Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Ιστοσελίδας Τουριστικού Οδηγού

ΜΠΡΑΧΙΜΑΙ ΣΕΡΚΙΝΤ ΑΜ: 3036

Επιβλέπων Καθηγητής: Παπαδάκης Νικόλαος

ΗΡΑΚΛΕΙΟ

2017

Ευχαριστίες

Οι πρώτοι άνθρωποι που πρέπει να ευχαριστήσω για την μέχρι τώρα πορεία μου και για όλα τα πράγματα που έχω καταφέρει είναι οι γονείς μου και η αδερφή μου που μου στάθηκαν με όλες τους τις δυνάμεις ώστε να εκπληρώσω τα όνειρά μου.

Στην συνέχεια θέλω να ευχαριστήσω συγκεκριμένα τους φίλους μου Κωσταντίνο Γκεντζογλάνη, Αλέξανδρο- Απόλλων Ροδόπουλο και Μαρία Κολιαβασίλη που όταν είχα ανάγκη μου σταθήκανε σαν πραγματικοί φίλοι.

Πέρα από τα παιδιά θέλω να ευχαριστήσω όλους τους υπόλοιπους μου φίλους μου που με τον τρόπο τους με βοήθησαν και αυτοί.

Τέλος θέλω να ευχαριστήσω τους καθηγητές του ΤΕΙ για τις γνώσεις που μου προσφέρανε και πιο συγκεκριμένα τον κύριο Παπαδάκη που μου εμπιστεύτηκε την πτυχιακή αυτή.

Abstract

The purpose of this thesis is to develop a booking platform for various cities. The customer will be able to choose from a selection of areas hotel categories room capacities and optionally car or motorbikes he would like to rent and make a reservation. Additionally the customer will be provided with information about any tourist attractions or any extra trips he might want to take in the area.

Σύνοψη

Σκοπός αυτής της πτυχιακής εργασίας είναι η δημιουργία μια ιστοσελίδας που θα αποτελεί έναν τουριστικό οδηγό για διάφορες περιοχές. Ο χρήστης θα μπορεί να επιλέγει ξενοδοχείο, δωμάτιο για μια συγκεκριμένη ημερομηνία ή ημερομηνίες. Προαιρετικά θα μπορεί επίσης να νοικιάσει κάποιο αυτοκίνητο. Επιπροσθέτως, ο χρήστης θα μπορεί να πάρει πληροφορίες για τις διάφορες εκδρομές καθώς και τα τουριστικά αξιοθέατα που υπάρχουν στην περιοχή.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Κεφάλαιο 1: Εισαγωγή

1.1 Περίληψη

Η παρούσα πτυχιακή εργασία καταπιάνεται με την υλοποίηση ιστοσελίδας – τουριστικού οδηγού για ένα σύνολο περιοχών. Πρόκειται για μια ιστοσελίδα που με εύκολο τρόπο ένας χρήστης θα μπορέσει να κλείσει δωμάτιο από κάποιο ξενοδοχείο να ενοικιάσει κάποιο αυτοκίνητο από μια υπηρεσία ενοικίασης αυτοκινήτων και τέλος θα μπορέσει να πάρει πληροφορίες σχετικά με τα τουριστικά αξιοθέατα αλλά και τις διάφορες εκδηλώσεις, εκδρομές οι οποίες είναι διαθέσιμες στην εκάστοτε περιοχή.

Για την υλοποίηση της ιστοσελίδας – οδηγού θα χρησιμοποιηθεί για το front-end κομμάτι της Html, JavaScript και CSS και πιο συγκεκριμένα το framework Bootstrap. Για το back-end κομμάτι θα χρησιμοποιηθεί Java και πιο συγκεκριμένα το Spring MVC Framework. Για την αποθήκευση θα χρησιμοποιηθεί μια βάση δεδομένων σε Postgre.

1.2 Σκοπός και Στόχος

Σκοπός και στόχος της πτυχιακής εργασίας αυτής είναι η δημιουργία και η κατασκευή μιας ιστοσελίδας με την χρήση μοντέρνων frameworks και παραθέτοντας δυσκολίες και κολλήματα τα οποία μπορεί να προέκυψαν κατά την δημιουργία της σελίδας. Χρησιμοποιώντας τις νέες αυτές τεχνολογίες και frameworks θα μάθουμε αρκετά για το πώς μια ιστοσελίδα αναπτύσσετε σε ένα λίγο πιο επαγγελματικό επίπεδο.

1.3 Δομή της Εργασίας

Κεφάλαιο 1 : Εισαγωγή

Κεφάλαιο 2: Μεθοδολογία

Κεφάλαιο 3: Σχέδιο Δράσης και Τεχνολογίες

Κεφάλαιο 4: Υλοποίηση

Κεφάλαιο 5: Αποτελέσματα

Κεφάλαιο 2: Μεθοδολογία Υλοποίησης

Για την υλοποίηση της σελίδας θα λειτουργήσουμε με τον εξής τρόπο:

Αρχικά θα αναγνωρίσουμε ποιες θα είναι οι βασικές οντότητες που αφορούν την ιστοσελίδα. Στην συνέχεια αφού έχουμε αναγνωρίσει ποιες είναι οι βασικές οντότητες θα πάμε να σχεδιάσουμε την βάση δεδομένων πάνω στην οποία θα χτιστεί η εφαρμογή μας. Ο σχεδιασμός βέβαια μπορεί να αλλάξει καθώς δεν μπορούμε να είμαστε 100% σίγουροι τι θέματα μπορεί να προκύψουν.

Στην συνέχεια ανάλογα με τις απαιτήσεις της σελίδας ξεκινάμε και σχεδιάζουμε τις σελίδες όσον αφορά το γραφικό τους περιβάλλον δηλαδή τις θέσεις τον κουμπιών καθώς και τι στοιχεία θα περιλαμβάνουν οι φόρμες και τα διάφορα sections της σελίδας.

Τέλος αναπτύσσουμε τον server και παράλληλα αναπτύσσουμε και τις λειτουργίες στο front end της σελίδας τροποποιώντας την βάση η οτιδήποτε άλλο που είχαμε αναλύσει κατά την σχεδίαση της εφαρμογής.

2.1 Αλγόριθμοι

Ο πιο σημαντικός αλγόριθμος είναι αυτός που αναλαμβάνει να εξετάσει την διαθεσιμότητα των στοιχείων που αναζήτα ο χρήστης και λειτουργεί με τον εξής τρόπο:

Αφού βρούμε το ξενοδοχείο καθώς και το δωμάτιο το οποίο προκύπτει από τον έλεγχο της κατηγορίας του ξενοδοχείου αλλά και την χωρητικότητα και δωματίου ελέγχουμε αν το δωμάτιο διαθέτει κρατήσεις. Αν το δωμάτιο δεν εχει κρατήσεις τότε είναι εντάξει, αν όμως διαθέτει κρατήσεις κοιτάμε να δούμε αν η χρονική περίοδος που ζητάει ο χρήστης το δωμάτιο επικαλύπτει κάποια από αυτές των ήδη υπαρχουσών κρατήσεων.

Κεφάλαιο 3: Σχέδιο Δράσης Για την Εκπόνηση της Εργασίας-State of the Art

Για την υλοποίηση της εργασίας θα χρησιμοποιηθούν νέες τεχνολογίες και frameworks τόσο στο front end όσο και το backend. Για το front end θα χρησιμοποιήσουμε για τα βασικά HTML CSS και Javascript και πιο συγκεκριμένα το framework Bootstrap.

Στο backend από την άλλη θα χρησιμοποιήσουμε Java και πιο συγκεκριμένα το Spring MVC.

Java

Εικόνα 1

Ταυτότητα της Java

Η Java είναι μια γλώσσα αντικειμενοστραφούς προγραμματισμού ανοικτής αρχιτεκτονικής και ανήκει στην κατηγορία web programming. Είναι γλώσσα παραγωγής και θεωρείται ως θυγατρική της γλώσσας C++. Διαθέτει αρκετά στοιχεία και από άλλες γλώσσες όπως είναι η Mesa, η Lisp αλλά και βασικά στοιχεία της Algol και της Fortran. Αρχικά, χρησιμοποιήθηκε ως μέρος ενός ερευνητικού έργου για την ανάπτυξη λογισμικού για ηλεκτρονικές συσκευές. Ο τρόπος αυτός χρησιμοποίησης της την έκανε ιδανική για την διανομή εκτελέσιμων προγραμμάτων μέσω του internet αλλά και σε γλώσσα προγραμματισμού για ανάπτυξη λογισμικού που θα μπορούσε να μεταφερθεί και να λειτουργήσει ανεξάρτητα από τα διαφορετικά λειτουργικά συστήματα.

Ιστορική Αναδρομή

OAK

Η ανάπτυξη της Java ξεκίνησε στις αρχές του 1991 από την εταιρία Sun κυρίως από το ερευνητικό έργο των J. Gosling, E.Frank, M. Sheridan και C. Warth. Μετά από έρευνα περίπου 2 ετών κυκλοφόρησαν την γλώσσα OAK που είχε ως βασικό χαρακτηριστικό της την δυνατότητα ανάπτυξης προγραμμάτων για οικιακές ηλεκτρονικές συσκευές όπως είναι οι φούρνοι μικροκυμάτων, τηλεχειριστήρια τηλεοράσεων κ.α. Η OAK ήταν η πρώτη έκδοση της Java. Η ονομασία OAK λέγεται ότι επιλέχθηκε λόγω μιας βελανιδιάς (oak) που υπήρχε έξω από το γραφείο του Gosling.

Java 1

Το 1995 ο J. Gosling επέκτεινε τα σχέδια που είχε κάνει με στόχο την δημιουργία γλώσσας αντικειμενοστραφούς προγραμματισμού. Έτσι, το 1996, γεννήθηκε η Java. Το όνομα επιλέχθηκε μετά από μια σύσκεψη της ομάδας, καθώς η ονομασία OAK ήταν ήδη κατοχυρωμένη από την Oak Technologies. Σύμφωνα με τον Gosling η ομάδα ήταν ανάμεσα στην ονομασία **Silk** και στην ονομασία **Java** και τελικά προτιμήθηκε η Java καθώς φάνταζε τότε ως πιο μοναδική. Η ονομασία προέρχεται από ένα νησί στην Ινδονησία από όπου προέρχεται και η ομώνυμη ποικιλία καφέ που σύμφωνα με τις φήμες ήταν η ποικιλία καφέ που έπιναν στην σύσκεψη εκείνη.

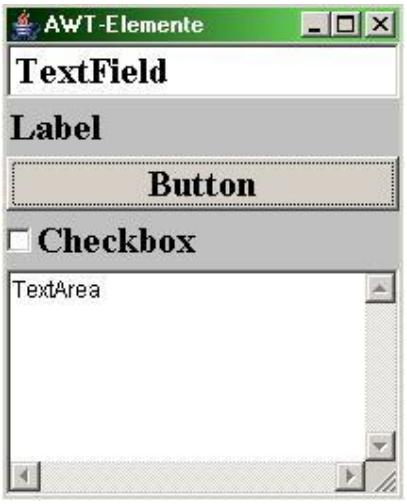
Η πρώτη σταθερή έκδοση της Java ήταν η έκδοση JDK 1.0.2 η οποία λέγεται και Java 1

Η Java γνώρισε ακαριαία αναγνώριση για την ανάπτυξη λογισμικού και εφαρμογών που λειτουργούσαν ανεξάρτητα από το περιβάλλον που έτρεχαν αλλά και τις αρχιτεκτονικές με τις οποίες ήταν κατασκευασμένα τα περιβάλλοντα αυτά.

JDK 1.1

Έναν χρόνο περίπου αργότερα κυκλοφόρησε η έκδοση JDK 1.1. Στην έκδοση αυτή:

- έγιναν εκτενείς αναδιαμορφώσεις στο AWT. Το AWT είναι από τις βασικές κλάσεις της Java και αποτελεί το πιο βασικό εργαλείο για την δημιουργία περιβάλλοντος χρήστη σε ένα πρόγραμμα Java “”.

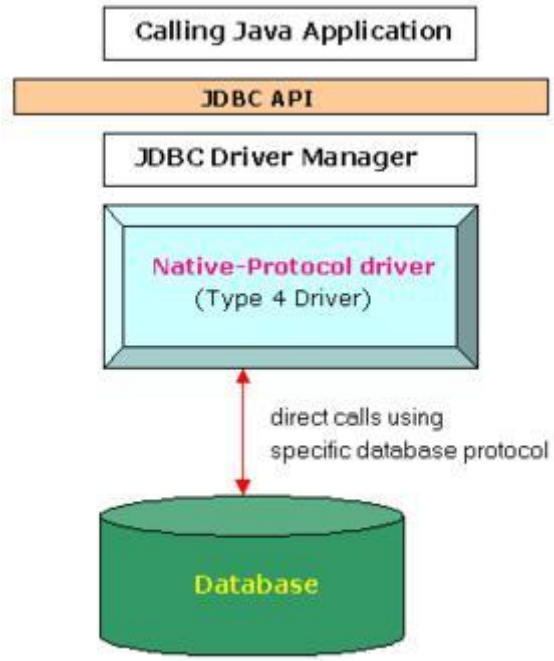


Εικόνα 2

- προστέθηκε στην γλώσσα το στοιχείο των εμφωλευμένων κλάσεων (nested classes). Εμφωλευμένες είναι οι κλάσεις που έχουν οριστεί μέσα σε μια άλλη κλάση. Όσες εμφωλευμένες κλάσεις είναι και μη στατικές ονομάζονται αλλιώς και **inner classes.**

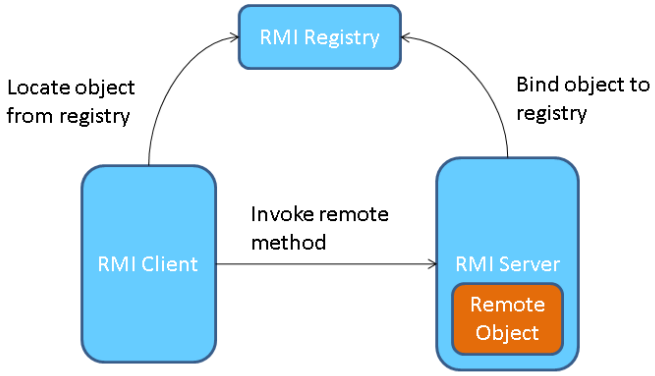
- Προστέθηκαν τα JavaBeans. Τα JavaBeans είναι κλάσεις της Java που ακολουθούν μια συγκεκριμένη σύμβαση. Οι κλάσεις αυτές πρέπει να διαθέτουν έναν constructor χωρίς παραμέτρους, να είναι serializable και να διαθέτει getters kai setters για κάθε μεταβλητή που διαθέτει.

- προστέθηκε το **JDBC.** Το Java Database Connectivity είναι ένα API το οποίο προσδιορίζει πως ένα πρόγραμμα πελάτης (client) θα πρέπει έχει πρόσβαση η να διαχειριστεί μια βάση δεδομένων. Το JDBC παρέχει μεθόδους ώστε ο χρήστης να κάνει αιτήματα προς την βάση αλλά και να ενημερώνει τα δεδομένα σε αυτή. Το JDBC προσανατολίζεται κυρίως σε σχεσιακές βάσεις δεδομένων.



Εικόνα 3

- προστέθηκε το **RMI**. Το Java Remote Method Invocation είναι ένα API το οποίο πραγματοποιεί απομακρυσμένες κλήσεις σε μεθόδους το αντικειμενοστραφές αντίστοιχο το RPC (remote procedure call), ουσιαστικά ένα πρόγραμμα μπορεί να εκτελεί μεθόδους οι οποίες βρίσκονται σε άλλες διευθύνσεις.



Εικόνα 4

- προστέθηκε επίσης ένα είδος **αντανακλασης. Η αντανακλαση είναι η δυνατότητα ενός προγράμματος να**

- το **JIT** (Just in Time) προστέθηκε επίσης. Το JIT είναι ένας τρόπος μετάφρασης/μεταγλώττισης (compile/compilation) ο οποίος γίνεται δυναμικά κατά την εκτέλεση του προγράμματος και όχι πιο πριν όπως συμβαίνει συνήθως. Στην έκδοσή αυτή προστέθηκε αυτός ο compiler για πλατφόρμες Microsoft Windows που είχαν φτιαχτεί για την JavaSoft από την Symantec

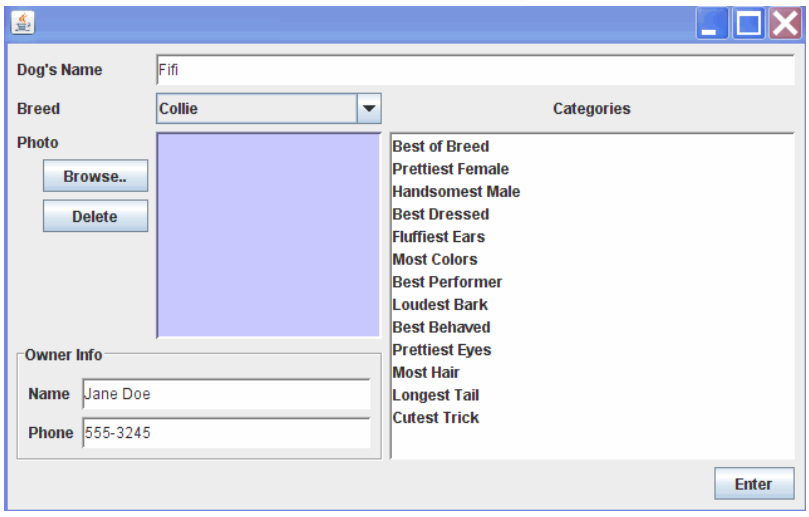
- τέλος στην έκδοση αυτή έγινε μια γενικότερη «διεθνοποίηση» καθώς προστέθηκε υποστήριξη Unicode από την Taligent. Το Teligent είναι το όνομα ενός αντικειμενοστραφούς λειτουργικού συστήματος και της εταιρίας αντίστοιχα που το δημιούργησε.

Java 2

Η έκδοση J2SE 1.2 καθώς και οι εκδόσεις που ακολούθησαν μέχρι την J2SE 5.0 ονομάστηκαν εκ των υστέρων **Java 2.** Η ονομασία των εκδόσεων άλλαξε από JDK σε J2SE για να μπορέσει να υπάρξει διάκριση ανάμεσα στις δυο βασικές πλατφόρμες της γλώσσας την **Standard Edition (SE)** και την **Enterprise Edition (EE).** Η έκδοση αυτή ήταν από τις σημαντικότερες εκδόσεις της Java καθώς τριπλασίασε το μέγεθός της πλατφόρμας της σε 1520 κλάσεις και σε 59 πακέτα. Οι πιο σημαντικές προσθήκες ήταν:

- η λέξη κλειδί **strictfp.** Η stictfp είναι μια λέξη κλειδί που χρησιμοποιείται στην Java για να περιορίσει τους υπολογισμούς κινητής υποδιαστολής για να διασφαλίσει την φορητότητα.

- Προστέθηκε το **Swing** στον πυρήνα της Java. Το Swing είναι μια εργαλειοθήκη που επιτρέπει στον προγραμματιστή να δημιουργήσει μια διεπαφή για το προγραμμά του. Το Swing αναπτύχθηκε για να προσφέρει πιο εκλεπτυσμένα στοιχεία για την δημιουργία διεπαφών από αυτά που πρόσφερε το AWT. Έχει πιο ισχυρά και πιο ευέλικτα στοιχεία από το AWT και πέρα από γνωστά κουμπιά, check boxes και τις ετικέτες προσφέρει και πιο προχωρημένα στοιχεία όπως είναι panels τα οποία διαθέτουν tabs, λίστες, πίνακες κτλ. Επίσης σε αντίθεση με το AWT τα στοιχεία για την δημιουργία των διάφορων διεπαφών στο Swing, είναι ανεξάρτητα από το περιβάλλον στο οποίο τρέχουν.



Εικόνα 5

- το JVM της Sun για πρώτη φορά εξοπλίστηκε με έναν JIT compiler

- προστέθηκαν τα Java Plug-Ins.

- Προστέθηκε το **Java IDL**. Το Java IDL είναι μια «εκτέλεση» της αρχιτεκτονικής **COBRA** που επιτρέπει σε μια εφαρμογή να επικοινωνεί με επιμέρους ετερογενή αντικείμενα ή εφαρμογές.

- προστέθηκε το **Collections Framework.** Το Collections Framework είναι ένα set από κλάσεις και interfaces το οποίο υλοποιεί data structures όπως είναι οι λίστες οι τα maps. Αν και αναγνωρίζεται ως ένα framework λειτουργεί ουσιαστικά σαν βιβλιοθήκη.

