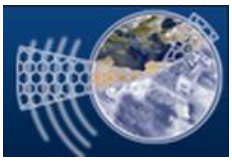
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Τίτλος:

Ολοκληρωμένο σύστημα για την παροχή τουριστικών υπηρεσιών με Τεχνολογίες JAVA

Αντώνης Παπαδάκης (ΑΜ:3270)

Επιβλέπων καθηγητής: Παπαδάκης Νικόλαος

Ημερομηνία παρουσίασης: 07-04-2017

Ευχαριστίες

Ευχαριστώ τον καθηγητή Παπαδάκη Νικόλαο για την προσφορά και τη βοήθεια του.

Abstract

This project is based on a system that guides and provides information about vacation and entertainment for a specific place. The system consists of two sub-systems, a back-end and a front-end. The back-end system is only for administration purposes. The administrator imports all the necessary information about hotels, rental offices, flights, ship routes, museums, landmarks and excursions and submit it to the database. After all information is inserted it will be visible in a dynamic way on the front-end system. The user will be able to search and make bookings. It is noteworthy the fact that the front-end system is built to be mobile friendly using the twitter-bootstrap framework.

Σύνοψη

Το project αυτό βασίζεται σε ένα σύστημα το οποίο καθοδηγεί και παρέχει πληροφορίες για διακοπές και διασκέδαση σε ένα συγκεκριμένο μέρος. Το σύστημα αποτελείται από δύο υποσυστήματα, το back-end και το front-end. Το back-end είναι μόνο για λόγους διαχείρισης. Ο διαχειριστής εισάγει όλες τις απαραίτητες πληροφορίες για ξενοδοχεία, ταξιδιωτικά γραφεία, πτήσεις, δρομολόγια πλοίων, μουσεία, αξιοθέατα και τις καταχωρεί στη βάση δεδομένων. Όταν καταχωρηθούν όλες οι πληροφορίες θα είναι πλέον διαθέσιμες με δυναμικό τρόπο στο front-end σύστημα. Ό χρήστης θα είναι σε θέση να αναζητήσει και να κάνει κρατήσεις. Είναι αξιοσημείωτο το γεγονός ότι το front-end σύστημα είναι υλοποιημένο με το framework twitter-bootstrap για είναι “mobile friendly”.

Πίνακας περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

1.1 Περίληψη

Σκοπός της πτυχιακής αυτής ήταν η μελέτη, η ανάλυση, η σχεδίαση και η υλοποίηση ενός τουριστικού οδηγού για ένα συγκεκριμένο μέρος. Θα υλοποιηθεί ένα ολοκληρωμένο σύστημα το οποίο θα αποτελείται από δύο υποσυστήματα, το back-end το οποίο αφορά μόνο τον διαχειριστή και το front-end το οποίο αφορά τον τελικό χρήστη. Στο back-end γίνεται εισαγωγή όλων των απαραίτητων πληροφοριών από τον διαχειριστή και καταχωρούνται στη βάση δεδομένων ενώ στο front-end όλες αυτές οι πληροφορίες θα παρουσιάζονται με δυναμικό τρόπο στο χρήστη ανάλογα με το τι ζητάει.

Το σύστημα αυτό θα παρέχει πληροφορίες για ξενοδοχεία, γραφεία ενοικιάσεων μεταφορικών μέσων, μουσεία, αξιοθέατα, δρομολόγια πτήσεων και πλοίων καθώς και για εκδρομές. Ο χρήστης θα είναι σε θέση να κάνει κρατήσεις και ακυρώσεις αυτών μέσω του διαδικτύου.

1.2 Κίνητρα

Τα κίνητρα για την εκπόνηση της πτυχιακής αυτής είναι εκπαιδευτικά καθώς θα βοηθήσει στην καλύτερη κατανόηση και πρακτική εφαρμογή των τεχνολογιών που χρησιμοποιήθηκαν.

1.3 Σκοπός και Στόχοι Εργασίας

Σκοπός της εργασίας είναι η υλοποίηση ενός συστήματος παροχής πληροφοριών και καθοδήγησης ενός συγκεκριμένου μέρους το οποίο θα αποτελείται από δύο υποσυστήματα, ένα για τον διαχειριστή και ένα για τον πελάτη. Οι στόχοι αφορούν την ευχρηστία του συστήματος και την εύκολη διαχείριση του. Όσον αφορά το κομμάτι του χρήστη θα πρέπει να είναι ξεκάθαρη η λειτουργία του και να μην τον μπερδεύει για το πώς θα κάνει κρατήσεις και ακυρώσεις αυτών. Επίσης το front-end υποσύστημα θα είναι φιλικό προς κινητά καθώς τα τελευταία χρόνια σημειώνεται ραγδαία εξέλιξη και χρήση αυτών στην καθημερινότητα μας.

1.4 Δομή Εργασίας

Το πρώτο κεφάλαιο περιλαμβάνει την εισαγωγή στην εργασία. Το δεύτερο κεφάλαιο αφορά τη μεθοδολογία υλοποίησης και τις τεχνικές. Το τρίτο κεφάλαιο αφορά την ανάλυση απαιτήσεων του συστήματος, το σχεδιασμό υλοποίησης καθώς και την υλοποίηση. Τέλος το τέταρτο κεφάλαιο αφορά τις μελλοντικές επεκτάσεις της εργασίας.

2 Μεθοδολογία Υλοποίησης

2.1 Μεθοδολογία

Το σύστημα που καλούμαστε να υλοποιήσουμε θα αποτελείται από δύο υποσυστήματα / επίπεδα, το front-end το οποίο αφορά την παρουσίαση των πληροφοριών σε ένα φιλικό και εύκολο περιβάλλον στον τελικό χρήστη και το back-end το οποίο αφορά την καταχώρηση και διαχείριση των πληροφοριών αυτών από τον διαχειριστή καθώς και την λογική της διαχείρισης τους.

Υποσύστημα	**Βιβλιογραφία**
Front-end	https://en.wikipedia.org/wiki/Front_and_back_ends
Back-end	https://en.wikipedia.org/wiki/Front_and_back_ends

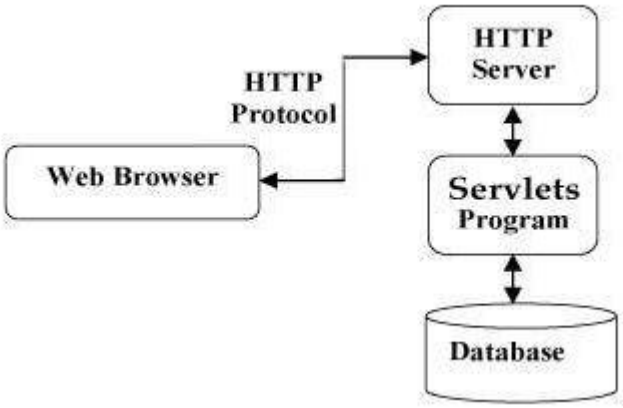
Πίνακας 1 - Υποσυστήματα

2.2 Τεχνολογίες

Το back-end είναι βασισμένο στις τεχνολογίες Java Servlets και JSP (JavaServer Pages) και η βάση δεδομένων που χρησιμοποιήθηκε είναι η Oracle sql.

2.2.1 Servlets

Τα servlets (μικροϋπηρεσίες) είναι προγράμματα τα οποία εκτελούνται σε ένα server (διακομιστή) και λειτουργούν ως ενδιάμεσο επίπεδο μεταξύ μιας αίτησης που προέρχεται από ένα web browser ή κάποιο πελάτη HTTP και μιας βάσης δεδομένων η μιας εφαρμογής στο server. [1]



Εικόνα 1 - Servlets

- ## Η δουλειά ενός servlet

1. **Ανάγνωση των ρητών δεδομένων που έχουν σταλεί από τον πελάτη**

   Συνήθως από μια φόρμα html.

2. **Ανάγνωση των έμμεσων δεδομένων από αιτήσεις HTTP που έχουν σταλεί από τον browser**
   Δηλαδή πληροφορίες HTTP που υπάρχουν στο παρασκήνιο όπως είναι τα μπισκότα (cookies), πληροφορίες για τα μέσα και τις μεθόδους συμπίεσης του browser κ.α.

3. **Παραγωγή των αποτελεσμάτων**

   Τα αποτελέσματα μπορούν να παραχθούν με σύνδεση σε μια βάση δεδομένων, με κλήση RMI / EJB, με την ενεργοποίηση κάποιας υπηρεσίας ιστού ή να παραχθούν άμεσα. Συνήθως ο browser δε μπορεί να επικοινωνήσει άμεσα με την βάση δεδομένων γι' αυτό το λόγο υπάρχει το servlet το οποίο λειτουργεί ως ενδιάμεσο επίπεδο για να εξάγει τα δεδομένα από τη βάση και να τα ενσωματώσει σε κάποιο έγγραφο.

4. **Αποστολή του εγγράφου στον πελάτη**

   Συνήθως το έγγραφο αυτό έχει μορφή HTML αλλά μπορεί να έχει κι άλλες μορφές όπως XML, ZIP, δυαδική (εικόνες).

5. **Αποστολή έμμεσων δεδομένων απάντησης HTTP**

   Είναι οι πληροφορίες HTTP που δεν είναι ορατές στον πελάτη. Γίνεται ενημέρωση στον browser για τον τύπο του εγγράφου που πρόκειται να λάβει, τον ορισμό των παραμέτρων για τα μπισκότα και την κρυφή μνήμη κ.α.

- ## Ο κύκλος ζωής ενός servlet

1. **Η μέθοδος init**

   Η μέθοδος αυτή καλείται στην αρχή της δημιουργίας ενός servlet και όχι κάθε φορά που κάνει μια αίτηση ο πελάτης. Έτσι ο κώδικας θα εκτελεστεί μόνο μια φορά Τη χρησιμοποιούμε για την απόδοση αρχικών τιμών εφόσον είναι απαραίτητο στο πρόγραμμα μας. Στη δική μας περίπτωση δεν τη χρησιμοποιούμε.

Παράδειγμα

```
private long modTime;
private int numbers = new int[10];

public void init() throws ServletException (
    modTime = System.currentTimeMillis()/1000*1000;

    for(int i = 0; i < numbers.length(); i++) {
       numbers[i] = randomNum();
    }
)

.....
```

2. Η μέθοδος service

Κάθε φορά που ο server λαμβάνει μια αίτηση για κάποια μικροϋπηρεσία ξεκινά ένα νέο νήμα και καλεί τη μέθοδο service. Η μέθοδος αυτή ελέγχει τον τύπο της αίτησης HTTP (get, post, delete κτλ.) και ανάλογα καλεί την doGet, doPost, doDelete κτλ. Οι πλέον συνηθισμένες μέθοδοι είναι οι doGet και doPost. Σε κάποιες περιπτώσεις χρειάζεται να χειριστούμε και τις δυο αυτές μεθόδους με τον ίδιο τρόπο. Τότε φτιάχνουμε το πρόγραμμα μας έτσι ώστε να καλεί η μια μέθοδος την άλλη.

Παράδειγμα

```
public void doGet (HttpServletResponse request,
HttpServletResponse response) throws        ServletException,
IOException (
                    .....
)

public void doPost (HttpServletResponse request,
HttpServletResponse response) throws        ServletException,
IOException (

         doGet(request , response);
)
```

3. Η μέθοδος destroy

Ο server μπορεί να αποφασίσει να κλείσει μια εφαρμογή servlet, δηλαδή να αφαιρέσει το στιγμιότυπο του από τη μνήμη είτε επειδή το ζήτησε ο διαχειριστής είτε επειδή η εφαρμογή ‘ήταν αδρανής για μεγάλο χρονικό διάστημα. Πριν γίνει αυτό όμως καλεί τη μέθοδο destroy για να κλείσει συνδέσεις με βάσεις δεδομένων, να σταματήσει νήματα που λειτουργούν στο παρασκήνιο, να γράψει στο δίσκο λίστες μπισκότων, μετρήσεις επισκέψεων ή να εκτελέσει εργασίες καθαρισμού.

- Αποσφαλμάτωση μικροϋπηρεσιών

1. **Με χρήση εντολών εκτύπωσης**

   Μπορούμε να χρησιμοποιήσουμε εντολές `System.out.println();` για να εμφανίσουμε στο παράθυρο εξόδου του διακομιστή μας μεταβλητές ή σταθερές και να ελέγξουμε την τιμή που έχουν σε περίπτωση που παίρνουμε λανθασμένα αποτελέσματα. Επίσης σε περιπτώσεις σφάλματος (exception) τοποθετούμε μερικές τέτοιες εντολές στον κώδικα μας για να δούμε σε ποια γραμμή υπάρχει το σφάλμα.

2. **Χρήση ολοκληρωμένου προγράμματος αποσφαλμάτωσης**

   Πολλά ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE) διαθέτουν εξειδικευμένα εργαλεία αποσφαλμάτωσης, στα οποία μπορούμε να ελέγξουμε συγκεκριμένες μεθόδους της εφαρμογής να κάνουμε διακοπή σε συγκεκριμένα σημεία κα. Επίσης παρέχουν τη δυνατότητα απομακρυσμένης σύνδεσης με κάποιο διακομιστή.

3. **Χρήση αρχείου καταγραφής**

   Η κλάση *HttpServlet* παρέχει τη μέθοδο *log* η οποία μας επιτρέπει να γράφουμε πληροφορίες του προγράμματος σε ένα αρχείο καταγραφής (log file). Η μέθοδος αυτή μπορεί να δέχεται ένα αντικείμενο String ή ένα αντικείμενο String και ένα Throwable.

4. **Apache Log4J**

   Είναι ένα πακέτο από Apache Jakarta Project το οποίο μας βοηθάει να εισάγουμε στον κώδικα μας "ημιμόνιμες" εντολές και με τη χρήση ενός αρχείου XML μπορούμε να ρυθμίσουμε ποιες από τις εντολές αυτές θα κληθούν κατά την αίτηση.

5. **Ξεχωριστές κλάσεις**

   Έχοντας ένα κομμάτι κώδικα το οποίο εκτελείται συνεχώς είναι μια καλή αρχή να το γράψουμε σε μια ξεχωριστή κλάση για να μην το γράφουμε συνεχώς. Αυτό μας βοηθάει και στον έλεγχο αυτής της κλάσης ανεξάρτητα από τον server.

6. **Ελλιπή ή λανθασμένα δεδομένα**

   Ένας άλλος τρόπος αποσφαλμάτωσης είναι να κάνουμε έλεγχο των δεδομένων που διαβάζουμε από τον πελάτη, δηλαδή αν τα δεδομένα έχουν τιμή null ή είναι ένα κενό αλφαριθμητικό.

7. **Εξέταση του κώδικα HTML**

   Με την επιλογή "view page source" που μας παρέχει ο browser μπορούμε να δούμε τυχόν λάθη στον πηγαίο κώδικα μας, για παράδειγμα να έχουμε γράψει `<table>` αντί για `</table>`.

8. Ξεχωριστή εξέταση τω δεδομένων αιτήσεων

Η κλάση *EchoServer* μας δίνει τη δυνατότητα να υποβάλουμε φόρμες HTML και να παίρνουμε ως αποτέλεσμα το πώς ακριβώς έφθασαν τα δεδομένα στο διακομιστή. Είναι ένας απλός διακομιστής ο οποίος κατασκευάζει ένα μια σελίδα HTML για όλες τις αιτήσεις στην οποία φαίνεται τι στάλθηκε.

9. Ξεχωριστή εξέταση των δεδομένων απαντήσεων

Με την κλάση *WebClient* μπορούμε να συνδεθούμε αλληλεπιδραστικά με το διακομιστή να στείλουμε προσαρμοσμένα HTTP δεδομένα και να δούμε τι επιστρέφει.

10. Τερματισμός και επανεκκίνηση του server

Ο server διατηρεί τα αρχεία servlet στη μνήμη μεταξύ των αιτήσεων για να μην τα φορτώνει σε κάθε εκτέλεση τους. Μερικοί servers ξαναφορτώνουν αυτόματα κάποια αρχεία servlets όταν αλλάζει το αρχείο των κλάσεων τους. Σε κάποιες περιπτώσεις οι servers μπορεί να "μπερδευτούν" κι έτσι η συμπεριφορά ενός servlet να μην είναι η αναμενόμενη. Οπότε είναι μια καλή τεχνική η επανεκκίνηση του server.

Παράδειγμα:

Παρακάτω βλέπουμε ένα πρόγραμμα στο οποίο περνάμε δύο τιμές με την μέθοδο GET μέσω του url

http://localhost:8080/HelloForm?first_name=Antonis&last_name=Papadakis

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloForm extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
```

```
        // Set response content type
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        String title = "Using GET Method to Read Form Data";

        String docType = "<!doctype html">\n";

        out.println(docType + "<html>\n" +
           "<head><title>" + title + "</title></head>\n" +
           "<body bgcolor=\"#f0f0f0\">\n" +
           "<h1 align=\"center\">" + title + "</h1>\n" +
           "<ul>\n" +
           "  <li><b>First Name</b>: "
           + request.getParameter("first_name") + "\n" +
           "  <li><b>Last Name</b>: "
           + request.getParameter("last_name") + "\n" +
           "</ul>\n" +
           "</body></html>");
    }
}
```

Αποτέλεσμα:

Using GET Method to Read Form Data

First Name: Antonis

Last Name: Papadakis

Παράδειγμα:

Στο παρακάτω πρόγραμμα χειριζόμαστε δεδομένα που έχουν εισαχθεί μέσω HTML φόρμας με τη μέθοδο POST.

Φόρμα

```
<html>
   <body>
      <form action="HelloForm" method="POST">
         First Name: <input type="text" name="first_name">
         <br />
         Last Name: <input type="text" name="last_name" />
         <input type="submit" value="Submit" />
      </form>
   </body>
</html>
```

Servlet

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloForm extends HttpServlet {

   // Method to handle GET method request.
   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      // Set response content type
      response.setContentType("text/html");

      PrintWriter out = response.getWriter();
      String title = "Using GET Method to Read Form Data";
```

```
      String docType = "<!doctype html">\n";

      out.println(docType + "<html>\n" +
         "<head><title>" + title + "</title></head>\n" +
         "<body bgcolor=\"#f0f0f0\">\n" +
         "<h1 align=\"center\">" + title + "</h1>\n" +
         "<ul>\n" +
         "  <li><b>First Name</b>: "
         + request.getParameter("first_name") + "\n" +
         "  <li><b>Last Name</b>: "
         + request.getParameter("last_name") + "\n" +
         "</ul>\n" +
         "</body></html>");
   }

   // Method to handle POST method request.
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      doGet(request, response);
   }
}
```

2.2.2 JavaServer Pages (JSP)

Η τεχνολογία JSP μας δίνει τη δυνατότητα να αναμείξουμε κώδικα HTML με δυναμικό περιεχόμενο. Μπορούμε να θεωρήσουμε τις σελίδες JSP ως σελίδες HTML με κώδικα Java στο εσωτερικό τους. Επίσης μπορούμε να παραδεχτούμε ότι η συγγραφή σελίδων JSP είναι ένας άλλος τρόπος γραφής μικροϋπηρεσιών καθώς μεταφράζονται σε μικροϋπηρεσίες, μεταγλωττίζονται και εκτελούνται κατά την αίτηση. Ο κώδικας με το δυναμικό περιεχόμενο περικλείεται σε ειδικές ετικέτες και συνήθως είναι οι <% ……%>. [1]

Παράδειγμα

```
<html>
<head><title>Hello World</title></head>
<body>
Hello World!<br/>
<%
out.println("Your IP address is " + request.getRemoteAddr());
%>
</body>
</html>
```

Αποτέλεσμα

Hello World!

Your IP address is 127.0.0.1

- ## Ανάγκη και οφέλη της τεχνολογίας JSP

  Τα servlets είναι πολύ καλά όσον αφορά την επεξεργασία και τον προγραμματισμό δεδομένων όμως δεν είναι τόσο καλά στην παρουσίαση τους.

  - Είναι δύσκολη η συγγραφή και η συντήρηση του κώδικα HTML διότι εισάγεται μέσα σε παρενθέσεις με ανάποδους καθέτους, με ελληνικά ερωτηματικά κτλ. Μερικές φορές είναι δύσκολο να παρατηρήσει κανείς ότι το αποτέλεσμα είναι κώδικας HTML. Στις σελίδες JSP ο στατικός κώδικας είναι κανονικός κώδικας HTML.
  - Δε μπορούμε να χρησιμοποιήσουμε εργαλεία κατασκευής ιστοσελίδων όπως είναι το Dreamweaver, με την τεχνολογία JSP μπορούμε συν το ότι δε λαμβάνονται υπόψιν οι ετικέτες JSP.
  - Για κάποιον που δε γνωρίζει JAVA είναι απροσπέλαστη η HTML με την έννοια ότι θα δυσκολευτεί να εξετάσει και να τροποποιήσει τον κώδικα.
  - Με τη χρήση της τεχνολογίας JSP μπορεί να διαιρεθεί η ομάδα ανάπτυξης δηλαδή οι προγραμματιστές JAVA μπορούν να ασχοληθούν με το δυναμικό κώδικα και οι προγραμματιστές ιστού με τον στατικό κώδικα.

- ### JSP vs ASP

  Τα πλεονεκτήματα της JSP είναι δύο. Πρώτον είναι platform independent δηλαδή φορητή σε πολλά λειτουργικά συστήματα και όχι μόνο για Windows , έτσι δεν ήμαστε αναγκασμένοι να γράψουμε ένα πρόγραμμα για ένα μόνο λειτουργικό. Δεύτερον το δυναμικό μέρος της εφαρμογής γράφετε σε JAVA και όχι σε Visual Basic η κάποια άλλη ειδική γλώσσα της Microsoft. Επιπλέον μια βασική διαφορά είναι ότι τα προγράμματα ASP ερμηνεύονται κάθε φορά που γίνεται προσπέλαση σε αυτά ενώ τα JSP ερμηνεύονται μόνο την πρώτη φορά που γίνεται προσπέλαση, αυτό σημαίνει ότι η εκτέλεση τους είναι ταχύτερη.

