Process Report

Airline Reservation System

**Heterogeneous System**

**Names of Students:**

Jan Vasilcenko – 293098

Karrtiigehyen Veerappa - 293076

Nicolas Popal - 279190

Patrik Horny – 293112

**Supervisors:**

Jakob Knop Rasmussen

Ole Ildsgaard Hougaard

**VIA University College**

****

**Number of characters:** including spaces

**Software Technology Engineering**

**Semester 3**

**27 November 2020**

Table of Contents

[1 Introduction (Karrtiigehyen and Nicolas Popal) 4](#_Toc58330514)

[2 Group Description (Karrtiigehyen and Nicolas Popal) 5](#_Toc58330515)

[2.1 Power Distance (Karrtiigehyen and Nicolas Popal) 6](#_Toc58330516)

[2.2 Individualism/Collectivism (Karrtiigehyen and Nicolas Popal) 6](#_Toc58330517)

[2.3 Masculinity/Femininity and Indulgence/Restraint (Karrtiigehyen and Nicolas Popal) 6](#_Toc58330518)

[2.4 Long/Short Term Orientation (Karrtiigehyen and Nicolas Popal) 7](#_Toc58330519)

[2.5 Uncertainty Avoidance (Karrtiigehyen and Nicolas Popal) 7](#_Toc58330520)

[3 Project Initiation (Karrtiigehyen and Nicolas Popal) 8](#_Toc58330521)

[3.1 Selection of Topic (Karrtiigehyen and Nicolas Popal) 8](#_Toc58330522)

[3.2 Formation of Group (Karrtiigehyen and Nicolas Popal) 9](#_Toc58330523)

[4 Project Description (Karrtiigehyen and Nicolas Popal) 10](#_Toc58330524)

[5 Project Execution (Karrtiigehyen, Nicolas Popal and Patrik Horny) 12](#_Toc58330525)

[5.1 Project Development and Methods (Karrtiigehyen, Nicolas Popal and Patrik Horny) 12](#_Toc58330526)

[5.2 SCRUM (Karrtiigehyen, Nicolas Popal and Patrik Horny) 12](#_Toc58330527)

[5.2.1 Product Backlog (Patrik Horny) 13](#_Toc58330528)

[5.2.2 Sprint Planning (Patrik Horny) 14](#_Toc58330529)

[5.2.3 Sprint Backlog (Patrik Horny) 15](#_Toc58330530)

[5.2.4 Burndown Chart (Patrik Horny) 15](#_Toc58330531)

[5.2.5 Sprint Review and Retrospective (Patrik Horny) 16](#_Toc58330532)

[5.3 GitHub (Patrik Horny) 17](#_Toc58330533)

[5.4 Blazor (Patrik Horny) 18](#_Toc58330534)

[5.5 Unified Process (Karrtiigehyen, Nicolas Popal and Patrik Horny) 19](#_Toc58330535)

[5.6 Critique to The Project (Jan Vasilcenko, Karrtiigehyen and Patrik Horny) 19](#_Toc58330536)

[5.6.1 General Assessment and Critique (Jan Vasilcenko, Karrtiigehyen and Patrik Horny) 19](#_Toc58330537)

[5.6.2 SOLID Principles (Jan Vasilcenko, Karrtiigehyen and Patrik Horny) 20](#_Toc58330538)

[5.6.3 DRY Rule (Jan Vasilcenko, Karrtiigehyen and Patrik Horny) 20](#_Toc58330539)

[6 Personal Reflections (Everybody) 22](#_Toc58330540)

[7 Supervision (Jan Vasilcenko, Karrtiigehyen and Patrik Horny) 23](#_Toc58330541)

[8 Sources of Information 24](#_Toc58330542)

**List of Figures**

[Figure 1: Hofstede’s Dimensions 5](#_Toc58330447)

[Figure 2: Some examples of product backlog 14](#_Toc58330448)