J2SE 1.3

Η έκδοση **J2SE 1.3** που είχε και την κωδική ονομασία **Kestrel.** Η έκδοση αυτή εκδόθηκε στις 8 Μαΐου του 2000 και οι πιο σημαντικές προσθήκες που είχε ήταν:

- προστέθηκε το HotSpot JVM. Το HotSpot JVM είναι ένα virtual machine για desktops αλλά και servers αντίστοιχα, το οποίο προσφέρει βελτιωμένη απόδοσή μέσα από μεθόδους όπως είναι το JIT και η προσαρμοστικότητα.

- Το **RMI** της Java τροποποιήθηκε για να προσφέρει προαιρετική συμβατότητα με το **COBRA**.

- Το **JNDI (Java Naming and Directory Interface)** προστέθηκε στον πυρήνα κλάσεων της Java. Το JNDI είναι ένα Java API για την δημιουργία υπηρεσίας καταλόγου το οποίο επιτρέπει σε ένα πρόγραμμα Java να ανακαλύψει η να αναζητήσει αντικείμενα μέσω των ονομάτων τους. Οι κυριότερες χρήσεις του JDNI είναι η σύνδεση ενός προγράμματος Java σε μια εξωτερική υπηρεσία καταλόγου καθώς και να επιτρέπει σε servlets να αναζητούν πληροφορίες διάταξης που βρίσκεται σε ένα web container.

- Προστέθηκε το **JPDA.** Το Java Platform Debugger Architecture είναι μια συλλογή από APIs τα οποία κάνουν debug τον κώδικα και περιλαμβάνει:

    - Το Java Debugger Interface – το οποίο προσφέρει ένα υψηλού επιπέδου Java περιβάλλον το οποίο οι προγραμματιστές μπορούν εύκολα να χρησιμοποιήσουν για να αναπτύξουν προγράμματα για debugging.
    - Το Java Virtual Machine Tools Interface, το οποίο είναι μια διεπαφή η οποία βοηθάει να επιθεωρείς το στάδιο και να ελέγχεις την εκτέλεση των διάφορων εφαρμογών που τρέχουν σε ένα JVM.

- Προστέθηκε το Java Sound. Το Java Sound είναι ένα χαμηλού επιπέδου API το οποίο σου επιτρέπει να ελέγξεις την εισαγωγή και την παραγωγή δεδομένων ήχου, συμπεριλαμβανομένου και MIDI αρχείων δεδομένων. Το API αυτό καλύπτει ένα μεγάλο εύρος των αναγκών που έχουν οι προγραμματιστές και μπορούν να αναπτύξουν με την βοηθειά του framework για επικοινωνία όπως είναι οι τηλεδιασκέψεις και η τηλεφωνική επικοινωνία, δημιουργία περιεχομένου αλλά και διόρθωση του περιεχομένου αυτό

- Προστέθηκαν οι συνθετικές κλάσεις στην Java.

J2SE 1.4

Τον Φεβρουάριο του 2002 με την κωδική ονομασία Merlin εκδόθηκε η πρώτη πλατφόρμα Java που αναπτύχθηκε από το Community της Java. Οι σημαντικότερες αλλαγές περιλαμβάνουν.

- Αλλαγές στην γλώσσα. Προστέθηκε η λέξη κλειδί **assert** η οποία χρησιμοποιείται για να δηλώσει μια αναμενόμενη δυαδική συνθήκη σε ένα προγραμμα.

- Τα **regular expressions,** που στην επιστήμη υπολογιστών αναφέρονται ως ένα μια ακολουθία χαρακτήρων που δημιουργούν ένα μοτίβο αναζήτησης, το οποίο μοντελοποιήθηκε με βάση την Perl.

- Έγινε αλυσιδοποίηση των exceptions που ουσιαστικά επέτρεπε σε exception να μπορουν να εμφωλεύουν εσωτερικά άλλα μικρότερου επιπέδου exceptions.

- προστέθηκε υποστήριξη για τα δικτυακό πρωτόκολλο **IPv6**.

- προστέθηκε το **non-blocking IO** το οποίο είναι μια σύνθεση από διαφορα APIs τα οποία προσφέρουν διάφορα στοιχεία για εισοδο/έξοδο δεδομένων.

- προστέθηκε Logging API το οποίο επιτρέπει την δημιουργία ενός αρχείου καταγραφής η να γίνεται καταγραφή συμβάντων σε ένα συγκεκριμένο μέρος.

- APIs εισόδου/εξοδου εικονών τα οποία επιτρέπουν την ανάγνωση αλλά και εγγραφή εικόνων σε format όπως είναι το JPEG η το PNG.

- Ενσωματωμένος XML parser και XSLT διερμηνεα.

- Ενσωματωμένη ασφάλεια και επεκτάσεις κρυπτογραφίας όπως είναι το JCE, JSSE αλλά και το JAAS.

- Προστέθηκε το **Preferences API.** Το πακέτα του API αυτού προσδιορίζουν έναν τρόπο με τον οποίο με τον οποίο η εφαρμογή για μπορέσει να επκοινωνήσει με το περιβάλλον έτσι ώστε να μπορέσει να διαβάσει διάφορα δεδομένα οι ρυθμίσεις που έχουν γίνει από τον σύστημα η από τον χρήστη.

J2SE 5.0 (1.5)

Στις 30 Σεπτέμβριου του 2004 έχουμε την έκδοση του J2SE 5.0 με την κωδική ονομασία Tiger. Ενώ εσωτερικά η αρίθμηση της έκδοσης παραμένει 1.5 επιλέχτηκε αυτό να αλλάξει σε 5.0 για να αντικατοπτρίζει καλύτερα το επίπεδο ωριμότητας, σταθερότητας, ασφάλειας και ικανότητας κλιμάκωσης που είχε. Κάποιες από τις κυριότερες προσθήκες στην έκδοση αυτή είναι:

- προσθήκη των Generics. Τα Generics επέκτειναν το type system που χρησιμοποιούσε μέχρι τότε η Java και επέτρεψαν σε διαφορετικους τύπους μεθόδων να ενεργούν πάνω σε διαφορετικους τύπου προσφέροντας παραλλήλα ασφάλεια κατά την διάρκεια της εκτέλεσης του προγράμματος.

- Προστέθηκαν τα **Metadata.** Τα metadata ή αλλιώς υποσημειώσεις επιτρέπουν στον προγραμματιστή να προσθέσει σε διάφορες κλάσεις περισσότερες «ετικέτες» πληροφορίας που αυτές αργότερα θα μπορούσαν να διαβαστούν από κάποιο πρόγραμμα και να ερμηνευτούν ανάλογα.

- Προστέθηκε το **autoboxing/unboxing.** Η λειτουργία αυτή επέτρεπε στην μετατροπή και συσχέτιση των πρωτογόνων (primitive) τύπων με τους αντίστοιχες primitive wrapper κλάσσεις.

- προστέθηκε τα **Enumerations.** Τα enumerations και συγκεκριμένα η λέξη κλειδί enum σου επιτρέπει να δημιουργήσεις έναν κατά παραγγελία τύπο δεδομένων κατασκευάζοντας στην ουσία μια ταξινομημένη λίστα.

- προστέθηκαν τα **varargs.** Στην ουσία η τελευταία παράμετρος μια μεταβλητής μπορεί να δηλωθεί με τον τύπο και 3 τελείες και στην συνέχεια όταν γίνει κλήση αυτής της μεθόδου μπορούμε να βάλουμε όσες μεταβλητές του ίδιου τύπου θέλουμε στην συγκεκριμένη θέση.

- Η δομή επανάληψης **for each** έγινε πιο ισχυρή καθώς επεκτάθηκε η σύνταξή της ώστε να μπορεί να προσπελάσει δεδομένα από διάφορους τύπους array όπως είναι τα Iterable όπως είναι πχ τα Collections

- Βελτίωση στην εκτέλεση multi – thread προγραμμάτων Java καθως το καινούργιο μοντέλο Java απευθύνεται σε θέματα πολυπλοκότητας αποτελεσματικότητας και απόδοσης.

- προστέθηκαν τα **Static Imports.** Τα static imports είναι ένα χαρακτηριστικό το οποίο συστήθηκε στην Java και επιτρέπει την δήλωση μεθόδων ή μελών σε μια κλάση τα χωρίς να χρειάζεται να προσδιορίσουμε σε ποια κλάση είναι αυτά γραμμένα.

Η Java 5 (1.5) είναι η τελευταία έκδοση της Java που έτρεχε σε Windows 95 και τα Windows NT 4.0.

Java SE 6

Ακολουθώντας την αλλαγή στην διαδικασία της ονομασίας των εκδόσεων της Java η Sun κατάργησε το “J2SE” και έμεινε με το Java SE. Επίσης καταργήθηκε το δεκαδικό στοιχείο στην αρίθμηση των εκδόσεων οπότε το 6.0 έγινε 6. Ακόμα συνεχίζοντας την αλλαγή της αρίθμησης που έγινε στην προηγούμενη έκδοση το 6 στο Java SE 6 αντιπροσωπεύει την έκδοση 1.6 στην ουσιαστική αρίθμιση.

Η Java SE 6 κυκλοφόρησε στις 11 Δεκεμβρίου του 2006 και περιελάβανε τις εξής κύριες αλλαγές:

- καταργήθηκε η υποστήριξη για παλαιότερες εκδόσεις Win9x ανεπίσημα. Το 7^{ο} update της Java 6 ήταν το τελευταίο που λειτουργούσε σε αυτές τις εκδόσεις Windows.

- Σημαντικές αλλαγές στην απόδοση της κυρίας πλατφόρμας καθώς και στο Swing.

- Βελτιωμένη απόδοση στα web services μέσω του **JAX-WS.** Το JAX-WS ή αλλιώς Java API for XML web services είναι ένα API της Java που σου επιτρέπει την δημιουργία web services.

- Προστέθηκε υποστήριξη για το JDBC 4.0

- Έγιναν πολλές βελτιώσεις στην ικανότητα δημιουργίας διεπαφών με την προσθήκη στο API του SwingWorker.

- Βελτιώσεις στο JVM συμπεριλαμβανομένου του βελτιστοποίησης του compiler, νέοι αλγόριθμοι και βελτιώσεις στις ηδη υπάρχοντες για την συλλογή «σκουπιδίων».

Java SE 7

Τον Ιούλιο του 2011 εκδόθηκε η 7ή έκδοση της Java με την κωδική ονομασία Dolphin. Η έκδοση αυτή ήταν διαθέσιμη στους προγραμματιστές στις 28 Ιουλίου της ίδιας χρονιάς. 8 αναβαθμίσεις εκδόθηκαν για την έκδοση αυτή . Προσθήκες στην Java 7 περιλαμβάνουν:

- Το JVM υποστήριζε δυναμικές γλώσσες ακολουθώντας το παράδειγμα που γινόταν στο Multilanguage Virtual Machine.

- Συμπεσμένοι 64bit δείκτες.

- Μικρές αλλαγές στην στην γλώσσα που μερικές από αυτές ήταν ότι υπήρχαν string σε switch, αυτόματη διαχείριση πόρων σε try-catch, απλοποιημένη δήλωση varargs και πιάσιμο περισσότερων exceptions με βελτιωμένο ελεγκτή.

- Νέα βιβλιοθήκη για την είσοδο/έξοδο αρχείων για πολλαπλά συστήματα αρχείων.

- προστέθηκε το **Timsoft** για την ταξινόμηση ενός array από αντικείμενα. Το TImsoft είναι ένας αλγόριθμος ταξινόμησης που πηγάζει από την ταξινόμηση συγχώνευσης και την ταξινόμηση εισαγωγής και είναι σχεδιασμένο για να αποδίδει πολύ καλά σε πολλά δεδομένα του πραγματικού κόσμου.

- Υποστήριξη στο επίπεδο βιβλιοθήκης για τον αλγόριθμο **elliptic curve cryptography.** Ο αλγόριθμος αυτός είναι πια προσέγγιση στην κρυπτογραφία δημόσιου κλειδιού βασισμένο στην αλγεβρική κατασκευή ελλειπτικών καμπυλών πάνω σε ορισμένα πεδία. Ο αλγόριθμος αυτός χρειάζεται ουσιαστικά ένα μικρότερο κλειδί σε σχέση με άλλους αλγορίθμους για να προσφέρει τον ίδιο αριθμό ασφαλείας.

- ένα “σωλήνα» **XRender** για το Java 2D το οποίο βελτίωνε την διαχείριση των χαρακτηριστικών τα οποία ήταν συγκεκριμένα στις καινούργιες κάρτες γραφικών.

- Βελτιωμένη βιβλιοθήκη για τα καινούργια πρωτόκολλα δικτύου συμπεριλαμβανομένου το SCTP και το SDP.

- Βελτίωση στην χρήση XML και βελτιώσεις στο Unicode.

Java SE 8

Στις 18 Μαρτίου του 2014 κυκλοφόρησε η Java 8 με την κωδική ονομασία **Spider,** και περιλάμβανε κάποια στοιχεία τα οποία αρχικά είχαν προγραμματιστεί να υπάρχουν στην Java 7 αλλά τελικά αναβλήθηκαν. Οι σημαντικότερες αλλαγές που έγιναν σε αυτή την έκδοση ήταν οι εξής:

- προσθήκη υποστήριξης στην γλώσσα για τις εκφράσεις **lambda**. Lambda συναρτήσεις ή ανώνυμες συναρτήσεις ονομάζονται οι συναρτήσεις οι οποίες περιέχουν το σώμα της συνάρτησης αλλά δεν περιέχουν όνομα. Οι συναρτήσεις αυτές χρησιμοποιούνται συνήθως για να περαστούν ως παράμετροι σε συναρτήσεις μεγαλύτερου βαθμού ή ως αποτέλεσμα μιας συνάρτησης μεγαλύτερου βαθμού. Η προσθήκη των lambda συναρτήσεων επέτρεψε στην προσθήκη μεθόδων σε interfaces χωρίς να χρειάζεται να υλοποιηθούν ξανά.

- Η προσθήκη ενός runtime με το όνομα Nashom το οποίο επιτρέπει σε προγραμματιστές την ενσωμάτωση κώδικα JavaScript μέσα στις εφαρμογές.

- προσθήκη μη προσημασμένων ακεραίων.

- προσθήκη API για Ημερομηνίες και Χρόνο (Date and Time)

- απευθείας εκτέλεση εφαρμογών JavaFX.

Η Java 8 δεν υποστηριζόταν στα Windows XP μέχρι να γίνει το update 25. Οι προηγούμενες εκδόσεις για να εγκατασταθούν έπρεπε να αποσυμπιεστούν τα αρχεία χειροκίνητα στον φάκελο της εγκατάστασης.

Μελλοντικές Εκδόσεις

Το 2011 σε συζητήσεις που έγιναν στο JavaOne ή Oracle σχεδίαζε να την έκδοση Java 9 στις αρχές του 2016. Το 2016 ή ημερομηνία άλλαξε για τον Μάρτιο του 2017 μετά για τον Ιούλιο του 2017 και εν τέλει η Java 9 με τα πλήρη χαρακτηρίστηκα της θα είναι διαθέσιμη στις 21 Σεπτεμβρίου του 2017.

Πλεονεκτήματα της Java

Τα πλεονεκτήματα για την χρήστη της Java στην ανάπτυξη του server side κομματιού της εφαρμογής είναι πολλά.

Αρχικά η Java είναι μια οικία γλώσσα. Όπως αναφέρθηκε και στην αρχή του κεφαλαίου η Java είναι θυγατρική γλώσσα της C++ αλλά τα στοιχεία και «ικανότητες» που έχει είναι από γλώσσες όπως Smalltalk, Fortran, Lisp, Mesa και Cedar κ.α. Κάτι τέτοιο σημαίνει ότι η Java δεν είναι μόνο σε προγραμματιστές που ξέρουν C++ αλλά και σχεδόν σε όλους τους προγραμματιστές.

Η Java είναι επίσης πολύ απλή. Οι δομές που απαιτούν αυστηρό ορισμό είναι πολύ λίγες. Έχει πολλούς αυτοματισμούς όσον αφορά την διαχείριση μνήμης. Στην Java έχει αποφευχθεί η πολλαπλή κληρονομικότητα καθώς οι περισσότερες υποκλασεις της είναι υποκλάσεις ακριβώς μιας άλλης κλάσης. Επιπλέον κάτι το οποίο η java μπόρεσε να αποτάξει από πάνω της ήταν η αριθμητική δεικτών κάτι το οποίο έχει παρατηρηθεί και είναι πραγματικά άσκοπο στην C++.

Σε επίπεδο δικτύου η Java είναι αρκετά κατανεμημένη. Επιτρέπει να κάνουμε προσπέλαση αντικειμένων που βρίσκονται σε διάφορες κορυφές μέσα στο δίκτυο. Όσον αφορά το επίπεδο των servers η Java μπορεί να χρησιμοποιηθεί για την ανάπτυξη προγραμμάτων που ονομάζονται servlets τα οποία μπορούν υποστηρίξουν ολοκληρωμένα συστήματα που αντλούν δεδομένα από κάποιες βάσεις δεδομένων μέσα από το διαδίκτυο εξυπηρετώντας τις ανάγκες του κάθε χρήστη.

Ακόμα ένα πλεονέκτημα της Java είναι η φορητότητα της και η οικουμενικότητά της. Ο ενδιάμεσος κώδικας που μετατρέπει ο μεταγλωττιστής την Java είναι ουσιαστικά ίδιος σε κάθε πλατφόρμα η σύστημα το οποίο τρέχει Java η μόνη διαφορά έγκειται στον διερμηνέα ο οποίος είναι ενσωματωμένος στον αντίστοιχο JVM που βρίσκεται στην πλατφόρμα. Κάτι τέτοιο επιτρέπει την εκτέλεση προγραμμάτων Java μέσα από browsers αλλά σε επίπεδο διαδικτύου καθώς όλα σχεδόν τα προγράμματα αυτά έχουν ενσωματωμένο ένα JVM. Μια τέτοια μέθοδος βέβαια δεν είναι κάποια καινοτομία καθώς ήταν γνωστή για πολλά χρόνια, αλλά η Java μέσω τον JIT (Just in time compiler). Κάτι τέτοιο την κάνει να είναι μεταφέρσιμη στις περισσότερες αρχιτεκτονικές και στα περισσότερα λειτουργικά συστήματα αυτή την στιγμή όπως είναι τα Ms-Office, MacOS, Linux αλλά και στα περισσότερα endpoints του διαδικτύου.

Ένα άλλο σημαντικό πλεονέκτημα της Java που μπορεί να σημειωθεί είναι ότι πρόκειται για μια από τις πιο ασφαλείς γλώσσες προγραμματισμού που υπάρχουν. Διαθέτει σύστημα που ελέγχει τον ενδιάμεσο κώδικα για αυτοκατάρρευση του προγράμματος αλλά και γενικά κατάρρευση κεντρικών κομματιών του υπολογιστικού συστήματος. Υπάρχει επίσης η διάθεση υποσυστήματος δικαιωμάτων πρόσβασης χρηστών σε εφαρμογές διαδίκτυο χωρίς να υπάρχουν ανεπιθύμητες παρενέργειες στο σύστημα. Υπάρχει αυτόματο σύστημα διαχείρισης της μνήμης και τέλος μπορεί και κάνει object oriented exception handling που είναι ειδικό στο να προλαμβάνει λάθη τα οποία μπορεί να συμβούν κατά την διαδικασία της εκτέλεσης.

Η δυναμικότητα της Java είναι επίσης ένα από τα πολύ σημαντικά πλεονεκτήματα της. Διαθέτει μια σύνδεση μεταξύ των αρχείων που αποτελούν μια εφαρμογή στην διάρκεια εκτέλεσης του προγράμματος. Στο επίπεδο των βιβλιοθηκών η δυναμικότητα της φαίνεται από την χρήση που κάνει στα πλακέτα μέσα στα οποία υπάρχουν κατά ομάδα κλάσεις και διάφορα interfaces που παρέχονται στον προγραμματιστή ώστε αυτός να τα τροποποιήσει η να τα συμπληρώσει τοπικά η και μέσω του διαδικτύου.

Η δομή της Java την κάνει να είναι μία πολύ «τακτοποιημένη» γλώσσα κάτι την οποίο την καθιστά το κατάλληλο εργαλείο για την ανάπτυξη συμπαγούς και αξιόπιστου λογισμικού. Διαθέτει ένα αναγκαστικό σύστημα δηλώσεων, ένα ισχυρό σύστημα δηλώσεων και ένα ισχυρό σύστημα τύπων. Δεν χρησιμοποιεί δείκτες κάνοντας έτσι εύκολη την ζωή των προγραμματιστών, χωρίς να υπάρχει κίνδυνος καταστροφής στο memory map και με την αυτόματη διαχείριση μνήμης που έχει προαναφερθεί παρατηρούμε ότι παρέχει ένα δέσιμο αξιοπιστίας στο λογισμικό που αναπτύσσεται.

