ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υπηρεσίες Διαδικτύου σε Ruby και Java

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ε. Ανδρεαδάκης

Επιβλέπων: Γεώργιος Στασινόπουλος

Καθηγητής Ε.Μ.Π.

ΑΘΗΝΑ Ιούνιος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υπηρεσίες Διαδικτύου σε Ruby και Java

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ε. Ανδρεαδάκης

Επιβλέπων: Γεώργιος Στασινόπουλος

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28η Ιουνίου 2017.

.....	.....	.....
Γ. Στασινόπουλος	Ε. Συκάς	Ν. Μήτρου
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

ΑΘΗΝΑ Ιούνιος 2017

.....

Γεώργιος Ε. Ανδρεαδάκης

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

Περίληψη

Σκοπός της εργασίας είναι η υλοποίηση ενός συστήματος επικοινωνίας με μία βάση δεδομένων όπου όλα τα στοιχεία υλοποίησης και διασύνδεσης θα πραγματοποιηθούν με τη γλώσσα προγραμματισμού Ruby . Το μοντέλο θα στηρίζεται στην αρχιτεκτονική MVC (model – view – controller) όπου κάθε στοιχείο από τα M V C έχει το δικό του ρόλο σε ένα ολοκληρωμένο σύστημα επικοινωνίας . Για το σκοπό αυτό θα χρησιμοποιήσουμε ένα framework από τα διαθέσιμα για την γλώσσα Ruby όπως είναι τα Ruby-On-Rails και Sinatra .
Τελικά προτιμήθηκε το Ruby-On-Rails για λόγους που θα αναλυθούν στη συνέχεια διεξοδικότερα για την αποσαφήνιση αυτής της επιλογής . Και τα δύο είναι ανοιχτού λογισμικού και η χρήση τους δεν προϋποθέτει κανένα περιορισμό . Οι ομοιότητες τους πολλές και οι διαφορές του επίσης .
Τέλος για την επικοινωνία και την διαχείριση της βάσης θα δημιουργήσουμε μια εφαρμογή σε JavaFX ,την οποία μπορεί ο καθένας χρησιμοποιώντας ένα “περιτύλλιγμα “, να μεταφέρει αυτούσια σε ένα συγχρονο κινητό τηλέφωνο που χρησιμοποιεί το λειτουργικό σύστημα Android .

Ένας περιφρραστικός τίτλος της εργασίας είναι :
“ **Σύστημα εξυπηρετητή με Web services γραμμένες στη γλώσσα προγραμματισμού Ruby , γραφικό περιβάλλον διαχείρισης της βάσης δεδομένων της εφαρμογής γραμμένο σε JavaFX και εφαρμογή πελάτη για το λειτουργικό σύστημα Android ,για την κατανάλωση αυτών .“**

Λέξεις κλειδιά : εφαρμογή ιστού , Ruby ,JAVAFX, framework , αρχιτεκτονική Model – View – Controller,CRUD , RestFul , DAO.

ABSTRACT

The purpose of the work is to implement a communication system with a database where all the elements of implementation and interconnection will be realized with the Ruby programming language. The model will be based on the MVC (model - view - controller) architecture where each MVC element has its own role in an integrated communication system.
To do this, we will use a framework between Ruby-On-Rails and Sinatra available for the Ruby language.
Ruby-On-Rails was eventually preferred for reasons that will be discussed later in more detail to clarify this option. Both are open source software and their use does not require any limitations. Their many similarities and differences also.
Finally, for the communication and management of the database, we will create a JavaFX application, which anyone can use with an "wrapper", to transfer itself to a modern mobile phone using the Android operating system.

KEYWORDS : Web Application , Ruby ,JAVAFX, framework , Model – View –Controller, CRUD , RestFul , DAO.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ, ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

1.1 Χρησιμότητα των διαδικτυακών εφαρμογών

Η ανάπτυξη του παγκόσμιου ιστού έχει επιφέρει αλλαγές κρίσιμες στα μοντέλα συμπεριφοράς των καταναλωτών και γενικότερα στον τρόπο με τον οποίο οι χρήστες αναζητούν, αξιολογούν, παράγουν, αγοράζουν και καταναλώνουν πληροφορίες, προϊόντα και υπηρεσίες. Μιλώντας γενικότερα, το Web 2.0 έχει ενδυναμώσει τους καταναλωτές δίνοντάς τους την πληροφορία αλλά και τα εργαλεία ώστε να συμμετέχουν ενεργά στις διάφορες παραγωγικές διαδικασίες των επιχειρήσεων, όπως ο σχεδιασμός και η ανάπτυξη και διανομή των προϊόντων .

Από την άλλη μεριά, πολλές επιχειρήσεις έχουν ήδη συνειδητοποιήσει τη δύναμη και την αξία που έχουν οι εφαρμογές και οι τεχνολογίες του Web 2.0 και προσπαθούν ολοένα και περισσότερο να συλλέξουν, να διαχειριστούν και να επωφεληθούν από από αυτήν την εξέλιξη. Η αξιοποίηση του Web 2.0 στις διάφορες λειτουργίες του μάρκετινγκ αποτελεί σήμερα την πλέον γνωστή και ευρέως υιοθετημένη εφαρμογή του Web 2.0 από τις επιχειρήσεις. Ωστόσο, οι περισσότερες καινοτομικές επιχειρήσεις ενσωματώνουν Web 2.0 σε διάφορα στάδια της αλυσίδας παραγωγής τους, αποκομίζοντας πολλά οφέλη και ενισχύοντας την πελατειακή τους πιστότητα . Η ενεργή όμως συμμετοχή των χρηστών στην ανάπτυξη και σχεδιασμό δεν εμφανίζεται βέβαια χωρίς προβλήματα, π.χ. Δικαιώματα πνευματικής ιδιοκτησίας ενώ οι επιχειρήσεις τις περισσότερες φορές βρίσκονται ανέτοιμες μπροστά σε τέτοιες περιπτώσεις. Ακόμη, πολλές εταιρίες χρησιμοποιούν το Web 2.0 για την ενεργή συμμετοχή του πελάτη στην παραγωγή και κατανάλωση προϊόντων.

Τελειώνοντας, είναι γεγονός ότι βρισκόμαστε σε μια νέα φάση εξέλιξης όπου το διαδίκτυο παρέχει στους πάντες κάθε είδους πληροφορία και όπου οι καταναλωτές έχουν ακόμη και τη δυνατότητα διαμόρφωσης πραγμάτων και καταστάσεων. Το πως αυτό θα επηρεάσει τις επιχειρήσεις και τι επιπτώσεις θα έχει γενικότερα είναι κάτι που το μάθουμε στο μέλλον.

1.2 Αντικείμενο της διπλωματικής

Η ενοποίηση των παραπάνω υπηρεσιών σε μία εφαρμογή όπου μέσω κινητού τηλεφώνου κάποιος χρήστης θα μπορεί να διαχειρίζεται μία σχεσιακή βάση δεδομένων που έχει κατασκευαστεί αποκλειστικά με εργαλεία ανοικτού κώδικα και χρησιμοποιεί τη γλώσσα προγραμματισμού Ruby και Java για την κατασκευή της .

2 ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Τεχνολογίες

Η αρχιτεκτονική MVC (model view controller)

Είναι αλήθεια ότι ο όρος MVC , ο οποίος σημαίνει Model-View-Controller, δεν είναι και τόσο νέος στον προγραμματισμό.Εμφανίστηκε για πρώτη φορά στην γλώσσα προγραμματισμού smalltalk.

Η smalltalk είναι μία γλώσσα προγραμματισμού αντικειμενοστραφής (oop) η οποία έχει την εξής δυνατότητα: Κατά τη διάρκεια που τρέχουμε ένα πρόγραμμα η smalltalk μπορεί να αλλάζει τη δομή και την συμπεριφορά των δεδομένων του προγράμματος καθώς και διαφόρων συναρτήσεων αυτού κλπ.Υπάρχει από το 1978 και το project Smalltalk της εταιρίας XEROX.

Σε αυτό λοιπόν το project ορίστηκε για πρώτη φορά ο όρος MVC. Για να γίνει πιο κατανοητό το MVC είναι ένα είδος αρχιτεκτονικής ,είναι δηλαδή ένας συγκεκριμένος τρόπος με τον οποίο “χτίζουμε” ένα application. Ο τρόπος αυτός θα αναλυθεί με απλό τρόπο παρακάτω. Πρώτα όμως ας δούμε τα πλεονεκτήματα που έχει η αρχιτεκτονική MVC.

2.2 Βασικά Πλεονεκτήματα MVC

Χτίζοντας μία εφαρμογή με MVC έχουμε τα εξής βασικά πλεονεκτήματα:

2.2.1 Διαχωρισμός Προβλημάτων (Separation of Concerns).

Αυτό είναι και το πιο βασικό πλεονέκτημα του MVC. Ουσιαστικά δημιουργείται μία εφαρμογή η οποία έχει τρία επίπεδα, το επίπεδο των models , το επίπεδο των controllers και το επίπεδο των views -που θα αναλυθούν παρακάτω- και το κάθε επίπεδο επιτελεί ξεχωριστό έργο και ταυτόχρονα συνεργάζεται με τα άλλα επίπεδα. Μία σωστή MVC εφαρμογή είναι εκείνη που τα τρία επίπεδα είναι ξεκάθαρα καθορισμένα και δεν συμπλέκονται. Για παράδειγμα είναι λάθος στο επίπεδο των View να υπάρχει κώδικας που “μιλάει” με την βάση δεδομένων και “τραβάει” δεδομένα.

2.2.2 Επεκτασιμότητα.

Το δεύτερο πλεονέκτημα της MVC αρχιτεκτονικής είναι πολύ σημαντικό επίσης. “Επεκτασιμότητα” είναι η δυνατότητα που διαθέτει μία εφαρμογή , κατά την οποία μπορούμε μελλοντικά να προσθέσουμε λειτουργίες σε αυτή ή να αλλάξουμε κάποιες από τις ήδη υπάρχουσες λειτουργίες και να έχουμε άλλα αποτελέσματα. Για να το δούμε εντελώς απλά και κατανοητά, η πλατφόρμα WordPress είναι επεκτάσιμη με τη χρήση των διάφορων plugins διότι προσθέτουμε στις ήδη υπάρχουσες λειτουργίες και άλλες λειτουργίες. Τα προγράμματα που είναι φτιαγμένα με MVC αρχιτεκτονική έχουν βασικό χαρακτηριστικό ότι είναι επεκτάσιμα.

2.2.3Ελεγξιμότητα (Testability)

Αυτό είναι ένα πολύ κρίσιμο χαρακτηριστικό. Οι MVC εφαρμογές έχουν την δυνατότητα να είναι ελέγξιμες και με τον τρόπο αυτό συντηρούνται πιο εύκολα. Ας κάνουμε ένα απλό παράδειγμα. Έστω ότι έχουμε μία εφαρμογή η οποία διαθέτει μία λειτουργία login, δηλαδή ζητά από τον χρήστη να πληκτρολογήσει κάποια στοιχεία σε μία φόρμα και εν συνέχεια τον εισάγει μέσα στο σύστημα. Αυτή τη λειτουργία την ελέγχει κάποιος loginController ο οποίος περιέχει κώδικα που διαχειρίζεται τα δεδομένα αυτά που εισήχθησαν από τον χρήστη. Αυτός ο controller θεωρείται μία “μονάδα” ή αλλιώς unit. Στα MVC frameworks μπορούμε με πολλή ευκολία να γράψουμε απλό κώδικα-tests με τον οποίο τεστάρουμε αυτόν τον controller αλλά και κάθε μία από τις λειτουργίες του. Παίρνουμε τα αποτελέσματα και βλέπουμε αν η συγκεκριμένη μονάδα της εφαρμογής μας λειτουργεί σωστά.

2.2.4 “Καθαρά” URLs.

Τα περισσότερα MVC frameworks για web applications δίνουν τη δυνατότητα να έχουμε “καθαρά” urls. Ας κάνουμε ένα παράδειγμα. Έστω ότι έχουμε ένα blog και πατάμε το link για να διαβάσουμε ένα άρθρο. Ένα τυπικό URL θα μπορούσε να ήταν

http://example.com/article/Page.aspx?action=show&art_id=236.

http://example.com/article/Page.aspx?action=show&art_id=236.

Με το MVC μπορούσαμε να έχουμε το εξής

http://example.com/articles/nice-link

http://example.com/articles/nice-link

Βλέπουμε λοιπόν ότι με το MVC μπορούμε να έχουμε πιο όμορφα και εύληπτα από το χρήστη αλλά και την μηχανή αναζήτησης URLs.Ας πάμε λοιπόν στα βασικά.Τι σημαίνουν τα αρχικά του MVC;

2.2.5 Model

Στο model τοποθετούμε τις λειτουργίες της εφαρμογής που σχετίζονται με την πρόσβαση στη βάση δεδομένων. Οι λειτουργίες αυτές είναι με τη μορφή function(μεθόδων στον προγραμματισμό). Είναι κάποιες συναρτήσεις με τις οποίες εκτελούμε διάφορες λειτουργίες διαχείρισης των δεδομένων που λαμβάνουμε από τη βάση. Για παράδειγμα αν θέλουμε σε μία σελίδα να εμφανίσουμε κάποια βιβλία από τη βάση δεδομένων ,το πρώτο βήμα είναι ότι στο model των βιβλίων(θα μιλήσουμε σε λίγο γι αυτό) υπάρχει κάποια function για παράδειγμα “getAllBooks()” η οποία περιέχει κώδικα που μιλάει με τη βάση και τραβάει τα δεδομένα που θέλουμε.

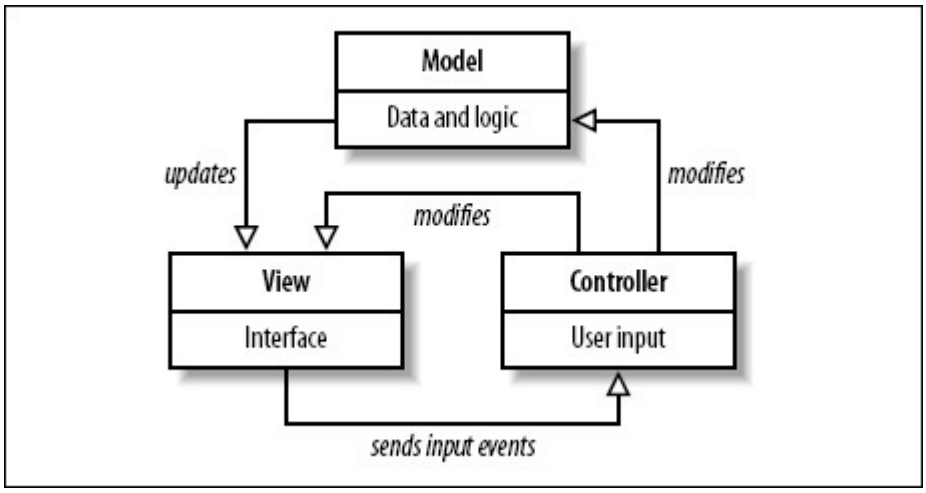
2.2.6 View

Μέσα στη view υπάρχει το HTML της σελίδας της εφαρμογής μας.Είναι αυτό που βλέπουμε. Τις περισσότερες φορές μία View μιλάει με ένα controller και αφού ο controller κάνει τις διάφορες επεξεργασίες των δεδομένων στέλνει στη View συγκεκριμένα δεδομένα να εμφανίσει.

2.2.7 Controller

Ο controller είναι ο μεσίτης μεταξύ Model και της View. Ελέγχει το πώς “τρέχει” η εφαρμογή.Μιλάει με το Model, παίρνει τα δεδομένα που ζητά και εν συνεχεία και αφού τα επεξεργαστεί τα στέλνει πίσω στη View για απεικόνιση.Ας δούμε μία συνολική εικόνα για το MVC.

2.2.8 Ένα παράδειγμα-επεξήγηση



Σχήμα 2.2.8 : MVC αρχιτεκτονική

Στην παραπάνω συμβαίνουν τα εξής: Ο χρήστης δίνει κάποιο input στο site. Για παράδειγμα συμπληρώνει μία φόρμα και πατάει το κουμπί submit. Στη συνέχεια ο controller έχοντας λάβει το input του χρήστη μιλάει με το model χρησιμοποιώντας το input που του χρήστη σαν μεταβλητή και ζητάει δεδομένα από το model τα οποία όταν τα λαμβάνει τα προσαρμόζει ανάλογα με αυτό που ζήτησε ο χρήστης.Στη συνέχεια ο controller με τα ΝΕΑ δεδομένα πλέον , αλλάζει την view.Για παράδειγμα εάν σε ένα application βιβλιοπωλείου ο χρήστης συμπληρώσει μία search form για βιβλία με συγκεκριμένη κατηγορία ο αντίστοιχος controller θα “μιλήσει” με κάποια function του model που θα ζητάει όλα τα βιβλία και θα δέχεται ως παράμετρο την κατηγορία.Το model θα βρίσκει τα βιβλία αυτά και μέσω της χρήσης της function στο controller, αυτός θα τα εμφανίζει στην οθόνη.

2.2.9 Οντότητες-entities στην MVC αρχιτεκτονική

Τον όρο “οντότητα” ή στα Αγγλικά “entity” τον συναντούμε πολύ συχνά στο MVC μοντέλο αλλά και γενικότερα στον προγραμματισμό. Είναι όρος που προκύπτει κατά τη διαδικασία μοντελοποίησης ενός συστήματος και δίνει την δυνατότητα στον developer να δει από ποιες βασικές οντότητες απαρτίζεται η εφαρμογή του. Για παράδειγμα σε μία εφαρμογή με blogs, άρθρα, σχόλια, χρήστες οι οντότητες είναι: ο χρήστης, το blog, το comment, το άρθρο κλπ.Είναι πράγματα δηλαδή που υπάρχουν και πάνω σε αυτά βασίζουμε τις λειτουργίες του συστήματός μας. Συνήθως με τον όρο entity παρουσιάζουμε και ένα πίνακα από μία βάση δεδομένων. Δημιουργούμε “οντότητες” και μας βοηθά αυτό να δημιουργήσουμε την εφαρμογή μας πιο εύκολα και πιο σίγουρα.

2.3 Frameworks που χρησιμοποιούν την αρχιτεκτονική MVC

Για να δημιουργήσουμε μία εφαρμογή σε MVC μπορούμε να χρησιμοποιήσουμε κάποιο framework,ανάλογα τη γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε, το οποίο framework κάνει τη διαδικασία δημιουργίας του application πιο γρήγορη ,πιο ασφαλή-διότι σε ένα framework υπάρχουν πολλές λειτουργίες by default , δεν χρειάζεται να γράψουμε ολόκληρο τον κώδικα- και πιο ευχάριστη. Σε γλώσσα Ruby γνωστά frameworks είναι τα Sinatra και RubyOnRails.