- ### JSP vs PHP

  Ένα πλεονέκτημα της JSP σε σχέση με την PHP είναι ότι ο κώδικας γράφεται σε JAVA η οποία διαθέτει ήδη μια εκτενή API για δίκτυα, προσπέλαση βάσεων δεδομένων, κατανεμημένα αντικείμενα. Επίσης η JSP διαθέτει μεγαλύτερη υποστήριξη από τους κατασκευαστές εργαλείων και διακομιστών.

Παράδειγμα:

Στο παρακάτω πρόγραμμα χειριζόμαστε δεδομένα που έχουν εισαχθεί μέσω HTML φόρμας με τη μέθοδο POST.

Φόρμα

```
<html>
<body>
<form action="main.jsp" method="POST">
First Name: <input type="text" name="first_name">
<br />
Last Name: <input type="text" name="last_name" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

JSP

```
<html>
<head>
<title>Using GET and POST Method to Read Form Data</title>
</head>
<body>
<center>
<h1>Using GET Method to Read Form Data</h1>
<ul>
<li><p><b>First Name:</b>
   <%= request.getParameter("first_name")%>
</p></li>
<li><p><b>Last  Name:</b>
   <%= request.getParameter("last_name")%>
</p></li>
</ul>
</body>
</html>
```

Αποτέλεσμα:

Using GET Method to Read Form Data

First Name: Antonis

Last Name: Papadakis

- Δηλώσεις JSP

  <%! *δήλωση* %>

  Πχ <%! `int i = 1` %>

- JSP Expressions

  <%= *expression* %>

  Πχ <%= `new java.util.Date().toLocaleString()` %>

- Σχόλια JSP

  <%-- *σχόλιο* --%>

  Πχ <%-- this is a comment --%>

- Οδηγίες JSP

  <%@ page.. %>

  Ορίζει ιδιότητες σελίδας όπως scripts , σελίδες σφάλματος, απαιτήσεις buffering.

  <%@ include … %>

  Εισαγωγή αρχείου κατά τη διάρκεια της μετάφρασης.

  <%@ taglib … %>

  Δήλωση μιας βιβλιοθήκης ετικέτας.

2.2.3 JDBC

Επισήμως δεν είναι ακρωνύμιο αλλά ανεπίσημα σημαίνει Java DataBase Connectivity. Η JDBC παρέχει μια βιβλιοθήκη για την προσπέλαση σχεσιακών βάσεων δεδομένων. Χρησιμοποιώντας την API της μπορούμε μια μεγάλη ποικιλία βάσεων δεδομένων με την ίδια σύνταξη Java. [1]

- Βήματα εκτέλεσης ερωτημάτων με την JDBC

1) **Φόρτωση του προγράμματος οδήγησης**
Καθορίζουμε το όνομα κλάσης της βάσεως δεδομένων με τη μέθοδο *Class.forName*. αυτή η κλάση μπορεί να μεταβιβάσει μια εξαίρεση *ClassNotFoundException* και γι' αυτό το λόγο θα πρέπει να βρίσκεται μέσα σε ένα μπλοκ try/catch.

```
Try {
      Class.forName("oracle.jdbc.driver.OracleDriver");
} catch (ClassNotFoundException e) {
      System.err.println("Error: " + e);
}
```

2) **Ορισμός url για τη σύνδεση**
Καθορίζουμε τον υπολογιστή υπηρεσίας του server τη θύρα και το όνομα της βάσης.

```
String host = "localhost";
String dbName = "orcl";
int port = 1521;

String url = "jdbc.oracle:thin:@" + host +":" + port +
":" + dbName;
```

3) **Δημιουργία της σύνδεσης**
Εισάγουμε το url σύνδεσης, το όνομα χρήστη της βάσης δεδομένων και τον κωδικό στην κλάση *getConnection* της κλάσης *DriverManager*. Μπορούμε να εκτελούμε ερωτήματα μέχρι να κλείσει η σύνδεση.

```
String username = "name"';
String password = "1111";

Connection conn =
DriverManager.getConnection(url,username,password)
```

4) **Δημιουργία ενός Statement**
Δημιουργείται από το αντικείμενο *Connection* με τη μέθοδο *getConnection* και χρησιμοποιείται για την αποστολή ερωτημάτων στη βάση δεδομένων.

```
Statement stm = conn.createStatement();
```

5) **Εκτέλεση ερωτήματος η ενημέρωσης**
Οι πιο συχνές μέθοδοι: *executeQuery*, *executeUpdate*.

   - **executeQuery**: Εκτελεί ένα ερώτημα sql κι επιστρέφει ένα αντικείμενο ResultSet το οποίο δεν είναι ποτέ null.

- **executeUpdate:** Χρησιμοποιείται για τις εντολές Update, Insert και Delete κι επιστρέφει τον αριθμό των γραμμών που επηρεάζονται οποίος μπορεί να είναι και μηδέν.

6) Επεξεργασία των αποτελεσμάτων

Όταν εκτελείται ένα ερώτημα επιστρέφεται ένα αντικείμενο ResultSet το οποίο αντιπροσωπεύει ένα σύνολο γραμμών και στηλών το οποίο μπορούμε να επεξεργαστούμε με κλήσεις μεθόδων *next* και *getXXX*. Για παράδειγμα για τιμή String έχουμε τη μέθοδο *getString* και για τιμές int έχουμε τη μέθοδο *getInt*. Η πρώτη στήλη σε μια γραμμή ενός ResultSet έχει αριθμοδείκτη 1 και όχι 0 όπως συμβαίνει με τους πίνακες.

```
While(rs.next()){
      System.out.println(resultSet.getString(1)+ "" +
      resultSet.getInt(2));
}
```

7) Κλείσιμο της σύνδεσης

Με την ολοκλήρωση των ερωτημάτων κλείνουμε τη σύνδεση για αποδέσμευση των πόρων του συστήματος χρησιμοποιώντας την εντολή:

```
connection.close();
```

Παράδειγμα

```
//STEP 1. Import required packages
import java.sql.*;

public class FirstExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";
```

```
public static void main(String[] args) {
Connection conn = null;
Statement stmt = null;
try{
   //STEP 2: Register JDBC driver
   Class.forName("com.mysql.jdbc.Driver");

   //STEP 3: Open a connection
   System.out.println("Connecting to database...");
   conn = DriverManager.getConnection(DB_URL,USER,PASS);

   //STEP 4: Execute a query
   System.out.println("Creating statement...");
   stmt = conn.createStatement();
   String sql;
   sql = "SELECT id, first, last, age FROM Employees";
   ResultSet rs = stmt.executeQuery(sql);

   //STEP 5: Extract data from result set
   while(rs.next()){
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);
   }
   //STEP 6: Clean-up environment
   rs.close();
   stmt.close();
```

```
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end FirstExample
```

- *Prepared Statements*

Τα prepared statements (προκατασκευασμένες εντολές) μειώνουν τον χρόνο εκτέλεσης όταν θέλουμε να εκτελέσουμε ένα αντικείμενο Statement πολλές φορές. Ένα μεγάλο συν αυτών είναι το ότι τους δίνουμε μια sql εντολή τη στιγμή που δημιουργούνται. Αυτό σημαίνει ότι θα σταλεί στο σύστημα διαχείρισης της βάσης δεδομένων και θα μεταγλωττιστεί, άρα τελικά το αντικείμενο PreparedStatement αποτελείται από μια εντολή sql η οποία έχει προ-μεταγλωττιστεί κι έτσι το σύστημα θα εκτελέσει την εντολή χωρίς να τη μεταγλωττίσει.

Παράδειγμα

```
Connection conn = DriverManager.getConnection(url,username,password);
String query = "select * from Customers where age=? and city=?";
PreparedStatement pst = conn.prepareStatement(query);
pst.setInt(25);
pst.setString(2 , "Athens");
pst.execute();
```

2.2.4 Oracle sql

Η oracle είναι μια σχεσιακή βάση δεδομένων δηλαδή στηρίζεται σε σχέσεις (relations) που δηλώνονται με βάση τα κοινά πεδία διαφορετικών πινάκων. Αυτές τις σχέσεις, δηλαδή τα κοινά πεδία μπορούμε να τα δηλώσουμε είτε πριν είτε μετά τη δημιουργία των αρχείων.

- Εντολές της Oracle sql

  - **Εντολές ερωτημάτων (queries)**

    select **,** για εμφάνιση πληροφοριών

    Παράδειγμα: Εμφάνιση όλων των δεδομένων του πίνακα Customers

    ```
    select * from Customers;
    ```

  - **Εντολές αλλαγής στοιχείων**

    insert , για εισαγωγή νέας εγγραφής

    Παράδειγμα: Εισαγωγή δεδομένων στον πίνακα

    ```
    insert into Customers values
    (10,'Antonis','Papadakis','Heraklion');
    ```

    update , για τροποποίηση εγγραφών

    Παράδειγμα ενημέρωση της διεύθυνσης του πελάτη

    ```
    update table Customers set Address='Mires' where id=10;
    ```

delete , για διαγραφή εγγραφών

Παράδειγμα: Διαγραφή του πελάτη με id 10

```
delete from Customers where id=10;
```

- **Εντολές δημιουργίας αντικειμένων**

create table , για δημιουργία νέου πίνακα

Παράδειγμα:

```
CREATE TABLE Hotels(
  Name VARCHAR(25) PRIMARY KEY NOT NULL,
  Category VARCHAR(15) NOT NULL,
  Address VARCHAR(35) NOT NULL,
  Phone VARCHAR(20) NOT NULL,
  Email VARCHAR(30) NOT NULL
);
```

create view , για δημιουργία νέας όψης

Παράδειγμα:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = No;
```

alter table , για τροποποίηση ενός πίνακα

Παράδειγμα: Αλλαγή του τύπου του πεδίου id σε ακέραιο

```
Alter table Customers modify id int;
```

- **Εντολές ελέγχου ασφαλείας της βάσης**

grand , παρέχει πρόσβαση στα δεδομένα της βάσης

revoke , αφαιρεί την πρόσβαση στα δεδομένα της βάσης

commit , ολοκλήρωση της συναλλαγής και αποθήκευση των δεδομένων στη βάση

rollback , αναίρεση / επαναφορά των δεδομένων

- Χαρακτηριστικά της Oracle

  - **Object types**
    Αντικείμενα ορισμένα από το χρήστη (user - defined data types UDT) ή αφηρημένοι τύποι (abstract types ADT)

    Τέτοια αντικείμενα μπορεί να είναι σύνθετοι τύποι δεδομένων. Παράδειγμα:

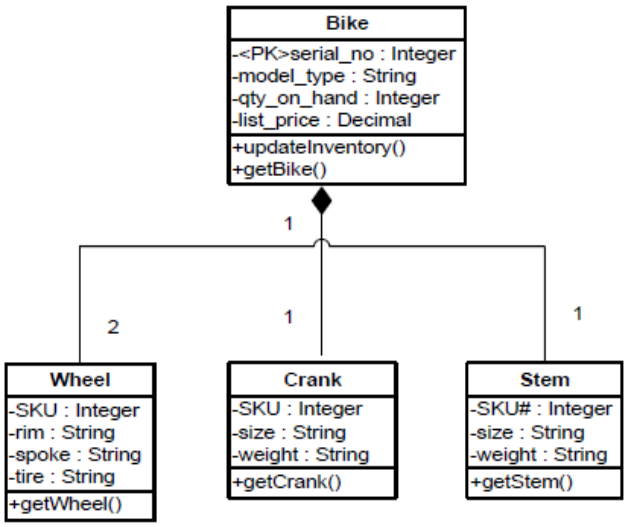
```
CREATE TYPE address_t  AS OBJECT ( street varchar(255),
      city     varchar(30),
      state  varchar(30),
      zip     varchar(5)
);

CREATE TYPE geopoint_t AS OBJECT (
      x double,
      y double
);

CREATE TYPE geoaddres_t AS OBJECT (
      address     address_t,
      coordinate varchar(30),
      geopoint  geopoint_t
);
```

  - **Μέθοδοι**
    Για κάθε object type ο χρήστης μπορεί να ορίσει μεθόδους για πρόσβαση

  - **Varray**
    Ο τύπος δεδομένων varray είναι μια ”συλλογή” που επιτρέπει την ομαδοποίηση ομοιογενών δεδομένων σε πίνακα (1 × ν) που συνιστά αντικείμενο.

  - **Φωλευμένος πίνακας (Nested table)**
    Ο τύπος δεδομένων nested table είναι μια ‘’συλλογή’’ που επιτρέπει την ομαδοποίηση ομοιογενών δεδομένων σε πίνακα (μ x ν) που αποτελεί τμήμα άλλου πίνακα.

    Παράδειγμα:



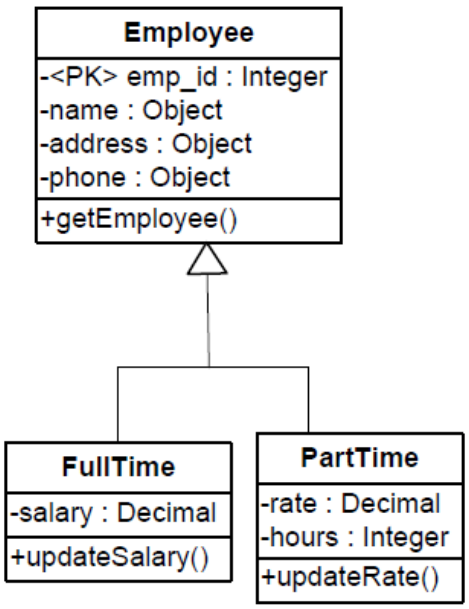
Εικόνα 2 - Nested table

```
CREATE TYPE wheel_type AS OBJECT (
      SKU VARCHAR2(15),
      rim VARCHAR2(30),
      spoke VARCHAR2(30),
      tire VARCHAR2(30));
      CREATE TYPE crank_type AS OBJECT (
      SKU VARCHAR2(15),
      crank_size VARCHAR2(15),
      crank_weight VARCHAR2(15) );
      CREATE TYPE stem_type AS OBJECT (
      SKU VARCHAR2(15),
      stem_size VARCHAR2(15),
      stem_weight VARCHAR2(15)
);
CREATE TYPE nested_table_wheel_type   AS TABLE OF wheel_type;
CREATE TYPE nested_table_crank_type   AS TABLE OF crank_type;
CREATE TYPE nested_table_stem_type    AS TABLE OF stem_type;
```

```
CREATE TABLE bike (
      serial_no INTEGER PRIMARY KEY,
      model_type VARCHAR2(20),
      front_wheel nested_table_wheel_type,
      rear_wheel nested_table_wheel_type,
      crank nested_table_crank_type,
      stem nested_table_stem_type)
      NESTED TABLE front_wheel STORE AS front_wheel,
      NESTED TABLE rear_wheel STORE AS rear_wheel,
      NESTED TABLE crank STORE AS nested_crank,
      NESTED TABLE stem STORE AS nested_stem
);
```

- **Κληρονομικότητα**
  Είναι δυνατή η κληρονομικότητά μεταξύ των τύπων

  Παράδειγμα:



Εικόνα 3 - Inheritance

```
CREATE TYPE employee_ty AS OBJECT (
      emp_id NUMBER,
      name name_ty,
      address address_ty,
      phone varray_phone_ty
)NOT FINAL NOT INSTANTIABLE;

CREATE TYPE full_time_ty UNDER employee_ty (
      Salary NUMBER(8,2)
);

CREATE OR REPLACE TYPE part_time_ty UNDER employee_ty (
      rate Number(7,2),
      hours Number(3)
)NOT FINAL;

CREATE TABLE FullTimeEmp of full_time_ty;
CREATE TABLE PartTimeEmp of part_time_ty
```

- **Object View**
  Επιτρέπει δημιουργία δομών object σε πίνακες

- Oracle vs MySQL

Σύγκριση Schema Objects [2]

Oracle	**MySQL**
AFTER trigger	trigger
BEFORE trigger	trigger
Check constraint	Check constraint
Column default	Column default
Database	Database
Foreign key	Foreign key
Index	Index
Package	N/A
PL/SQL function	Routine
PL/SQL procedure	Routine
Primary key	Primary key
Role	N/A
Schema	Schema
Sequence	AUTO_INCREMENT for a column
Snapshot	N/A
Synonym	N/A
Table	Table
Tablespace	N/A
Temporary table	Temporary table
Trigger for each row	Trigger for each row

Unique key	Unique key
User	User
View	View

Πίνακας 2 - Oracle and MySQL Schema Objects

Σύγκριση προνομίων [2]

Level	**Privilege**	**System Privilege(s) on Oracle**
Global	ALTER	ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY CUSTER, COMMENT ANY TABLE
Global	ALTER ROUTINE	ALTER ANY PROCEDURE, DROP ANY PROCEDURE
Global	CREATE	CREATE ANY TABLE, CREATE ANY SEQUENCE, CREATE ANY CLUSTER, CREATE DATABASE LINK, COMMENT ANY TABLE
Global	CREATE ROUTINE	CREATE ANY PROCEDURE
Global	CREATE USER	CREATE USER, GRANT ANY PRIVILEGE
Global	CREATE VIEW	CREATE ANY VIEW
Global	DELETE	ALTER ANY TABLE, DROP USER, DELETE ANY TABLE
Global	DROP	DROP ANT TABLE, DROP ANY SEQUENCE, DROP ANY CLUSTER, DROP ANY VIEW
Global	EXECUTE	EXECUTE ANY PROCEDURE
Global	INDEX	CREATE ANY INDEX, ALTER ANY INDEX, DROP ANY INDEX
Global	INSERT	INSERT ANY TABLE
Global	LOCK TABLES	LOCK ANY TABLE
Global	SELECT	SELECT ANY TABLE
Global	SUPER	CREATE ANY TRIGGER, DROP ANY TRIGGER
Global	UPDATE	UPDATE ANY TABLE
Global	USAGE	CREATE SESSION, ALTER SESSION, UNLIMITED TABLESPACE
Database	CREATE	CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE TABLE
Database	CREATE ROUTINE	CREATE PROCEDURE
Database	CREATE VIEW	CREATE VIEW
Table	CREATE	CREATE TABLE
Table	CREATE VIEW	CREATE VIEW

Πίνακας 3 - Oracle and MySQL Privileges

Αντιστοιχία τύπων δεδομένων [2]

MySQL Data Type	**Oracle Data Type**
BIGINT	NUMBER(19, 0)
BIT	RAW
BLOB	BLOB, RAW
CHAR	CHAR
DATE	DATE
DATETIME	DATE

DECIMAL	FLOAT (24)
DOUBLE	FLOAT (24)
DOUBLE PRECISION	FLOAT (24)
ENUM	VARCHAR2
FLOAT	FLOAT
INT	NUMBER(10, 0)
INTEGER	NUMBER(10, 0)
LONGBLOB	BLOB, RAW
LONGTEXT	CLOB, RAW
MEDIUMBLOB	BLOB, RAW
MEDIUMINT	NUMBER(7, 0)
MEDIUMTEXT	CLOB, RAW
NUMERIC	NUMBER
REAL	FLOAT (24)
SET	VARCHAR2
SMALLINT	NUMBER(5, 0)
TEXT	VARCHAR2, CLOB
TIME	DATE
TIMESTAMP	DATE
TINYBLOB	RAW
TINYINT	NUMBER(3, 0)
TINYTEXT	VARCHAR2
VARCHAR	VARCHAR2, CLOB
YEAR	NUMBER

Πίνακας 4 - Oracle and MySQL Data Types

Σύγκριση αριθμητικών τύπων [2]

MySQL	**Size**	**Oracle**
BIGINT	8 Bytes	NUMBER (19,0)
BIT	approximately (M+7)/8 Bytes	RAW
DECIMAL(M,D)	M+2 bytes if D > 0, M+1 bytes if D = 0 (D+2, if M < D)	FLOAT(24), BINARY_FLOAT
DOUBLE	8 Bytes	FLOAT(24), BINARY_FLOAT, BINARY_DOUBLE
DOUBLE PRECION	8 Bytes	FLOAT(24), BINARY_DOUBLE
FLOAT(25<=X <=53)	8 Bytes	FLOAT(24), BINARY_FLOAT
FLOAT(X<=24)	4 Bytes	FLOAT, BINARY_FLOAT
INT	4 Bytes	NUMBER (10,0)
INTEGER	4 Bytes	NUMBER (10,0)
MEDIUMINT	3 Bytes	NUMBER (7,0)
NUMERIC	M+2 bytes if D > 0, M+1 bytes if D = 0 (D+2, if M < D)	NUMBER
REAL	8 Bytes	FLOAT(24), BINARY_FLOAT
SMALLINT	2 Bytes	NUMBER(5,0)
TINYINT	1 Byte	NUMBER(3,0)

Πίνακας 5 - Oracle and MySQL Number Types

Σύγκριση τύπων ημερομηνίας και ώρας [2]

MySQL	Size	Oracle
DATE	3 Bytes	DATE
DATETIME	8 Bytes	DATE
TIMESTAMP	4 Bytes	DATE
TIME	3 Bytes	DATE
YEAR	1 Byte	NUMBER

Πίνακας 6 - Oracle and MySQL Date Time

Σύγκριση τύπων text [2]

MySQL	Size	Oracle
BLOB	L + 2 Bytes whereas L<2^16	RAW, BLOB
CHAR(m)	M Bytes, 0<=M<=255	CHAR
ENUM (VALUE1, VALUE2, ...)	1 or 2 Bytes depending on the number of enum. values (65535 values max)	
LONGBLOB	L + 4 Bytes whereas L < 2 ^ 32	RAW, BLOB
LONGTEXT	L + 4 Bytes whereas L < 2 ^ 32	RAW, CLOB
MEDIUMBLOB	L + 3 Bytes whereas L < 2^ 24	RAW, BLOB
MEDIUMTEXT	L + 3 Bytes whereas L < 2^ 24	RAW, CLOB
SET (VALUE1, VALUE2, ...)	1, 2, 3, 4 or 8 Bytes depending on the number of set members (64 members maximum)	
TEXT	L + 2 Bytes whereas L<2^16	VARCHAR2, CLOB
TINYBLOB	L + 1 Bytes whereas L<2 ^8	RAW, BLOB
TINYTEXT	L + 1 Bytes whereas L<2 ^8	VARCHAR2
VARCHAR(m)	L+1 Bytes whereas L<=M and0<=M<=255 before MySQL 5.0.3 (0 <= M <= 65535 in MySQL 5.0.3 and later; effective maximum length is 65,532 bytes)	VARCHAR2, CLOB

Πίνακας 7 - Oracle and MySQL Text Types

2.2.5 HTML

Η HTML (HyperText Markup Language) είναι μια γλώσσα σήμανσης υπερκειμένου και χρησιμοποιείται για κατασκευή ιστοσελίδων. Τα στοιχεία της γλώσσας αποτελούνται από ετικέτες οι οποίες περικλείονται στα σύμβολα < ….. > και λειτουργούν συνήθως ανά ζεύγη δηλαδή την ετικέτα έναρξης και την ετικέτα λήξης. Μέσα σε αυτές τις ετικέτες μπορεί να περιλαμβάνεται κείμενο, εικόνα , πίνακες , βίντεο κτλ. Ο σκοπός ενός browser είναι τα διαβάζει τα έγγραφα HTML και να μεταφράζει τις ετικέτες αυτές στο ανάλογο περιεχόμενο. Δηλαδή να συνθέσει την ιστοσελίδα έτσι ώστε ο τελικός χρήστης να μπορεί αν την διαβάσει. Επίσης η HTML παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων δηλαδή το περιεχόμενο και τη μορφοποίηση του καθορίζοντας σημαντικά στοιχεία του κειμένου όπως παραγράφους , κεφαλίδες, λίστες συνδέσμους κα [3].