Η απόδοση της Java είναι επίσης ένα από τα πολύ σημαντικά της προτερήματα. Διαθέτει δυο πολύ σημαντικά εργαλεία τα οποία την καθιστούν τόσο αποδοτική. Το πρώτο είναι ότι υποστηρίζει πολλαπλά νήματα κατά την εκτέλεση των εφαρμογών που είναι γραμμένες σε Java. Το δεύτερο σημαντικό εργαλείο που την καθιστά αποδοτική την Java είναι τα JITs που είναι ουσιαστικά ένας compiler της τελευταίας στιγμής που μετέτρεπε στον ενδιαμεσο κώδικα σε bytecodes και υπόγεια σε έναν εκτελέσιμο κώδικα.

Spring Framework

Εικόνα 6

Το Spring Framework είναι ένα πλαίσιο εφαρμογών για την πλατφόρμα της Java. Τα κυριότερα χαρακτηριστικά του πλαισίου μπορούν να χρησιμοποιηθούν από κάθε εφαρμογή Java αλλά το πλαίσιο διαθέτει επίσης επεκτάσεις για την δημιουργία web εφαρμογών πάνω στην πλατφόρμα της Java EE. Παρόλο που το πλαίσιο του Spring δεν επιβάλλει κάποιο συγκεκριμένο προγραμματιστικό μοντέλο, έχει γίνει δημοφιλές στην κοινότητα της Java καθώς αποτελεί εναλλακτικά, τον αντικαταστάτη του μοντέλου JavaBeans.

Το Spring με τα χρόνια έχει σταματήσει να είναι απλά μία αρχή σχεδιασμού και πλέον συμπεριλαμβάνει διάφορες «μονάδες» οι οποίες προσφέρουν μια ευρεία γκάμα από υπηρεσίες όπως είναι ο προγραμματισμός aspect- oriented, πρόσβαση σε δεδομένα, διαχείριση συναλλαγών , το μοντέλο αρχιτεκτονικής MVC, εξουσιοδότηση και επαλήθευση, μηνύματα και testing.

Το Spring Framework είναι ένα λογισμικό ανοιχτού κώδικα.

Ιστορία του Spring

Τον Οκτώβριο του 2002 ο Rob Johnson ένας Αυστραλός computer specialist έγραψε ένα βιβλίο με τον τίτλο “Expert One-on-One J2EE Design and Development”.

Σε αυτό το βιβλίο πρότεινε μια πιο απλή λύση βασισμένη σε συνηθισμένες Java κλάσεις, το γνωστό POJO, και στο dependency injection. Ο Johnson έγραψε παραπάνω 30000 γραμμές

κώδικα υποδομής το οποίο περιλάμβανε έναν αριθμό από interfaces και κλάσεις για την ανάπτυξη εφαρμογών. Τον Φεβρουάριο του 2003 ο Rob, ο Juergen και ο Yann άρχισαν να συνεργάζονται στην ανάπτυξη του Spring. Το όνομα **Spring (**Άνοιξη**)** δόθηκε γιατί ήταν μια καινούργια αρχή από τον «χειμώνα» του J2EE.

Πρώτη Έκδοση

Η πρώτη έκδοση για το Spring ήταν τον Ιούνιο του 2003 που εκδόθηκε και η πρώτη έκδοση του σύμφωνα με το Apache License που ουσιαστικά προσέφερε στους χρήστες την δυνατότητα να τροποποιήσουν το λογισμικό και αργότερα να το επανεκδώσουν.

Επόμενες Εκδόσεις

Το πρώτο ορόσημο για το Spring , η έκδοση 1.0, ήταν τον Μάρτιο του 2004 με περισσότερες μικρότερες εκδόσεις να λαμβάνουν χώρα τον Σεπτέμβριο της ίδιας χρονιάς και τον Μάρτιο του 2005. Η έκδοση 1.2.6 κέρδισε ένα βραβείο παραγωγικότητας Jolt και ένα JAX.

Στην συνέχεια η επόμενη ολοκληρωμένη έκδοση είναι το Spring 2.0 που εκδόθηκε τον Οκτώβριο του 2006. Τον Νοέμβριο του επόμενου χρόνου έχουμε την έκδοση του Spring 2.5. 2 χρόνια αργότερα περίπου τον Δεκέμβριο του 2009 και το Spring 3.1 τον Δεκέμβριο του 2011 και το Spring 3.2.0 τον Νοέμβριο του 2013. Τον αμέσως επόμενο μήνα εκδόθηκε το Spring 4.0.

Κάποιες πολύ σημαντικές προσθήκες που υπήρχαν στο Spring 4.0 περιλάμβαναν υποστήριξη για το Java EE 8, το Groovy 2 μερικά aspects της Java EE 7 και τα Websockets.

Το Spring Framework 4.2.0 εκδόθηκε στις 31 Ιουλίου του 2015 και στην συνέχεια σχεδόν άμεσα αναβαθμίστηκε στην έκδοση 4.2.1 την πρώτη Σεπτεμβρίου του 2015 και ήταν συμβατή με τις Java 6, 7 , 8 και στόχευε κυρίως σε βελτίωση του «πυρήνα» της πλατφόρμας και σε πιο μοντέρνες ικανότητας web.

Η τελευταία έκδοση, το Spring 4.3, που έγινε στις 10 Ιουνίου του 2016 αλλά και καθώς η έκδοση 4.3.0 RC1 είναι διαθέσιμη. Κατά δήλωση των προγραμματιστών ειπώθηκε πως θα ήταν η τελευταία γενιά που θα βρίσκεται μέσα στις «προϋποθέσεις» συστήματος για το Spring 4 (Java 6+, Servlet 2.5+).

Μελλοντικές Εκδόσεις

Το Spring 5 ανακοινώθηκε, και σύμφωνα με την ανακοίνωση θα είναι χτισμένο πάνω στον Reactor Core των Reactive Streams.

Οι Μονάδες του Spring

Οι μονάδες ή modules του Spring είναι τα διάφορα εργαλεία η υπηρεσίες που παρέχει το Spring στις διάφορες web εφαρμογές που θέλουμε να δημιουργήσουμε.

Inversion of Control Container – Dependency Injection

Κεντρικό κομμάτι του Spring είναι το **Inversion of Control Container**, που όπως αναφέρθηκε και πιο πάνω είναι μια αρχή σχεδιασμού του κώδικα, το οποίο παρέχει τα μέσα για να ρυθμίζεις και να διαχειρίζεσαι τα Java αντικείμενα, χρησιμοποιώντας την ιδιότητα της απεικόνισης(reflaction). Ο container είναι υπεύθυνος για την διαχείριση του κύκλου ζωής των αντικειμένων, την δημιουργία αυτών των αντικειμένων, την κλήση των μεθόδων αρχικοποίησης τους και τέλος την διαμόρφωση τους.

Τα αντικείμενα που δημιουργήθηκαν από τον container λέγονται και beans. Το container μπορεί να ρυθμιστεί φορτώνοντας XML αρχεία. Αυτά τα αρχεία xml περιέχουν τους ορισμούς για κάθε bean και παρέχει την πληροφορία για την δημιουργία των beans.

Η χρήση βέβαια του container που περιέχεται μέσα στο Spring δεν είναι απαραίτητη προϋπόθεση για την χρήση του framework, παρόλα ταύτα η χρήση του κάνει πιο εύκολη την διαμόρφωση και την προσαρμογή της εφαρμογής.

Aspect-Oriented Programming Framework

Το AOP συμπληρώνει τον αντικειμενοστραφή προγραμματισμό προσφέροντας έναν διαφορετικό τρόπο σκέψης όσων αφορά την δομή ενός προγράμματος. Η κύρια μονάδα στον αντικειμενοστραφή προγραμματισμό είναι η κλάση σε αντίθεση με τον AOP που είναι το aspect (όψη). Αν και η χρήση του AOP Framework στο Spring δεν είναι υποχρεωτική, το AOP συμπληρώνει το Spring IoC και προσφέρει έτσι ένα πολύ καλό μεσολογισμικό.

Το AOP χρησιμοποιείται στο Spring για να προσφέρει μια δηλωτική υπηρεσία, ιδιαιτέρως σαν αντικαταστάτης για το EJB. Η πιο σημαντική τέτοια υπηρεσία ήταν η δηλωτική διαχείριση συναλλαγών. Επίσης επιτρέπει στους χρήστες να δημιουργήσουν δικά τους aspects, συμπληρώνοντας έτσι τον δικό τους αντικειμενοστραφή προγραμματισμό μαζί με το AOP.

Το κίνητρο για να δημιουργηθεί ένα ξεχωριστό AOP Framework μέσα στο Spring ήταν λόγω της γνώμης που είχαν οι προγραμματιστές ότι θα έπρεπε να προσφέρουν βασικά χαρακτηριστικά του AOP χωρίς να περιπλέκονται τα πράγματα ούτε στον σχεδιασμό, ούτε στην υλοποίηση αλλά ούτε στο στήσιμο της εφαρμογής. Το Spring AOP εκμεταλλεύεται πλήρως το Spring container.

Για να μην δημιουργηθεί σύγχυση οι δημιουργοί του Spring αποφάσισαν να μην χρησιμοποιήσουν καινούργια ορολογία για το Spring AOP έτσι στο documentation οι αναφορές, και τα παραπεμπτικά έχουν την ίδια ονοματολογία που έχουν και σε οποιοδήποτε άλλο AOP framework.

Data Access Framework

Το data access framework που διαθέτει το Spring προσπαθεί να εξαλείψει τα περισσότερα προβλήματα που αντιμετωπίζουν οι προγραμματιστές που ασχολούνται με βάσεις δεδομένων στις εφαρμογές τους. Υποστήριξη παρέχεται για τα περισσότερα δημοφιλή data access frameworks της Java όπως είναι το JDBC, Hibernate, JDO, JPA , Toplink , OJB και Cayenne μεταξύ άλλων.

Για όλα τα παραπάνω frameworks το Spring διαθέτει χαρακτηριστικά όπως είναι η διαχείριση πόρων, αντιμετώπιση exception, διαμεσολάβηση στις διάφορες συναλλαγές, καθώς και υποστήριξη όταν η βάση πρέπει να διαχειριστεί δεδομένα BLOB η CLOB. Όλα αυτά τα χαρακτηριστικά γίνονται διαθέσιμα όταν κάνουμε χρήση των πρότυπων κλάσεων που μας παρέχει το Spring για κάθε υποστηριζόμενο framework, πολλοί κατακρίνουν βέβαια αυτά τα εργαλεία καθώς υποστηρίζουν ότι δεν προσφέρουν κανένα επιπλέον πλεονέκτημα από το να χρησιμοποιήσεις απευθείας API όπως είναι το Hibernate. Σαν απάντηση οι προγραμματιστές του Spring επιτρέπου να χρησιμοποιείς APIs όπως το Hibernate ή το JPA απευθείας. Κάτι τέτοιο απαιτεί βέβαια την χρήση ακόμα μιας από τις

μονάδες του Spring καθώς ο κώδικας που τρέχει στην εφαρμογή δεν είναι υπεύθυνος πλέον για την δημιουργία και το κλείσιμο των πόρων στην βάση δεδομένων. Η μονάδα αυτή είναι η «διαφανείς» διαχείριση συναλλαγών (transparent transaction management).

Μαζί οι μονάδες αυτές προσφέρουν μια ευέλικτη λύση για να δουλεύει με data access frameworks. Το Spring Framework είναι το μόνο framework διαθέσιμο στην Java που προσφέρει διαχειρίσιμα περιβάλλοντα διαχείρισης δεδομένων εκτός εφαρμογής η server.

Transaction Management Framework

Το Transaction Management Framework του Spring προσθέτει έναν πιο αφηρημένο μηχανισμό στην πλατφόρμα της Java. Το framework αυτό μεταξύ άλλων είναι ικανό να κάνει συναλλαγές μεταξύ δυο συστημάτων σε τοπικό αλλά και σε δικτυακό επίπεδο, να λειτουργεί με εμφωλευμένες συναλλαγές, να λειτουργεί με savepoints και τέλος να λειτουργεί σε σχεδόν όλα τα περιβάλλοντα της πλατφόρμας Java. Σε αντίθεση το JTA (Java Transaction API) υποστηρίζει μόνο εμφωλευμένες και καθολικές συναλλαγές και επίσης απαιτεί έναν application server.

Το Spring παρέχει έναν PlatformTransactionManager που μπορεί να χρησιμοποιηθεί για έναν μεγάλο αριθμό από στρατηγικές για transaction management όπως οι συναλλαγές που γίνονται με JDBC, Object oriented mapping, JTA, Use Transaction αλλά και τέλος διαφορές συναλλαγές όπως είναι αυτές που αφορούν βάσεις δεδομένων που ειδικεύονται σε αντικείμενα (objects).

Μαζί με αυτόν τον αφηρημένο αυτόν μηχανισμό το framework επίσης προσφέρει 2 τρόπους για να προσθέσουμε διαχείριση συναλλαγών για τις εφαρμογές μας. Ο πρώτος τρόπος είναι προγραμματιστικά χρησιμοποιώντας το Transaction Template που περιέχει μέσα το Spring, και με διαμόρφωση χρησιμοποιώντας metadata όπως XML η annotations της Java.

Με το Spring Data Access και το Transaction Management framework - το Data access υλοποιεί ουσιαστικά το transaction management framework - είναι δυνατόν να φτιαχτεί ένα σύστημα συναλλαγών με την βάση χρησιμοποιώντας μόνο τον τρόπο της διαμόρφωσής χωρίς να είναι απαραίτητο να βασιστούμε σε μοντέλα όπως είναι το JTA η το EJB. Τέλος το framework συναλλαγών μπορεί να υλοποιηθεί και από συστήματα μηνυμάτων καθώς και από συστήματα που κάνουν caching.

MVC - Model View Controller Framework

Το Spring διαθέτει το δικό του model-view-controller framework κάτι που δεν ήταν αρχικά σχεδιασμένο να υπάρχει. Το MVC είναι ένα αρχιτεκτονικό πρότυπο λογισμικού για την ανάπτυξη διεπαφών χρήστη στους υπολογιστές. Το MVC διαχωρίζει το δοσμένο λογισμικό σε 3 διασυνδεδεμένα μέρη με τον σκοπό να ξεχωρίσει τις εσωτερικές απεικονίσεις της πληροφορίας από τον τρόπο που η πληροφορία παρουσιάζεται και γίνεται αντιλιπ΄τη από τον χρήστη. Η σχεδίαση MVC διαχωρίζει αυτά τα τρία μέρη με τέτοιο τρόπο επιτρέποντας την επανάχρηση του κώδικα όπως και επίσης τον παράλληλο προγραμματισμό. Παραδοσιακά χρησιμοποιούταν για την δημιουργία διεπαφής χρήστης υπολογιστή για επιτραπέζιες εφαρμογές, παρόλαυτα έχει γίνει πολύ δημοφιλές για την ανάπτυξη web εφαρμογών.

Οι προγραμματιστές του Spring αποφάσισαν να γράψουν τον δικό τους web framework σαν μια αντίδραση καθώς πίστευαν ότι τα μέχρι τότε δημοφιλή web frameworks όπως ήταν το Jakarta Struts, αλλά και επίσης καθώς δεν υπήρχε αρκετά διακριτή διαφορά ανάμεσα στα 3 επίπεδα που παρουσιαζόταν σε μια εφαρμογή.

Όπως το Struts έτσι και το Spring MVC είναι ένα framework που βασίζεται στην δημιουργία αιτημάτων. Το framework επίσης ορίζει interface στρατηγικής για κάθε ευθύνη που πρέπει να αναλάβει ένα μοντέρνο framework που βασίζεται σε αιτήματα. Σκοπός του κάθε interface είναι να είναι απλό και ξεκάθαρο για να επιτρέπει σε κάθε προγραμματιστή να κάνει την δικιά του υλοποίηση αν αυτός το επιλέξει. Το Spring MVC ανοίγει δρόμο για έναν πιο καθαρό front-end κώδικα. Όλα τα interfaces που αποτελούν το framework είναι άρρηκτα συνδεδεμένα με το Servlet API. Κάτι τέτοιο βέβαια έχει θεωρηθεί από πολλούς ότι αποτρέπει το framework να προσφέρει μια υψηλού επιπέδου αφαιρετικότητα όσον αφορά την δημιουργία web εφαρμογών.

Το DispacherServlet είναι ο front controller του framework και είναι υπεύθυνος για να παραχωρεί έλεγχος στα διάφορα interfaces που καταπιάνονται με τα αιτήματα HTTP.

Interfaces Μέσα στο Spring

Τα σημαντικότερα interfaces που υπάρχουν στο Spring MVC είναι τα παρακάτω:

- Ο **Controller.** Ο controller είναι κομμάτι που βρίσκεται ενδιάμεσα από το model και το view και είναι υπεύθυνο για να διαχειρίζεται τα εισερχόμενα αιτήματα και να ανακατευθύνει στην κατάλληλη απάντηση. Λειτουργεί σαν μια πύλη που κατευθύνει τις πληροφορίες ανάλογα.

- Ο **HandlerAdapter.** Ο HandlerAdapter είναι υπεύθυνος για την διαχείριση των εισερχόμενων αιτημάτων.

- Ο **HandlerInterceptor.** Ο HandlerInterceptor είναι υπεύθυνος για την αναχαίτηση των εισερχόμενων εντολών κατά κάποιο τρόπο ως ένα φίλτρο.

- Ο **HandlerMapping.** Ο HandlerMapping είναι υπεύθυνος για την επιλογή των αντικειμένων που θα διαχειριστούν τα διάφορα αιτήματα, με βάση διάφορα χαρακτηριστικά οι συνθήκες που είναι είτε εσωτερικά είτε εξωτερικά στα αιτήματα αυτά.

- Ο **Locale Resolver.** Ο δουλειά του Locale Resolver είναι να αποφασίζει και προαιρετικά να αποθηκεύει το locale ενός χρήστη. Στον προγραμματισμό και γενικότερα στην επιστήμη των υπολογιστών το locale είναι ένα σύστημα παραμέτρων το οποίο είναι ορίζει την γλώσσα του χρήστη την περιοχή αλλά και κάποια ειδική ρύθμιση που έχει κάνει ο χρήστης.΄

- Ο **MultipartResolver.** Ο MultipartResolver διευκολύνει την διαδικασία του να δουλεύεις με αρχεία.

- Το **View.** Άλλο ένα σημαντικό κομμάτι του Spring MVC είναι το κομμάτι του View. Το View είναι υπεύθυνο για την επιστροφή απαντήσεων στον client. Πολλά από τα

αιτήματα πηγαίνουν κατευθείαν στο κομμάτι του view χωρίς να περνάνε από το κομμάτι του model.

- Ο **ViewResolver.** Η χρήση του ViewResolver είναι προαιρετική. Βασίζεται στο ότι το MVC σου σου επιτρέπει να δημιουργήσεις μοντέλα τα οποία εμφανίζονται στον browser σου χωρίς να είναι απαραίτητο να τα δέσεις με κάποιο συγκεκριμένο view. Το View Resolver ουσιαστικά αντιστοιχίζει τα ονόματα των views με τα views αυτά καθαυτά.

Όλα τα παραπάνω στοιχεία τα οποία αναφέραμε και αποτελούν το όλο framework έχουν το καθένα ξεχωριστά μια σημαντική ευθύνη. Τα εργαλεία και οι δυνατότητες που προσφέρει αυτό το MVC είναι πραγματικά είναι πραγματικά πολύ ισχυρές και συνοδεύουν στο Spring όντας υλοποιημένες και προσφέροντας μεγάλες ευκολίες στην ανάπτυξη κάποιας εφαρμογής, χωρίς αυτό βέβαια να μην επιτρέπει σε κάποιον προγραμματιστή να γράψει και αυτός την δικιά του υλοποίηση για τον interface που τον ενδιαφέρει κατ' ανάγκη.

Κάτι άλλο που είναι πολύ σημαντικό για το Spring MVC είναι η ευκολία που προσφέρει στο να δοκιμαστούν οι υλοποιήσεις των interfaces. Το DispatcherServlet είναι συνδεδεμένο με το inversion of control container για να διαμορφώσει το web επίπεδο της εφαρμογής. Παρόλα αυτά οι web εφαρμογές που είναι γραμμένες σε Spring μπορούν να χρησιμοποιήσουν άλλα κομμάτια του Spring Framework και να μην χρησιμοποιήσουν καθόλου το MVC του.

Remote Access Framework

Το Remote Access framework του Spring είναι ένα σύνολο interfaces για να δουλέψουμε με διάφορες τεχνολογίες που βασίζονται στο RPC (απομακρυσμένη κλήση διεργασιών) τα οποία είναι στην πλατφόρμα της Java, για την συνδεσιμότητα του χρήστη αλλά καθώς και για να τακτοποιεί τα αντικείμενα που βρίσκονται στον server. Το πιο σημαντικό χαρακτηριστικό που προσφέρει αυτό το framework είναι η ευκολία για την διαρρύθμιση και χρήση αυτών των τεχνολογιών με το να συνδυάζει το inversion of control και το AOP.