2.3.1 Ποιό framework είναι κατάλληλο για την εργασία μας ;

Γενικότερα υπάρχουν πολλές ομοιότητες στις ιδιότητες αυτών των εφαρμογών αλλά και κάποιες μικρές διαφορές που αφορούν κυρίως το μέγεθος της εργασίας και την πολυπλοκότητα του έργου . Δηλαδή για περισσότερο πολύπλοκες εφαρμογές συνίσταται η χρήση του RubyOnRails .
Η επιλογή του στηρίχθηκε κυρίως σε δύο κριτήρια .
Το πρώτο είναι ότι η ελλειπής τεκμηρίωση για το Sinatra σε θέματα που αφορούν διαχείριση μεγάλων βάσεων δεδομένων όπου οι συνδέσεις μεταξύ των στοιχείων της είναι πιο πολύπλοκες .
Δεύτερο στοιχείο επιλογής είναι ο μονολιθικός χαρακτήρας του Sinatra σε θέματα δημιουργίας της αρχιτεκτονικής MVC . Πολλά από τα models , Controllers θα πρέπει να δημιουργηθούν σε ένα αρχείο μαζί κάτι που κάνει τη χρήση ,την επίβλεψη και την επέκταση του δυσκολότερη .

Ένα επιπλέον θετικό χαρακτηριστικό για το Rails μπορεί να θεωρηθεί η ύπαρξη κονσόλας επικοινωνίας με τον sql server όπου μπορούμε με απλές εντολές γραμμένες στη γλώσσα Ruby ν' αλληλεπιδράσουμε με τη βάση δεδομένων , κάτι που είναι εξαιρετικά χρήσιμο κατά το στάδιο της ανάπτυξης .

2.4 Η γλώσσα προγραμματισμού Ruby

Η **Ruby** είναι μια δυναμική, ανακλαστική, αντικειμενοστρεφής γλώσσα προγραμματισμού γενικής χρήσης που συνδυάζει μια σύνταξη επηρεασμένη από την Perl με χαρακτηριστικά από τη Smalltalk. Η Ruby προήλθε από την Ιαπωνία στα μέσα της δεκαετίας του 1990 και αρχικά σχεδιάστηκε και αναπτύχθηκε από τον Yukihiro "Matz" Matsumoto. Βασικές της επιρροές είναι η Perl, η Smalltalk, η Eiffel και η Lisp.

Η Ruby υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού όπως ο συναρτησιακός προγραμματισμός, ο αντικειμενοστρεφής προγραμματισμός, ο προστακτικός προγραμματισμός και ο ανακλαστικός (reflective) προγραμματισμός. Έχει σύστημα δυναμικών τύπων και αυτόματη διαχείριση μνήμης, επομένως μοιάζει σε κάποια χαρακτηριστικά της με την Python, την Perl, τη Lisp, τη Dylan, την Pike και τη CLU.

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

2.4.1 Χαρακτηριστικά

- Πλήρως αντικειμενοστρεφής με κληρονομικότητα, mixin, και μετακλάσεις
- Δυναμικοί τύποι και Duck typing
- Τα πάντα είναι εκφράσεις (ακόμα και οι εντολές) και τα πάντα εκτελούνται προστακτικά (ακόμα και οι δηλώσεις)
- Σύντομη και ευέλικτη σύνταξη που ελαχιστοποιεί το "συντακτικό θόρυβο" και είναι η βάση για γλώσσες συγκεκριμένων πεδίων (domain specific languages)
- Δυναμική ανάκλαση και τροποποίηση αντικειμένων στο χρόνο εκτέλεσης για τη διευκόλυνση του μεταπρογραμματισμού
- Κλεισίματα, απαριθμητές και γεννήτριες, με μια μοναδική σύνταξη ενοτήτων (blocks)
- Ρητός συμβολισμός για δυναμικούς πίνακες, πίνακες κατακερματισμού (hashes), κανονικές εκφράσεις και σύμβολα
- Ενσωμάτωση κώδικα σε συμβολοσειρές (variable interpolation)
- Παράμετροι με εξορισμού τιμές (Default arguments)
- Τέσσερα επίπεδα εμβέλειας μεταβλητών: καθολική, κλάσης, στιγμιότυπου κλάσης και τοπική, που σημειώνονται με σύμβολα (sigils) και χρήση κεφαλαίων-μικρών ανά περίπτωση

- Αυτόματη συλλογή απορριμμάτων
- Συνέχειες (continuations) πρώτης τάξης
- Αυστηροί κανόνες έμμεσης μετατροπής τιμών αληθείας (τα πάντα είναι αληθή εκτός του `false` και του `nil`)
- Χειρισμός εξαιρέσεων
- Υπερφόρτωση τελεστών
- Ενσωματωμένη υποστήριξη για ρητούς αριθμούς, μιγαδικούς αριθμούς και αριθμητική άπειρης ακρίβειας
- Ρυθμιζόμενη διανομή των κλήσεων μεθόδων (dispatch) (με τις `method_missing` και `const_missing`)
- Εγγενή νήματα και συνεργατικές ίνες (cooperative fibers)
- Αρχική υποστήριξη για το πρότυπο Unicode και πολλαπλές κωδικοποιήσεις χαρακτήρων (αν και με κάποια προβλήματα μέχρι την έκδοση 1.9)
- plug-in API σε C
- Το κέλυφος Ruby
- Κεντρική διαχείριση πακέτων μέσω των RubyGems
- Έχει υλοποιηθεί σε όλες τις σημαντικές πλατφόρμες
- Μεγάλη βασική βιβλιοθήκη

2.4.2 Η Φιλοσοφία της γλώσσας

Σύμφωνα με τον δημιουργό της γλώσσας Yukihiro Matsumoto , η γλώσσα θα έπρεπε να έχει τα εξής χαρακτηριστικά

1. Να είναι εύκολη η εκμάθηση της

2. Να είναι ευέλικτη αρκετά ώστε να μπορεί να χειριστεί κάθε προγραμματιστική απαίτηση

3. Να στρεσσάρει ελάχιστα τον προγραμματιστή

4. Να είναι αντικειμενοστρεφής

2.4.3 Σημασιολογία

Η Ruby είναι αντικειμενοστρεφής: κάθε τύπος δεδομένων είναι αντικείμενο, συμπεριλαμβανομένων και των κλάσεων και των τύπων που άλλες γλώσσες θεωρούν βασικούς (όπως οι ακέραιοι, οι τιμές αλήθειας και η τιμή "nil"). Κάθε συνάρτηση είναι μια μέθοδος. Οι τιμές με όνομα (μεταβλητές) πάντα είναι αναφορές σε αντικείμενα, και όχι τα ίδια τα αντικείμενα. Η Ruby υποστηρίζει κληρονομικότητα με δυναμική διανομή μεθόδων (dynamic dispatch), mixin και μεθόδους singleton (που ανήκουν και ορίζονται μόνο σε ένα αντικείμενο και όχι σε ολόκληρη την κλάση). Αν και η Ruby δεν υποστηρίζει πολλαπλή κληρονομικότητα, οι κλάσεις μπορούν να εισάγουν μονάδες κώδικα (modules) σαν mixin. Η διαδικαστική σύνταξη υποστηρίζεται αλλά όλες οι μέθοδοι που ορίζονται εκτός της εμβέλειας ενός συγκεκριμένου αντικειμένου είναι στην πραγματικότητα μέθοδοι της κλάσης Object. Επειδή η κλάση αυτή είναι η βασική κλάση κάθε άλλης βάσης, οι αλλαγές είναι ορατές σε όλες τις κλάσεις και τα αντικείμενα.

Η Ruby έχει περιγραφεί σαν γλώσσα προγραμματισμού πολλών παραδειγμάτων: επιτρέπει διαδικαστικό προγραμματισμό (ο ορισμός συναρτήσεων/μεταβλητών εκτός κλάσεων τις κάνει μέρος της κλάσης ρίζας, 'self' Object), με αντικειμενοστρεφή χαρακτηριστικά (τα πάντα είναι αντικείμενα) ή συναρτησιακά χαρακτηριστικά και συνέχειες - όλες οι εντολές έχουν τιμές, και οι συναρτήσεις επιστρέφουν την τελευταία τιμή τους). Επιτρέπει ενδοσκόπηση (introspection), ανάκλαση (reflection) και μεταπρογραμματισμό, και υποστηρίζει [18] νήματα βασισμένα στο διερμηνέα. Η Ruby έχει δυναμικό σύστημα τύπων και υποστηρίζει παραμετρικό πολυμορφισμό.

2.4.4 Σύνταξη

Η σύνταξη της Ruby είναι παρόμοια με αυτή της Perl και της Python. Οι ορισμοί κλάσεων και μεθόδων ξεχωρίζουν από λέξεις-κλειδιά. Σε αντίθεση με την Perl, δεν είναι απαραίτητο οι μεταβλητές να έχουν στην αρχή τους ένα αναγνωριστικό σύμβολο. Όταν αυτό όμως χρησιμοποιείται αλλάζει η σημασία της εμβέλειας της μεταβλητής. Η εμφανέστερη διαφορά σε σχέση με τη C και την Perl είναι ότι συνήθως χρησιμοποιούνται λέξεις-κλειδιά για τον ορισμό λογικών ενοτήτων κώδικα, χωρίς αγκύλες (δηλ., ζευγάρι { και }). Για πρακτικούς λόγους, δεν υπάρχει διαχωρισμός μεταξύ εκφράσεων και εντολών[20]. Οι αλλαγές γραμμής είναι σημαντικές και αποτελούν το τέλος μιας εντολής - επίσης μπορεί να χρησιμοποιηθεί ένα ελληνικό ερωτηματικό στη θέση τους. Σε αντιδιαστολή με την Python, οι εσοχές (indentation) δεν έχουν σημασία.

Μια από τις διαφορές της Ruby σε σχέση με την Python και την Perl είναι ότι η Ruby κρατά όλες

τις μεταβλητές ενός αντικειμένου σαν ιδιωτικές (private) μέσα σε μια κλάση και τις εκθέτει προς τα έξω μόνο μέσω μεθόδων πρόσβασης (attr_writer, attr_reader, κλπ.). Σε αντίθεση με τις μεθόδους "getter" και "setter" άλλων γλωσσών όπως η C++ ή η Java, οι μέθοδοι πρόσβασης στη Ruby δημιουργούνται με μια γραμμή κώδικα με χρήση μεταπρογραμματισμού. Επειδή η κλήση αυτών των μεθόδων δε χρειάζεται παρενθέσεις, είναι εύκολο να αλλάξουμε μια μεταβλητή ενός αντικειμένου σε μια πλήρη συνάρτηση, χωρίς να αλλάξουμε ούτε μια γραμμή κώδικα ή refactoring, επιτυγχάνοντας παρόμοια λειτουργικότητα με τα μέλη ιδιοτήτων της C# και της VB.NET. Οι περιγραφείς ιδιοτήτων της Python (property descriptors) είναι παρόμοιοι αλλά έχουν ένα κόστος στη φάση της ανάπτυξης του λογισμικού. Αν κάποιος αρχίσει στην Python με μια δημόσια μεταβλητή ενός αντικειμένου και στη συνέχεια αλλάξει την υλοποίησή της σε μια ιδιωτική μεταβλητή που φαίνεται μέσα από έναν περιγραφέα ιδιοτήτων, ο εσωτερικός κώδικας της κλάσης μπορεί να χρειαστεί να διορθωθεί ώστε να χρησιμοποιεί την ιδιωτική μεταβλητή και όχι τη δημόσια. Η Ruby αφαιρεί αυτήν την επιλογή της σχεδίασης, επιβάλλοντας όλες οι μεταβλητές αντικειμένων να είναι ιδιωτικές, αλλά παρέχει έναν απλό τρόπο να ορίζονται μέθοδοι get και set. Αυτό συμφωνεί με την ιδέα ότι στη Ruby, κανείς δεν έχει άμεση πρόσβαση στα εσωτερικά μέλη μιας κλάσης από το εξωτερικό της αλλά μπορεί μόνο να στείλει ένα μήνυμα στην κλάση και να λάβει μια απάντηση.

2.5 Το RubyOnRails web framework

Το Ruby-On-Rails (ROR) είναι ένα πλαίσιο ανάπτυξης λογισμικού ιστού ανοιχτού κώδικα για την γλώσσα προγραμματισμού Ruby. Προορίζεται για χρήση σε συνδυασμό με ευέλικτες μεθοδολογίες ανάπτυξης , οι οπόίες χρησιμοποιούνται από τους προγραμματιστές ιστού για ταχεία ανάπτυξη εφαρμογών ιστού .

2.5.1 Ιστορία

Το Ruby on Rails προήλθε από τη δουλειά του David Heinemeier Hansson στο Basecamp, ένα εργαλείο διαχείρισης project από την εταιρεία 37signals (η οποία τώρα είναι εταιρεία ανάπτυξης λογισμικού Ιστού). Ο David Hansson αρχικά κυκλοφόρησε το Rails σαν ανοιχτό κώδικα τον Ιούλιο του 2004 αλλά δεν επέτρεπε σε άλλους προγραμματιστές να συνεισφέρουν κώδικα στο εγχείρημα μέχρι το Φεβρουάριο του 2005.[9] Τον Αύγουστο του 2006 υπήρξε κομβικό σημείο για το Rails,

όταν η Apple ανακοίνωσε ότι θα κυκλοφορούσε το Ruby on Rails μαζί με το Mac OS X v10.5 "Leopard", το οποίο κυκλοφόρησε τον Οκτώβριο του 2007.

Η έκδοση 2.3 του Rails κυκλοφόρησε στις 15 Μαρτίου του 2009. Βασικά νέα χαρακτηριστικά του Rails ήταν τα πρότυπα (templates), οι μηχανές (engines), το Rack και οι εμφωλευμένες φόρμες μοντέλων.

- Τα πρότυπα επιτρέπουν στον προγραμματιστή να δημιουργεί το σκελετό μιας εφαρμογής με ειδικά gems και ρυθμίσεις (configurations).
- Οι μηχανές επιτρέπουν τη χρήση τμημάτων εφαρμογών σε άλλες εφαρμογές, συμπεριλαμβανομένων των χαρακτηριστικών "routes", "view paths" και "models".
- Η διαπροσωπεία Rack και το Metal επιτρέπουν στον προγραμματιστή να γράφει βελτιστοποιημένα κομμάτια κώδικα που μπορούν να δρομολογούνται (route) σε σχέση με την ActionController.

Στις 23 Δεκεμβρίου 2008 ξεκίνησε το Merb, ένα άλλο πλαίσιο ανάπτυξης εφαρμογών Ιστού, και το Rails ανακοίνωσε ότι επρόκειτο να συνεργαστούν. Η ομάδα του Rails ανακοίνωσαν ότι θα συνεργάζονταν με το εγχείρημα Merb για να φέρουν "τις καλύτερες ιδέες του Merb" στο Rails 3, δίνοντας τέλος στο φαινόμενο παρόμοιας δουλειάς που γινόταν σε δύο σημεία, στις δύο κοινότητες.

Η πιο πρόσφατη έκδοση του είναι η 5.0.2 και τα χαρακτηριστικά που προσφέρει σε σχέση με τις προηγούμενες εκδόσεις μπορεί κανείς να τα διαβάσει στη σελίδα ανάπτυξης http://weblog.rubyonrails.org/releases/changelog .Η τελευταία major έκδοση είναι η 5.0

και είναι αυτή που θα χρησιμοποιήσουμε για την εφαρμογή μας .

2.5.2 Τεχνική σύνοψη

Όπως πολλά πλαίσια Ιστού, το Rails χρησιμοποιεί αρχιτεκτονική Model-View-Controller (MVC) για να οργανώσει τον προγραμματισμό των εφαρμογών.

Το Ruby on Rails περιλαμβάνει εργαλεία που διευκολύνουν κοινές προγραμματιστικές εργασίες, όπως η δημιουργία σκελετών προγραμμάτων (scaffolding) που μπορεί να δημιουργήσει αυτόματα κάποια από τα μοντέλα (models) και τις όψεις (views) που χρειάζεται μια βασική σελίδα Ιστού. Επίσης περιέχει τον WEBrick, έναν απλό εξυπηρετητή Ιστού σε Ruby, και το Rake, ένα σύστημα κατασκευής προγραμμάτων. Αυτά τα εργαλεία, σε συνδυασμό με το Rails, προσφέρουν ένα πλήρες

περιβάλλον ανάπτυξης διαδικτυακών εφαρμογών.

Το Ruby on Rails βασίζεται σε έναν εξυπηρετητή Ιστού για την εκτέλεσή του. Συνήθως προτιμάται ο Mongrel έναντι του WEBrick αλλά μπορεί να χρησιμοποιηθεί και ο Lighttpd, ο Abyss, ο Apache (σαν μονάδα κώδικα - π.χ. Passenger - ή μέσω του CGI, του FastCGI ή του mod_ruby), και πολλοί άλλοι. Από το 2008 ο εξυπηρετητής Passenger δείχνει να προτιμάται αντί για τον Mongrel. Πρόσφατα, παρατηρήθηκε συχνή χρήση του εξυπηρετητή puma που βρίσκεται στην έκδοση 3.xx και είναι αυτός που θα χρησιμοποιήσουμε για την εφαρμογή μας .

Το Rails είναι επίσης γνωστό για την εκτενή χρήση των βιβλιοθηκών JavaScript Prototype και Script.aculo.us για Ajax. Το Rails αρχικά έκανε χρήση ελαφρών κλήσεων SOAP για web services - αργότερα αυτό αντικαταστάθηκε από RESTful web services.

Από την έκδοση 2.0, το Ruby on Rails προσφέρει σαν μορφές εξόδου HTML και XML, με τη δεύτερη να χρησιμοποιείται και στα RESTful web services.Μεταγενέστερες εκδόσεις προσφέρουν και json .

2.5.3 Δομή

Το Ruby on Rails είναι χωρισμένο σε διάφορα πακέτα, το ActiveRecord (ένα σύστημα αντικειμενοστρεφούς-σχεσιακής αντιστοίχισης (object-relational mapping) για την πρόσβαση σε βάσεις δεδομένων), το ActiveResource (παρέχει web services), το ActionPack, το ActiveSupport και το ActionMailer. Πριν την έκδοση 2.0, το Rails περιλάμβανε και το πακέτο Action Web Service που τώρα αντικαθίσταται από το Active Resource. Εκτός από τα βασικά πακέτα, οι προγραμματιστές μπορούν να δημιουργήσουν plugins για να επεκτείνουν τα υπάρχοντα πακέτα.

3 Η ΕΦΑΡΜΟΓΗ

Θα δημιουργήσουμε μια σχεσιακή βάση δεδομένων για την διαχείριση των χρονοδιαγραμμάτων για τα projects μια εταιρείας μηχανικών ,ώστε να μπορούμε να κρατάμε εγγραφές για το χρόνο απασχόλησης των εταιρειών πελατών της εταιρείας .
Οι σχέσεις των μοντέλων της βάσης μεταξύ της θα επεξηγηθούν κατά την κατασκευή της με τη βοήθεια του framework RubyOnRails καθώς και κάποιες εγγραφές που θα πραγματοποιήσουμε κατά την ανάπτυξη της εφαρμογής .
Στο τέλος θα δημιουργήσουμε μια δεύτερη εφαρμογή για τη διαχείριση της βάσης δεδομένων από ένα κινητό τηλέφωνο με λειτουργικό σύστημα Android .Αρχικά θα δημιουργήσουμε την εφαρμογή σε JavaFx και έπειτα θα κάνουμε μεταφορά ολόκληρο το project στο λειτουργικό Android .

