**Project Report**

**Fly High – Airline Management System**

**Created by:**

|  |  |
| --- | --- |
| **Cristina Ailoaei (266543)** | **Dragoș Sîrbu (266500)** |
|  |  |
| **Michał Jurewicz (266892)** | **Michał Podgórni (267128)** |
|  |  |

**Supervisors:**

**Ib Havn**

**Jens Cramer Alkjærsig**

**Mona Wendel Andersen**

**Steffen Vissing Andersen**

**[Name and logo of educational institution]**

**[Logo of companies included]**

**[Number of characters]**

**Information and Communication Technology** **Engineering**

**2nd Semester**

**IT-SEP2Y-A17**

**Group I**

**07.07.2018**

****

**Table of content**

[List of figures and tables v](#_Toc515741505)

[Executive summary vi](#_Toc515741506)

[Abstract vii](#_Toc515741507)

[1 Introduction 1](#_Toc515741508)

[2 Theory/literature survey 2](#_Toc515741509)

[3 Methods 3](#_Toc515741510)

[3.1 Analysis 4](#_Toc515741511)

[3.1.1 Requirements 4](#_Toc515741512)

[3.1.2 Use case diagram 6](#_Toc515741513)

[3.1.3 Use case descriptions 7](#_Toc515741514)

[3.1.4 Activity diagrams 8](#_Toc515741515)

[3.1.5 Analysis class diagram 9](#_Toc515741516)

[3.2 Design 9](#_Toc515741517)

[3.2.1 Design class diagram 9](#_Toc515741518)

[3.2.2 TCP connection diagram 9](#_Toc515741519)

[3.2.3 Sequence diagrams 9](#_Toc515741520)

[3.2.4 GUI design 9](#_Toc515741521)

[3.3 Implementation 9](#_Toc515741522)

[3.4 Testing 9](#_Toc515741523)

[3.4.1 Test cases 9](#_Toc515741524)

[3.4.2 JUnit testing 11](#_Toc515741525)

[4 Results/findings and Discussion 11](#_Toc515741526)

[4.1 Results 11](#_Toc515741527)

[4.2 Discussion 11](#_Toc515741528)

[5 Conclusions 12](#_Toc515741529)

[6 Sources of information 13](#_Toc515741530)

Appendices

# List of figures and tables

Figure 1 – Use Case Diagram x

Figure 2 – Book a flight use case description x

Figure 3 – Book a flight activity diagram x

Figure 4 - x

Figure 5 - x

Figure 6 - x

Figure 7 - x

Figure 8 - x

Figure 9 - x

Figure 10 - x

Figure 11 - x

Figure 12 - x

Figure 13 - x

Figure 14 - x

Figure 15 - x

# Executive summary

The purpose of the project was to create a java application with a database connection that uses client-server architecture for Fly High Airline. The application was supposed to manage and store data about flights, airports, airplanes, crew members and club members. This study analyses the process of developing the project step by step. Its goal is to create a better way of understanding how java programs work and to cache and maintain information in a more efficient and modern way.

# Abstract

The use of flights, both in terms of travelling and airfreight, has greatly increased over the past decades, air travel becoming one of the most important modes of transport nowadays. Dating back to 1903 with the first flight of two American engineering brothers, the air transport is and has demonstrated itself to be a thriving market which doubles its volume of passengers every 15 years. Fly High is an airline company that brought the request to the team to build a system which could help them storing everything in a well-organized way and allow the customers to use their services at once.

The unified process was used in order to complete the project, meaning that more parts of the project were done concurrently, but with different intensity.

At first, in the inception phase, was the one where the project group made most of the business modelling, set the goals and assured that the customer will accept the result of the work. Then, in elaboration phase the group focused more on the analysis and design parts. During the construction phase there was created the majority of the implementation. At the end, in the transition phase the group assured that everything meets the requirements, tested the application and tried to solve the minor issues.

It can be concluded that the final version of the project is functional and meets the set goals to a big extent.