[Figure 3: An example of a sprint planning 15](#_Toc58330449)

[Figure 4: Burndown chart 16](#_Toc58330450)

[Figure 5: An example of sprint retrospective 17](#_Toc58330451)

[Figure 6: How the Github projects looked like 18](#_Toc58330452)

**List of Tables**

[Table 1: An early version of the Risk Assessment 11](#_Toc58330500)

# Introduction (Karrtiigehyen and Nicolas Popal)

We were introduced to the 2nd semester project on 2nd of September where we were given lots of information about how the process of making the project is going to look like. We discussed about the examples of distributed systems, along with skills and knowledge required to create a functional and well-structured project.

On 16th of September, we handed in our project proposal where we proposed to work on Hospital System, Food Tinder, and Airline Reservation System. Our supervisors gave us a feedback about our project proposal and after some discussion we decided that we will develop an airline reservation system. This seemed like the most reasonable option, since none of us had full grasp of a hospital system works, and the idea for the food tinder was half-baked.

During following weeks, we started laying a foundation for our project, where we defined the necessary functionalities and the capabilities, as well as the requirements. The Software Development of Distributed Systems (SDJ3) course provided us with knowledge on how to develop distributed systems. The Internet Technologies, C# and .Net (DNP1) course taught us how to work with C# and .Net, which we would have to use since the airline reservation system was a heterogenous system using C# and Java. The security aspects of the project were learned from the Networking and Security (NES1) course. Knowledge from the previous semesters such as SOLID principles, design patterns, and client/server systems were used extensively in this semester as well.

On the 11th of November, we had to hand in a Proof of Concept of our system, which greatly helped us to realize how to start to implement the distributed system using the three-tier architecture.

In the previous semester, SCRUM (Schwaber & Southerland, 2017) was taught to us, and this is the framework we used to structure our group collaboration. This is done to keep the group more engaged to work on the project. This framework made sure that we did not lose focus on the project and that our time-schedule will be used efficiently.

# Group Description (Karrtiigehyen and Nicolas Popal)

Our group consists of four people:

Jan Vasilcenko – Czech Republic

Karrtiigehyen Veerappa – Malaysia

Nicolas Popal – Czech Republic

Patrik Horny – Slovakia

Four members of the group already worked together during the first, second and third semester projects. Our group consists of 2 Czechs, 1 Slovak and 1 Malaysian. Even though the Czech-Slovaks, who are very close due to their historical and cultural background make most part of the group, there we no problems in between the group as we get along very well with each other. Even though Czech and Slovak nationalities are quite close, this is not reflected in the Hofstede’s Dimensions (as can be seen in the picture below).

Czech Republic, Malaysia, Slovakia

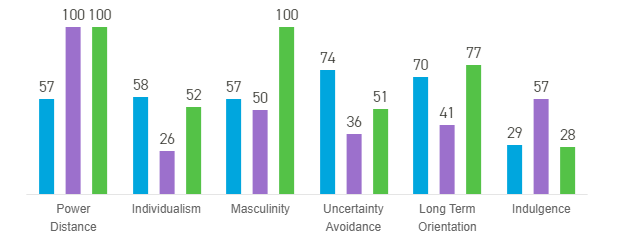


Figure 1: Hofstede’s Dimensions

## Power Distance (Karrtiigehyen and Nicolas Popal)

Power Distance measures inequalities between lower and higher ranked members of a certain hierarchy. (family elders vs. children, high management vs. employees) However, power distance is hard to measure inside of the project work as there are no big differences in hierarchy structure since all the members are equal with each other. Most probable scenario would be a person responsible for the project acting arrogant over others, distributing work without much contribution etc. However, nothing similar happened in our case, as we were all satisfied with our roles in our group.

## Individualism/Collectivism (Karrtiigehyen and Nicolas Popal)

Individualism and Collectivism in project work relates on how people usually work as part of the group. In our case, even though Czechs and Slovaks have mediocre individualism on Hofstede’s Scale, this was not our case as we all are quite social and work well as part of the group, tending more towards Collectivism.