3.1.1 Κατασκευάζοντας το DOMAIN μου με το RubyOnRails

Αρχικά θα χρειαστούμε μια βάση δεδομένων την οποία θα δημιουργήσουμε με το Rails αποκλειστικά . Η βάση μπορεί να είναι οποιουδήποτε συστήματος βάσεων δεδομένων εδώ θα χρησιμοποιήσουμε την (sqlite3) . Ολόκληρη η εφαρμογή τουλάχιστον από την πλευρά του διακομιστή θα βασίζεται σε στοιχεία που θα δανειστούμε από το Rails και αργότερα θα επικοινωνήσουμε από το κινητό τηλέφωνο μαζί της.
Χρησιμοποιώντας το τερματικό του λειτουργικού συστήματος επιλέγουμε το φάκελο που επιθυμούμε να σχεδιάσουμε την εφαρμογή .

Με την εντολή **rails new CompanyTracker**
Αυτόματα το Rails δημιουργεί το σκελετό της εφαρμογής μέσα σε ένα φάκελο με το όνομα της δηλαδή CompanyTracker . (screenshot 1)

Όπως βλέπουμε στην εικόνα ,έχουμε το σκελετό μιας καλά δομημένης web εφαρμογή έτοιμης για επεξεργασία και ανάπτυξη .
Επόμενο βήμα είναι να περιηγηθούμε στην εφαρμογή .
Στον **root** φάκελο της εφαρμογής **CompanyTracker/** το πιο σημαντικό αρχείο που βλέπουμε είναι το αρχείο **Gemfile** .

Στην πραγματικότητα είναι το αρχείο που περιέχει όλες τις βιβλιοθήκες που θα χρησιμοποιηθούν για την ανάπτυξη της εφαρμογής .

Ανοίγω το Gemfile και προσθέτω τις παρακάτω γραμμές :

gem 'devise'
gem 'will_paginate'
gem 'puma'

αποθηκεύω το αρχείο και εκτελώ την εντολή *bundle install*
όπου η εφαρμογή συμπληρώνεται με τις απαραίτητες βιβλιοθήκες εργαλείων (gems) που θα χρησιμοποιήσουμε .

Στο σημείο αυτό το σύστημα είναι έτοιμο για την δημιουργία της εφαρμογής , αν χρειαστώ κάποιο module αργότερα να σημπληρώσω μπορώ να το κάνω με τον ίδιο τρόπο και επανάλληψη της παραπάνω διαδικασίας . Το πρόγραμμα bundle αναλλαμβάνει να εγκαταστήσει όλα τα απαραίτητα εργαλεία χρειάζονται .

Σε αυτό το σημείο να πω ότι η εφαρμογή είναι έτοιμη για οποιονδήποτε application/web server για να τρέξει , καθώς επίσης ήδη έχει ενσωματώσει τον δικό της puma . Για να ελέγξω αν όλα έχουν γίνει σωστά μπορώ είτε να δω ότι ο puma έχει συμπεριληφθεί , είτε απλά να τον τρέξω ...
Για κάθε module μπορώ να λάβω πληροφορίες για το που έχει εγκατασταθεί με την εντολή *bundle show [gemname]*
Για τον puma θα εκτελέσω bundle show puma .
Όπως είπα και πριν μπορώ απλά να ενεργοποιήσω τον puma και να επισκευτώ την αρχική σελίδα του .
Με την εντολή *rails server θα δούμε τα μηνύματα υποδοχής του puma webserver στο τερματικό μας :*

```
=> Booting Puma
=> Rails 5.0.0.1 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
```

```
* Version 3.6.0 (ruby 2.3.1-p112), codename: Sleepy Sunday Serenity
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
```

3.1.2 Δημιουργία της βάσης δεδομένων (Modelling the Domain)

Με βάση τον αρχικό σχεδιασμό που έχουμε πραγματοποιήσει ακριβώς παρακάτω γνωρίζοντας τις απαιτήσεις της εφαρμογής μας η οποία αποτελείται από πέντε πίνακες με τα παρακάτω πεδία ο κάθένας (Να σημειωθεί ότι το rails συμπληρώνει τους πίνακες με επιπλέον πεδία χωρίς να του ζητηθεί κατά την κατασκευή όπως για παράδειγμα το id που χρησιμοποιείται ως πρωτεύον κλειδί για κάθε πίνακα καθώς και ένα (δύο) πεδία τύπου String που χρησιμοποιεί για timestamps ώστε να κρατάει tracks για την δημιουργία αντικειμένων του πίνακα) :

1. Company

- Name

2.User

-Username

-Email (devise authentication module)

-Password (devise authentication module)

3. Worker

- First Name

- Second Name

-Email

- Company

4. Project

- Name

- Company

- Default Rate

5. Work

- Project

- Worker

- Date Performed

- Number of Hours

3.1.3 Models (M-V-C)

Η κατασκευή της βάσης δεδομένων με τη χρήση του RubyOnRails γίνεται πολύ απλά από ένα τερματικό και αφού βρισκόμαστε στον αρχικό φάκελο της εφαρμογής μας . Αρχικά δημιουργούμε τα μοντέλα για τους πίνακες μας με την απλή εντολή

rails g(enerate) model ModelName

Ειδικότερα εδώ

rails generate model Company

Αμέσως θα παρατηρήσουμε την δημιουργία ορισμένων φακέλων με κάποια αρχεία μέσα . Αυτά που μας ενδιαφέρουν άμεσα είναι τα δύο αρχεία ruby με κατάληξη .rb :

1. /app/models/company.rb και
2. /db/migration/timestamp_create_companies.rb για τα οποία θα μιλήσουμε αμέσως παρακάτω .

Όμοια για τις υπόλοιπες οντότητες της βάσης μου .

rails generate model User

rails generate model Project

rails generate model Worker

rails generate model Work

3.1.4 Active Record Migrations

Επόμενο βήμα από τη δημιουργία των κλάσεων είναι να επιλέξω τα πεδία που θα έχει η κάθε μια στον πίνακα της βάσης .

Αν ανοίξω τα αρχεία που ανέφερα παραπάνω (1 και 2) θα δω ότι το model κάθε οντότητας είναι άδειο και ότι ο πίνακας που θα δημιουργήσει το εργαλείο rake αν εκτελέσω *rake db:migrate* δεν είναι παρά τα default πεδία που έχω αναφέρει , δηλαδή το model.id και τα timestamps created_at , updated_at .

Για να αλλάξω τα χαρακτηριστικά της βάσης μου και εμπλουτίσω τις κλάσεις με τα πεδία που επιθυμώ θα πρέπει να κάνω τις όποιες αλλαγές θέλω στα αρχεία timstamp_create_model.db που βρίσκονται μέσα στο φάκελο /db/migration πρωτού εκτελέσω την εντολή rake db:migrate
Κάνω τις απαραίτητες αλλαγές βάση του αρχικού σκελετού που έχω για τη βάση μου :
Για απλότητα θα περιγράψω τη διαδικασία μονάχα για το company model . Η ίδια διαδικασία ακολουθείται για κάθε κλάση της βάσης δεδομένων που θέλω να φτιάξω . Σχετικά με τις αναφορές και το διαχωρισμό σε πρωτεύοντα και ξένα κλειδιά θα πω λίγα λόγια αμέσως μετά για τους συσχετισμούς μεταξύ των οντοτήτων της βάσης , τί σημαίνουν και πώς υλοποιούνται αυτοί .

t.string :name για το όνομα της εταιρείας .

Αφού τελείωσα εκτελώ την εντολή *rake db:migrate*

```
rake db:migrate

(in /home/trojan/Programming/WebApps/CompanyTracker)
== 20170405081225 CreateUsers: migrating
=============================================
-- create_table(:users)
   -> 0.0037s
== 20170405081225 CreateUsers: migrated (0.0045s)
```

```
==============================

== 20170405081239 CreateProjects: migrating
====================================
-- create_table(:projects)
   -> 0.0024s
== 20170405081239 CreateProjects: migrated (0.0026s)
============================

== 20170405081250 CreateWorkers: migrating
====================================
-- create_table(:workers)
   -> 0.0015s
== 20170405081250 CreateWorkers: migrated (0.0016s)
============================

== 20170405081258 CreateWorks: migrating
=======================================
-- create_table(:works)
   -> 0.0013s
== 20170405081258 CreateWorks: migrated (0.0023s)
============================
```

Τώρα που έχω τη βάση δεδομένων μου έτοιμη μπορώ να αρχίσω να εκτελώ είτε SQL Queries σε αυτή είτε να χρησιμοποιήσω την Rails/Ruby console και να εξετάσω αν τα μοντέλα που έχω φτιάξει για τις οντότητες μου λειτουργούν όπως επιθυμώ .

3.1.5 Rails Console (Ruby Console)

Ίσως το χρησιμότερο εργαλείο του framework RubyOnRails κατά το στάδιο της ανάπτυξης και της συντήρησης μια βάσης δεδομένων . Δίνει την δυνατότητα στον προγραμματιστή , για κάθε οντότητα με τα πεδία της που προσθέτει στη βάση δεδομένων του , να εξετάσει τη συμπεριφορά τους με απλά Ruby scripts αντί να γράφει SQL Queries και να τα δοκιμάζει μέσα στο περιβάλλον της SQLite .

Με την εντολή *rails c(onsole)* μέσα σε κάποιο φάκελο της εφαρμογής από το τερματικό , αυτό μετατρέπεται στην rails console και ο prompt σου δίνει την έκδοση γλώσσας Ruby που χρησιμοποιείται . Εκεί εκτός από τα scripts για τη βάση δεδομένων , μπορούμε να εκτελέσουμε οποιαδήποτε εντολή Ruby όπως ακριβώς κάνουμε στον interpreter (irb) της γλώσσας όταν την εγκαταστήσουμε στο σύστημα μας .

```
2.3.1 :005 > Company.all
  Company Load (2.7ms)  SELECT "companies".* FROM "companies"
 => #<ActiveRecord::Relation []>
2.3.1 :006 > Company
 => Company(id: integer, name: string, created_at: datetime, updated_at: datetime)
2.3.1 :007 >
```

Στο παραπάνω σύντομο παράδειγμα βλέπουμε πως υλοποιούμε σύνδεση με τη βάση και πως απλά χρησιμοποιώντας το όνομα της οντότητας στη βάση μπορούμε να πάρουμε τα πεδία της με τη σειρά που έχουν καταχωρηθεί στη βάση .

Εδώ μπορούμε να δοκιμάσουμε όλες τις βοηθητικές μεθόδους που θα χρησιμοποιήσουμε στον controller της οντότητας όταν βρισκόμαστε στο στάδιο της ανάπτυξης . Δείτε πως η εντολή Company.all που μου εμφανίζει τη λίστα με τις εγγραφές στον πίνακα Company μεταφράζεται στο SQL Query SELECT “companies”.* FROM “companies” .Το παραπάνω script δεν έχουμε παρά να το περάσουμε στον controller της κλάσης Company για να ζητάμε τη λίστα στην έξοδο μέσω του view/companies/index.html.erb[1] .

Μερικά παραδείγματα ακόμα

```
p.works
```

1 Filename.html.erb Αρχεία html με ενσωματωμένο κώδικα ruby μεταξύ των <% %>

```
 Work Load (0.1ms)  SELECT "works".* FROM "works" WHERE "works"."project_id" = ?
[["project_id", 1]]
 => #<ActiveRecord::Associations::CollectionProxy [#<Work id: 9, project_id: 1, worker_id: 4,
datetime: "2017-04-03 12:33:00", created_at: "2017-04-03 12:33:29", updated_at: "2017-04-03
12:33:29", hours: 4, doc: nil>]>
2.3.1 :009 > w = Worker.first
  Worker Load (0.2ms)  SELECT  "workers".* FROM "workers" ORDER BY "workers"."id" ASC
LIMIT ?  [["LIMIT", 1]]
 => #<Worker id: 1, fname: "George", lname: "Andres", company_id: 1, created_at: "2017-03-01
22:47:56", updated_at: "2017-03-01 22:47:56", email: "andgeo@gmail.com",
encrypted_password: "", reset_password_token: nil, reset_password_sent_at: nil,
remember_created_at: nil, sign_in_count: 0, current_sign_in_at: nil, last_sign_in_at: nil,
current_sign_in_ip: nil, last_sign_in_ip: nil>
2.3.1 :010 > w.company
  Company Load (0.1ms)  SELECT  "companies".* FROM "companies" WHERE "companies"."id"
= ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
 => #<Company id: 1, name: "Evil Inc Edited", created_at: "2017-03-01 12:07:33", updated_at:
"2017-04-03 12:38:01", slug: "evil">
2.3.1 :011 > w.projects
  Project Load (0.2ms)  SELECT "projects".* FROM "projects" INNER JOIN "works" ON
"projects"."id" = "works"."project_id" WHERE "works"."worker_id" = ?  [["worker_id", 1]]
 => #<ActiveRecord::Associations::CollectionProxy []>
2.3.1 :012 > Work.recentdays(2)
  Work Load (0.4ms)  SELECT "works".* FROM "works" WHERE (datetime > '2017-04-04
06:06:53 +0300')
 => #<ActiveRecord::Relation []>
2.3.1 :013 >
```

3.1.6 ActiveRecord Model Associations

Για το Rails association είναι η σύνδεση μεταξύ δύο μοντέλων της κλασης ActiveRecord και είναι ιδιαίτερα χρήσιμη λόγους που εξηγεί το παρακάτω παράδειγμα από τα http://guides.rubyonrails.org/association_basics.html

```
class Author < ApplicationRecord
end
class Book <ApplicationRecord
end
```

Δημιουργία βιβλίου και το συσχετίζουμε με κάποιον υπάρχοντα συγγραφέα , αυτόν που έχει το συγκεκριμένο id.

```
@book = Book.create(published_at :Time.now , author_id : @author.id)
@books = Book.where(author_id: @author.id) # Η λίστα από τα βιβλία του συγγραφέα
        @books.each do |b|
                b.destroy
        end
 @author.destroy
```

Η παραπάνω διαδικασία είναι απαραίτητη αν θέλουμε να καταστέψουμε την εγγραφή συγγραφέας στη βάση μας . Πρώτα θα πρέπει να βρούμε όλες τις εγγραφές από βιβλία δικά του , να τα καταστρέψουμε και έπειτα να καταργήσουμε τον ίδιο το συγγραφέα .

Με τα ActiveRecord Associations η διαδικασία αυτή αυτοματοποιείται ως εξής .

```
Class Author <ApplicationRecord
has_many :books , dependant: :destroy
end

Class Book <ApplicationRecord
belongs_to :author
end
```

και τώρα οι παραπάνω ενέργειες πραγματοποιούνται αυτόματα

```
@book = @author.book.create(published_at :Time.now)
@author.destroy
```

Το Rails υποστηρίζει έξι τύπους συσχετισμών μεταξύ των μοντέλων του.

- belongs_to (ένα προς ένα συσχετισμός)
- has_one (ένα προς ένα)
- has_many (ένα προς πολλά)
- has_many :throught (πολλά προς πολλά)
- has_one :throught (ένα προς ένα)
- has_and_belongs_to_many (πολλά προς πολλά)

Στην πραγματικότητα όταν ορίζουμε κάποιο συσχετισμό μεταξύ δύο μοντέλων ,έστω πχ belongs_to διατάζουμε το Rails να κρατάει πληροφορίες τύπου Primary-key , Foreing-key δύο στιγμιοτύπων αντικειμένων των παραπάνω κλάσεων και συνάμα εμπλουτίζουμε τα ίδια τα μοντέλα με μια βιβλιοθήκη βοηθητικών μεθόδων .

Οι κλάσεις μου όπως διαμορφώθηκαν :

```
application_record.rb[2]
class ApplicationRecord < ActiveRecord::Base
  self.abstract_class = true
end
```

```
company.rb
class Company < ActiveRecord::Base
        has_many :users
        has_many :workers
        has_many :projects

        validates :name, length: { minimum: 5 }

        def to_s
                name
        end
end
```

```
 project.rb
class Project < ActiveRecord::Base
        belongs_to :company
        belongs_to :user
        has_many :users, :through => :works
        has_many :works
        has_many :workers, :through => :works
        belongs_to :owner , class_name: "Worker"
```

2 application_record.rb :Κλάση που δημιουργεί το RubyOnRails για κάθε νέα εφαρμογή

```
        validates :name, length: { minimum: 5 }
        validates :company_id, presence: true
        validates :default_rate, numericality: { only_integer: true,}
        validates :owner_id , presence: true
        validates :slug, length: { minimum: 3 }
        validates :slug, uniqueness: true

        scope :lowdefaultrate, -> { where("default_rate < 100") }

        def to_s
                "#{name} (#{company})"
        end
```

```
        def self.export_csv(projects)
                CSV.generate() do |csv|
                        csv << ['name','company','default_rate','created_at','owner','most
recent work item']
                        projects.each do |project|
                                csv <<  [
                                                project.name,
                                                project.company,
                                                project.default_rate,
                                                project.created_at,
                                                project.owner,
                                                project.works.order('created_at
DESC').first
                                                ]
                        end
                end
        end
end
```

```
work.rb
class Work < ActiveRecord::Base
        belongs_to :project
        belongs_to :worker
        validates :project_id, presence: true
        validates :worker_id, presence: true
        validates :datetime, presence: true
        validate :date_is_in_past
        validates :hours, numericality: { only_integer: true, greater_than: 0,
                                                less_than_or_equal_to: 8 }

        scope :fullday, -> { where("hours >= 8") }
        scope :recent, -> { where("datetime > '#{Time.now - 7.days}'")}

        def self.recentdays(numdaysago)
```

```
                        since_date = Time.now - numdaysago.to_i.days
                        where("datetime > '#{since_date}'")
                end
                def date_is_in_past
                        if datetime.present? && datetime > Time.now
                                errors.add(:datetime, "can't be in the future")
                        end
                end
                def to_s
                        "#{worker}: #{datetime.strftime('%m/%d/%Y %H:%M')} - #{hours} hours"
                end
end
```

```
user.rb
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
        :recoverable, :rememberable, :trackable, :validatable
                belongs_to :company
                has_many :works
                has_many :projects, :through => :works
                has_many :projects_owned, :foreign_key => 'owner_id', :class_name => 'Project'
                validates :username, length: { minimum: 2 }
                validates :company, presence: true
                def to_s
                        "#{username} #{email}"
                end
end
```

```
worker.rb
class Worker < ApplicationRecord
                belongs_to :company
                has_many :works
                has_many :projects, :through => :works
                has_many :projects_owned, :foreign_key => 'owner_id', :class_name => 'Project'

                def to_s
                        "#{fname} #{lname}"
                end
end
```

3.1.7 Validations και scopes για πεδία της κλασης

Αρχικά θα πρέπει να εξηγήσω τί είναι τα validations και που χρησιμοποιούνται .Με απλά λόγια είναι ελεγκτικοί μηχανισμοί για τα πεδία της κλάσης όπως για παράδειγμα αν μια τιμή βρίσκεται σε κάποια αποδεκτά όρια ή την επιβολή της παρουσίας της τιμής υποχρεωτικά σε κάποιο πεδίο της κλάσης . Δηλαδή αν δεν θέλω να επιτρέπω εγγραφές activerecord με null τιμές θα πρέπει να το ορίζω στο μοντέλο ώστε να μην γίνεται αποδεκτή μια εγγραφή τέτοιου τύπου .

