

Technical University of Cluj-Napoca

Software Design

Laboratory – Assignment 1

Travel Agency Documentation



Laboratory Assistant:

Teodora Vezan

Student:

Enoiu Diana-Cristina

1. Assignment objective

To design and implement a system for a travelling agency. The client can book vacation packages from the agency and the administrator can manage the vacation packages. The system has two types of users that can log in using a email and a password: administrator and vacay seeker/client.

The Travelling Agency should be able to:

1. add vacation destination
2. add vacation packages for a specific destination that should contain information: name, price, period, extra details, number of people that can book the vacation
3. edit an existing vacation package
4. delete an existing vacation package
5. view all its listed vacation packages(with status: BOOKED,NOT_BOOKED,IN_PROGRESS)
6. delete vacation destination

The Regular User should be able to:

1. register on the platform using some credentials(username/email-unique & password)
2. login on the platform
- 3.view all available vacation packages
- 4.filter vacation packages by destination/price/period
- 5.book a vacation
- 6.view all its booked vacations

2. Analysis

a) Problem analysis

This application should be able to fulfil all the requirements from assignment objective. The performing of these operations will be done with the help of a GUI. It will have two main windows for the admin and clients, and another window to login/signup in the application. The data, about users, vacation packages, destinations and bookings, will be stored in a database with the help of Hibernate.

b) Modeling

All the data about the vacation packages and destinations, users and bookings will be placed in a database and we will use Hibernate to make this process faster. We will have as entities/model classes User, VacationPackage, VacationDestination, Booking and we will also use some enum classes for Status and UserType. Using the layered architecture the controller will call each time it needs to retrieve data the Service classes, that implement the business logic in the application. The service classes will further call the repository classes to instantiate the Entity Manager to retrieve or insert data into the database. We will use an interface to declare the basic operations that are performed on the database like find, select, delete etc.

Different scenarios and use cases

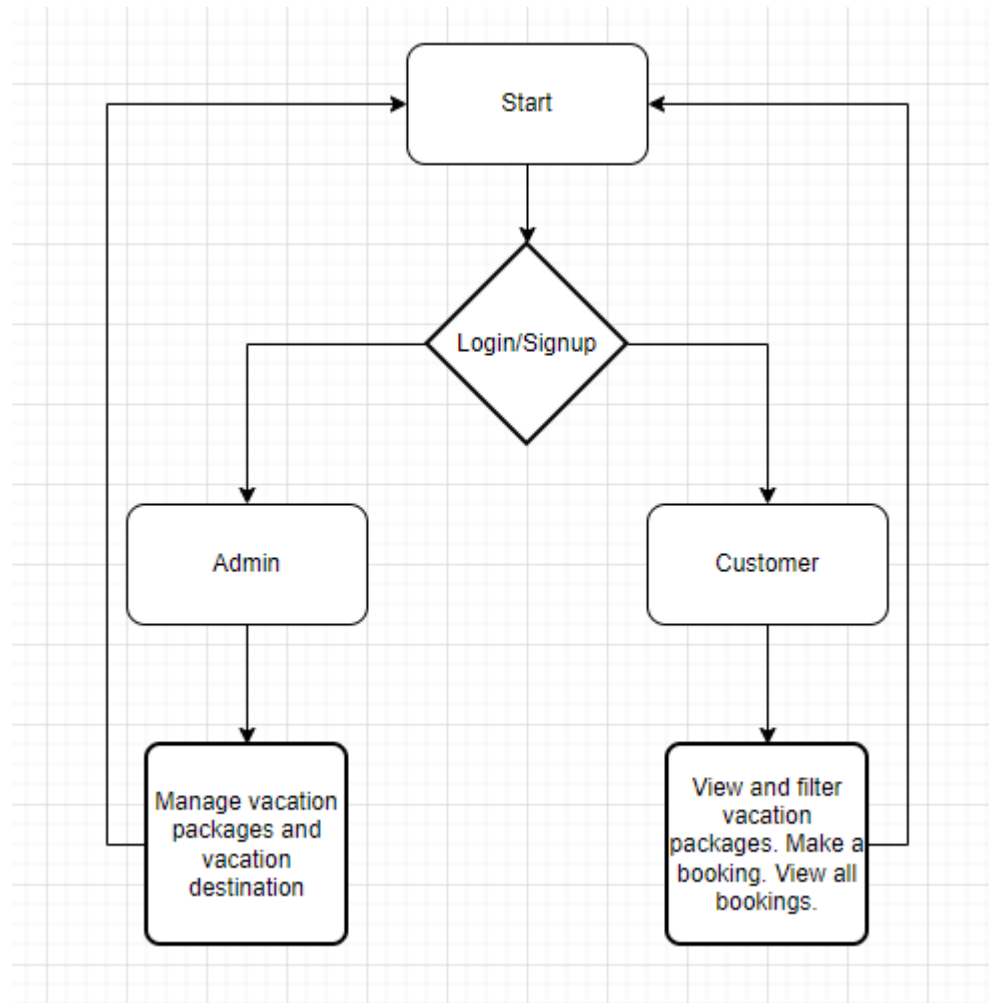
A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