## Masculinity/Femininity and Indulgence/Restraint (Karrtiigehyen and Nicolas Popal)

Masculinity and Femininity represents the values of the group and society. Masculine society strives for success, heroism, and assertiveness, while Feminine strives for cooperation, caring and satisfaction in their work-life balance. Indulgence values enjoying life and having fun, while restraint focuses on social norms and following social standards. In the case of our group, we had a great balance between work and free time, so, we tend to lean more towards Femininity and Indulgence, as we value our time more than a potential success, so the Hofstede’s Dimensions are not represented correctly, especially for the member from Slovakia in case of Femininity where Slovakia leans toward Masculinity, and Czechs and Slovaks in case of Indulgency, where both countries lean toward Restraint, as completely opposed to our group’s nature. This work-life balance is probably inherited from Denmark’s nature as Danes (and generally all Nordics) tend to value their work-life balance more than strive for success, so living in such environment might affect a person to some extent.

## Long/Short Term Orientation (Karrtiigehyen and Nicolas Popal)

Long/Short Term orientation displays if the members of the team are focused on smaller tasks rather than long-term ones. In our case, I would say inexperience causes us to be oriented specifically on smaller tasks, than the long-term ones, so at first glance, it might seem that the Hofstede’s Dimensions are not represented correctly. However, this might not be the real case.

## Uncertainty Avoidance (Karrtiigehyen and Nicolas Popal)

Uncertainty Avoidance measures the amount of precautions made to handle unexpected situations, adapting to them, and preventing them. In our case, we did not encounter many unexpected situations, so it is very hard to estimate whether we represent the Hofstede’s Dimensions correctly.

# Project Initiation (Karrtiigehyen and Nicolas Popal)

## Selection of Topic (Karrtiigehyen and Nicolas Popal)

When the time had come to choose the project topic, we had trouble with deciding what system we wanted to develop. There were some requirements for the project that we were about to plan and construct. For example, it had to be a heterogenous system, using a distributed architecture, and using sockets and web services. Other than those requirements, we were allowed to choose any topic that we wanted to work with.

After a long debate, three project proposals were selected, a Hospital System, Food Tinder, and Airline Reservation System. The Hospital System was a system for keeping medical records about patients. The Food Tinder was a system where people could meet by posting and liking pictures of food. The Airline Reservation System is a system where people can buy tickets for flights and also manage the flights. The Hospital System was very interesting for us all, but only a crude representation would have been made by us since none of us are experienced in how a hospital system functions. The Food Tinder was by far the most unique proposal, but it was a half-baked idea and none of us had a full realization of how the system would function. In the end, the Airline Reservation System was chosen because we felt like we understood how the an airline reservation system should function, given that we implemented a Library System where people can reserve and borrow books for SEP2. All of us have also bought tickets online through airline reservation systems before such as Wizz Air, giving us experience on how the interface for the user should be.

These conclusions proved to be true after consulting these proposals with our supervisors, who assured us that the Airline Reservation System is the best option out of the three, and it was consequently selected as our main topic for the third semester project.

## Formation of Group (Karrtiigehyen and Nicolas Popal)

All four of us have worked in the previous two semester projects together. In the previous semester project, there was a fifth group member, but we chose to work with only the four of us this semester as we felt like we could work very well within the four of us.

# Project Description (Karrtiigehyen and Nicolas Popal)

In this phase of the project, the problem domain was established, and ultimately the goal of the project was set. The goal of the project was to create an airline reservation system, where users can book flight tickets, and some users can manage the flights. This was a realistic goal, but some delimitations had to be set in order to not complicate the project too much. For example, one of the delimitations was that no type of payment or money system will be added to the application. This made it easier for us to implement the system without worrying the problems that can arise from implementing a transaction system, which in itself is a complex system to implement.