Εκτός από τα πεδία της κλάσης μπορώ επίσης να ορίσω validates για βοηθητικές μεταβλητές αναφοράς στην κλάση όπως είναι η date_is_in_the_past .

Παραδείγματα :

```
validates :company_id , presence: true
validates  :date_is_in_the_past

    def         date_is_in_the_past
                    if datetime.present?  &&  datetime > Time.now
                    errors.add(:datetime , ‘ Can’t be in the future’)
    end
```

Το παραπάνω παράδειγμα είναι από την κλάση work και ελέγχει κατά την δημιουργία εγγραφής work item στη βάση αν θα γίνει αποδεκτή η εγγραφή , διαφορετικά θα την απορρίψει και παράλληλα θα δώσει στον χρήστη το μήνυμα λάθους “Can’t be in the future” .

Τα μηνύματα λάθους που επιστρέφει η βάση στο χρήστη μπορούμε να τα δούμε από την rails console είτε επιχειρώντας να κάνουμε νέα εγγραφή που δεν συμμορφώνεται με το μοντέλο της , είτε ρωτώντας την βάση αν το στιγμιότυπο του αντικείμενου που θέλουμε να αποθηκεύσουμε είναι αποδεκτό .

Παράδειγμα :

```
 2.3.1 :027> w = Work.new
 => #<Work id: nil, project_id: nil, worker_id: nil, datetime: nil, created_at: nil, updated_at: nil,
hours: nil, doc: nil>
2.3.1 :028 > w.datetime = '2020-2-19 10:10:03'
 => "2020-2-19 10:10:03"
2.3.1 :029 > w.save
   (0.1ms)  begin transaction
   (0.1ms)  rollback transaction
 => false
2.3.1 :030 > w.valid?
 => false
```

2.3.1 :031 > w.errors
=> #<ActiveModel::Errors:0x007fd6443bbec0 @base=#<Work id: nil, project_id: nil, worker_id: nil, datetime: "2020-02-19 10:10:03", created_at: nil, updated_at: nil, hours: nil, doc: nil>, @messages={:project=>["must exist"], :worker=>["must exist"], :project_id=>["can't be blank"], :worker_id=>["can't be blank"], ***:datetime=>["can't be in the future"]****, :hours=>["is not a number"]}, @details={:project=>[{:error=>:blank}], :worker=>[{:error=>:blank}], :project_id=>[{:error=>:blank}], :worker_id=>[{:error=>:blank}], :datetime=>[{:error=>"can't be in the future"}], :hours=>[{:error=>:not_a_number, :value=>nil}]}>*

Με ανάλογο τρόπο μπορώ να ορίσω scopes για κάποια μεταβλητή
scope :fullday, -> {where(“hours >=8”)}
Εδώ το fullday ορίζεται σαν μεταβλητή αναφοράς σε κάποια μέθοδο και θα λειτουργεί σαν τέτοια δηλαδή από rails κονσόλα μπορώ να καλέσω την εντολή
Work.fullday (Enter) και να πάρω τ’ αποτελέσματα που απαντούν το ερώτημα μου .
όμοια
scope :lowdefaultrate, -> {where(“default_rate < 100”)}
Πρόκειται για actions (ενέργειες) , στην πραγματικότητα Ruby methods που κάνουν μια εργασία κατά κανόνα ελέχου .

3.1.8 Controllers (M-V-C)

Οι controllers έχουν το ρόλο του ενδιάμεσου μεταξύ της βάσης δεδομένων (αφού αντλούν τα δεδομένα από το model) και παρουσιάζουν τα αποτελέσματα του χρήστη αφού ορίζουν τον τρόπο και τις παραμέτρους που θα εμφανιστούν στα “views” σε κάποιο για παράδειγμα http request του χρήστη . Ανάλογα με το είδος του request ο controller αντιδρά και διαφορετικά .
Μπορεί για παράδειγμα να στέλνει δεδομένα στη βάση με ένα post request μέσω μια φόρμας .
Σε συμβατικές περιπτώσεις Restful εφαρμογών , ο controller θα λάβει το request από το χρήστη , θα φέρει ή θ’ αποθηκεύσει δεδομένα και θα χρησιμοποιήσει ένα αρχείο view για να κάνει render μια σελίδα HTML για να παρουσιάσει τ’ αποτελέσματα στο χρήστη . Για μια τυπική Resful εφαρμογή θα πρέπει να έχει τουλάχστον τέσσερις μεθόδους CRUD[3] για την επικοινωνία με την κλάση , της οποίας το όνομα φέρει .
Για τους παραπάνω λόγους θεωρείται ενδιάμεσος μεταξύ model και view

3 CRUD : Αρχικά των ενεργειών CREATE , READ , UPDATE , DELETE

```
rails g(enerate) controller companies
Running via Spring preloader in process 12640
      create  app/controllers/companies_controller.rb
      invoke  erb
      create    app/views/companies
      invoke  test_unit
      create    test/controllers/companies_controller_test.rb
      invoke  helper
      create    app/helpers/companies_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/companies.coffee
      invoke    scss
      create      app/assets/stylesheets/companies.scss
```

companies_controller.rb

```
class CompaniesController < ApplicationController
        def index
                @companies = Company.all
        end
        def show
                @company = Company.find(params[:id])
                respond_to do |format|
                        format.html # show.html.erb
                        format.xml  { render xml: @company}
                        format.json { render json: @company}
                end
        end

        def new
                @company = Company.new
        end
        def create
                @company = Company.new(params[:company].permit(:name))
                if @company.save
                        flash[:notice] = 'New Company Created'
```

```
                        redirect_to @company
                else
                        render 'new'
                end
        end

        def edit
                @company = Company.find(params[:id])
        end
        def update
                @company = Company.find(params[:id])

                if @company.update(params[:company].permit(:name))
                        flash[:notice] = 'Company Updated'
                        redirect_to @company
                else
                        render 'edit'
                end
        end
end
```

projects_controller.rb

```
class ProjectsController < ApplicationController

        #before_filter :authenticate_user!
        def index
                @projects = Project.all
                respond_to do |format|
                format.html
                format.csv{send_data Project.export_csv(@projects), type: 'text/csv;
charset=utf-8; header=present', disposition: 'attachment;
        filename=contacts.csv'}
                end
        end

        def show
                if (params[:slug])
                        @project = Project.find_by slug: params[:slug]
                else
                        @project = Project.find(params[:id])
                end
                @work = Work.new
                @work.project = @project
        end

        def new
                @project = Project.new
        end
```

```
        def create
                @project = Project.new(params[:project].permit(:name, :slug,
:company_id, :default_rate))
                if @project.save
                        flash[:notice] = 'Project Created'
                        redirect_to @project
                else
                        render 'new'
                end
        end

        def edit
                @project = Project.find(params[:id])
        end

        def update
                @project = Project.find(params[:id])

                if @project.update(params[:project].permit(:name, :slug, :company_id,
:default_rate))
                        redirect_to @project
                else
                        render 'edit'
                end
        end
end
```

works_controller.rb

```
class WorksController < ApplicationController
        def index
                if (params[:days])
                        @works = Work.recentdays(params[:days]).order('datetime
desc')
                else
                        @works = Work.all.order('datetime desc')
                end
        end

        def show
                @work = Work.find(params[:id])
        end

        def new
                @work = Work.new
        end

        def create
                @work = Work.new(params[:work].permit(:project_id, :worker_id,
:datetime, :hours))
```

```
                    if params[:doc]
                    uploaded_io = params[:doc]

        File.open(Rails.root.join('public','uploads',uploaded_io.original_filename),'wb') do |
file|
                            file.write(uploaded_io.read)
                            @work.doc = uploaded_io.original_filename
                            end
                    end
                    respond_to do |format|
                            if @work.save
                                    format.html { redirect_to @work, notice: 'Work
Created' }
                                    format.js { }
                            else
                                    format.html { render 'new' }
                                    format.js { }
                            end
                    end
        end

        def edit
                    @work = Work.find(params[:id])
        end

        def update
                    @work = Work.find(params[:id])

                    if @work.update(params[:work].permit(:project_id, :worker_id,
:datetime, :hours))
                            flash[:notice] = 'Work Updated'
                            redirect_to @work
                    else
                            render 'edit'
                    end
        end
end
```

worker_controller.rb

```
class WorkersController < ApplicationController
        def index
        @workers = Worker.all
        end
        def show
        @worker = Worker.find(params[:id])
        def new
```

```
            @worker = Worker.new
            end
            def create
            end
            def update
            end
            def delete
            end

            end
end
```

Πριν προχωρήσουμε στα views του MVC θα πρέπει να αναφερθούμε στη σημασία του αρχείου config/routes.rb , τη χρησιμότητα του , τον τρόπο που αυτό σχετίζεται με τα models και τους controllers και πώς όλα τα παραπάνω δημιουργούν τη διεπαφή με τον τελικό χρήστη μέσω των views .

Τι είναι λοιπόν το αρχείο routes.rb ; Όπως καταλαβαίνουμε από την κατάληξη του αρχείου , πρόκειται για ακόμα ένα αρχείο γραμμένο στη γλώσσα ruby και δημιουργείται από το Rails με την δημιουργία της εφαρμογής μέσα στο φάκελο config .

Ο σκοπός της ύπαρξης του είναι ν' αναγνωρίζει διευθύνσεις URLs και να τις αποστείλει σε κάποια από τις μεθόδους του controller . Επίσης δημιουργεί μονοπάτια και διευθύνσεις URLs για την εφαρμογή μας , ώστε να μην είναι αναγκαία η χρήση hardcoded διευθύνσεων στα views αρχεία μας.

Με τη δημιουργία κάποιου controller , έστω του projects , απαιτείται η προσθήκη κάποιου resource με την προσθήκη της παρακάτω γραμμής για να είναι πλέον γνωστή (ανακοινώσιμη) η τοποθεσία της . Τώρα με ένα οποιοδήποτε browser είναι προσβάσιμη η πηγή projects http://localhost:3000/projects προς τον χρήστη .

Το σύνολο των διαφορετικών ενεργειών και τοποθεσιών αυτής και μόνο της εγγραφής μπορούμε να το δούμε όταν στην κονσόλα δώσουμε την εντολή

rake routes .

```
rake routes
                Prefix Verb   URI Pattern                  Controller#Action
      new_user_session GET    /users/sign_in(.:format)     devise/sessions#new
          user_session POST   /users/sign_in(.:format)     devise/sessions#create
  destroy_user_session DELETE /users/sign_out(.:format)    devise/sessions#destroy
     new_user_password GET    /users/password/new(.:format) devise/passwords#new
    edit_user_password GET    /users/password/edit(.:format) devise/passwords#edit
```

```
    user_password PATCH  /users/password(.:format)      devise/passwords#update
                  PUT    /users/password(.:format)      devise/passwords#update
                  POST   /users/password(.:format)      devise/passwords#create
cancel_user_registration GET    /users/cancel(.:format)       devise/registrations#cancel
 new_user_registration GET    /users/sign_up(.:format)      devise/registrations#new
edit_user_registration GET    /users/edit(.:format)         devise/registrations#edit
    user_registration PATCH  /users(.:format)              devise/registrations#update
                  PUT    /users(.:format)               devise/registrations#update
                  DELETE /users(.:format)               devise/registrations#destroy
                  POST   /users(.:format)               devise/registrations#create
         companies GET    /companies(.:format)          companies#index
                  POST   /companies(.:format)          companies#create
       new_company GET    /companies/new(.:format)      companies#new
      edit_company GET    /companies/:id/edit(.:format) companies#edit
           company GET    /companies/:id(.:format)      companies#show
                  PATCH  /companies/:id(.:format)      companies#update
                  PUT    /companies/:id(.:format)      companies#update
                  DELETE /companies/:id(.:format)      companies#destroy
          workers GET    /workers(.:format)            workers#index
                  POST   /workers(.:format)            workers#create
       new_worker GET    /workers/new(.:format)        workers#new
      edit_worker GET    /workers/:id/edit(.:format)   workers#edit
           worker GET    /workers/:id(.:format)        workers#show
                  PATCH  /workers/:id(.:format)        workers#update
                  PUT    /workers/:id(.:format)        workers#update
                  DELETE /workers/:id(.:format)        workers#destroy
           works GET    /works(.:format)              works#index
                  POST   /works(.:format)              works#create
        new_work GET    /works/new(.:format)          works#new
       edit_work GET    /works/:id/edit(.:format)     works#edit
            work GET    /works/:id(.:format)          works#show
                  PATCH  /works/:id(.:format)          works#update
                  PUT    /works/:id(.:format)          works#update
                  DELETE /works/:id(.:format)          works#destroy
```

```
    projects GET    /projects(.:format)          projects#index
             POST   /projects(.:format)          projects#create
 new_project GET    /projects/new(.:format)      projects#new
edit_project GET    /projects/:id/edit(.:format) projects#edit
     project GET    /projects/:id(.:format)      projects#show
             PATCH  /projects/:id(.:format)      projects#update
             PUT    /projects/:id(.:format)      projects#update
             DELETE /projects/:id(.:format)      projects#destroy
        root GET    /                           companies#index
             GET    /recentworks/:days(.:format) works#index
             GET    /allprojects/:slug(.:format) projects#show
             GET    /allcompanies/:slug(.:format) companies#show
```

Ας δούμε τη σημασία αυτών των διευθύνσεων και των ενεργειών για ένα από τους controllers μου , έστω για τον projects .

Αριστερά στη λίστα βρίσκεται το path δηλαδή (για τον projects_controller αυτό αντιστοιχεί στη μεταβλητή projects_path και είναι το localhost:3000/projects) .Ακολουθεί το Http Verb που χαρακτηρίζει το είδος του request GET , POST , PATCH , PUT , DELETE , μετά είναι το ακριβές path με τις όποιες παραμέτρους απαιτούνται και τέλος το όνομα του contollername#action με την ενέργεια που θα πρέπει να λάβει χώρα όταν έρθει ένα request αυτού του τύπου .

Πχ στο GET /projects -> θα πρέπει ο projects_controller να εκτελέσει τον κώδικα που βρίσκεται στην μέθοδο/ενέργεια index . Τι θα κάνει render για τον χρήστη ο controller δίνεται και στο action αλλά κυρίως στον κώδικα που περιέχει η index action του . Εκεί θα δούμε ότι επιλέγει να κάνει render την σελίδα index.html.erb που βρίσκεται στο φάκελο views/projects/ και να δώσει τη λίστα με τα projects στον χρήστη .

3.1.9 **Views** (MVC)

Είχαμε μιλήσει νωρίτερα για τη σημασία του controller σαν ενδιάμεσος μεταξύ χρήστη και βάσης δεδομένων . Είπαμε για τα resources/routes από το αρχείο config/routes.rb , για τη σημασία των controllername_path στο αρχείο routes , αναφέραμε τα actions / μεθόδους στον κώδικα του controller και κάναμε μια απλή αναφορά τα views layer που αποτελούν και την διεπαφή μεταξύ χρήστη και εφαρμογής . Ο τρόπος που συνδέονται όλα τα παραπάνω μεταξύ τους , λίγο πολύ έχει εξηγηθεί , αυτό που ίσως απομένει είναι λίγα λόγια για τις τοπικές μεταβλητές σε κάθε controller και πως μπορούμε να τις κάνουμε render στο αντίστοιχο αρχείο views που ορίζεται από την μέθοδο του controller . Ας δούμε μερικά απλά παραδείγματα .

Αρχικά θα μιλήσουμε για το πέρασμα παραμέτρων μέσω του controller με την ανάθεση σε μεταβλητές της μορφής @variable ...

Κάθε μεταβλητή με @όνομα_μεταβλητής είναι προσβάσιμη από το View layer και η εμφάνιση της σε αυτό γίνεται όπως παρακάτω

def show
@hello = 'Say hello to Ruby' // string
end
για να γίνει render από το View layer στο show.html.erb
με την ανάθεση των
<%= @hello %>

Αντίστοιχα για την ενέργεια/μέθοδο index, είναι συνήθες η βασική της λειτουργία , να παρουσιάζει την λίστα των αντικειμένων που βρίσκονται στην βάση δεδομένων . Για παράδειγμα για τις επιχειρήσεις , έχουμε

```
def index
        @companies = Company.all
end
```

Για να γίνει αυτό render στην index.html.erb θα πρέπει να κληθεί με κάποιο τρόπο η μεταβλητή @companies και αυτό γίνεται ως εξής :

```
<ul>
    <% @companies.each do |company| %>
        <li><%= company.name%></li>
    <%end%>
</ul>
```

με την χρήση της βοηθητικής μεθόδου του Rails link_to (String , path)

```
<ul>
    <% @companies.each do |company| %>
    <li><%= link_to company.name, company %></li>
    <%end%>
</ul>
```

με οδηγεί στην σελίδα show με παράμετρο το id της κάθε εταιρίας . Δλδ δημιουργούνται links για την κάθε εταιρεία όπου με κατάλληλη επεξεργασία της σελίδα show θα μπορώ να εμφανίζω πληροφορίες για την εταιρεία που λαμβάνονται από τη βάση δεδομένων όπως το όνομα της κλπ.