Το framework επίσης προσφέρει το fault factory το οποίο κάνει αυτόματη επανασύνδεση σε περίπτωση αποτυχίας σύνδεσης αλλά καθώς και επίσης προσφέρει κάποιες βελτιστοποιήσεις για χρήση από τον πρόγραμμα πελάτη.

Το Spring υποστηρίζει εξ' ορισμού τα εξής πρωτόκολλα και προϊόντα

- Τα πρωτόκολλα που έχουν σαν βάση το HTTP
    - Hessian: Σειριακό πρωτόκολλο το οποίο το συντηρείται από το COBRA πρωτόκολλο.
    - RMI (1). Χρήση RMI το οποίο ήταν κατασκευασμένο για το Spring.
    - RMI (2). Χρήση RMI το οποίο είναι κατασκευασμένο για να ανταπεξέρχεται στην κανονική χρήση RMI.
    - RMI-IIOP (COBRA).
- Ενσωμάτωση Enterprise JavaBean στο πρόγραμμα πελάτη.

    - Τοπικά EJB τα οποία συνδέονται σε session beans τοπικά.
    - Απομακρυσμένα EJB τα οποία συνδέονται σε session beans δικτυακά.

- SOAP
    - Ενσωμάτωση με τον Apache Axis framework υπηρεσιών δικτύου.

Πλεονεκτήματα του Spring Framework

Σε ένα ποιο γενικό επίπεδο η χρήστη ενός framework είναι κοινή για όλα τα frameworks. Καθώς τα frameworks είναι εξαρχής γραμμένα σε μια συγκεκριμένη αρχιτεκτονική σε βοηθούν και παράλληλα σε περιορίζουν να δημιουργήσεις την εφαρμογή που θες να φτιάξεις με έναν τρόπο και με μια δομή η οποία έχει ήδη βελτιστοποιηθεί για σένα από το framework.

Πιο συγκεκριμένα το spring framework δίνει την δυνατότητα δημιουργίας εφαρμογών web με την χρήση POJO's. Το πλεονέκτημα της χρήσης POJO είναι ότι πλέον δεν χρειάζεσαι έναν application server για να τρέξεις την εφαρμογή σου μέσα σε αυτόν αλλά χρειάζεται απλά ένα servlet container.

Ένα άλλο πλεονέκτημα είναι ότι το Spring περιέχει στον πυρήνα του υπάρχουσες τεχνολογίες όπως είναι το ORM framework, logging framework και άλλα πολλά τα οποία είναι απαραίτητα για την δημιουργία εφαρμογών οπότε δεν είναι απαραίτητη χειροκίνητη ένταξη τους στο project που δουλεύουμε κάθε φορά.

Για την ανάπτυξη των web εφαρμογών το framework περιέχει ένα πολύ καλοσχεδιασμένο MVC framework το οποίο συγκεκριμένα για την Java είναι ίσως το πιο δημοφιλές.

Τέλος το Spring καλύπτει ένα αρκετά ευρύ φάσμα για την δημιουργία web εφαρμογών κάτι που το κάνει να είναι επεκτάσιμο, πράγμα που σημαίνει ότι μπορεί να λειτουργήσει καλά σε μικρού αλλά και σε μεγάλου μεγέθους project.

Μειονεκτήματα του Spring Framework

Αν και το Spring είναι ένα από τα πιο δημοφιλή frameworks αυτή την στιγμή για την ανάπτυξη web κυρίως εφαρμογών σε Java έχει , όπως και όλα τα πράγματα βέβαια στην ζωή, κάποια μειονεκτήματα.

Κάθε Java Framework που υπάρχει αυτή την στιγμή βασίζει την λειτουργεία του στα servlets, έτσι κάνει και το spring. Αν και το Spring είναι ένα επίπεδο πιο πάνω από τα servlets και προσφέρει μια μεγαλύτερη ευκολία στο χτίσιμο εφαρμογών πάνω σε αυτά και σε πολλές περιπτώσεις σου λύνει τα χέρια δεν παύει να εξαρτάτε πάντα από αυτά.

Ένα άλλο μειονέκτημα του Spring είναι ότι έχει πάψει να είναι πιο απλό την αρχική του έκδοση. Λογικό καθώς ήταν καθώς αυξάνονταν οι ανάγκες για την προσθήκη νέων χαρακτηριστικών και εργαλείων να αυξάνεται και η πολυπλοκότητα του framework. Αν και αρχικά υπερτερούσε της Java EE στην απλότητα του πλέον με την βελτίωση της Java EE το Spring έχασε το πλεονέκτημα του εκείνο και απέκτησε ένα μειονέκτημα.

Στην συνέχεια το Spring διαθέτει μια μεγάλη **καμπύλη μάθησης**. Πολλοί προγραμματιστές αδυνατούν να καταλάβουν έννοιες όπως είναι το dependency injection αλλά και άλλες

τεχνικές που χρησιμοποιεί το Spring κάτι που έχει σαν αποτέλεσμα να μην μπορούν να συνειδητοποιήσουν τις πλήρεις δυνατότητες του Spring.

Ένα άλλο μειονέκτημα του Spring είναι ότι το Spring βρίσκεται σε μια διαρκή φάση αλλαγής. Για παράδειγμα η ρύθμιση της εφαρμογής με την χρήση annotations που υπάρχει αυτή την στιγμή στο Spring είναι κάτι που δεν είναι εύκολο για όλους να χρησιμοποιηθεί. Αυτό δημιουργεί ένα μεγάλο εμπόδιο στην απλή χρήση του framework.

Ένα ακόμα μειονέκτημα είναι το dependency injection που προσπαθεί να προωθήσει το spring είναι κάτι που με την διαρκή χρήση του Spring η εφαρμογή που προσπαθούμε παύει να εξαρτάται από τα dependencies τα εξωτερικά και γίνεται σχεδόν απόλυτα εξαρτημένο από το Spring Framework αυτό καθαυτό. Κάτι τέτοιο δημιουργεί ένα παράδοξο στην χρήση του Spring αλλά και βέβαια ένα πρόβλημα καθώς θα είναι αδύνατο να ξεχωρίσεις την εφαρμογή από το framework με το οποίο δημιουργήθηκε.

Τέλος από την εταιρία που αναπτύσσει το Spring δεν έχουν δοθεί αρκετές οδηγίες και κατευθυντήριες γραμμές κάτι που καθιστά το να γίνεις εξπέρ στο κάτι ακόμα πιο δύσκολο.

Spring Boot

Τι είναι το Spring Boot

Καθώς συνέχεια αυξάνεται ο αριθμός των διάφορων χαρακτηριστικών που προστίθενται στο Spring οι προγραμματιστές που το ανέπτυξαν αποφάσισαν να δημιουργήσουν έναν τρόπο ώστε ο χρήστης να μπορέσει να αναπτύξει μια εφαρμογή χωρίς να χρειάζεται απαραίτητα να ρυθμίσει αλλά και να κατασκευάσει χειροκίνητα την δομή του όλου project. Το Spring Boot κάνει αυτόματα όλα αυτά τα πράγματα χτίζοντας ένα project χωρίς απαραίτητα να σε περιορίζει στις αρχικές καταστάσεις που έχουν όλα αυτά τα συστήματα αλλά να σου επιτρέπει να κάνεις ότι αλλαγές θες ώστε να φέρεις το πρόγραμμα στα μέτρα σου.

Το Spring Boot λοιπόν είναι μια πλήρως εξοπλισμένη σουίτα με την οποία μπορείς να δημιουργήσεις μια εφαρμογή σε Spring MVC είτε αυτή είναι web είτε αυτή είναι desktop γράφοντας όσο το δυνατόν λιγότερες γραμμές κώδικα για το στήσιμο και την αρχική αρχικοποίηση της εφαρμογής.

Front End

Για την δημιουργία του front end χρησιμοποιήθηκε **HTML**, **Javascript** καθώς και **CSS**. Εκτός από τις τρείς αυτές γλώσσες προγραμματισμού χρησιμοποιήθηκε πιο συγκεκριμένα το front end framework **Bootstrap** που εμπεριέχει τα τρία παραπάνω στοιχεία

Εικόνα 7

Bootstrap

Το **Bootstrap** είναι ένα δωρεάν front-end web framework το οποίο χρησιμοποιείται για την δημιουργία web εφαρμογών αλλά και ιστοσελίδων. Περιέχει διάφορα σχεδιαστικά υποδείγματα τα οποία έχουν σαν βάση την HTML και το CSS και περιέχει διάφορες επιλογές για τυπογραφία, φόρμες , κουμπιά, ετικέτες, μενού καθώς και άλλα διάφορα στοιχεία διεπαφής. Για τα στοιχεία του αυτά περιέχει και διάφορες επεκτάσεις σε επίπεδο Javascript κάτι που εξυπηρετεί όταν θέλεις να κάνεις την ιστιοσελίδα σου πιο δυναμική.

Αναπτύχθηκε από τον Mark Otto και τον Jacob Thornton εργαζόμενους στο Twitter εξού και το όνομα που είχε αρχικά «Twitter Blueprint». Ήταν μια προσπάθεια των παραπάνω κυρίων στην δημιουργία μια συνοχής στα εργαλεία που χρησιμοποιούνταν εσωτερικά από την εταιρία με σκοπό τον περιορισμό των λαθών, αντιφάσεων και προβλημάτων που δημιουργούσε ή τόσο μεγάλη ποικιλία εργαλείων που χρησιμοποιούταν μέχρι πριν το Bootstrap.

Εκδόσεις του Bootstrap

- Η πρώτη έκδοση του bootstrap έγινε τον Αύγουστο του 2011 ως ένα open source project και ανανεώνεται κάθε τόσο από τους Otto και Thorton καθώς και από μια μερίδα προγραμματιστές που ασχολήθηκαν στις αρχές του project και του πυρήνα αυτού.
- Τον Ιανουάριο του 2012 έχουμε την έκδοση του Bootstrap 2 το οποίο περιείχε υποστήριξη για το 12-column layout και ήταν responsive.
- Τον Αύγουστο του 2013 το Bootstrap 3 έκανε την εμφάνισή του το οποίο έκανε τα στοιχεία της σελίδας να έχουν ένα πιο επίπεδο design και να έχουν μια mobile προσέγγιση πρώτα.
- Τον Οκτώβριο του 14 ο Otto ανακοίνωσε ότι το Bootstrap 4 ήταν ήδη σε διαδικασία ανάπτυξης με τα alpha και beta version να κάνουν την εμφάνισή τους τον Αύγουστο του 2015 και τον Αύγουστο του 2017 αντίστοιχα.

Χαρακτηριστικά του Bootstrap

- Το Boostrap υποστηρίζει τους κυριότερους web browser.

- Από την έκδοση 2.0 Και μετά υποστηρίζει responsive designs κάτι που σημαίνει ότι η σελίδα η γενικά το template αλλάζει ανάλογα σε τι οθόνη το προβάλουμε.

- Από την έκδοση 3.0 και μετά το responsive design έγινε η default κατάσταση του bootstrap και το framework άρχισε να υποστηρίζει πλέον by default κινητες συσκευές.

Database – Database Management – Database Server

PostgreSql

Η ανάπτυξη της βάσης δεδομένων έγινε σε Postgre. Η Postgre είναι ένα σύστημα ανάπτυξης και διαχείρισης βάσεων δεδομένων open source. Η Postgre λειτουργεί και ως ένας server βάσης δεδομένων, με κύριες λειτουργίες της να είναι η αποθήκευση δεδομένων αλλά και η ανάκλησή τους ανάλογα με τις κλήσεις που δέχεται από τον χρήστη.

Η Postgre μπορεί να χρησιμοποιηθεί εξίσου για μικρές εφαρμογές του ενός και δυο υπολογιστών αλλά και για εφαρμογές τύπου web που οι απαιτήσεις και το payload εκατονταπλασιάζεται εξυπηρετώντας πολλούς servers ταυτόχρονα.

Η Διαχείριση

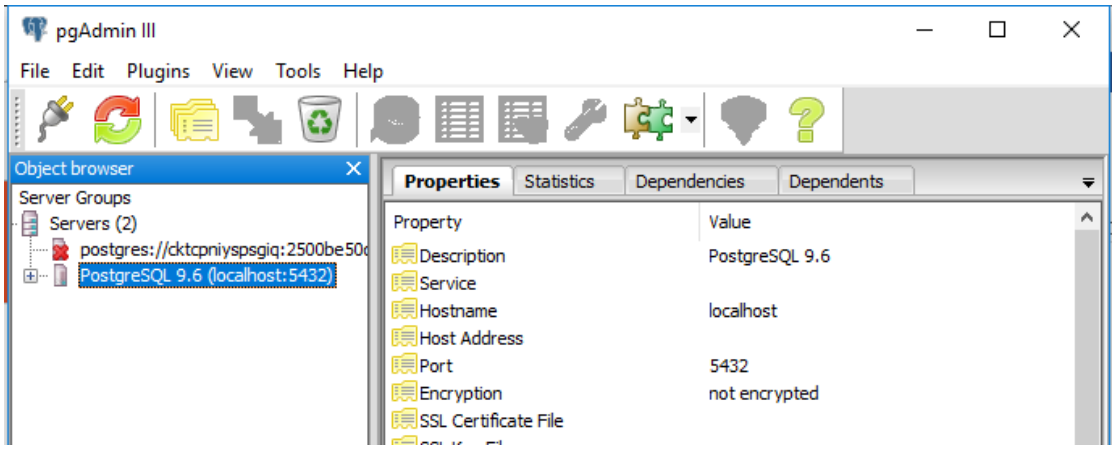
Η διαχείριση της βάσης έγινε με το pgAdmin. Το pgAdmin με την είναι ένα εργαλείο γραφικής διαχείρισης βάσεων postgre. Αναπτύχθηκε αρχικά σαν πρωτότυπο με το όνομα pgManager το 1998 και αργότερα εκδόθηκε ξανά κάτω από την άδεια GPL με το όνομα pgAdmin. Η δεύτερη έκδοση που ονομάστηκε pgAdmin II και εκδόθηκε το 2002 ήταν μια πλήρης επανεγγραφή της πρώτης έκδοσης.

Η τρίτης έκδοση που εκδόθηκε κάποια χρόνια αργότερα σε αντίθεση με τις άλλες εκδόσεις που ήταν γραμμένες σε Visual Basic η συγκεκριμένη ήταν γραμμένη σε C++ και χρησιμοποιώντας το wxWidgets framework του επέτρεπε να τρέχει στα περισσότερα λειτουργικά συστήματα που υπάρχουν.

Η Βάση

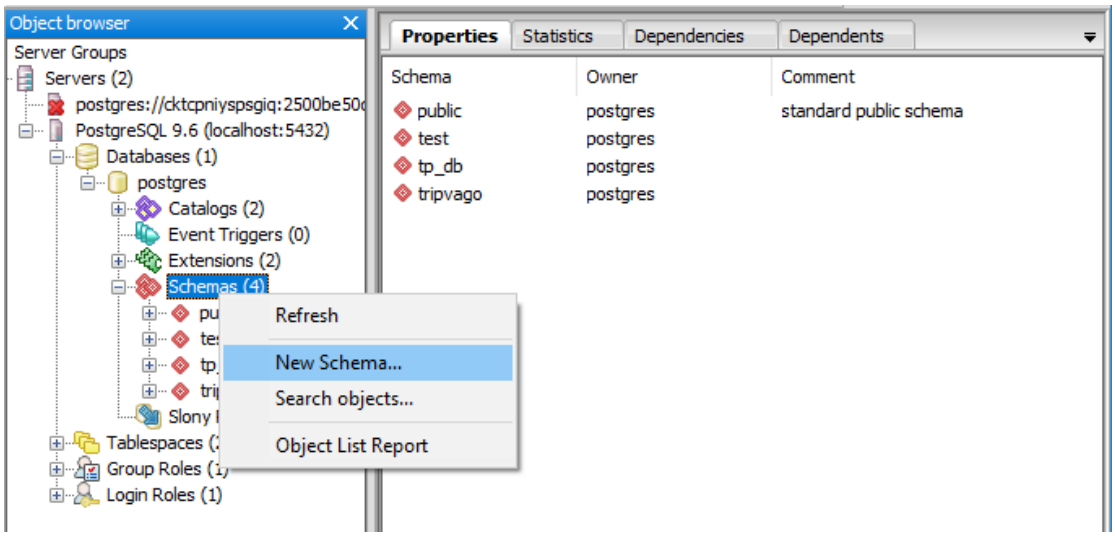
Αρχικά για την δημιουργία της βάσης εγκατέστησα τον postgres server και τον έτρεξα. Χρησιμοποιώντας το πρόγραμμα διαχείρισης pgAdmin III δημιούργησα στην βάση το σχήμα της βάσης μου:

Επιλέγουμε τον Server:



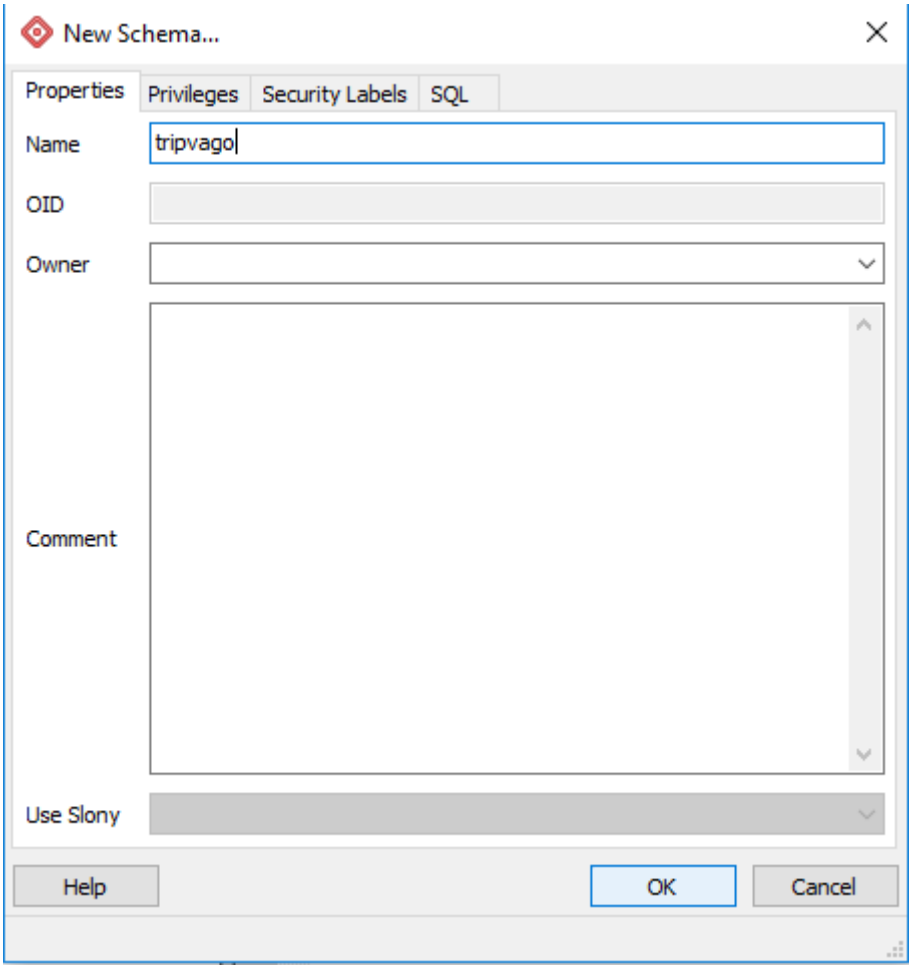
Εικόνα 8

Δεξί κλίκ και επιλέγουμε New Schema.



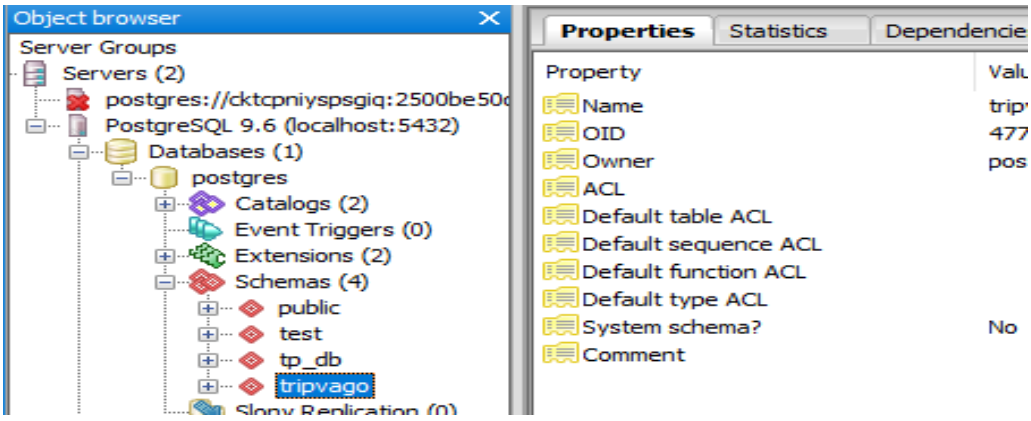
Εικόνα 9

Επιλέγουμε το όνομα που θέλουμε και πατάμε κλικ στο OK



Εικόνα 10

Και τέλος έχουμε έτοιμο το schema που θα χρησιμοποιήσουμε.