After entering initiation phase, we started by making a risk assessment table to pinpoint the riskiest scenarios that might occur during project development and to prevent them and how to react to them. Risk assessment table can be seen below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Cause** | **Prevention/Mitigation** | **Responsible person** | **Response** |
| Illness (Normal/Corona) | Medium | Medium/ High | Various/Corona epidemic | Healthy lifestyle/Self-prevention | Patrik Horný (SCRUM Master) | Redistributing work |
| Technical Failure | Low | High | Various | Cloud/Git backups, reliable equipment | Patrik Horný (SCRUM Master) | Replace malfunctioning equipment |
| Member Sabotage | Low | High | Loss of Motivation, Personal | Team-building, frequent progress checking | Patrik Horný (SCRUM Master) | Redistributing work, Supervisor meeting |
| Change in User Stories | High | Medium | Inexperience, Inadequate Analysis | Detailed analysis | Karrtiigehyen (Product Owner) | Implement changes |
| Work Overload | Low | Medium | Inexperience | Careful planning, workload distribution acording to skills and needs | Patrik Horný (SCRUM Master) | Redistributing work |
| Misunderstanding Customer | High | High | Inappropriate communication with the customer | Communication with customer | Karrtigehyen (Product Owner) | Changing the product according to customer's needs |

Table 1: An early version of the Risk Assessment

This is an early version of the risk assessment, where the risks were made to be broad and not system specific. But the risk assessment was later revised, which can be found in the Project Report subsubsection 2.5.2.

# Project Execution (Karrtiigehyen, Nicolas Popal and Patrik Horny)

Project execution went better than expected. This semester we tried to be more punctual on what needs to be done in time which helped us to finish the system sooner and focus on documentation and polishing the code and the system. On the other hand, we were able to finish sooner given the fact we did not have a member from last semester who slowed the process down.

## Project Development and Methods (Karrtiigehyen, Nicolas Popal and Patrik Horny)

The project was developed during third semester with an extra two weeks reserved for the semester project from end of November.

Since we did not really spend that much time in school, meetings were held mostly on Discord and occasionally on Zoom. This year it was mandatory to use GitHub for version control of the project. At first it was little bit tricky to set it up and learn the commands, but after some time we got familiar with it and we will be using it for the rest of our upcoming projects since it is the easiest way of managing the whole project.

As our main framework we have used SCRUM together with a web application called Jira, which helped us to be on a track with sprints and project backlog. We did discuss on whether we should use SCRUM or Kanban to manage the work for the project. Ultimately, we settled for SCRUM since we learned it the previous semester and have had experience working with SCRUM.

For planning and tracking the progress of the project we used ClickUp as our SCRUM management tool.

## SCRUM (Karrtiigehyen, Nicolas Popal and Patrik Horny)

Scrum proved to be a handy framework during the work on second semester project. At the beginning of each sprint, we held a discussion of what do we want to put into the sprint, what are the main points to put attention to and how much time it is going to take us to do it. In Sprint Retrospective, we realized our mistakes and put into consideration what needs to be fixed and improved.

We formed a group where we set a role for each member of the group with a condition to change roles if somebody does not like his role.

The roles were split as seen below:

* Patrik – Scrum Master – his role was to manage the whole development team and overall making sure that the whole process of making the project is going smoothly. This includes setting up the meetings, managing project timeline and making sure that the team members have no problem with their given User Story.
* Kartiigehyen – Product owner – he was responsible for setting up a Product Backlog and making sure that it was fully understood.
* All members of the group – Development Team.

It was agreed on that the length of the sprint was set to be 3 days with the amount of work done per day set to be 8 hours / member.

### Product Backlog (Patrik Horny)

In the product backlog we put all the requirements that product owner wishes to be implemented in his system. After the backlog was established, we split it to priorities and begun working on critical priority requirements.