Αρχεία views για την κλάση company

index.html.erb

```
<h1>Companies:index</h1>

<p>Here are all of the companies:</p>

<table>
    <tbody>
        <tr>
            <th>Company</th>
            </tr>
            <%= render partial: 'company', collection: @companies %>
        </tbody>
     </table>


    <% content_for :aside do %>
    <% render 'sidebar' %>
    <% end %>
```

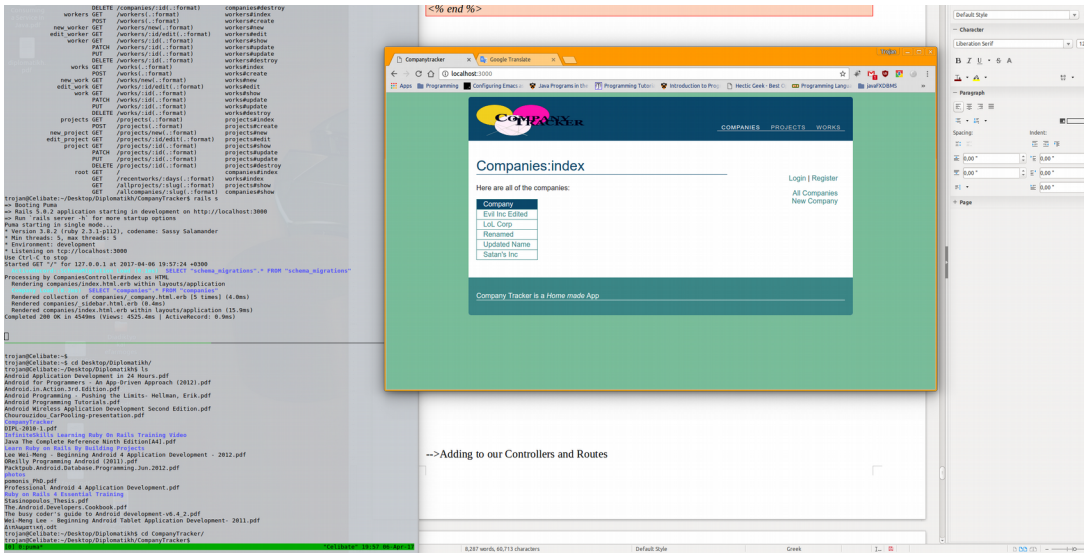
show.html.erb

```
<h1><%= @company.name %></h1>

<p><%= link_to 'Edit Company', edit_company_path(@company) %></p>

<% if @company.projects.size > 0 %>
    <table>
        <tbody>
            <tr>
```

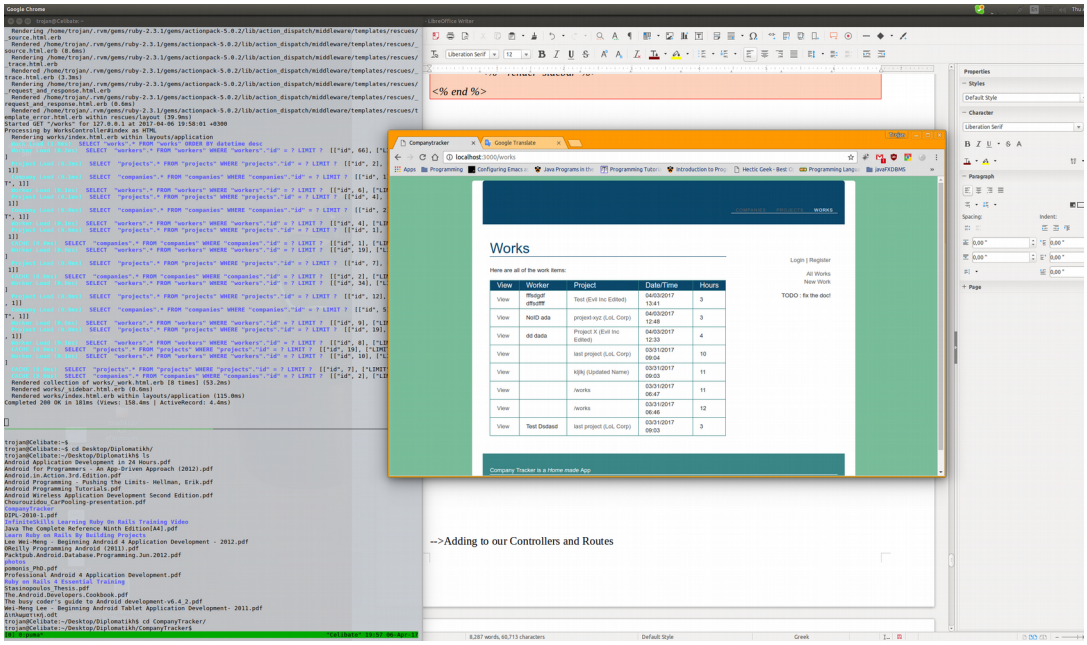
```
                                        <th>Project</th>
                                        <th>Company</th>
                                    </tr>
                                    <%= render partial: "projects/project", collection:
@company.projects %>
                                </tbody>
                        </table>
<% else %>
                        <p>No projects</p>
<% end %>
```

```
_form.html.erb
% if @company.errors.any? %>
                        <div id="errors">
                                <p>Please correct the following <%= pluralize(@company.errors.count,
"error") %>:</p>
                                <ul>
                                        <% @company.errors.full_messages.each do |msg| %>
                                                <li><%= msg %></li>
                                        <% end %>
                                </ul>
                        </div>
<% end %>

<%= form_for(@company) do |f| %>
                        <div>
                                <%= f.label :name %>
                                <%= f.text_field :name %>
                        </div>
                        <div>
                                <%= f.submit @company.new_record? ? "Create Company" : "Update
Company"  %>
                        </div>
<% end %>
<% content_for :aside do %>
                        <%= render 'sidebar' %>
<% end %>
```

Μια τυπική εικόνα της αρχικής σελίδας που δημιουργήθηκε από τα παραπάνω είναι



Εικόνα 3.1.9(α) : Companies Index View

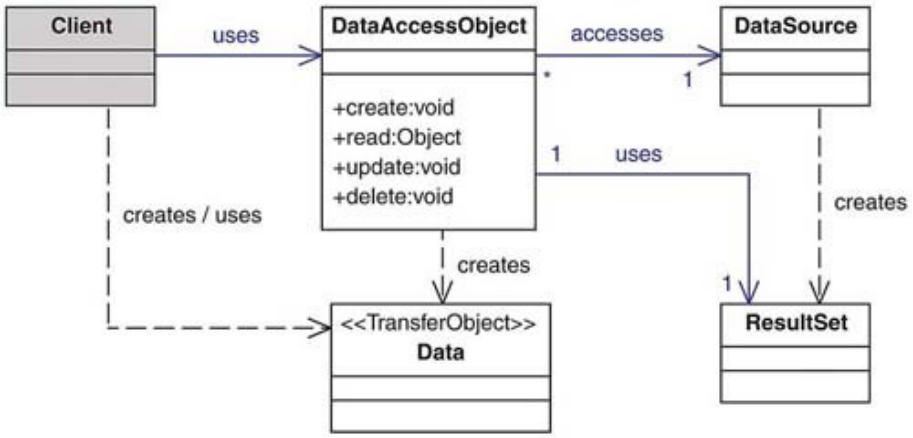


Εικόνα 3.1.9 (β): Works Index View όπου είναι πιο ξεκάθαρα τα μηνύματα του

webserver από τα request του χρήστη στον works_controller

3.2.1 Κατασκευή της διεπαφής χρήστη για την διαχείριση της βάσης δεδομένων

Προσωρινά δεν θ' αναφερθώ εκτενώς στη κατασκευή του java app, θα πω μονάχα ότι για την κατασκευή του θα χρησιμοποιήσουμε ακριβώς το ίδιο μοντέλο δηλαδή MVC μεταξύ java classes και database models . Στο ενδιάμεσο εκτός από τις κλάσεις των αντικειμένων την βάσης που θα αναπαραστήσουν τα models της βάσης θα χρησιμοποιήσουμε για την επικοινωνία με τη βάση δεδομένων data access objects (DAO) java classes που προσφέρουν μια αφηρημένη διεπαφή με τη βάση δεδομένων και παρέχουν τη δυνατότητα να επικοινωνούν με τη βάση και να ενεργούν σε αυτήν χωρίς να εκθέτουν τα ιδιαίτερα χαρακτηριστικά της βάσης . Το βασικό πλεονέκτημα αυτής της τεχνολογίας είναι ότι ξεχωρίζει τα δύο μέρη μια εφαρμογής (UI και bussiness logic) μπορούν και πρέπει να συνεργάζονται αλλά δεν είναι απαραίτητο να γνωρίζει ακριβώς το ένα τις αλλαγές που συντελούνται στο άλλο ,δίνοντας έτσι την δυνατότητα σε αυτά να εξελίσονται ευκολότερα και συχνότερα .



Σχήμα 3.2 : UML Data Access Object

3.2.2 Η εφαρμογή διαχείρισης της βάσης δεδομένων

Όπως και στην περίπτωση του εξυπηρετητή θα περιορίσουμε την ανάλυση σε μία από τις οντότητες της βάσης για συντομία . Όσα ακολουθούν είναι σχεδόν όμοια με κάποιος αλλαγές στον controller ανάλογα με τις ιδιότητες που θέλουμε να παρουσιάζει η κάθε οντότητα χωριστά . Αν υπάρχουν μεγάλες αλλαγές σε κάποια από αυτές θα αναφερθούμε χωριστά .
Πριν προχωρήσουμε σε κάποια από τις οντότητες της βάσης δεδομένων θα πρέπει ν' αναφερθώ στην πορεία που ακολουθείται για να φτάσουμε στην διαχείριση αυτής της οντότητας .

Η εφαρμογή διαχείρισης της βάσης δεδομένων ξεκινάει από την αρχική σελίδα που φορτώνεται από τη "main" της κλάσης Diplomatikh.java .
Κάθε javafx εφαρμογή πρέπει να είναι επέκταση της κλάσης Application . Αυτό μπορούμε το δούμε στην επικεφαλίδα της κλάσης με την οδηγία extends Application .
Με το που τρέχουμε την εφαρμογή , αυτή όπως όλες οι εφαρμογές σε java , αρχίζει από την main μέθοδο που έχει τον εκτελέσιμο κώδικα της και αυτή μας παραπέμπει στο σημείο όπου διαβάζει το αρχείο UserLogin.fxml
Τα fxml αρχεία δεν είναι τίποτα άλλο από ένα δημιούργημα της Oracle βασισμένο στην γλώσσα XML (Extended Markup Language) για να ορίζουν τη διεπαφή των εφαρμογών javafx . Μπορείς έτσι να δημιουργήσεις ολόκληρη τη διεπαφή χρήστη σε ένα αρχείο και να το φορτώσει ο μεταγλωτιστής της java στο User Interface που έχουμε ορίσει στο αρχείο .
Η πορεία που ακολουθείται είναι κοινή για όλες τις εφαρμογές .Με το που θα δηλωθεί μία διεπαφή σε κάποιο αρχείο με κατάληξη fxml χρησιμοποιώντας μία κλάση της java που ονομάζεται FXMLLoader , αυτή αναλαβάνει να κάνει δύο πράγματα ταυτόχρονα . Αρχικά φορτώνει αρχικοποιεί όλα τα αντικείμενα που έχουν οριστεί σε αυτό το αρχείο και που έχουν δηλωθεί στον controller της οντότητας με την οδηγία @FXML , ώστε να είναι ορατά και διαχειρίσιμα από την εφαρμογή κατά τη συνεδρία που θα είναι φορτωμένη η εφαρμογή και συνδέει τον controller που έχουμε ορίσει για την οντότητα με την ίδια την οντότητα . Συνεπώς κάθε οδηγία για μια νέα διεπαφή χρήστη (κοινώς παράθυρο) κάνει ακριβώς τις δύο παραπάνω εργασίες .

Diplomatikh.java

```
package diplomatikh;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */


import javafx.application.Application;
import javafx.scene.Parent;
import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 *
 * @author trojan
 */
public class Diplomatikh extends Application {

    @Override
    public void start(Stage primaryStage) {

        try{

            Parent root = FXMLLoader.load(getClass().getResource("/user/UserLogin.fxml"));
            Scene scene = new Scene(root,800,600);

scene.getStylesheets().add(getClass().getResource("/user/userlogin.css").toExternalForm());




        primaryStage.setScene(scene);
        primaryStage.setTitle("Connect to RoR Ryby db!");
        primaryStage.setScene(scene);
        primaryStage.show();
         }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
```

```
    }

}
```

Στη συνέχεια , με την οδηγία

Parent root = FXMLLoader.load(getClass().getResource("/user/UserLogin.fxml"));

φορτώνουμε τη διεπαφή σύνδεσης χρήστη με την βάση δεδομένων και θέτουμε σε ισχύ τον αντίστοιχο controller UserController.java απ' όπου με κάποιο action event στη μέθοδο mainUI(ActionEvent event) μπορούμενα φορτώσουμε τη διεπαφή MainFXML.fxml

```
package user;

import diplomatikh.LoginModel;
import dbtools.SqliteConnection;
import java.io.IOException;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author trojan
 */



public class UserController implements Initializable {

    Connection con ;
```

```
    private LoginModel loginModel = new LoginModel();


    @FXML
    private Button btnLogin;
    @FXML
    private Button btnDataBaseUI;
    @FXML
    private TextField txtUsername;
    @FXML
    private TextField txtEmail;
    @FXML
    private Label lblStatus;
    @FXML
    private Button btnBrowser;


    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
        con = SqliteConnection.connector();

        btnDataBaseUI.setDisable(true);
                }



    @FXML
    private void mainUI(ActionEvent event) throws  IOException{

        try {


                Stage mainStage = new Stage();
                FXMLLoader fxmlLoader = new FXMLLoader();

                Parent root =
fxmlLoader.load(getClass().getResource("/diplomatikh/MainFXML.fxml"));

                System.out.println("after the controller launched ...");
                Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/diplomatikh/mainfxml.css").toExternalForm())
;
                mainStage.setScene(scene);
                mainStage.setTitle("Database Control Center");
                mainStage.setScene(scene);
```

```
            mainStage.show();
        ((Node)(event.getSource())).getScene().getWindow().hide();



      } catch (IOException e) {
          Logger.getLogger(UserController.class.getName()).log(Level.SEVERE, null,e);
      }
   }

   @FXML
    public void login(ActionEvent event)  throws SQLException
    {
     //Connection con = SqliteConnection.connector();

     PreparedStatement statement ;
     String query = "select * from users where username = ? and email= ?";
     ResultSet resultSet;

     try {
        statement = con.prepareStatement(query);
        statement.setString(1, txtUsername.getText());
        statement.setString(2, txtEmail.getText());
        resultSet = statement.executeQuery();
//         while(resultSet.next()){
//             System.out.println("username ="+resultSet.getString("username")
+"encrypted_password = "+ resultSet.getString("encrypted_password")+"\n");
//         }

        while(resultSet.next()){
           if(resultSet.getString(2)!=null  && resultSet.getString(6) != null)
           {
              //System.out.println("User "+txtUsername.getText()+ " connected !");
              lblStatus.setText("User "+txtUsername.getText()+ " connected !");

              btnDataBaseUI.setDisable(false);
           }
           else
           {
            txtUsername.clear();
            txtEmail.clear();
             lblStatus.setText("Sorry, try again");
           }
          }

        }
     catch (SQLException e)
          {
           e.printStackTrace();
          }
```

```
    }
    @FXML
    private void browseDatabase(ActionEvent ev) throws IOException{
      // WebEngine engine = new WebEngine();
       try {


            Stage webStage = new Stage();
            FXMLLoader fxmlLoader = new FXMLLoader();
           // WebView wv = new WebView();
            Parent root = fxmlLoader.load(getClass().getResource("/browser/browser.fxml"));
            System.out.println("after the controller launched ...");
           // BrowserController mybrowser = (BrowserController)fxmlLoader.getController();

            Scene scene = new Scene(root);



scene.getStylesheets().add(getClass().getResource("/browser/browser.css").toExternalForm());
            webStage.setScene(scene);
            webStage.setTitle("Database WebView");
            webStage.setScene(scene);
            webStage.show();
         //((Node)(event.getSource())).getScene().getWindow().hide();



       } catch (IOException e) {
          Logger.getLogger(UserController.class.getName()).log(Level.SEVERE, null,e);
//       }

       }


   }
    @FXML
    private void exitSystem(ActionEvent e){
       System.exit(0);
    }


}
```

Αφού πραγματοποιήσουμε επιτυχή σύνδεση με τη βάση και ο χρήστης επικυρωθεί για την εγκυρότητα του , μπορούμε να ανοίξουμε την κεντρική διεπαφή διαχείρισης της βάσης δεδομένων με το όνομα MainFXML.fxml και τον αντίστοιχο controller MainFXMLController.java , απ' όπου αποκτούμε πρόσβαση σε όλες της οντότητες που περιέχονται στη βάση δεδομένων για τη ανάγνωση ή τη διαχείριση τους .

Η κλάση MainFXMLController.java

```
package diplomatikh;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import javafx.event.ActionEvent;
import java.net.URL;
import java.sql.SQLException;
import java.time.ZonedDateTime;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author trojan
 */
public class MainFXMLController implements Initializable {

    /**
     * Initializes the controller class.
```

```
 */
public LoginModel loginModel = new LoginModel();

@FXML
private Label isConnected;

@FXML
private TextField txtfName;

@FXML
private TextField txtlName;

@FXML
private TextField txtPassword;

@FXML
private Button loginButton;

@FXML
private Button btnCompanies;

@FXML
private Button btnProjects;

@FXML
private Button btnWorkers;

@FXML
private Button btnWorks;
@FXML
private TextArea txtSystemOUT;

@FXML
private PrintStream ps;

@Override
public void initialize(URL url, ResourceBundle rb) {



    if (loginModel.isDbConnected()) {
        isConnected.setText("Connected");
    } else {
        isConnected.setText("Not Connected");
    }
    // Console Code Test
    ps = new PrintStream(new Console(txtSystemOUT));
    System.setOut(ps);
    System.setErr(ps);
    System.out.println("Hello you new console now is on the database  main UI");
```

```
    }

    public void Login(ActionEvent event) {
        try {
            if (loginModel.userLogin(txtfName.getText(), txtlName.getText(), txtPassword.getText()))
                    {
                isConnected.setText("username " + txtfName.getText() + "\nemail" + txtlName.getText()
+ "\n logged in at " + ZonedDateTime.now().toString());
                //((Node)event.getSource()).getScene().getWindow().hide();
            } else {
                isConnected.setText("The User does not exist");
            }
        } catch (SQLException ex) {
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null, ex);
            isConnected.setText("No user found");
            ex.printStackTrace();
        }
    }

    public void companiesTableView(ActionEvent event) throws  Exception{
        try {

            if (loginModel.isDbConnected()) {
                //TODO
                //perimene na eleg8ei giati an getWindows().hide() mporei na mpo kleinei to connection
me thn db 8a to doume ...
                // ((Node)event.getSource()).getScene().getWindow().hide();
                //Parent root;
                Stage companiesStage = new Stage();
                FXMLLoader fxmlLoader = new FXMLLoader();
                System.out.println("before companies launch ... ");
                Parent root =
fxmlLoader.load(getClass().getResource("/company/Companies.fxml").openStream());
               // CompaniesController companiesController = (CompaniesController)
fxmlLoader.getController();
                System.out.println("after the controler launch ...");
                Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/company/companies.css").toExternalForm());

                companiesStage.setScene(scene);
                companiesStage.setTitle("Connected to Companies Table");
                companiesStage.setScene(scene);
                companiesStage.show();
            } else {
                System.out.println("You r not connected to the database.");
            }

        } catch (Exception e) {
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null,e);
```

```
        }
    }
//ok
    public void workersTableView(ActionEvent event) throws SQLException,
ClassNotFoundException, IOException {
        try {
            System.out.println("Button Workers pressed");

            if (loginModel.isDbConnected()) {

                Stage workersStage = new Stage();

                FXMLLoader fxmlLoader = new FXMLLoader();

                Parent root =
fxmlLoader.load(getClass().getResource("/worker/worker.fxml").openStream());
                //  worker.WorkerController workerController = (WorkerController)
fxmlLoader.getController();

                System.out.println("Just passed the getResource worker.fxml");
                Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/worker/worker.css").toExternalForm());
                workersStage.setScene(scene);
                workersStage.setTitle("Connected to Workers Table");
                workersStage.setScene(scene);
                workersStage.show();
            } else {
                System.out.println("You r not connected to the database.");
            }

        } catch (IOException e) {
            System.out.println("Exception occured while trying to connect to the database..." + e);
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE,null,e);

        }

    }
    // end OK

    @FXML
    public void worksTableView(ActionEvent event) throws
SQLException,ClassNotFoundException,IOException
    {
        try {
            System.out.println("Button Works pressed");

            if (loginModel.isDbConnected()) {
```

```
            Stage worksStage = new Stage();
            FXMLLoader fxmlLoader = new FXMLLoader();
            Parent root =
fxmlLoader.load(getClass().getResource("/work/works.fxml").openStream());

            //WorkerController workerController = fxmlLoader.getController();

           // Parent root = fxmlLoader.load(getClass().getResource("works/works.fxml"));
           // WorkerController workerController = (WorkerController) fxmlLoader.getController();

           // System.out.println("Just passed the getResource work/works.fxml");
            Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/work/works.css").toExternalForm());
            worksStage.setScene(scene);
            worksStage.setTitle("Connected to Works Table");
            worksStage.setScene(scene);
            worksStage.show();
          } else {
            System.out.println("You r not connected to the database.");
          }

       } catch (IOException ex) {
          Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null, ex);
       }
    }
//ok
    /**
     *
     * @param event
     * @throws java.io.IOException
     * @throws java.lang.ClassNotFoundException
     */
    @FXML
    public void projectsTableView(ActionEvent event) throws IOException ,
ClassNotFoundException
    {
       try {
          System.out.println("Button Projects pressed");

          if (loginModel.isDbConnected()) {
            Stage projectsStage = new Stage();
            FXMLLoader fxmlLoader = new FXMLLoader();

            Parent root =
fxmlLoader.load(getClass().getResource("/project/projects.fxml").openStream());
            //  worker.WorkerController workerController = (WorkerController)
fxmlLoader.getController();

           //  System.out.println("Just passed the getResource ptojects.fxml");
```

```
            Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/project/projects.css").toExternalForm());

            projectsStage.setScene(scene);
            projectsStage.setTitle("Connected to Projects Table");
            projectsStage.setScene(scene);
            projectsStage.show();
        } else {
            System.out.println("You r not connected to the database.");
        }

    } catch (IOException e) {
        System.out.println("Exception occured while trying to connect to the database..." + e);
        Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE,null,e);

    }
  }
  @FXML
  private Button btnEXIT;
  @FXML
  private void systemExit(ActionEvent e){
    System.exit(0);
  }

  //Console Code continued

  public class Console extends OutputStream{

    private final TextArea console;

    public Console(TextArea console){
      this.console=console;

    }
    public void appendText(String valueOf){
      Platform.runLater(()->console.appendText(valueOf));
    }

    @Override
    public void write(int b) throws IOException {

      appendText(String.valueOf((char)b));

    }

  }
}
```

Από το κύρια διεπαφή μεταξύ του χρήστη της εφαρμογής και της βάσης δεδομένων , αν αναλύσουμε τον κώδικα διεξοδικά , θα δούμε ότι έχουμε πρόσβαση σε όλες τις οντότητες της βάσης δεδομένων για να τις διαχειριστούμε .

