψη μπορεί να παρουσιάσει στον χρήστη αποτελέσματα με διάφορους τρόπους, δεν μπορεί ποτέ να χειριστεί δεδομένα που έρχονται από την χρήστη στην εφαρμογή. Η δουλειά του view τελειώνει όταν τελικά παρουσιαστούν τα δεδομένα στον χρήστη. Στο ίδιο μοντέλο μπορούν να υπάρχουν πολλές όψης, κάθε μια για διαφορετικό σκοπό.



Εικόνα 3.1 – Το μοντέλο MVC

Οι ελεγκτές (controllers) είναι αυτοί που οργανώνουν την εφαρμογή. Οι ελεγκτές δέχονται δεδομένα από τον έξω κόσμο (χρήστη) αλληλεπιδρούν με το μοντέλο και παρουσιάζουν τα κατάλληλα αποτελέσματα μέσω της όψης στον χρήστη.

Το Ruby On Rails είναι ένα framework που χρησιμοποιεί την τεχνολογία MVC. Δημιουργεί στον προγραμματιστή μια τέτοια δομή, και όταν ο προγραμματιστής αναπτύξει τα μοντέλα, τις όψεις και τους ελεγκτές, το Rails τα συνδυάζει και παράγεται η τελική εφαρμογή. Ένα από τα πλεονεκτήματα του Rails είναι ότι έχει την "ευφυΐα" να τα συνδυάσει μεταξύ τους

και έτσι ο προγραμματιστής δεν χρειάζεται να γράψει επιπλέον κώδικα ώστε να τα κάνει να δουλέψουν όλα μαζί. Και αυτό είναι ένα παράδειγμα της ευκολίας που προσφέρει το Framework ruby on Rails.

Σε μια Rails εφαρμογή, ένα εισερχόμενο αίτημα στέλνεται αρχικά σε κάποιον δρομολογητή, ο οποίος αποφασίζει που πρέπει να σταλεί και πως πρέπει να επεξεργαστεί το αίτημα. Σε αυτό το σημείο καλείται η κατάλληλη μέθοδος στον ελεγκτή, ο ελεγκτής θα επεξεργαστεί το αίτημα, αν χρειαστεί θα αλληλεπιδράσει με το μοντέλο για να εξάγει κάποια δεδομένα από την βάση δεδομένων και τελικά θα εμφανίσει τα αποτελέσματα στον χρήστη μέσω της όψης.

3.3 Η αρχή DRY (Don't Repeat Yourself – μην επαναλαμβάνεστε)

Το Ruby On Rails υποστηρίζει την αρχή του DRY (don't repeat yourself) προγραμματισμού. Η λογική αυτής της αρχής, είναι ότι όταν κάποτε χρειαστεί να αλλάξουμε την συμπεριφορά σε κάποιο σημείο της εφαρμογής, δεν χρειάζεται να αλλάξουμε τον κώδικα σε περισσότερα από ένα σημεία.

Αυτό γίνεται εφικτό με τον εξής τρόπο. Αντί να γράφουμε συνεχώς τον ίδιο κώδικα σε διάφορα σημεία της εφαρμογής, γράφουμε μια φορά τον κώδικα σε κάποιο “κεντρικό” σημείο και όταν θέλουμε να χρησιμοποιήσουμε αυτό το τμήμα του κώδικα κάπου στην εφαρμογή, απλά κάνουμε μια αναφορά στο μέρος όπου βρίσκεται γραμμένος ο κώδικας. Έτσι αν θέλουμε να αλλάξουμε την συμπεριφορά της εφαρμογής δεν χρειάζεται να αλλάξουμε τα σημεία όπου καλείται ο κώδικας, αλλά μόνο το σημείο που είναι γραμμένος ο κώδικας.

Ένα παράδειγμα για το πως το Rails υποστηρίζει αυτή την αρχή είναι το εξής. Σε αντίθεση με την JAVA δεν χρειάζεται να ορίζουμε συνεχώς την βάση δεδομένων και να κάνουμε συνεχώς σύνδεση με αυτή μέσα στην ίδια εφαρμογή. Πολύ απλά ορίζουμε μια φορά την βάση δεδομένων και όταν χρειάζεται να αντληθούν πληροφορίες από την βάση το Rails, χρησιμοποιεί τον ήδη υπάρχοντα ορισμό. Ένα ακόμη παράδειγμα αυτής της αρχής είναι η ανάπτυξη τεχνικών όπως το AJAX (asynchronous javascript and XML), όπου όταν κάποιος browser δεν υποστηρίζει αυτή την τεχνική, ο προγραμματιστής πρέπει να εμφανίσει τα αποτελέσματα χωρίς να

χρησιμοποιήσει την τεχνική AJAX. Σε τέτοιες περιπτώσεις πολλοί προγραμματιστές πιάνουν τον εαυτό τους να γράφει τον ίδιο κώδικα πολλές φορές. Αυτό μπορεί πολύ εύκολα να το παρακάμψει το Framwrork Ruby On Rails χρησιμοποιώντας την αρχή DRY.

3.4 Convention over Configuration (Συμβάσεις αντί για ρυθμίσεις)

Η έννοια του *Convention over Configuration* αναφέρεται στο γεγονός ότι το Ruby On Rails έχει ένα μεγάλο αριθμό προεπιλεγμένων ρυθμίσεων για την κατασκευή μιας τυπικής web εφαρμογής.

Πολλά frameworks (όπως το Java-based Struts) χρειάζονται ένα μεγάλο αριθμό ρυθμίσεων, πριν ξεκινήσει η ανάπτυξη της εφαρμογής, ακόμη και αν αυτή είναι πολύ απλή. Οι ρυθμίσεις συνήθως αποθηκεύονται σε ένα μεγάλο αριθμό από XML αρχεία, όπου πολλές φορές αυτά τα αρχεία γίνονται πολύ μεγάλα και είναι δύσκολη η συντήρηση τους και η κατανόηση από τρίτους και στις περισσότερες περιπτώσεις είσαι αναγκασμένος να επαναλάβεις τις ίδιες ρυθμίσεις για κάθε νέα εφαρμογή.

Ο Hansson σκοπίμως δημιούργησε το Rails έτσι ώστε να μην χρειάζεται ιδιαίτερες ρυθμίσεις, αυτό είχε ως επακόλουθο να υπάρχουν κάποιες συμβάσεις. Το αποτέλεσμα ήταν να μην χρειάζεται μεγάλος αριθμός αρχείων που περιέχουν ρυθμίσεις. Στην πραγματικότητα, εάν δεν χρειάζεται να αλλάξουμε αυτές τις προεπιλεγμένες ρυθμίσεις, το μόνο που θέλει το Rails από μας, είναι να αλλάξουμε ένα μόνο μικρό αρχείο. Αυτό το αρχείο αφορά τις ρυθμίσεις για την σύνδεση με την βάσης δεδομένων.

02-database.yml

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  timeout: 5000
test:
  adapter: sqlite3
  database: db/test.sqlite3
  timeout: 5000
production:
  adapter: sqlite3
  database: db/production.sqlite3
  timeout: 5000
```

Εικόνα 3.2 - Παράδειγμα αρχείου για την σύνδεση με την βάση δεδομένων

Άλλες συμβάσεις που υπάρχουν στο Rails αφορούν την ονομασία για τα αντικείμενα που συσχετίζονται με την βάση δεδομένων, και η διαδικασία με την οποία οι ρυθμιστές (controllers) αντιστοιχίζεται με τα μοντέλα (models) και τις όψης (views).

3.5 Agile Development (ευέλικτος προγραμματισμός)

Οι περισσότερο παραδοσιακές προσεγγίσεις για την ανάπτυξη εφαρμογών, συνήθως οργανώνουν ένα μακροχρόνιο και στατικό σχέδιο για την επίτευξη των στόχων και των αναγκών μιας εφαρμογής, χρησιμοποιώντας "προφητικές" μεθόδους. Αυτός ο τύπος προγραμματισμού συνήθως προσεγγίζει την εφαρμογή "από κάτω προς τα πάνω", που σημαίνει πως ο προγραμματιστής ξεκινάει να δουλεύει με τα τα δεδομένα και έπειτα ασχολείται με το σχεδιαστικό μέρος.

Σε αντίθεση οι μέθοδοι agile development, χρησιμοποιούν μια πιο προσαρμοσμένη προσέγγιση. Μικρές ομάδες προγραμματιστών αναλαμβάνουν να ολοκληρώσουν με επαναλαμβανόμενο τρόπο μικρά μέρη της εφαρμογής. Πριν ξεκινήσει μια επανάληψη, η ομάδα επανεκτιμά τους στόχους και τις προτεραιότητες του τμήματος που έχει αναλάβει να αναπτύξει, με βάση την μέχρι τότε πορεία όλου του project. Οι στόχοι και οι προτεραιότητες ενδέχεται να έχουν αλλάξει σε σχέση με την προηγούμενη επανάληψη, όποτε χρειάζεται να επαναπροσδιοριστούν. Οι προγραμματιστές που χρησιμοποιούν μεθόδους Agile development, αρχίζουν το "χτίσιμο" της εφαρμογής "από πάνω προς τα κάτω", ξεκινώντας με το σχεδιαστικό μέρος που συνήθως είναι ένα απλό σκίτσο της διεπαφής σχεδιασμένο σε χαρτί.

Όταν μια εφαρμογή αναπτύσσεται με μεθόδους agile development, μοιάζει να είναι εκτός ελέγχου κατά την διάρκεια του προγραμματισμού, εξαιτίας των συνεχών επαναπροσδιορισμών των στόχων και των προτεραιοτήτων. Ξοδεύοντας όμως λίγο χρόνο στη σχεδιασμό λειτουργικών προδιαγραφών και πιο μακροπρόθεσμων σχεδίων, χωρίς να βέβαια να ξεφεύγουμε από την λογική του agile development, η ανάπτυξη της εφαρμογής αποκτά συνοχή και ξεπερνιέται το πρόβλημα που αναφέρθηκε.

Μερικά παραδείγματα που φανερώνουν πως το Rails χρησιμοποιεί μεθόδους του agile development είναι:

- Έχουμε την δυνατότητα να ξεκινήσουμε με τον σχεδιασμό (layout), πριν ξεκινήσουμε να

ασχολούμαστε με το κομμάτι των δεδομένων. Δεν χρειάζεται να ασχοληθούμε ξανά με το σχεδιαστικό μέρος όταν ξεκινήσουμε να δίνουμε λειτουργικότητα στην εφαρμογή μας, εκτός αν επιθυμούμε να αλλάξουμε την εμφάνισή της.

- Αντίθετα με γλώσσες όπως η C και η Java, μια Rails εφαρμογή δεν χρειάζεται να κάνει μεταγλώττιση για να γίνει εκτελέσιμη. Ο κώδικας σε Ruby ερμηνεύεται αμέσως έτσι δεν χρειάζεται την οποιαδήποτε μεταγλώττιση για να γίνει εκτελέσιμος. Η αλλαγή κάποιου τμήματος κώδικα δίνει στον προγραμματιστή αμέσως αποτελέσματα και έτσι αυξάνεται η ταχύτητα της ανάπτυξης μιας εφαρμογής.

- Το Rails παρέχει ένα χρήσιμο framework αυτόματης δοκιμής (testing) κάποιου τμήματος του κώδικα της εφαρμογής. Οι προγραμματιστές που κάνουν χρήση αυτού του εργαλείου είναι σίγουροι πως δεν υπάρχει περίπτωση κάνοντας δοκιμές να χαλάσουν ότι έχουν φτιάξει μέχρι εκείνη την στιγμή.

- Όταν χρειαστεί να αλλαχτεί κάποιο τμήμα του κωδικά για να βελτιστοποιηθεί η εφαρμογή αλλάζοντας κάποιες προτεραιότητες, ή αν χρειαστεί να προστεθούν περισσότερα χαρακτηριστικά στην εφαρμογή, γίνεται πολύ απλά αν ο προγραμματιστής χρησιμοποιήσει τις αρχές του DRY. Αυτό ισχύει γιατί χρειάζονται πολύ λιγότερες αλλαγές αν χρησιμοποιήσουμε ήδη υπάρχοντα κώδικα που έχει γραφτεί μια φορά και επαναχρησιμοποιείται σε διάφορα σημεία της εφαρμογής.

3.6 Το Ruby on Rails και η βάση δεδομένων

Το μεγαλύτερα μέρος των εφαρμογών, αν όχι όλες, χρειάζονται μια βάση δεδομένων ώστε να αποθηκεύουν τα δεδομένα και να τα χρησιμοποιούν όποτε τα χρειάζονται. Το Framework Ruby On Rails υποστηρίζει μια αρκετά μεγάλη γκάμα από συστήματα διαχείρισης βάσεων δεδομένων. Μια ενδεικτική λίστα με τις διευθύνσεις που μπορούμε να βρούμε τους προσαρμογείς (adapters) είναι:

- MySQL http://www.tmtm.org/en/mysql/ruby/

- Oracle http://rubyforge.org/projects/ruby-oci8/
- SQL Server http://rubyforge.org/projects/ruby-dbi/
- SQLite http://rubyforge.org/projects/sqlite-ruby/
- Postgres http://ruby.scripting.ca/postgres/
- Firebird http://rubyforge.org/projects/fireruby/
- DB2 http://rubyforge.org/projects/ruby-ibm/

Το προεπιλεγμένο σύστημα διαχείρισης βάσεων δεδομένων που έχει το Rails, όταν ξεκινάει για μια νέα εφαρμογή είναι το SQLite, το οποίο μπορεί πολύ εύκολα να αλλάξει, εγκαθιστώντας τον προσαρμογέα για την βάση δεδομένων, και παραμετροποιώντας ένα μικρό αρχείο που έχει πληροφορίες σχετικά με την σύνδεση με την βάση δεδομένων. Τα περιεχόμενα αυτού του αρχείου είναι:

development:
adapter: sqlite3
database: db/development.sqlite3
pool: 5
timeout: 5000
test: This organizes your application components. It's got subdirectories that hold the view (views and helpers), controller (controllers), and the backend business logic (models).
adapter: sqlite3
database: db/test.sqlite3
pool: 5
timeout: 5000
production:
adapter: sqlite3
database: db/production.sqlite3
pool: 5
timeout: 5000

Οπότε για να αλλάξουμε σύστημα διαχείρισης βάσεων δεδομένων στο οποίο ίσως η πρόσβαση να απαιτεί κωδικό για κάθε χρήστη, απλά αλλάζουμε το αρχείο και το φέρνουμε σε μια μορφή σαν και αυτή:

development:

```
  adapter: mysql
  database: library_development
  username: root
  password: [password]
  host: localhost
test:
  adapter: mysql
  database: library_test
  username: root
  password: [password]
  host: localhost
production:
  adapter: mysql
  database: library_production
  username: root
  password: [password]
  host: localhost
```

Το Rails διαχειρίζεται την βάση με το Object Relational Mapping. Σύμφωνα με αυτή την λογική, ένας πίνακας σε μια βάση δεδομένων διαχειρίζεται ως μία κλάση, τα πεδία του πίνακα ως τα δεδομένα-μεταβλητές της κλάσης και οι εγγραφές του πίνακα ως αντικείμενα της κλάσης. Οι κλάσεις αυτές έχουν αρκετές συναρτήσεις για την διαχείριση της βάσης. Έτσι για παράδειγμα εάν υπάρχει ένας πίνακας με το όνομα Rooms, θα υπάρχει και μια κλάση στην εφαρμογή με το όνομα Room, η οποία θα αντιστοιχίζεται με τον πίνακα στην βάση δεδομένων. Έτσι για να βρούμε μια εγγραφή στην βάση με Id = 1. Την βρίσκουμε πολύ απλά γράφοντας

```
x=Room.find(1)
```

Ενώ για να αλλαχτεί μια τιμή σε κάποιο πεδίο γράφουμε

```
x.price = 40
```

Αυτή η λογική κάνει πιο εύκολη την συγγραφή κώδικα μιας και ο προγραμματιστής δεν χρειάζεται να μάθει γλώσσες όπως η SQL για να υποβάλλει ερωτήματα στην βάση, ώστε να αντλήσει πληροφορίες που είναι αποθηκευμένες στην βάση δεδομένων, αυτό γίνεται πολύ απλά με τυπικές γνώσεις της γλώσσας Ruby και της έννοιας της αντικειμενοστρέφειας.

3.7 Η δομή των καταλόγων (directory structure) σε μια ROR εφαρμογή

Χρησιμοποιώντας το βασικό script για την δημιουργία μιας νέας εφαρμογής, δημιουργείτε η βασική δομή των καταλόγων για μια εφαρμογή βασισμένη στο Ruby On Rails η οποία είναι:

demo/

..../app
...../controller
...../helpers
...../models
...../views
...../layouts
..../components
..../config
..../db
..../doc
..../lib
..../log
..../public
..../script
..../test
..../tmp
..../vendor
README
Rakefile

Ο σκοπός των βασικών καταλόγων και υποκαταλόγων είναι:

- **app**: Εδώ βρίσκονται οι κατάλογοι όπου βρίσκονται τα αρχεία των μοντέλων (models), των όψεων (views) και των ελεγκτών (controllers).
- **app/controllers:** Σε αυτόν τον υποκατάλογο βρίσκονται τα αρχεία των ελεγκτών.
- **app/views:** Εδώ θα ψάξει η εφαρμογή για να βρει ένα αρχείο όψης.
- **app/models:** Σε αυτόν τον υποκατάλογο βρίσκονται τα αρχεία των μοντέλων.
- **app/helpers:** Εδώ βρίσκονται βοηθητικά αρχεία τα οποία βοηθούν ώστε να μείνει σχετικά μικρός και ευανάγνωστος ο κώδικας που υπάρχει στους ελεγκτές και στα μοντέλα.
- **app/views/layouts:** Εδώ φυλάσσονται τα tamplates τα οποία θα χρησιμοποιηθούν σε συνδυασμό με τις όψεις.

- **components:** Σε αυτόν τον κατάλογο υπάρχουν μικρές αυτόνομες εφαρμογές που χρησιμοποιούνται στα μοντέλα, στους ελεγκτές και στις όψεις.

- **config:** Σε αυτόν τον κατάλογο βρίσκονται αρχεία ρυθμίσεων που χρειάζονται για την εφαρμογή, όπως το αρχείο για την σύνδεση με την βάση δεδομένων database.yml.

- **db:** Συνήθως σε μια Rails εφαρμογή υπάρχουν μοντέλα τα απoία αντιστοιχούν σε έναν πίνακα της βάσης δεδομένων. Σε αυτόν τον κατάλογο τοποθετούνται scripts τα όποια διαχειρίζονται την βάση δεδομένων.

- **doc:** Η Ruby έχει ένα πολύ χρήσιμο εργαλείο το οποίο ονομάζεται rdoc και παράγει αρχεία με οδηγίες χρήσης (docimentation files) μιας εφαρμογής με βάση τα σχόλια που υπάρχουν στον κώδικα. Τα αρχεία αυτά τοποθετούνται στον κατάλογο doc.

- **lib:** Σε αυτόν τον κατάλογο τοποθετούνται οι επιπλέον βιβλιοθήκες που χρησιμοποιεί η εφαρμογή.

- **log:** Τα αρχεία στα οποία καταγράφεται το ιστορικό από διάφορα λάθη φυλάσσονται σε αυτόν τον φάκελο. Τέτοια αρχεία είναι τα αρχεία που καταγράφουν τα λάθη του server (server.log), της εφαρμογής (development.log) κτλ.

- **public:** Εδώ βρίσκονται αρχεία τα οποία χρησιμοποιεί η εφαρμογή και δεν αλλάζουν, όπως JavaScript αρχεία, εικόνες, stylesheets κτλ.

- **scripts:** Σε αυτόν τον κατάλογο φυλάσσονται scripts τα οποία συνήθως χειρίζονται κάποια εργαλεία. Για παράδειγμα εργαλεία τα οποία παράγουν κώδικα, ξεκινούν τον server κτλ.

- **tests:** Εδώ υπάρχουν αρχεία τα οποία παράγονται όταν εργαζόμαστε στο test environment.

- **tmp:** Σε αυτόν τον κατάλογο βρίσκονται προσωρινά αρχεία τα οποία ίσως χρειαστούν κάποια στιγμή.

- **vendor:** Εδώ υπάρχουν βιβλιοθήκες οι οποίες είναι για άλλες εφαρμογές, όπως η βάση δεδομένων.

- **README:** Αυτό το αρχείο παρέχει πληροφορίες σχετικά με το Framework Ruby On Rails.

- **rakefile:** Αυτό το αρχείο είναι παρόμοιο με το αρχείο makefile που υπάρχει στο Linux. Αυτό το αρχείο βοηθάει στο "χτίσιμο", την πακετοποίηση (packaging), και στην δοκιμή (testing) του κώδικα.

3.8 Ο server webrick

Ο webrick είναι ο προεπιλεγμένος server που χρησιμοποιεί το Framework Ruby On Rails. Είναι γραμμένος σε γλώσσα Ruby από τους Mashayoshi Takahashi και Yuuzou Gotou. Εκδόθηκε το 2003 κάτω από open source άδεια και είναι cross-platform, δηλαδή λειτουργεί στα περισσότερα λειτουργικά συστήματα. Είναι ένα πολύ χρήσιμο εργαλείο για κάθε Ruby On Rails προγραμματιστή, μιας και είναι απόλυτα συμβατός με το Framework. Η προεπιλεγμένη πόρτα που τρέχει ο server είναι η 3000, αλλά πολύ εύκολα μπορεί να ξεκινήσει σε άλλη πόρτα.