The process of accomplishing requirements was in the beginning rather slow, but after we got more familiar with Blazor and figuring out how forms work, accomplishing the requirements became easier.

At first, when we looked at all the requirements, we were a little bit sceptical if we are going to be able to accomplish them all. We wanted to make a system that resembles real-world airline reservation system, so we made requirements based on that. We know that this kind of system is a little bit complex, so we didn’t know what to expect therefore we had to put more thought into the design of the system. But at the end of the day we were able to accomplish them all.

The product backlog is included in the Appendix ….

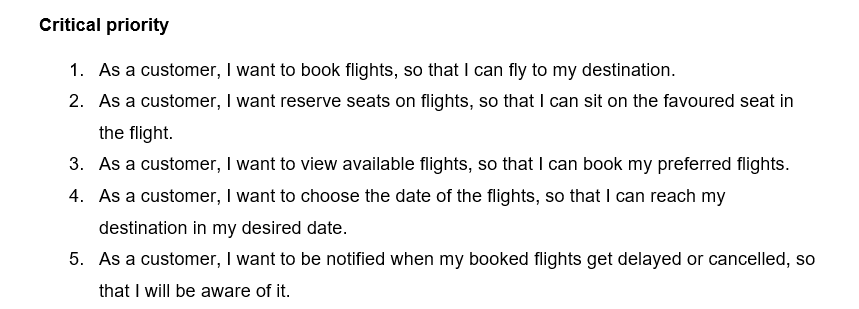


Figure 2: Some examples of product backlog

### Sprint Planning (Patrik Horny)

Before we begun a new sprint, we did a planning on what we’re going to do in the sprint. First sprints were all about implementing core features of the system and if the time permitted, we would focus on low priority requirements in the last sprints.

Planning went surprisingly well at the beginning and at the end. The beginning was just setting up the whole project and by the end of it we gained more experience in coding, so we were able to do more work.

Problems occurred in the middle of the November, as you can see in the Burndown chart, where we set way too ambitious goals on which user stories to finish which just resulted in pushing them to next sprints. Also, one of our colleagues was travelling back home at that time so very little was done in one particular sprint.

We wanted to finish the system around 20th of November and started testing at the end of November. This of course didn’t happen, and things had to be pushed. Even though our estimations weren’t precise, we were able to work out most of the things at the end of November and begun the testing around at the beginning of December.

Sprint planning helped us built the system gradually and gave us a good overview on what to focus.

The planning for each sprint can be found in the Appendix ….

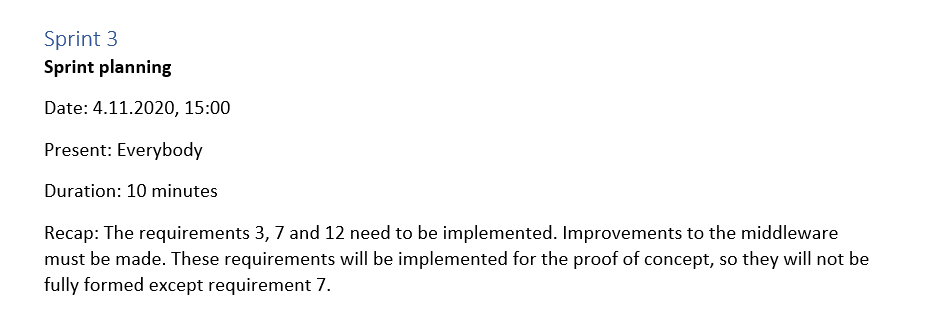


Figure 3: An example of a sprint planning

### Sprint Backlog (Patrik Horny)

During the sprint planning we picked specific user stories from product backlog and put them into sprint backlog. Then we gave each user story an estimate time of completion. We decided not to split user story into the subtasks since each user story share the similar tasks, so we thought of it as redundant.

The planning for each sprint can be found in the Appendix ….