3.2.3 Η κλάση Company.java

Άρχικά ξεκινάμε την κατασκευή της οντότητας μας έχοντας πάντα στο νου μας το database scheme που μας δίνει μια εποπτική εικόνα από τα αντικείμενα που θέλουμε να κατασκευάσουμε . Όταν για παράδειγμα θέλουμε να κάνουμε μια νέα εισαγωγή στη βάση δεδομένων ,(μια νέα εγγραφή) θα πρέπει να κατασκευάσουμε ένα στιγμιότυπο αντικείμενου αυτής της κλάσης και κατόπιν να το κάνουμε εισαγωγή στη βάση δεδομένων . Επίσης όταν πρόκειται να διαβάσουμε τα περιεχόμενα της βάσης δεδομένων και θέλουμε να εμφανίζονται τα στοιχεία του κάθε αντικειμένου σε κάποιους πίνακες στην εφαρμογή μας , θα πρέπει να έχουμε μια αναλογία ένα προς ένα για τα πεδία του αντικειμένου με τις στήλες του πίνακα ώστε να παρέχουμε όσο το δυνατό πληρέστερη πληροφορία για το στιγμιότυπο ενός αντικειμένου .

Έτσι προχωράμε στη κατασκευή της πρώτης μας κλάσης που αναφέρεται σε αντικείμενα τύπου Company .

Για την κατασκευή του φτιάχουμε ένα αρχείο java , με το όνομα του αντικειμένου/ τύπου όπου τα επιμέρους χαρακτηριστικά του θα εμφανίζονται σαν πεδία της κλάσης . Χρησιμοποιούμε οποιονδήποτε editor και δημιουργούμε το αρχείο Company.java

Τυπικά υπάρχουν δύο τεχνικές για τον κατασκευαστή αυτής της κλάσης .

Η πρώτη είναι να πάρουμε ένα προς ένα όλα τα πεδία της οντότητας από την βάση και να έχουμε ένα κατασκευαστή που θα περιέχει τα πάντα κατά τη δημιουργία του αντικειμένου με τους όποιους περιορισμούς προϋποθέτει το σχήμα της βάσης . Παράδειγμα

από το schema.rb αρχείο της βάσης για την οντότητα companies με την εντολή cat schema.rb έχω :

```
create_table "companies", force: :cascade do |t|
  t.string   "name"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.string   "slug"
end
```

Αν προσθέσω και τα πεδία που βάζει αυτόματα το framework κατά τη κατασκευή της οντότητας στη βάση θα έχω ακόμα ένα πεδίο , στην θέση 1 για το id . Στην πραγματικότητα μπορώ να δω αυτή τη λεπτομέρεια αν χρησιμοποιήσω την rails console απλά χρησιμοποιώντας την εντολή

Company :

```
2.3.1 :005 > Company
 => Company(id: integer, name: string, created_at: datetime, updated_at: datetime, slug: string)
2.3.1 :006 >
```

Συνεπώς ο κατασκευαστής μου θα πρέπει να περιέχει πεδία για όλες τις παραπάνω τιμές του αντικειμένου .

Έτσι έχω τον κατασκευαστή της κλάσης Company :

```
public class Company {

    private StringProperty name;
    private IntegerProperty id;
    private StringProperty created_at;
    private StringProperty updated_at;
    private StringProperty slug;


    public Company(){
        this.id = new SimpleIntegerProperty();
        this.name = new SimpleStringProperty();
        this.slug = new SimpleStringProperty();
        this.created_at = new SimpleStringProperty();
        this.updated_at = new SimpleStringProperty();
    }
```

Μια παρατήρηση εδώ για το γεγονός ότι θα περιμέναμε να έχουμε String και integer για τα πεδία της κλάσης και στην πραγματικότητα έχουμε StringProperty και IntegerProperty αντίστοιχα . Αυτό έγινε γιατί τα στοιχεία του όποιου στιγμιοτύπου της οντότητας θα εμφανίζονται σε μια Observable List σε κάποιο πίνακα της εφαρμογής . Για να μπορούμε να τα διαχειριστούμε δυναμικά από τον πίνακα , θα πρέπει να είναι τύπου StringProperty και IntegerProperty αντίστοιχα και όχι απλά String και integer όπως θα περιμέναμε . Αυτά για την κατασκευή του αντικειμένου και εμφάνιση του στο TableView της εφαρμογής . Στην περίπτωση όμως που επιθυμούμε μια εγγραφή στη βάση δεδομένων αυτά τα πεδία θα πρέπει να τροποιηθούν στις πραγματικές τιμές της βάσης , συνεπώς θα πρέπει να μπορώ να παίρνω την τιμή String / integer από το περιτύλιγμα και να κάνω την εγγραφή μου στη βάση . Έτσι θα πρέπει να έχω getters που θα επιστρέφουν αυτούς τους τύπους που απαιτεί το σχήμα της βάσης .

```
public String getName(){
        return name.get();
    }
    public int getId(){
```

```
        return id.get();
    }
    public String getCreatedAt(){
        return created_at.get();
    }
    public String getUpdatedAt(){
        return updated_at.get();
    }
    public String getSlug(){
        return slug.get();
    }
```

Ακολουθούν οι υπόλοιπες μέθοδοι της κλάσης Company.

```
public void setName(String newName){
        name.set(newName);
    }
//    public void setId(int newId){
//        id.set(newID);
//    }
    public void updatedAt(){
    //    zdt = ZonedDateTime.now();
        updated_at.set(ZonedDateTime.now().toString());

    }
    public void setSlug(String newSlug){
        slug.set(newSlug);
    }

    public void setCreated(){
        created_at.set(ZonedDateTime.now().toString());
    }
    public void setUpdated(String updated)
    {
        updated_at.set(updated);
    }

    public void setId(int id){
        this.id.set(id);
    }
```

Η δεύτερη τεχνική είναι να έχουμε ένα κατασκευαστή αντικειμένων που θα περιορίζεται στα πεδία της οντότητας που απαιτούνται για την εισαγωγή στη βάση δεδομένων και μόνο . Δηλαδή θα μπορούσα να παραλείψω το πεδίο created_at και ν αφήσω τη συμπλήρωση του από την ίδια την βάση δεδομένων . Κάθε φορά που κατασκευάζω ένα αντικείμενο της κλάσης αυτής για να το εισάγω στη βάση , μπορώ ν' αδιαφορήσω για την τιμή του πεδίου created_at και να αφήσω τη βάση να του δώσει τιμή κατά την εισαγωγή . Από το schema.rb βλέπω ότι οι τιμές created_at και

updated_at δεν μπορεί να είναι null . Επίσης μπορώ να διαπιστώσω από την ruby console ποια εισαγωγή στον πίνακα companies θεωρείται valid και ποια όχι .Με βάση αυτόν τον οδηγό θα μπορούσα να έχω τον κατασκευστή της κάθε κλάσης μου . Στην περίπτωση που κάποιο αντικείμενο δεν θα μπορούσε να θεωρηθεί “valid” από τη βάση δεδομένων , στην κονσόλα της εφαρμογής θα μου επέστρεφε ένα μήνυμα λάθους ή καλύτερα ένα SQLException ότι η οδηγία sql προς τη βάση δεν ήταν σωστή ή πλήρης . Για τον παραπάνω λόγο , στην εφαρμογή έχουμε μεταφέρει την κονσόλα της εξόδου στο αρχικό παράθυρο της κεντρικής διαχείρισης της βάσης δεδομένων , ώστε όλα τα μηνύματα να είναι ορατά στον χρήστη της εφαρμογής χωρίς ν’ απαιτείται η παρουσία της κονσόλας.

3.2.4 Βοηθητικές Κλάσεις

1. Σύνδεση με τη βάση δεδομένων

Το πακέτο dbtools περιέχει δύο κλάσεις τις DBUtil.java και SQLConnection.java

Η SQLConnection.java περιέχει απλά τον driver που χρησιμοποιεί κάθε java application για την σύνδεση της με μια βάση δεδομένων SQLite

```
package dbtools;


import java.sql.Connection;
import java.sql.DriverManager;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author trojan
 */

public class SqliteConnection {
    public static Connection connector(){
        try{
            Class.forName("org.sqlite.JDBC");
            Connection conn = DriverManager.getConnection("jdbc:sqlite:"
                    + "/home/trojan/Desktop"
                    + "/Diplomatikh/CompanyTracker/db/development.sqlite3");
            System.out.println("Connected to database!");
            return conn;
```

```
        }
        catch (Exception e){
            //ToDO : handle Exception
            e.printStackTrace();
            System.out.println(e);
            return null;
        }

    }

}
```

Η κλάση DBUtil.java περισσότερο χρήσιμη αφού εκτός από τη σύνδεση με τη βάση δεδομένων , περιέχει τις κατάλληλες μεθόδους που καταναλώνουν τα web services από τη βάση δεδομένων . Ουσιαστικά λειτουργεί σαν ένα ενδιάμμεσο επίπεδο μετάξύ του DAO αντικειμένου της κάθε οντότητας και της βάσης δεδομένων . Οι δύο βασικές μέθοδοι της κλάσης (dbExecuteQuery() ,dbExecuteUpdate()), DB είναι αυτές που εκτελούν τα βασικά Web Verbs με τη μορφή queries για την βάση (Create/Read/Update/Delete) .

```
package dbtools;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetProvider;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author trojan
 */
public class DBUtil {

    //Connection
    private static Connection conn = null;
```

```
//Connect to DB
public static void dbConnect() throws SQLException, ClassNotFoundException {
    //Setting SQlite  Driver
    try {
        Class.forName("org.sqlite.JDBC");
    } catch (ClassNotFoundException e) {
        System.out.println("Sqlite JDBC Driver not found!");
        e.printStackTrace();
        throw e;
    }

    System.out.println("SQLite JDBC Driver Registered!");

    //Establish the DB Connection using Connection String
    try {
        conn = DriverManager.getConnection("jdbc:sqlite:"
                + "/home/trojan/Desktop/Diplomatikh/CompanyTracker/db/development.sqlite3");
    } catch (SQLException e) {
        System.out.println("Connection Failed! Check output console" + e);
       e.printStackTrace();
        throw e;
    }
}

//Close Connection
public static void dbDisconnect() throws SQLException {
    try {
        if (conn != null && !conn.isClosed()) {
            conn.close();
        }
    } catch (Exception e){
       throw e;
    }
}

//DB Execute Query Operation
public static ResultSet dbExecuteQuery(String queryStmt) throws SQLException,
ClassNotFoundException {
    //Declare statement, resultSet and CachedResultSet as null
    Statement stmt = null;
    ResultSet resultSet = null;
    CachedRowSet crs = RowSetProvider.newFactory().createCachedRowSet();
    try {
        //Connect to DB (Establish Oracle Connection)
        dbConnect();
        System.out.println("Select statement: " + queryStmt + "\n");

        //Create statement
        stmt = conn.createStatement();
```

```
            //Execute select (query) operation
            resultSet = stmt.executeQuery(queryStmt);
            crs.populate(resultSet);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeQuery operation : " + e);
            throw e;
        } finally {
            if (resultSet != null) {
                //Close resultSet
                resultSet.close();
            }
            if (stmt != null) {
                //Close Statement
                stmt.close();
            }
            //Close connection
            dbDisconnect();
        }
        //Return CachedRowSet
        return crs;
    }

    //DB Execute Update (For Update/Insert/Delete) Operation
    public static void dbExecuteUpdate(String sqlStmt) throws SQLException,
ClassNotFoundException {
        //Declare statement as null
        Statement stmt = null;

        try {
            dbConnect();
            stmt = conn.createStatement();
            stmt.executeUpdate(sqlStmt);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeUpdate operation : " + e);
            throw e;
        } finally {
            if (stmt != null) {
                stmt.close();
            }
            dbDisconnect();
        }
    }
}
```

3.2.5 Ο controller της οντότητας Company,ConpanyController.java

```
package company;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.net.URL;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

/**
 * FXML Controller class
 *
 * @author trojan
 */
public class CompaniesController implements Initializable {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Label lblCompanies;

    @FXML
    private TableView<Company> tableCompanies;

    @FXML
    private TableColumn<Company, Integer> idColumn;
```

```
    @FXML
    private TableColumn<Company, String> nameColumn;

    @FXML
    private TableColumn<Company, String> slugColumn;

    @FXML
    private TableColumn<Company, String> createdColumn;

    @FXML
    private TableColumn<Company, String> updatedColumn;

    @FXML
    private Button btnListCompanies;

    @FXML
    private Button btnBackToMain;

    @FXML
    private TextField txtCompanyId;

    @FXML
    private TextField txtCompanyName;

    @FXML
    private TextField txtCompanySlug;

    @FXML
    private Button btnCreate;

    @FXML
    private Button btnUpdate;

    @FXML
    private Button btnDelete;

    @FXML
    private Button btnSearch;

     @FXML
    private TextArea systemOutput;

    @FXML
    public ObservableList<Company> companyList;
   // public LoginModel lg;

    /**
     * Initializes the controller class.
     *
     * @param url
```

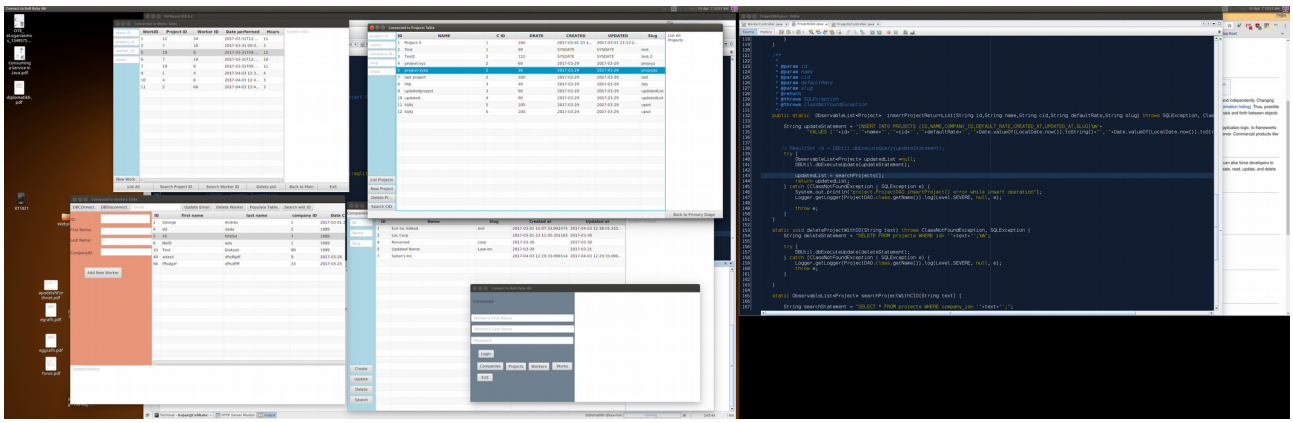
```
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        idColumn.setCellValueFactory(celldata -> celldata.getValue().idProperty().asObject());
        nameColumn.setCellValueFactory(celldata -> celldata.getValue().nameProperty());
        slugColumn.setCellValueFactory(celldata -> celldata.getValue().slugProperty());
        createdColumn.setCellValueFactory(celldata -> celldata.getValue().createdProperty());
        updatedColumn.setCellValueFactory(celldata -> celldata.getValue().updatedProperty());

    }

    @FXML
    public void returnToprimaryStage(ActionEvent event) {

        try {

            ((Node) event.getSource()).getScene().getWindow().hide();
//              Stage primaryStage = new Stage();
//              FXMLLoader fxmlLoader = new FXMLLoader();
//
//              Parent root =
fxmlLoader.load(getClass().getResource("MainFXML.fxml").openStream());
//              //CompaniesController companiesController = (CompaniesController)
fxmlLoader.getController();
//              Scene scene = new Scene(root);
//              scene.getStylesheets().add(getClass().getResource("mainfxml.css").toExternalForm());
//
//              primaryStage.setScene(scene);
//              primaryStage.setTitle("Connected to database");
//              primaryStage.setScene(scene);
//              primaryStage.show();

        } catch (Exception e) {
        }
    }

    private Connection connect() {
        // SQLite connection string
        String url =
"jdbc:sqlite:/home/trojan/Desktop/Διπλωματική/CompanyTracker/db/development.sqlite3";
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        return conn;
    }
```

```
    @FXML
    public void listCompaniesTable(ActionEvent event) throws SQLException,
ClassNotFoundException {

        try {

            ObservableList<Company> compData = CompanyDAO.listCompanies();
            //populate Companies on  TAbleView

            populateCompanies(compData);

        } catch (SQLException e) {
            System.out.println("Error occured while getting companis information from db.\n" + e);
            throw e;
        }
    }

    //Search all companies
    @FXML

    private void searchCompanies(ActionEvent actionEvent) throws SQLException,
ClassNotFoundException {

        try {
            //Get all Employees information
            ObservableList<Company> compData =
CompanyDAO.searchCompanies(txtCompanyId.getText());
            //Populate Employees on TableView
            populateCompanies(compData);
        } catch (SQLException e) {
            System.out.println("Error occurred while getting companies information from DB.\n" + e);
            throw e;
        }
    }

    @FXML
    private void populateCompany(Company comp) throws ClassNotFoundException {
        ObservableList<Company> compData = FXCollections.observableArrayList();
        compData.add(comp);
        tableCompanies.setItems(compData);

    }

    @FXML
    private void setCompInfoToTextArea(Company comp){
        systemOutput.setText("Company Name:" +comp.getName()+"\n"+
                "company Slug:"+comp.getSlug());
    }

    @FXML
```

```
    private void populateAndShowCompany(Company comp) throws ClassNotFoundException
    {
        if(comp !=null){
            populateCompany(comp);
            setCompInfoToTextArea(comp);
        }else{
            systemOutput.setText("This company does not exist!");
        }

    }

    @FXML
    private void populateCompanies(ObservableList<Company> compData)
            throws ClassNotFoundException {

        tableCompanies.setItems(compData);
    }



    @FXML
    private void createCompanyName(ActionEvent ev) throws
SQLException,ClassNotFoundException
    {
        try{
        CompanyDAO.insertComp(txtCompanyId.getText(),txtCompanyName.getText(),
txtCompanySlug.getText());
        systemOutput.setText("Company created \n"
        +"Company ID:"+txtCompanyId.getText()+"\n"+
                "Company Name:"+txtCompanyName.getText()+"\n"
        +"Slug:"+txtCompanySlug.getText()+"\n");
        }
        catch(SQLException e){
            systemOutput.setText("Problem occured while inserting company "+e);
            throw  e;
        }

    }

    @FXML
    private void updateCompanyName(ActionEvent event) throws
SQLException,ClassNotFoundException{
        try {
            CompanyDAO.updateCompName(txtCompanyId.getText(),txtCompanyName.getText());
            systemOutput.setText("Name has been changed for company with
id:"+txtCompanyId.getText() +"\n"
            +"Company Name: "+txtCompanyName.getText()+"\n"
            +""+Date.valueOf(LocalDate.now()).toString()+"\n");
        } catch (Exception e) {
            systemOutput.setText("Problem occured while updating Company' name "+e);
```

```
            throw e;
        }
    }

    @FXML void deleteCompany(ActionEvent ev) throws SQLException,ClassNotFoundException{

        try {
            CompanyDAO.deleteCompanyWithID(txtCompanyId.getText());
            systemOutput.setText("Company deleted! Company id: "+txtCompanyId.getText());
        } catch (Exception e) {
            systemOutput.setText("Problem occured while updating Company' name "+e);
            throw e;
        }
    }
    @FXML Button btnExit;
    @FXML
    private void systemExit(ActionEvent e){
        systemOutput.setText("The system is going to shut down !!!\n");
        System.exit(0);
    }

}
```