The use cases are correlated with the steps the user makes when using the application.

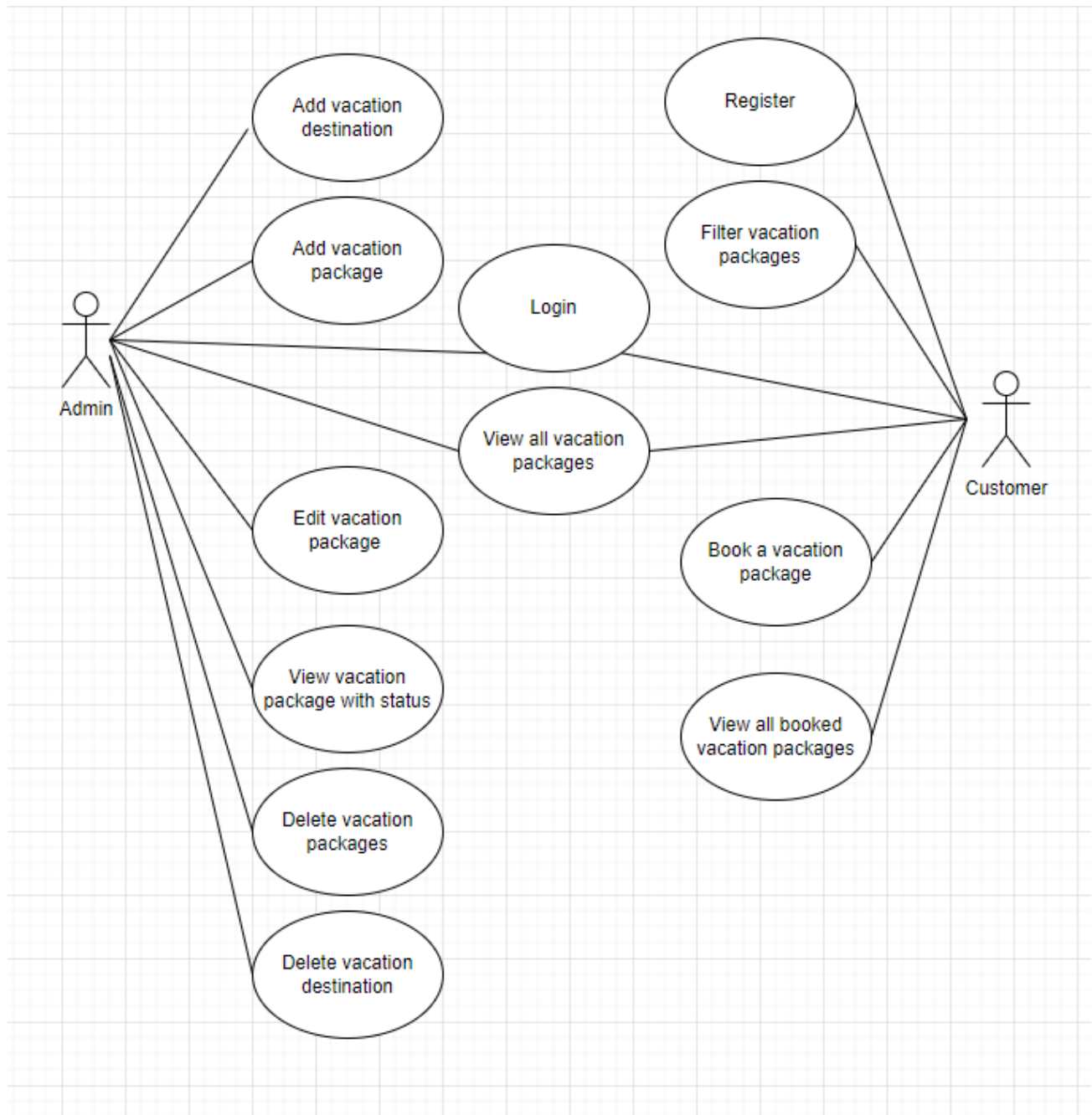
The application should have a user friendly interface in order for the user to understand faster what are the steps that are needed to be performed in order to achieve the desired result.