### Burndown Chart (Patrik Horny)

Each sprint has its own burndown chart which we then combined together and put it into the main chart. At the end, this gave us a perspective of how well we were doing and if the team was on track with the time schedule.

The Burndown Chart is included in the Appendix …

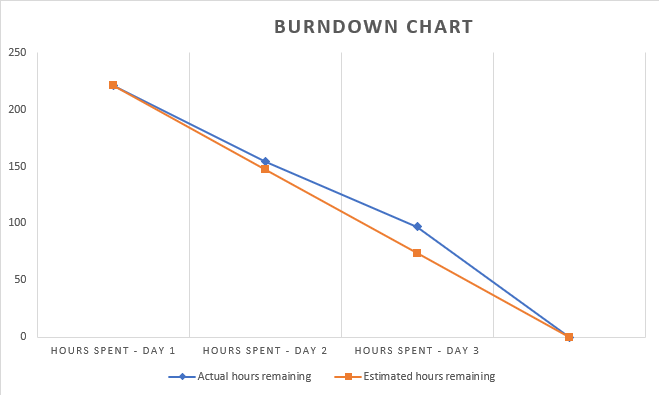


Figure 4: Burndown chart

### Sprint Review and Retrospective (Patrik Horny)

T In sprint retrospective we took look on what we managed to complete at the end of the sprint. There we documented our thoughts on what to focus next, what went well and what not and what to dismiss in the next sprints.

The Sprint Review and Restrospective is included in the Appendix …

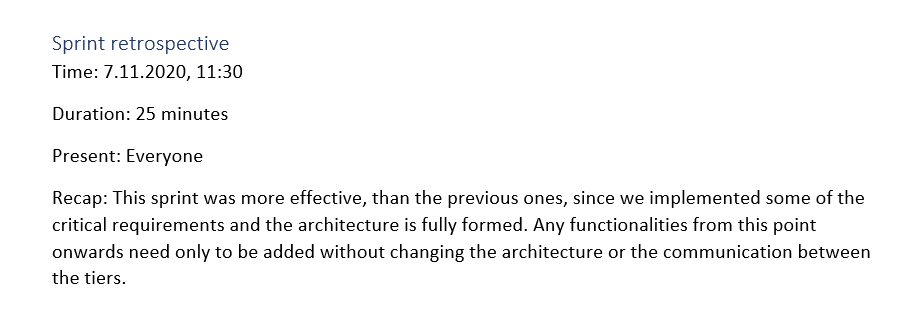


Figure 5: An example of sprint retrospective

## GitHub (Patrik Horny)

This semester we were asked to use Git as version control framework for the semester project and using it was very useful. Last semester we were sending files through Discord and e-mails which was inconvenient since every time we had to download the file and save. Git resolves these issues very well.

We set up a shared repository which we all got access to. Then we made a 3-tier architecture folder-wise structure. This gave us a good overview on where what changes were made and where were we supposed to upload new files, especially in the documentation folder where we made subfolders based on sections in a project report.

At first, we were making all changes to the main branch, but we ran into some issues when we merged our versions of the system and then we had to fix the bugs. We decided that we will make more branches where each of us would work on their stuff and then when we were finished, we would make an overview of the changes we made and then merge the versions.

Using Git was at first little bit problematic with learning new commands and setting up the repository but after everything was finished, using it wasn’t a problem.

We will definitely include Git in our upcoming projects mainly because of the convenience of the version control of the projects and clear overview who made the changes and where with an option to revert changes if something breaks.