Ένα στιγμιότυπο από την εφαρμογή στην σημερινή της μορφή ακολουθεί



Εικόνα 3.2.5 : Στιγμιότυπο της εφαρμογής πελάτη

3.2.6 Κώδικας των αρχείων της αρχικής εναρκτήριας κλάσης της εφαρμογής και οι κλάσεις μιας από τις οντότητες της βάσης δεδομένων

Diplomatikh.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */


import javafx.application.Application;
import javafx.scene.Parent;
import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 *
 * @author trojan
 */
public class Diplomatikh extends Application {

    @Override
    public void start(Stage primaryStage) {

        try{
            Parent root = FXMLLoader.load(getClass().getResource("MainFXML.fxml"));
            Scene scene = new Scene(root,800,1200);
            scene.getStylesheets().add(getClass().getResource("mainfxml.css").toExternalForm());




        primaryStage.setScene(scene);
        primaryStage.setTitle("Connect to RoR Ryby db!");
        primaryStage.setScene(scene);
        primaryStage.show();
         }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

MainFXMLController.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import java.io.Console;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import static java.lang.System.console;
import javafx.event.ActionEvent;
import java.net.URL;
import java.sql.SQLException;
import java.time.ZonedDateTime;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author trojan
 */
public class MainFXMLController implements Initializable {

    /**
     * Initializes the controller class.
     */
    public LoginModel loginModel = new LoginModel();

    @FXML
    private Label isConnected;

    @FXML
    private TextField txtfName;
```

```
    @FXML
    private TextField txtlName;

    @FXML
    private TextField txtPassword;

    @FXML
    private Button loginButton;

    @FXML
    private Button btnCompanies;

    @FXML
    private Button btnProjects;

    @FXML
    private Button btnWorkers;

    @FXML
    private Button btnWorks;
    @FXML
    private TextArea txtSystemOUT;

    private PrintStream ps;

    @Override
    public void initialize(URL url, ResourceBundle rb) {



        if (loginModel.isDbConnected()) {
            isConnected.setText("Connected");
        } else {
            isConnected.setText("Not Connected");
        }
        // Console Code Test
       // ps = new PrintStream(new Console(txtSystemOUT));
    }

    public void Login(ActionEvent event) {
        try {
            if (loginModel.isLogin(txtfName.getText(), txtlName.getText())) {
                isConnected.setText("User " + txtfName.getText() + " " + txtlName.getText() + " logged
in at " + ZonedDateTime.now().toString());
                //((Node)event.getSource()).getScene().getWindow().hide();
            } else {
                isConnected.setText("The Worker does not exist");
            }
        } catch (SQLException ex) {
```

```
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null, ex);
            isConnected.setText("No worker with such credentials");
            ex.printStackTrace();
        }
    }

    public void companiesTableView(ActionEvent event) throws  Exception{
        try {

            if (loginModel.isDbConnected()) {
                //TODO
                //perimene na eleg8ei giati an getWindows().hide() mporei na mpo kleinei to connection
me thn db 8a to doume ...
                // ((Node)event.getSource()).getScene().getWindow().hide();
                Parent root;
                Stage companiesStage = new Stage();
                FXMLLoader fxmlLoader = new FXMLLoader();
                System.out.println("before companies launch ... ");
                root = fxmlLoader.load(getClass().getResource("Companies.fxml").openStream());
               // CompaniesController companiesController = (CompaniesController)
fxmlLoader.getController();
                System.out.println("after the controler launch ...");
                Scene scene = new Scene(root);
                scene.getStylesheets().add(getClass().getResource("companies.css").toExternalForm());

                companiesStage.setScene(scene);
                companiesStage.setTitle("Connected to Companies Table");
                companiesStage.setScene(scene);
                companiesStage.show();
            } else {
                System.out.println("You r not connected to the database.");
            }

        } catch (Exception e) {
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null,e);
        }
    }
//ok
    public void workersTableView(ActionEvent event) throws SQLException,
ClassNotFoundException, IOException {
        try {
            System.out.println("Button Workers pressed");

            if (loginModel.isDbConnected()) {

                Stage workersStage = new Stage();

                FXMLLoader fxmlLoader = new FXMLLoader();

                Parent root =
```

```
fxmlLoader.load(getClass().getResource("worker/worker.fxml").openStream());
            //  worker.WorkerController workerController = (WorkerController)
fxmlLoader.getController();

            System.out.println("Just passed the getResource worker.fxml");
            Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("worker/worker.css").toExternalForm());
            workersStage.setScene(scene);
            workersStage.setTitle("Connected to Workers Table");
            workersStage.setScene(scene);
            workersStage.show();
         } else {
            System.out.println("You r not connected to the database.");
         }

      } catch (IOException e) {
         System.out.println("Exception occured while trying to connect to the database..." + e);
         Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE,null,e);

      }

   }
   // end OK

   @FXML
   public void worksTableView(ActionEvent event) throws
SQLException,ClassNotFoundException,IOException
   {
      try {
         System.out.println("Button Works pressed");

         if (loginModel.isDbConnected()) {


            Stage worksStage = new Stage();
            FXMLLoader fxmlLoader = new FXMLLoader();
            Parent root =
fxmlLoader.load(getClass().getResource("work/works.fxml").openStream());

            System.out.println("Just passed the getResource work/works.fxml");
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("work/works.css").toExternalForm());
            worksStage.setScene(scene);
            worksStage.setTitle("Connected to Works Table");
            worksStage.setScene(scene);
            worksStage.show();
         } else {
            System.out.println("You r not connected to the database.");
         }
```

```
        } catch (IOException ex) {
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    /**
     *
     * @param event
     * @throws java.io.IOException
     * @throws java.lang.ClassNotFoundException
     */
    @FXML
    public void projectsTableView(ActionEvent event) throws IOException ,
ClassNotFoundException
    {
        try {
            System.out.println("Button Projects pressed");

            if (loginModel.isDbConnected()) {
                Stage projectsStage = new Stage();
                FXMLLoader fxmlLoader = new FXMLLoader();

                Parent root =
fxmlLoader.load(getClass().getResource("project/projects.fxml").openStream());

                System.out.println("Just passed the getResource ptojects.fxml");
                Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("project/projects.css").toExternalForm());

                projectsStage.setScene(scene);
                projectsStage.setTitle("Connected to Projects Table");
                projectsStage.setScene(scene);
                projectsStage.show();
            } else {
                System.out.println("You r not connected to the database.");
            }

        } catch (IOException e) {
            System.out.println("Exception occured while trying to connect to the database..." + e);
            Logger.getLogger(MainFXMLController.class.getName()).log(Level.SEVERE,null,e);

        }
    }
    @FXML
    private Button btnEXIT;
    @FXML
    private void systemExit(ActionEvent e){
        System.exit(0);
    }
```

MainFXML.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.net.URL?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane id="AnchorPane" prefHeight="482.0" prefWidth="829.0" style="-fx-background-color: slategrey;" styleClass="mainFxmlClass" xmlns="http://javafx.com/javafx/8.0.111" xmlns:fx="http://javafx.com/fxml/1" fx:controller="MainFXMLController">
    <stylesheets>
        <URL value="@mainfxml.css" />
    </stylesheets>
    <children>
        <Label fx:id="isConnected" layoutX="10.0" layoutY="2.0" prefHeight="60.0" prefWidth="754.0" text="Status" textAlignment="CENTER" />
        <Button fx:id="loginButton" layoutX="28.0" layoutY="201.0" mnemonicParsing="false" onAction="#Login" text="Login" />
        <PasswordField fx:id="txtPassword" layoutX="2.0" layoutY="155.0" prefHeight="26.0" prefWidth="357.0" promptText="Password" />
        <TextField fx:id="txtfName" layoutX="3.0" layoutY="77.0" prefHeight="26.0" prefWidth="357.0" promptText="Worker's First Name" />
        <TextField fx:id="txtlName" layoutX="5.0" layoutY="115.0" prefHeight="26.0" prefWidth="357.0" promptText="Worker's Last Name" />
        <Button fx:id="btnWorks" layoutX="285.0" layoutY="244.0" mnemonicParsing="false" onAction="#worksTableView" prefHeight="26.0" prefWidth="70.0" text="Works" />
        <Button fx:id="btnProjects" layoutX="124.0" layoutY="245.0" mnemonicParsing="false" onAction="#projectsTableView" text="Projects" />
        <Button fx:id="btnWorkers" layoutX="203.0" layoutY="245.0" mnemonicParsing="false" onAction="#workersTableView" text="Workers" />
        <Button fx:id="btnCompanies" layoutX="24.0" layoutY="244.0" mnemonicParsing="false" onAction="#companiesTableView" text="Companies" />
      <Button fx:id="btnEXIT" layoutX="24.0" layoutY="283.0" mnemonicParsing="false" onAction="#systemExit" prefHeight="26.0" prefWidth="54.0" text="Exit" />
      <TextArea fx:id="txtSystemOUT" layoutX="363.0" prefHeight="482.0" prefWidth="466.0" />
    </children>
</AnchorPane>
```

Company.java

```
import java.time.ZonedDateTime;
```

```
import javafx.beans.property.*;


//The Company MODEL class
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.

 */

/**
 *
 * @author trojan
 */
public class Company {

    //menei na do an gia thn Observable list xreiazetai na oriso auta ta pedia os
    // SrtingProperty kai oxi String gia na einai obsevable apo th list dynamika ...
    //auto 8a eleg8ei parakato
    private StringProperty name;
    private IntegerProperty id;
    private StringProperty created_at;
    private StringProperty updated_at;
    private StringProperty slug;

    //private ZonedDateTime zdt ;

    public Company(){
        this.id = new SimpleIntegerProperty();
        this.name = new SimpleStringProperty();
        this.slug = new SimpleStringProperty();
        this.created_at = new SimpleStringProperty();
        this.updated_at = new SimpleStringProperty();
    }

//    public Company(int id,String name,String slug,String created,String updated){
//        this.id.set(id);
//
//        if(name==null){
//            nameNull();
//        }
//        else{
//
//
//        this.name.set(name);
//        }
//        if(slug == null){
//          slugNull();
//        }
```

```
//        else
//        {
//        this.slug.set(slug);
//        }
//        this.created_at.set(created);
//        this.updated_at.set(updated);
//    }
//
//
//    public Company(String name)
//    {
//
//
//        //TODO
//
//        //this.id=id; prosoxh giati kata th dimiougia tou antikeimenou to id pou 8a parei einai h
//        //default timh arxikopoihshs ton int dld 0 , opos kai to to slug 8a parei thn timh ""
//        //to 8ema einai an h bash meta 8a ths dosei to sosto id , den m endiaferei gia to slug
//        //alloste  den exo blaei kapoio periosmo oste na ginetai validate gia to slug opote mporei na
einai kai keno
//        // se anti8esh me to company_id pou to dinei h bash kai den mporei na einai 0  kai prepei na
einau monadiko
//        //autoi oi periorismoi ananferontai sth  dhmiourgia ths bashs , opote an auto den ginetai
dekto apo th bash 8a prepei na
//        //tropopoihso to kataskeuasth tou antikeimenou Company oste na diabazei ton pinaka
Companies , na briskei to megalutero id kai meta na dinei id
//        // id.max +1 ; Auto 8a ginei sto telos !
//
//        //zdt = ZonedDateTime.now();
//        this.name = new SimpleStringProperty(name);
//        this.created_at = new SimpleStringProperty(ZonedDateTime.now().toString());
//        this.updated_at = new SimpleStringProperty(ZonedDateTime.now().toString());
//
//    }
//
    public void setName(String newName){
        name.set(newName);
    }
//    public void setId(int newId){
//        id.set(newID);
//    }
    public void updatedAt(){
    //    zdt = ZonedDateTime.now();
        updated_at.set(ZonedDateTime.now().toString());

    }
    public void setSlug(String newSlug){
        slug.set(newSlug);
    }
```

```
public void setCreated(String created){
    created_at.set(created);
}
public void setUpdated(String updated)
{
    updated_at.set(updated);
}

public void setId(int id){
    this.id.set(id);
}

public String getName(){
    return name.get();
}
public int getId(){
    return id.get();
}
public String getCreatedAt(){
    return created_at.get();
}
public String getUpdatedAt(){
    return updated_at.get();
}
public String getSlug(){
    return slug.get();
}

//Property Values

public StringProperty nameProperty(){
    return name;
}
public IntegerProperty idProperty(){
    return id;
}
public StringProperty slugProperty(){
    return slug;
}
public StringProperty createdProperty(){
    return created_at;

}
public StringProperty updatedProperty(){
    return updated_at;
}
public final void slugNull(){
    slug.set("");
}
public final void nameNull(){
```

```
        name.set("");
    }


}
```

CompanyController.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.net.URL;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

/**
 * FXML Controller class
 *
 * @author trojan
 */
public class CompaniesController implements Initializable {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Label lblCompanies;
```

```
@FXML
private TableView<Company> tableCompanies;

@FXML
private TableColumn<Company, Integer> idColumn;

@FXML
private TableColumn<Company, String> nameColumn;

@FXML
private TableColumn<Company, String> slugColumn;

@FXML
private TableColumn<Company, String> createdColumn;

@FXML
private TableColumn<Company, String> updatedColumn;

@FXML
private Button btnListCompanies;

@FXML
private Button btnBackToMain;

@FXML
private TextField txtCompanyId;

@FXML
private TextField txtCompanyName;

@FXML
private TextField txtCompanySlug;

@FXML
private Button btnCreate;

@FXML
private Button btnUpdate;

@FXML
private Button btnDelete;

@FXML
private Button btnSearch;

 @FXML
private TextArea systemOutput;

@FXML
public ObservableList<Company> companyList;
```

```
    // public LoginModel lg;

    /**
     * Initializes the controller class.
     *
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        idColumn.setCellValueFactory(celldata -> celldata.getValue().idProperty().asObject());
        nameColumn.setCellValueFactory(celldata -> celldata.getValue().nameProperty());
        slugColumn.setCellValueFactory(celldata -> celldata.getValue().slugProperty());
        createdColumn.setCellValueFactory(celldata -> celldata.getValue().createdProperty());
        updatedColumn.setCellValueFactory(celldata -> celldata.getValue().updatedProperty());

    }

    @FXML
    public void returnToprimaryStage(ActionEvent event) {

        try {

            ((Node) event.getSource()).getScene().getWindow().hide();
//              Stage primaryStage = new Stage();
//              FXMLLoader fxmlLoader = new FXMLLoader();
//
//              Parent root =
fxmlLoader.load(getClass().getResource("MainFXML.fxml").openStream());
//              //CompaniesController companiesController = (CompaniesController)
fxmlLoader.getController();
//              Scene scene = new Scene(root);
//              scene.getStylesheets().add(getClass().getResource("mainfxml.css").toExternalForm());
//
//              primaryStage.setScene(scene);
//              primaryStage.setTitle("Connected to database");
//              primaryStage.setScene(scene);
//              primaryStage.show();

        } catch (Exception e) {
        }
    }

    private Connection connect() {
        // SQLite connection string
        String url =
"jdbc:sqlite:/home/trojan/Desktop/Διπλωματική/CompanyTracker/db/development.sqlite3";
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url);
```

```
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        return conn;
    }

    @FXML
    public void listCompaniesTable(ActionEvent event) throws SQLException,
ClassNotFoundException {

        try {

            ObservableList<Company> compData = CompanyDAO.listCompanies();
            //populate Companies on  TAbleView

            populateCompanies(compData);

        } catch (SQLException e) {
            System.out.println("Error occured while getting companis information from db.\n" + e);
            throw e;
        }
    }

    //Search all companies
    @FXML

    private void searchCompanies(ActionEvent actionEvent) throws SQLException,
ClassNotFoundException {

        try {
            //Get all Comapanies information
            ObservableList<Company> compData =
CompanyDAO.searchCompanies(txtCompanyId.getText());
            //Populate Employees on TableView
            populateCompanies(compData);
        } catch (SQLException e) {
            System.out.println("Error occurred while getting companies information from DB.\n" + e);
            throw e;
        }
    }

    @FXML
    private void populateCompany(Company comp) throws ClassNotFoundException {
        ObservableList<Company> compData = FXCollections.observableArrayList();
        compData.add(comp);
        tableCompanies.setItems(compData);

    }

    @FXML
```

```
    private void setCompInfoToTextArea(Company comp){
        systemOutput.setText("Company Name:" +comp.getName()+"\n"+
                "company Slug:"+comp.getSlug());
    }

    @FXML
    private void populateAndShowCompany(Company comp) throws ClassNotFoundException
    {
        if(comp !=null){
            populateCompany(comp);
            setCompInfoToTextArea(comp);
        }else{
            systemOutput.setText("This company does not exist!");
        }