```
nim@nimbox:~/Desktop/ruby/ror/hotel$ ruby script/server
=> Booting WEBrick...
=> Rails 2.2.3 application started on http://127.0.0.1:3000
=> Ctrl-C to shutdown server; call with --help for options
[2010-03-15 22:40:39] INFO  WEBrick 1.3.1
[2010-03-15 22:40:39] INFO  ruby 1.8.7 (2009-06-12) [x86_64-linux]
[2010-03-15 22:40:39] INFO  WEBrick::HTTPServer#start: pid=4230 port=3000
```

Εκκίνηση του server webrick

ΚΕΦΑΛΑΙΟ 4ο: Βάση δεδομένων και η MySQL

4.1 Ορισμός της βάσης δεδομένων

Μια Βάση Δεδομένων (Database) είναι ένας οργανωμένος τρόπος αποθήκευσης πληροφοριών και πρόσβασης σε αυτές. Μια βάση δεδομένων είναι κάτι παραπάνω από μια απλή συλλογή αποθηκευμένων στοιχείων.

Ένας άλλος ορισμός είναι ότι μια βάση δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από δεδομένα και από το κατάλληλο λογισμικό, τα οποία χρησιμοποιώντας το υλικό (hardware) βοηθούν στην ενημέρωση και πληροφόρηση των χρηστών.

Ένα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται *Σύστημα Διαχείρισης Βάσεων Δεδομένων (*DBMS*, Database Management System)* και με την βοήθειά του μπορούμε να αποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε, εμφανίσουμε ή και διαγράψουμε τα αποθηκευμένα δεδομένα.

Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι :

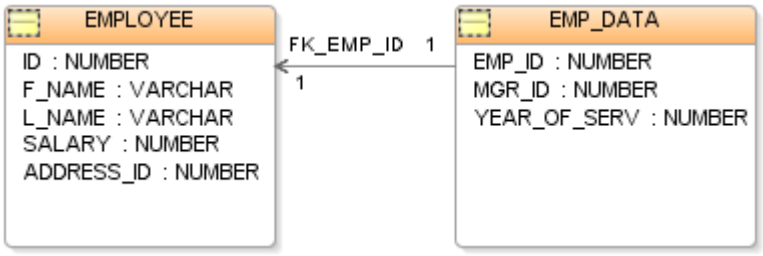
- **Ολοκληρωμένα (Integrated)**, δηλαδή τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων.

- **Καταμεριζόμενα (Shared)**, δηλαδή να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.

4.2 Οι σχεσιακές βάσεις δεδομένων

Το σχεσιακό μοντέλο περιγράφει τη Βάση Δεδομένων και οργανώνει τις εγγραφές με βάση τις σχέσεις. Γι αυτό το λόγο μια Βάση δεδομένων σχεδιασμένη με βάση το σχεσιακό μοντέλο, μπορεί εύκολα να υλοποιηθεί με ένα μοντέλο Οντοτήτων - Συσχετίσεων.

Στις σχεσιακές Βάσεις Δεδομένων, οι εγγραφές οργανώνονται σε πίνακες. Οι πίνακες σε μια σχεσιακή Βάση Δεδομένων, αποτελούνται από μια ή περισσότερες στήλες που αντιστοιχούν σε τιμές πεδίων (ή στα χαρακτηριστικά για τα μοντέλα Οντοτήτων - Συσχετίσεων) και από γραμμές που αντιστοιχούν σε εγγραφές για αυτά τα πεδία.



Εικόνα 4.1 - Παράδειγμα σχεσιακής βάσης δεδομένων.

4.3 Σύστημα διαχείρισης βάσεων δεδομένων

Το Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, Database Management System) είναι ένα σύνολο από "προγράμματα" που επιτρέπουν τον χειρισμό των δεδομένων μιας ή περισσοτέρων βάσεων δεδομένων που ανήκουν στο ίδιο σύστημα. Το DBMS περιέχει κάποια εργαλεία γενικής χρήσης για να μπορούμε να δημιουργούμε και να χειριζόμαστε τα δεδομένα.

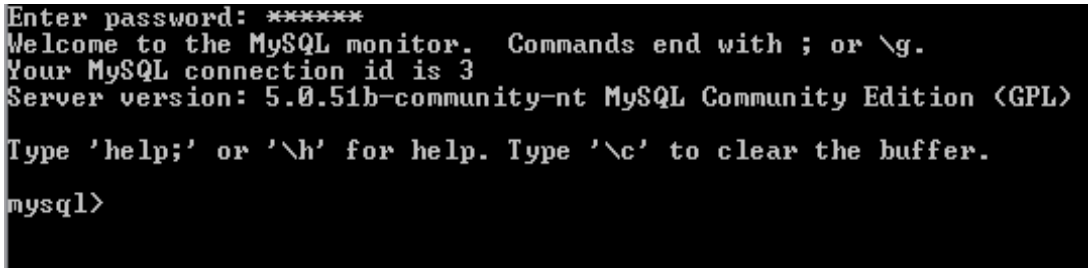
Στα νεώτερα DBMS μπορούμε να έχουμε άμεση πληροφόρηση χωρίς να απαιτείται η παρουσία ενός προγραμματιστή. Τα δεδομένα ενός DBMS μπορούν να χρησιμοποιηθούν σε κάθε μορφής ερώτημα (query) ώστε να αντλήσουμε τις πληροφορίες θέλουμε.

4.4 Η MySQL

Η MySQL είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων, που υποστηρίζει πολλούς χρήστες. Αναπτύχθηκε κυρίως από τον Mickael Widenius και είναι γραμμένη σε γλώσσα C και C++ . Εκδόθηκε το 1995 κάτω από την άδεια GNU/GPL, είναι δηλαδή μια Open Source εφαρμογή. Επίσης η MySQL είναι μια cross-platform εφαρμογή, που σημαίνει πως είναι συμβατή σχεδόν με όλα τα λειτουργικά συστήματα.

Η MySQL μπορεί πολύ εύκολα να εγκατασταθεί σε έναν υπολογιστή χωρίς να χρειαστεί ιδιαίτερη παραμετροποίηση. Για παράδειγμα στις περισσότερες διανομές Linux ο Package Manager βρίσκει, κατεβάζει και εγκαθιστά την MySQL πολύ γρήγορα και εύκολα και είναι έτοιμη μετά την εγκατάσταση για χρήση.

Αν και η MySQL ξεκίνησε σαν ένα χαμηλότερου επιπέδου σύστημα διαχείρισης βάσεων δεδομένων, αυτή την στιγμή θεωρείται ένα πλήρες επαγγελματικό εργαλείο με τεράστιες δυνατότητες.



Εικόνα 4.2 - Η default γραμμή εντολών της MySQL

4.4.1 Ο MySQL Query Browser

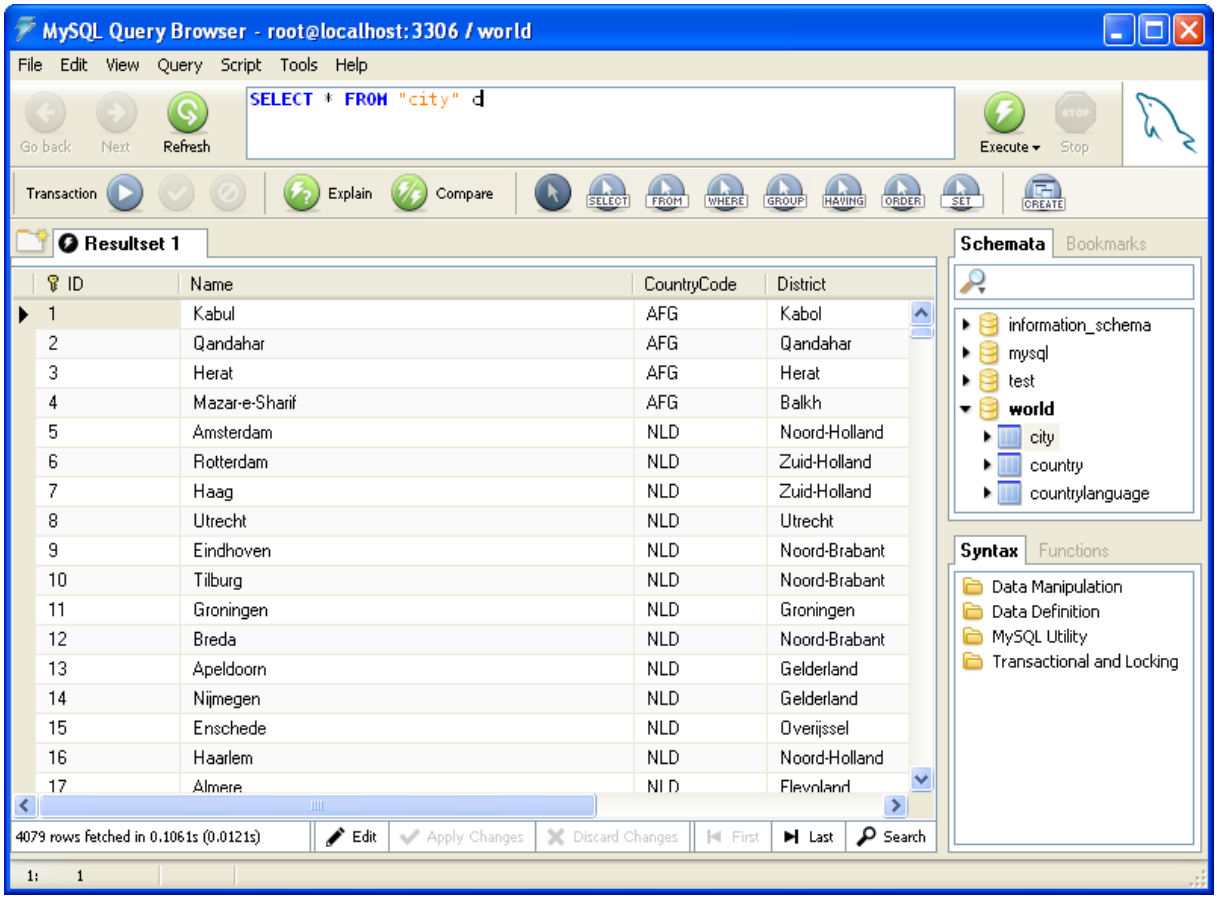
Ο MySQL Query Browser είναι ένα Open Source εργαλείο, που παρέχεται από την MySQL. Δίνει την δυνατότητα στον χρήστη να σχεδιάσει, να τροποποιήσει και να εκτελέσει ερωτήματα στην βάση δεδομένων σε γραφικό περιβάλλον και να αντλήσει πληροφορίες από τα δεδομένα που είναι στην βάση, ακόμη και να αλλάξει τα χαρακτηριστικά της βάσης.

Ο MySQL Query Browser παρέχει επίσης μια γραμμή εντολών στην οποία οι ενέργειες που γίνονται γραφικά, μεταφράζονται εκεί σε ερωτήματα SQL, δίνοντας την δυνατότητα στον χρήστη να τροποποιήσει αυτά τα ερωτήματα και να τα εκτελέσει, μέσα από την εφαρμογή.

Ο MySQL Query Browser είναι ένα πάρα πολύ χρήσιμο εργαλείο διότι δίνει την δυνατότητα στον χρήστη να χειριστεί την βάση του, και γράφοντας ο ίδιος ερωτήματα SQL, αλλά

και να εκτελέσει ερωτήματα με ένα πιο διαισθητικό, γραφικό τρόπο.

Ο MySQL Query Browser αυτή την στιγμή βρίσκεται στην έκδοση 1.2.12 και είναι σχεδιασμένος να συνεργάζεται με τις εκδόσεις 4.0 και πάνω της MySQL.



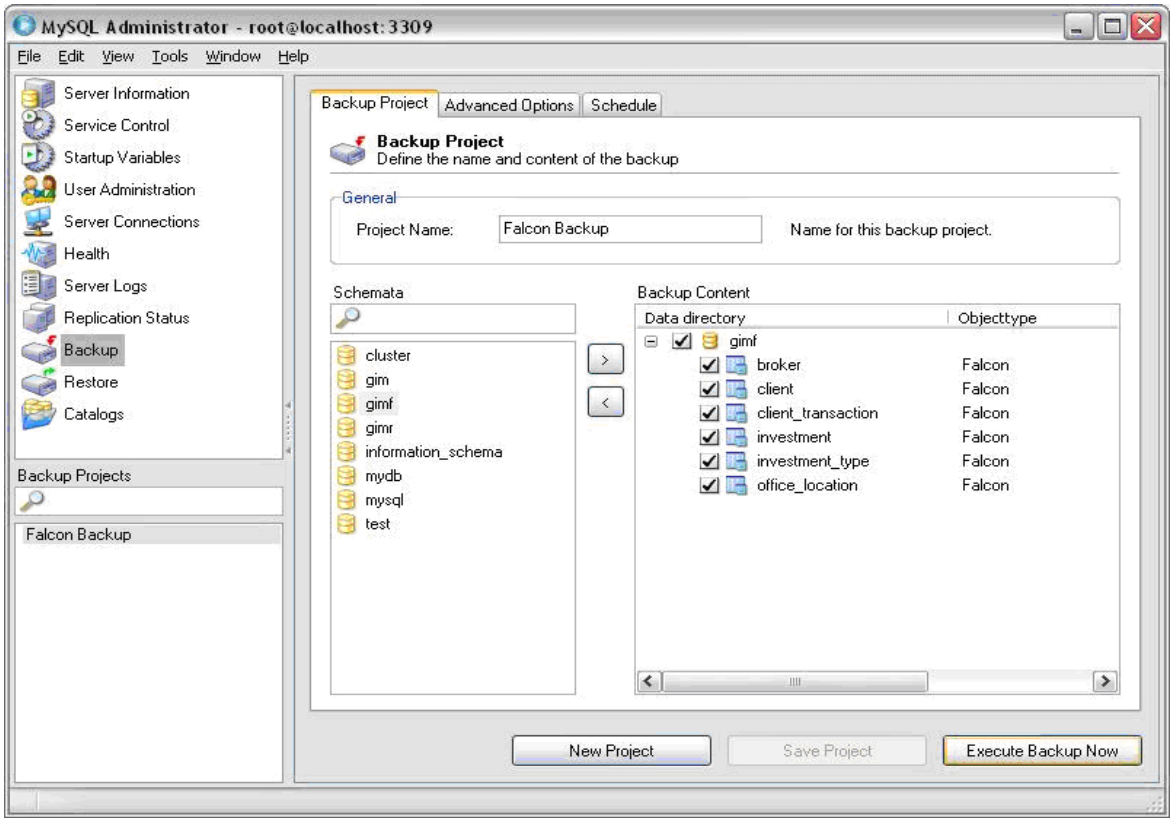
Εικόνα 4.3 - Ο MySQL Query Browser

4.4.2 Ο MySQL Administrator

Ο MySQL Administrator είναι μια Open Source εφαρμογή που παρέχεται από την MySQL και επιτρέπει στον χρήστη της να εκτελεί ενέργειες διαχειριστή σε ένα εύχρηστο, γραφικό περιβάλλον. Ο διαχειριστής έχει την δυνατότητα μέσω της εφαρμογής να δημιουργεί αντίγραφα ασφαλείας, να διαχειρίζεται τους χρήστες και τα δικαιώματα που έχει ο κάθε χρήστης

στις βάσεις δεδομένων, να παρακολουθεί την καταγραφή γεγονότων (logs) του server της MySQL, να εκκινεί και να σταματάει τον server της MySQL κτλ.

Ο MySQL Administrator αυτή την στιγμή βρίσκεται στην έκδοση 1.2.12 και είναι σχεδιασμένος να συνεργάζεται με τις εκδόσεις 4.0 και πάνω της MySQL.



Εικόνα 4.4 - Ο MySQL Administrator

4.4.3 Τα πλεονεκτήματα της MySQL

Τα πλεονεκτήματα της MySQL είναι γενικά πάρα πολλά, δεν μπορούμε όμως να πούμε πως είναι πολύ περισσότερα από τα πλεονεκτήματα που προσφέρουν άλλα συστήματα διαχείρισης βάσεων δεδομένων. Η επιλογή ενός συστήματος διαχείρισης βάσεων δεδομένων γίνεται σύμφωνα με τις απαιτήσεις της εφαρμογής, την συμβατότητα με τις τεχνολογίες που θα χρησιμοποιηθούν και φυσικά την εμπειρία του προγραμματιστή πάνω στο σύστημα διαχείρισης της βάσης. Παρόλα αυτά τα σημαντικότερα πλεονεκτήματα που προσφέρει η MySQL είναι:

- Αξιοπιστία μιας και η MySQL είναι ένα μεγάλο Open Source project που αναπτύσσεται από μια μεγάλη εταιρία, και από πάρα πολλούς προγραμματιστές
- Είναι πολύ ταχύτερη από άλλα συστήματα όπως η PostgreSQL
- Ασφάλεια αφού απαιτείται πιστοποίηση για την είσοδο οποιουδήποτε χρήστη. Επίσης οι κωδικοί αποθηκεύονται κρυπτογραφημένοι.
- Δυνατότητα πρόσβασης από πολλούς χρήστες αφού δίνει την δυνατότητα ταυτόχρονης σύνδεσης πολλών χρηστών.
- Το κόστος για την απόκτηση είναι μηδαμινό, διότι η MySQL είναι Open Source όποτε οποιοσδήποτε μπορεί να πάρει τον κωδικά, να τον χρησιμοποιήσει και να τον βελτιώσει δωρεάν.
- Η MySQL μπορεί να διαχειριστεί τεράστιες σε μέγεθος βάσεις δεδομένων καθώς το προεπιλεγμένο μέγεθος μιας βάσης που μπορεί να διαχειριστεί είναι 4GB το οποίο μπορεί να φτάσει έως και τα 8TB
- Είναι συμβατή σχεδόν με όλα τα λειτουργικά συστήματα οπότε είναι cross-platform.
- Ευκολία στην χρήση αφού τα εργαλεία γραφικής διασύνδεσης χρήστη που προσφέρει, κάνουν την χρήση της MySQL πολύ απλή.
- Και τέλος προσφέρει όλα τα πλεονεκτήματα που μπορεί να προσφέρει μια Open Source εφαρμογή.

ΚΕΦΑΛΑΙΟ 5ο: Η ΕΦΑΡΜΟΓΗ ON-LINE ΚΡΑΤΗΣΕΩΝ ΔΩΜΑΤΙΩΝ

Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση της εφαρμογής on-line κρατήσεων δωματίων και παρουσίαση και επεξήγηση του κώδικα που γράφτηκε για την υλοποίησή της. Το κεφάλαιο θα χωριστεί σε τρία μέρη. Το πρώτο μέρος θα αναφέρεται στην κατασκευή του κορμού της εφαρμογής. Στο δεύτερο μέρος θα παρουσιαστούν οι υπηρεσίες στην πλευρά του χρήστη καθώς και ο κώδικάς τους. Και στο τρίτο μέρος θα παρουσιαστούν οι λειτουργίες και οι υπηρεσίες στην πλευρά του διαχειριστή. Επίσης η εφαρμογή αναπτύχθηκε σε περιβάλλον Linux (Debian), οπότε όταν γίνεται αναφορά σε εντολές, οι εντολές αυτές δίνονται στην κονσόλα του λειτουργικού.

5.1 Η δημιουργία του κορμού της εφαρμογής

Το framework Ruby On Rails δίνει την δυνατότητα στον προγραμματιστή, με μια εντολή να δημιουργήσει τον βασικό κορμό της Web εφαρμογής που πρόκειται να κατασκευάσει. Έτσι λοιπόν δίνοντας την εντολή

rails hotel

το Rails δημιουργεί την βασική δομή της εφαρμογής.

