Computer Science program at

First Year Professional Assignment
“EMPLOYEE MANAGEMENT SYSTEM”

Group members:
Tadas Salkauskas
Edgaras Kazlauskas
Grzegorz Goraj
Jakub Fryszczyn

2015
December

Table of Content:

1. ITO

1.1. Organizational structure (Tadas)

To define what is the organizational structure of FK Distribution we have reviewed 5 different Mintzberg's organizational types and find to which type belongs FK Distribution. The main successful organizational structures that he identifies are as follows:

- **The entrepreneurial organization.**
- **The machine organization (bureaucracy).**
- **The professional organization.**
- **The divisional (diversified) organization.**
- **The innovative organization ("adhocracy").**

After research of all types we find out that Machine organization and the Divisional organization definitions are those that belongs to FK Distribution, and the definitions of each as follows:

The Machine Organization (Bureaucracy)

The machine organization is defined by its standardization. Work is very formalized, there are many routines and procedures, decision-making is centralized, and tasks are grouped by functional departments. Jobs will be clearly defined; there will be a formal planning process with budgets and audits; and procedures will regularly be analyzed for efficiency.

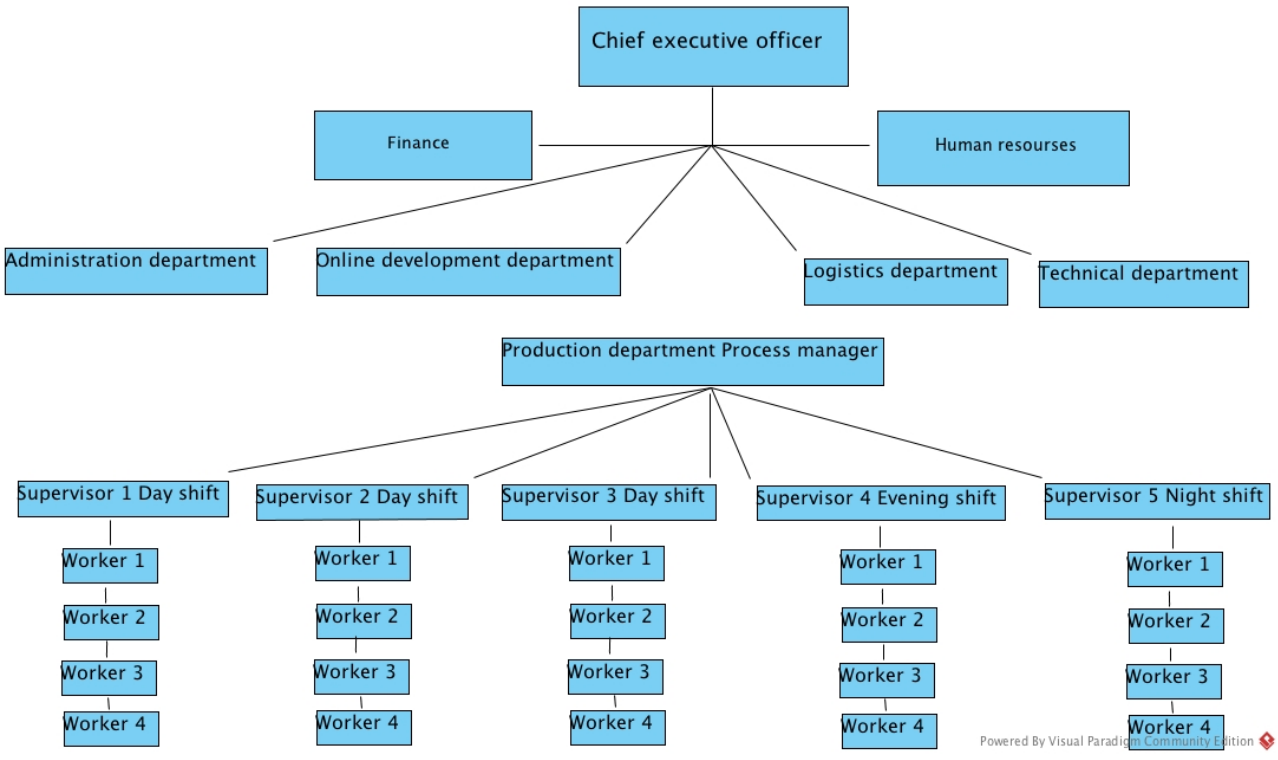
Large manufacturers are often machine organizations, as are government agencies and service firms that perform routine tasks. If following procedures and meeting precise specifications are important, then the machine structure works well.

The Divisional (Diversified) Organization

Organization has many different product lines and business units. A central headquarters supports a number of autonomous divisions that make their own decisions, and have their own unique structures. Usually this type of structure is in large and mature organizations that have a variety of brands, produce a wide range of products, or operate in different geographical regions. Any of these can form the basis for an autonomous division.

The key benefit of a divisional structure is that it allows line managers to maintain more control and accountability than in a machine structure. Also, with day-to-day decision-making decentralized, the central team can focus on "big picture" strategic plans. This allows them to ensure that the necessary support structures are in place for success.

The following picture is the structure of FK Distribution.



1.2. SWOT-analysis (Edgaras)

S.W.O.T. analysis of FK Distribution

SWOT analysis is a very useful tool which can help to evaluate the capabilities and uncover the opportunities of the company. This tool consists of evaluating company's strengths, weaknesses, opportunities and threats. Strengths and weaknesses are usually internal factors that depend on exact company while opportunities and threats are the external factors that are mostly influenced by the outside factors that are not depending from the company itself. The examples of external factors are such as legislation, economical changes, technological improvements.