For this application, the flow-chart from below is describing the way the application is expected to behave, based on the user inputs.

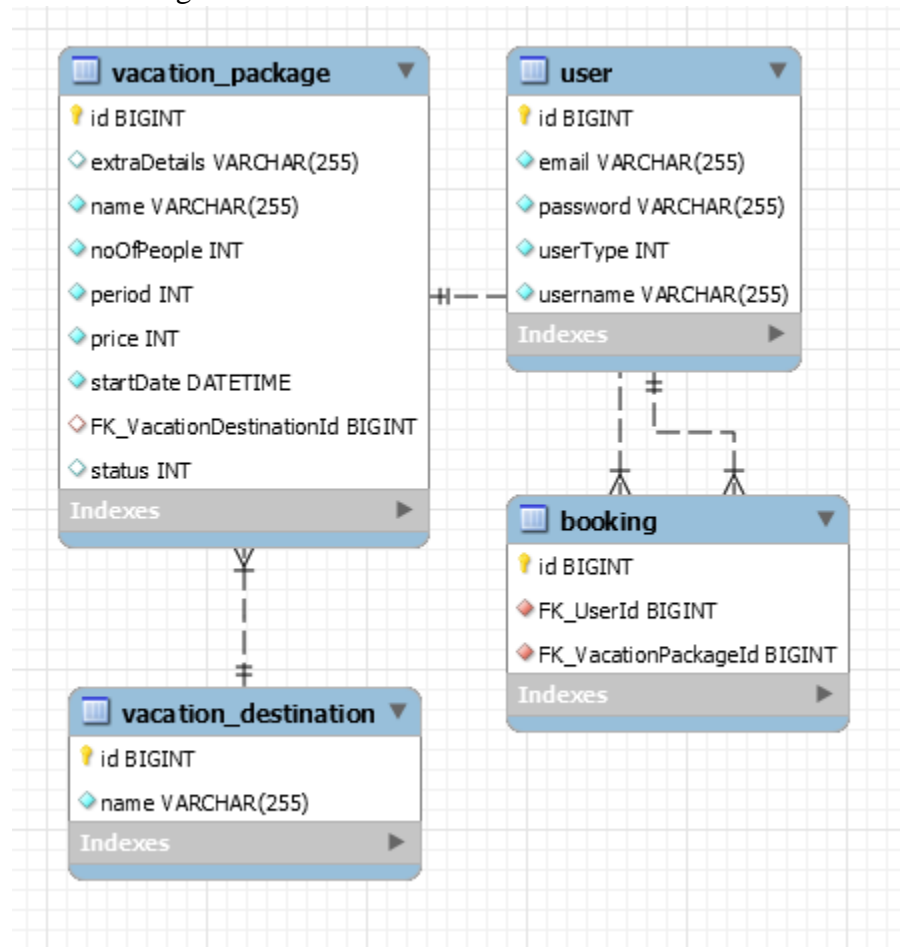
a. Flowchart



b. Use cases



c. Database diagram



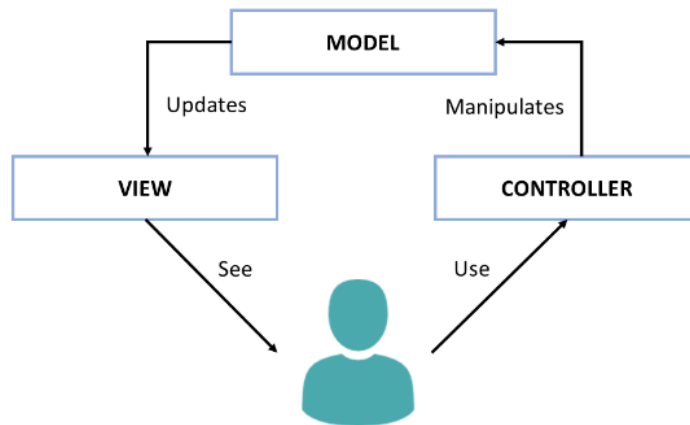
3. Design and Implementation

a) *Design decisions*

This application uses an object-oriented programming design. The application has a graphical user interface made in JavaFX, with the help of Scene Builder.