- Σήμανση

Η σήμανση της HTML αποτελείται από βασικά συστατικά τα οποία είναι τα στοιχεία, οι τύποι δεδομένων, οι αναφορές χαρακτήρων, οι αναφορές οντοτήτων και η δήλωση του τύπου του εγγράφου [3].

- **Στοιχεία**

Ένα έγγραφο HTML αποτελείτε από στοιχεία HTML τα οποία εν γένει έχουν τρία συστατικά, τις ετικέτες εκκίνησης και τερματισμού, ιδιότητες μέσα στην ετικέτα εκκίνησης και το περιεχόμενο ανάμεσα στις ετικέτες.

Γενική μορφή: <ετικέτα ιδιότητα=”τιμή”> *περιεχόμενο* </ετικέτα>

Παράδειγμα:

Κεφαλίδα του εγγράφου

```
<head>
      <title>Ο τίτλος</title>
</head>
```

Παράγραφος:

```
<p>Παράγραφος 1</p>
```

Σχόλια:

<!- - Σχόλια - ->

Δομική σήμανση: περιγράφει το σκοπό του κειμένου, για παράδειγμα

```
<h2> Επικεφαλίδα </h2>
```

Δηλώνει μια επικεφαλίδα δευτέρου επιπέδου

Σήμανση οπτικής μορφοποίησης: περιγράφει την εμφάνιση του κειμένου, για παράδειγμα

```
<b> έντονο κείμενο </b>
```

Το κείμενο αυτό θα είναι **έντονο**

- **Ιδιότητες**

Τα στοιχεία έχουν ιδιότητες. Οι περισσότερες από αυτές τις ιδιότητες είναι ζεύγη ονομάτων και τιμών και γράφονται μέσα στην ετικέτα εκκίνησης ενός στοιχείου μετά το όνομα του. Η τιμή της ιδιότητας περικλείεται από μονά ή διπλά εισαγωγικά. Μερικά παραδείγματα είναι το id, class , style, title, lang κα.

Μια παράγραφος με id *paragraph*

```
<p id=”paragraph”>This is a paragraph</p>
```

Μια κεφαλίδα με κόκκινο χρώμα

```
<h1 style=”color:red”> Header </h1>
```

- **Αναφορές οντοτήτων χαρακτήρων**

Από την έκδοση 4.0 της HTML και μετά ορίζεται ένα σύνολο οντοτήτων χαρακτήρων κι ένα σύνολο οντοτήτων αριθμών τα οποία επιτρέπουν τη γραφή μοναδικών χαρακτήρων ως σήμανση, αντί να χρησιμοποιήσουμε τους ίδιους τους χαρακτήρες. Ένας χαρακτήρας θεωρείται ίδιος με την αντίστοιχη σήμανση του. Αυτή η δυνατότητα διαφυγής επιτρέπει την ερμηνεία των χαρακτήρων ως κείμενο κι όχι ως σήμανση. Εάν για παράδειγμα θέλαμε να εισάγουμε διπλά εισαγωγικά μέσα στο κείμενο μας θα πρέπει να γράψουμε *"* και όχι σκέτα ". Αν παραλείψουμε να γράψουμε σύμβολα ως οντότητες χαρακτήρων ή αριθμών τότε ο Browser μπορεί να μπερδευτεί και να μην αναγνωρίσει σωστά το κείμενο μας.

Επίσης αυτή η διαφυγή διευκολύνει τη χρήση κάποιων χαρακτήρων που είναι δύσκολο να δακτυλογραφηθούν ή δε συμπεριλαμβάνονται στην κωδικοποίηση του εγγράφου, για παράδειγμα ο χαρακτήρας é ο οποίος μπορεί να γραφεί ως *é*.

- **Τύποι δεδομένων**

Στην HTML υποστηρίζονται τύποι δεδομένων όπως σενάρια εντολών ή stylesheet, τύποι για τιμές ιδιοτήτων, αριθμοί, μονάδες μήκους, γλώσσες, πολυμέσα χρώματα , κωδικοποίηση χαρακτήρων κ.α.

- **Δήλωση τύπου εγγράφου**

Όλα τα HTML αρχεία ξεκινάνε με μια δήλωση τύπου εγγράφου η οποία ορίζει τι είδους έγγραφο είναι αυτό. Αυτή η δήλωση βοηθάει τους browsers να διαβάσουν και να παρουσιάσουν το περιεχόμενο.

Παράδειγμα:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- Μεταφορά της HTML

  Οι δύο πιο συνηθισμένοι τρόποι μεταφοράς αρχείων HTML είναι από έναν εξυπηρετητή μέσω του πρωτοκόλλου HTTP ή μέσω email. Για να μεταφερθεί ένα αρχείο μέσω HTTP χρειάζεται ένας τρόπος αναγνώρισης του τύπου του εγγράφου που μεταφέρεται. Γι' αυτό το λόγο μεταφέρονται μαζί με το έγγραφο και μεταδεδομένα, δηλαδή δεδομένα για τα δεδομένα καθώς επίσης και ο τύπος του εγγράφου. Πολλά προγράμματα αλληλογραφίας παρέχουν τη δυνατότητα μεταφοράς html αρχείων ή τη δυνατότητα στο χρήστη να μορφοποιήσει το κείμενο που στέλνει [3].

2.2.6 HTML 5

Η HTML 5 πρόκειται να είναι επόμενη μεγάλη έκδοση και η εξέλιξη της HTML 4.0 και να αντικαταστήσει την XHTML 1.0 και την DOM Level 2 HTML. Επίσης σκοπεύει να απεξαρτηθεί από ιδιόκτητα plug-ins και εφαρμογές όπως το Adobe Flash, το Microsoft Silverlight , το Apache Pivot και την Sun JavaFX. Η έκδοση αυτή υποστηρίζεται απ' όλους τους μοντέρνους browsers και απ' όλα τα smartphones [4].

- Νέες δυνατότητες [5]

  - Νέα σημασιολογικά στοιχεία όπως header, footer, sections
  - Βελτιωμένες φόρμες
  - Τοπική αποθήκευση χωρίς χρήση plug-in
  - Web sockets
  - Server-Sent Events (SSE) από το server στο browser
  - Καμβάς για σχεδίαση με χρήση της JavaScript
  - Ενσωμάτωση ήχου και βίντεο χωρίς χρήση plug-in
  - Τοποθεσία
  - Μικροστοιχεία που μας επιτρέπουν να δημιουργήσουμε δικά μας λεξιλόγια
  - Drag and Drop μέσα στην ίδια σελίδα

- Συμβατότητα

  Η HTML 5 έχει σχεδιαστεί έτσι ώστε να είναι backward compatible με τους browsers, δηλαδή τα αρχεία να υποστηρίζονται παρέχοντας εναλλακτικό περιεχόμενο σε παλιότερους browsers [5].

2.2.7 CSS

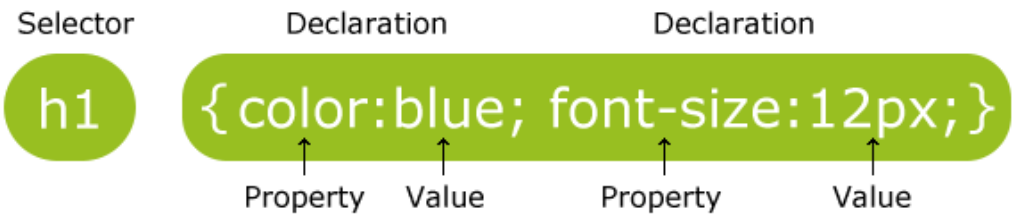
Η CSS (Cascading Style Sheet) είναι μια γλώσσα στυλ η οποία χρησιμοποιείται για να καθορίσει τη σχεδίαση, δηλαδή την εμφάνιση μιας ιστοσελίδας σε οποιοδήποτε μέσο. Ουσιαστικά βοηθάει το προγραμματιστή να διαμορφώσει την ιστοσελίδα του προσθέτοντας χαρακτηριστικά σε αυτήν όπως χρώματα, στοίχιση, backgrounds κ.α. Θα μπορούσαμε να πούμε ότι η CSS είναι συμπληρωματική καθώς και απαραίτητη για την HTML. Η HTML δομεί το περιεχόμενο ενώ η CSS το μορφοποιεί [6] [7].

- Χρησιμότητα της CSS

  Η CSS δίνει τη δυνατότητα ελέγχου της διαμόρφωσης πολλών σελίδων ταυτόχρονα χρησιμοποιώντας ένα μόνο αρχείο CSS κι έτσι γλιτώνει πάρα πολύ χρόνο από τον προγραμματιστή. Για παράδειγμα εάν θέλουμε να αλλάξουμε το χρώμα των επικεφαλίδων <h1> σε ολόκληρο το αρχείο θα πρέπει σε κάθε ετικέτα να το κάνω ξεχωριστά χωρίς CSS ενώ με CSS αυτό θα το κάνω μόνο μια φορά για όλες τις ετικέτες <h1>.

- Σύνταξη της CSS

  Αποτελείται από ένα selector και από ένα μπλοκ δηλώσεων. Ο selector δείχνει στον στοιχείο HTML που θέλουμε να διαμορφώσουμε, οι δηλώσεις μπορεί να είναι μια ή περισσότερες , χωρίζονται με ελληνικό ερωτηματικό και περικλείονται σε αγκύλες [8].



Εικόνα 4 - CSS Syntax

Παράδειγμα: Όλα τα στοιχεία παραγράφου θα έχουν κείμενο σε κόκκινο χρώμα και στοίχιση στο κέντρο.

```
p {
    color: red;
    text-align: center;
}
```

```
h1, h2, p  {
   text-align:  center;
   color:  red;
}
```

Σχόλια

```
p {
   color: red;
   /* This is a single-line comment */
   text-align: center;
}

/* This is
a multi-line
comment */
```

- Τρόποι εισαγωγής CSS

  Υπάρχουν τρεις τρόποι εισαγωγής [9].

  I. **Με εξωτερικό αρχείο.**
  Η δήλωση γίνεται πάντα στο <head>.

```
<head>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
```

  II. **Εσωτερικά στο ίδιο αρχείο.**
  Πάντα μέσα στις ετικέτες του <head>.

```
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
```

```
</style>
</head>
```

III. **Στην ίδια γραμμή του στοιχείου (inline).**
Χρήσιμο όταν θέλουμε μοναδικό στυλ για ένα στοιχείο. Έχει τη μεγαλύτερη προτεραιότητα εφαρμογής στυλ.

```
<h1 style="color:blue;margin-left:30px;">This is a
heading</h1>
```

2.2.8 Bootstrap

Το Bootstrap είναι ένα δημοφιλές HTML, CSS και JavaScript framework ανοιχτού κώδικα για ανάπτυξη responsive ιστοσελίδων και δίνει κύρια βάση στις κινητές συσκευές, δηλαδή είναι mobile first. Ένα responsive website μπορεί να προσαρμοστεί σε οποιαδήποτε οθόνη, από ένα μικρό smartphone μέχρι ένα μεγάλο υπολογιστή. Δημιουργήθηκε από το twitter και το 2012 ήταν το δημοφιλέστερο project στο GitHub. Επίσης είναι συμβατό με όλους τους μοντέρνους browsers [10] [11] [12]. Αξίζει να σημειωθεί ότι έχει επαναχρησιμοποιήσιμα συστατικά όπως dropdowns, input groups, navigation, alerts κ.α. [13].

- Πώς να ξεκινήσεις

  Υπάρχουν δύο τρόποι για να λάβεις κ να χρησιμοποιήσεις το framework. Ο πρώτος είναι να κατεβάσεις τα απαραίτητα αρχεία και να τα εγκαταστήσεις και ο άλλος είναι να το εισάγεις μέσω ενός CDN (Content Delivery Network) εισάγοντας και την JQuery (βιβλιοθήκη της JavaScript).