```
nim@nimbox:~/Desktop/ptixiaki$ rails hotel
    create
    create  app/controllers
    create  app/helpers
    create  app/models
    create  app/views/layouts
    create  config/environments
```

create config/initializers

create config/locales

create db

create doc

create lib

create lib/tasks

create log

create public/images

create public/javascripts

create public/stylesheets

create script/performance

create script/process

create test/fixtures

create test/functional

.

.

.

.

.

Έτσι λοιπόν έχουμε την βασική δομή της εφαρμογής.

Το επόμενο βήμα που πρέπει να γίνει είναι η δημιουργία και η σύνδεση με την βάση δεδομένων. Αφού στη MySQL δημιουργηθεί μια βάση δεδομένων με όνομα hotel_develpoment, πρέπει να παραμετροποιηθεί το αρχείο database.yml που βρίσκεται στην θέση hotel/config, για να ορίσουμε ότι το σύστημα διαχείρισης της βάσης είναι η MySQL μιας και το προεπιλεγμένο που χρησιμοποιεί το Rails είναι το SQLite. Ανοίγοντας λοιπόν το αρχείο με κάποιον text editor

αλλάζουμε το περιεχόμενό του. Το νέο περιεχόμενο του αρχείου για σύνδεση με την MySQL είναι αυτό

```
# MySQL.  Versions 4.1 and 5.0 are recommended.
# Install the MySQL driver:
#   gem install mysql
# On Mac OS X:
#   sudo gem install mysql -- --with-mysql-dir=/usr/local/mysql
# On Mac OS X Leopard:
#       sudo env ARCHFLAGS="-arch i386" gem install mysql -- --with-mysql-config=/usr/local/mysql/bin/mysql_config
#       This sets the ARCHFLAGS environment variable to your native architecture
# On Windows:
#   gem install mysql
#       Choose the win32 build.
#       Install MySQL and put its /bin directory on your path.
#
# And be sure to use new-style password hashing:
#   http://dev.mysql.com/doc/refman/5.0/en/old-client.html
development:
  adapter: mysql
  encoding: utf8
  database: hotel_development
  username: root
  password: 123456
 # Warning: The database defined as "test" will be erased and
```

 # re-generated from your development database when you run "rake".

 # Do not set this db to the same as development or production.

#test:

 # adapter: mysql

 # encoding: utf8

 # database: hotel_test

 # pool: 5

 # username: root

 # password:

 # socket: /var/run/mysqld/mysqld.sock

#production:

 # adapter: mysql

 # encoding: utf8

 # database: hotel_production

 # pool: 5

 # username: root

 # password:

 # socket: /var/run/mysqld/mysqld.sock

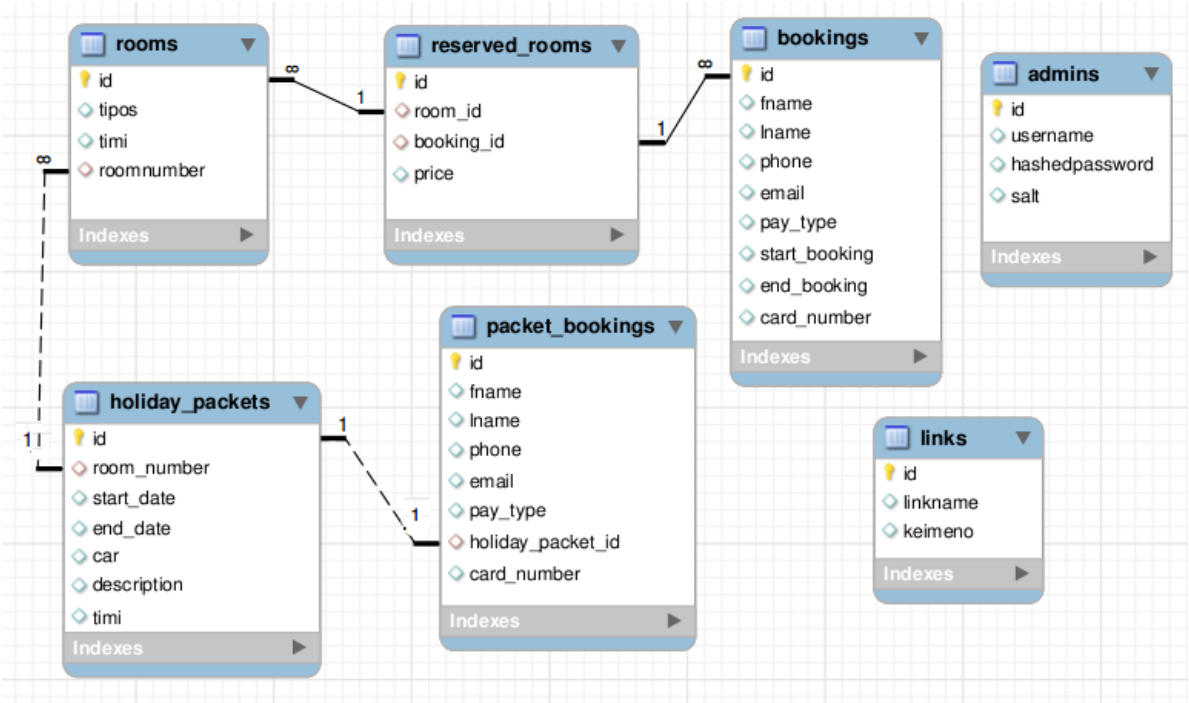
Οι γραμμές που στην αρχή περιέχουν το σύμβολο δίεση (#), θεωρούνται σχόλια και δεν λαμβάνονται υπόψη από την εφαρμογή.

Η σύνδεση με την βάση δεδομένων ολοκληρώνεται με την εντολή

rake db:migrate

η εντολή αυτή δίνεται κάθε φορά που γίνονται αλλαγές που αφορούν την βάση δεδομένων, όπως για παράδειγμα προσθήκης μιας στήλης σε κάποιον πίνακα.

Η βάση δεδομένων αποτελείται από τέσσερις πίνακες οι οποίοι είναι: admins, bookings, holiday packets, links, packet_bookings, reserved_rooms, rooms. Το σχεσιακό μοντέλο της βάσης δεδομένων φαίνεται σε μια πιο ολοκληρωμένη μορφή παρακάτω:



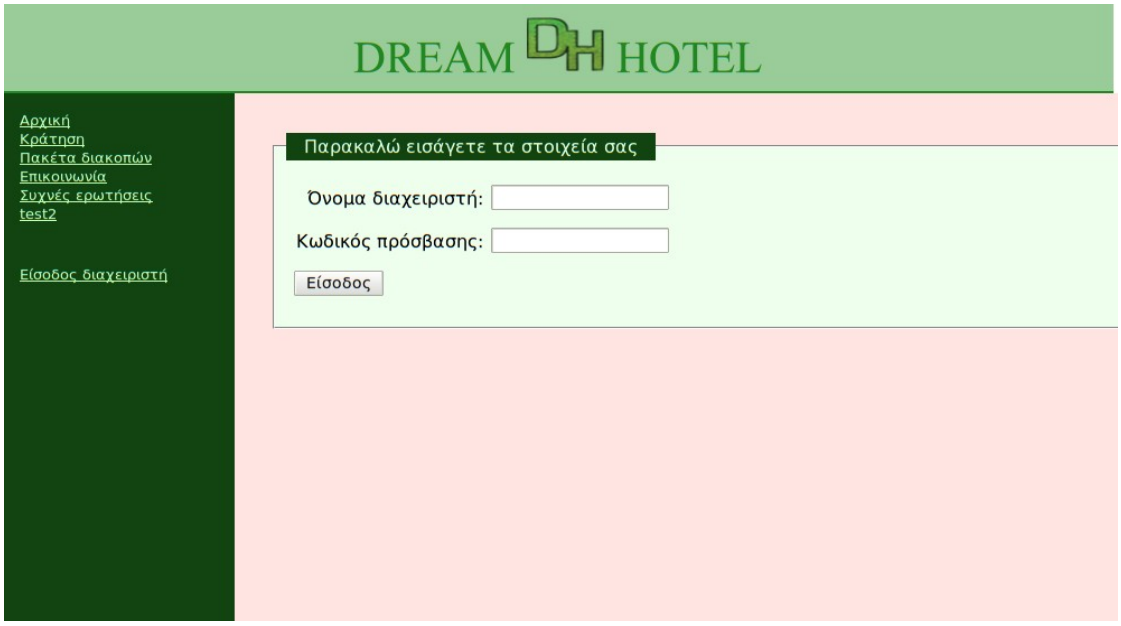
Εικόνα 5.1 - Η βάση δεδομένων της εφαρμογής

5.2 Οι λειτουργίες του διαχειριστή

Για να θεωρηθεί μια εφαρμογή ολοκληρωμένη θα πρέπει να παρέχει στον διαχειριστή όσο το δυνατόν μεγαλύτερη ευελιξία, ώστε να μπορεί να κάνει εύκολα τις ενέργειες που θεωρεί πως χρειάζονται μέσα από την εφαρμογή.

5.2.1 Είσοδος του διαχειριστή

Για να ξεκινήσει ο διαχειριστής την οποιαδήποτε ενέργεια θα πρέπει να συμπληρώσει μια φόρμα με το username του και το password ώστε να γίνει ταυτοποίηση. Η σελίδα με την φόρμα που συναντάει κάποιος που θέλει να συνδεθεί σαν διαχειριστής είναι:



Εικόνα 5.2 - Η φόρμα για την είσοδο του διαχειριστή

ο κώδικας για την εμφάνιση της φόρμας είναι:

```
<div class="hotel-form">
<fieldset>
  <legend>Παρακαλώ εισάγετε τα στοιχεία σας</legend>
   <% form_tag do %>
    <p>
      <label for="username">Όνομα διαχειριστή:</label>
      <%= text_field_tag :username, params[:username], :size => 15 %>
    </p>
    <p>
      <label for="password">Κωδικός πρόσβασης:</label>
      <%= password_field_tag :password, params[:password], :size => 15 %>
    </p>
    <p>
      <%= submit_tag "Login" %>
    </p>
   <% end %>
  </fieldset>
</div>
```

(hotel/app/views/administrator/login.html.erb)

Πατώντας το κουμπί *login* καλείται η μέθοδος:

```
  def login
    session[:admin_id] = nil
    if request.post?
      admin = Admin.authenticate(params[:username], params[:password])
      if admin
        session[:admin_id] = admin.id
        redirect_to(:action => "index")
      else
        flash.now[:notice] = "Το όνομα χρήστη ή ο κωδικός πρόσβασης είναι λάθος !"
      end
    end
  end
```

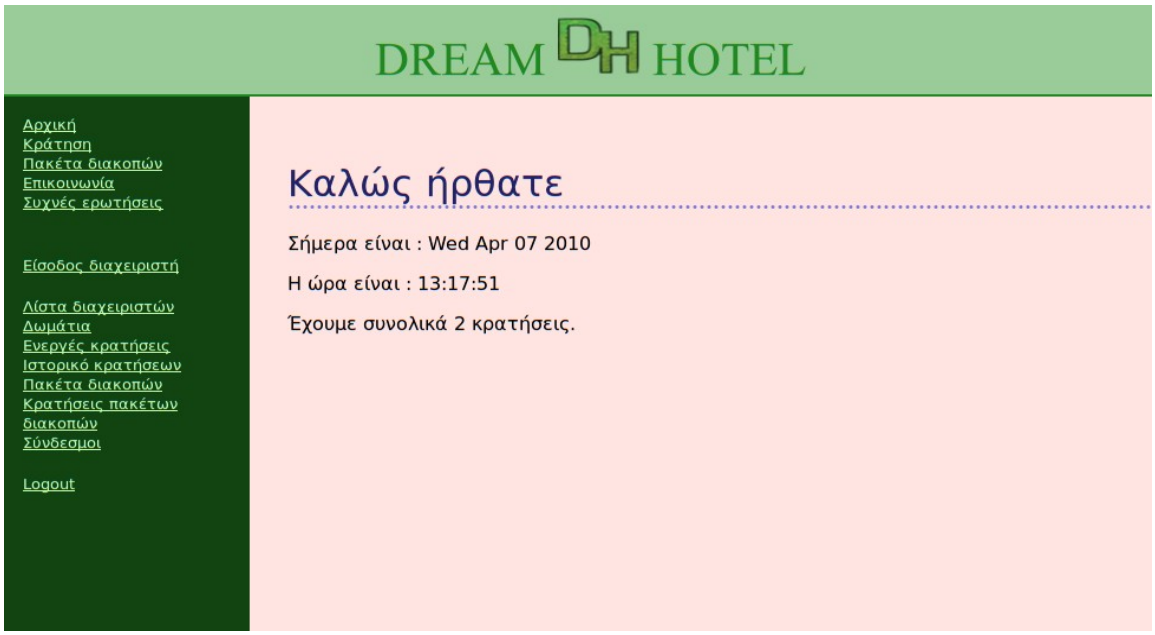
(hotel/app/controllers/administrator_controller.rb)

μέσα σε αυτή την μέθοδο καλείται η μέθοδος:

```
  def self.authenticate(username, password)
    admin = self.find_by_username(username)
    if admin
      expected_password = encrypted_password(password, admin.salt)
      if admin.hashed_password != expected_password
```

```
        admin = nil
      end
    end
    admin
  end
```

(hotel/app/models/admin.rb)



Εικόνα 5.3 - Η αρχική σελίδα για τον διαχειριστή μετά το login

Σε αυτές τις μεθόδους γίνεται έλεγχος για το εάν τα στοιχεία που έχουν μπει στην φόρμα είναι σωστά. Εάν τα στοιχεία είναι σωστά γίνεται redirect στην αρχική σελίδα του διαχειριστή και ξεκινάει ένα session που παίρνει ως παράμετρο το id του διαχειριστή και πλέον εμφανίζονται τα links για τις λειτουργίες του διαχειριστή, διαφορετικά εμφανίζεται ένα μήνυμα που λέει πως τα στοιχεία που έχουν δοθεί δεν είναι έγκυρα.

5.2.2 Οι διαχειριστές

Κάθε διαχειριστής έχει την δυνατότητα να δει την λίστα με όλους τους διαχειριστές και να κάνει ενέργειες που αφορούν τον ίδιο αλλά και τους άλλους διαχειριστές, πατώντας στον σύνδεσμο *λίστα διαχειριστών*. Η σελίδα που θα εμφανιστεί είναι:

Εικόνα 5.4 - Η λίστα των διαχειριστών

ο κώδικας για την εμφάνιση της λίστας διαχειριστών είναι:

* *

<fieldset>

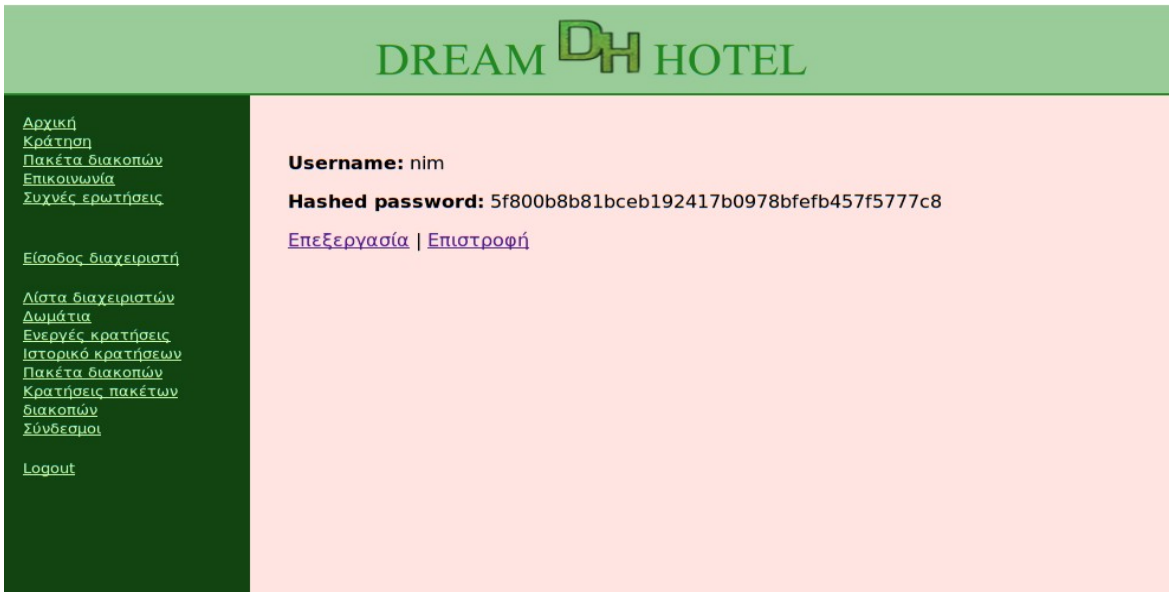
<legend>Διαχειριστές</legend>