Εικόνα 11

Στην συνέχεια πηγαίνουμε και ρυθμίζουμε τις παραμέτρους για την επικοινωνία του project μας με την βάση.

Κεφάλαιο 4: Υλοποίηση Εφαρμογής

4.1 Βάση

Για το γέμισμα της βάσης χρησιμοποιούμε ένα σύστημα annotation του Hibernate και του JPA.

4.1.1 Ρύθμιση Παραμέτρων Εφαρμογής

Στο αρχείο application.properties μπορούμε να ρυθμίσουμε το Hibernate. Το Hibernate είναι ένα build-in σύστημα στο Spring που χρησιμοποιείται για όλες τις διεργασίες που κάνει το σύστημα με την βάση.

```
security.user.password= test
spring.datasource.url= jdbc:postgresql://localhost:5432/postgres
spring.datasource.username  = postgres
spring.datasource.password = 1234


spring.jpa.properties.hibernate.default_schema = tripvago
spring.jpa.hibernate.ddl-auto=create-drop
```

Εικόνα 12

Παρατηρώντας την παραπάνω εικόνα

Στο **spring.datasource.url** θέτουμε το url στο οποίο τρέχει ο database server. Το url στο πρώτο μέρος του έχει τα αναγνωριστικά του είδους της βάσης που τρέχει στην συνέχεια το port του server που τρέχει και τέλος το όνομα της βάσης που θα διαβάσει.

Στην παράμετρο **spring.datasource.username** θέτουμε το username που απαιτείται για την σύνδεση στον database server. Το username και το password που ακολουθεί μετά έχουν ρυθμιστεί κατά την εγκατάσταση του database server.

Στην παράμετρο **spring.datasource.password** θέτουμε τον κωδικό που απαιτείται για την σύνδεση στην βάση.

Στην συνέχεια και στην επόμενη γραμμή του αρχείου έχουμε την παράμετρο **spring.jpa.properties.hibernate.default_schema.** Στην συγκεκριμένη παράμετρο θέτουμε το default σχήμα με το οποίο θα συνδιαλέγεται η εφαρμογή μας. Στην περίπτωση που δεν θέσουμε αυτή την παράμετρο η εφαρμογή συνδιαλέγετε με το default schema της postgre που είναι το **public**.

Τέλος έχουμε την παράμετρο **spring.jpa.hibernate.ddl-auto.** Στην παράμετρο αυτή έχουμε θέσει την τιμή **create-drop.** Αυτό σημαίνει ότι το σύστημα θα δημιουργήσει την βάση κατά την πρώτη εκτέλεσή του και στην συνέχεια την όταν τερματίσουμε το sessionFactory με άλλα λόγια όταν τερματίσουμε την εκτέλεση της εφαρμογής θα κάνει drop ή με άλλα λόγια θα διαγράψει την βάση. Πέρα από την **create-drop** υπάρχουν και οι εξής επιλογές:

- **Validate.** Το validate απλά επικυρώνει το σχήμα της βάσης.

- **Update.** Το update βρίσκει αν έχουν γίνει αλλαγές κατά την προηγούμενη η αυτή την εκτέλεση της εφαρμογής και ενημερώνει ανάλογα την βάση.

- **Create.** Το create δημιουργεί το σχήμα διαγράφοντας κάθε προηγούμενα δεδομένα.

4.1.2 Δημιουργία Πινάκων

Για την δημιουργία των πινάκων της βάσης χρησιμοποιούμε τα annotations του Spring/Hibernate. Αυτό σημαίνει ότι κάθε πίνακας είναι και μια κλάση στην Java. Οι συγκεκριμένες κλάσεις λέγονται και POJO (Plain Old Java Objects). Με τα annotations θα κάνουμε το λεγόμενο mapping με JPA. Το JPA καθώς και τα POJOs είναι ανήκουν στις persistent οντότητες του Hibernate. Ένα παράδειγμα του JPA Mapping καθώς και ένα POJO ακολουθεί παρακάτω:

Car

```
5
6
7   @Entity
8   @Table(name = "trip_car")
9   public class Car {
10
11      @Id
12      @GeneratedValue(strategy = GenerationType.IDENTITY )
13      @Column(name = "car_id",insertable = false)
14      private long carId;
15
16      @Column(name= "rent_a_car_id")
17      private long rentACarId;
18
19      @Column(name = "registration_number")
20      private String registrationNumber;
21
22      @Column(name = "cubic_capacity")
23      private int cubicCapacity;
24
25      @Column(name = "number_of_seats")
26      private int numberOfSeats;
27
28      @Column(name = "car_category")
29      private String carCategory;
30
31      @Column(name = "created")
32      private Date created;
33
34      @Column(name = "price")
35      private float price;
36
37
```

Εικόνα 13

- Το annotation **@Entity** δηλώνει ότι η παρακάτω κλάση είναι ένα entity δηλαδή ένα POJO αντικείμενο.

- Το annotation **@Table** δηλώνεται σε επίπεδο κλάσης. Σου επιτρέπει να δηλώσεις τον πίνακα τον κατάλογο η τον σχήμα που παραφέρεσαι. Στην περίπτωσή μας δηλώνουμε το όνομα της πίνακα.

- Το annotation **@Id** δηλώνει ποιο θα είναι το αναγνωριστικό για την οντότητα στην οποία κάνουμε mapping αυτή την στιγμή. Στην περίπτωση μας

- Στην συνέχεια και πιο συγκεκριμένα για το αναγνωριστικό της οντότητάς μας μπορούμε να δηλώσουμε με το annotation **@GeneratedValue** το τι αποτελεί το συγκεκριμένο πεδίο για την οντότητα πίνακα μας. Στην συγκριμένη περίπτωση έχουμε το IDENTITY που σημαίνει ότι το συγκεκριμένο πεδίο αποτελεί κλειδί για τον συγκεκριμένο πίνακα.

- Το **@Column** annotation χρησιμοποιείται για να υποδηλώσουμε ότι το πεδίο που ακολουθεί θα είναι στήλη του πίνακα που κάνουμε mapping. Με το χαρακτηριστικό name δηλώνουμε το όνομα που θα έχει η στήλη αυτή που κάνουμε map μέσα στην βάση.

4.1.3 Πίνακες Βάσης

Οι κυριότεροι πίνακες με τα πεδία τους είναι οι εξείς

- **trip_attraction(**attraction_name,created,description,entrance_fee,working_days,working_hours,location**)** Ο πίνακας trip_attraction περιλαμβάνει όλα τα αξιοθέατα που υπάρχουν καθώς και πληροφορίες για αυτά και χωρίζονται ανά περιοχή.

- **trip_car (**car_id**,** rent_a_car_id ,car_category, created, cubic_capacity, number_of_seats, registration_number, price**).** Ο πίνακας trip_car περιέχει πληροφορίες για τα αμάξια και τις μηχανές που έχει κάθε γραφείο ενοικιάσεως αυτοκινήτων. στην περίπτωση του πίνακα αυτουνού το rent a car id είναι ξένο κλειδί συνδέοντας το αυτοκίνητο η την μηχανή με το αντίστοιχο γραφείο.

- **trip_excursion (**excrursion_id, created, agency_id,days,destination_from,destination_to,excursion_description,excursion_location,hours,trip_fee**).** Περιλαμβάνει όλες τις εκδρομές και τις πληροφορίες που αφορούν αυτές. Το πεδίο agency_id λειτουργεί σαν ξένο κλειδί στο πίνακα αυτόν και συνδέει τις εκδρόμές με τα γραφεία ταξιδίων από τα οποία προσφέρονται.

- **Trip_hotel(**hotel_id, hotel_name, address, category, number_of_rooms, phone_number, created**)** περιλαμβάνει τα ξενοδοχεία καθώς και τις πληροφορίες που αφορούν αυτά. Τα ξενοδοχεία είναι χωρισμένα ανά περιοχή.

- **trip_museum (**museum_id, created, entrance_fee, category, museum_description, museum_location,museum_name, working_days,working_hours**)** Ο πίνακας museum περιλαμβάνει όλα τα μουσεία καθώς και τις πληροφορίες όπως μερες και ώρες λειτουργίας , το εισιτήριο που απαιτείται για την είσοδο κτλ. Τα μουσεία χωρίζονται ανά περιοχή.

- **trip_rent_a_car (**rent_ id, rent_address, created, rent_number_of_cars, rent_name**).** Ο πίνακας αυτός περιλαμβάνει όλα τα γραφεία ενοικιάσεως αυτοκινήτων και τις πληροφορίες που αφορούν αυτά. Τα γραφεία ενοικιάσεως είναι χωρισμένα ανα περιοχή.

- **trip_reservation (**reservation_id, user_id, hotel_id, room_id, day_in, day_out, car_id, car_in, car_out, total, rent_a_car_id, status, created**)** περιλαμβάνει όλα τα reservarions καθώς και τις πληροφορίες που αφορούν αυτά. Έχει σαν ξένα κλειδιά του κωδικούς αυτοκινήτου, δωματίου, ξενοδοχείου, γραφείου ενοικιάσεων και χρήστη. Πέρα από αυτά περιλαμβανει πληροφορία για το σύνολο που έχει πληρώσει ο χρήστης καθώς και την κατάσταση για την οποία βρίσκεται το reservation.

- **trip_room(**room_id, , created, hotel_id, price, room_category**)** ο πίνακας περιλαμβάνει πληροφορίες για τα δωμάτια όπως τιμή ανά ημέρα και ξενοδοχείο στο οποίο ανήκουν.

- **trip_route(**route_id, created, route_category, route_days, route_from, route_time, route_to**)** περιλαμβάνει πληροφορίες για τα δρομολόγια αεροπορικά, ακτοπλοικά και αλλά.

- **trip_travel_agency(**agency_id, agency_address,agency_location,agency_name,agency_created**)** Ο πίνακας αυτός περιλαμβάνει τα γραφεία ταξιδιών καθώς και πληροφορίες για αυτά. Τα γραφεία ταξιδιών αναγνωρίζονται ανά περιοχή.

4.2 Κύριο Μέρος της Εφαρμογής

Όπως είπαμε παραπάνω η εφαρμογή είναι γραμμένη σε Java ποιο συγκεκριμένα για το χτίσιμό της χρησιμοποιήσαμε το Spring Framework.

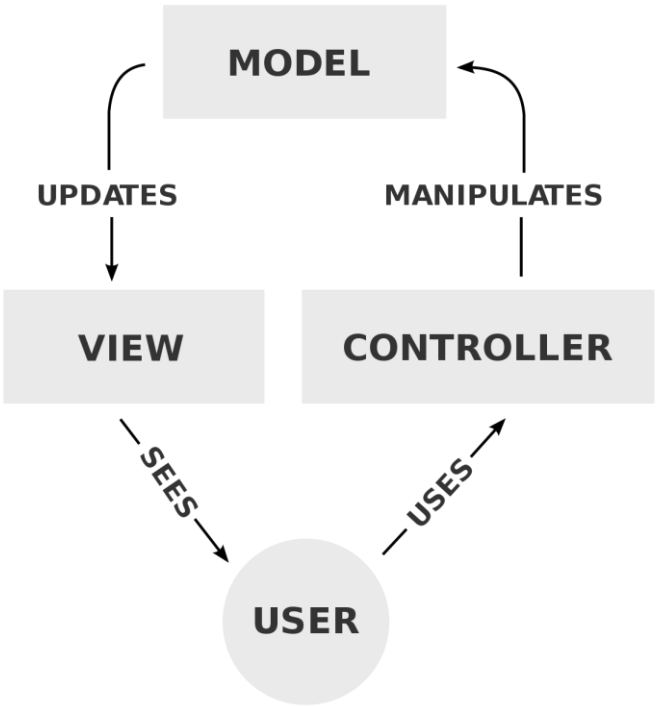
Η εφαρμογή χωρίζεται σε 3 μέρη.

Τις σελίδες του front end που θα τις αναλύσουμε παρακάτω και αποτελούν το view. Εδώ είναι που γίνεται η αλληλεπίδραση με τον χρήστη

Τα μοντέλα που ουσιαστικά είναι οι κλάσεις POJO που αναφέραμε πιο πάνω και σαν σκοπό έχουν την οπτική απεικόνιση των δεδομένων που εμφανίζονται στον χρήστη.

Τέλος τα controllers τα οποίο προσφέρουν στο model view δεδομένα μοντέλων αλλά και ερμηνεύουν πράξεις του χρήστη όπως είναι κάποιο κλικ σε κάποιο κουμπί ή το submit κάποιας φόρμας και κάνουν ανάλογα τις διάφορες ενέργειες.

Παρακάτω βλέπουμε μια οπτική απεικόνιση του πως αυτά τα τρία λειτουργούν μαζί σε ένα MVC Framework:



Εικόνα 14

4.2.1 Μοντέλα - Models

Τα μοντέλα ή models ή POJO κλάσεις είναι τα δομικά στοιχεία που αποτελούν την εφαρμογή και η προγραμματιστική απεικόνιση κάθε πληροφορίας που βλέπει ο χρήστης στις σελίδες της εφαρμογής.

Μερικές από τις κλάσεις αυτές γίνονται και map για την δημιουργία των πινάκων στην βάση. Τα περισσότερα μοντέλα έχουν ως σκοπό να επεξεργαστούν από τον server ανάλογα του τι εντολές δέχεται από τον χρήστη και να επιστέφονται και πάλι στον χρήστη σύμφωνα με τις αλλαγές που ζήτησε.

Τα μοντέλα που χρησιμοποιούνται από την εφαρμογή είναι τα παρακάτω:

- Attraction
  Η κλάση attraction είναι η κλάση POJO για να δημιουργήσουμε τον πίνακα attraction και περιέχει όλες τις απαραίτητες πληροφορίες για τα αξιοθέατα που βρίσκονται σε κάθε περιοχή.

- Car
  Η κλάση car είναι η κλάση POJO που γίνεται Map για την δημιουργία του πίνακα Car που περιέχει όλες τις απαραίτητες πληροφορίες για την αυτοκίνητα και τις μηχανές που χρησιμοποιεί το σύστημα.

- Excursion
  Η κλάση excursion είναι η POJO κλάση που χρησιμοποιούμε για να κανουμε Map και να δημιουργήσουμε τον πίνακα excursion στην βάση. Μια λίστα με Excursions επιστρέφεται όταν ο χρήστης ζητάει να δει τι εκδρομές προσφέρει το κάθε γραφείο ταξιδίων.

- Hotel
  Η POJO κλάση Hotel γίνεται Map για να φτιαχτεί στην βάση ο πίνακας Hotel και αποτελεί ουσιαστικά την προγραμματιστική απεικόνιση για το ξενοδοχείο στην εφαρμογή. Η κλάση αυτή χρησιμοποιείτε στις περιπτώσεις που ο χρήστης θέλει να κλείσει άμεσα η να αναζητήσει ένα ξενοδοχείο.

- Museum
  Η POJO κλάση Museum είναι εκείνη που μέσω τον annotations του JPA/Hibernate κάνουμε map για να δημιουργήσουμε τον πίνακα με τα μουσεία.

- MyReservations
  Η κλάση MyReservations χρησιμοποιείτε και αντιπροσωπεύει την πληροφορία που επιστρέφει πίσω στον χρήστη όταν στο userpage.html εμφανίζεται η λίστα με τις κρατήσεις που έχει κάνει.

- RentACar
  Η κλάση RentACar είναι και αυτή μια POJO κλάση η οποία γίνεται Map για να δημιουργήσει τον πίνακα των γραφείων ενοικιάσεως αυτοκινήτων στην βάση.

Χρησιμεύει για να μεταφέρει πληροφορία για να μπορέσει ο χρήστης να κάνει αναζητήσεις η να κάνει κάποια κράτηση.

- Request
  Το Request είναι μια κλάση που χρησιμοποιείτε για να αναπαραστήσει το σύνολο των πληροφοριών που στέλνει ο χρήστης για να κάνει μια κράτηση η μια αναζήτηση για κράτηση.

- Reservation
  Το reservation είναι και αυτή ένα μοντέλο και παράλληλα μια POJO κλάση που μέσω αυτής φτιάχνεται ο πίνακας reservations στο σύστημα. Αποτελεί την βασική απεικόνιση πληροφορίας για να γίνει μια κράτηση και για την αναζήτηση.

- Response
  Η κλάση response χρησιμοποιείτε κατά την λειτουργία της κράτησης η της αναζήτησης κρατήσεων. Επιστέφει τα αντικείμενα της αναζήτησης που κάναμε για να δημιουργήσει το τελικό αποτέλεσμα.

- Result
  Τα Results είναι οι κλάσεις που είναι υπεύθυνες για να μεταφέρουν τα αποτελέσματα της αναζήτησης πίσω στον χρήστη. Μεταφέρουν πληροφορία όπως ονομα ξενοδοχείου χρώσεις κτλ.

- Room
  Room είναι η κλάση την οποία κάνουμε Map για την δημιουργία του πίνακα Room στην βάση και τελικά η προγραμματιστική απεικόνιση των δωματίων μέσα στον κώδικά.

- Route
  Το Route είναι η POJO κλάση που χρησιμοποιούμε για την δημιουργία του πίνακα routes στην βάση και η προγραμματιστική απεικόνιση των δρομολογίων στην εφαρμογή.

- TravelAgency
  Η κλάση TravelAgency είναι που γίνεται Map για την δημιουργία του πίνακα TravelAgency Και αντιπροσωπεύει όλα τα ταξιδιωτικά γραφεία. Χρησιμοποιείται σαν λίστα αποτελεσμάτων για να γεμίσει δυναμικά τον dropdown για την επιλογή γραφείου ταξιδιών στην αναζήτηση εκδρομών.

- Users
  Η κλάση users είναι αυτή που αντιπροσωπεύει τους χρήστες της εφαρμογής. Γίνεται Map για να δημιουργηθεί ο πίνακας users στην βάση και ουσιαστικα πάιρνει μέρος στην διαδικασία της εγγραφής και της εισόδου στην εφαρμογή.

4.2.2 Controllers

Τα Controllers όπως είπαμε και παραπάνω έχουν σαν δουλειά να δέχονται τα «ερωτήματα» του χρήστη και να του απαντάνε με το ανάλογα τι ζητάει.

Τα Controllers είναι και αυτά απλές κλάσεις οι οποίες διαμορφώνονται με τα annotations του Spring.

Στην περίπτωση της εφαρμογής μας χρησιμοποιούμε RestControllers. Η διαφορά από τους απλούς controllers του Spring είναι ότι στους απλούς controllers η απάντηση του Spring βασίζεται στο view. Στην περίπτωση του RestController το response γράφεται κατευθείαν στο HTTP response body. Το RestController μπορεί να επιστρέψει ακόμα και ένα ολόκληρο αντικείμενο το οποίο θα μετατραπεί σε μορφή JSON και θα ερμηνευτεί ανάλογα από το front-end.

```
package com.serkid.controllers;


import ...

@RestController
@RequestMapping("/travel")
public class TravelAgencyController {

    @Autowired
    private TravelAgencyRepository myTravelAgencyRepo;

    @RequestMapping(value = "/agency" ,method = RequestMethod.GET)
    public ResponseEntity<List<TravelAgency>> getAgenciesByArea (@RequestParam(value = "location") String location)
    {

        List<TravelAgency> myTravelAgencies = myTravelAgencyRepo.getTravelAgenciesByAgencyLocation(location);
        return new ResponseEntity<>(myTravelAgencies, HttpStatus.OK);
    }


}
```

Εικόνα 15

Στην παραπάνω εικόνα βλέπουμε ένα παράδειγμα RestController

Το annotation **@RestController** χρησιμοποιείται για να δηλώσουμε ότι η κλάση που ακολουθεί θα οριστεί σαν ένας RestController. Στην συνέχεια με το @RequestMapping αναθέτουμε συγκεκριμένα web requests σε έναν συγκεκριμένο handler. Στην προκειμένη περίπτωση αναθέσαμε σε ολόκληρη την κλάση να ακούει στο request '/travel'.