  CDN

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstra
p/3.3.7/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquer
y.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/boots
trap.min.js"></script>
```

Εικόνα 5 - Bootstrap

3 Κύριο Μέρος

3.1 Ανάλυση απαιτήσεων

- Λειτουργικές απαιτήσεις για το back-end
    1. Θα υπάρχει δυνατότητα στο σύστημα να γίνει εγγραφή νέου χρήστη-διαχειριστή και έλεγχος εισαγωγής στοιχείων χρήστη για είσοδο στο σύστημα. Αν τα στοιχεία εισαγωγής είναι λανθασμένα θα εμφανίζεται παράθυρο στο χρήστη για να ξαναβάλει τα στοιχεία του.
    2. Για κάθε υποκατηγορία πληροφοριών των ξενοδοχείων, ενοικιάσεων, μουσείων, αξιοθέατων, πτήσεων, πλοίων, εκδρομών θα υπάρχει ξεχωριστή φόρμα εισαγωγής στοιχείων.
    3. Για κάθε κατηγορία το σύστημα θα παρέχει όλα τα δεδομένα που υπάρχουν στη βάση σε μορφή πίνακα κι ο χρήστης θα έχει τη δυνατότητα ανανέωσης και διαγραφής εγγραφών.
    4. Για τα ξενοδοχεία, τις πτήσεις, τα πλοία και τις εκδρομές το σύστημα θα παρέχει έναν πίνακα με όλες τις κρατήσεις που έχουν γίνει κι έχουν αποθηκευτεί στη βάση.

- Μη λειτουργικές απαιτήσεις για το back-end
    1. Το σύστημα θα πρέπει να υποστηρίζεται σε όλα τα λειτουργικά συστήματα (Windows, Linux , Macintosh).
    2. Ο χρόνος απόκρισης δε θα πρέπει να ξεπερνά τα 3 δευτερόλεπτα.
    3. Είσοδος στο σύστημα μόνο από εγγεγραμμένους χρήστες.

- Λειτουργικές απαιτήσεις για το front-end
    1. Το σύστημα θα επιτρέπει αναζήτηση και κράτηση δωματίων, ενοικιάσεων, πτήσεων , δρομολογίων πλοίων και εκδρομών.
    2. Το σύστημα θα επιτρέπει αναζήτηση πληροφορίας για μουσεία και αξιοθέατα.
    3. Το σύστημα θα επιτρέπει ακύρωση κρατήσεων.

- Μη λειτουργικές απαιτήσεις για το front-end
    1. Το σύστημα θα έχει responsive design και θα είναι mobile friendly.
    2. Το σύστημα θα υποστηρίζεται από όλα τα λειτουργικά συστήματα Windows, Linux, Macintosh για desktop και Windows Phone, Android , iOS για κινητές συσκευές.
    3. Ο χρόνος απόκρισης του συστήματος να μην ξεπερνά τα 3 δευτερόλεπτα.
    4. Όταν πραγματοποιηθεί μια κράτηση να ενημερώνεται ο πελάτης για την επιτυχία της κράτησης.
    5. Όταν πραγματοποιηθεί ακύρωση μιας κράτησης να ενημερώνεται ο πελάτης για την επιτυχία της.

3.1.2 Απαιτήσεις Συστήματος

Το σύστημα μας θα πρέπει να αποθηκεύει την παρακάτω πληροφορία:

1. Τα ξενοδοχεία της περιοχής (ονομασία, διεύθυνση, κατηγορία, τηλέφωνο, email, αριθμός & κατηγορία δωματίων, υπηρεσίες δωματίων)
2. Τα γραφεία ενοικιάσεων αυτοκινήτων (ονομασία, διεύθυνση, τηλέφωνο) καθώς και
    a. Πληροφορία για το κάθε αυτοκίνητο που διαθέτει το κάθε γραφείο (αριθμό κυκλοφορίας, κυβικά, θέσεις επιβατών και κατηγορία στην οποία ανήκει. Επίσης θέλομε να αποθηκεύομε πληροφορία όσον αφορά την χρέωση ανά μέρα και αν αυτή αλλάζει καθώς αυξάνονται οι μέρες ενοικιάσεων.
    b. Πληροφορία για το κάθε μηχανάκι/μηχανή που διαθέτει το κάθε γραφείο (αριθμό κυκλοφορίας, κυβικά και κατηγορία). Πληροφορίες για την χρέωση.
3. Πληροφορίες για τα μουσεία (ονομασία, διεύθυνση, τηλέφωνο, email, τιμή εισιτηρίου, ωράριο λειτουργίας).
4. Πληροφορίες για τα αξιοθέατα (ονομασία, διεύθυνση, τηλέφωνο, email, τιμή εισιτηρίου, ωράριο λειτουργίας).
5. Πληροφορίες για τα δρομολόγια των αεροπλάνων και πλοίων για την άφιξη και αναχώρηση στο/από συγκεκριμένο μέρος. Θα περιλαμβάνει τιμές, ώρες και μέρες και εταιρία η οποία κάνει την αντίστοιχη πτήση / ακτοπλοϊκό δρομολόγιο.
6. Πληροφορίες για τα εκδρομικά γραφεία της περιοχής (ονομασία, διεύθυνση, τηλέφωνο, email). Για κάθε γραφείο θα περιέχονται πληροφορίες για ποιες εκδρομές διοργανώνει (αφετηρία, προορισμό και ώρες που γίνονται) και ποιες είναι οι τιμές.

Λειτουργίες συστήματος

Το σύστημα θα πρέπει:

- Να επιτρέπει στο χρήστη να κάνει αναζήτηση, κράτηση και ακύρωση κράτησης δωματίων μέσω του διαδικτύου. Για παράδειγμα ένα δωμάτιο δίκλινο κατηγορίας Α με Wi-Fi.
- Να επιτρέπει στο χρήστη να κάνει αναζήτηση και κράτηση μεταφορικών μέσων για το χρονικό διάστημα που επιθυμεί και να έχει τη δυνατότητα ακύρωσης της κράτησης.
- Να επιτρέπει στο χρήστη να κάνει αναζήτηση για πτήσεις και δρομολόγια πλοίων την ημερομηνία που επιθυμεί και να έχει δυνατότητα ακύρωσης της κράτησης.
- Να επιτρέπει στο χρήστη να αναζητήσει πληροφορίες για αξιοθέατα και μουσεία.
- Να επιτρέπει στο χρήστη να δηλώσει συμμετοχή σε εκδρομές.

3.2 Σχεδιασμός υλοποίησης

- Σύστημα Back-End

Όσον αφορά το back-end σύστημα θα πρέπει να είναι εύκολο στη χρήση του έτσι ώστε ο διαχειριστής να είναι σε θέση να εισάγει τις απαραίτητες πληροφορίες. Το πρώτο βήμα θα είναι μια σελίδα για εγγραφή χρήστη και είσοδο στο σύστημα. Μετά την είσοδο του χρήστη θα μπορεί να εισάγει πληροφορίες στο σύστημα. Θα πρέπει να είναι ξεκάθαρες οι κατηγορίες και τα πεδία των φορμών στις οποίες ανήκουν. Για να το πετύχουμε αυτό θα υλοποιήσουμε ένα μενού πλοήγησης στην κορφή της σελίδας. Σε κάθε κουμπί του μενού θα αντιστοιχεί μια κατηγορία η οποία θα οδηγεί σε άλλη σελίδα. Σε κάθε κατηγορία θα υπάρχει ένα υπό-μενού στην άκρη της σελίδας για πλοήγηση στις υποκατηγορίες. Πιο συγκεκριμένα το μενού θα περιλαμβάνει τα εξής:

- Hotels, Rentals, Museums, Landmarks, Flights, Ships, Excursions

Και τα υπό-μενού θα είναι,

Για **Hotels** οι σελίδες:

- **Hotels, Rooms**, **Room Services, Bookings** οι οποίες περιλαμβάνουν φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα σε κάθε σελίδα με όλες τις εγγραφές στη βάση

Για **Rentals** οι σελίδες:

- **Rental Offices, Cars, Car Charges, Motorcycles, Motorcycle Charges, Car Bookings, Motorcycle Bookings** οι οποίες περιλαμβάνουν φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα σε κάθε σελίδα με όλες τις εγγραφές στη βάση

Για **Museums**:

- Μια μόνο σελίδα **Museums** η οποία περιλαμβάνει φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα με όλες τα καταχωρημένα μουσεία

Για **Landmarks**:

- Μια μόνο σελίδα **Landmarks** η οποία περιλαμβάνει φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα με όλες τα καταχωρημένα αξιοθέατα

Για **Flights** οι σελίδες:

- **Companies**, **Airplanes**, **Routes**, **Bookings** οι οποίες περιλαμβάνουν φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα σε κάθε σελίδα με όλες τις εγγραφές στη βάση

Για **Ships** οι σελίδες:

- **Companies**, **Ships**, **Routes**, **Bookings** οι οποίες περιλαμβάνουν φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα σε κάθε σελίδα με όλες τις εγγραφές στη βάση

Για **Excursions** οι σελίδες:

- **Excursion Offices**, **Excursions**, **Packets**, **Bookings** οι οποίες περιλαμβάνουν φόρμα για την εισαγωγή των στοιχείων και ένα πίνακα σε κάθε σελίδα με όλες τις εγγραφές στη βάση

Όταν ο χρήστης κάνει καταχώρηση στη βάση δεδομένων ενημερώνεται αυτόματα ο πίνακας με τις εγγραφές. Επίσης όταν κάνει ανανέωση ή διαγραφή μιας εγγραφής θα εμφανίζεται μήνυμα επιτυχίας και κάνοντας refresh θα ανανεωθεί ο πίνακας με τις εγγραφές.

- Σύστημα Front-End
  - Όσον αφορά το front-end σύστημα θα πρέπει να είναι φιλικό και κατανοητό προς τον πελάτη. θα πρέπει να είναι σε θέση να βρει αυτό που αναζητά χωρίς δυσκολίες στην πλοήγηση. Για επιτευχθεί αυτό θα χρησιμοποιήσουμε περίπου την ίδια μέθοδο με το back-end σύστημα δηλαδή ένα μενού πλοήγησης με τα εξής:
    - Home Page, Hotels, Rentals, Museums, Landmarks, Flights, Ships, Excursions
    - **Home Page**

      Αυτή είναι η αρχική σελίδα που του συστήματος. Είναι η σελίδα που θα καλωσορίσει τον πελάτη και θα πρέπει να έχει φιλική εικόνα.
    - **Hotels**

      Σε αυτή τη σελίδα ο πελάτης θα είναι σε θέση να αναζητήσει δωμάτιο με κριτήρια την ημερομηνία check in και check out, τα άτομα και την κατηγορία του ξενοδοχείου. Το σύστημα θα υπολογίζει αυτόματα τον τύπο του δωματίου ανάλογα με τα άτομα.
    - **Rental Offices**

      Σε αυτή τη σελίδα ο πελάτης θα είναι σε θέση να κάνει αναζήτηση αυτοκινήτων και μηχανών με κριτήρια την ημερομηνία, τα κυβικά, την κατηγορία και τους επιβάτες (για αυτοκίνητο)
    - **Museums**

      Σε αυτή τη σελίδα θα εμφανίζονται με τη μορφή πίνακα όλα τα διαθέσιμα μουσεία της περιοχής

- **Landmarks**

  Σε αυτή τη σελίδα θα εμφανίζονται με τη μορφή πίνακα όλα τα διαθέσιμα αξιοθέατα της περιοχής

- **Flights**

  Σε αυτή τη σελίδα ο πελάτης θα είναι σε θέση να αναζητήσει πτήσεις με βάση την ημέρα αναχώρησης και άφιξης καθώς και τα άτομα

- **Ships**

  Σε αυτή τη σελίδα ο πελάτης θα είναι σε θέση να αναζητήσει δρομολόγια πλοίων με βάση την ημέρα αναχώρησης και άφιξης καθώς και τα άτομα

- **Excursions**

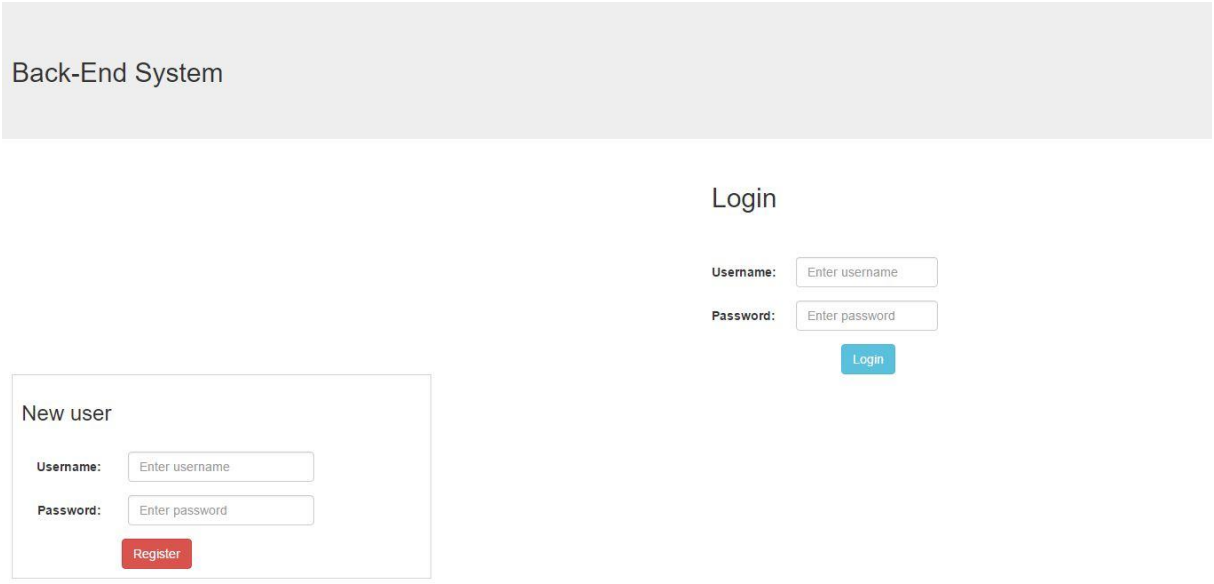
  Σε αυτή τη σελίδα ο πελάτης θα είναι σε θέση να δει τις διαθέσιμες εκδρομές και να δηλώσει συμμετοχή στην εκδρομή που επιθυμεί.

Σε όλες τις σελίδες που γίνονται κρατήσεις θα υπάρχει ένας σύνδεσμος για ακύρωση της κράτησης. Ο πελάτης το μόνο που χρειάζεται να κάνει είναι να εισάγει τον κωδικό της κράτησης του και να πατήσει ακύρωση.

3.3 Υλοποίηση

3.3.1 Σύστημα Back-End

Η αρχική σελίδα που εμφανίζεται είναι το αρχείο *login.jsp*.



Εικόνα 6 - Login Page

Εγγραφή χρήστη

Εάν ο χρήστης-διαχειριστής δεν είναι εγγεγραμμένος θα κάνει εγγραφή εισάγοντας username και password στο αριστερό πλαίσιο. Το αρχείο που χρησιμοποιείται είναι το *RegisterUser.java.*

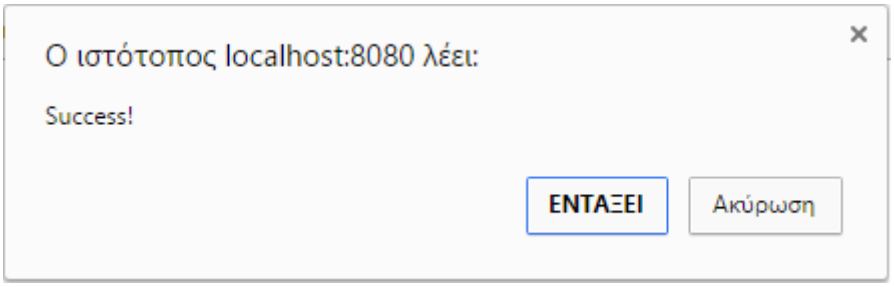
Εικόνα 7 - Register

Το servlet αρχείο *RegisterUser.java*

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    String username = request.getParameter("reg_username");
    String password = request.getParameter("reg_password");

    try{
        Class.forName("oracle.jdbc.driver.OracleDriver");
        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

        ps = conn.prepareStatement("insert into  Users values(?,?)");

        ps.setString(1 , username);
        ps.setString(2 , password);
        rs = ps.executeQuery();


        if(rs.next()){
            writer.println("<script type=\"text/javascript\"> if(confirm('Success!') == true) window.location='http://localhost:8080/ThesisAdmin/login.jsp'; </script>");

            ps.close();
            rs.close();
            conn.close();

        } else {
            writer.println("<script type=\"text/javascript\"> if(confirm('Error!') == true) window.location='http://localhost:8080/ThesisAdmin/login.jsp'; </script>");
        }


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

Εικόνα 8 - RegisterUser.java

Με την εγγραφή το σύστημα θα εμφανίσει μήνυμα επιτυχίας και θα επιστρέψει στην αρχική οθόνη για είσοδο στο σύστημα



Εικόνα 9 - Success Message

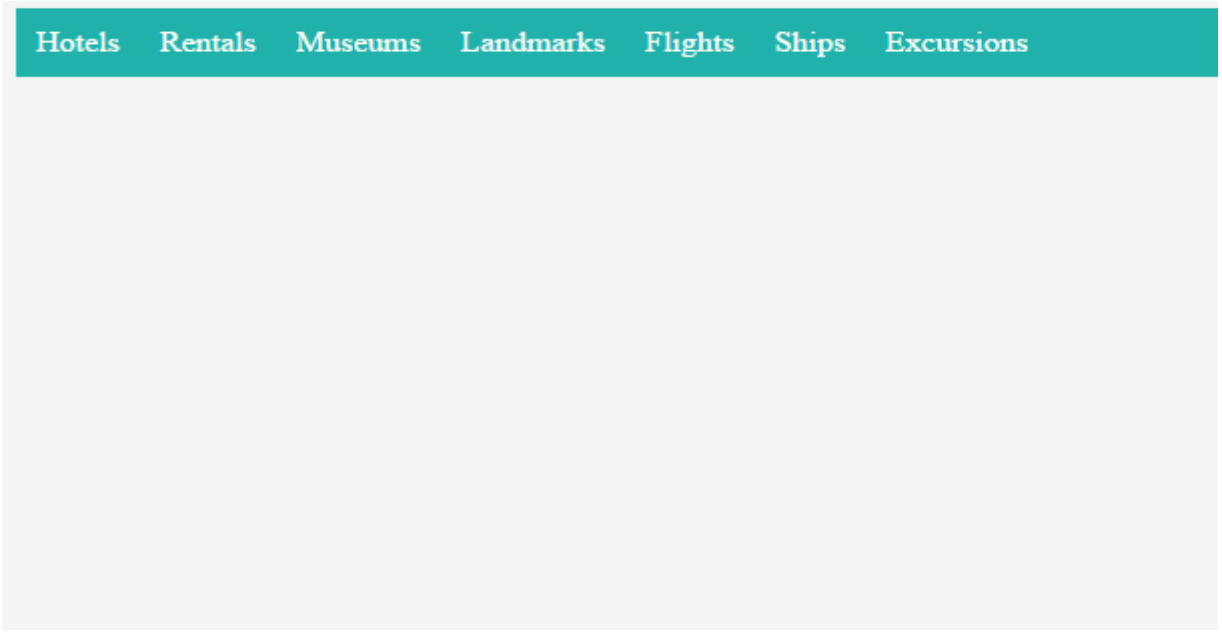
Login χρήστη

Εισάγοντας τα στοιχεία του ο χρήστης και πατώντας login θα εμφανιστεί η αρχική σελίδα index.html. Το αρχείο που χρησιμοποιείται είναι το *CheckLogin.java*.

Το servlet αρχείο *CheckLogin.java*

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    PrintWriter writer = response.getWriter();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    String username = request.getParameter("username");
    String password = request.getParameter("password");

    try{
        Class.forName("oracle.jdbc.driver.OracleDriver");
        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

        ps = conn.prepareStatement("select username,pass from Users where username=? and pass=?");

        ps.setString(1 , username);
        ps.setString(2 , password);
        rs = ps.executeQuery();


        if(rs.next()){
            response.sendRedirect("http://localhost:8080/ThesisAdmin/index.html");

            ps.close();
            rs.close();
            conn.close();

        } else {
            writer.println("<script type=\"text/javascript\"> if(confirm('Your login data is wrong!') == true) "
                    + "window.location='http://localhost:8080/ThesisAdmin/login.jsp'; </script>");
        }
```

Εικόνα 10 - CheckLogin.java

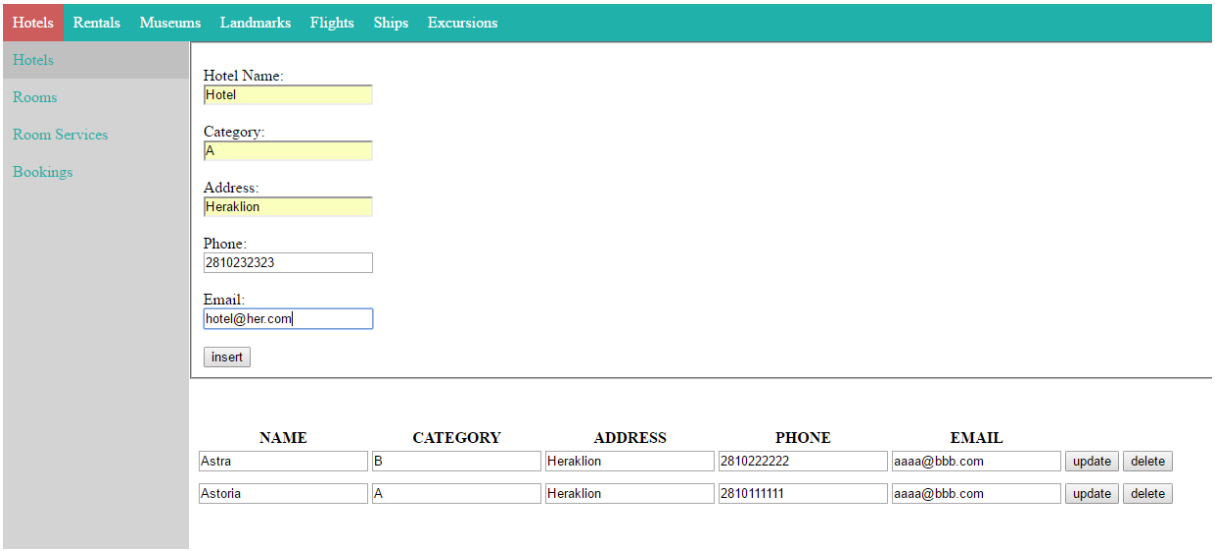


Εικόνα 11 - index.html

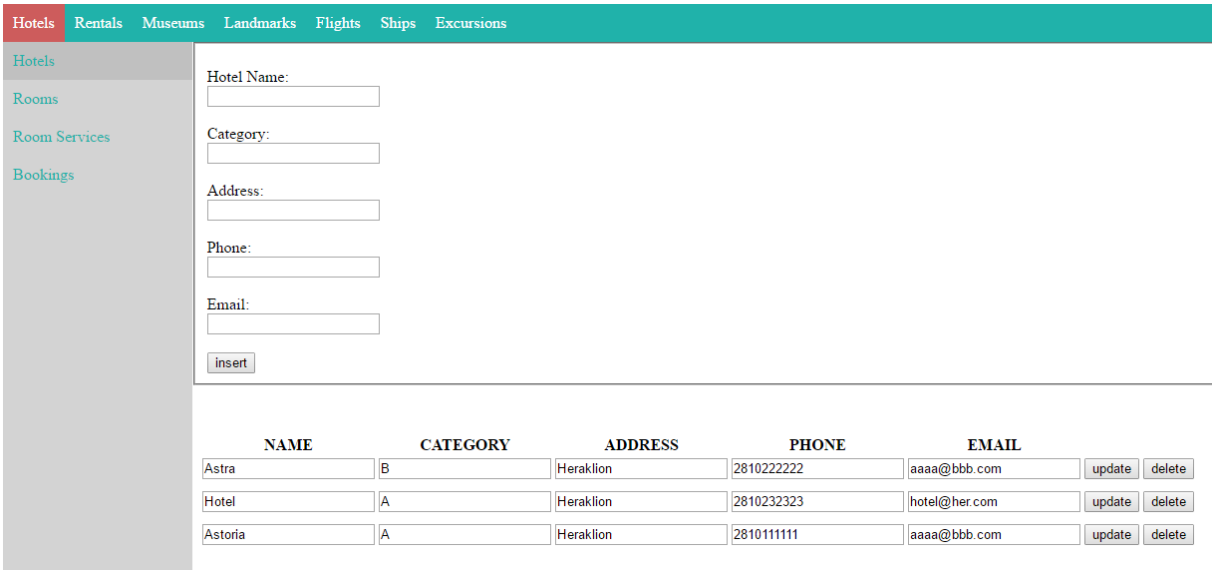
Πλέον ο διαχειριστής είναι σε θέση να περιηγηθεί σε όποια κατηγορία επιθυμεί, να εισάγει πληροφορίες, να τις ανανεώσει ακόμα και να τις διαγράψει.

Εισαγωγή τιμών

Παρακάτω έχουμε επιλέξει τα ξενοδοχεία κι έχουμε εισάγει δεδομένα στη φόρμα για ένα νέο ξενοδοχείο. Όταν πατήσουμε insert ο πίνακας με τα ξενοδοχεία που βρίσκεται αποκάτω θα ανανεωθεί αυτόματα.



Εικόνα 12 - New Hotel 1



Εικόνα 13 - New Hotel 2

Για την εισαγωγή τιμών και την εμφάνιση του πίνακα με τις εγγραφές χρησιμοποιούμε το servlet αρχείο *DbConnect.java*

```
if(request.getParameter("hotelButton") != null){
    String hotelName = request.getParameter("hotelName");
    String hotelCategory = request.getParameter("hotelCategory");
    String hotelAddress = request.getParameter("hotelAddress");
    String hotelPhone = request.getParameter("hotelPhone");
    String hotelEmail = request.getParameter("hotelEmail");

    try{
        Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

        PreparedStatement pst = conn.prepareStatement("insert into Hotels values (?,?,?,?,?)");
        pst.setString(1, hotelName);
        pst.setString(2, hotelCategory);
        pst.setString(3, hotelAddress);
        pst.setString(4, hotelPhone);
        pst.setString(5, hotelEmail);
        result = pst.executeUpdate();

        response.sendRedirect("http://localhost:8080/ThesisAdmin/Hotels/hotelsForm.html");

        conn.close();
        pst.close();

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

Εικόνα 14 - Insert Values Code

```
if(request.getParameter("hotelTableBtn") != null){
    String query = "select * from Hotels";
    response.setCharacterEncoding("utf-8");
    Connection conn;

    try{
        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount();

        pw.println("<table border=1 style=border-collapse:collapse>");

        pw.println();
        String name = "";
        int cnt = 0;

        while(rs.next()){
            int i;

            pw.println("<tr>");
            pw.println("<table><tr><td>");
            pw.println("<tr>");

            if(cnt == 0){
                for(i = 1; i <= columnCount; i++){
                    pw.println("<th>" + metaData.getColumnName(i));
                }
                cnt = 1;
            }
```

Εικόνα 15 - Hotel Table 1

```
        pw.println("</tr>");
        pw.write("<form action=/ThesisAdmin/UpdateTables method=post target=_self>");

        for(i = 1; i <= columnCount; i++){

            if(i == 1)
                name = "hotelName";
            else if (i == 2)
                name = "hotelCategory";
            else if (i == 3)
                name = "hotelAddress";
            else if (i == 4)
                name = "hotelPhone";
            else
                name = "hotelEmail";

            pw.println("<td style=border:none> <input type=text value=" + rs.getString(i) + " name=" + name + "> </td>");
        }

            pw.println("<td> <input type=submit value=update name=updateBtn_Hotel> </td>");
            pw.println("<td><input type=submit value=delete name=deleteBtn_Hotel> </td>");

            pw.write("</form>");
            pw.write("</td></tr></table>");
            pw.println("</tr>");
            pw.println();
        }
        pw.println("</table>");

        conn.close();
        st.close();
        rs.close();

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

Εικόνα 16 - Hotel Table 2

Ανανέωση τιμών

Για παράδειγμα επιλέγουμε να αλλάξουμε το email του ξενοδοχείου Astoria

NAME	CATEGORY	ADDRESS	PHONE	EMAIL		
Astra	B	Heraklion	2810222222	aaaa@bbb.com	update	delete
Hotel	A	Heraklion	2810232323	hotel@her.com	update	delete
Astoria	A	Heraklion	2810111111	asto@her.com	update	delete

Εικόνα 17 - Update Values

Όταν ο διαχειριστής πατήσει το κουμπί update το σύστημα θα εμφανίσει pop-up παράθυρο να με μήνυμα επιτυχίας. Έπειτα κάνοντας ανανέωση της σελίδας θα έχει ανανεωθεί ο πίνακας. Για την εργασία αυτή χρησιμοποιούμε το servlet αρχείο *UpdateTables.java.*

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    if (request.getParameter("updateBtn_Hotel") != null) {

        String update = "update Hotels set Category=?,Address=?,Phone=?,Email=? where Name=? ";
        pst = conn.prepareStatement(update);

        pst.setString(1, value2);
        pst.setString(2, value3);
        pst.setString(3, value4);
        pst.setString(4, value5);
        pst.setString(5, value1);
        result = pst.executeUpdate();