```
  <table cellpadding="5" cellspacing="0">
  <tr>
    <th>Username</th>
  </tr>
<% for admin in @admins %>
    <tr valign="top" class="<%= cycle('list-line-odd', 'list-line-even') %>">
    <td><%=h admin.username %></td>
    <td class="list-actions"><%= link_to 'Εμφάνιση', admin %>
    <%= link_to 'Επεξεργασία', edit_admin_path(admin) %>
     <%= link_to 'Διαγραφή', admin, :confirm => 'Είσαστε σίγουρος ότι θέλετε να διαγράψετε τον διαχειριστή;', :method => :delete %></td>
  </tr>
<% end %>
</table>
<br />
<%= link_to 'Προσθήκη νέου διαχειριστή', new_admin_path %>
</fieldset>
</div>
```

(hotel/app/views/admins/index.html.erb)

Εδώ δίνεται η δυνατότητα της διαχείρισης των διαχειριστών, όπου ένας διαχειριστής

μπορεί να δει και να επεξεργαστεί τα στοιχεία των διαχειριστών ή να διαγράψει κάποιον από την λίστα.



Εικόνα 5.5 - Εμφάνιση πληροφοριών διαχειριστή

Εάν υπάρχει μόνο ένας διαχειριστής στην εφαρμογή δεν είναι δυνατή η διαγραφή του. Η μέθοδος που καλείται για την διαγραφή είναι:

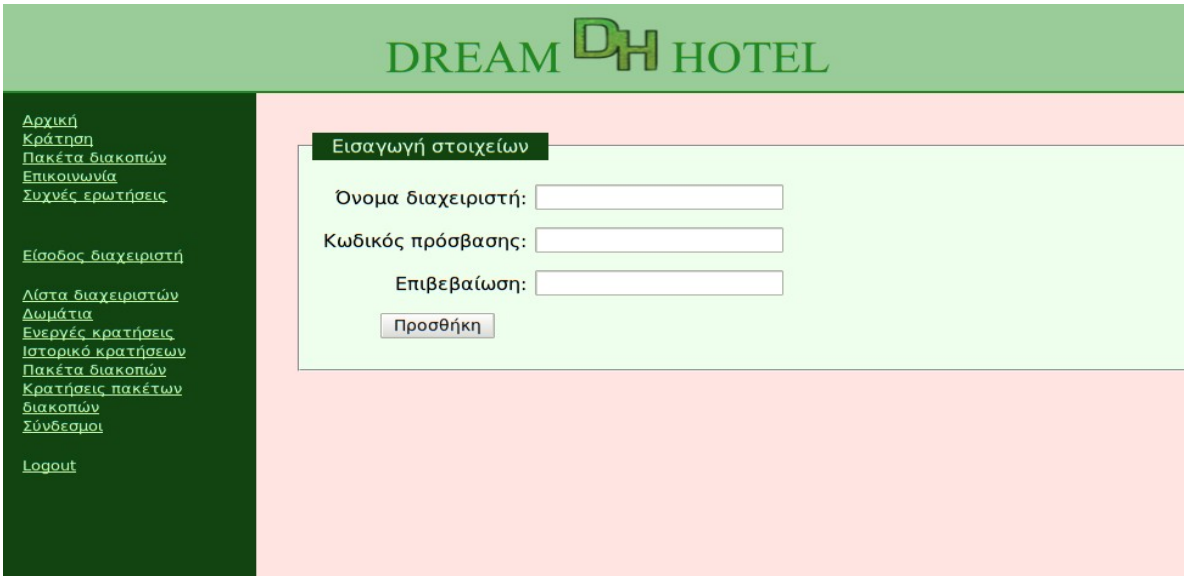
```
def destroy
  @admin = Admin.find(params[:id])
  begin
    @admin.destroy
    flash[:notice] = "Ο διαχειριστής #{ @admin.username } έχει διαγραφεί"
  rescue Exception => e
    flash[:notice] = e.message
  end
  respond_to do |format|
```

```
    format.html { redirect_to(admins_url) }
    format.xml  { head :ok }
  end
end
```

και η μέθοδος που καλείται για να γίνει έλεγχος για το εάν ο διαχειριστής είναι ο μοναδικός στην λίστα είναι:

```
def after_destroy
  if Admin.count.zero?
    raise "Δεν επιτρέπεται η διαγραφή αυτού του διαχειριστή, επειδή είναι ο τελευταίος"
  end
end
```

Ένας διαχειριστής έχει επίσης την δυνατότητα να προσθέσει άλλους διαχειριστές στην εφαρμογή.



Εικόνα 5.6 - Προσθήκη νέου διαχειριστή

μόλις πατηθεί το κουμπί *προσθήκη* γίνεται έλεγχος για την εγκυρότητα των τιμών που δόθηκαν. Αυτό γίνεται από τους κανόνες (validation rules) που έχουν προστεθεί στο μοντέλο admin.

```
validates_presence_of :username

validates_uniqueness_of :username

validates_confirmation_of :password

validate :password_non_blank
```

την λειτουργία των τριών πρώτων κανόνων την παρέχει το rails, ενώ για τον τον τελευταίο κανόνα χρειάστηκε να υλοποιηθεί η μέθοδος:

```
def password_non_blank

  errors.add_to_base("Παρακαλώ είσαγετε κωδικό πρόσβασης.") if hashed_password.blank?

end
```

αφού ικανοποιηθούν οι παραπάνω κανόνες γίνεται προσθήκη του νέου διαχειριστή στη βάση δεδομένων χρησιμοποιώντας την μέθοδο:

```
def create

  @admin = Admin.new(params[:admin])

  respond_to do |format|

    if @admin.save

      flash[:notice] = "O Administrator #{ @admin.username } έχει προστεθεί."

      format.html { redirect_to(:action => :index) }

      format.xml  { render :xml => @admin, :status => :created, :location => @admin }
```

```
    else
      format.html { render :action => "new" }
      format.xml  { render :xml => @admin.errors, :status => :unprocessable_entity }
    end
  end
end
def create
  @admin = Admin.new(params[:admin])
  respond_to do |format|
    if @admin.save
      flash[:notice] = "O Administrator #{ @admin.username } έχει προστεθεί."
      format.html { redirect_to(:action => :index) }
      format.xml  { render :xml => @admin, :status => :created, :location => @admin }
    else
      format.html { render :action => "new" }
      format.xml  { render :xml => @admin.errors, :status => :unprocessable_entity }
    end
  end
end
```

(hotel/app/controller/admins_controller.rb)

Για μεγαλύτερη ασφάλεια αποφασίστηκε ο κωδικός πρόσβασης των διαχειριστών να αποθηκεύεται στην βάση δεδομένων κρυπτογραφημένος. Για την κρυπτογράφηση χρησιμοποιήθηκε η τεχνική των salted κωδικών. Οι μέθοδοι που υλοποιούν την παραπάνω ενέργεια είναι:

```
def password=(pwd)
  @password = pwd
  return if pwd.blank?
  create_new_salt
  self.hashed_password = Admin.encrypted_password(self.password, self.salt)
end

def create_new_salt
  self.salt = self.object_id.to_s + rand.to_s
end

def self.encrypted_password(password, salt)
  string_to_hash = password + "wibble" + salt
  Digest::SHA1.hexdigest(string_to_hash)
end
```

(hotel/app/controller/admins_controller.rb)

Για την υλοποίηση της κρυπτογράφησης χρησιμοποιήθηκε το gem digest/sha1.

require 'digest/sha1'

5.2.3 Η διαχείριση των δωματίων

Η διαχείριση των δωματίων και η διαχείριση των κρατήσεων μπορούν να θεωρηθούν τα σημαντικότερα τμήματα όσον αφορά την διαχείριση, μιας και ο κύριος σκοπός της εφαρμογής είναι οι on-line κρατήσεις δωματίων και πακέτων διακοπών.

Η αρχική σελίδα που εμφανίζεται πατώντας το link που αφορά την διαχείριση των δωματίων, παρέχει μια λίστα με τα δωμάτια που έχουν προστεθεί στην εφαρμογή με τα χαρακτηριστικά τους (τιμή, αριθμός δωματίου, τύπος δωματίου), καθώς επίσης και διάφορες ενέργειες που μπορεί να κάνει ο διαχειριστής στα δωμάτια (διαγραφή, επεξεργασία, προσθήκη νέου δωματίου).

Εικόνα 5.7 - Η λίστα δωματίων

ο κώδικας που εμφανίζει την παραπάνω σελίδα είναι:

```
<div id="room-list">

<div class="hotel-form">

<fieldset>

  <legend>Λίστα δωματίων</legend>

  <table cellpadding="5" cellspacing="0">

   <% for room in @rooms %>

    <tr valign="top" class="<%= cycle('list-line-odd', 'list-line-even') %>">

      <td width="60%">

                  <p>Αριθμός δωματίου: <span class="list-title"><%=h room.roomnumber %></span><br /> </p>

      <p>Τύπος: <span class="list-title"><%=h room.tipos %></span><br /> </p>

      <p>Τιμή: <span class="list-title"><%=h room.timi %></span><br /> </p>

      </td>

      <td class="list-actions">

        <%= link_to 'Εμφάνιση', room %><br/>

        <%= link_to 'Επεξεργασία', edit_room_path(room) %><br/>

        <%= link_to 'Διαγραφή', room,

              :confirm => 'Είσαστε σίγουρος ότι θέλετε να διαγράψετε το δωμάτιο από την λίστα;',

              :method => :delete %>

      </td>

    </tr>
```

```
  <% end %>

  </table>

<br />

<%= link_to 'Προσθήκη νέου δωματίου', new_room_path %>

</fieldset>

</div>

</div>
```

(hotel/app/views/rooms/index.html.erb)

Η μέθοδος που καλείται για την διαγραφή είναι:

```
def destroy

    @room = Room.find(params[:id])

    @room.destroy

    respond_to do |format|

      format.html { redirect_to(rooms_url) }

      format.xml  { head :ok }

    end

  end
```

(hotel/app/controllers/rooms_controller.rb)

Εικόνα 5.8 - Διαγραφή δωματίου από την λίστα

Ο διαχειριστής μπορεί πολύ εύκολα να επεξεργαστεί τα χαρακτηριστικά των δωματίων που υπάρχουν στην εφαρμογή. Πατώντας το link για την επεξεργασία του δωματίου εμφανίζεται μια φόρμα που περιέχει τα τα χαρακτηριστικά του εκάστοτε δωματίου, τα οποία μπορεί ο διαχειριστής να αλλάξει.

Εικόνα 5.9 - Επεξεργασία δωματίου

Ο κώδικας για την εμφάνιση της φόρμας που αφορά την επεξεργασία του δωματίου είναι:

```
<h1>Επεξεργασία δωματίου</h1>

<% form_for(@room) do |f| %>

  <%= f.error_messages %>

 <p>

   <%= f.label "Αριθμός δωματίου" %>

   <%= f.text_field :roomnumber %>

  </p>

  <p>

   <%= f.label "Τύπος" %>

   <%= f.select :tipos, ["Μονόκλινο","Δίκλινο","Τρίκλινο","Σουίτα"]%>

  </p>

  <p>

   <%= f.label "Τιμή" %>

   <%= f.text_field :timi %>

  </p>

  <p>

   <%= f.submit "Επιβεβαίωση" %>

  </p>

<% end %>
```

```
<%= link_to 'Εμφάνιση', @room %> |

<%= link_to 'Επιστροφή', rooms_path %>
```

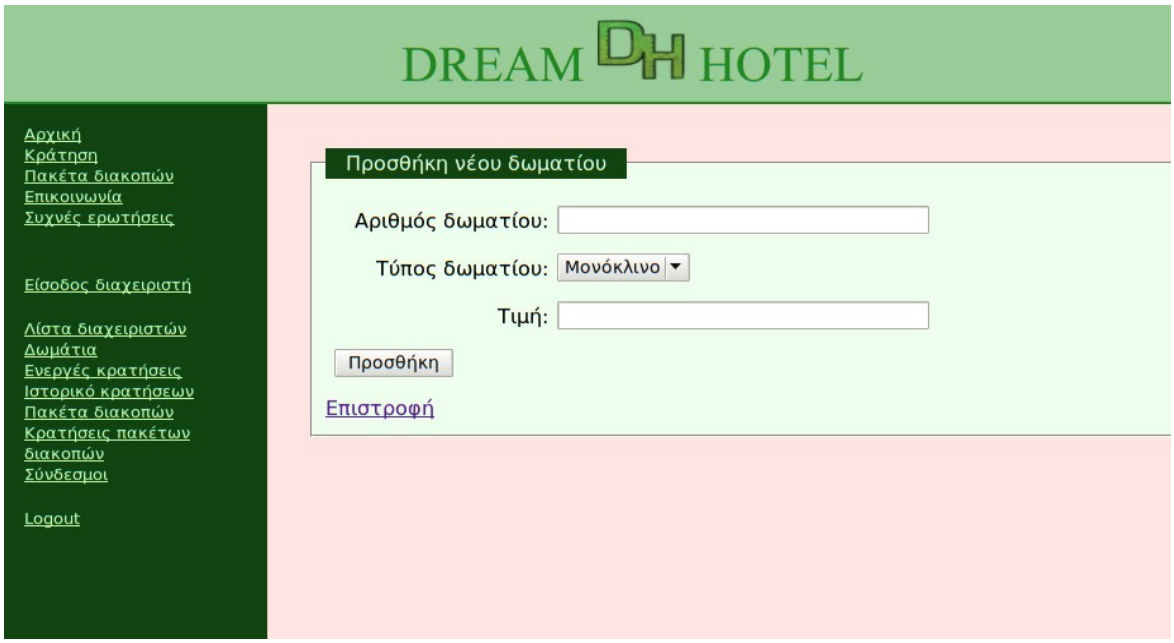
(hotel/app/views/rooms/edit.html.erb)

Η μέθοδος που αποθηκεύει τις αλλαγές μετά την επεξεργασία του δωματίου είναι:

```
  def update

    @room = Room.find(params[:id])



    respond_to do |format|

      if @room.update_attributes(params[:room])

        flash[:notice] = 'Οι αλλαγές στο δωμάτιο έγιναν με επιτυχία.'

        format.html { redirect_to(@room) }

        format.xml  { head :ok }

      else

        format.html { render :action => "edit" }

        format.xml  { render :xml => @room.errors, :status => :unprocessable_entity }

      end

    end

  end
```

(hotel/app/controllers/rooms_controller.rb)

Η εφαρμογή δίνει την δυνατότητα στον διαχειριστή να προσθέσει επιπλέον δωμάτια στην εφαρμογή. Έτσι πατώντας στον σύνδεσμο *προσθήκη νέου δωματίου* εμφανίζεται η σελίδα που περιέχει την φόρμα που αφορά το νέο δωμάτιο.



Εικόνα 5.10 - Προσθήκη νέου δωματίου

ο κώδικας για την εμφάνιση αυτής της σελίδας είναι:

```
<div class="hotel-form">
 <fieldset>
<legend>Προσθήκη νέου δωματίου</legend>
<% form_for(@room) do |f| %>
  <%= f.error_messages %>
  <p>
    <%= f.label "Αριθμός δωματίου:" %>
    <%= f.text_field :roomnumber %>
  </p>
```

```
  <p>
    <%= f.label "Τύπος δωματίου:" %>
    <%= f.select :tipos, ["Μονόκλινο","Δίκλινο","Τρίκλινο","Σουίτα"]%>
  </p>
  <p>
    <%= f.label "Τιμή:" %>
    <%= f.text_field :timi %>
  </p>
  <p>
    <%= f.submit "Προσθήκη" %>
  </p>
<% end %>
<%= link_to 'Επιστροφή', rooms_path %>
 </fieldset>
</div>
```

(hotel/app/views/rooms/new.html.erb)

αφού γίνει έλεγχος για το εάν ικανοποιούνται οι κανόνες που ορίστηκαν στο μοντέλο room, τότε καλείται η μέθοδος που αποθηκεύει το νέο δωμάτιο στην βάση δεδομένων.

Οι κανόνες για το μοντέλο room είναι:

```
  validates_presence_of :tipos, :timi
  validates_numericality_of :timi
```

```
validates_uniqueness_of :roomnumber, :message => "Το δωμάτιο υπάρχει ήδη στην λίστα."

validate :price_at_least_a_cent
```

Οι τρεις πρώτοι κανόνες υπάρχουν έτοιμοι στο rails, ενώ ο τέταρτος έπρεπε να υλοποιηθεί. Η συνάρτηση που τον υλοποιεί είναι:

```
def price_at_least_a_cent

    errors.add(:timi, 'Η τιμή του δωματίου πρέπει να είναι μεγαλύτερη από 0.01') if timi.nil? || timi < 0.01

  end
```

(hotel/app/models/room.rb)

Μετά την ικανοποίηση των κανόνων καλείται η μέθοδος που αποθηκεύει το νέο δωμάτιο στην βάση δεδομένων.

```
def create

  @room = Room.new(params[:room])

  respond_to do |format|

    if @room.save

      flash[:notice] = 'Το δωμάτιο προστέθηκε με επιτυχία.'

      format.html { redirect_to(@room) }

      format.xml  { render :xml => @room, :status => :created, :location => @room }

    else

      format.html { render :action => "new" }
```

```
    format.xml  { render :xml => @room.errors, :status => :unprocessable_entity }
  end
 end
end
```

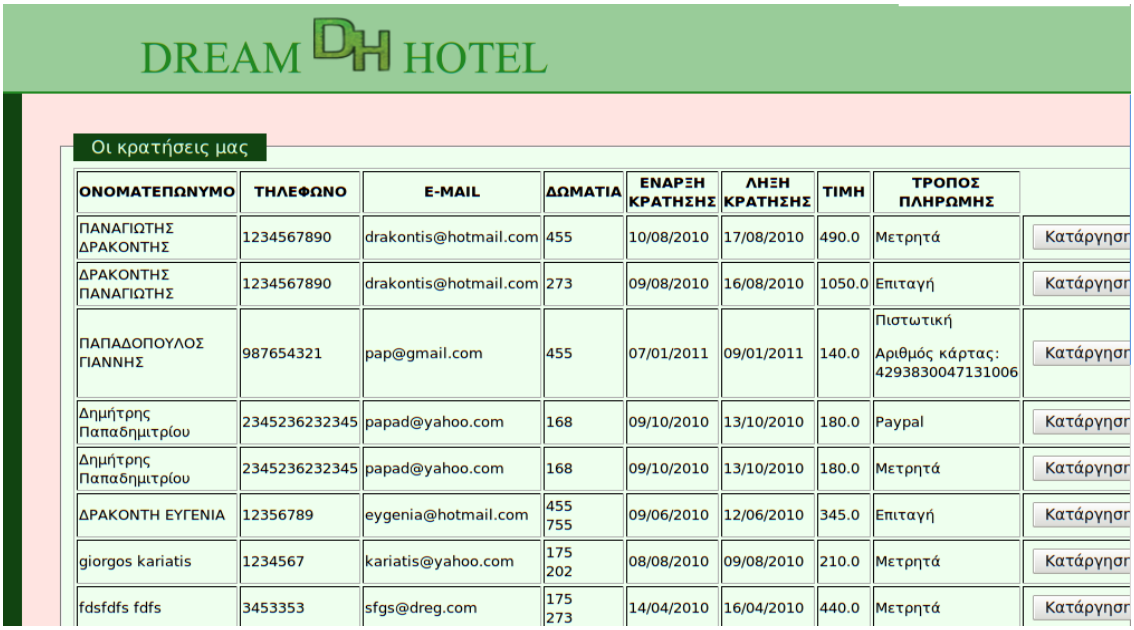
(hotel/app/controllers/room_controller.rb)

5.2.4 Οι κρατήσεις δωματίων

Οι κρατήσεις δωματίων χωρίζονται σε δύο κατηγορίες. Στις ενεργές κρατήσεις και στο ιστορικό κρατήσεων. Στις ενεργές κρατήσεις εμφανίζονται οι κρατήσεις που η ημερομηνία λήξης της κράτησης δεν έχει περάσει, δηλαδή εάν μια κράτηση λήγει στις 10-7-2010 και η ημέρα που ο διαχειριστής κοιτάει τις κρατήσεις είναι πριν την ημερομηνία λήξης της κράτησης, η κράτηση αυτή θεωρείται ενεργή, διαφορετικά η ημερομηνία λήξης της κράτησης θα έχει παρέλθει και η κράτηση θα περάσει στην κατηγορία με το ιστορικό κρατήσεων.

5.2.4.1 Οι ενεργές κρατήσεις

Πατώντας στο link *ενεργές κρατήσεις* ο διαχειριστής βλέπει μια λίστα με τις ενεργές κρατήσεις, καθώς επίσης και διάφορες πληροφορίες που αφορούν την κράτηση, όπως το όνομα στο οποίο έγινε η κράτηση, τον αριθμό δωματίου, την ημερομηνία έναρξης και λήξης της κράτησης, τον τρόπο πληρωμής κτλ. Επιπλέον ο διαχειριστής μπορεί να καταργήσει μια κράτηση.



Εικόνα 5.11 - Οι ενεργές κρατήσεις

ο κώδικας για την εμφάνιση της σελίδας είναι:

```
def show_active_bookings
    @books = Booking.find(:all)
    @bookings = Array.new
    @bookings = []
    $nowdate = Time.now
    @books.each { |book|
      if $nowdate < book.end_booking
          @bookings << book
```

```
        end
      }
  end
```

(hotel/app/controllers/administrator_controller.rb)

```
<h1><font size = '5'> <b> Οι κρατήσεις μας </b></font></h1>
<% if @bookings.empty? || @bookings.nil? %>
<b>
<p>Δεν υπάρχουν κρατήσεις</p>
</b>
<% else %>
<font size = '2'>
<table border="1">
<tr>
<th>ΟΝΟΜΑΤΕΠΩΝΥΜΟ</th>
<th>ΤΗΛΕΦΩΝΟ</th>
<th>E-MAIL</th>
<th>ΔΩΜΑΤΙΑ</th>
<th>ΕΝΑΡΞΗ ΚΡΑΤΗΣΗΣ</th>
<th>ΛΗΞΗ ΚΡΑΤΗΣΗΣ</th>
```

```
<th>ΤΙΜΗ</th>

<th>ΤΡΟΠΟΣ ΠΛΗΡΩΜΗΣ</th>

</tr>

<% for booking in @bookings -%>

<tr>

<td><%=h booking.fname %> <%=h booking.lname %></td>

<td><%=h booking.phone %></td>

<td><%=h booking.email %></td>

<td>

<% @reserved_rooms = ReservedRoom.find(:all) -%>

   <% total_price = 0.0 -%>

   <% for room in @reserved_rooms -%>

   <% if booking.id == room.booking_id -%>

    <% total_price = total_price+room.price -%>

      <% @rooms = Room.find(:all) -%>

      <% for domatio in @rooms -%>

        <%= domatio.roomnumber if domatio.id == room.room_id %>

      <% end %>

     <br/>

   <% end %>
```

<% end %>

</td>

<td><%=h booking.start_booking.to_s[0..10] %></td>

<td><%=h booking.end_booking.to_s[0..10] %></td>

<td><%=h total_price %></td>

<td><%=h booking.pay_type %> <% unless booking.card_number==nil %> Αριθμός κάρτας: <%= booking.card_number %> <% end %> </td>

<td><%= button_to 'Κατάργηση', {:action => 'destroy_booking', :id => booking}, :confirm => "Είσαστε σίγουρος ότι θέλετε να καταργήσετε την κράτηση;" %></td>

</tr>

<% end %>

</table>

<% end %>

**

(hotel/app/views/administrator/show_active_bookings.html.erb)

Εδώ πριν εμφανιστεί η σελίδα με την λίστα των κρατήσεων καλείται η συνάρτηση *show_active_bookings* (που έχει το ίδιο όνομα με την το αρχείο που εμφανίζει της κρατήσεις, έτσι καλείται αυτόματα) και αποθηκεύει σε έναν πίνακα όλες τις ενεργές κρατήσεις, έπειτα το αρχείο show_active_bookings.html.erb εμφανίζει τα περιεχόμενα του πίνακα.

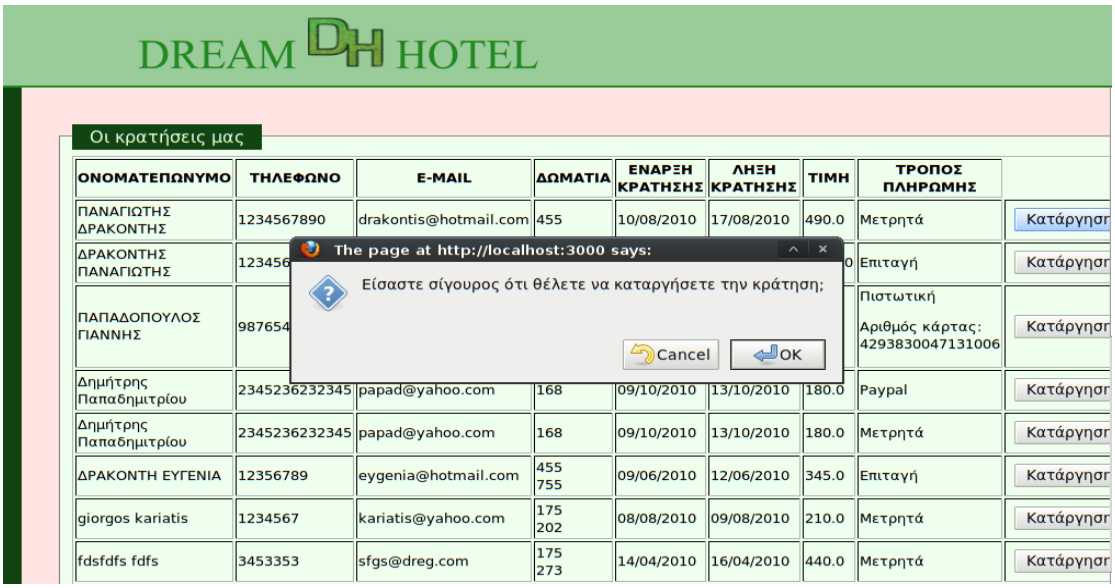
πατώντας στο κουμπί *κατάργηση* καλείται η μέθοδος:

def destroy_booking

```
    @book = Booking.find(params[:id])
    @booked_rooms = ReservedRoom.find(:all)
    @booked_rooms.each {|room|
      if room.booking_id == @book.id
        room.destroy
      end
    }
    @book.destroy
    flash[:notice] = "Η κράτηση έχει καταργηθεί."
    redirect_to :action => 'show_active_bookings'
  end
```

(hotel/app/controllers/administrator_controller.rb)

και η κράτηση διαγράφεται από την λίστα των κρατήσεων.



Εικόνα 5.12 - Κατάργηση κράτησης

5.2.4.2 Το ιστορικό κρατήσεων

Πατώντας στο link *ιστορικό κρατήσεων, ο διαχειριστής* μπορεί να δει μια λίστα με όλες τις κρατήσεις που έχουν γίνει μέχρι εκείνη την στιγμή.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΤΗΛΕΦΩΝΟ	E-MAIL	ΔΩΜΑΤΙΑ	ΕΝΑΡΞΗ ΚΡΑΤΗΣΗΣ	ΛΗΞΗ ΚΡΑΤΗΣΗΣ	ΠΟΣΟ ΠΛΗΡΩΜΗΣ	ΤΡΟΠΟΣ ΠΛΗΡΩΜ
gerassimos stavridis	123456789	stavridis@stavridis.gr		18/03/2010	19/03/2010	55.0	Επιταγή
ΠΑΝΑΓΙΩΤΗΣ ΔΡΑΚΟΝΤΗΣ	1234567890	drakontis@hotmail.com	455	10/08/2010	17/08/2010	490.0	Μετρητά
ΔΡΑΚΟΝΤΗΣ ΠΑΝΑΓΙΩΤΗΣ	1234567890	drakontis@hotmail.com	273	09/08/2010	16/08/2010	1050.0	Επιταγή
ΠΑΠΑΔΟΠΟΥΛΟΣ ΓΙΑΝΝΗΣ	987654321	pap@gmail.com	455	07/01/2011	09/01/2011	140.0	Πιστωτική Αριθμός κάρτας: 4293830047131006
Δημήτρης Παπαδημητρίου	2345236232345	papad@yahoo.com	168	09/10/2010	13/10/2010	180.0	Paypal
Δημήτρης Παπαδημητρίου	2345236232345	papad@yahoo.com	168	09/10/2010	13/10/2010	180.0	Μετρητά
ΔΡΑΚΟΝΤΗ ΕΥΓΕΝΙΑ	12356789	eygenia@hotmail.com	455 755	09/06/2010	12/06/2010	345.0	Επιταγή
giorgos kariatis	1234567	kariatis@yahoo.com	175 202	08/08/2010	09/08/2010	210.0	Μετρητά
fdsfdfs fdfs	3453353	sfgs@dreg.com	175 273	14/04/2010	16/04/2010	440.0	Μετρητά

Εικόνα 5.13 - Το ιστορικό κρατήσεων

ο κώδικας για την παραπάνω σελίδα που εμφανίζει το ιστορικό κρατήσεων είναι:

```
def bookings_history
    @bookings = Booking.find(:all)
end
```

(hotel/app/controllers/administrator_controller.rb)

```
<div class="hotel-form">
<fieldset>
<legend>Το ιστορικό των κρατήσεών μας</legend>
```

```
<% if @bookings.empty? || @bookings.nil? %>

<b>

<p>Δεν υπάρχουν κρατήσεις στο ιστορικό</p>

</b>

<% else %>

<font size = '2'>

<table border = "1">

<tr>

<th>ΟΝΟΜΑΤΕΠΩΝΥΜΟ</th>

<th>ΤΗΛΕΦΩΝΟ</th>

<th>E-MAIL</th>

<th>ΔΩΜΑΤΙΑ</th>

<th>ΕΝΑΡΞΗ ΚΡΑΤΗΣΗΣ</th>

<th>ΛΗΞΗ ΚΡΑΤΗΣΗΣ</th>

<th>ΠΟΣΟ ΠΛΗΡΩΜΗΣ</th>

<th>ΤΡΟΠΟΣ ΠΛΗΡΩΜΗΣ</th>

</tr>

<% for booking in @bookings -%>

<tr>

<td><%=h booking.fname %> <%=h booking.lname %></td>
```

```
<td><%=h booking.phone %></td>

<td><%=h booking.email %></td>

<td>

<% @reserved_rooms = ReservedRoom.find(:all) -%>

   <% total_price = 0.0 -%>

   <% for room in @reserved_rooms -%>

   <% if booking.id == room.booking_id -%>

    <% total_price = total_price+room.price -%>

      <% @rooms = Room.find(:all) -%>

      <% for domatio in @rooms -%>

        <%= domatio.roomnumber if domatio.id == room.room_id %>

      <% end %>

     <br/>

   <% end %>

   <% end %>

</td>

<td><%=h booking.start_booking.to_s[0..10] %></td>

<td><%=h booking.end_booking.to_s[0..10] %></td>

<td><%=h total_price %></td>

<td><%=h booking.pay_type %> <% unless booking.card_number==nil %> <p> Αριθμός κάρτας: <%= booking.card_number %></p> <% end %> </td>
```

</tr>

<% end %>

</table>

<% end %>

**

</fieldset>

* *

(hotel/app/show/administrator/booking_history.html.erb)

Εδώ πριν εμφανιστεί η σελίδα με την λίστα των κρατήσεων καλείται η συνάρτηση *booking_history* (που έχει το ίδιο όνομα με την το αρχείο που εμφανίζει της κρατήσεις, έτσι καλείται αυτόματα) και αποθηκεύει σε έναν πίνακα όλες τις ενεργές κρατήσεις, έπειτα το αρχείο booking_history.html.erb εμφανίζει τα περιεχόμενα του πίνακα.

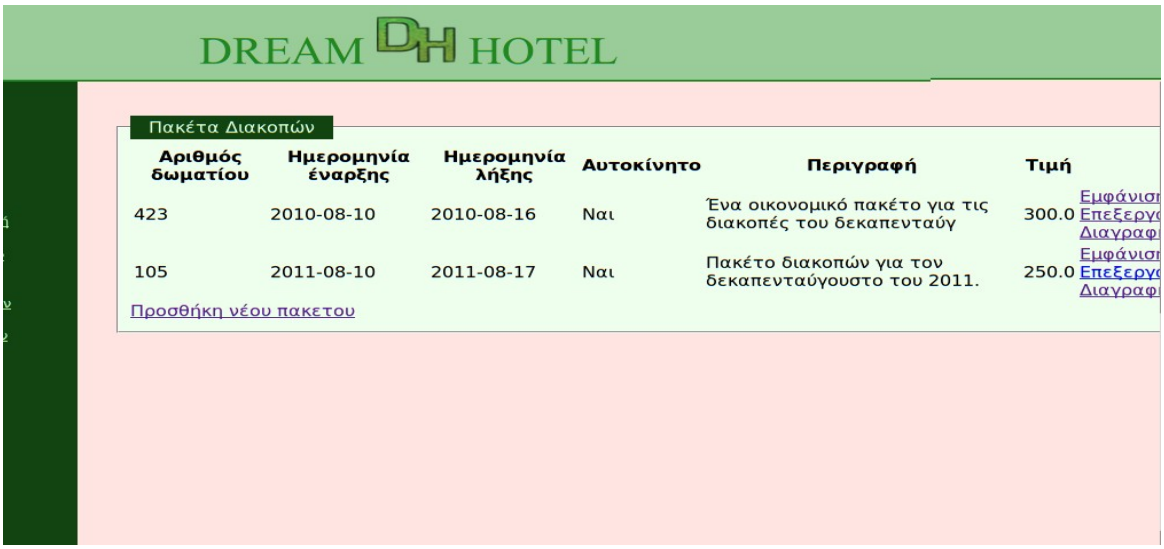
5.2.5 Τα πακέτα διακοπών

Εκτός από τα δωμάτια η εφαρμογή δίνει την δυνατότητα διαχείρισης και προσθήκης νέων πακέτων διακοπών από τον διαχειριστή και κράτηση κάποιου πακέτου από τον χρήστη. Ο διαχειριστής έχει την δυνατότητα να δει τις κρατήσεις των πακέτων διακοπών και διάφορες πληροφορίες σχετικά με την κάθε κράτηση και επιπλέον μπορεί να διαγράψει μια κράτηση από την λίστα με τις κρατήσεις.

5.2.5.1 Η διαχείριση των πακέτων διακοπών

Πατώντας στο link *πακέτα διακοπών* εμφανίζεται η λίστα με όλα τα πακέτα διακοπών.

Εδώ ο διαχειριστής έχει την δυνατότητα να επεξεργαστεί ή να διαγράψει ένα υπάρχον πακέτο ή ακόμη και να προσθέσει και άλλα πακέτα διακοπών.



Εικόνα 5.14 - Η λίστα με τα πακέτα διακοπών

* *

<fieldset>

<legend>Πακέτα Διακοπών</legend>

<table>

<tr>

<th>Αριθμός δωματίου</th>

<th>Ημερομηνία έναρξης</th>

<th>Ημερομηνία λήξης</th>

<th>Αυτοκίνητο</th>

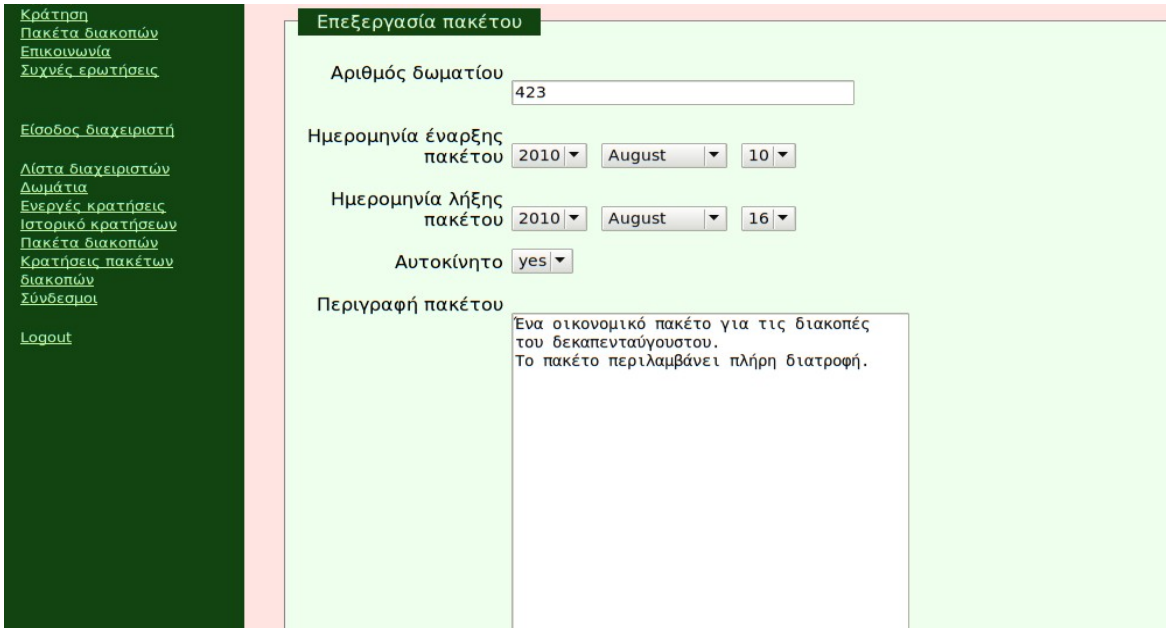
<th>Περιγραφή</th>

```
    <th>Τιμή</th>
  </tr>
<% for holiday_packet in @holiday_packets %>
  <tr>
    <td><%=h holiday_packet.room_number %></td>
    <td><%=h holiday_packet.start_date.to_s[0..10] %></td>
    <td><%=h holiday_packet.end_date.to_s[0..10] %></td>
    <td><%=h holiday_packet.car %></td>
    <td><%=h holiday_packet.description[0..100] %></td>
    <td><%=h holiday_packet.timi %></td>
    <td class="list-actions">
      <%= link_to 'Εμφάνιση', holiday_packet %><br />
      <%= link_to 'Επεξεργασία', edit_holiday_packet_path(holiday_packet) %>
        <%= link_to 'Διαγραφή', holiday_packet, :confirm => 'Είσται σίγουρος?', :method =>
:delete %>
    </td>
  </tr>
<% end %>
</table>
<%= link_to 'Προσθήκη νέου πακετου', new_holiday_packet_path %>
</fieldset>
```

* *

(hotel/app/views/holiday_packets/index.html.erb)

Πατώντας στο link επεξεργασία εμφανίζονται οι πληροφορίες για το εκάστοτε πακέτο, τις οποίες μπορεί ο διαχειριστής να τις αλλάξει.



Εικόνα 5.15 - Επεξεργασία πακέτου διακοπών

Μετά την ολοκλήρωση των αλλαγών πατώντας το κουμπί *επιβεβαίωση* καλείται η μέθοδος:

```
def update
  @holiday_packet = HolidayPacket.find(params[:id])
  respond_to do |format|
    if @holiday_packet.update_attributes(params[:holiday_packet])
      flash[:notice] = 'Το πακέτο διακοπών προστέθηκε με επιτυχία.'
      format.html { redirect_to(@holiday_packet) }
```

```
      format.xml  { head :ok }
    else
      format.html { render :action => "edit" }
      format.xml  { render :xml => @holiday_packet.errors, :status => :unprocessable_entity }
    end
  end
end
```

(hotel/app/controllers/holiday_packets_controller.rb)

που αποθηκεύει τις αλλαγές που έγιναν στο πακέτο διακοπών.

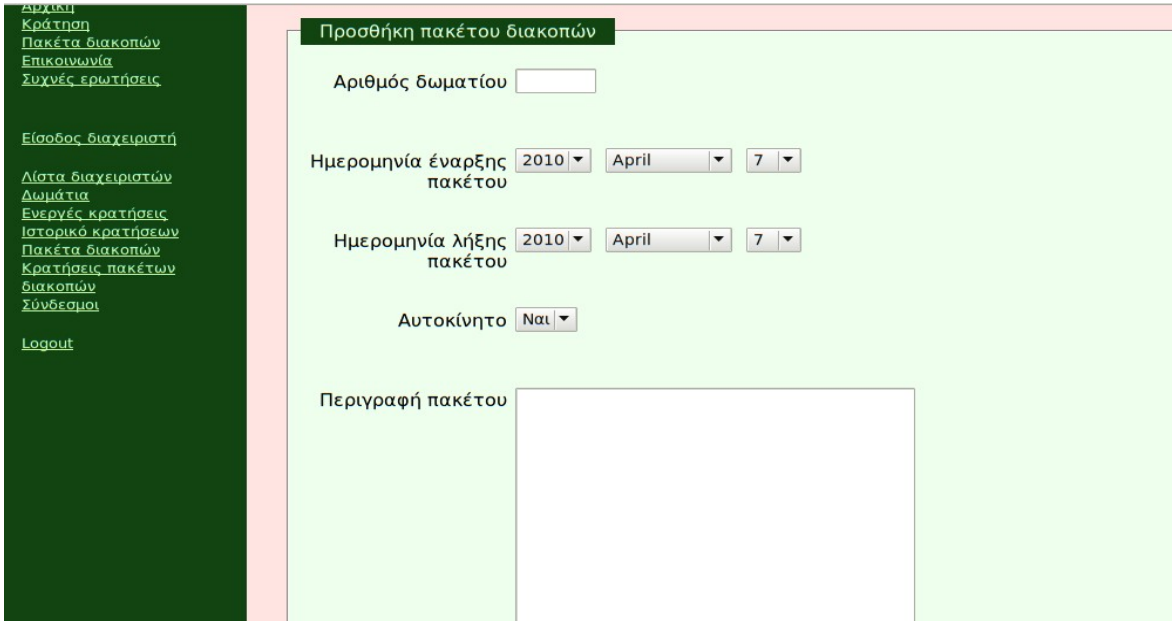
Ενώ πατώντας στο link *διαγραφή* καλείται η μέθοδος:

```
def destroy
  @holiday_packet = HolidayPacket.find(params[:id])
  @holiday_packet.destroy
  respond_to do |format|
    format.html { redirect_to(holiday_packets_url) }
    format.xml  { head :ok }
  end
end
```

(hotel/app/controllers/holiday_packets_controller.rb)

που διαγράφει το πακέτο από την λίστα με τα πακέτα διακοπών.

Ο διαχειριστής μπορεί πολύ εύκολα να προσθέσει ένα νέο πακέτο διακοπών πατώντας στο link *προσθήκη νέου πακέτου διακοπών.*



Εικόνα 5.16 - Προσθήκη νέου πακέτου διακοπών

ο κώδικας που χρειάζεται για να εμφανιστεί η παραπάνω σελίδα είναι:

* *

<fieldset>

*<legend>Π*ιριστής μπορεί πολύ εύκολα να προσθέσει ένα νέο πακέτο διακοπών πατώντας σ*ροσθήκη πακέτου διακοπών</legend>*

<% form_for(@holiday_packet) do |f| %>

<%= f.error_messages %>

<p>

```
    <%= f.label "Αριθμός δωματίου" %>
    <%= f.text_field :room_number, :size => 5 %>
  </p>
<br />
  <p>
    <%= f.label "Ημερομηνία έναρξης πακέτου" %>
    <%= f.date_select :start_date %>
  </p>
<br />
  <p>
    <%= f.label "Ημερομηνία λήξης πακέτου" %>
    <%= f.date_select :end_date %>
  </p>
<br />
  <p>
    <%= f.label "Αυτοκίνητο" %>
    <%= f.select :car, ['Ναι', 'Όχι'] %>
  </p>
<br />
  <p>
```

```
    <%= f.label "Περιγραφή πακέτου" %>
    <%= f.text_area :description, :size => 20 %>
  </p>
<br />
  <p>
    <%= f.label "Τιμή" %>
    <%= f.text_field :timi %>
  </p>
<br />
  <p>
    <%= f.submit "Προσθήκη" %>
  </p>
<% end %>
<%= link_to 'Back', holiday_packets_path %>
</fieldset>
</div>
```

(hotel/app/views/holiday_packets/new.html.erb)

Πατώντας το κουμπί προσθήκη καλείται η μέθοδος:

```
  def create
    @holiday_packet = HolidayPacket.new(params[:holiday_packet])
    respond_to do |format|
      if @holiday_packet.save
        flash[:notice] = 'Το πακέτο διακοπών προστέθηκε με επιτυχία.'
        format.html { redirect_to(@holiday_packet) }
        format.xml  { render :xml => @holiday_packet, :status => :created, :location => @holiday_packet }
      else
        format.html { render :action => "new" }
        format.xml  { render :xml => @holiday_packet.errors, :status => :unprocessable_entity }
      end
    end
  end
```

(hotel/app/controllers/holiday_packets_controller.rb)

που αποθηκεύει το νέο πακέτο διακοπών στην βάση δεδομένων.

5.2.5.2 Οι κρατήσεις των πακέτων διακοπών

Ο διαχειριστής μπορεί να δει μια κράτηση ενός πακέτου διακοπών και διάφορες πληροφορίες που αφορούν αυτό το πακέτο, όπως το όνομα στο όποιο έγινε η κράτηση, τον τρόπο πληρωμής, την τιμή, τον αριθμό τηλεφώνου του προσώπου που έκανε την κράτηση κτλ. Επιπλέον ο διαχειριστής μπορεί εάν θέλει να καταργήσει μια τέτοια κράτηση.

Εικόνα 5.17 - Κρατήσεις πακέτων διακοπών

ο κώδικας που εμφανίζει την λίστα με τις κρατήσεις των πακέτων διακοπών είναι:

* *

<fieldset>

<legend>Κρατήσεις πακέτων</legend>

<% @pbookings = PacketBookings.find(:all) %>

<% if @pbookings.empty? || @pbookings.nil? %>

**

* Δεν υπάρχουν κρατήσεις *

**

<% else %>

```
<font size='2'>

<table border = "1" width = "100">

<tr>

<th>ΟΝΟΜΑΤΕΠΩΝΥΜΟ</th>

<th>ΤΗΛΕΦΩΝΟ</th>

<th>E-MAIL</th>

<th>ΤΡΟΠΟΣ ΠΛΗΡΩΜΗΣ</th>

<th>ΑΡΙΘΜΟΣ ΔΩΜΑΤΙΟΥ</th>

<th>ΛΗΞΗ ΠΑΚΕΤΟΥ</th>

<th>ΛΗΞΗ ΠΑΚΕΤΟΥ</th>

<th>ΤΙΜΗ</th>

<th>ΑΥΤΟΚΙΝΗΤΟ</th>

</tr>

<% for booking in @pbookings -%>

<tr>

<td><%=h booking.fname %> <%=h booking.lname %></td>

<td><%=h booking.phone %></td>

<td><%=h booking.email %></td>

<td><%=h booking.pay_type %> <% unless booking.card_number == nil %> <p> Αριθμός κάρτας: <%= booking.card_number %></p> <% end %> </td>

<% @packets = HolidayPacket.find(:all) %>
```

<% for packet in @packets %>

<% if packet.id == booking.holiday_packet_id %>

<td><%=h packet.room_number %></td>

<td><%=h packet.start_date.to_s[0..10] %></td>

<td><%=h packet.end_date.to_s[0..10] %></td>

<td><%=h packet.timi %></td>

<td><%=h packet.car %></td>

<td><%= button_to 'Κατάργηση', {:action => 'destroy_packet_booking', :id => booking}, :confirm => "Είσαστε σίγουρος ότι θέλετε να καταργήσετε την κράτηση;" %></td>

</tr>

<% end %>

<% end %>

</table>

**

<% end %>

<% end %>

</fieldset>

* *

(hotel/app/views/administrator/show_packet_bookings.html.erb)

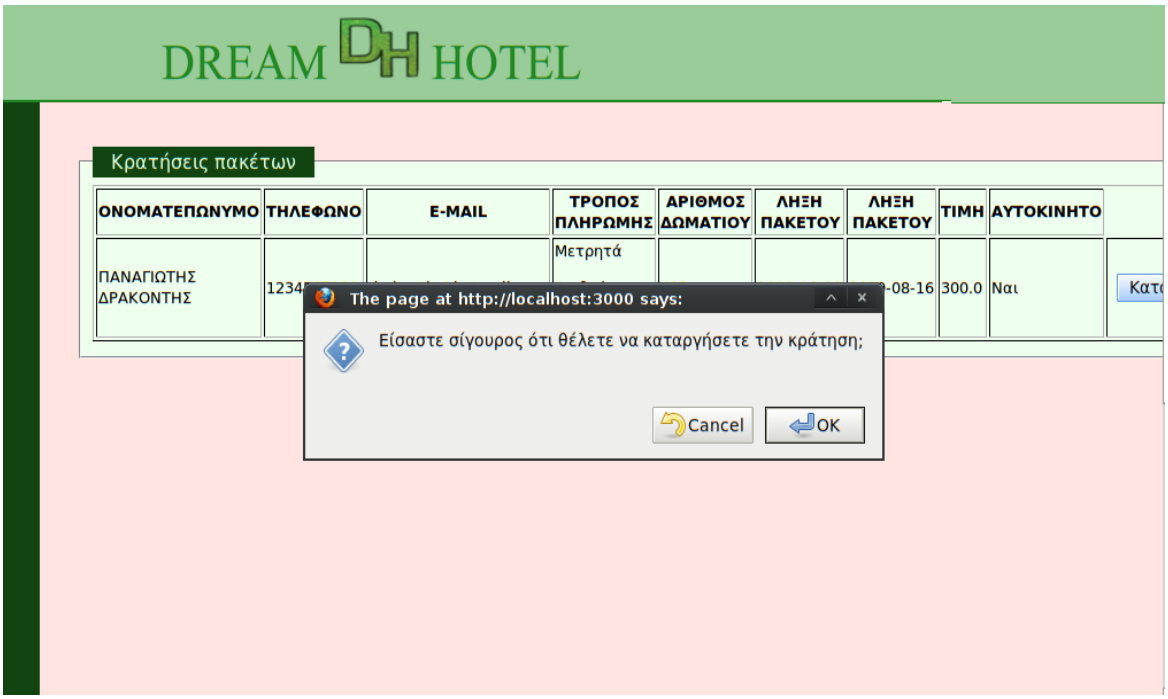
Η κατάργηση μιας κράτησης ενός πακέτου διακοπών γίνεται δυνατή πατώντας στο κουμπί

κατάργηση, τότε καλείται η συνάρτηση:

```
def destroy_packet_booking
    @pbooking = PacketBookings.find(params[:id])
    @pbooking.destroy
    flash[:notice] = "Η κράτηση έχει καταργηθεί."
    redirect_to :action => 'show_packet_bookings'
end
```

(hotel/app/controllers/administrator_controller.html.erb)

που διαγράφει την κράτηση από την βάση δεδομένων και κατά συνέπεια και από την λίστα με τις κρατήσεις των πακέτων διακοπών.



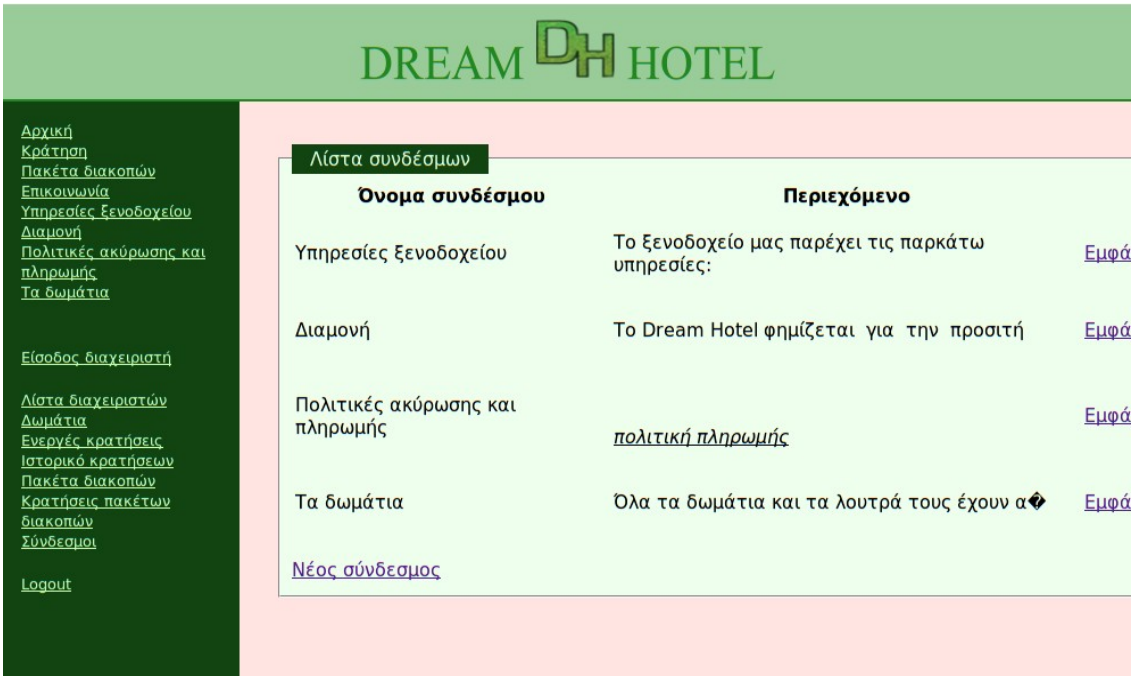
Εικόνα 5.18 - Διαγραφή πακέτου διακοπών

5.2.6 Προσθήκη νέων συνδέσμων

Η εφαρμογή δίνει την δυνατότητα στην διαχειριστή να προσθέτει με δυναμικό τρόπο νέα links στην εφαρμογή, προσβάσιμα από τον χρήστη. Αυτό είναι ένα πολύ δυνατό εργαλείο διότι προσφέρει την δυνατότητα στον διαχειριστή να προσθέσει οτιδήποτε θελήσει στην σελίδα του, χωρίς να χρειαστεί την παραμικρή βοήθεια του προγραμματιστή.

Πατώντας λοιπόν στο link *σύνδεσμοι* η εφαρμογή οδηγεί τον διαχειριστή σε μια σελίδα όπου φαίνεται μια λίστα με όλα τα δυναμικά links με τις ενέργειες που μπορεί να κάνει σε κάθε link (επεξεργασία, διαγραφή κτλ).

Εικόνα 5.19 - Οι λίστα των συνδέσμων



Εικόνα 5.19 - Οι λίστα των συνδέσμων

Ο κώδικας για την παραπάνω σελίδα είναι:

```
def index
   @links = Link.find(:all)
   respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @links }
   end
 end
```

(hotel/app/controller/links_controller.rb)

```
<div class="hotel-form">
<fieldset>
<legend>Λίστα συνδέσμων</legend>
<table>
 <tr>
   <th>Όνομα συνδέσμου</th>
   <th>Περιεχόμενο</th>
 </tr>
<% for link in @links %>
 <tr>
```

<td><%=h link.linkname %></td>

<td><%= link.keimeno[0..100] %></td>

<td><%= link_to 'Εμφάνιση', link %></td>

<td><%= link_to 'Επεξεργασία', edit_link_path(link) %></td>

<td><%= link_to 'Διαγραφή', link, :confirm => 'Είσται σίγουρος ότι θέλετε να διαγράψετε τον σύνδεσμο από την λίστα?', :method => :delete %></td>

</tr>

<% end %>

</table>

*
*

<%= link_to 'Νέος σύνδεσμος', new_link_path %>

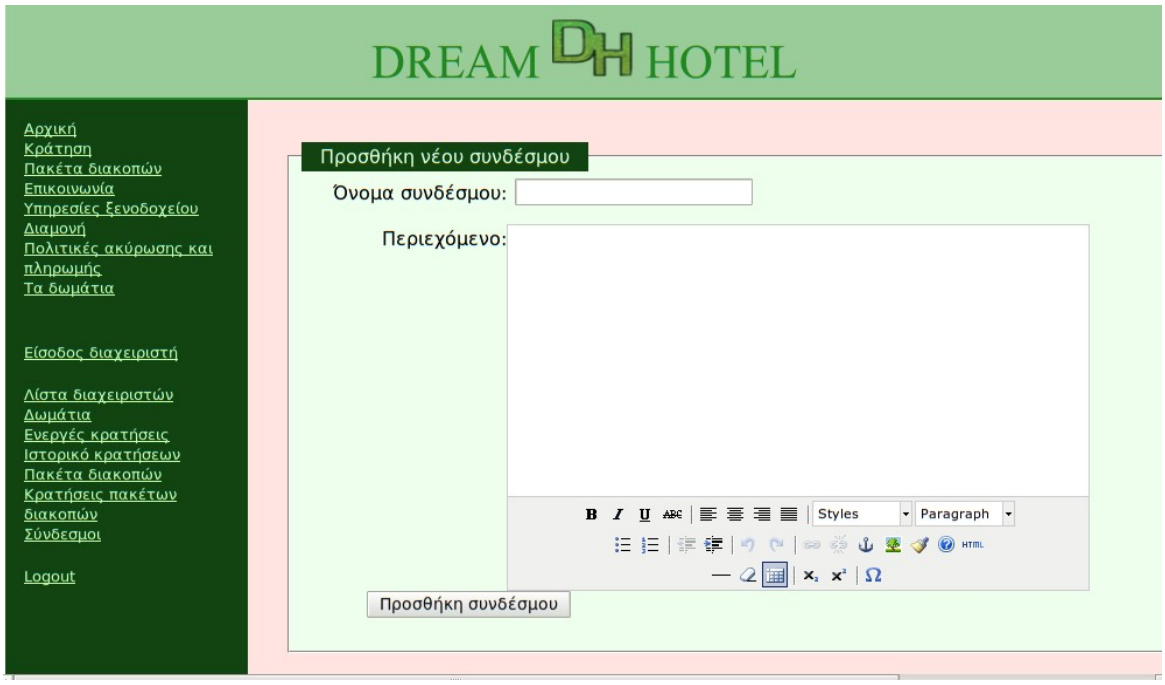
</fieldset>

* *

(hotel/app/views/links/index.html.erb)

Η πρώτη μέθοδος καλείται αυτόματα, επειδή έχει το ίδιο όνομα με την σελίδα (index). Αυτό που κάνει είναι να αντιγράψει όλα τα αντικείμενα τύπου link από την βάση δεδομένων σε έναν πίνακα. Και μετά ο κώδικας που βρίσκεται στο αρχείο (hotel/app/views/links/index.html.erb) εμφανίζει τα περιεχόμενα του πίνακα.

Πατώντας στο link προσθήκη νέου συνδέσμου η εφαρμογή οδηγεί τον διαχειριστή στην παρακάτω σελίδα.



Εικόνα 5.20 - Εισαγωγή νέου συνδέσμου

Ο κώδικας για αυτή την σελίδα είναι:

```
<div class="hotel-form">

<fieldset>

  <legend>Προσθήκη νέου συνδέσμου</legend>

    <% form_tag :action => :add_link do %>

        <label for="linkname">Όνομα συνδέσμου:</label>

        <%= text_field_tag :linkname %>

        <p />

        <label for="keimeno">Περιεχόμενο:</label>

        <% use_tinymce -%>
```

<%= text_area_tag "keimeno" %>

<%= submit_tag "Προσθήκη συνδέσμου", :class => "submit" %>

<% end %>

</fieldset>

* *

(hotel/app/views/links/new.html.erb)

Αν παρατηρήσουμε στο κομμάτι του κώδικα φαίνεται μια γραμμή με διαφορετικό χρώμα.

<% use_tinymce -%>

σε αυτή την γραμμή καλείται ο text editor που βλέπουμε στην σελίδα για την προσθήκη νέου συνδέσμου.

Ο text editor είναι μια javascript εφαρμογή που ονομάζεται tinymce. Η πλήρης ονομασία του είναι **Tiny Moxiecode Content Editor** και έχει εκδοθεί κάτω από την open source άδεια LGPL από την Moxiecode Systems AB. Έχει την δυνατότητα να μετατρέπει κώδικα HTML σε απλό κείμενο, επίσης δίνει την δυνατότητα εισαγωγής εικόνων και συνδέσμων (links) στο κείμενο. Είναι συμβατός σχεδόν με όλους τους Web browsers. Το αρχείο ρυθμίσεων για τον tinymce editor είναι το hotel/public/javascripts/mce_editor.js.

Πατώντας το κουμπί *Προσθήκη συνδέσμου* καλείται η μέθοδος:

def create

@link = Link.new(params[:link])

respond_to do |format|

```
    if @link.save
      flash[:notice] = 'Link was successfully created.'
      format.html { redirect_to(@link) }
      format.xml  { render :xml => @link, :status => :created, :location => @link }
    else
      format.html { render :action => "new" }
      format.xml  { render :xml => @link.errors, :status => :unprocessable_entity }
    end
  end
end
```

(hotel/app/controllers/links_controller.rb)

που αποθηκεύει τον σύνδεσμο και το περιεχόμενο του στην βάση δεδομένων.

Πατώντας στα links *επεξεργασία* και *διαγραφή,* ο διαχειριστής μπορεί να επεξεργαστεί και να διαγράψει ένα link. Ο κώδικας για αυτές τις λειτουργίες είναι:

```
def update
  @link = Link.find(params[:id])
  respond_to do |format|
    if @link.update_attributes(params[:link])
      flash[:notice] = 'Ο σύνδεσμος ανανεώθηκε με επιτυχία !'
      format.html { redirect_to(@link) }
      format.xml  { head :ok }
```

```
    else
      format.html { render :action => "edit" }
      format.xml  { render :xml => @link.errors, :status => :unprocessable_entity }
    end
  end
end
```

(hotel/app/controllers/links_controller.rb)

```
def destroy
  @link = Link.find(params[:id])
  @link.destroy
  respond_to do |format|
    format.html { redirect_to(links_url) }
    format.xml  { head :ok }
  end
end
```

(hotel/app/controllers/links_controller.rb)

5.2.7 Δικαιώματα πρόσβασης μόνο σε διαχειριστή

Μπορεί μεν τα links που αφορούν τον διαχειριστή να μην εμφανίζονται στον χρήστη. Ο χρήστης όμως βρίσκοντας την ονομασία του link και γράφοντάς την στον browser θα μπορέσει να

έχει πρόσβαση στο πεδίο που αφορά τον διαχειριστή. Για να προφυλαχθούν από αδιάκριτα μάτια πληροφορίες που είναι μόνο για τον διαχειριστή χρησιμοποιήθηκαν κάποια φίλτρα που παρέχει το rails. Η χρησιμοποίηση του φίλτρου φαίνεται στο παρακάτω τμήμα κώδικα.