Στη συνέχεια κάνουμε @Autowire το repository που αφορά την συγκεκριμένη ****

Στην συνέχεια έχουμε την μέθοδο **getAgenciesByArea**. Την συγκεκριμένη μέθοδο την αναθέτουμε στο request "/agency". Η συγκεκριμένη μέθοδος επιστρέφει όλα τα γραφεία ταξιδιών σε μια JSON λίστα, ανάλογα το location που δίνει ο χρήστης και επιστρέφει. Στην συγκεκριμένη μέθοδο εκτός από την ανάθεση του Request που της έχει γίνει έχει δηλωθεί και το RequestMethod που είναι στην συγκεκριμένη περίπτωση το GET

Παρακάτω αναφέρονται όλα τα controllers που υπάρχουν στην εφαρμογή μαζί με τα request που τους έχουν ανατεθεί και τις μεθόδους που περιέχουν καθώς και οι λειτουργίες τους:

- AttractionController

  Το Attraction Controller έχει χαρτογραφηθεί με το request “/attraction” και περιέχει την μέθοδο **getAttractionsbyArea** η οποία έχει σαν σκοπό να επιστρέφει μια λίστα από όλα τα αξιοθέατα ανά περιοχή.

- ExcursionController

  Το ExcursionController είναι η κλάση που εχει οριστεί και χαρτογραφηθεί να ακούει στο request “/excursion” και περιέχει την μέθοδο **getTours** που έχει σαν δουλειά να επιστρέφει μια λίστα με όλες τις εκδρομές ανά γραφείο ταξιδιών που επιλέγουμε.

- HotelController

  Είναι η κλάση που εχει οριστεί ως ο controller των hotel και αν και δεν χρησιμοποιήθηκε εν τελη ακούει στο endpoint “/hotel”

- LoginController

  Ο LoginController αναλαμβάνει 3 εργασίες την εγγραφή ενός χρήστη, την είσοδο του και την δημιουργία session και τέλος το logout. Αποτελεί αυτός έναν Rest Controller και του έχει ανατεθεί το request “/login”. Για τις υπόλοιπες λειτουργίες του χρησιμοποιεί την μέθοδο loginUser που απαιτεί ένα username και ένα password και κάνει τον έλεγχο αν είναι εγγεγραμμένος ο χρήστης. Το registerUser δέχεται όλες τις απαραίτητες πληροφορίες για την εγγραφή του χρήστη ελέγχει αν ο χρηστής υπάρχει ήδη στο σύστημα και στην συνέχεια ανάλογα σε συνδέει η όχι.

- MuseumController

  Το MuseumController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/museum". Περιέχει τη μέθοδο getMuseumsbyArea η οποία έχει ως λειτουργία να επιστρέφει μια λίστα με όλα τα μουσεία ανάλογα με την τοποθεσία που έχει επιλέξει ο χρήστης.

- ReservationController

  Το ReservationController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/reservation". Ο controller αυτός είναι ο σημαντικότερος της εφαρμογής. Η μέθοδος **checkAvailability** είναι η κύρια μέθοδος του controller καθώς είναι υπεύθυνη για τη δημιουργία κρατήσεων. Αρχικά, παίρνει ως παράμετρο ένα αντικείμενο που είναι κλάσης **Request** το οποίο περιέχει όλα τα

στοιχεία που έχει επιλέξει ο χρήστης. Ελέγχει αν ο χρήστης έχει επιλέξει να αναζητήσει είτε ξενοδοχεία και αυτοκίνητα είτε μόνο ξενοδοχεία, και τρέχει τις αντίστοιχες συναρτήσεις. Η συνάρτηση **checkHotelAvailability** βρίσκει δωμάτια σύμφωνα με τον τύπο ξενοδοχείου που έχει επιλέξει ο χρήστης καθώς και με τη χωρητικότητα του δωματίου. Στη συνέχεια, ελέγχει αν το δωμάτιο έχει κάποιες κρατήσεις από τον πίνακα κρατήσεων. Αν το δωμάτιο δεν έχει κρατήσεις τότε το δωμάτιο κατοχυρώνεται άμεσα και σταματάει η διαδικασία της αναζήτησης. Στην περίπτωση που το δωμάτιο αυτό έχει κάποιες κρατήσεις τότε ελέγχεται αν η χρονική περίοδος που έχει επιλέξει ο χρήστης συμπίπτει με κάποια από τις χρονικές περιόδους των άλλων κρατήσεων. Αν δεν συμπίπτει τότε το δωμάτιο κατοχυρώνεται και επιστρέφεται. Στην περίπτωση που ο χρήστης έχει επιλέξει και αυτοκίνητα τότε ενεργοποιείται η μέθοδος **checkCarAvailability.**

Με τον ίδιο τρόπο που λειτουργεί το checkHotelAvailability έτσι λειτουργεί και αυτή η μέθοδος. Επιλέγει τα αυτοκίνητα με γνώμονα την περιοχή τον τύπο και τον αριθμό των θέσεων και στην συνέχει ελέγχει αν έχουν κάποιες κρατήσεις. Τέλος επιστρέφει το αυτοκίνητο που εντόπισε. Αφού συγκεντρώσει όλες τις απαραίτητες πληροφορίες κάνει κανονικά την κράτηση υπολογίζοντας τον αριθμό των μερών καθώς και το τι θα πληρώσει ο χρήστης σαν σύνολο αποθηκεύοντας την κράτηση ως ενεργή.

Αν για κάποιο λόγο κάποιο από αυτά τα στοιχεία που ζήτησε ο χρήστης δεν βρεθεί τότε ο controller επιστρέφει μήνυμα ότι δεν βρέθηκαν όλα αυτά που αναζητούσε ο χρήστης.

Επίσης ο controller αυτός περιέχει την μέθοδο **userReservations** στην οποία έχει ανατεθεί το request “/user” που έχει σαν λειτουργία να επιστρέφει όλα τα reservations του χρήστη.

Στην συνέχεια στον ίδιο controller βρίσκεται και η μέθοδος **cancelReservation** η οποία παίρνει σαν παράμετρο το id μιας κράτησης και στην συνέχεια την ακυρώνει αλλάζοντας το στάτους της.

Τέλος η μέθοδος **bookReservation** χρησιμοποιείται για να γίνεται μια κράτηση όταν επιλέγει ο χρήστης τα στοιχεία που θέλει σύμφωνα με την αναζήτηση που είχει κάνει.

- RoomController

  Το RoomController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/room". Έχει την μία και μοναδική μέθοδο η οποία είναι το **getAllRooms** η οποία επιστρέφει στον χρήστη όλα τα δωμάτια της βάσης.

- RouteController

Το RouteController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/routes".

- SearchController

  Το SearchController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/ search". Περιέχει την κύρια μέθοδο **searchFor** η οποία όπως και στην περίπτωση του ReservationController παίρνοντας ως παράμετρο ένα στιγμιότυπο της κλάσης Request που περιέχει όλες τις απαραίτητες πληροφορίες που ζήτησε ο χρήστης επιστρέφει μια λίστα όλα τα διαθέσιμο δωμάτια και οχήματα αν ο χρήστης ζήτησε για αυτά.

- TravelAgencyController

  Το TravelAgencyController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/travel". Με την μία και μοναδική του μέθοδο που είναι η **getAgenciesByArea** στην οποία έχει ανατεθεί το request “/agency” επιστρέφει όλα τα ταξιδιωτικά γραφεία ανά περιοχή.

- UserController

  Το UserController είναι η κλάση που έχει οριστεί ως Rest Controller και του έχει ανατεθεί το request "/user". Περιέχει την μέθοδο **getUser** η οποία σαν κύρια λειτουργία είναι να επιστρέφει τα στοιχεία του χρήστη για να ενημερωθεί το front end της σελίδας.

4.2.3 Repositories

Τα Repositories είναι μέρος του Spring Data και έχουν σαν σκοπό την μείωση της ποσότητας του κώδικα που χρειάζεται για την δημιουργία του data layer.

Πιο συγκριμένα όλα τα repositories που χρησιμοποιούμε για την δημιουργία του δικού μας data layer είναι επεκτάσεις του CrudRepository. Το CrudRepository που η ονομασία του προκύπτει από τα αρχικά CRUD (Create, Read, Update, Delete) προσφέρει μια πιο εκλεπτυσμένη λύση για την για τις διεργασίες για την κλάση για την οποία υλοποιείται.

Παρακάτω έχουμε τα interfaces τα οποία είναι επεκτάσεις του CrudRepository καθώς και τις μεθόδους που αυτά ορίζουν.

- *AttractionRepository*

  Το Attraction Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα attraction και περιέχει τις παρακάτω συναρτήσεις

  1. **getAttractionById**(long id) η οποία επιστρέφει μια attraction σύμφωνα με τον κωδικό που του έχουμε δώσει.
  2. **getAttractionsByLocation**(String location) η οποία επιστρέφει μια λίστα από locations σύμφωνα με την τοποθεσία που του έχουμε δώσει.

- *CarRepository*

Το Car Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα car και περιέχει τις παρακάτω συναρτήσεις.

1. **getAllByRentACarIdAndNumberOfSeatsAndCarCategory**(long rentACarId, int numberOfSeats, String category)Επιστρέφει όλα τα αυτοκίνητα σύμφωνα με τις δοσμένες παραμέτρους.

- *ExcursionRepository*

  Το Excursion Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα Excursion και περιέχει τις παρακάτω συναρτήσεις.

  1. **getExcursionByExcursionId**(long excursionId) επιστρέφει μια εκδρομή ανάλογα το δοσμένο Id
  2. **getExcursionsByAgencyId(**long agencyId**)** επιστρέφει μια λίστα με όλες τις εκδρομές σύμφωνα με το ποιο γραφείο ταξιδιών ανήκουν
  3. **getExcursionσByExcursionLocation**(String location) επιστρέφει όλες τις εκδρομές ανάλογα την τοποθεσία στην οποία βρίσκονται.

- *HotelRepository*

  Το Hotel Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα Hotel και περιέχει τις παρακάτω συναρτήσεις.

  1. **getAllByAddressAndCategory**(java.lang.String address, int category) επιστρέφει όλα τα ξενοδοχεία σύμφωνα με την διεύθυνση και την κατηγορία.
  2. **getByAddress**(java.lang.String address) επιστρέφει όλα τα ξενοδοχεία στην βάση σύμφωνα με την δοσμένη διεύθυνση.
  3. **getByAddressLike**(java.lang.String address) επιστρέφει όλα τα ξενοδοχεία σύμφωνα με την διευθυνση.
  4. **getByHotelId**(long hotelId) επιστρέφει ένα ξενοδοχείο σύμφωνα με το id του ξενοδοχείου.

- *MuseumRepository*

  Το Museum Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα Museum και περιέχει τις παρακάτω συναρτήσεις.

  1. **getMuseumByMuseumId**(long museumId) επιστρέφει το μουσείο σύμφωνα με το id που του δίνουμε.
  2. **getMuseumsByMuseumLocation**(java.lang.String location) επιστρέφει μια λίστα με τα μουσεία που βρίσκονται στην εκάστοτε περιοχή.

- *RentACarRepository*

Το RentACar Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα RentACar και περιέχει τις παρακάτω συναρτήσεις.

1. **getAllByRentAddress**(String rentAddress) επιστρέφει όλα τα γραφεία ενοικιάσεως αυτοκινήτων σύμφωνα με την τοποθεσία στην οποία βρίσκονται.
2. **getByRentId**(long rentId) επιστρέφει το γραφείο ενοικιάσεως αυτοκινήτων σύμφωνα με το rentId.

- *ReservationRepository*

Το Reservation Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα Reservation και περιέχει τις παρακάτω συναρτήσεις.

1. **getByReservationsId**(long reservationsId) επιστρέφει ένα reservation σύμφωνα με το id που του έχουμε δωσει.
2. **getReservationByRoomId**(long roomId) επιστρέφει τα reservations τα οποία αφορούν ένα συγκεκριμένο roomId.
3. **getReservationByUserId**(long userId) επιστρέφει όλα τα reservations τα οποία αφορούν έναν συγκεκριμένο χρήστη.
4. **getReservationByUserIdAndStatus**(long userId, String status) επιστρέφει όλα τα reservations τα οποία αφορούν έναν έναν συγκεκριμένο χρήστη και ανταποκρίνονται σε ένα συγκεκριμένο status.

- *RoomRepository*

Το Room Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα Room και περιέχει τις παρακάτω συναρτήσεις.

1. **getByHotelIdAndRoomCategory**(long hotelId, int roomCategory) επιστρέφει όλα τα δωμάτιο σύμφωνα με το roomId και σύμφωνα με το roomcategory.

- *RouteRepository*

Το Route Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα route και περιέχει τις παρακάτω συναρτήσεις.

1. **getRouteByRouteId**(long routeId) επιστρέφει ένα route σύμφωνα με το δοσμένο routeId.
2. **getRoutesByRouteCategory**(java.lang.String routeCategory) Επιστρέφει όλα τα routes σύμφωνα με το δοσμένο route category.

3. **getRoutesByRouteTo**(java.lang.String routeTo) επιστρέφει όλα τα routes σύμφωνα με το πεδίο routeTo.

- *TravelAgencyRepository*

  Το TravelAgency Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για τον πίνακα TravelAgency και περιέχει τις παρακάτω συναρτήσεις.

  1. **getTravelAgenciesByAgencyLocation**(java.lang.String agencyLocation) επιστρέφει όλα τα travel agencies σύμφωνα με το agency location

- *UsersRepository*

  Το User Repository είναι το Repository υπεύθυνο για τις οποιεσδήποτε συναλλαγές που γίνονται για την τον πίνακα users και περιέχει τις παρακάτω συναρτήσεις.

  1. **getUsersByUserId**(long userId) επιστρέφει τον user σύμφωνα με το επιλεγμένο user id.
  2. **getUsersByUsernameAndPassword**(java.lang.String username, java.lang.String password) επιστρέφει όλους τους users σύμφωνα με το δοσμένο username και το password.

4.2.4 Λοιπές Κλάσεις.

Μια τελευταία κλάση που συναντάμε στο project είναι η κλάση **Validator**

```
public class Validator {
    public static boolean checkUser(UsersRepository userRepo, String
username, String password, HttpSession session)
    {
        boolean isUser = false;
        List<Users> myUsers =
userRepo.getUsersByUsernameAndPassword(username,password);
        if(myUsers != null && !myUsers.isEmpty())
        { isUser = true;
        for (Users myUser: myUsers)
        session.setAttribute("userId",myUser.getUserId());

        }


        return isUser;

    }
}
```

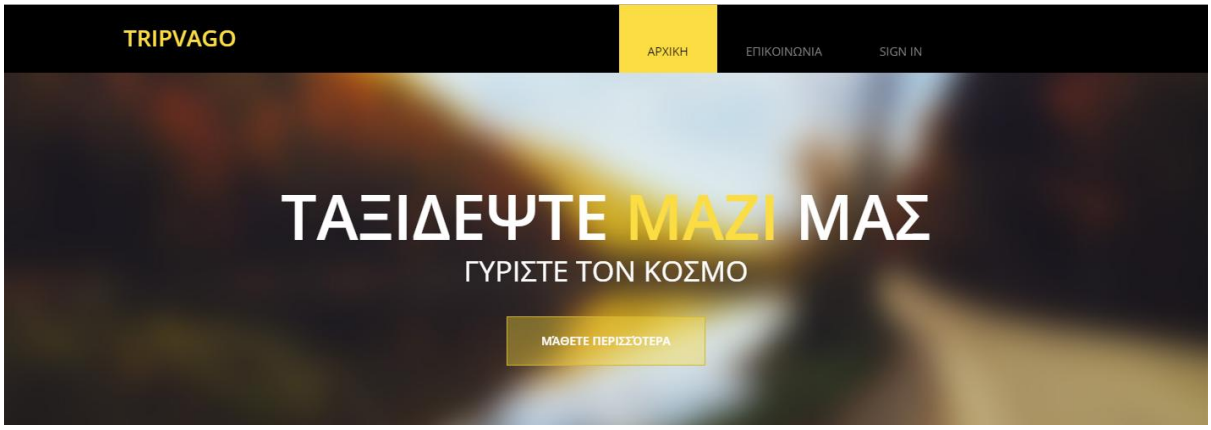
Η κλάση αυτή δεν περιέχει constructor και περιέχει μια και μόνο στατική μέθοδο. Η

checkUser έχει σαν αποκλειστική της δουλεία να ελέγχει αν ο χρήστης που κάνει Login ή Register μέσα στο σύστημα, υπάρχει η όχι καταχωρημένος στον βάση.

4.3 Υλοποίηση του Front-End και Παρουσίαση Εφαρμογής

Εν τέλει και καθώς όλα τα παραπάνω που αναλύσαμε λειτουργούν μαζί σε αρμονία πάμε να δούμε το τελικό αποτέλεσμα της σελίδας.

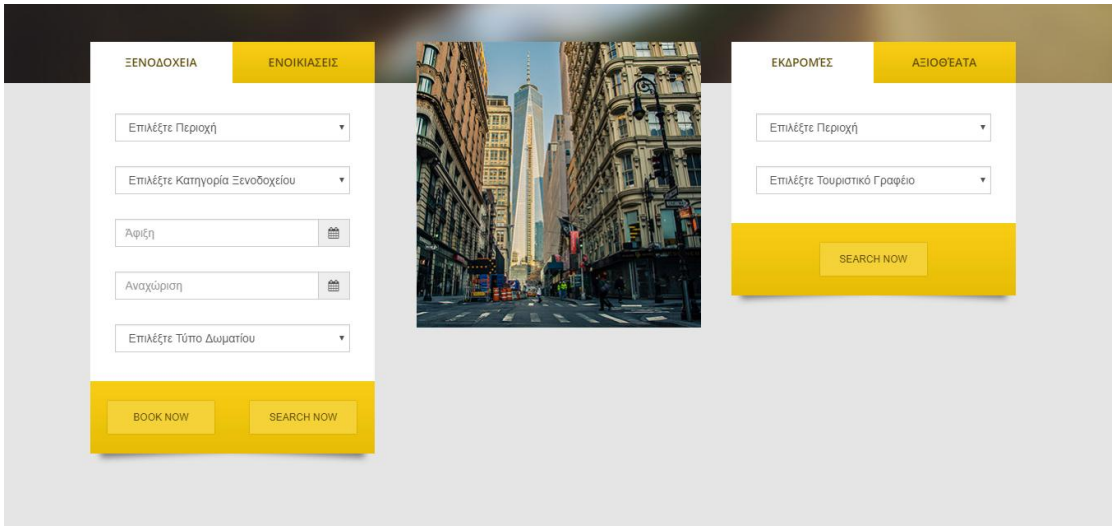
Παρακάτω βλέπουμε στις εικόνες 16 και 17 βλέπουμε την «Αρχική Σελίδα» της σελίδας μας. Στο πάνω μέρος βλέπουμε την μπάρα πλοήγησης με τις τρείς επιλογές με την ενεργή επιλογή να κίτρινη.



Εικόνα 16

Στο κάτω μέρος τις σελίδας έχουμε τις φόρμες με τις οποίες ο χρήστης κάνει όλες τις λειτουργίες του. Στην δεξιά φόρμα ο χρήστης μπορεί να επιλέξει όλα τα απαραίτητα για να κλείσει κάποιο ξενοδοχείο ή να ενοικιάσει κάποιο αυτοκίνητο.

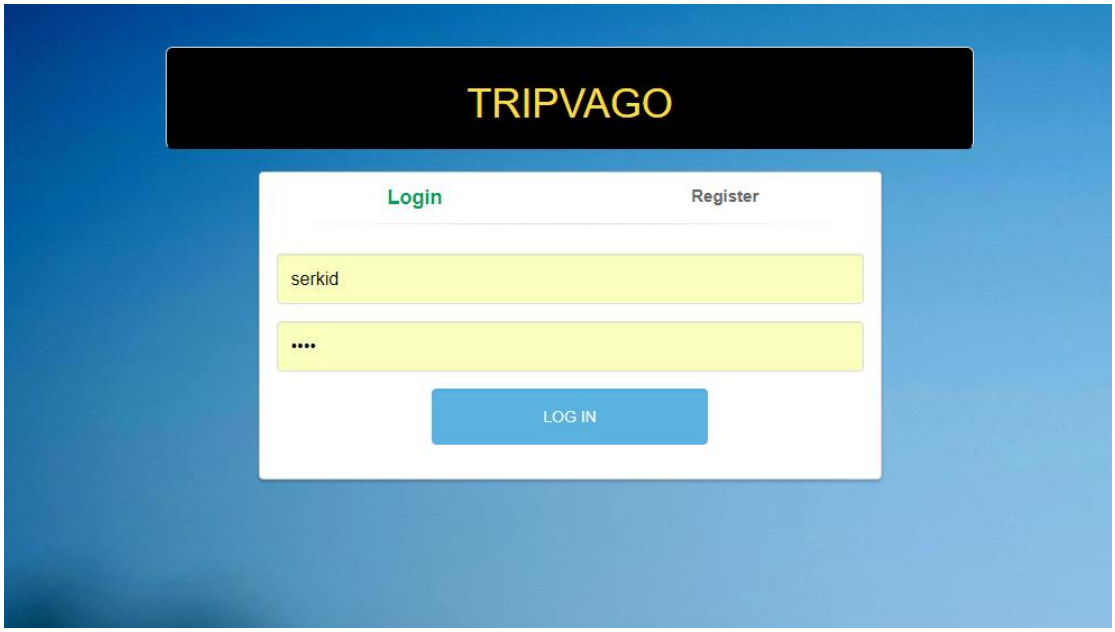
Στην δεξιά μεριά είναι η φόρμα στην οποία ο χρήστης μπορεί να επιλέξει να δει πληροφορίες για τις εκδρομές η τα διάφορα αξιοθέατα ανάλογα την περιοχή που επιθυμεί.