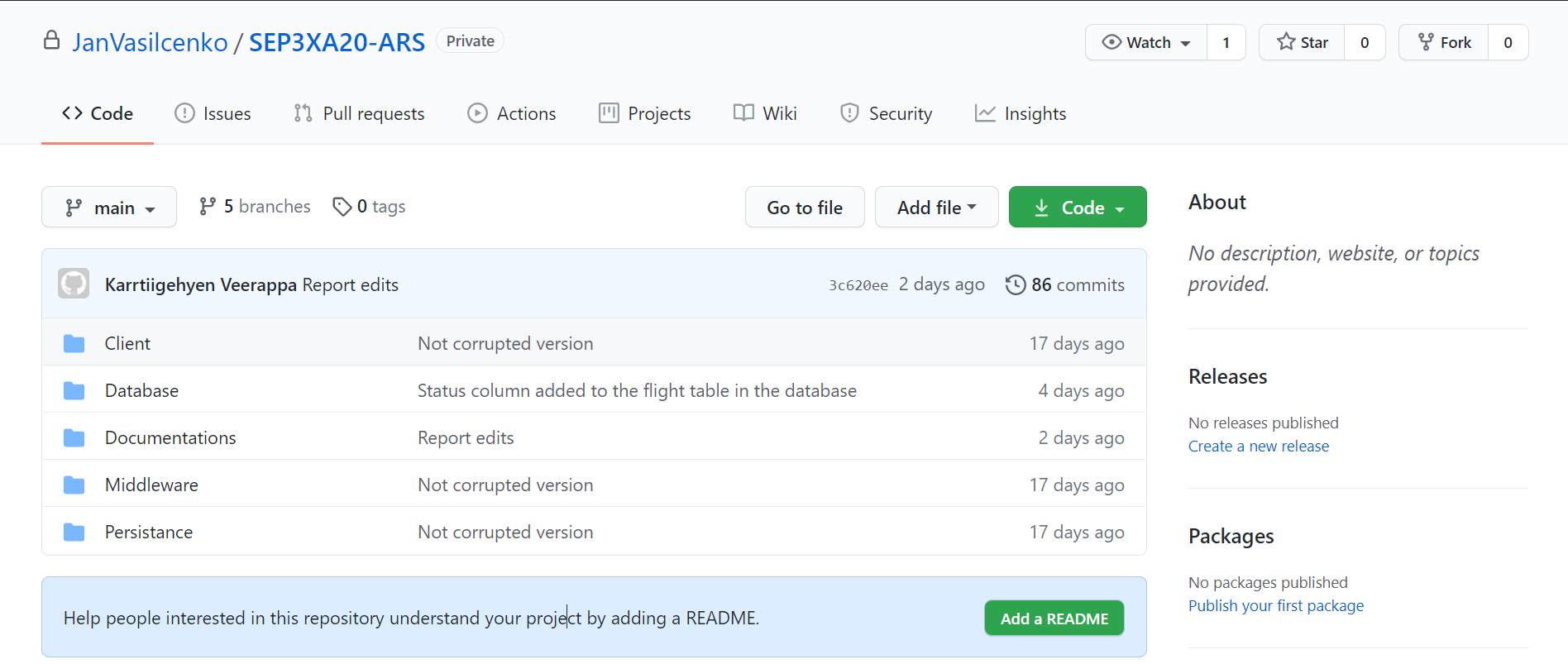


Figure 6: How the Github projects looked like

## Blazor (Patrik Horny)

Blazor is new web framework developed by Microsoft. It allows to create web apps using C# and HTML. Blazor apps are based on reusable components that can be shared between users.

We were excited to make good looking UI since making UI in JavaFX in the last semesters wasn’t the best way and it was difficult to use, and UI didn’t look good. Since we worked with Java for the last two semesters, switching to C# wasn’t that much big of an issue since they share some similarities and Blazor uses C#, connecting the code to the front-end wasn’t a problem, at least most of a time.

Of course, there were times where we wanted to use the Javascript, but we didn’t manage to initialize it, since in Blazor Javascript is replaced by C#. Also, most of the issues we had in the beginning was with figuring out how the format of the forms and the binding with the database. In the end we found out we should had just use the Blazor forms and not to mix them with regular HTML forms and inputs. Also, when we ran into some problems, often it was hard to find answers to the problems since Blazor is beginning to get recognition and there was very little information on the Internet on how to solve them. Therefore, we had to improvise a lot on how to tackle our problems.