```
before_filter :authorize, :except => :login

protected

  def authorize

    unless Admin.find_by_id(session[:admin_id])

      flash[:notice] = "ΠΑΡΑΚΑΛΩ ΚΑΝΤΕ LOGIN ΓΙΑ ΝΑ ΣΥΝΕΧΙΣΕΤΕ"

      redirect_to :controller => :administrator, :action => :login

    end

  end
```

(hotel/app/controllers/application.rb)

Αυτό το τμήμα κώδικα δίνει πρόσβαση στην εφαρμογή μόνο στον διαχειριστή, ακόμη και σε τμήματα που θέλουμε να έχει προσβαση οχρήστης, κάτι που δεν είναι επιθυμητό. Για να δοθεί πρόσβαση σε ορισμένα μόνο τμήματα της εφαρμογής στον χρήστη, πρέπει με κάποιον τρόπο να παρακαμφθούν τα φίλτρα. Η παράκαμψη των φίλτρων είναι επιθυμητή μόνο στον controller freerooms και σε μερικές μεθόδους του controller links. Η παράκαμψη των φίλτρων στους δύο controllers φαίνεται παρακάτω.

Παράκαμψη φίλτρων για τον controller freerooms:

```
  before_filter :find_cart, :except => :empty_cart

protected

  def authorize
```

```
  end
```

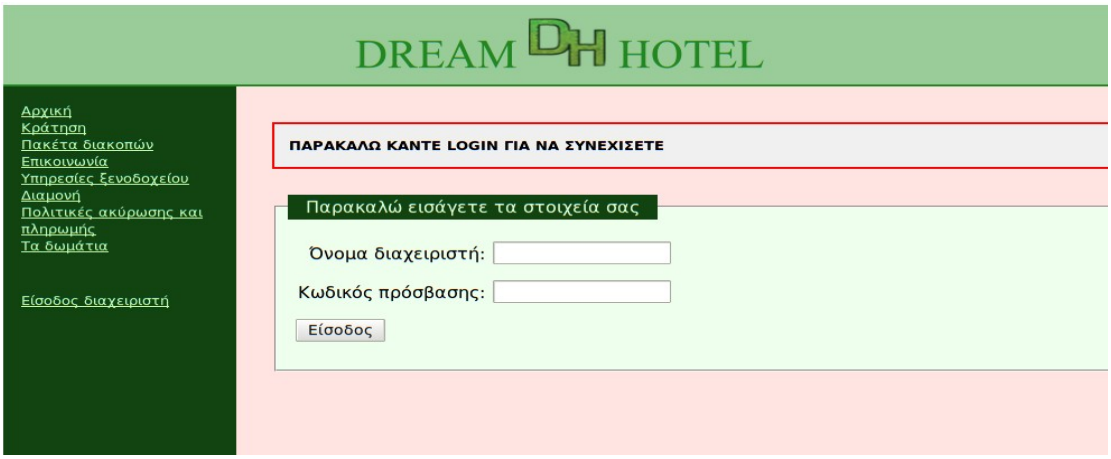
(hotel/app/controllers/freerooms_controller.rb)

Παράκαμψη φίλτρων για τον controller links:

```
  before_filter :authorize, :except => :show

protected

    def authorize

      unless Admin.find_by_id(session[:admin_id])

        flash[:notice] = "ΠΑΡΑΚΑΛΩ ΚΑΝΤΕ LOGIN ΓΙΑ ΝΑ ΣΥΝΕΧΙΣΕΤΕ"

        redirect_to :controller => :administrator, :action => :login

      end

    end
```

(hotel/app/controllers/links_controller.rb)

Έτσι με αυτόν τον τρόπο δίνεται περιορισμένη πρόσβαση στην εφαρμογή για τον χρήστη. Αν ο χρήστης επιχειρήσει να μπει σε μια σελίδα που έχει πρόσβαση μόνο ο διαχειριστής η εφαρμογή θα τον οδηγήσει στην σελίδα για να κάνει login.



Εικόνα 5.21 - Περιορισμένη πρόσβαση

5.3 Οι λειτουργίες για τον χρήστη

Μια εφαρμογή για να θεωρηθεί πετυχημένη θα πρέπει να είναι όσο το δυνατόν περισσότερο φιλική και εύκολη στην χρήση, ειδικότερα στο τμήμα με τις λειτουργίες που αφορούν τον χρήστη – πελάτη.

Ο χρήστης μόλις επισκεφτεί την εφαρμογή για τις on-line κρατήσεις δωματίων, θα βρεθεί στη αρχική σελίδα της εφαρμογής.

Εικόνα 5.22 - Η αρχική σελίδα της εφαρμογής

Ο κώδικας που παράγει την παραπάνω σελίδα είναι:

<h1>Καλώς ήρθατε στο Dream Hotel </h1>

*
*

<%= image_tag("hotel3.jpg") %>

(hotel/app/views/freerooms/index.html.erb)

Η εικόνα βρίσκεται αποθηκευμένη στον φάκελο: hotel/public/images

5.3.1 Οι κρατήσεις δωματίων – πακέτων διακοπών

Η κύρια λειτουργία της εφαρμογής είναι η on-line αναζήτηση και κράτηση ελεύθερων δωματίων. Ο χρήστης έχει αυτή την δυνατότητα πατώντας στο link *Κράτηση*. Μόλις το κάνει αυτό οδηγείται σε μια σελίδα όπου πρέπει να ορίσει την ημερομηνία έναρξης και λήξης της κράτησης ώστε να βρει η εφαρμογή τα ελεύθερα δωμάτια για εκείνη την περίοδο. Πριν ξεκινήσει η αναζήτηση των ελεύθερων δωματίων, η εφαρμογή ελέγχει για το εάν οι ημερομηνίες που έχουν δοθεί είναι έγκυρες. Για να είναι έγκυρες οι ημερομηνίες θα πρέπει η ημερομηνία έναρξης να είναι μεγαλύτερη από την σημερινή και οι ημερομηνία έναρξης να μην είναι μεγαλύτερη από την ημερομηνία λήξης.

Εικόνα 5.23 - Αναζήτηση ελεύθερων δωματίων

Ο κώδικας που εμφανίζει την παραπάνω σελίδα είναι:

```
<% form_tag :action => :find_rooms do %>

    <fieldset style="BACKGROUND-COLOR: #efe">

    <legend>  <font color= "#dfd" style="BACKGROUND-COLOR: #141">  Παρακαλώ δώστε την ημερομηνία έναρξης και λήξης που επιθυμείτε για την κράτηση </font></legend>

    <p>Ημερομηνία έναρξης</p>

    <p>
```

<label for="start_day">Ημέρα:</label>

<%= select_tag :start_day, options_for_select ([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]) %>

<label for="start_month">Μήνας:</label>

<%= select_tag :start_month, options_for_select([["jan", 1],["feb",2],["mar",3],["apr",4], ["may",5],["jun",6],["jul",7],["aug",8],["sep",9],["oct",10],["nov",11],["dec",12]]) %>

<label for="start_year">Έτος:</label>

<%= select_tag :start_year, options_for_select ([2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020]) %>

* *

*
*

* Ημερομηνία λήξης *

* *

<label for="end_day">Ημέρα:</label>

<%= select_tag :end_day, options_for_select ([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]) %>

<label for="end_month">Μήνας:</label>

<%= select_tag :end_month, options_for_select([["jan", 1],["feb",2],["mar",3],["apr",4], ["may",5],["jun",6],["jul",7],["aug",8],["sep",9],["oct",10],["nov",11],["dec",12]]) %>

<label for="end_year">Έτος:</label>

<%= select_tag :end_year, options_for_select ([2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020]) %>

```
    </p>

    <br />

    <%= submit_tag "Εύρεση", :class => "submit" %>

    </fieldset>

  <% end %>
```

(hotel/app/views/freerooms/searchrooms.html.erb)

αφού δοθούν οι έγκυρες ημερομηνίες ξεκινάει η αναζήτηση των ελεύθερων δωματίων. Η μέθοδος που καλείται να κάνει αυτήν την δουλεία είναι:

```
  def find_rooms

    session[:cart] = nil

    start_day = params[:start_day]

    start_month = params[:start_month]

    start_year = params[:start_year]

    end_day = params[:end_day]

    end_month = params[:end_month]

    end_year = params[:end_year]

    $startdate = Date.new(start_year.to_i,start_month.to_i,start_day.to_i)

    $enddate = Date.new(end_year.to_i,end_month.to_i,end_day.to_i)

    $days = ($enddate-$startdate).to_i
```

```
    if start_day.nil? || start_day.empty? || start_month.nil? || start_month.empty? || start_year.nil?
|| start_year.empty? || end_day.nil? ||  end_day.empty? || end_month.nil? || end_month.empty? ||
end_year.nil? || end_year.empty?

        flash[:notice] = "Εχετε δώσει λάθος ημερομηνίες. Παρακαλώ εισάγετε πάλι τις ημερομηνίες
πιο προσεκτικά."

        redirect_to :action => :searchrooms

    else

    @rooms = Room.find_freerooms ($startdate, $enddate)

    if @rooms == "false"

       flash[:notice] = "Εχετε δώσει λάθος ημερομηνίες. Παρακαλώ εισάγετε πάλι τις ημερομηνίες
πιο προσεκτικά."

      redirect_to :action => :searchrooms

    end

    @cart = find_cart

   end

  end
```

(hotel/app/controllers/freerooms_controller.rb)

Στην γραμμή με το γαλάζιο φόντο καλείται η συνάρτηση που βρίσκει και αποθηκεύει σε έναν πίνακα τα ελεύθερα δωμάτια για εκείνη την περίοδο. Η συνάρτηση που το κάνει αυτό είναι:

```
def self.find_freerooms(startdate, enddate)
```

```
@packets = HolidayPacket.find(:all)
$nowdate = Date.new(Time.now.year.to_i,Time.now.month.to_i,Time.now.day.to_i)
if $startdate < $nowdate || $enddate <= $nowdate || $enddate <= $startdate
  "false"
else
  @bookings = Booking.find(:all)
  flag = 0
  @books = Array.new
  @books = []
  @bookings.each { |book|
                                                    startbooking =
Date.new(book.start_booking.year,book.start_booking.month,book.start_booking.day)
                                                    endbooking =
Date.new(book.end_booking.year,book.end_booking.month,book.end_booking.day)
          if (($startdate >= startbooking && $startdate < endbooking) || ($enddate >
startbooking && $enddate <= endbooking) || ($startdate <= startbooking && $enddate >=
endbooking))          -0.25
    flag = 1
  end
  }
  if flag == 0
    @rooms = Room.find(:all, :order => 'tipos')
```

```
  @rooms
else
 @reservedrooms = Array.new
 @bookings.each {|booked|
   @reserved_rooms = ReservedRoom.find(:all)
   @reserved_rooms.each { |reservedroom|
   if reservedroom.booking_id == booked.id
     @reservedrooms << reservedroom
   end
   }
}
@freerooms = Array.new
@freerooms = []
@rooms = Room.find(:all)
@rooms.each { |room|
flag2 = 0
  @reserved_rooms.each { |reservedroom|
    if reservedroom.room_id == room.id
      flag2 = 1
    end
```

```
        }
        if flag2 == 0
          @packets.each { |packet|
                  startpacket = Date.new(packet.start_date.year, packet.start_date.month, packet.start_date.day)
                  endpacket = Date.new(packet.end_date.year, packet.end_date.month, packet.end_date.day)
                unless ((($startdate >= startpacket && $startdate < endpacket) || ($enddate > startpacket && $enddate <= endpacket) || ($startdate <= startpacket && $enddate >= endpacket)) && (room.roomnumber == packet.room_number))
            @freerooms << room
            end
            }
          end
        }
       @freerooms
      end
    end
  end
```

και έπειτα η εφαρμογή εμφανίζει τα περιεχόμενα του πίνακα.

ο κώδικας που εμφανίζει τα δωμάτια είναι:

```
<div class="hotel-form">
<fieldset>
<legend>Τα ελεύθερα δωμάτιά μας</legend>
<% for room in @rooms -%>
  <p> <font size = '4'><b>Τύπος δωματίου: </b><%=h room.tipos %>       <b>Τιμή: </b><%= room.timi %> </font> </p>
  <p>
  <% form_remote_tag :url => { :action => 'add_to_cart', :id => room } do %>
    <%= submit_tag "Προσθήκη στην κάρτα" %>
  <% end %>
  </p>
 <h1> </h1>
<% end %>
</fieldset>
</div>
```

(hotel/app/views/freerooms/find_rooms.html.erb)

ο χρήστης τώρα μπορεί να επιλέξει δωμάτια και να τα βάλει στην κάρτα κρατήσεων του.

Εικόνα 5.24 - Ελεύθερα δωμάτια και κάρτα κρατήσεων

Σε αυτό το σημείο η εφαρμογή χρησιμοποιεί την τεχνολογία javascript ώστε να μην ανανεώνεται όλη η σελίδα κάθε φορά που προστίθεται κάτι στην κάρτα, αλλά ένα μόνο μέρος της. Ο κώδικας για την προσθήκη δωματίων στην κάρτα είναι:

```
class Cart

  attr_reader :items

  def total_price

    total_price = @items.sum { |item| item.timi * $days }

  end

  def initialize

    @items = []
```

```
  end
  def booking_room(room)
    @items << room
  end
  def getitems
    @items
  end
end
```

(hotel/app/models/cart.rb)

```
  def find_cart
   unless session[:cart]
     session[:cart] = Cart.new
   end
   session[:cart]
  end
```

(hotel/app/controllers/freerooms_controller.rb)

```
  def add_to_cart
    begin
```

```
    room = Room.find(params[:id])
  rescue ActiveRecord::RecordNotFound
    logger.error("ΜΗ ΕΓΚΥΡΟ ΔΩΜΑΤΙΟ #{params[:id]}")
    flash[:notice] = "Μη έγκυρο δωμάτιο"
    redirect_to_index("Μη έγκυρο δωμάτιο")
  else
  @cart = find_cart
  @cart.booking_room(room)
  respond_to { |format| format.js }
  end
 end
```

(hotel/app/controllers/freerooms_controller.rb)

```
<div class="cart-title">Η κάρτα κρατήσεων μου</div>
<table>
 <%= render(:partial => "cart_item", :collection => cart.items) %>
 <tr class="total-line">
  <td colspan="2">Σύνολο</td>
  <td class="total-cell"><%= number_to_currency(cart.total_price) %></td>
 </tr>
```

</table>

<%= button_to "Κράτηση", :action => 'checkout' %>

<%= button_to "Άδειασμα κάρτας", :action => :empty_cart %>

(hotel/app/views/freerooms/_cart.html.erb)

<tr>

* <td><%=h cart_item.roomnumber %></td>*

* <td class="item-price"><%= number_to_currency(cart_item.timi) %></td>*

</tr>

(hotel/app/views/freerooms/_cart_item.html.erb)

page.select("div#notice").each { |div| div.hide }

page.replace_html("cart", :partial => "cart", :object => @cart)

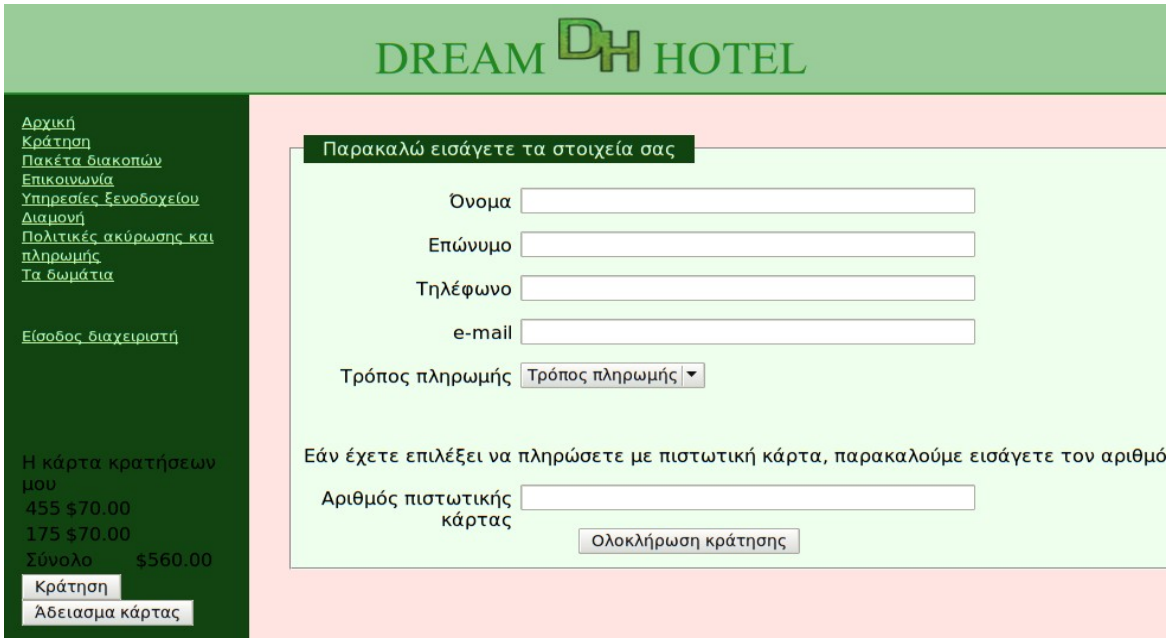
(hotel/app/views/freerooms/add_to_cart.js.rjs)

page.replace_html("cart", :partial => "cart", :object => @cart)

(hotel/app/views/freerooms/empty_cart.js.rjs)

Όταν επιλεχθεί ένα δωμάτιο να μπει στην κάρτα, δημιουργείται ένα νέο αντικείμενο της κλάσης cart, και ξεκινάει ένα νέο session που παίρνει σαν παράμετρο το id του νέου αντικειμένου cart που μόλις δημιουργήθηκε.

Πατώντας στο κουμπί της κάρτας *κράτηση,* η εφαρμογή οδηγεί τον χρήστη σε μια φόρμα, ώστε να συμπληρώσει τα στοιχεία του και τον τρόπο πληρωμής, και έπειτα πατώντας το κουμπί *ολοκλήρωση κράτησης,* ολοκληρώνεται η κράτηση και τα στοιχεία της κράτησης και του πελάτη αποθηκεύονται στην βάση δεδομένων.



Εικόνα 5.25 - Εισαγωγή στοιχείων του πελάτη, για ολοκλήρωση της κράτησης

Εάν ο χρήστης επιλέξει να πληρώσει μέσω paypal, μετά την ολοκλήρωση της κράτησης, οδηγείτε αυτόματα στην αρχική σελίδα του paypal. Εάν ο χρήστης επιλέξει τελικά να πληρώσει με πιστωτική κάρτα, πρέπει να συμπληρώσει τον αριθμό της κάρτας του και αφού γίνει έλεγχος για το εάν ο αριθμός της κάρτας είναι έγκυρος, ολοκληρ'ωνεται η κράτηση. Για τον έλεγχο της πιστωτικής κάρτας χρησιμοποιήθηκε το gem creditcard.

Ο κώδικας που εμφανίζει την φόρμα και ολοκληρώνει την κράτηση είναι:

```
<div class="hotel-form">

  <%= error_messages_for 'booking' %>

  <% form_for :booking, :url => { :action => :save_booking } do |form| %>
```

```
 <fieldset>
  <legend>Παρακαλώ εισάγετε τα στοιχεία σας</legend>
<p>
  <%= label :booking, :fname, "Όνομα" %>
  <%= form.text_field :fname, :size => 40 %>
</p>
<p>
  <%= label :booking, :lname, "Επώνυμο" %>
  <%= form.text_field :lname, :size => 40 %>
</p>
<p>
  <%= label :booking, :phone, "Τηλέφωνο" %>
  <%= form.text_field :phone, :size => 40 %>
</p>
<p>
  <%= label :booking, :email, "e-mail" %>
  <%= form.text_field :email, :size => 40 %>
</p>
<p>
  <%= label :booking, :pay_type, "Τρόπος πληρωμής" %>
```

```
        <%=
         form.select :pay_type,
                     Booking::PAYMENT_TYPES,
                    :prompt => "Τρόπος πληρωμής"
        %>
      </p>
      <br />
       <p> Εάν έχετε επιλέξει να πληρώσετε με πιστωτική κάρτα, παρακαλούμε εισάγετε τον αριθμό της κάρτας </p>
      <p>
        <%= label :booking, :card_number, "Αριθμός πιστωτικής κάρτας" %>
        <%= form.text_field :card_number, :size => 40 %>
      </p>
      <%= submit_tag "Ολοκλήρωση κράτησης", :class => "submit" %>
    </fieldset>
  <% end %>
</div>
```

(hotel/app/views/freerooms/checkout.html.erb)

```
  def checkout
    @cart = find_cart
```

```
  if @cart.items.empty?
    redirect_to_index("Δεν έχετε προσθέσει δωμάτια προς κράτηση στο καλάθι σας.")
  else
    $booking = Booking.new
  end
 end
```

(hotel/app/controllers/freerooms_controller.rb)

```
def save_booking
    @cart = find_cart
    @booking = Booking.new(params[:booking])
    @booking.start_booking = $startdate
    @booking.end_booking = $enddate
    @booking.add_reserved_rooms_from_cart(@cart)
    @card = @booking.card_number
    @booking.card_number = nil
    if @booking.pay_type == "Πιστωτική κάρτα"
        if @card.creditcard?
            @booking.card_number = @card
            if @booking.save
```

```
        session[:cart] = nil
      flash[:notice] = "Η κράτηση ολοκληρώθηκε. Σας περιμένουμε!"
      redirect_to_index
        else
        render :action => :checkout
      end
    else
      flash[:notice] = "Ο αριθμός της πιστωτικής κάρτας δεν είναι εγκυρος."
      redirect_to :action => :checkout
    end
  else
   if @booking.save
    if @booking.pay_type == "Paypal"
      redirect_to http://www.paypal.com
    else
     session[:cart] = nil
       flash[:notice] = "Η κράτηση ολοκληρώθηκε. Σας περιμένουμε!"
     redirect_to_index
    end
   else
```

```
      render :action => :checkout
    end
    end
  end
```

(hotel/app/controllers/freerooms_controller.rb)

Η κράτηση του πακέτου διακοπών είναι ίδια με την κράτηση δωματίων με την μόνη διαφορά πως εδώ παραπέμπεται η προσθήκη του πακέτου στο καλάθι αγορών.

5.3.2 Δυνατότητα επικοινωνίας

Ο χρήστης εκτός από τηλεφωνικά και ταχυδρομικά, μπορεί να επικοινωνήσει με τον ιδιοκτήτη – διαχειριστή της εφαρμογής στέλνοντας e-mail μέσα από την ίδια την εφαρμογή, χωρίς να χρειαστεί να έχει δικιά του διεύθυνση e-mail. Πολύ απλά το mail θα στέλνεται από την διεύθυνση dreamhotelsend@gmail.com στην διεύθυνση dreamhotelreceive@gmail.com. Και οι δύο διευθύνσεις ανήκουν στον διαχειριστή, με αυτόν τον τρόπο ο πελάτης δεν έχει καμία σχέση όσον αφορά την διεύθυνση που θα αποσταλεί mail, ή την διεύθυνση από την οποία θα σταλεί το mail.

Για την ανάπτυξη του mailler χρησιμοποιήθηκαν οι μέθοδοι της κλάσης ActionMailler. Ο ActionMailler δίνει την δυνατότητα στον προγραμματιστή να αναπτύξει μια φόρμα επικοινωνίας μέσω e-mail, χωρίς να χρειαστεί να γράψει πολλές γραμμές κώδικα.

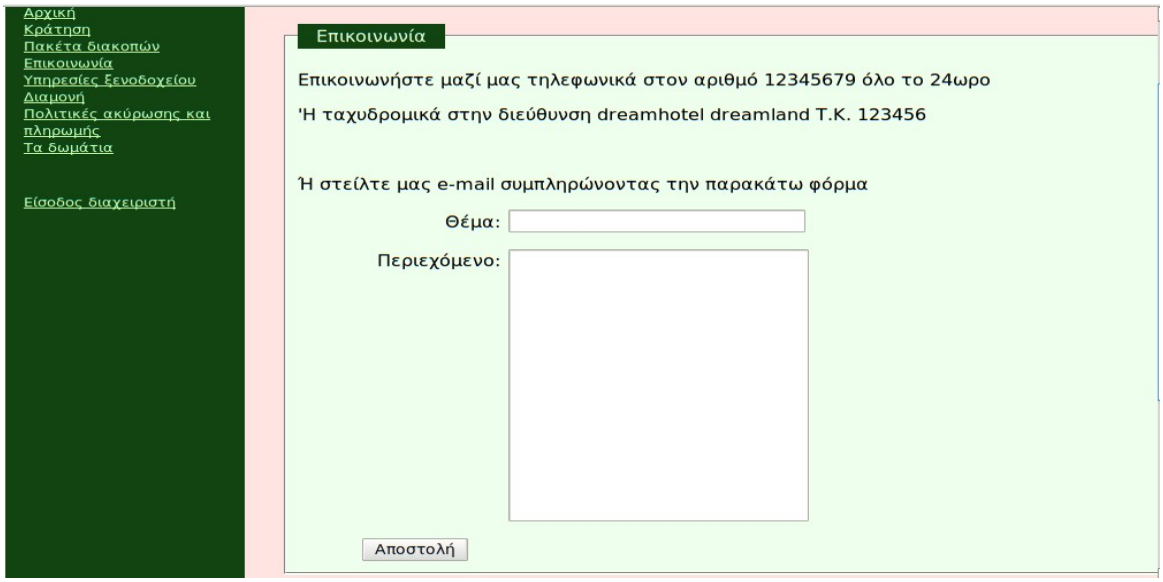
Ξεκινώντας ο προγραμματιστής αναπτύσσει ένα μοντέλο που θα χρησιμοποιηθεί από τον mailler. Αυτό γίνετε δίνοντας την εντολή

script/generate mailer Contact

Για να δουλέψει όμως σωστά ο mailler θα πρέπει προσθέσουμε και μερικές ρυθμίσεις στο τέλος του αρχείου hotel/config/environment.rb. Οι ρυθμίσεις αυτές είναι:

```
config.action_mailer.delivery_method = :smtp

config.action_mailer.smtp_settings = {

  :enable_starttls_auto => true,

  :address => 'smtp.gmail.com',

  :port => 587,

  :authentication => :plain,

  :domain => 'localhost',

  :user_name => 'dreamhotelsend@gmail.com',

  :password => 'dream123hotel'

}
```

Εδώ ορίζουμε μέσω ποιου mail server θα στέλνονται τα mail, το username και το password ώστε να γίνει το authentication στον mail server και διάφορες άλλες ρυθμίσεις που αφορούν την σύνδεση με τον mail server.



Εικόνα 5.26 - Σελίδα επικοινωνίας

Αφού ο χρήστης συμπληρώσει την φόρμα και πατήσει το κουμπί *αποστολή* καλείται η συνάρτηση:

```
def sendmail
   message = params[:message]
   subj = params[:subj]
   Contact.deliver_content (message, subj)
   flash[:notice] = 'Το μήνυμά σας είχε αποσταλεί.'
   redirect_to :action => :index
 end
```

(hotel/app/controllers/contact_controller.rb)

Η εντολή Contact.deliver_content (message, subj) που έχει και διαφορετικό φόντο, αντλεί πληροφορίες για την αποστολή του mail, όπως το θέμα του μηνύματος, τον αποστολέα (που σε αυτή την εφαρμογή είναι η διεύθυνση dreamhotelreceive@gmail.com) κτλ, καλώντας μια συνάρτηση το όνομα της οποίας ακολουθεί την εντολή Contact.deliver που στην προκειμένη περίπτωση είναι content, και ως περιεχόμενο του mail στέλνει το περιεχόμενο του αρχείου hotel/app/views/contact/content.text.plain.erb.

```
 def content (message, subj)
   @message = message
   subject    subj
```

```
    recipients  'dreamhotelreceive@gmail.com'
    from        'dream hotel'
    sent_on     Time.now
  end
```

το περιεχόμενο του αρχείου hotel/app/views/contact/content.text.plain.erb. είναι πολύ απλά το:

```
<%= @message %>
```

που έρχεται σαν παράμετρος από την φόρμα που συμπληρώνει ο χρήστης.

ΚΕΦΑΛΑΙΟ 6ο: ΣΥΜΠΕΡΑΣΜΑΤΑ

Τελειώνοντας και το τελευταίο κομμάτι τις πτυχιακής έπειτα από 7 μήνες ενασχόλησης, είμαι σε θέση να πω πως έχω αποκτήσει αρκετά μεγάλη πείρα στην ανάπτυξη μιας WEB εφαρμογής. Αυτό ήταν αποτέλεσμα των σίγουρων, σταθερών και όχι επιπόλαιων βημάτων από την στιγμή που άρχισα να δουλεύω πάνω στην πτυχιακή.

Μην έχοντας άλλη επαφή με την γλώσσα, παρά μόνο από την στιγμή που ξεκίνησε η ενασχόλησή μου με την πτυχιακή, έπρεπε να αφιερώσω αρκετό χρόνο στην σωστή εκμάθηση της γλώσσας Ruby, ώστε να αποφευχθούν συντακτικά λάθη και λάθη που αφορούσαν την δομή της γλώσσας. Έπειτα από τρεις μήνες διαβάσματος βιβλίων που αφορούσαν την Ruby και νιώθοντας ότι έχω αποκτήσει δυνατές βάσεις, ξεκίνησε η ανάπτυξη της εφαρμογής, που είχε διάρκεια περίπου τρεις μήνες και έχοντας έτοιμη λοιπόν την εφαρμογή, ξεκίνησε η ανάπτυξη του θεωρητικού μέρους της πτυχιακής, το οποίο κράτησε περίπου ένα μήνα.

Δεν μπορώ να μην αναφερθώ στην πολύτιμη βοήθεια του επιβλέποντα καθηγητή, που με τις παρατηρήσεις του, μου έδειξε πως θα μπορεί να δομηθεί μια WEB εφαρμογή.

Μέτα την περάτωση της πτυχιακής και έχοντας αποκτήσει ικανοποιητική πείρα μπορώ να πω, πως η Ruby και το Web framework Ruby On Rails είναι δύο πολύ δυνατά εργαλεία για τον οποιονδήποτε θέλει να ασχοληθεί με το Web programming. Μεγάλο μέρος της δύναμης αυτής αντλείται από την μεγάλη κοινότητα των προγραμματιστών Ruby, που έχουν αναπτύξει έναν τεράστιο αριθμό από gems (βιβλιοθήκες) και της open source φιλοσοφίας που έχει ενστερνιστεί η γλώσσα και κατά συνέπεια και η κοινότητα των προγραμματιστών που την χρησιμοποιεί. Το μεγαλύτερο όμως μέρος της δύναμης της Ruby (και κατά συνέπεια και του RoR) προέρχεται από τον ίδιο της τον εαυτό, μιας και είναι μια πολύ ευέλικτη high level γλώσσα, που δίνει την δυνατότητα στον προγραμματιστή να γράψει κώδικα όχι με βάση την γλώσσα αλλά με βάση τον τρόπο που ο ίδιος θα επιλέξει. Το framework ruby on rails αν και στην αρχή κάποιον ίσως να μην τον ενθουσιάσει ιδιαίτερα, όταν όμως εξοικειωθεί μαζί του θα διαπιστώσει ότι μπορεί να αναπτύξει πολύ “δυνατές” εφαρμογές με πολύ λίγο κόπο.

Τα πλεονεκτήματα που μπορούν να αναφερθούν για την Ruby και το Ruby On Rails επιγραμματικά είναι:

- Ευέλικτη και σωστά δομημένη γλώσσα προγραμματισμού.
- Εύκολη στη εκμάθηση.
- Μεγάλος αριθμός έτοιμων βιβλιοθηκών (gems).
- Open source φιλοσοφία.
- Εύκολη σύνδεση – επικοινωνία με την βάση δεδομένων.
- Μιας και είναι open source δεν χρειάζεται να καταβάλουμε κάποιο χρηματικό κόστος.
- Είναι cross platform, δηλαδή δουλεύει σε όλα τα λειτουργικά συστήματα.

Τα μειονεκτήματα αν και είναι λίγα αξίζει να αναφερθούν:

- θεωρείται "βαριά" γλώσσα, όπως οι περισσότερες high level γλώσσες.
- Άργησε να γίνει ευρέος χρησιμοποιούμενη κάτι που έχει ως αποτέλεσμα να θεωρείται καινούργια γλώσσα, κάτι που ίσως να αποτρέπει κάποιον νέο προγραμματιστή να ασχοληθεί με τη Ruby.

Βιβλιογραφία

1. “Beginning ruby, from novice to professional”, Peter Cooper, Μάρτιος 2006 APRESS

2. “Ruby On Rails, Enterprice application development” Elliot Smith & Rob Nichols, Οκτώβριος 2007, PACKT

3. “Agile web development with rails. third edition” Sam Ruby, Dave Thomas & David Heinemeier Hansson, Ιούνιος 2008, Pragmatic Bookself

4. “Practical rest on rails projects 2”, Ben Scofield, Απρίλιος 2008, APRESS

5. “Simply rails 2”, Patrick Lenz, Μάιος 2008, SitePoint

6. “The art of rails”, Edward Benson, Μάιος 2008, WROX

7. “The rails way”, ObieFernandez, Νοέμβριος 2007, Addison-Wesley

8. "Εργαστήριο - Ειδικά θέματα Βάσεων Δεδομένων.", Γ. Γκαράνη, ΤΕΙ Λάρισας

9. "Εισαγωγή στην HTML και τα CSS.", Χ. Κόπανος, ΤΕΙ Λάρισας

10.http://el.wikipedia.org/wiki/Ηλεκτρονικά_καταστήματα Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την έννοια του ηλεκτρονικού εμπορίου.

11.http://el.science.wikia.com/wiki/Ηλεκτρονικό_Εμπόριο Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikia.org που αναλύει την έννοια του ηλεκτρονικού εμπορίου.

12.http://www.efpolis.gr/el/diasfalisi-oikonomikon-symefronton-katanaloton/ilektroniko-emporio.html. Άρθρο στην σελίδα της γενικής γραμματείας καταναλωτή.

13. 2tee-n-smyrn.att.sch.gr/E-Commerce.doc Εργασία των Διαμαντάκης Σταύρος – Μαραβέλιας Θεόφιλος – Σάρλος Στέλιος – Τέλιος Σαβέριος, 2ο ΤΕΕ νέας σμύρνης.

14. http://eclass.farm.teithe.gr/FARM132/work/429d59a7d1cf4/POLYDOROS GEORGIOS.doc εργασία του Γιώργου Πολύδωρου, ΤΕΙ Θεσσαλονίκης.

15. http://en.wikipedia.org/wiki/Ruby_(programming_language) Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την γλώσσα προγραμματισμού Ruby.

16. http://www.ruby-lang.org/en/ Επίσημη ιστοσελίδα της γλώσσας προγραμματισμού Ruby

17. http://en.wikipedia.org/wiki/Open_source Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την έννοια του Open Source.

18. http://en.wikipedia.org/wiki/Basecamp_(software) Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει το πρόγραμμα Basecamp.

19. http://en.wikipedia.org/wiki/37signals Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που διαπραγματεύεται την εταιρία 37signals.

20. http://en.wikipedia.org/wiki/Interactive_Ruby_Shell Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που διαπραγματεύεται το Interactive Ruby Shell.

21. http://en.wikipedia.org/wiki/Ruby_on_Rails Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει το framework ruby on rails.

22. http://www.tutorialspoint.com/ruby-on-rails/index.htm Επίσημη ιστοσελίδα του framework ruby on rails που περιέχει οδηγούς (tutorials) σχετικά με το framwork.

23. http://el.wikiversity.org/wiki/Βάσεις_Δεδομένων/Σχεσιακό_μοντέλο Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει το σχεσιακό μοντέλο.

24. http://en.wikipedia.org/wiki/Database Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την έννοια της βάσης δεδομένων.

25. http://en.wikipedia.org/wiki/Relational_database Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την έννοια των σχεσιακών βάσεων δεδομένων.

26. http://en.wikipedia.org/wiki/Database_management_system Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει τα συστήματα διαχείρισης βάσεων δεδομένων.

27. http://en.wikipedia.org/wiki/MySQL Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που παρουσιάζει την MySQL

28. http://dev.mysql.com/doc/query-browser/el/mysql-query-browser-introduction.html Άρθρο στην επίσημη ιστοσελίδα της MySQL.

29. http://www-css.fnal.gov/dsg/external/freeware/mysqlInfo.html 'Αρθρο στην επίσημη ιστοσελίδα της Computing Division.

30. www.w3schools.com Επίσημη ιστοσελίδα της w3schools.

31. http://en.wikipedia.org/wiki/Salt_(cryptography) Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που αναλύει την κρυπτογράφηση salt.

32. http://en.wikipedia.org/wiki/TinyMCE Άρθρο στην διαδικτυακή εγκυκλοπαίδεια wikipedia.org που παρουσιάζει το TinyMCE

33. http://am.rubyonrails.org/ Οδηγός στην επίσημη ιστοσελίδα του ruby on rails.

34. http://api.rubyonrails.org/classes/ActionMailer/Base.html Οδηγός στην επίσημη ιστοσελίδα του ruby on rails.