    }
```

Εικόνα 18 - Update Values Code

Διαγραφή μιας καταχώρησης

Έστω ότι ο διαχειριστής επιλέγει να σβήσει το ξενοδοχείο με όνομα Hotel. Πατώντας το κουμπί delete το σύστημα θα εμφανίσει pop-up παράθυρο να με μήνυμα επιτυχίας. Έπειτα κάνοντας ανανέωση της σελίδας θα έχει ανανεωθεί ο πίνακας. Για την εργασία αυτή χρησιμοποιούμε το servlet αρχείο *UpdateTables.java.*

NAME	CATEGORY	ADDRESS	PHONE	EMAIL		
Astra	B	Heraklion	2810222222	aaaa@bbb.com	update	delete
Astoria	A	Heraklion	2810111111	aaaa@bbb.com	update	delete

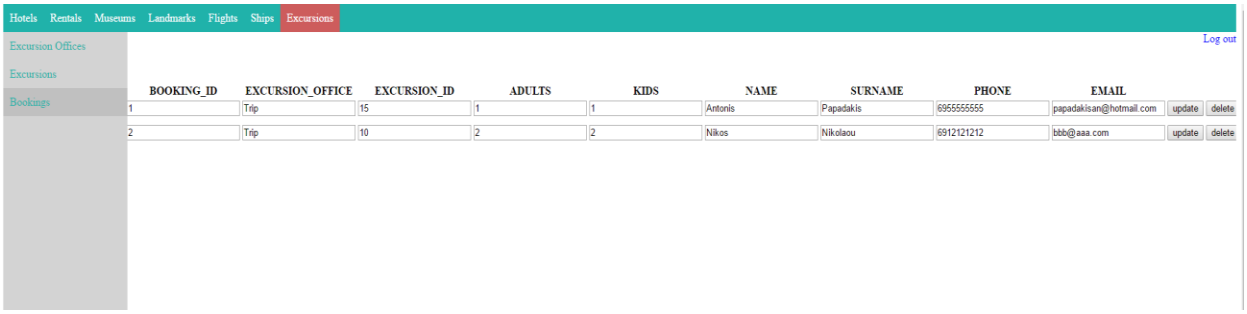
Εικόνα 19 - Delete Record

```
else if (request.getParameter("deleteBtn_Hotel") != null) {

    stm = conn.createStatement();

    stm.executeUpdate("delete from Hotel_Bookings where Hotel_Name= '" + value1 + "' ");
    stm.executeUpdate("delete from RoomServices where Hotel_Name= '" + value1 + "' ");
    stm.executeUpdate("delete from Rooms where Hotel_Name= '" + value1 + "' ");
    result = stm.executeUpdate("delete from Hotels where Name= '" + value1 + "' ");

}
```

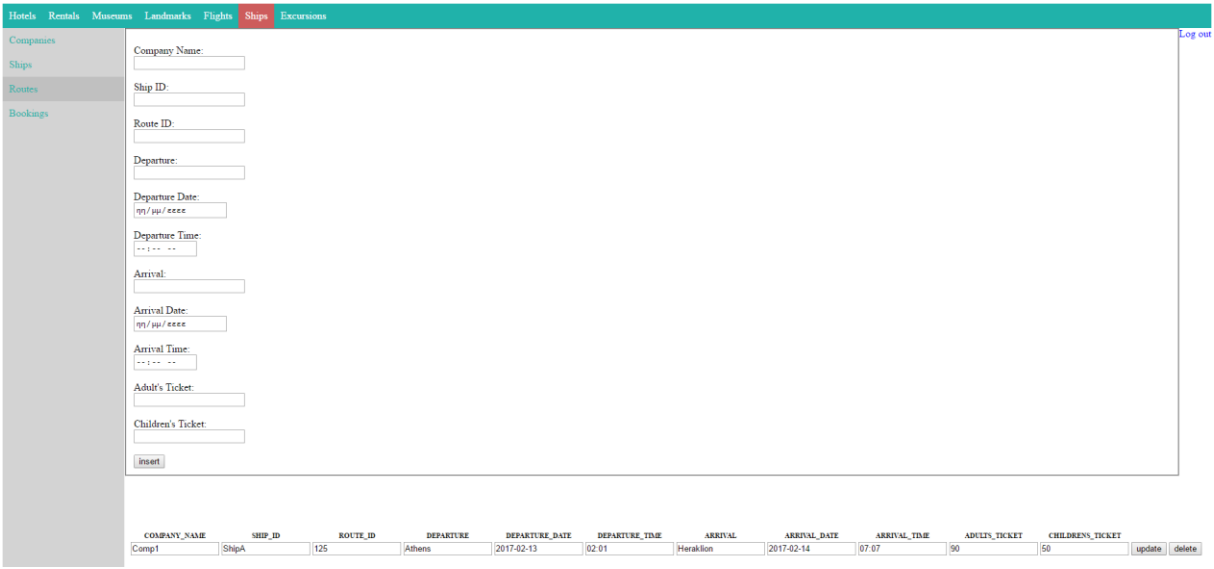
Εικόνα 20 - Delete Record Code

Κρατήσεις

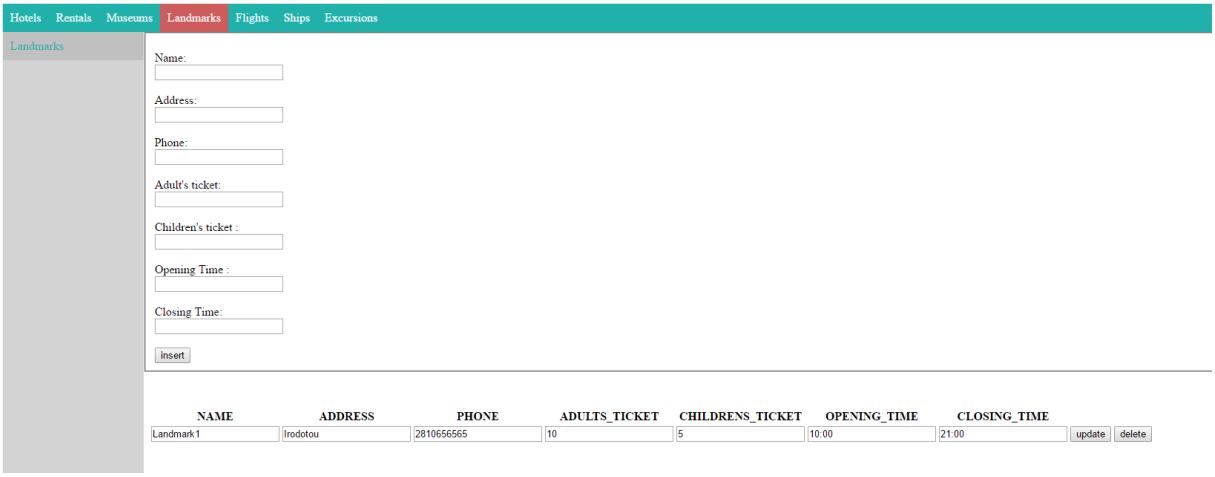
Ο διαχειριστής έχει τη δυνατότητα να διαχειριστεί τον πίνακα κρατήσεων. Θα δούμε ως παράδειγμα τον πίνακα κρατήσεων των εκδρομών.



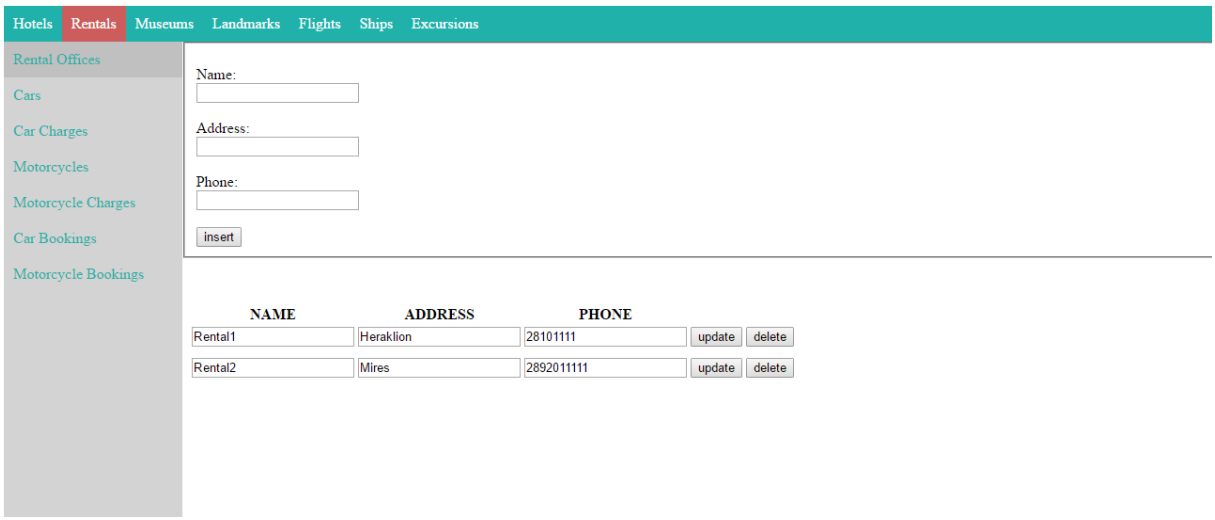
Μερικά παραδείγματα ακόμα



Εικόνα 21 - Ship Routes



Εικόνα 22 - Landmarks

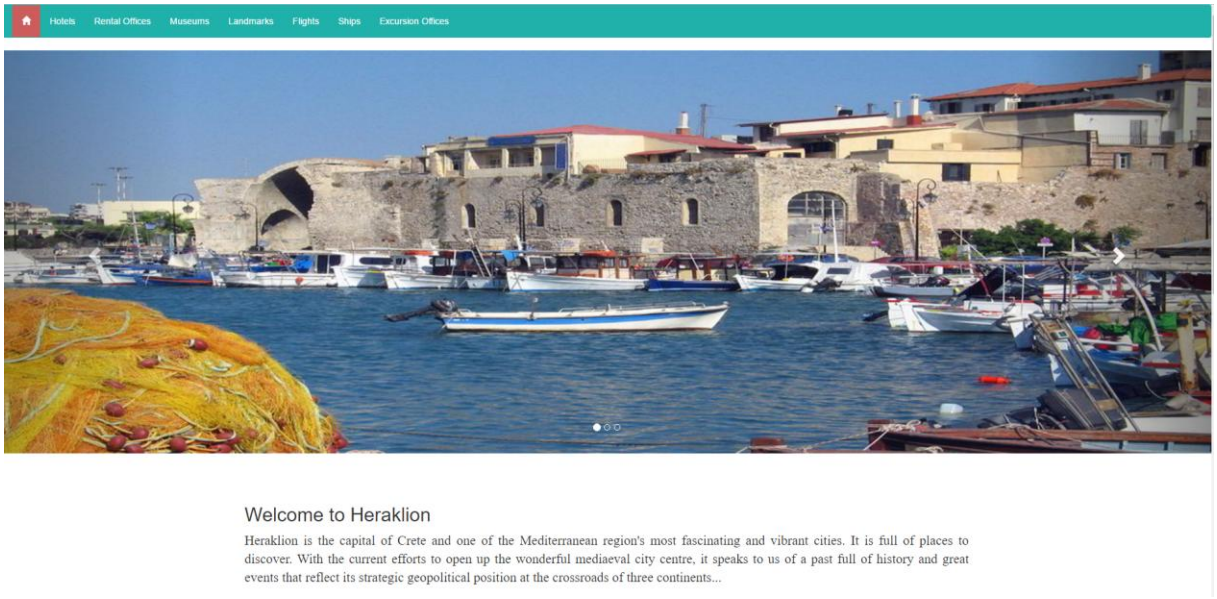


Εικόνα 23 - Rentals

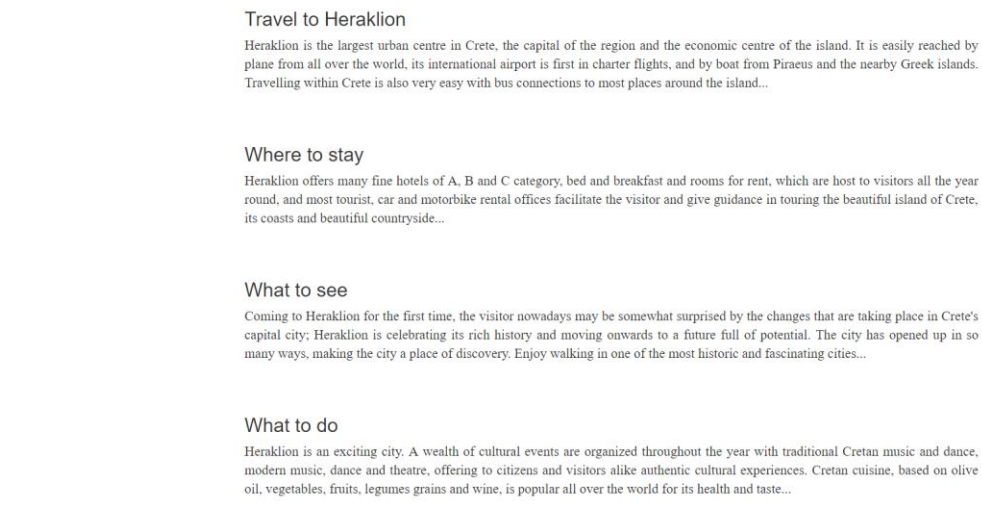
3.3.2 Σύστημα Front-End

Αρχική σελίδα

Παρακάτω παρουσιάζεται η πρώτη σελίδα που θα δει ο πελάτης. Στην κορυφή της σελίδας βρίσκεται το μενού πλοήγησης, ακριβώς αποκάτω βρίσκεται ένα carousel το οποίο αλλάζει φωτογραφία κάθε 5 δευτερόλεπτα κι έπειτα ακολουθεί μια αναφορά στις παροχές του μέρους. Ο πελάτης μπορεί εύκολα να περιηγηθεί στο σύστημα με τη βοήθεια του μενού και να αναζητήσει πληροφορίες.



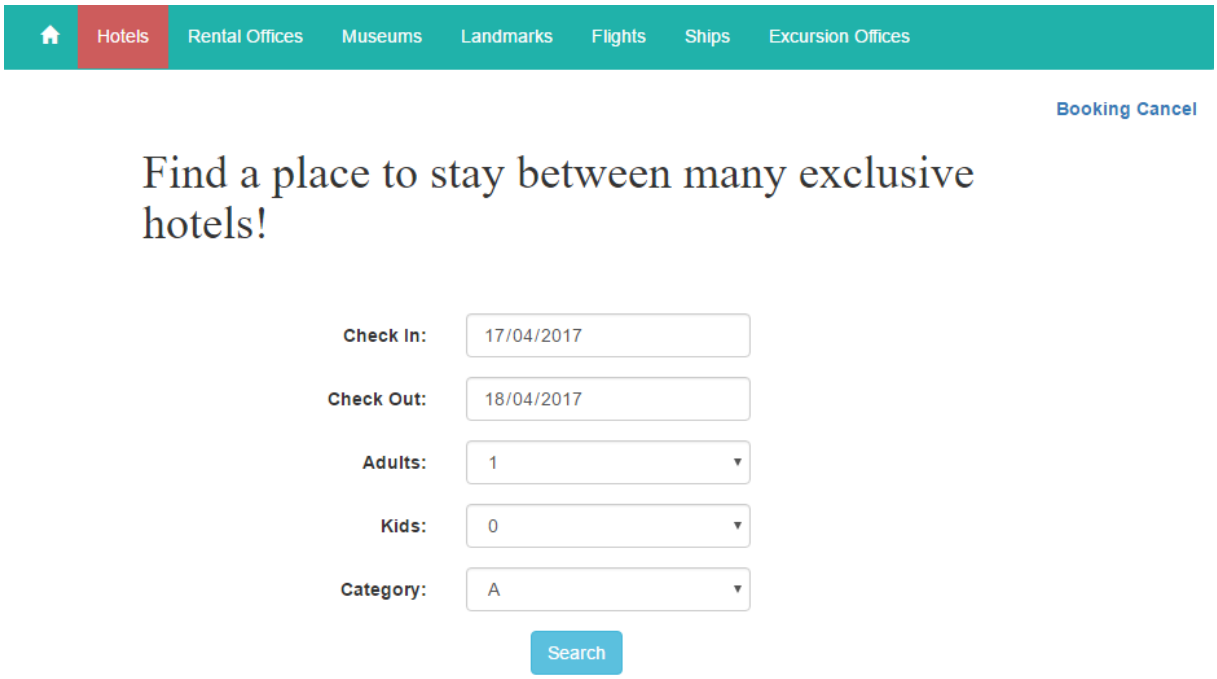
Εικόνα 24 - Home Page 1



Εικόνα 25 - Home Page 2

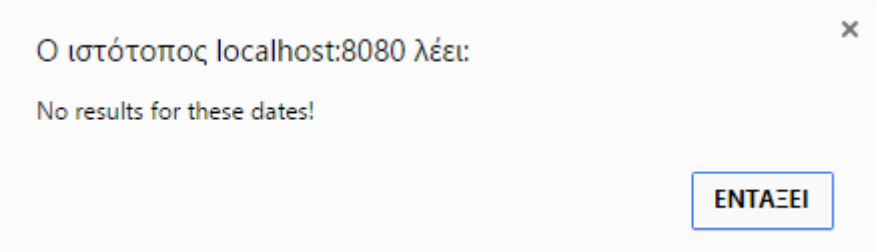
Αναζήτηση Ξενοδοχείου

Ο πελάτης είναι σε θέση να αναζητήσει δωμάτιο για διαμονή ανάλογα με το πότε επιθυμεί να κάνει check-in και check-out, την κατηγορία του ξενοδοχείου και για πόσα άτομα πρόκειται να κάνει κράτηση. Ο υπολογισμός του τύπου δωματίου γίνεται αυτόματα από το σύστημα βάσει του αθροίσματος των ενηλίκων και των παιδιών (εάν υπάρχουν). Εάν δηλαδή επιλέξει ο πελάτης έναν ενήλικα κι ένα παιδί το σύστημα θα αναζητήσει δωμάτιο με δυο κρεβάτια. Για την εργασία αυτή χρησιμοποιείται το αρχείο *search.jsp*.



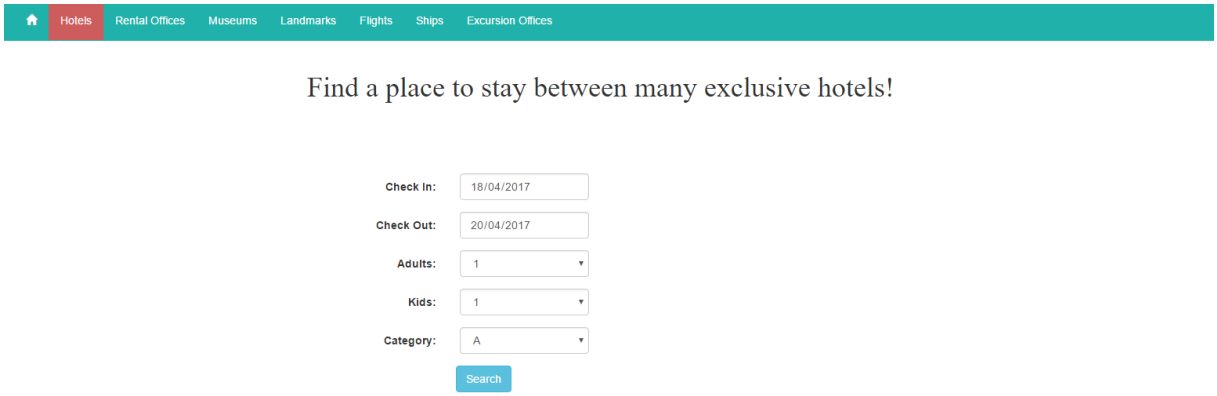
Εικόνα 26 - Hotel Search

Εάν δεν υπάρχουν διαθέσιμα δωμάτια το σύστημα θα εμφανίσει pop-up παράθυρο προς ενημέρωση του πελάτη.



Εικόνα 27 - No Result Message

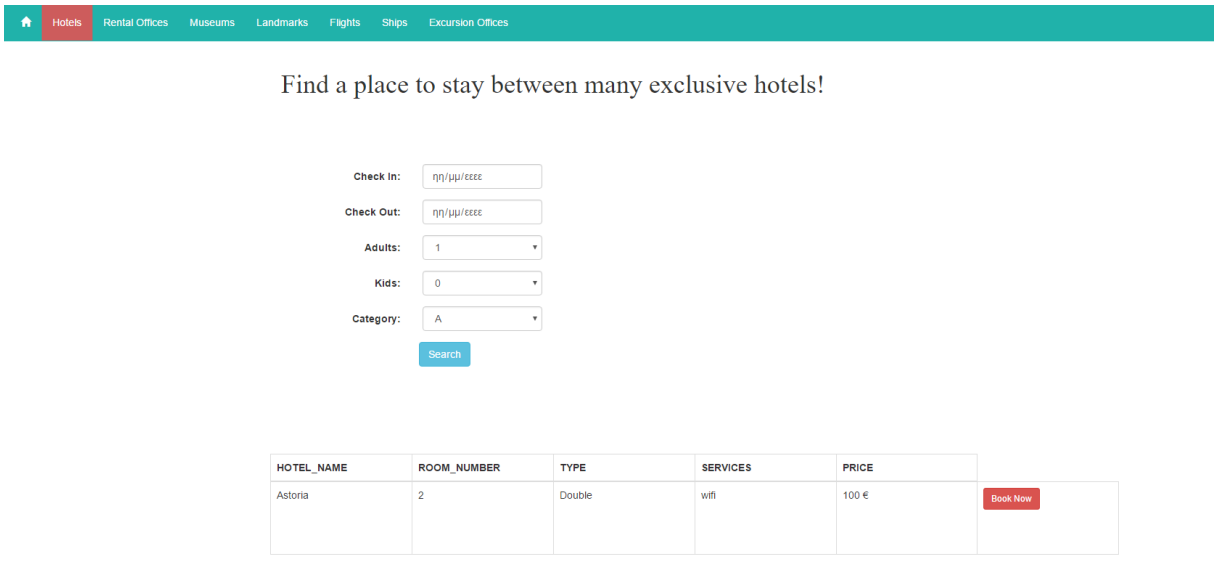
Με διαφορετικά δεδομένα εισόδου



Εικόνα 28 - Hotel Search 2

Αποτελέσματα Αναζήτησης

Εφόσον η αναζήτηση είναι επιτυχής το σύστημα θα εμφανίσει τα διαθέσιμα δωμάτια. Η χρέωση του δωματίου υπολογίζεται αυτόματα ανάλογα με τις μέρες κράτησης.



Εικόνα 29 - Search Result

Το αρχείο search.jsp