Εικόνα 17

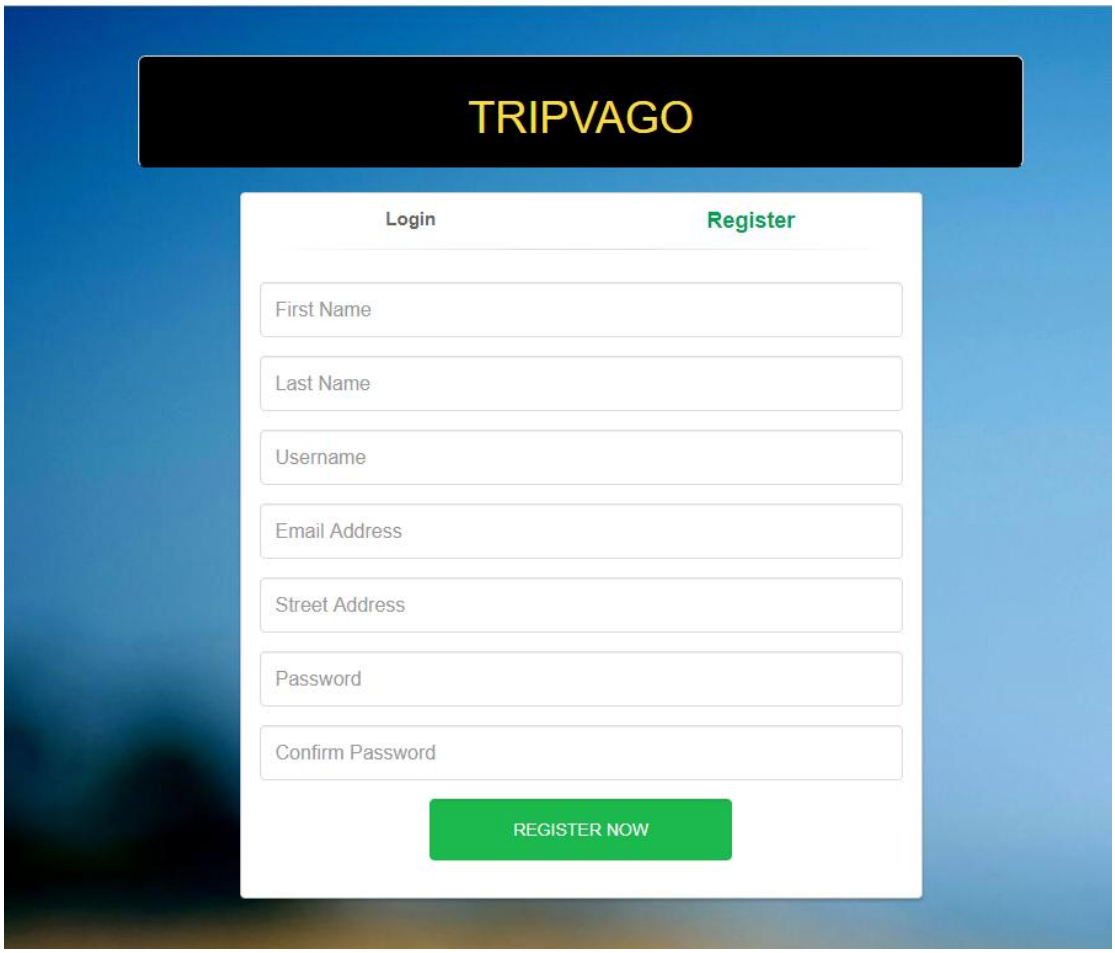
Επιλέγοντας «Sign In» από το navigation menu ο χρήστης μεταφέρετε στην σελίδα login/register.

Στην φόρμα «Login» ο χρήστης μπορεί να εισάγει το username και το password του και να συνδεθεί στην εφαρμογή.



Εικόνα 18

Στην φόρμα “Register” ο χρήστης πρέπει να συμπληρώσει τα απαιτούμενα στοιχεία για να δημιουργήσει έναν λογαριασμό στην βάση.



Εικόνα 19

Αφού συνδεθεί ο χρήστης στην σελίδα το navigation μενού αλλάζει και εμφανίζεται το όνομά του καθώς και η επιλογή “Logout”. Επιλέγοντας το “Logout” ο χρήστης μπορεί να αποσυνδεθεί από την εφαρμογή.

Εικόνα 20

Στην συνέχεια επιλέγοντας το όνομα του ο χρήστης μεταφέρεται στην σελίδα με όλα τις κρατήσεις που έχει κάνει. Στην λίστα ο χρήστης μπορεί να δει την κράτηση που έχει κάνει ποιο ξενοδοχείο και ποιο δωμάτιο έχει κλείσει αλλά και ποιο γραφείο ενοικιάσεως και ποιο όχημα καθώς και την κατάσταση που έχει η κράτηση του αν είναι δηλαδή πληρωμένη η ακυρωμένη, το σύνολο αλλά και την ημερομηνία που έγινε η κράτηση.

Στην συνέχεια προσφέρεται η δυνατότητα στην χρήστη να ακυρώσει την κράτηση του επιλέγοντας το κουμπί «Cancel»

Εικόνα 21

Ο χρήστης στην συνέχεια ερωτάται αν είναι σίγουρα για την επιλογή του αυτή και αν επιλέξει και πάλι “Are you Sure?”

Εικόνα 22

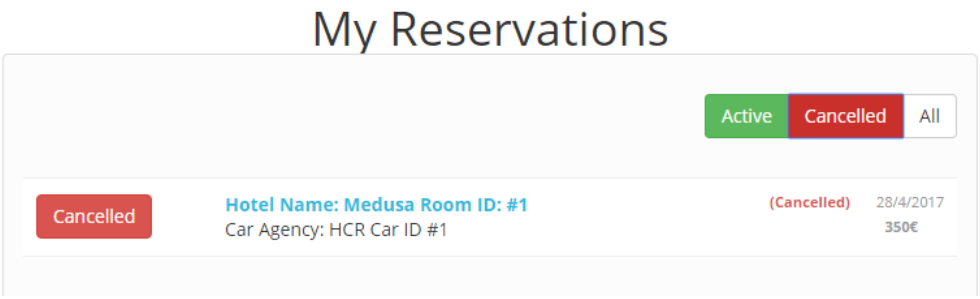
Στην συνέχεια επαναφορτώνει η σελίδα και η συγκεκριμένη κράτηση εμφανίζεται ως ακυρωμένη.

Εικόνα 23

Στο μενού που βρίσκεται πάνω από την λίστα με τις κρατήσεις εμφανίζεται ένα απλό filtering σύστημα. Επιλέγοντας “Active” εμφανίζονται όλες οι ενεργές κρατήσεις που υπάρχουν στην λίστα.

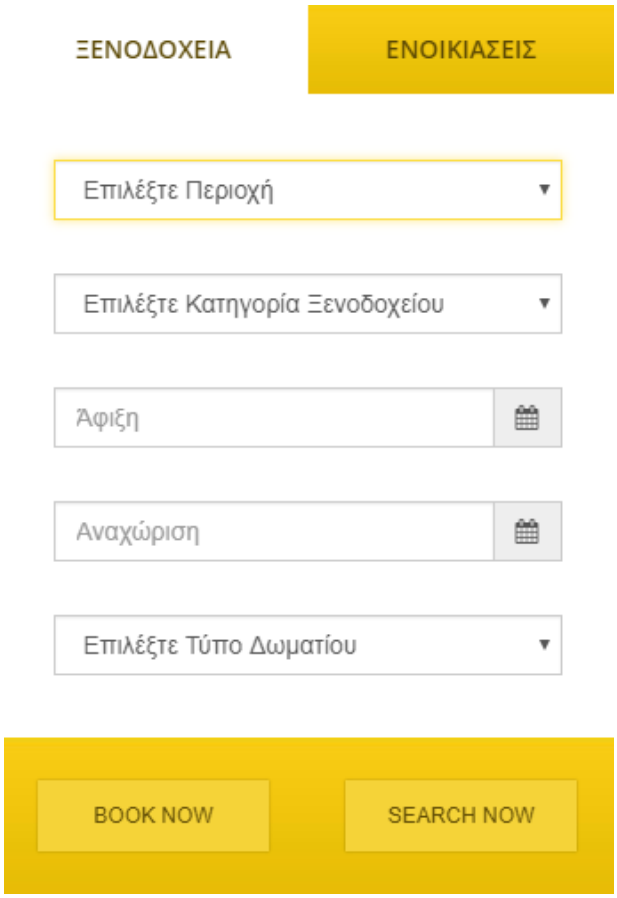
Εικόνα 24

Επιλέγοντας το ‘’Cancelled’’ εμφανίζονται όλες οι ακυρωμένες κρατήσεις και τέλος επιλέγοντας all εμφανίζονται όλες οι ακυρωμένες κρατήσεις.

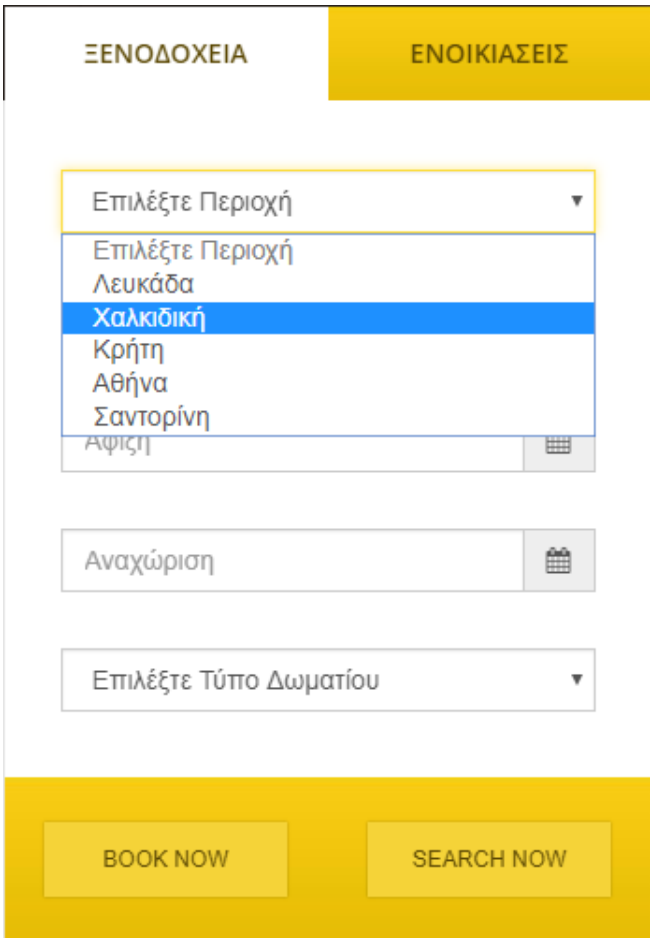


Εικόνα 25

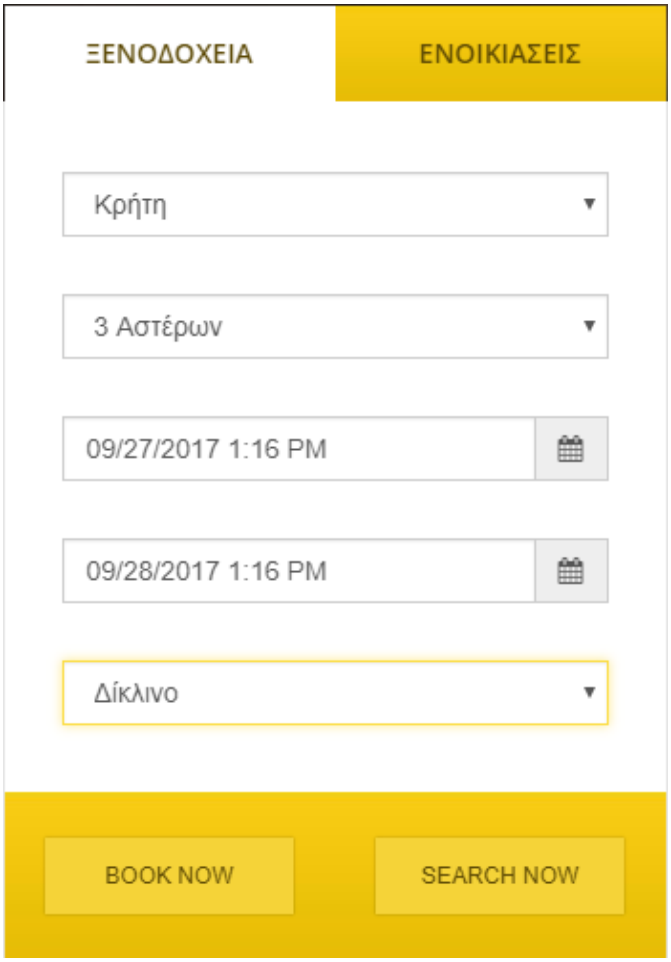
Επιστέφοντας στην αρχική σελίδα ας δοκιμάσουμε να κάνουμε μια κράτηση. Ο χρήστης μπορεί να επιλέξει περιοχή, κατηγορία, άφιξη αναχώρηση, καθώς και τύπο δωματίου. Στο tab ενοικιάσεις επιλέγει ο χρήστης τύπο μεταφορικού, άφιξη-αναχώρηση και αριθμό θέσεων αυτοκινήτου.


ΞΕΝΟΔΟΧΕΙΑ
ΕΝΟΙΚΙΑΣΕΙΣ
Επιλέξτε Περιοχή
Επιλέξτε Κατηγορία Ξενοδοχείου
Άφιξη
Αναχώριση
Επιλέξτε Τύπο Δωματίου
BOOK NOW
SEARCH NOW

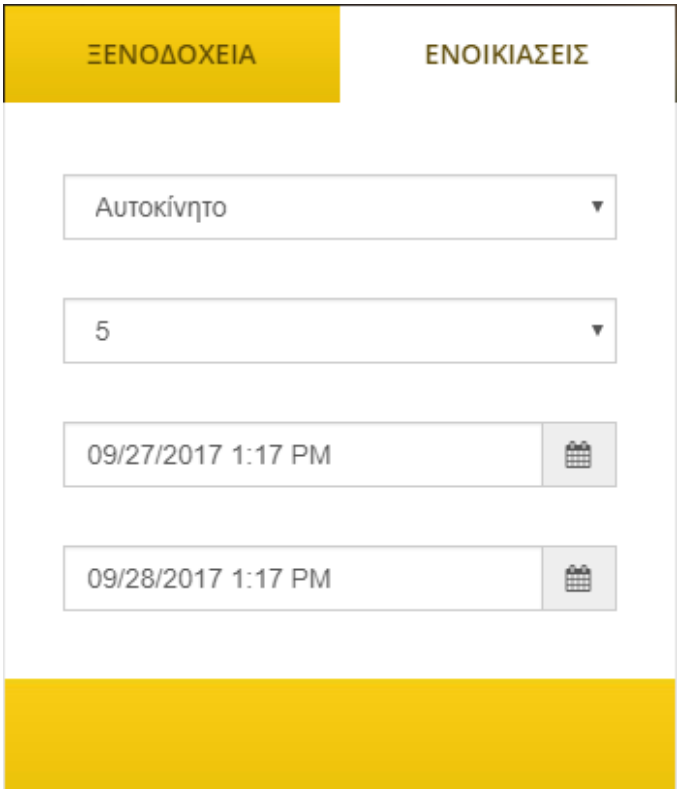
Εικόνα 26


ΞΕΝΟΔΟΧΕΙΑ
ΕΝΟΙΚΙΑΣΕΙΣ
Επιλέξτε Περιοχή
Επιλέξτε Περιοχή
Λευκάδα
Χαλκιδική
Κρήτη
Αθήνα
Σαντορίνη
Αφιξη
Αναχώριση
Επιλέξτε Τύπο Δωματίου
BOOK NOW
SEARCH NOW

Εικόνα 27

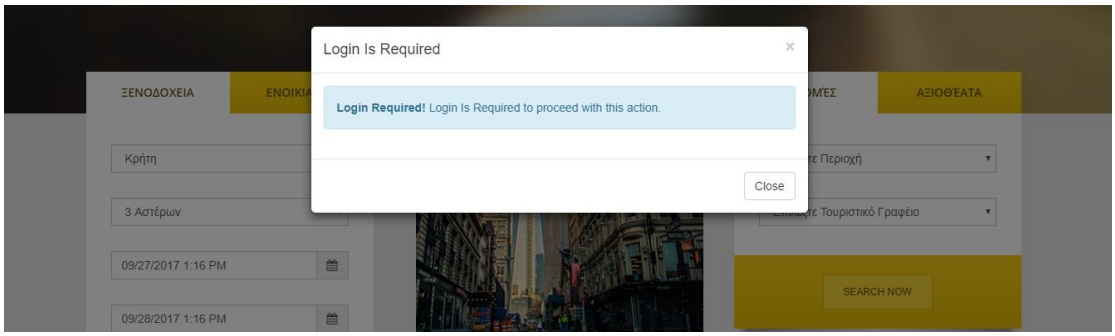


Εικόνα 28



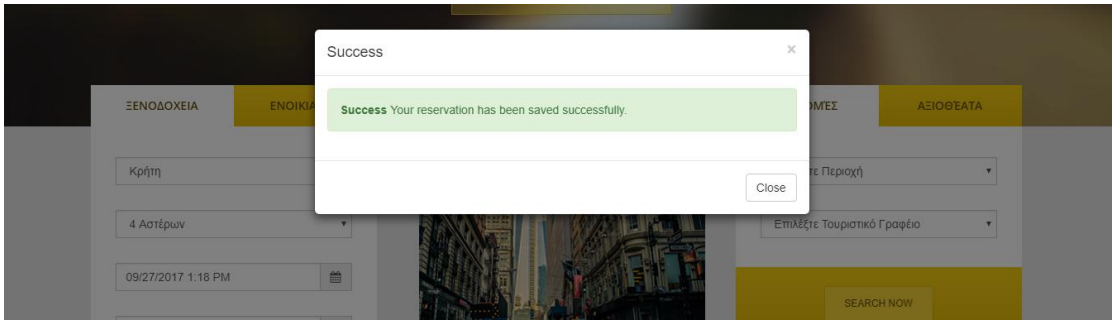
Εικόνα 29

Στην περίπτωση που ο χρήστης δεν έχει κάνει σύνδεση και επιλέξει να κάνει κράτηση εμφανίζεται το παρακάτω μήνυμα.



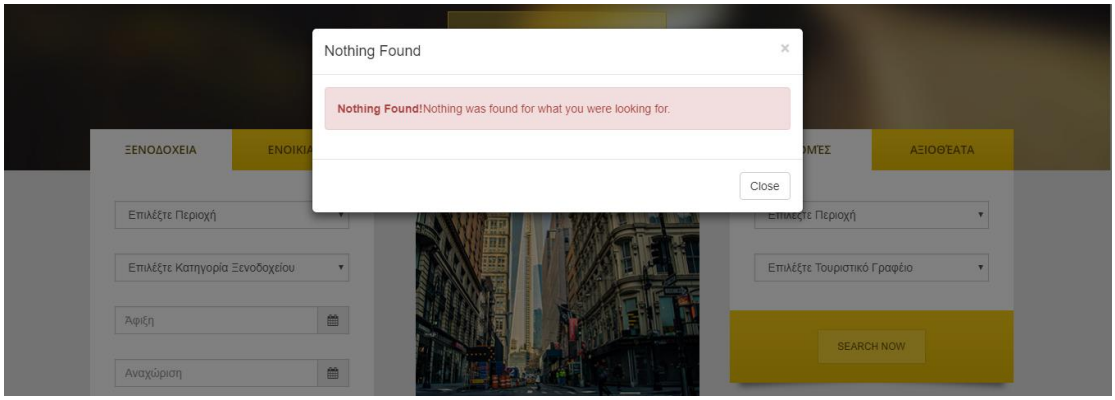
Εικόνα 30

Αν όλα είναι διαθέσιμα και αυτά που ζήτησε ο χρήστης είναι ελεύθερα τότε εμφανίζεται το παρακάτω μήνυμα επιτυχίας.