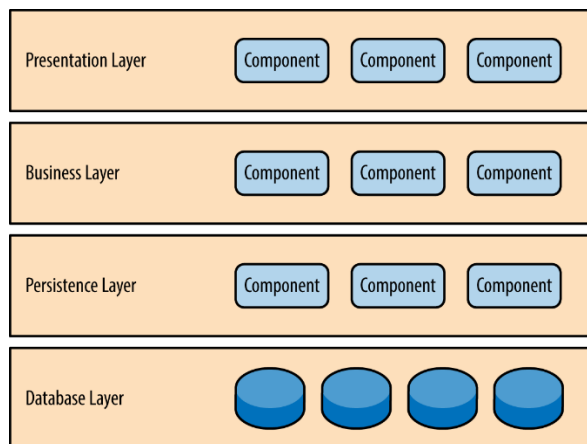
For the design of this application it was used, also, an architectural pattern, the MVC pattern, (Model-View-Controller pattern), which is used to separate application's concerns.

The Model-View-Controller is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.



This application also uses the most common architecture pattern which is the layered architecture.

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Although the layered architecture pattern does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of four standard layers: presentation, business, persistence, and database.



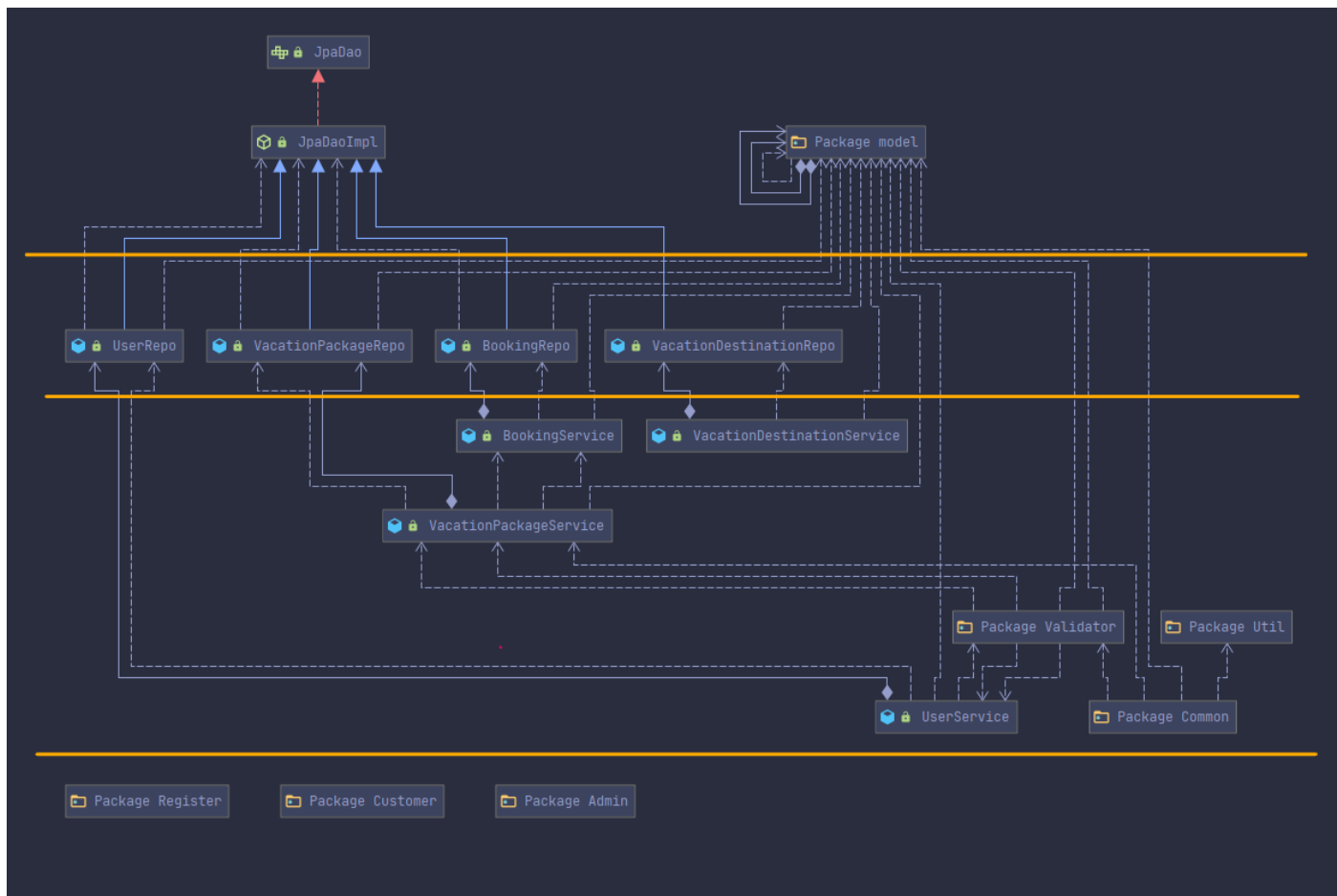
Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn't need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn't need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data (e.g., calculate values or aggregate data), and pass that information up to the presentation layer.