```
        //set room type

        int persons = Integer.parseInt(adults) + Integer.parseInt(kids);

        if (persons == 1)
            roomType = "Single";
        else if (persons == 2)
            roomType = "Double";
        else if (persons == 3)
            roomType = "Triple";

    try{
        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

        query = "select Rooms.Hotel_Name,Rooms.Room_Number,Rooms.Type,RoomServices.Services,Rooms.Price from Rooms inner join RoomServices on Rooms.Hotel_name=RoomServices.Hotel_Name and " +
                "RoomServices.Room_Number=Rooms.Room_Number and Rooms.Type=? "+
                "and not exists (select Room_Number from hotel_bookings where Rooms.Room_Number=hotel_bookings.Room_Number and hotel_bookings.Check_in>=? and hotel_bookings.Check_out<=?)";

        ps = conn.prepareStatement(query);
        ps.setString(1, roomType);
        ps.setDate(2 , java.sql.Date.valueOf(checkIn));
        ps.setDate(3 , java.sql.Date.valueOf(checkOut));
        rs = ps.executeQuery();

        Duration duration = Duration.between(date1.atStartOfDay(), date2.atStartOfDay());
        long days = duration.toDays();

        if(rs.isBeforeFirst()){

            metaData = rs.getMetaData();
            columnCnt = metaData.getColumnCount();


            while(rs.next()){
                rowCnt = rs.getRow();

                //calculate charge

                long price = Integer.parseInt(rs.getString("Price")) * days;

                %>
```

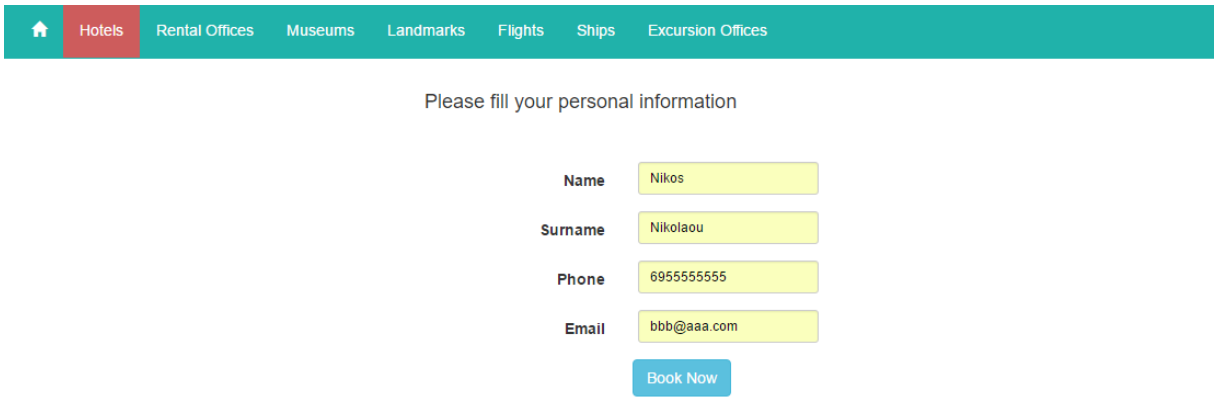
Εικόνα 30 - Room Type, Charge , Search

```
<table class='table table-bordered'>

        <%//if( k == 0){  %>

        <thead>
            <tr>

            <%  for(int i = 1; i <= columnCnt; i++){ %>

            <th><%= metaData.getColumnName(i) %></th>
            <%  } %>

        </tr>
    </thead>
        <% //   k = 1;
    //} %>

    <tbody>
        <tr>

            <% for(int i = 1; i <= columnCnt; i++){
                String name = "";


                if(i == 1){
                    name = "hotel";
                }
                else if(i == 2){
                    name = "number";
                }
                else if(i == 3){
                    name = "type";
                }
                else if(i == 4){
                    name = "services";
                }
                else if(i == 5){
                    name = "price";
                }

            if(i == 5) { %>
                <td height='100' class="col-xs-12 col-md-2"><input class='form-control' style='border: none' type='hidden' value="<%=price%>" name="<%=name%>" /> <%= price  %> &euro;</td>
        <%   } else { %>
                <td height='100' class="col-xs-12 col-md-2"><input class='form-control' style='border: none' type="hidden" value="<%=rs.getString(i)%>" name="<%=name%>" /> <%= rs.getString(i) %></td>
        <%      }
            }%>

                <td height='100' class="col-xs-12 col-md-2"><button class="btn btn-danger btn-sm" type="submit" value="Book Now">Book Now</button></td>
            </tr>
```

Εικόνα 31 - Search Results

Καταχώρηση Στοιχείων Πελάτη

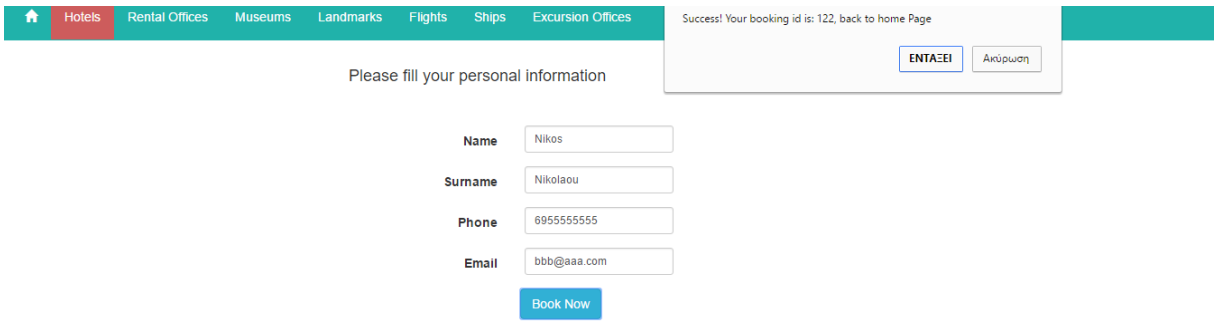
Πατώντας Book Now ο πελάτης θα μεταφερθεί στη σελίδα εισαγωγής των στοιχείων του.



Εικόνα 32 - Personal Information

Κράτηση

Πατώντας ξανά Book Now ο πελάτης θα λάβει μήνυμα επιτυχίας και τον κωδικό της κράτησης του. Το σύστημα αναζητά τον κωδικό της τελευταίας εγγραφής στον πίνακα Bookings στη βάση δεδομένων. Έπειτα γίνεται redirect στην αρχική σελίδα. Για την εργασία αυτή χρησιμοποιείται το servlet *ThanksPage.java.*



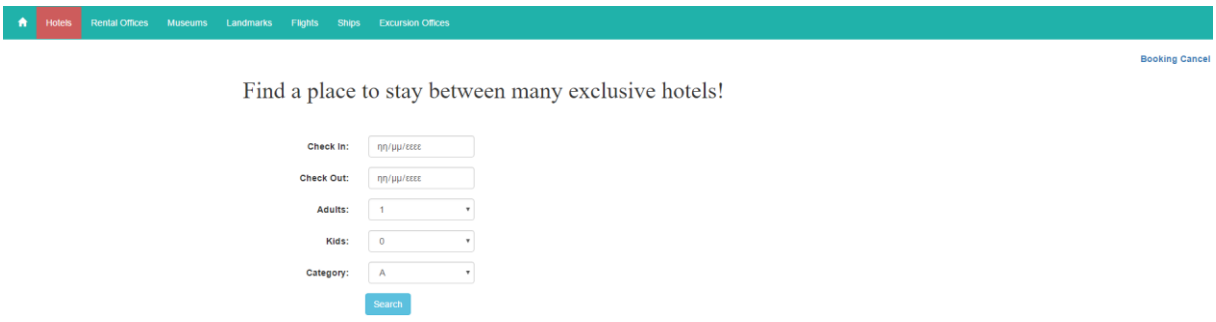
Εικόνα 33 - Booking

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");

    String checkin = request.getParameter("checkin");
    String checkout = request.getParameter("checkout");
    String adults = request.getParameter("adults");
    String kids = request.getParameter("kids");
    String hotelName = request.getParameter("hotel");
    String roomNumber = request.getParameter("number");
    String charge = request.getParameter("price");
    String name = request.getParameter("name");
    String surname = request.getParameter("surname");
    String phone = request.getParameter("phone");
    String email = request.getParameter("email");

    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");
    pst = conn.prepareStatement("insert into Hotel_Bookings values (?,?,?,?,?,?,?,?,?,?,?,?)");

    pst.setString(1, null);
    pst.setString(2, hotelName);
    pst.setString(3, roomNumber);
    pst.setDate(4, java.sql.Date.valueOf(checkin));
    pst.setDate(5, java.sql.Date.valueOf(checkout));
    pst.setString(6, adults);
    pst.setString(7, kids);
    pst.setString(8, charge);
    pst.setString(9, name);
    pst.setString(10, surname);
    pst.setString(11, phone);
    pst.setString(12, email);
    result = pst.executeUpdate();

    if (result > 0){

        stm = conn.createStatement();
        rs = stm.executeQuery("select max(booking_id) from Hotel_Bookings");
        rs.next();

        writer.println("<script type=\"text/javascript\"> if(confirm('Success! Your booking id is: "+rs.getString("max(booking_id)")+", "+
        "back to home Page') == true) window.location='http://localhost:8080/Thesis_User/index.html'; </script>");
    }

    pst.close();
    stm.close();
    rs.close();
    conn.close();
```

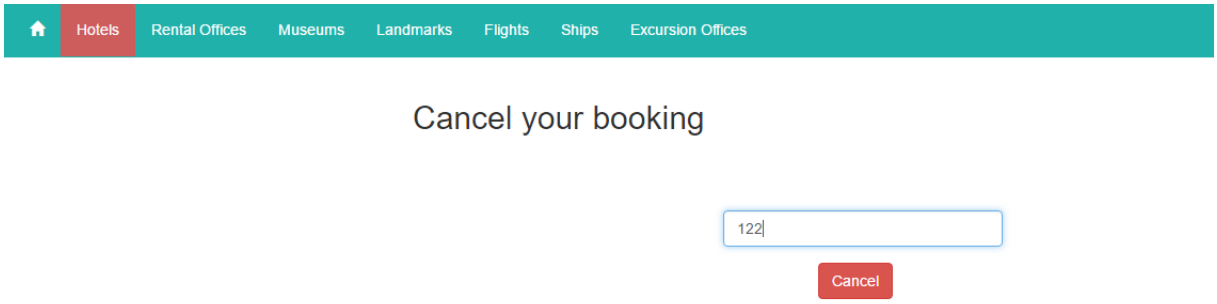
Εικόνα 34 - Thanks Page Servlet

Ακύρωση Κράτησης

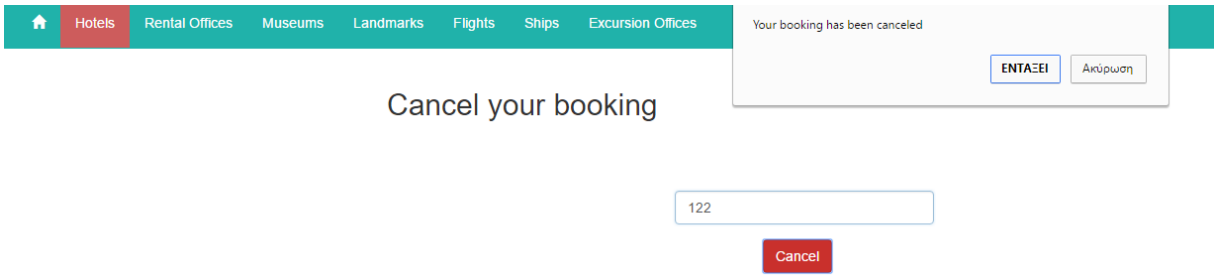
Το σύστημα παρέχει τη δυνατότητα στον πελάτη να ακυρώσει την κράτηση του κάνοντας κλικ στο σύνδεσμο που βρίσκεται στην δεξιά γωνία της σελίδας. Θα του ζητηθεί η εισαγωγή του κωδικού της κράτησης και θα εμφανιστεί μήνυμα επιτυχίας εφόσον πραγματοποιηθεί η ακύρωση. Για την εργασία αυτή χρησιμοποιείται το servlet *CancelBooking.java*.



Εικόνα 35 - Booking Cancel



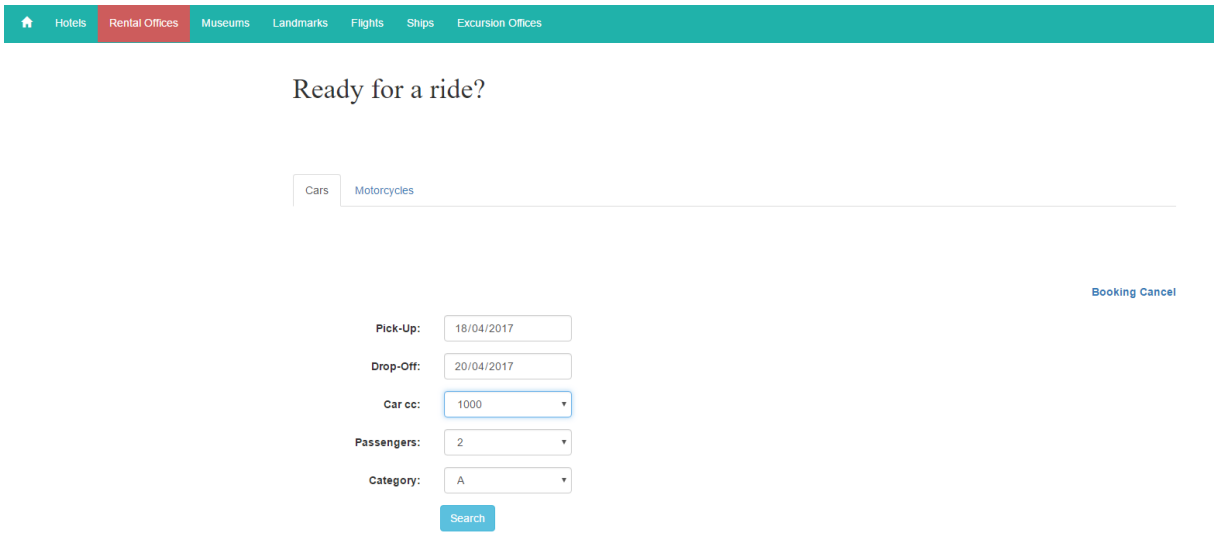
Εικόνα 36 - Booking id



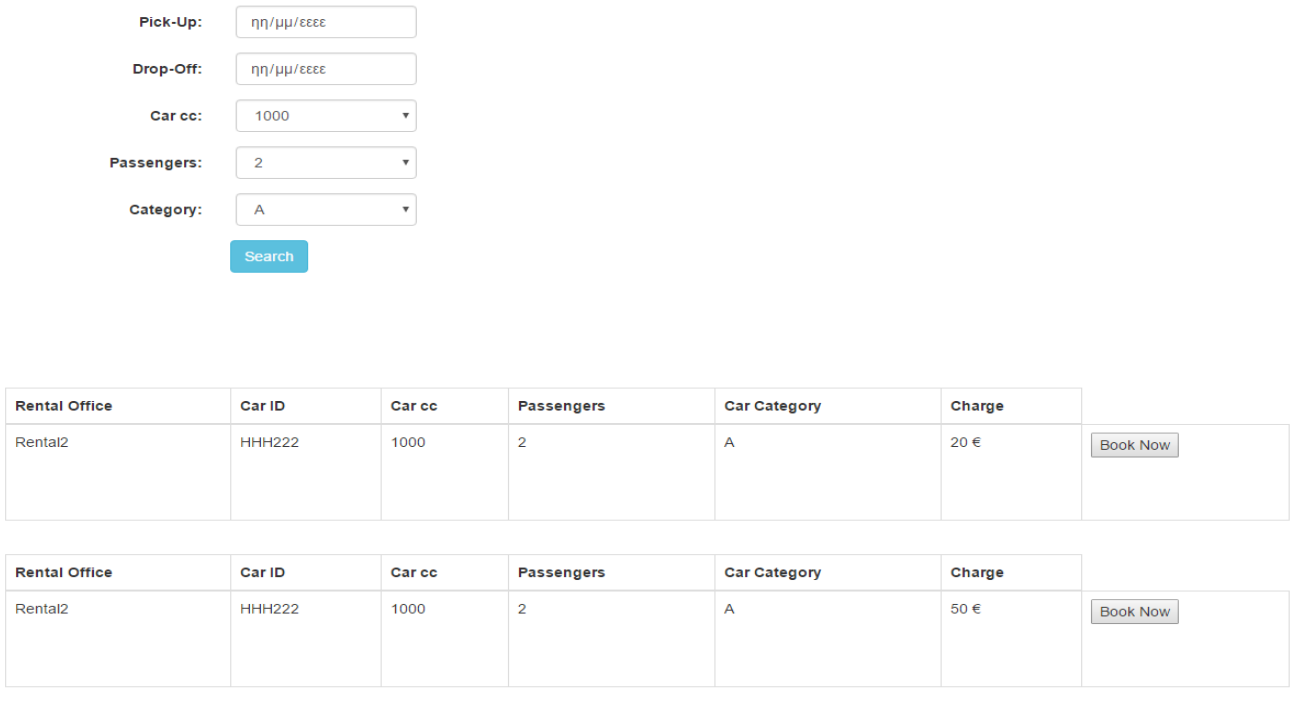
Εικόνα 37 - Cancel Success

Αναζήτηση Μεταφορικών Μέσων

Το σύστημα δίνει τη δυνατότητα στο χρήστη να κάνει αναζήτηση ανάλογα με την ημερομηνία παραλαβής και παράδοσης , τα κυβικά, την κατηγορία και τους επιβάτες αν πρόκειται για αυτοκίνητο. Η χρέωση υπολογίζεται αυτόματα βάσει των ημερών ενοικίασης. Για την εργασία αυτή χρησιμοποιείται το αρχείο *RentalSearch.jsp.*



Εικόνα 38 - Rental Search



Εικόνα 39 - Rental Results

Το αρχείο *RentalSearch .jsp*

```
try{
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

    if(request.getParameter("car_search") != null){

        query = "select Cars.Rental_Office,Cars.Car_ID,Cars.Car_cc,Cars.Passengers,Cars.Category,Car_Charges.Charge from Cars inner join Car_Charges on Cars.Car_cc=? and Cars.Passengers=? " +
                "and Cars.Category=? and Car_Charges.Days=1 and Cars.Car_id=Car_Charges.Car_id and not exists (select Car_ID from car_bookings where Cars.Car_ID=car_bookings.Car_ID and car_bookings.pick_up>=? "+
        "and car_bookings.drop_off<=? )" ;

        ps = conn.prepareStatement(query);
        ps.setString(1, car_cc);
        ps.setString(2, passengers);
        ps.setString(3, car_category);
        ps.setDate(4 , java.sql.Date.valueOf(car_pickup));
        ps.setDate(5 , java.sql.Date.valueOf(car_dropoff));
        rs = ps.executeQuery();

        date1 = LocalDate.parse(car_pickup,dtf);
        date2 = LocalDate.parse(car_dropoff,dtf);

    }else{

        query = "select Motorcycles.Rental_Office,Motorcycles.Motorcycle_ID,Motorcycles.Motorcycle_cc,Motorcycles.Category,Motorcycle_Charges.Charge from Motorcycles inner join Motorcycle_Charges on "+
                "Motorcycles.Motorcycle_id=Motorcycle_Charges.Motorcycle_id and Motorcycles.Motorcycle_cc=? and Motorcycles.Category=? and Motorcycle_Charges.Days=1 and not exists "+
                "(select Motorcycle_ID from Motor_bookings where Motorcycles.Motorcycle_ID=Motor_bookings.Motorcycle_ID and Motor_bookings.pick_up>=? "+
                "and Motor_bookings.drop_off<=? )" ;

            ps = conn.prepareStatement(query);
            ps.setString(1, motor_cc);
            ps.setString(2, motor_category);
            ps.setDate(3 , java.sql.Date.valueOf(motor_pickup));
            ps.setDate(4 , java.sql.Date.valueOf(motor_dropoff));
            rs = ps.executeQuery();

            date1 = LocalDate.parse(motor_pickup,dtf);
            date2 = LocalDate.parse(motor_dropoff,dtf);

    }

    Duration duration = Duration.between(date1.atStartOfDay(), date2.atStartOfDay());
    long days = duration.toDays();
```

Εικόνα 40 - Rental Search Code

Αποτελέσματα αναζήτησης αυτοκινήτου

```
<tbody>
    <tr>

        <% for(int i = 1; i <= columnCnt; i++){
            String name = "";

            if(i == 1){
                name = "rental_office";
            }
            else if(i == 2){
                name = "car_id";
            }
            else if(i == 3){
                name = "car_cc";
            }
            else if(i == 4){
                name = "passengers";
            }
            else if(i == 5){
                name = "car_category";
            }
            else if(i == 6){
                name = "charge";
            }

            if(i!=6){%>
                <td height='100'><input class='form-control' style='border: none' type="hidden" value="<%=  rs.getString(i) %>" name="<%= name %>" /> <%= rs.getString(i) %></td>
            <%}else{ %>
                <td height='100'><input class='form-control' style='border: none' type="hidden" value="<%=  charge %>" name="<%= name %>" /> <%= charge %> &amp;euro;</td>

    <%      }
        }%>

        <td height='100'><input type="submit" value="Book&amp;nbsp;Now" /></td>
    </tr>
```