Other than that, Blazor provides many interesting features like form validation, routing, binding the functions to the UI, it runs on WebAssembly which provides very fast performance in the browser. Working with Blazor was definitely interesting and it will be interesting what will the development of this framework look like in the future.

## Unified Process (Karrtiigehyen, Nicolas Popal and Patrik Horny)

Since SCRUM had no defined development techniques, Unified Process was used. It was used in each sprint, where we would go through the Elaboration, Construction and Transition phases. This gave us a guideline of what should be done in each sprint. Because of the unified process, the functional requirements that were supposed to be done in the sprints were working with no hiccups, and documentation of the implementation of the functional requirements were also done.

## Critique to The Project (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

We think that it is important to talk about things that have not gone as well as we hoped for and to talk about the design flaws of the project. This will hopefully make future projects free from these flaws, even though it is sometimes easier said than done.

### General Assessment and Critique (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

This project can be called a success since all of the functional requirements are met, but there have been many obstacles and problems during the development of the airline reservation system. A prime example of the hurdles we faced is *Blazor.* While we were taught how to use *Blazor* in the DNP course, it was difficult to work with it since not many helpful documentations could be found on it, given *Blazor* is a relatively new and evolving technology.

While every user story was implemented and tested, there was not enough time to polish the code to handle all types of exceptions. During the implementation of the application, we encountered several problems due to our lack of experience designing and working with distributed systems and three-tier architecture. An example of this is the implementation of sockets, where it is just a barebones TCP sockets which uses a *Request* class to send objects back and forth. A better implementation of the sockets would implement it with a proxy design pattern, and custom exception handling within the sockets. But ultimately, we decided to keep the design patterns simple to avoid second-system effect (C2 Wiki, 2004), and to focus more on the architecture of the distributed system.

### SOLID Principles (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

These principles served as a guideline how to construct the system and make it flexible and reliable.

The only notable violation of the SOLID principles can be found in some of the DAOs. For example, in *FlightDAO* class the method that adds the flights to the database also handled some logic where the ID of the flight is autoincremented, violating the Single Responsibility Principle. As a result of this, cohesion of code is reduced. A possible solution to this is would be to handle the autoincrementing of ID in the middleware instead of in the DAO.

Other than that, other principles were not violated and if they were, they were violated in a less severe manner than the Single Responsibility Principle.

### DRY Rule (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

Applying DRY rule to the semester project was our goal from the start. But applying the DRY rule in implementation was harder than expected. For example, a lot of the DAO concrete classes have a code repetition. Code repetition was tried to be nullified by making some reusable methods that can be reused in the same classes. But this did not entirely eradicate code repetition. A good solution to this could be to use the command design pattern.

# Personal Reflections (Everybody)

Text

# Supervision (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

For this semester, our project supervisors were Jakob Knop Rasmussen and Ole Ildsgaard Hourgaard. Their supervision was very good. The questions we asked were answered clearly and supervisors were very helpful when solving our problems. The feedback on some of the mandatory assignments, such as Project Description and Architecture, were the most useful as they guided us clearly to our path. The biggest issue in the supervision was the response time, when we sometimes had to wait for number of days to even get a reply. However, due to a coronavirus situation and the fact that some of these questions were asked during the semester project work weeks, when teachers usually get large number of questions and requests from other semester project groups, it is understandable that the response time was slower than we expected.

# Sources of Information

1. Schwaber, K. & Southerland, J., 2017. The Scrum Guide. [pdf] Unknown: Creative Commons. Available at: < https://www.scrumguides.org/docs/scrumguide/v2017/2017- Scrum-Guide-US.pdf> [Accessed 31 May 2020].

2. CatB, 2004. *Second-System Effect*. [online] Available at : <http://www.catb.org/jargon/html/S/second-system-effect.html> [Accessed 08 December 2020].