Documentation - Group Kelde

Philip Tibom, 891025-5093, philiptibom@gmail.com
Hossein Hussain, 941001-3859, hossein.hussain@me.com
Daniel Olsson, 8804227158, daniel.dsc.olsson@gmail.com

March 14, 2016

1 General overview of the system

This project is a web application with the intent to work as a Travel Agency.

The website will offer packaged travels, where the user gets to individually search cities, hotels and flights to the chosen destination. And allow the user create their own packaged travel.

The idea is to enter some search criteria. Such as destination city and dates. The website will present available hotels on these dates. Once chosen a hotel, flights for the chosen dates will be presented. Where the user may select both outbound and return flight individually. Then at last place an order into the system.

There users will not have to login to the website. As this is not practical on a travel website. However there is a login system for the Administrators. Where the admins can edit, delete and add more or less anything into the database.

2 Use cases

Here is a list of **88** fully functional use cases.

- Search view: Enter From-City name. Get immediate AJAX suggestions. Also immediate AJAX validation.
- Search view: Enter To-City name. Get immediate AJAX suggestions. Also immediate AJAX validation.
- Search view: Enter Number of passengers. Get immediate AJAX validation.
- Search view: Enter From-Date by using a calendar. Only Future dates will work. Get immediate AJAX validation.

- Search view: Enter To-Date by using a calendar. Only Future dates will work. Get immediate AJAX validation.
- Search view: Press the search button. Validates all input again. Then redirects to search for hotels.
- Hotel view: Visit this page from URL by using parameters. And thus can be shared or bookmarked.
- Hotel view: Click on a hotel to get more details.
- Hotel view: Push select to choose the hotel. Redirect to Flights view.
- Flights view: Visit this page from URL by using parameters. And thus can be shared or bookmarked.
- Flights view: Select a outbound flight. AJAX loads in a list of return flights.
- Flights view: Select a return flight. Redirects to next page.
- AdminLogin view: Enter username.
- AdminLogin view: Enter password.
- AdminLogin view: Press login. Redirects to admin panel.
- AdminLogin view: Press Logout. Invalidates session and redirects to login page.
- Admin Panel: Press City. Redirects to a list of cities.
- Admin Panel: Press Plane. Redirects to a list of planes.
- Admin Panel: Press Hotel. Redirects to a list of hotels.
- Admin Panel: Press Booking. Redirects to a list of bookings.
- Admin City view: Select edit on an item from the list view. Redirects to edit page.
- Admin City view: Select Delete on an item from the list view. Redirects to delete page.
- Admin City view: Press Add City. Redirects to create a city, page.
- Admin City view: Press "Back to admin menu". Redirects back to admin menu.
- Admin Add City: Insert the city name.
- Admin Add City: Press Save. Saves into database and redirects back to city view.
- Admin Add City: Press Cancel. Cancels the add and redirects back to city view.

- Admin Edit City: Change the city name.
- Admin Edit City: Press Save. Saves into database and redirects back to city view.
- Admin Edit City: Press Cancel. Cancels the edit and redirects back to city view.
- Admin Delete City: Press delete, to confirm deletion from database.
- Admin Plane view: Select edit on an item from the list view. Redirects to edit page.
- Admin Plane view: Select Delete on an item from the list view. Redirects to delete page.
- Admin Plane view: Press plane. Redirects to create a plane, page.
- Admin Plane view: Press "Back to admin menu". Redirects back to admin menu.
- Admin Add Plane: Insert plane type.
- Admin Add Plane: Insert plane capacity.
- Admin Add Plane: Press Add plane. To save into database.
- Admin Add Plane. Press cancel. To cancel the add and redirect back to plane view.
- Admin Edit Plane: Change the plane name.
- Admin Edit Plane: Change the plane capacity.
- Admin Edit Plane: Press Save. Saves into database and redirects back to plane view.
- Admin Edit Plane: Press Cancel. Cancels the edit and redirects back to plane view.
- Admin Delete Plane: Press delete, to confirm deletion from database.
- Admin Hotel view: Select edit on an item from the list view. Redirects to edit page.
- Admin Hotel view: Select Delete on an item from the list view. Redirects to delete page.
- Admin Hotel view: Press add hotel. Redirects to create a hotel, page.
- Admin Hotel view: Press "Back to admin menu". Redirects back to admin menu.
- Admin Add Hotel: Insert hotel name.
- Admin Add Hotel: Insert price.