# Introduction

Air traffic is a big and essential branch of transport sector nowadays. Although air travel can be named a relatively new option of transport, this market has already achieved a lot, promising even much more than that. Studies show that until 2036, the traffic flow in Europe will expand 2.6 times than in present. (AIRBUS S.A.S., 2017) The reasons of its enhanced popularity are mainly supported by the title of the fastest form of transport and a good safety record for commercial air transport

One specific Danish airline, Fly High, founded by Tobias Jensen and William Christensen in 2009 is a company headquartered in Vejle, which initially operated domestic flights. Now, the airline wants to expand its flights across Europe, which brought the request for a new management system. While operating internal flights, the only way for booking a ticket for a specific flight was calling the company and discussing with an employee all the details of the flight, starting with destination and ending with check-in, the ticket being sent afterwards via email. Due to the difficulty in booking tickets it is not surprising that “Fly High” has been left behind and it is not very popular. Therefore, expanding across Europe in the lack of a better management system would be totally ineffective, as the current state of things implies that both the clients and employees have to put a lot of effort into it. If there is not a more convenient way of purchasing the flight, most of the people would be very much tempted to fly with other companies. In the context of today’s society, something that does not put in use the available technological possibilities, it will not achieve the biggest possible success. Regarding “Fly High”, once the amount of data started to increase in size (new flights, new planes, change in the crew), operations such as storing and managing company’s information became very complicated to be handled on files. A simple scenario of a client who wants to change the date of a flight could create serious problems in such a system. And things can become even more dangerous, as the slightest issue can lead to cancellations and delays, risking the future of the company.

# Theory/literature survey

For creating the project, the group had to develop a proper java application together with all documentation. In order to do so, it used 3 sources of knowledge, which are following:

* Lectures and exercises at school
* Books:
  + Craig Larman – “Applying UML and patterns”
  + Thomas Connolly – “Database systems”
* Webpages:
  + Java 8 Documentation
  + Database Documentation
  + Stack Overflow website

Owing to the oral presentations in class, the project group acknowledged both the technical and business planning issues. Knowledge from Software Development with UML and Java 2 course allowed to implement the most of application, Database System course showed how to create the database, which is a necessary part of the project, while Software Engineering course helped with the documenting and assuring that everything is logical and straight. For organizing the work and using adequate way of developing the project, Semester Project course was the crucial one.

If it comes to books, both of them have very wide range of content, therefore the project group has not used everything that could be learnt from them but only the most important parts. They were especially helpful for including the design patterns in the java application and for the most complicated parts of the database.

The internet sites were the place, where the project group could check how something works and find help with understanding some parts. Every time there was a minor issue that was not discussed in any of the books, the group used Java Documentation, Stack Overflow or Database Documentation websites. The online research allowed to complete all the gaps in the process of creating the project.

# Methods

Because of the requirements of the project were quite specific from the beginning, the project team could know very early what the exact tasks will be and how the work should be divided. The system had to be implemented in Java and use the client-server architecture. The group members started working with IntelliJ IDEA, because creating the Graphical User Interface (GUI) was more efficient this way. the team decided to use JavaFx package, owing to its simple and efficient work style.

For organizing the work, the group used the Unified Process which assumes that more than one step of developing the project can be done at the same time. However, it divides all the process into four main phases which are following:

1. Inception
2. Elaboration
3. Construction
4. Transition

After creating the *Requirements (3.1.1)* and making sure they are setting the correct goals, the group started creating the *Use Case diagram (3.1.2)* together with the *Use Case descriptions (3.1.3)*, which are the explanation of each use case. A graphical way of showing the behavior of the system was covered by the *Activity diagrams (3.1.4)*. The exact classes together with their methods that explain the operational part of the program were illustrated on *Analysis class diagram (3.1.5)* while the *Design class diagram (3.2.1)* gives an overview on the whole application. Connection between client and server side is presented on the *TCP connection diagram (3.2.2).* The *Sequence diagrams (3.2.3)* explain how objects operate with the other ones in the most complex methods. The graphical representation of all the system functionalities is shown on the *GUI Design part (3.2.4).* All possible scenarios of the application’s behavior are discussed and shown in the *Test cases (3.4.1).*