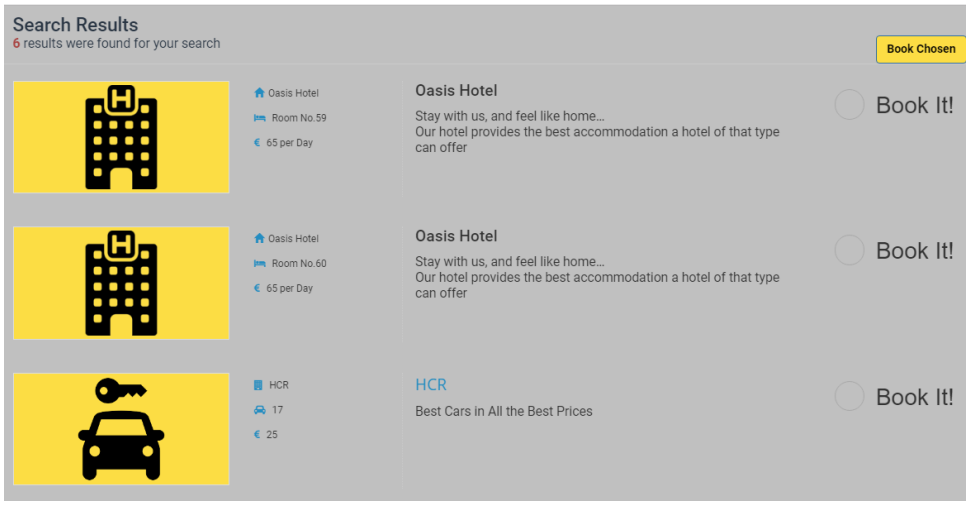
Εικόνα 31

Αν κάτι από αυτά που ζήτησε ο χρήστης δεν είναι διαθέσιμο εμφανίζεται το παρακάτω μήνυμα αποτυχίας.



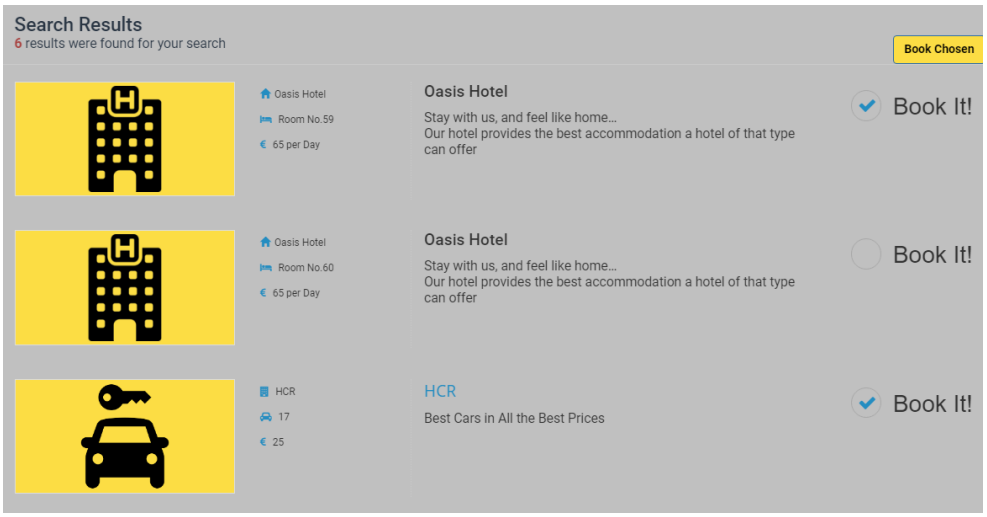
Εικόνα 32

Στην συνέχεια επιλέγοντας “Search” ο χρήστης έχει την δυνατότητα να του επιστραφεί ένας πίνακας με όλα τα αποτελέσματα.



Εικόνα 33

Αφού επιλέξει ο χρήστης οτιδήποτε χρειάζεται από τα αποτελέσματα επιλέγει «Book Chosen» και κάνει την κράτηση που επιθυμεί.

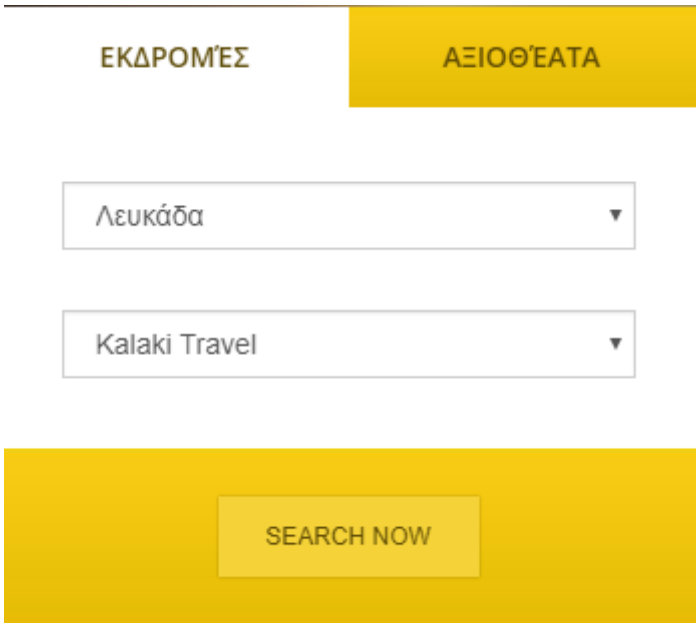


Εικόνα 34

Ανάλογο με το αν κάτι πήγε καλά η στραβά ο χρήστης ενημερώνεται με τα κατάλληλα μηνύματα.

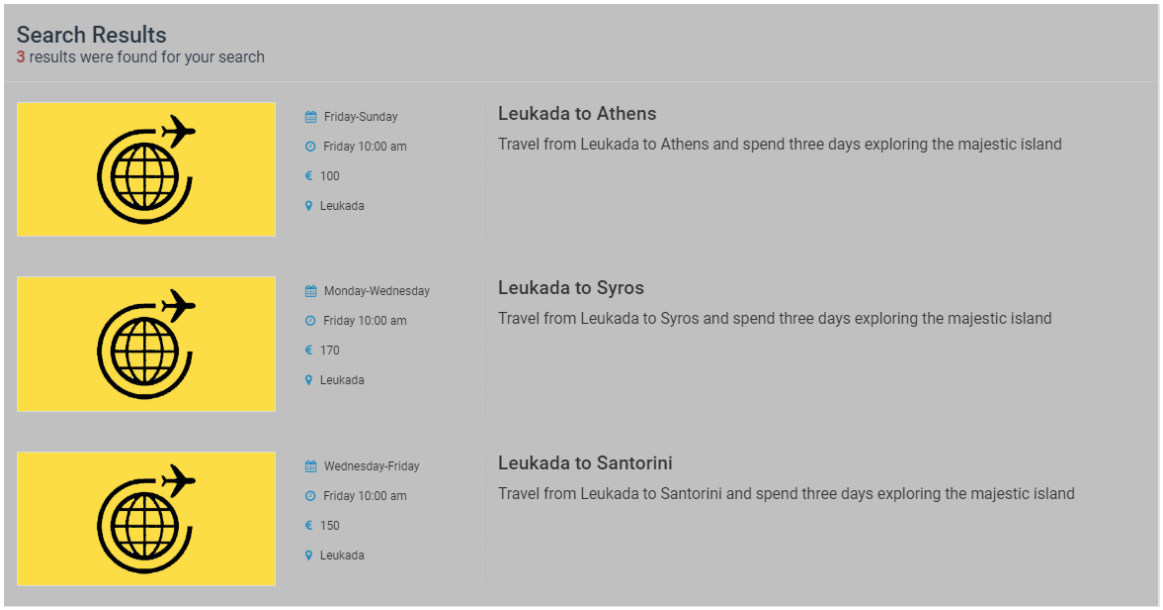
Στην δεξιά μερία αντίστοιχα στο tab «Εκδρομές» ο χρήστης αφου επιλέξει περιοχή γεμίζει το αμέσως επόμενο select με τα διαθέσιμα γραφεία ταξιδιών που υπάρχουν στην περιοχή.

Αφού ο χρήστης επιλέξει «Search Now»



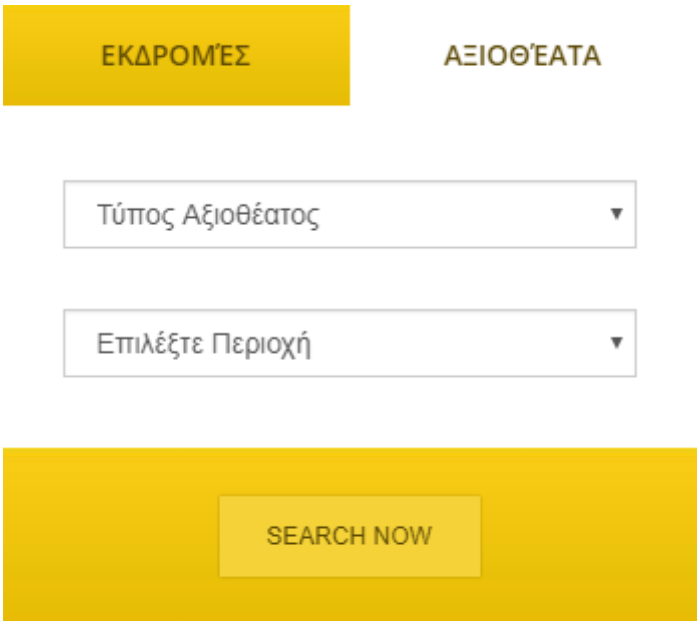
Εικόνα 35

Μπορεί να δει μια λίστα με τις διαθέσιμες εκδρομές που προσφέρει το εκάστοτε γραφείο για συγκεκριμένες περιοχές.

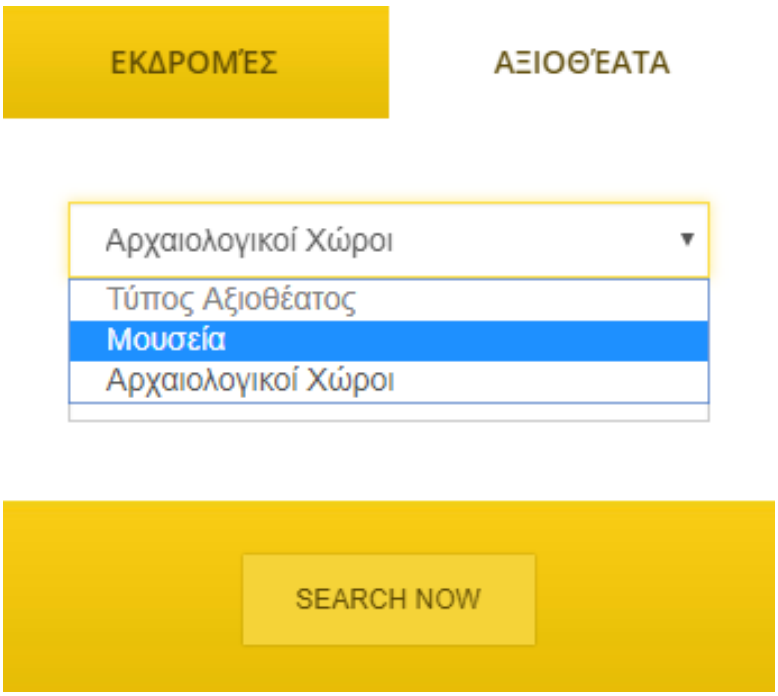


Εικόνα 36

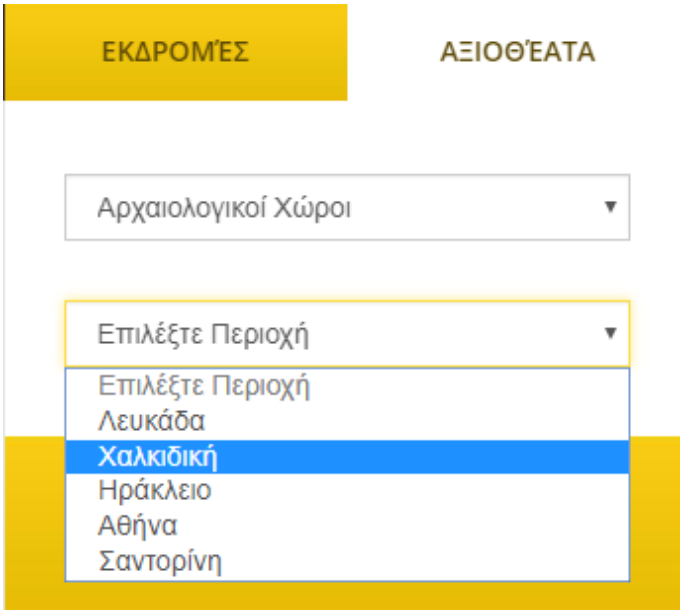
Στο tab «Αξιοθέατα» ο χρήστης μπορεί να επιλέξει τον τύπο αξιοθέατος και στην συνέχεια να επιλέξει την περιοχή για την οποία θέλει να δεί αξιοθέατα.



Εικόνα 37

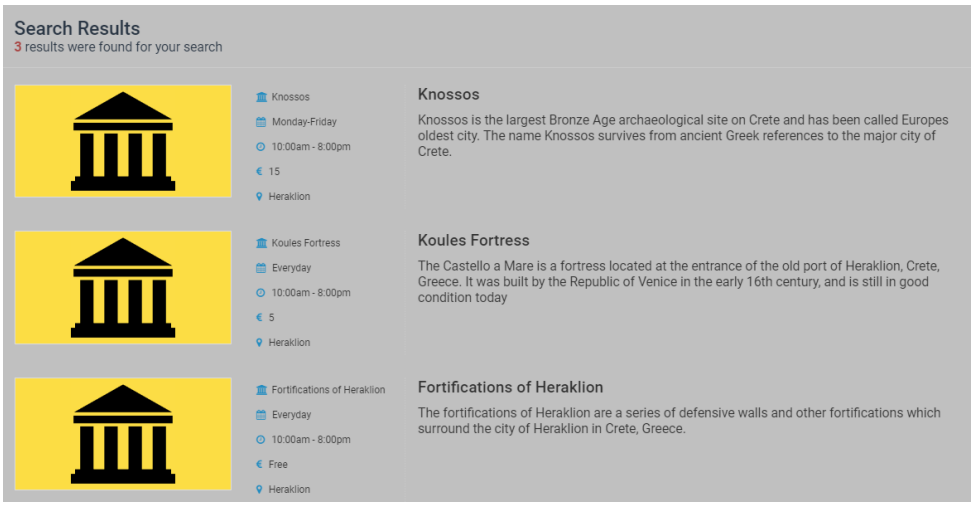


Εικόνα 38



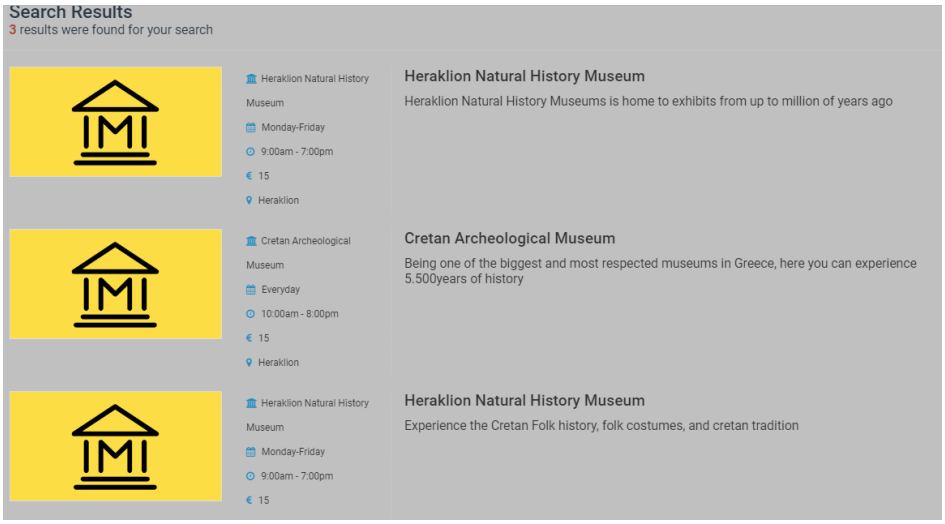
Εικόνα 39

Εμφανίζεται μια λίστα με τους αρχαιολογικούς χώρους της περιοχής που επιλέξαμε.



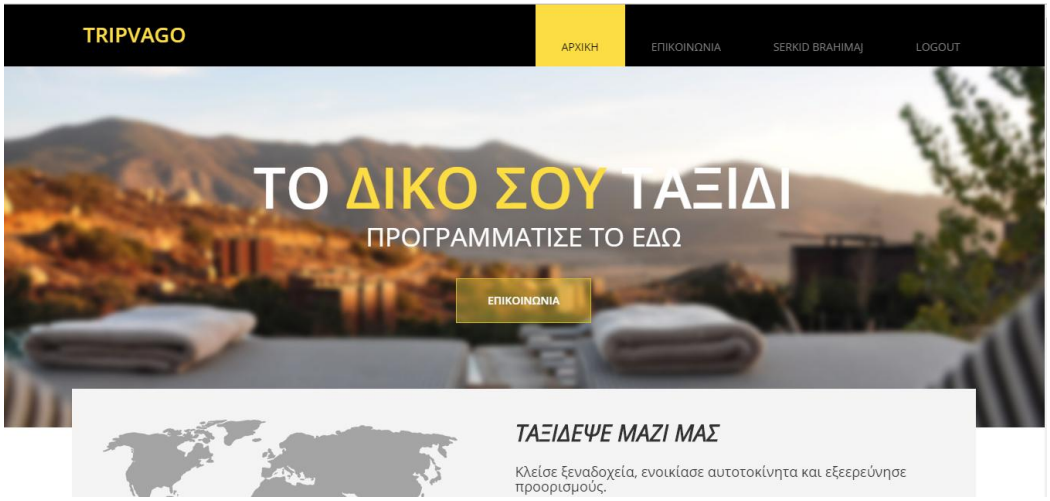
Εικόνα 40

Αντίστοιχα για τα μουσεία εμφανίζεται ένα παράδειγμα τις παρακάτω λίστας.

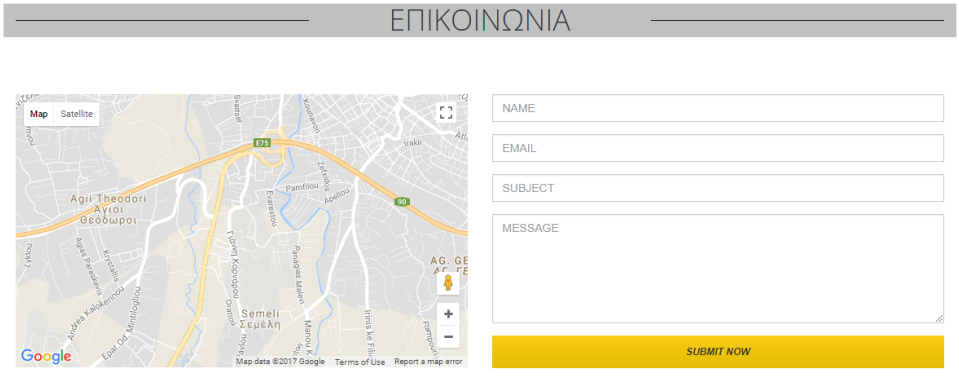


Εικόνα 41

Τέλος επιλέγοντας από το navigation menu την επικοινωνία εμφανίζεται μια στατική σελίδα με στοιχεία επικοινωνίας για την σελίδα μας καθώς και με μια φόρμα για να στέλνετε μηνύματα στον admin.



Εικόνα 42



Εικόνα 43

Κεφάλαιο 5: Αποτελέσματα

Το αποτέλεσμα της πτυχιακής εργασίας είναι ένα λειτουργικό και σταθερό σύστημα κρατήσεων καθώς και ενημέρωσης για τις τουριστικές αποδράσεις ενός ατόμου. Αυτή η πτυχιακή μπορεί να ωφελήσει ένα άτομα να σχεδιάσει τις διακοπές του αλλά και να ενημερωθεί για την ιστορία και τα αξιοθέατα ενός τόπου.

Σε προσωπικό επίπεδο η πτυχιακή αυτή με βοήθησε να αναπτύξω γνώσεις που είχα σε μικρό βαθμό αλλά και να με συστήσει σε νέες τεχνολογίες που είναι αυτή την στιγμή σε μεγάλη ζήτηση στην αγορά εργασίας.

Σε δεύτερο βαθμό αυτή η πτυχιακή μου έμαθε να σχεδιάζω σωστά μια web εφαρμογή τόσο σε γραφικό επίπεδο όσο και σε προγραμματιστικό.

Βιβλιογραφία – Πηγές

1. Ι.Χ. Παναγιωτόπουλος «Εφαρμογές Διαδικτυακού Προγραμματισμού με Java»
2. Java EE Documentation
3. Spring Documentation
4. Mak, Gary "*Spring Recipes: A Problem-Solution Approach"*
5. Bootstrap Documentation
6. JQuery Documentation
7. PostgreSql Documentation