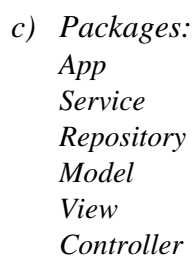
Example of a layered architecture:

This application' classes have, thus, been split into seven main packages:

- The application package which contains the Main class;
- The model package used in the modelling of the program;
- The view package containing .fxml, .css files used for displaying the actual interface;
- The controller package, which manages the view and model packages, contains the FXMMLControllers classes to manage the .fxml files in the View package.
- The presentation layer holds the view and controller packages
- In the Data layer we will have the repository classes used to extract the data from the database
- The Business Logic package which will encapsulate the application logic

b) UML class diagram





App

Service

Repository

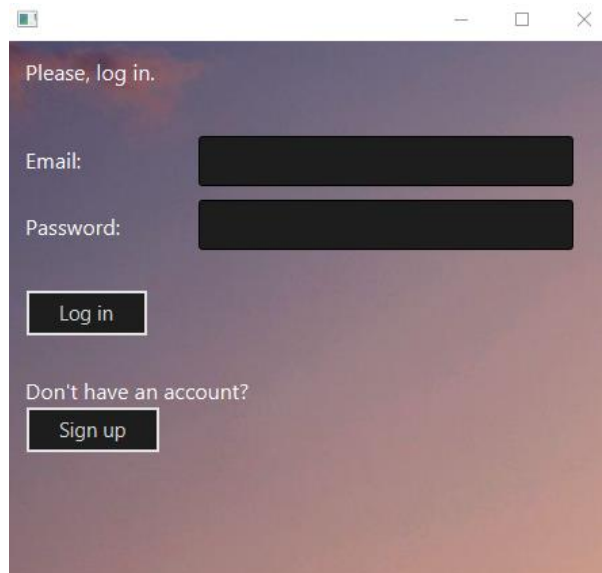
Model

View

Controller

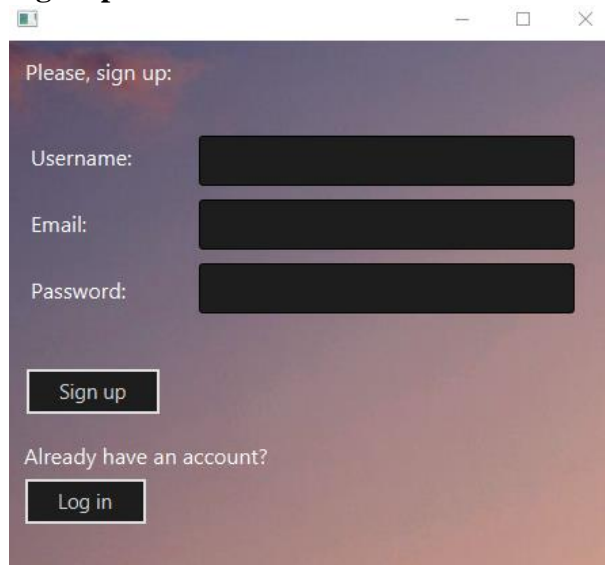
d) User interface

Log in



A screenshot of a 'Log in' window. The window has a title bar with a small icon and standard minimize, maximize, and close buttons. The background is a dark purple-to-pink gradient. The text 'Please, log in.' is at the top left. Below it are two input fields: 'Email:' and 'Password:'. Each field has a black rectangular placeholder. Below the password field is a 'Log in' button. At the bottom, the text 'Don't have an account?' is followed by a 'Sign up' button.