Εικόνα 41 - Car Results Code

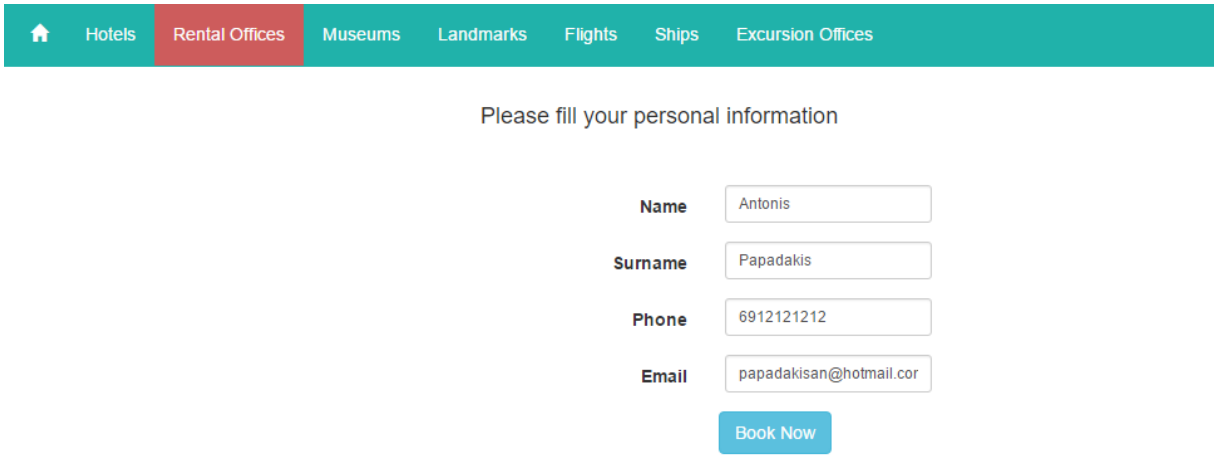
Αποτελέσματα αναζήτησης μοτοσυκλετών

```
<tbody>
    <tr>

        <% for(int i = 1; i <= columnCnt; i++){
            String name = "";


            if(i == 1){
                name = "rental_office";
            }
            else if(i == 2){
                name = "motor_id";
            }
            else if(i == 3){
                name = "motor_cc";
            }
            else if(i == 4){
                name = "motor_category";
            }
            else if(i == 5){
                name = "charge";
            }

            if(i!=5){%>
                <td height='100'><input class='form-control' style='border: none' type="hidden" value="<%=  rs.getString(i) %>" name="<%= name %>" /> <%= rs.getString(i) %></td>
            <%}else{ %>
                <td height='100'><input class='form-control' style='border: none' type="hidden" value="<%=  charge %>" name="<%= name %>" /> <%= charge %> &amp;euro;</td>

    <%      }
        }%>

        <td height='100'><input type="submit" value="Book&amp;nbsp;Now" /></td>
    </tr>
</tbody>
```

Εικόνα 42 - Moto Results Code

Κρατήσεις Μεταφορικών Μέσων

Η διαδικασία της κράτησης είναι η ίδια με αυτήν που περιγράφηκε την ενότητα κράτησης των δωματίων. Για την εργασία αυτή χρησιμοποιείται το servlet *RentalsThanksPage.java*



Εικόνα 43 - Rental Booking

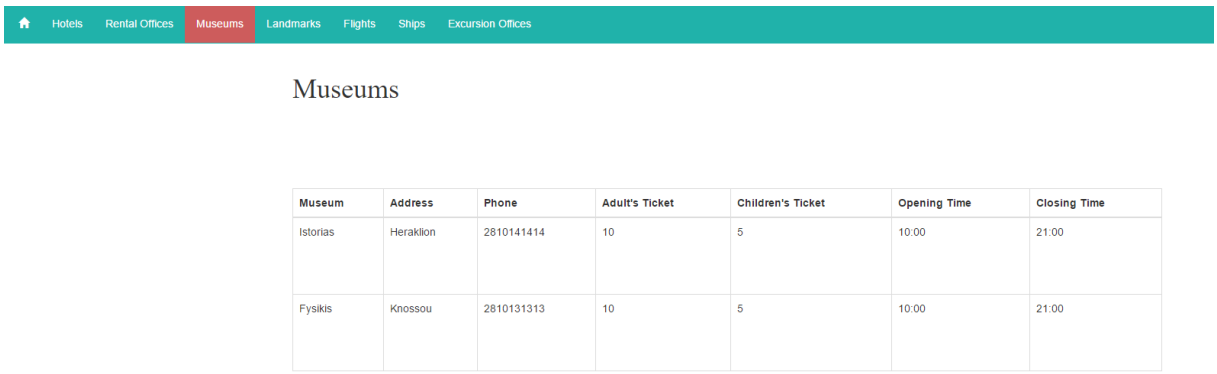
RentalsThanksPage.java

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    if (request.getParameter("booknow_car") != null) {
        pst = conn.prepareStatement("insert into Car_Bookings values (?,?,?,?,?,?,?,?,?,?)");

        pst.setString(1, null);
        pst.setString(2, rentalOffice);
        pst.setString(3, car_id);
        pst.setDate(4, java.sql.Date.valueOf(car_pickup));
        pst.setDate(5, java.sql.Date.valueOf(car_dropoff));
        pst.setLong(6, charge);
        pst.setString(7, name);
        pst.setString(8, surname);
        pst.setString(9, phone);
        pst.setString(10, email);
        result = pst.executeUpdate();

        query = "select max(Booking_id) from car_bookings";

    } else if (request.getParameter("booknow_motor") != null) {

        pst = conn.prepareStatement("insert into Motor_Bookings values (?,?,?,?,?,?,?,?,?,?)");

        pst.setString(1, null);
        pst.setString(2, rentalOffice);
        pst.setString(3, motor_id);
        pst.setDate(4, java.sql.Date.valueOf(motor_pickup));
        pst.setDate(5, java.sql.Date.valueOf(motor_dropoff));
        pst.setLong(6, charge);
        pst.setString(7, name);
        pst.setString(8, surname);
        pst.setString(9, phone);
        pst.setString(10, email);
        result = pst.executeUpdate();

        query = "select max(Booking_id) from motor_bookings";
    }

    if (result > 0){
        stm = conn.createStatement();
        rs = stm.executeQuery(query);
        rs.next();

        writer.println("<script type=\"text/javascript\"> if(confirm('Success! Your booking id is " + rs.getString("max(Booking_ID)")
                + ", back to home Page') == true) window.location='http://localhost:8080/Thesis_User/index.html'; </script>");
    }
```

Εικόνα 44 - Rentals Booking Code

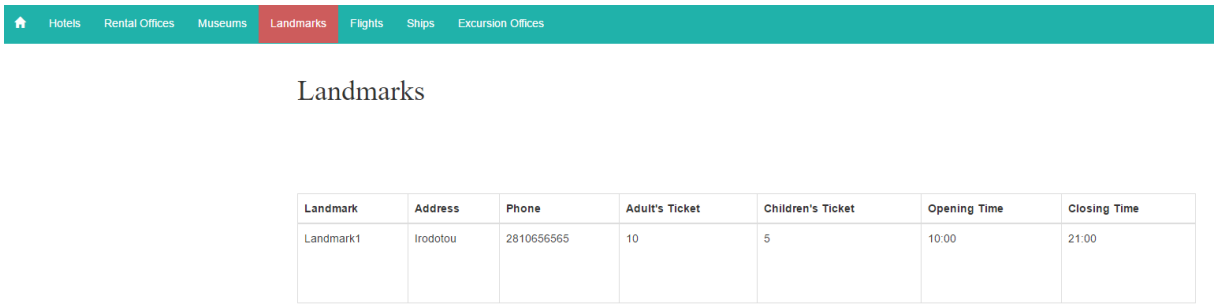
Αναζήτηση Μουσείων

Στο JSP αρχείο *Museums.jsp* εμφανίζονται όλες οι πληροφορίες για τα διαθέσιμα μουσεία της περιοχής.



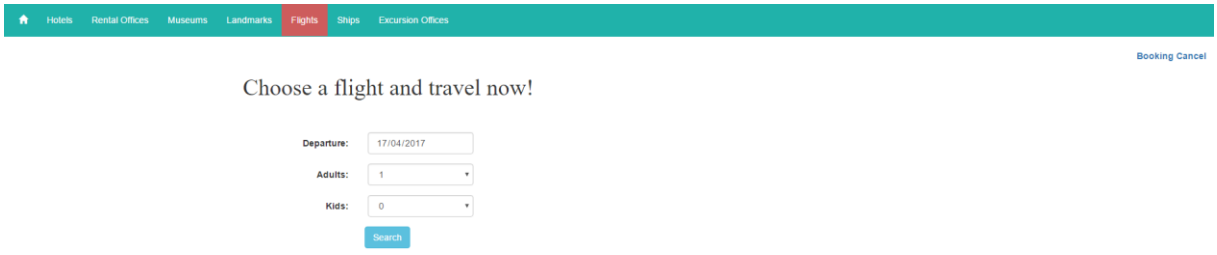
Αναζήτηση Αξιοθέατων

Στο JSP αρχείο *Landmarks.jsp* εμφανίζονται όλες οι πληροφορίες για τα διαθέσιμα αξιοθέατα της περιοχής.



Αναζήτηση Πτήσεων

Το σύστημα επιτρέπει στον πελάτη να κάνει αναζήτηση για τις διαθέσιμες πτήσεις την ημερομηνία που επιθυμεί και επιλέγοντας τα άτομα που πρόκειται να ταξιδέψουν. Για την εργασία αυτή χρησιμοποιείται το αρχείο *search.jsp*

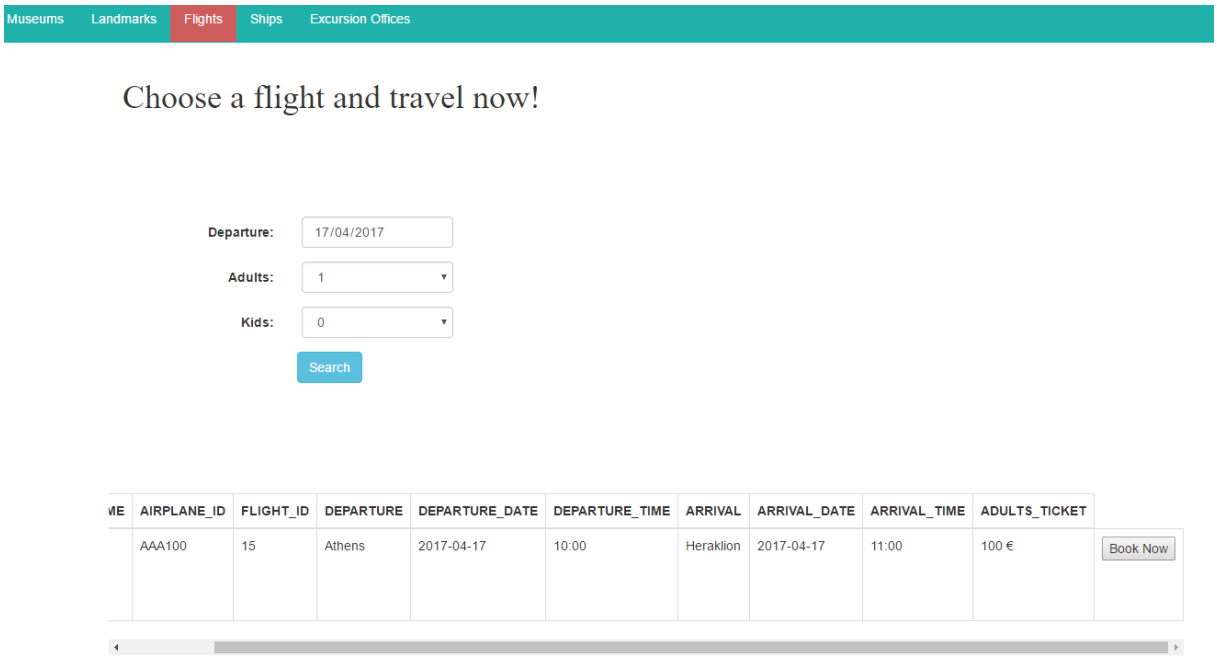


```
String departure = request.getParameter("departure");
String arrival = request.getParameter("arrival");
int adults = Integer.parseInt(request.getParameter("adults"));
String kid = request.getParameter("kids");

int kids = 0;

try {
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    query = "select * from flight_routes where departure_date=?";

    ps = conn.prepareStatement(query);
    ps.setString(1, departure);
    rs = ps.executeQuery();

    if (rs.isBeforeFirst()) {

        metaData = rs.getMetaData();
        columnCnt = metaData.getColumnCount();

        while (rs.next()) {

            int adultsTicket = adults * rs.getInt("Adults_Ticket");
            int kidsTicket = 0;

            if (kid != null) {
                kids = Integer.parseInt(kid);
                kidsTicket = kids * rs.getInt("Childrens_Ticket");
            }
```

Εικόνα 45 - Flight Search Code

Κρατήσεις Πτήσεων

Η διαδικασία της κράτησης είναι η ίδια με αυτήν που περιγράφηκε την ενότητα Κράτησης των δωματίων. Για την εργασία αυτή χρησιμοποιείται το servlet *FlightsThanksPage.java*



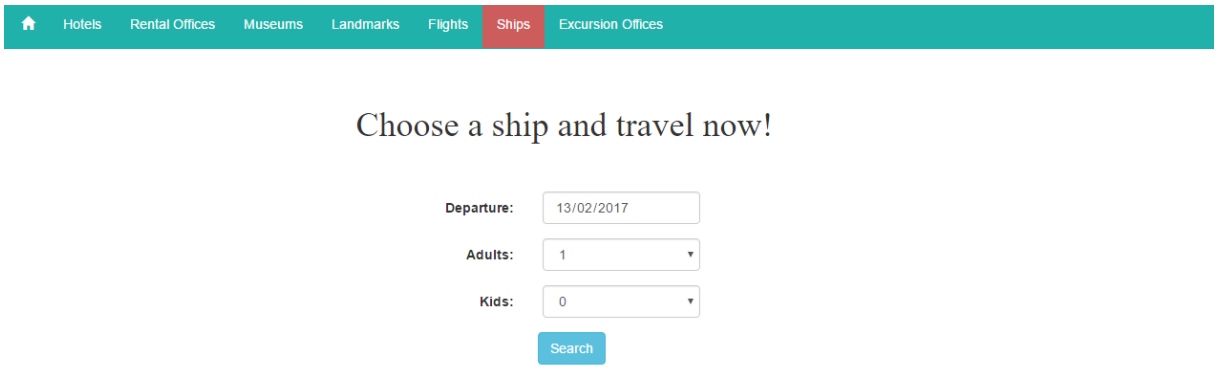
Εικόνα 46 - Flight Booking

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    pst = conn.prepareStatement("insert into Flight_Bookings values (?,?,?,?,?,?,?,?,?,?)");

    pst.setString(1, null);
    pst.setString(2, company_name);
    pst.setString(3, airplane_id);
    pst.setString(4, flight_id);
    pst.setInt(5, adults_ticket);
    pst.setInt(6, childrens_ticket);
    pst.setString(7, name);
    pst.setString(8, surname);
    pst.setString(9, phone);
    pst.setString(10, email);
    result = pst.executeUpdate();

    if (result > 0){

        stm = conn.createStatement();
        rs = stm.executeQuery("select max(Booking_ID) from Flight_Bookings");
        rs.next();

        writer.println("<script type=\"text/javascript\"> if(confirm('Success! Your Booking id is: " + rs.getString("max(Booking_ID)")
                + ", back to home Page') == true) window.location='http://localhost:8080/Thesis_User/index.html'; </script>");

    }

    pst.close();
    stm.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
```

Εικόνα 47 - FlightsThanksPage.java

Αναζήτηση Πλοίων

Το σύστημα επιτρέπει στον πελάτη να κάνει αναζήτηση για τα διαθέσιμα δρομολόγια πλοίων την ημερομηνία που επιθυμεί και επιλέγοντας τα άτομα που πρόκειται να ταξιδέψουν. Για την εργασία αυτή χρησιμοποιείται το αρχείο *search.jsp*



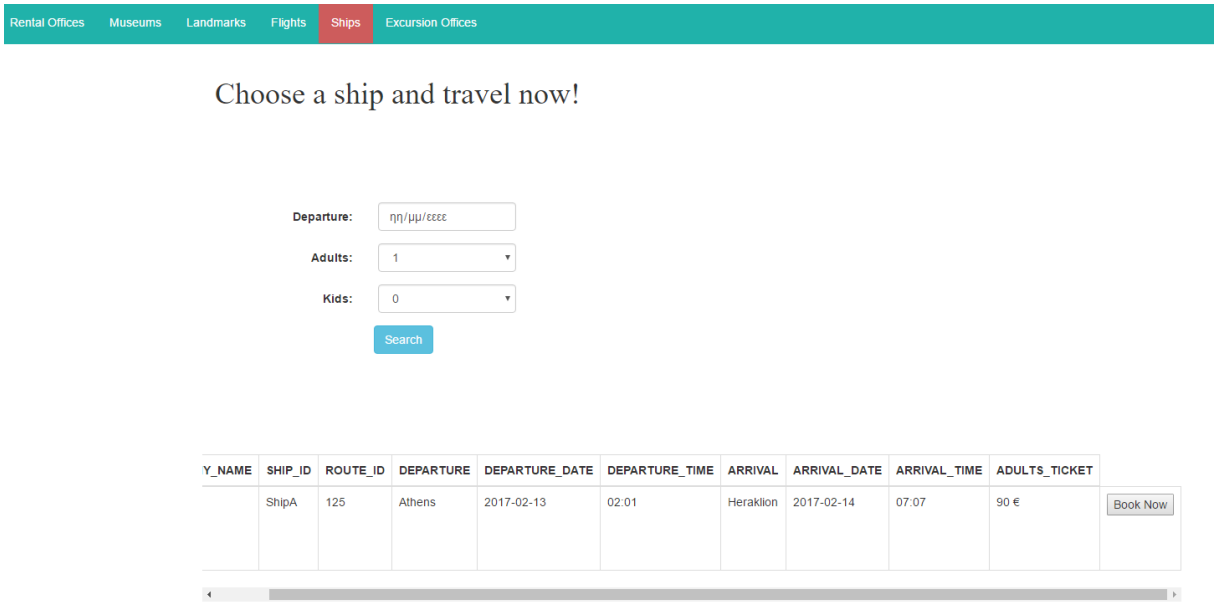
Εικόνα 48 - Ship Search

```
String departure = request.getParameter("departure");
int adults = Integer.parseInt(request.getParameter("adults"));
String kid = request.getParameter("kids");

int kids = 0;

try {
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    query = "select * from ship_routes where departure_date=?";

    ps = conn.prepareStatement(query);
    ps.setString(1, departure);
    rs = ps.executeQuery();

    if (rs.isBeforeFirst()) {

        metaData = rs.getMetaData();
        columnCnt = metaData.getColumnCount();

        while (rs.next()) {

            int adultsTicket = adults * rs.getInt("Adults_Ticket");
            int kidsTicket = 0;

            if (kid != null) {
                kids = Integer.parseInt(kid);
                kidsTicket = kids * rs.getInt("Childrens_Ticket");
            }
```

Εικόνα 49 - Ship search.jsp

Κρατήσεις Δρομολογίων Πλοίων

Η διαδικασία της κράτησης είναι η ίδια με αυτήν που περιεγράφηκε την ενότητα Κράτησης των δωματίων. Για την εργασία αυτή χρησιμοποιείται το servlet *ShipsThanksPage.java*



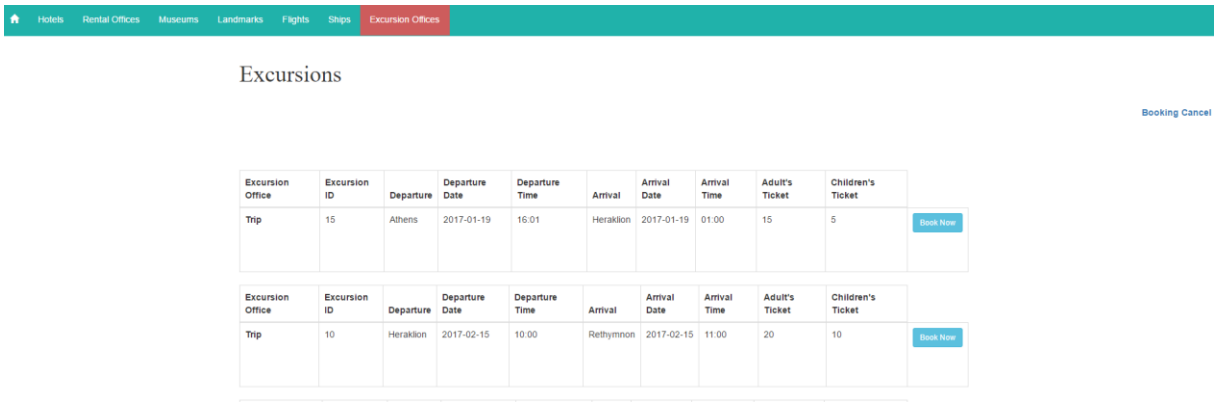
Εικόνα 50 - Ship Booking

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    pst = conn.prepareStatement("insert into Ship_Bookings values (?,?,?,?,?,?,?,?,?,?)");

    pst.setString(1, null);
    pst.setString(2, company_name);
    pst.setString(3, ship_id);
    pst.setString(4, route_id);
    pst.setInt(5, adults_ticket);
    pst.setInt(6, childrens_ticket);
    pst.setString(7, name);
    pst.setString(8, surname);
    pst.setString(9, phone);
    pst.setString(10, email);
    result = pst.executeUpdate();

    if (result > 0){

        stm = conn.createStatement();
        rs = stm.executeQuery("select max(Booking_ID) from Ship_Bookings");
        rs.next();

        writer.println("<script type=\"text/javascript\"> if(confirm('Success! Your Booking id is: " + rs.getString("max(Booking_ID)")
                + ", back to home Page') == true) window.location='http://localhost:8080/Thesis_User/index.html'; </script>");