- Admin Add Hotel: Insert rooms.
- Admin Add Hotel: Insert rating.
- Admin Add Hotel: Insert city (id).
- Admin Add Hotel: Insert description.
- Admin Add Hotel: Press Add Hotel. To save into database.
- Admin Add Hotel. Press cancel. To cancel the add and redirect back to hotel view.
- Admin Edit Hotel: Change hotel name.
- Admin Edit Hotel: Change price.
- Admin Edit Hotel: Change rooms.
- Admin Edit Hotel: Change rating.
- Admin Edit Hotel: Change city (id).
- Admin Edit Hotel: Change description.
- Admin Edit Hotel: Press Edit Hotel. To save into database.
- Admin Edit Hotel. Press cancel. To cancel the edit and redirect back to hotel view.
- Admin Delete Hotel: Press delete, to confirm deletion from database.
- Admin Booking view: Select edit on an item from the list view. Redirects to edit page.
- Admin Booking view: Select Delete on an item from the list view. Redirects to delete page.
- Admin Booking view: Press Add booking. Redirects to create a booking, page.
- Admin Booking view: Press "Back to admin menu". Redirects back to admin menu.
- Admin Add Booking: Insert destination city id.
- Admin Add Booking: Insert departure city id.
- Admin Add Booking: Insert flight "to" id.
- Admin Add Booking: Insert flight "back" id.
- Admin Add Booking: Insert fly to date.
- Admin Add Booking: Insert fly from date.
- Admin Add Booking: Insert hotel id.
- Admin Add Booking: Insert total price.

- Admin Add Booking: Press Add Booking. To save into database.
- Admin Add Booking. Press cancel. To cancel the edit and redirect back to booking view.
- Admin Edit Booking: Change destination city id.
- Admin Edit Booking: Change departure city id.
- Admin Edit Booking: Change flight "to" id.
- Admin Edit Booking: Change flight "back" id.
- Admin Edit Booking: Change fly to date.
- Admin Edit Booking: Change fly from date.
- Admin Edit Booking: Change hotel id.
- Admin Edit Booking: Change total price.
- Admin Edit Booking: Press Save Booking. To save into database.
- Admin Edit Booking. Press cancel. To cancel the edit and redirect back to booking view.

3 Model

We started by creating this simplified model. Which is the base for our whole application.

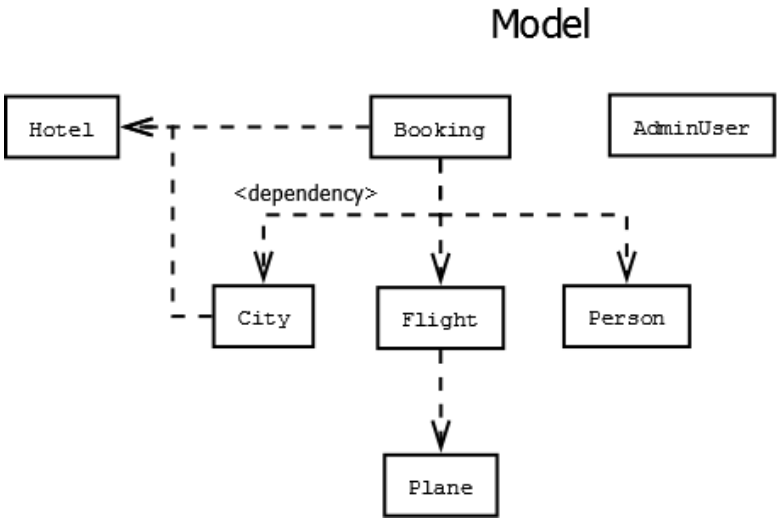

Figure 1: Simplified UML to describe the model

Then we went on to model the database as an ER-diagram. And then we created the classes to satisfy the Many-to-One and Many-to-Many relationships.

4 Architecture / Design

4.1 General

The application is built on JSF 2.2. Together with two component frameworks which works very well together. PrimeFaces and BootsFaces.

BootsFaces is a library that incorporates the HTML and CSS framework Bootstrap. Which allows us to focus more on the technical aspect of the application rather than the visual aspect. BootsFaces can seamlessly be combined with PrimeFaces. So we have choosen to use the input components from PrimeFaces as they provide rich functionality. And the layout components from BootsFaces as this reduces our need for adding HTML and CSS manually.

The architecture of our app follows the MVC pattern. Where the JSF pages serves as the view, as well as the backing beans in the view package. The views makes use of the JSF templating system and its various components.

The model is the database. We then inject DAOs (Data Access Objects) to fetch information from the database into the view. Or in the controllers case to store information into the database. To accomplish this the DAOs use JPA as the glue layer between the database and the application.

The controller in JSF is generally handled automatically. But in order to prevent the view from modifying the model, we have created a controller which does this job.

The persistence layer cosists of GenericDAOs which then specific DAOs inherits and add special functionality. For example our flight needs to search for a date and therefor needs a special query. While finding a specific flight by id is generic and is inherited by using generic types.

The only circular dependencies that we have is for the persistence model in situations where @ManyToOne and @ManyToMany are needed. This is unfortunately something that is required when using a relational database and JPA.

4.2 User experience

Where page reloads are needed we are using the PRG pattern. To prevent issues with double posting. It is also possible to use the back and forward buttons in the browser without any issues.

For validating user input we have chosen to make use of the Beans validations. As it gathers all validations in one spot. Rather than using JSF validations. To enhance the user experience we have chosen to display each validation message instantly by using AJAX. The user input is always validated on the server before inserted to our database.

When searching for cities on our main page, we are searching our database each key press and returning a list of suggestions (auto completion) to the user, again by using AJAX.

While searching for hotels and flights the URL will get updated along each step. So that the user may save the URL for later or share it with somebody else. And so the search can be returned to at any time.

While viewing hotels and flights, the user may chose to sort by name, price, departure and such. The data is then updated instantly with AJAX.