    }

    @FXML
    private void populateCcompanies(ObservableList<Company> compData)
            throws ClassNotFoundException {

        tableCompanies.setItems(compData);
    }



    @FXML
    private void createCompanyName(ActionEvent ev) throws
SQLException,ClassNotFoundException
    {
        try{
        CompanyDAO.insertComp(txtCompanyId.getText(),txtCompanyName.getText(),
txtCompanySlug.getText());
        systemOutput.setText("Company created \n"
        +"Company ID:"+txtCompanyId.getText()+"\n"+
                "Company Name:"+txtCompanyName.getText()+"\n"
        +"Slug:"+txtCompanySlug.getText()+"\n");
        }
        catch(SQLException e){
            systemOutput.setText("Problem occured while inserting company "+e);
            throw  e;
        }

    }

    @FXML
    private void updateCompanyName(ActionEvent event) throws
SQLException,ClassNotFoundException{
        try {
            CompanyDAO.updateCompName(txtCompanyId.getText(),txtCompanyName.getText());
```

```
            systemOutput.setText("Name has been changed for company with
id:"+txtCompanyId.getText() +"\n"
            +"Company Name: "+txtCompanyName.getText()+"\n"
            +""+Date.valueOf(LocalDate.now()).toString()+"\n");
        } catch (Exception e) {
            systemOutput.setText("Problem occured while updating Company' name "+e);
            throw e;
        }
    }

    @FXML void deleteCompany(ActionEvent ev) throws SQLException,ClassNotFoundException{

        try {
            CompanyDAO.deleteCompanyWithID(txtCompanyId.getText());
            systemOutput.setText("Company deleted! Company id: "+txtCompanyId.getText());
        } catch (Exception e) {
            systemOutput.setText("Problem occured while updating Company' name "+e);
            throw e;
        }
    }
    @FXML Button btnExit;
    @FXML
    private void systemExit(ActionEvent e){
        systemOutput.setText("The system is going to shut down !!!\n");
        System.exit(0);
    }

}

CompanyDAO.java

import dbtools.DBUtil;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import org.sqlite.SQLiteException;

public class CompanyDAO {

    //SELECT a COMPANY

    public static Company searchCompany(String companyId) throws
SQLiteException,SQLException, ClassNotFoundException {
        //Declare a SELECT statement
        String selectStmt = "SELECT * FROM companies WHERE company_id=" + companyId;

        //Execute SELECT statement
```

```
        try {
            //Get ResultSet from dbExecuteQuery method
            ResultSet rsCmp = DBUtil.dbExecuteQuery(selectStmt);

            //Send ResultSet to the getEmployeeFromResultSet method and get employee object
            Company comp = getCompanyFromResultSet(rsCmp);

            //Return employee object
            return comp;
        } catch (SQLiteException e) {
            System.out.println("While searching a Company with " + companyId + " id, an error
occurred: " + e);
            //Return exception
            throw e;
        }
    }

    //List all Companies
    private static Company getCompanyFromResultSet(ResultSet rs) throws
SQLiteException,SQLException {
        Company comp = null;

        if (rs.next()) {
            comp = new Company();
            comp.setId(rs.getInt(1));
            comp.setName(rs.getString(2));
            comp.setSlug(rs.getString(5));
            comp.setCreated(rs.getString(3));
            comp.setUpdated(rs.getString(4));

        }
        return comp;

    }

    //SELECT ALL COMPANIES
    public static ObservableList<Company> searchCompanies(String id) throws SQLException,
ClassNotFoundException {

        String selectStmt = "SELECT * FROM companies WHERE ID='"+id+"'\n";

        //execute select statement
        try {
            ResultSet rs = DBUtil.dbExecuteQuery(selectStmt);

            ObservableList<Company> compList = getCompanyListFrom(rs);
            return compList;
        } catch (Exception e) {

            System.out.println("SQL select operation has been failed: " + e);
```

```
            throw e;
        }

    }

    public static ObservableList<Company> listCompanies() throws SQLException,
ClassNotFoundException {

        String selectStmt = "SELECT * FROM companies \n";

        //execute select statement
        try {
            ResultSet rs = DBUtil.dbExecuteQuery(selectStmt);

            ObservableList<Company> compList = getCompanyListFrom(rs);
            return compList;
        } catch (Exception e) {

            System.out.println("SQL select operation has been failed: " + e);
            throw e;
        }

    }

    private static ObservableList<Company> getCompanyListFrom(ResultSet rs) throws
SQLException, ClassNotFoundException {

        ObservableList<Company> compList = FXCollections.observableArrayList();

        while (rs.next()) {
            Company comp = new Company();
            comp.setId(rs.getInt(1));
            comp.setName(rs.getString(2));
            comp.setSlug(rs.getString(5));
            comp.setCreated(rs.getString(3));
            comp.setUpdated(rs.getString(4));

            compList.add(comp);
        }
        return compList;

    }

    public static void updateCompName(String compID, String compName) throws SQLException,
ClassNotFoundException {

String updateStmt = "UPDATE companies SET NAME ='"+compName+"',UPDATED_AT
='"+Date.valueOf(LocalDate.now())+"' WHERE ID='"+compID+"'";

        //Execute UPDATE operation
```

```
        try {
            DBUtil.dbExecuteUpdate(updateStmt);
        } catch (SQLException e) {
            System.out.println("Error occured while UPDATE operation: " + e);
            throw e;
        }
    }

    //Delete a Company TODO: FIX STATEMENT
    public static void deleteCompanyWithID(String compID) throws SQLException,
ClassNotFoundException{

        String updateStmt =
                " DELETE from companies\n"
                + " WHERE ID ='" + compID + "';\n";

    try {
            DBUtil.dbExecuteUpdate(updateStmt);
    }catch (SQLException e) {
            System.out.println("No Company Deleted:" + e);
            throw e;
        }
    }

    //INSERT a Company STATEMENT OK
    public static void insertComp(String id,String name,String slug)throws
SQLException,ClassNotFoundException
    {
        String updateStmt
                = "INSERT INTO COMPANIES (ID,NAME,CREATED_AT,UPDATED_AT,SLUG)\n" +
                  "VALUES ('"+id+"','"+name +"','"+ Date.valueOf(LocalDate.now())
+"','"+Date.valueOf(LocalDate.now())+"','"+slug+"');";

   //     String updateStmt = "insert into companies where  name ="+name+"and slug
="+slug+";";

        try {
            DBUtil.dbExecuteUpdate(updateStmt);
        } catch (SQLException e) {
            System.out.println("Error occured while INSERT operation: " + e);
            throw e;
        }
    }
}
```

companies.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
```

```
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="800.0" prefWidth="1279.0" xmlns="http://javafx.com/javafx/8.0.111"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="CompaniesController">
   <center>
      <TableView fx:id="tableCompanies" prefHeight="747.0" prefWidth="1000.0"
BorderPane.alignment="CENTER">
        <columns>
          <TableColumn fx:id="idColumn" maxWidth="51.0" prefWidth="51.0" text="ID" />
          <TableColumn fx:id="nameColumn" maxWidth="345.0" minWidth="0.0"
prefWidth="300.0" text="Name" />
            <TableColumn fx:id="slugColumn" prefWidth="121.0" text="Slug" />
            <TableColumn fx:id="createdColumn" maxWidth="551.0" prefWidth="201.0"
text="Created at" />
            <TableColumn fx:id="updatedColumn" maxWidth="617.0" prefWidth="197.0"
text="Updated at" />
        </columns>
      </TableView>
   </center>
   <bottom>
      <ButtonBar BorderPane.alignment="CENTER">
        <buttons>
          <Button fx:id="btnListCompanies" mnemonicParsing="false"
onAction="#listCompaniesTable" text="List All Companies" />
          <Button fx:id="btnBackToMain" mnemonicParsing="false"
onAction="#returnToprimaryStage" text="Back to Main" />
          <Button fx:id="btnExit" layoutX="724.0" layoutY="10.0" mnemonicParsing="false"
onAction="#systemExit" text="Exit" />
        </buttons>
      </ButtonBar>
   </bottom>
   <left>
      <VBox prefHeight="200.0" prefWidth="100.0" style="-fx-background-color: lightblue;"
BorderPane.alignment="CENTER">
        <children>
          <TextField fx:id="txtCompanyId" promptText="ID">
            <VBox.margin>
              <Insets bottom="10.0" left="10.0" right="10.0" top="5.0" />
            </VBox.margin>
          </TextField>
```

```
        <TextField fx:id="txtCompanyName" promptText="Name">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" />
          </VBox.margin>
        </TextField>
        <TextField fx:id="txtCompanySlug" promptText="Slug">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" />
          </VBox.margin>
        </TextField>
        <Button fx:id="btnCreate" layoutX="10.0" layoutY="62.0" mnemonicParsing="false"
onAction="#createCompanyName" prefHeight="26.0" prefWidth="93.0" text="Create">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" top="400.0" />
          </VBox.margin>
        </Button>
        <Button fx:id="btnUpdate" layoutX="10.0" layoutY="36.0" mnemonicParsing="false"
onAction="#updateCompanyName" onDragDetected="#updateCompanyName"
prefHeight="26.0" prefWidth="87.0" text="Update">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" />
          </VBox.margin>
        </Button>
        <Button fx:id="btnDelete" mnemonicParsing="false" onAction="#deleteCompany"
prefHeight="26.0" prefWidth="83.0" text="Delete">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" />
          </VBox.margin>
        </Button>
        <Button fx:id="btnSearch" layoutX="10.0" layoutY="10.0" mnemonicParsing="false"
onAction="#searchCcompanies" prefHeight="26.0" prefWidth="80.0" text="Search">
          <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" />
          </VBox.margin>
        </Button>
      </children>
    </VBox>
  </left>
  <top>
    <HBox prefHeight="27.0" prefWidth="1000.0" BorderPane.alignment="CENTER">
      <children>
        <Label fx:id="lblCompanies" prefHeight="23.0" prefWidth="122.0" text="Companies
Table" textAlignment="RIGHT" />
      </children>
    </HBox>
  </top>
  <right>
    <TextArea fx:id="systemOutput" prefHeight="747.0" prefWidth="308.0" promptText="System
Output" BorderPane.alignment="CENTER" />
  </right>
```

```
</BorderPane>
```

DBUtils.java

```
package dbtools;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetProvider;

public class DBUtil {

   //Connection
   private static Connection conn = null;



   //Connect to DB
   public static void dbConnect() throws SQLException, ClassNotFoundException {
      //Setting SQlite  Driver
      try {
         Class.forName("org.sqlite.JDBC");
      } catch (ClassNotFoundException e) {
         System.out.println("Sqlite JDBC Driver not found!");
         e.printStackTrace();
         throw e;
      }

      System.out.println("SQLite JDBC Driver Registered!");

      //Establish the DB Connection using Connection String
      try {
         conn = DriverManager.getConnection("jdbc:sqlite:"
               + "/home/trojan/Desktop/Diplomatikh/CompanyTracker/db/development.sqlite3");
      } catch (SQLException e) {
         System.out.println("Connection Failed! Check output console" + e);
        e.printStackTrace();
         throw e;
      }
   }

   //Close Connection
   public static void dbDisconnect() throws SQLException {
```

```
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (Exception e){
            throw e;
        }
    }

    //DB Execute Query Operation
    public static ResultSet dbExecuteQuery(String queryStmt) throws SQLException,
ClassNotFoundException {
        //Declare statement, resultSet and CachedResultSet as null
        Statement stmt = null;
        ResultSet resultSet = null;
        CachedRowSet crs = RowSetProvider.newFactory().createCachedRowSet();
        try {
            //Connect to DB (Establish Oracle Connection)
            dbConnect();
            System.out.println("Select statement: " + queryStmt + "\n");

            //Create statement
            stmt = conn.createStatement();

            //Execute select (query) operation
            resultSet = stmt.executeQuery(queryStmt);
            crs.populate(resultSet);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeQuery operation : " + e);
            throw e;
        } finally {
            if (resultSet != null) {
                //Close resultSet
                resultSet.close();
            }
            if (stmt != null) {
                //Close Statement
                stmt.close();
            }
            //Close connection
            dbDisconnect();
        }
        //Return CachedRowSet
        return crs;
    }

    //DB Execute Update (For Update/Insert/Delete) Operation
    public static void dbExecuteUpdate(String sqlStmt) throws SQLException,
ClassNotFoundException {
        //Declare statement as null
```

```
        Statement stmt = null;

        try {
            //Connect to DB (Establish SQLite  Connection)
            dbConnect();
            //Create Statement
            stmt = conn.createStatement();
            //Run executeUpdate operation with given sql statement
            stmt.executeUpdate(sqlStmt);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeUpdate operation : " + e);
            throw e;
        } finally {
            if (stmt != null) {
                //Close statement
                stmt.close();
            }
            //Close connection
            dbDisconnect();
        }
    }
}
```

```
SqliteConnection.java

package dbtools;


import java.sql.Connection;
import java.sql.DriverManager;

public class SqliteConnection {
    public static Connection connector(){
        try{
            Class.forName("org.sqlite.JDBC");
            Connection conn =
DriverManager.getConnection("jdbc:sqlite:/home/trojan/Desktop/Diplomatikh/CompanyTracker/db/development.sqlite3");
            return conn;
        }
        catch (Exception e){
            //ToDO : handle Exception
            e.printStackTrace();
            System.out.println(e);
            return null;
        }
    }
}
```

4 ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα παραπάνω αποτελούν το βασικό μέρος μιας τυπικής Restful web εφαρμογής που προσφέρει υπηρεσίες ιστού προς κατανάλωση από κάποιο πρόγραμμα πελάτη . Φυσικά οι δυνατότητες του Rails δεν σταματούν εδώ και είναι μάταιο για κάποιον και να προσπαθήσει να τις απαριθμήσει. Τα χαρακτηριστικά που μπορεί να προσφέρει είτε μέσω των gem modules που μπορούν να ενσωματωθούν σε αυτήν κατά το στάδιο της ανάπτυξης , αλλά και μετά απλά ενημερώνοντας το αρχείο Gemfile και εκτελώντας την εντολή boundle install είναι αμέτρητα . Από την ανάλυση απουσιάζουν features όπως JSON ,AJAX , CSS stylesheets , helpers , mailers , ασφάλεια και πολλά ακόμα που μπορούν να χαρακτηρίζουν το Rails ως ένα (full stack) περιβάλλον ανάπτυξης διαδικτυακών εφαρμογών η ανάλυση των οποίων ξεφεύγει των ορίων της διπλωματικής εργασίας .
Τέλος η τεχνολογία JavaFX αποτελεί την τελευταία και πιο ολοκληρωμένη επέκταση της τεχνολογίας Java. Μέχρι σήμερα η γλώσσα/τεχνολογία Java επέτρεπε τη δημιουργία εφαρμογών για κάθε είδους ηλεκτρονική συσκευή ,όπως για παράδειγμα τα κινητά τηλέφωνα , επεκτείνοντας τις δυνατότητές τους σε σημαντικό βαθμό. Δεν επέτρεπε όμως τη δημιουργία εφαρμογών που απαιτούσαν τη χρήση υψηλής ποιότητας video, γραφικών, ήχου και περιεχομένου. Η JavaFX εγκαθίσταται πάνω στην υπάρχουσα αρχιτεκτονική της Java, επιτρέποντας τη δημιουργία πολυμεσικών εφαρμογών υψηλής ποιότητας. Η αρχιτεκτονική της δομή περιλαμβάνει τα εξής μέρη: α) Το κοινό πλαίσιο εκτέλεσης των προγραμμάτων της Java (JVM) β) Τις απαραίτητες επεκτάσεις για την εκτέλεση πολυμεσικών εφαρμογών (JavaFX Runtime) γ) Το κοινό προγραμματιστικό περιβάλλον (API) όπου, με τη χρήση του, όλα τα προγράμματα μπορούν να εκτελούνται σε οποιαδήποτε συσκευή υποστηρίζει την πλατφόρμα δ) Προγραμματιστικές επεκτάσεις που αφορούν τη δημιουργία εφαρμογών για συγκεκριμένες συσκευές, λαμβάνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά τους ε) Το σετ των εργαλείων που θα κάνουν χρήση οι προγραμματιστές . Ως πλεονεκτήματα της πλατφόρμας αναφέρουμε το μηδενικό κόστος εγκατάστασης και χρήσης, την εκτενή βιβλιογραφία και τη συνεχή επέκταση των δυνατοτήτων της αφού υποστηρίζεται επίσημα από το φορέα της γλώσσας Java , την ORACLE .
Το 2ο κομμάτι της εφαρμογής αποτελεί μια τυπική εφαρμογή διαχείρισης μια βάσης δεδομένων υλοποιημένη με την JavaFX ,όπου γίνεται ξεκάθαρη η δυνατότητα επέκτασης των δυνατοτήτων της σε διαδικτυακές εφαρμογές οποιουδήποτε μεγέθους χάρη στη δυνατότητα της να συνυπάρχει αρμονικά με τις υπόλοιπες πλατφόρμες της γλώσσας , όπως για παράδειγμα με την JavaEE που κυριαρχεί σήμερα σε διαδικτυκές εφαρμογές μεγάλης κλίμακας και πολλαπλών επιπέδων

ανάπτυξης .

Τέλος σήμερα η διασύνδεση κάθε JAVAFX εφαρμογής και με τα εργαλεία που υπάρχουν σήμερα για την μεταφορά ολόκληρων προγραμμάτων java σε συσκευές κινητών τηλεφώνων , όπως για παράδειγμα το το JavaFXPorts της Gluon , καθιστούν την πλατφόρμα ως μία από τις χρησιμότερες που που μπορούμε να εκμεταλευτούμε για επιχειρισιακές εφαρμογές δικτύων .

5 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] JavaFX 8: Introduction by Example May 2014 Mark Heckler and Gerrit Grunwald

[2] Pro JavaFX 8: A Definitive Guide to Building Desktop,Mobile, and Embedder Java Clients by Weaver and Weiqi Gao

[3] Mastering JavaFX 8 Controls (Oracle Press) July 2014 by Hendrick Ebbers

[4] Ruby On Rails: Up and Running Aug 2006 by Bruce Tate and Curt Hibbs

[5] Ruby On Rails 5: Web App Development for Beginners Dec 2016 by Mark Smart

[6] The Ruby Programming language: Everything You Need To Know February 2008 by David Flanagan and Yukihiro Matsumoto

[7] Intro to java Programming , Comprehensive Version (9th Edition) by Y.Daniel Liang

[8] Head First Java 2nd Edition by Kathy Sierra and Bert Bates

ΠΑΡΑΡΤΗΜΑ Α (Χρήσιμες διευθύνσεις στο ίντερνετ)

- **http://rubyonrails.org/ Τα πάντα σχετικά με το Rails Frameworks**
- **https://www.railstutorial.org Tutorials για το Rails**
- **http://www.sinatrarb.com/ Τα πάντα σχετικά με το Sinatra**
- **http://www.java2s.com/ Μαθήματα Java – Java FX**
- **https://www.codecademy.com/ Μαθήματα Ruby – Java – JavaFX**
- **https://netbeans.org/**
- **http://gluonhq.com/ Scheme Builder and JavaFXPorts**

ΠΑΡΑΡΤΗΜΑ Β (Εργαλεία που χρησιμοποιήθηκαν για την εργασία)

Ruby On Rails v5.0

Java 8 SE

JavaEE

Java FX 8

Netbeans 8.2

LibreOffice 5.3

Gedit

Emacs

Gluon Sceme Builder Ver

Ubuntu 16.02 LTS