## Analysis

### Requirements

#### Functional requirements

1. An administrator should be able to add airports to the system. While adding a new airport, the administrator has to specify the code, name, city, postcode, country, number of gates.
2. An administrator should be able to add airplanes to the system. While adding a new plane, the administrator has to specify the number, model, number of seats.
3. An administrator should be able to add crew members to the system. While adding a new crew member, the administrator has to specify the name, position, address, birthdate, id, phone number, e-mail.
4. An administrator should be able to add flights to the system. While adding a new flight, the administrator has to specify the number, departure time, arrival time, departure place, arrival place, plane, crew, price.
5. A customer booking a flight should specify all of the following: name, birthdate, nationality, type of ID, ID number, expiration date.
6. An administrator should be able to delete data from the system.
7. A head administrator should be able to cancel flights.
8. An administrator should be able to change data for club members, crew, flights, airplanes and airports.
9. An administrator should be able to select date/time range for flights in order to get flights in a specified range.
10. An administrator should be able to select cities for flights in order to get flights from/to the specified cities.
11. A customer should be able to choose a seat number, luggage size, payment method in order to book a ticket.
12. A customer should be able to select departure and destination airport and the departure and return date (or departure only) for flights in order to get the available flights.
13. An administrator should be able to get a list of all flights and club members.
14. An administrator should be able to set the annual fee for club members.
15. A customer should receive the ticket via email.
16. A customer should be able to become a club member in order to get discounts.
17. A club member should be able to search only for cheap flights from his/her city.
18. A customer should be able to subscribe to the newsletter in order to receive new information regarding flights and offers via email.
19. An administrator should be able to log in the system in order to manage data.
20. A head administrator should be able to see the profiles of all administrators.
21. A head administrator should be able to create or delete an administrator account in order to ease the management of accounts.

#### Non-functional requirements

1. The system has to use the client-server architecture.
2. The system has to store persistent data using a database.
3. The system has to have a GUI.
4. The system has to provide a log in.
5. The system has to be implemented in Java.
6. The system and the system development process have to be documented.

### Use case diagram

Figure 1 - Use case diagram

The use case shown above (Figure 1) presents all functional feature that every user of  
the FlyHigh application can perform. The use cases are following:

* **Add an element** – The administrator or the head administrator can add an airplane, airport, crew member or flight into the system.
* **Find an element** – The administrator or the head administrator can search for an existing airplane, airport, crew member or flight in the system.
* **Edit an element** – The administrator or the head administrator can edit the data of an airplane, airport, crew member or flight.
* **Delete an element** – The head administrator can delete an airplane, airport or crew member from the system.
* **Cancel a flight** – The head administrator can cancel a flight.
* **Find a flight** – The customer can search for an existing flight in the system.
* **Book a flight** – The customer can book an existing flight.

### Use case descriptions

|  |  |
| --- | --- |
| UseCase | Book a flight |
| Summary | A customer books a flight |
| Actor | Customer |
| Precondition | None. |
| Postcondition | The flight becomes booked, the changes are stores in the database. |
| Base Sequence | 1. The person goes through find a flight use case.  2. The person enters all the required personal data: name, birthdate, nationality, type of ID, ID number, expiration date, seat number, size of luggage, method of payment.  3. The person confirms the decision to book the given flight.  4. If one or more of the entered data is not valid then go to step 2 else the decision is confirmed and the given flight becomes booked, changes are stored in the database, person is redirected to another site in order to make a payment and the use case ends. |
| Branch Sequence |  |
| Exception Sequence | The entered data could not be valid:  4 as base sequence  The system informs that the entered data is not valid |
| Sub UseCase | Find a flight |
| Note |  |

Figure 2 - Book a flight use case description

### Activity diagrams

Figure 3 - Booking a flight activity diagram

One of the crucial functionalities, booking a flight, works as follows:

First the customer initiates an option to search for a flight.

Then all the necessary data (departure place, arrival place, departure time, arrival time) is entered.

If the system cannot find a flight with given details, it displays an appropriate message and asks the customer to fill the new data.

If the system can find a flight with given details, it is displayed for the customer.

In the last step the customer is asked to fill all his personal details (name, birthdate, nationality, type of id, id number, id expiration date) and flight details (seat number, luggage size, payment method).

If any of the entered data is not correct, the system displays an appropriate message and asks to fill the new data.

If all the entered data is correct, the system makes a reservation which is stored in the system and the customer is informed about it. The flow ends.

### Analysis class diagram

## Design

### Design class diagram

### TCP connection diagram

### Sequence diagrams

### GUI design

## Implementation

## Testing

### Test cases

In order to get a complete list of all functionalities of the system together with all possible combinations of events that can go in a different way than they were meant, the project team developed a list of test cases that show it. They are based on the requirements and use cases of the system and their main purpose is to make sure that any possible scenario will not be forgotten. Owing to it the system should not have any situations it would not know how to handle.



Figure 4 - Book a flight test case

### JUnit testing

# Results/findings and Discussion

## Results

## Discussion

# Conclusions

# Sources of information

**Appendices**