    }

    pst.close();
    stm.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
```

Εικόνα 51 - ShipsThanksPage.java

Αναζήτηση Εκδρομών

Το σύστημα παρέχει στον πελάτη πληροφορία για τις διαθέσιμες εκδρομές στις οποίες μπορεί να δηλώσει συμμετοχή επιλέγοντας και πόσα άτομα επιθυμεί να δηλώσει. Για την εργασία αυτή χρησιμοποιείται το αρχείο *excursions.jsp*



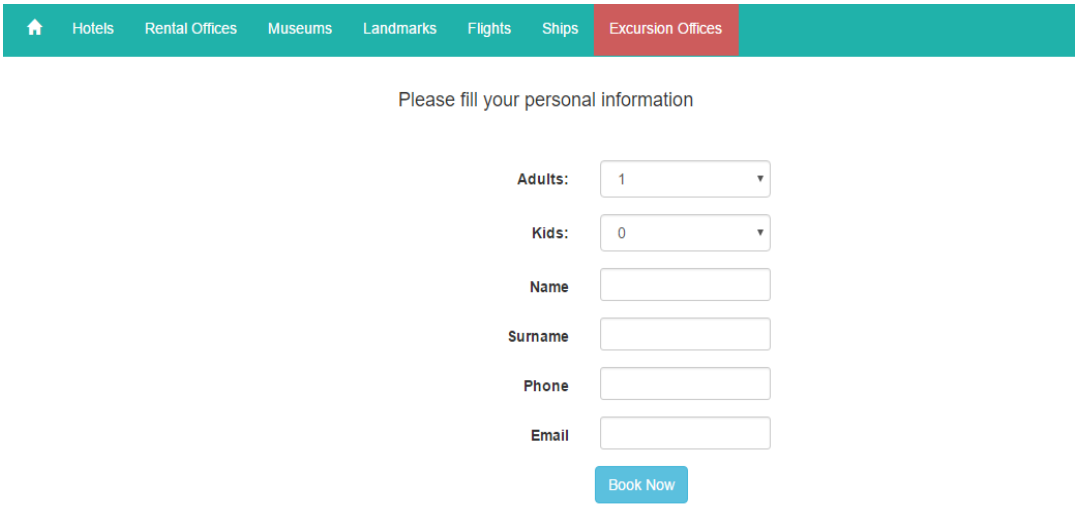
Εικόνα 52 - excursions.jsp

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");

    String query = "";
    Connection conn = null;
    Statement stm = null;
    ResultSet rs = null;
    ResultSetMetaData metaData = null;
    int columnCnt = 0;
    String[] header = null;
    int k = 0;
    int rowCnt = 0;
    String[] rowValues = null;

    query = "select * from Excursions ";

    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");
    stm = conn.createStatement();
    rs = stm.executeQuery(query);
    metaData = rs.getMetaData();
    columnCnt = metaData.getColumnCount();

    while (rs.next()) {
        rowCnt = rs.getRow();
```

Εικόνα 53 - Excursions jsp code

Κρατήσεις Εκδρομών

Ο πελάτης δηλώνει πλήθος ενηλίκων, πλήθος παιδιών και συμπληρώνει τα στοιχεία του. Για την εργασία αυτή χρησιμοποιείται το servlet *ExcursionsThanksPage.java.*



Εικόνα 54 - Excursions Booking

```
try{
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","root");

    ps = conn.prepareStatement("insert into excursion_bookings values (?,?,?,?,?,?,?,?,?)");

    ps.setString(1 , null);
    ps.setString(2, excOffice);
    ps.setString(3, excId);
    ps.setString(4, adults);
    ps.setString(5, kids);
    ps.setString(6, name);
    ps.setString(7, surname);
    ps.setString(8, phone);
    ps.setString(9, email);
    result = ps.executeUpdate();

    if (result > 0){
        stm = conn.createStatement();
        rs = stm.executeQuery("select max(Booking_id) from excursion_bookings");
        rs.next();

        writer.println("<script type=\"text/javascript\"> if(confirm('Success! Your booking id is " + rs.getString("max(Booking_ID)") +
                ", back to home Page') == true) window.location='http://localhost:8080/Thesis_User/index.html'; </script>");
    }

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

Εικόνα 55 - ExcursionsThanksPage.java

Ακυρώσεις Κρατήσεων

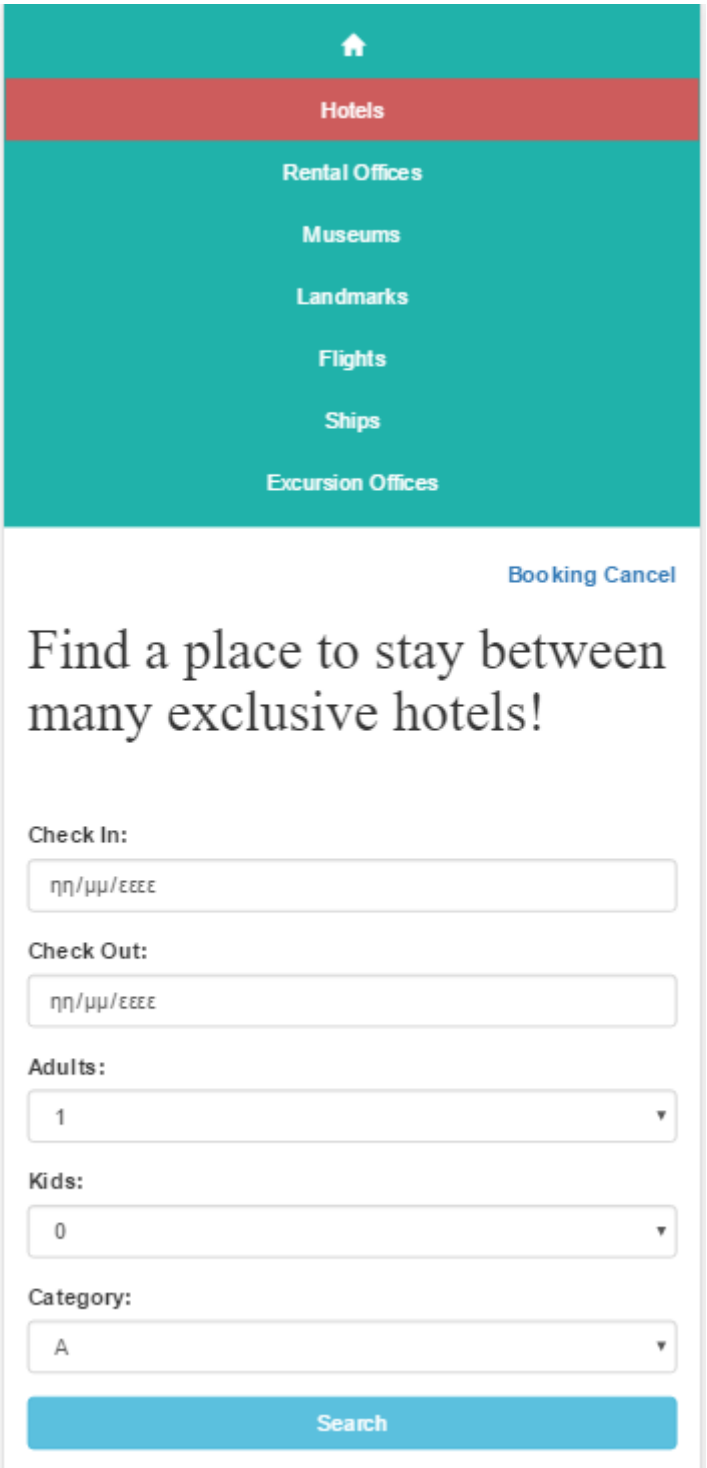
Όσον αφορά τις ακυρώσεις κρατήσεων για τις ενοικιάσεις, τις πτήσεις, τα πλοία και τις εκδρομές ακολουθείται η ίδια διαδικασία που περιγράφηκε στην ενότητα Ακύρωση Κράτησης ξενοδοχείου. Για την εργασία αυτή χρησιμοποιείται το servlet *CancelBooking.java*

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "root");

    if (hotel != null) {

        pst = conn.prepareStatement("delete from Hotel_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    } else if (car != null) {

        pst = conn.prepareStatement("delete from Car_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    } else if (motor != null) {

        pst = conn.prepareStatement("delete from Motor_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    } else if (flight != null) {

        pst = conn.prepareStatement("delete from Flight_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    } else if (ship != null) {

        pst = conn.prepareStatement("delete from Ship_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    } else if (excursion != null) {

        pst = conn.prepareStatement("delete from Excursion_Bookings where booking_id=?");

        pst.setString(1, bookingId);
        result = pst.executeUpdate();

    }
```

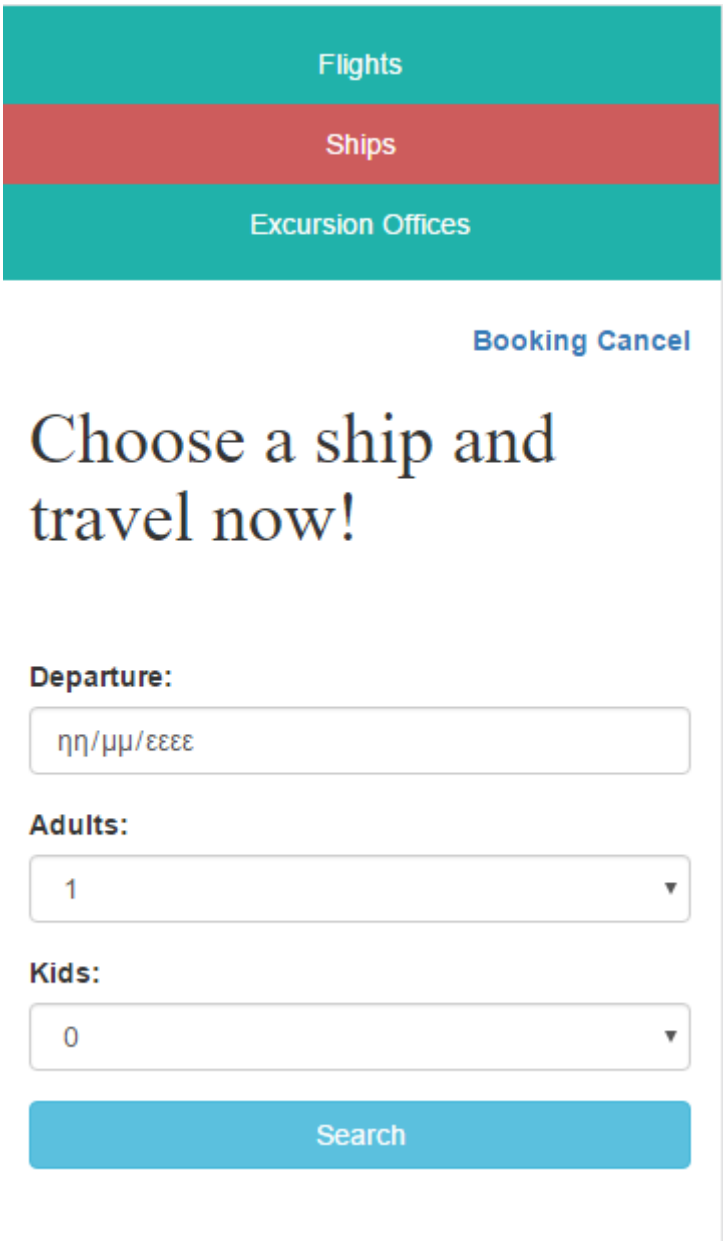
Εικόνα 56 - CancelBooking.java

Σε κινητές συσκευές

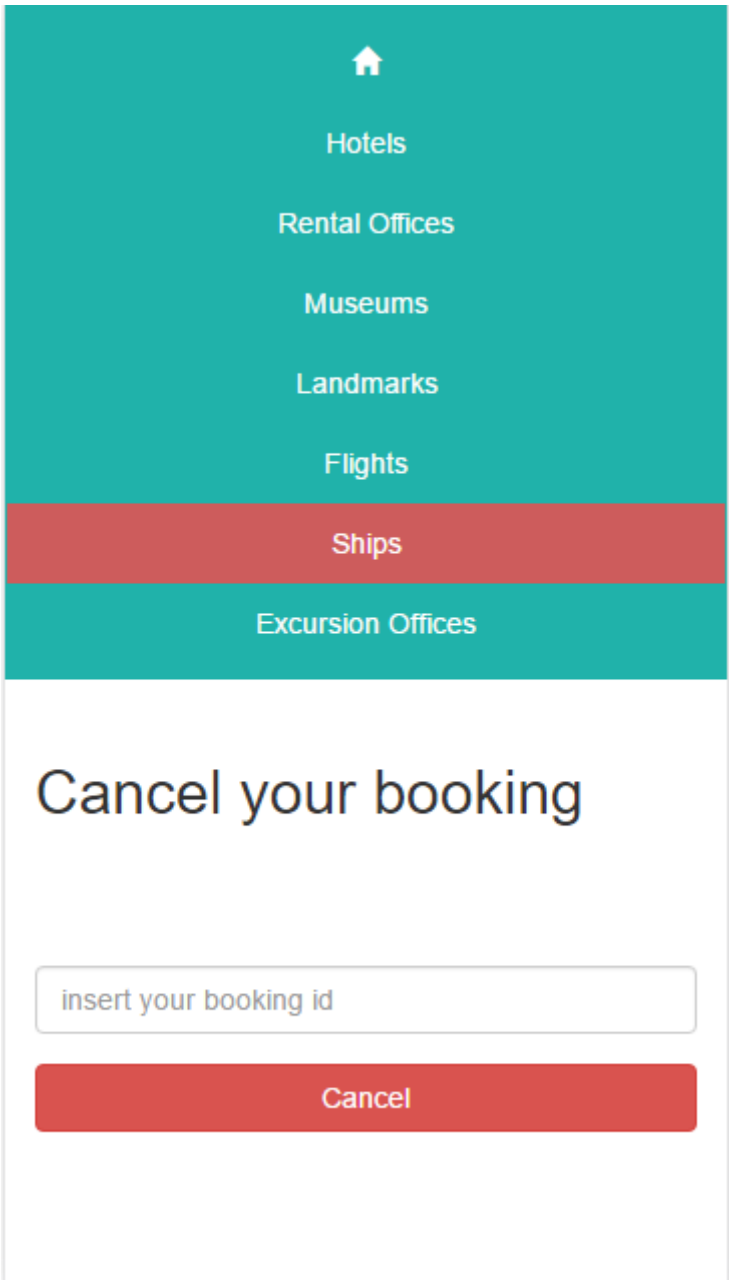
Παρακάτω παρουσιάζονται μερικά παραδείγματα του συστήματος σε κινητές συσκευές



Εικόνα 57 - Mobile 1



Εικόνα 58 - Mobile 2



Εικόνα 59 - Mobile 3

4 Αποτελέσματα

4.1 Συμπεράσματα

Σκοπός της εργασίας ήταν η ανάπτυξη των δυνατοτήτων μας και η διεύρυνση των γνώσεων μας. Με το πέρας της εργασίας αυτής έχουμε κατανοήσει εις βάθος τις τεχνολογίες που χρησιμοποιήθηκαν κι έχουμε θέσει γερά θεμέλια στις γνώσεις μας για να βγούμε στην αγορά εργασίας. Επιπρόσθετα μάθαμε να εργαζόμαστε μέσα στα πλαίσια ενός συγκεκριμένου χρονικού διαστήματος και να φέρουμε εις πέρας τις απαιτήσεις μιας εργασίας.

4.2 Μελλοντική Εργασία και Επεκτάσεις

Όσον αφορά την εργασία θα μπορούσαν να γίνουν κάποιες επεκτάσεις και βελτιώσεις. Σε επίπεδο διαχειριστή θα ήταν επιθυμητό να υπάρχει δυνατότητα αναζήτησης μιας εγγραφής, για παράδειγμα αναζήτηση ενός δωματίου βάσει του αριθμού του. Επίσης θα ήταν χρήσιμο να υπάρχει η δυνατότητα ταξινόμησης των στηλών των πινάκων, για παράδειγμα κατά αύξουσα ημερομηνία. Όσον αφορά το περιεχόμενο που εισάγει ο διαχειριστής θα ήταν επιθυμητό να μπορεί να εισάγει και πολυμεσικό περιεχόμενο όπως εικόνες και βίντεο.

Σε επίπεδο πελάτη θα ήταν χρήσιμο το σύστημα να εμφανίζει εναλλακτικές λύσεις εάν δεν υπάρχουν αποτελέσματα σε αυτά που ζητάει ο χρήστης. Επιπρόσθετα θα μπορούσε να κάνει προτάσεις όπως νέες εκδρομές ή νέα ξενοδοχεία και να υπάρχει δυνατότητα αξιολόγησης από τη μεριά του πελάτη. Τέλος θα ήταν χρήσιμο να υπάρχει δυνατότητα πληρωμής μέσω χρεωστικών καρτών και με τη χρήση τρίτων υπηρεσιών όπως για παράδειγμα το PayPal.

Βιβλιογραφία

[1] M. H. -. L. Brown, σε *Servlets και σελίδες διακομιστή Java*, Κλειδάριθμος.

[2] «oracle,» [Ηλεκτρονικό]. Available: https://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm#i1028247. [Πρόσβαση 30 Μάρτιος 2017].

[3] «wikipedia,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/HTML. [Πρόσβαση 1 Απρίλιος 2017].

[4] «wikipedia,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/HTML5. [Πρόσβαση 1 Απρίλιος 2017].

[5] «tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/html5/html5_overview.htm.

[6] «wikipedia,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/CSS. [Πρόσβαση 1 Απρίλιος 2017].

[7] «w3schools,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/css/css_intro.asp. [Πρόσβαση 1 Απρίλιος 2017].

[8] «w3schools,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/css/css_syntax.asp. [Πρόσβαση 1 Απρίλιος 2017].

[9] «w3wchools,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/css/css_howto.asp. [Πρόσβαση 1 Απρίλιος 2017].

[10] «wikipedia,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/Bootstrap. [Πρόσβαση 2 Απρίλιος 2017].

[11] «w3schools,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Πρόσβαση 2 Απρίλιος 2017].

[12] «getbootstrap,» [Ηλεκτρονικό]. Available: http://getbootstrap.com/. [Πρόσβαση 2 Απρίλιος 2017].

[13] «getbootstrap,» [Ηλεκτρονικό]. Available: http://getbootstrap.com/components/. [Πρόσβαση 2 Απρίλιος 2017].

[14] «eclass2.teicrete.gr, Προχωρημένα θέματα βάσεων δεδομένων, Ακουμιανάκης Δημοσθένης».

Παράρτημα

Περίληψη

Σκοπός της πτυχιακής αυτής ήταν η μελέτη, η ανάλυση, η σχεδίαση και η υλοποίηση ενός τουριστικού οδηγού για ένα συγκεκριμένο μέρος. Υλοποιήθηκε ένα ολοκληρωμένο σύστημα το οποίο αποτελείται από δύο υποσυστήματα, το back-end το οποίο αφορά μόνο τον διαχειριστή και το front-end το οποίο αφορά τον τελικό χρήστη. Στο back-end γίνεται εισαγωγή όλων των απαραίτητων πληροφοριών από τον διαχειριστή και καταχωρούνται στη βάση δεδομένων ενώ στο front-end όλες αυτές οι πληροφορίες θα παρουσιάζονται με δυναμικό τρόπο στο χρήστη ανάλογα με το τι ζητάει. Το σύστημα αυτό θα παρέχει πληροφορίες για ξενοδοχεία, γραφεία ενοικιάσεων μεταφορικών μέσων, μουσεία, αξιοθέατα, δρομολόγια πτήσεων και πλοίων καθώς και για εκδρομές. Ο χρήστης θα είναι σε θέση να κάνει κρατήσεις και ακυρώσεις αυτών μέσω του διαδικτύου.

Βασική υποδομή για την υλοποίηση του συστήματος είναι η τεχνολογία Java. Πιο συγκεκριμένα χρησιμοποιήθηκαν τεχνολογίες servlet και jsp. Άλλες τεχνολογίες - γλώσσες που χρησιμοποιήθηκαν είναι η oracle sql για τη βάση δεδομένων, η HTML 5, η CSS και τέλος για το σύστημα front-end το framework Bootstrap. Για όλες τις τεχνολογίες έγινε σύντομη αναφορά και παρουσίαση των δυνατοτήτων και σύγκριση με άλλες τεχνολογίες.

Όσον αφορά και τα δύο υποσυστήματα έγινε ανάλυση όλων των λειτουργιών και δυνατοτήτων τους με δείγματα κώδικα και εικόνων. Πραγματοποιήθηκε ανάλυση των απαιτήσεων του συστήματος, σχεδιασμός υλοποίησης και παρουσίαση της τελικής υλοποίησης. Επιπροσθέτως στην τελευταία ενότητα της εργασίας έγινε αναφορά στις πιθανές βελτιώσεις του συστήματος σε επίπεδο διαχειριστή και επίπεδο πελάτη.

Το τελικό αποτέλεσμα της εργασίας είναι η κατανόηση εις βάθος της δημιουργίας ενός πολύπλοκου συστήματος και η εκμάθηση νέων τεχνολογιών καθώς και η αποκόμιση περεταίρω γνώσεων με στόχο μια δυναμική παρουσία στον κλάδο της πληροφορικής και στην αναζήτηση εργασίας.