Sign Up



A screenshot of a 'Sign Up' window. The window has a title bar with a small icon and standard minimize, maximize, and close buttons. The background is a dark purple-to-pink gradient. The text 'Please, sign up:' is at the top left. Below it are three input fields: 'Username:', 'Email:', and 'Password:'. Each field has a black rectangular placeholder. Below the password field is a 'Sign up' button. At the bottom, the text 'Already have an account?' is followed by a 'Log in' button.

The administrator interface

Vacation Packages

Vacation Destination

Name	Destination	Price	Start Date	Period	Limit	Details
Pack 1	Maldiv	200	Jun 11, 2015	24	10	Nope
Pack 6	Maldiv	12	Jan 12, 2023	12	12	None
Pack 8	Maldiv	12	Jan 12, 2023	12	12	None
Best pack	Chile	100	Jan 12, 2024	12	12	None
Pack13	Israel	123	Jan 12, 2024	23	20	None
Tenerife pack 1	Tenerife	200	Feb 1, 2023	23	23	None

Show vacation packages

Delete vacation package

View status

Add a new vacation package

Select destination:

Name

Price

Start date(zz/mm/yyyy)

Period (in days)

Limit (no of people)

Add vacation package

Back

Vacation Packages

Vacation Destination

Add a new vacation destination

Add destination

Select a vacation destination:

Maldiv

Delete destination

The vacation packages associated with the destination

Name	Destination	Price	Start Date	Period	Limit	Details
Pack 1	Maldiv	200	Jun 11, 2015	24	10	Nope
Pack 6	Maldiv	12	Jan 12, 2023	12	12	None
Pack 8	Maldiv	12	Jan 12, 2023	12	12	None

Back

The Client Interface

Vacation Packages

Bookings

Choose the vacation packages that you want to book

Name	Destination	Price	Start Date	Period	Limit	Details
Pack 1	Maldiv	200	Jun 11, 2015	24	10	Nope
Pack 6	Maldiv	12	Jan 12, 2023	12	12	None
Pack 8	Maldiv	12	Jan 12, 2023	12	12	None
Best pack	Chile	100	Jan 12, 2024	12	12	None
Pack13	Israel	123	Jan 12, 2024	23	20	None
Tenerife pack 1	Tenerife	200	Feb 1, 2023	23	23	None

Show all vacation packages

or, show vacation packages by destination

Add vacation package to booking list

Destination

Price

Period

Search by criteria

Packages for booking

Total Price: 112

Pack 8

Best pack

Remove vacation package from list

Perform Booking

Back

Vacation Packages

Bookings

Name	Destination	Price	Start Date	Period	Limit	Details
Pack 1	Maldiv	200	Jun 11, 2015	24	10	Nope
Pack 6	Maldiv	12	Jan 12, 2023	12	12	None
Tenerife pack 1	Tenerife	200	Feb 1, 2023	23	23	None
Japan pack 1	Japan	300	Mar 3, 2023	23	2	None
Pack 8	Maldiv	12	Jan 12, 2023	12	12	None
Best pack	Chile	100	Jan 12, 2024	12	12	None

Show all bookings

Back

4. Conclusions

This assignment was also an introduction in building a project with Gradle and Hibernate, and it helped me familiarize with the building of a user interface with JavaFX.

The using of the MVC architectural was also important for me to understand because it is the basic structure which most web applications, mobile apps and desktop programs are built on.

The layered architecture was also used together with the MVC pattern because this will allow us to add new features, or change the current features more easily. Adding a new use case to the system, or extend the business rules on a particular domain object would have been much harder if the process or business logic is spread throughout the code.

5. Bibliography

<http://stackoverflow.com>

<https://openjfx.io/openjfx-docs/#gradle>

<https://ro.wikipedia.org/wiki/Model-view-controller>

<https://app.diagrams.net/>

Software Design– Lectures of prof. Mihaela Dinsoreanu