Strengths	**Weaknesses**
• Company is well known in the market • Offering high quality service • Good customer service is company's top priority • Reasonable prices • Strong management team	• Relatively high staff turnover • It takes time and financial resources to train new employees
Opportunities	**Threats**
• The cost of entering the market is very high therefore the competition is not very intensive • To enter the Swedish market • Businesses are more willing to advertise themselves	• New economical crisis would make the company to lose some clients • New competitors from other countries where labor force is cheaper might decide to enter the market • One big competitor in the Copenhagen area

After making the SWOT analysis of FK Distribution we could say that the company has really strong positions in the market and that the future also

looks to be very bright. One of the main things that make FK Distribution to look very strong also in the future is that this kind of business requires a lot of investment in order to enter the market. Therefore, it is not very attractive for new players to enter the market. The fact that there already are 2 very strong companies in the Copenhagen area makes it even less attractive.

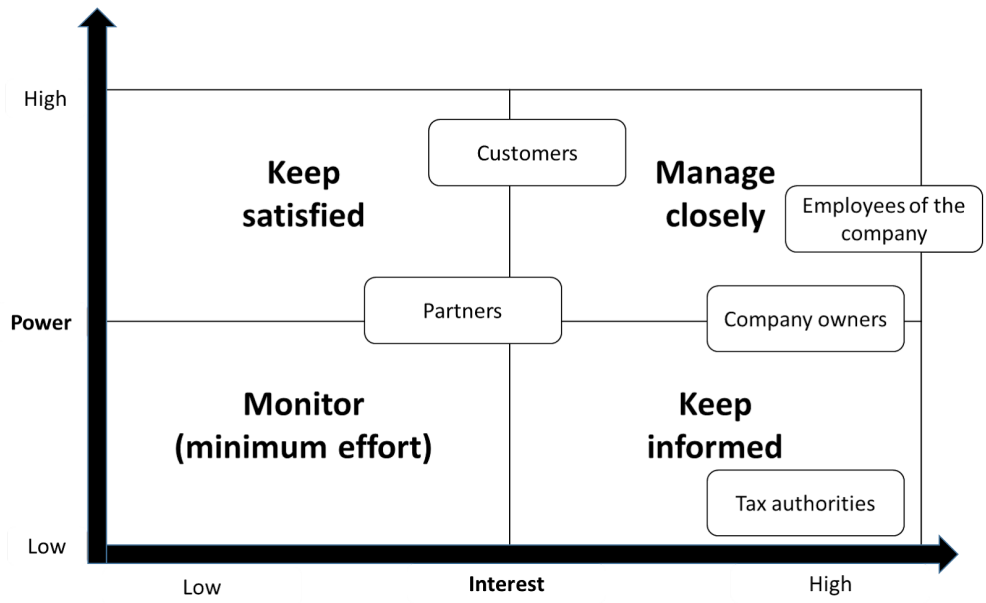
1.3. Stakeholder analysis (Edgaras)

FK Distribution stakeholder analysis

Stakeholder analysis is a commonly used tool that identifies individuals or organizations that will be affected by some kind of action or a change. A stakeholder is anyone who can be both positively or negatively impacted by an action or some kind of change made within the company. In this case we will discuss who and how will be affected by installing our staff management program. In order to clearly show the importance of all the companies stakeholders we will use a stakeholders matrix.

We have determined these stakeholders:

- Owners of the company;
- Employees of the company;
- Clients of FK Distribution;
- Tax authorities;



With the help of this matrix we have placed all of our stakeholders according to their interests and power in the company. In this way it is easier to understand how much effort will we have to put while engaging with these stakeholders. This diagram shows that the owners of the company and the employees will be those that needs to be satisfied the most. It is very important that both of these stakeholders knows everything that is directly connected to their work. Employees will have to be informed about the new system that we are creating for mainly two reasons. First of all, it is very important that employees will know exactly how to log in to the system, how to mark their availability and where to find the work schedule. The other reasons is, that if the employees are not well informed about the changes it might make them unsatisfied which might create some disappointment and frustration.

Company's clients should also be informed about the most important changes that are happening in the company. We know that representatives of the customer companies are regularly visiting FK Distribution in order to get a good knowledge about how the job is being done.

Tax authorities should also be informed as much as possible. SKAT is closely monitoring all businesses including FK Distribution and therefore the company should always give all the necessary information about any important changes involving the staff.

1.4. Feasibility study (Jakub)

Description

Our aim is to deliver an intuitive system that will help schedule the work for the employees. We want to allow employees to mark their availability in the system and then deliver the list of available workers to the managers. This will ensure that the managers do not waste time on calling people that are not able to work on a given day enabling them to take care of more useful tasks. We want both managers and employees to benefit from our system so we want to make the interface as intuitive as possible.

Timeline

We believe it will take a month to finish the project where in the first two weeks we will be mostly focused on the design part and in the second two weeks we will be implementing our solutions to code. Throught the process we will be in a frequent contact with our employer to ensure the best quality service and to deliver a product that our client is content with. We are going to use Unified Process for organising work.

Technology

We will need a database with a connection to a server which will allow users to take advantage of the system wherever there is a computer with internet. The cost of the server with space sufficient for a company of our capacity is…. Since we will not provide an option to create a manager account we need a person from our organisation to have access to the database for making potential permission changes.

Cost and budget

We are a student organisation so the only cost that the company has to take is the time that it invests for explaining us what they expect. In case of our failure to delivering the product they desired the only resource they lose is that time.

1.5. Problem statement (Edgaras)

While looking for a most appropriate problem statement our group was trying to clearly identify what is the purpose of our program, how it will benefit the company and who will be the end users. FK Distribution is using a lot of temporary employees who do not have a fixed schedule therefore sometimes there are problems while booking them for work. And that is where our system will be helpful. With the help of our booking program FK Distribution will be able to book people in a fast and an easy way.

Therefore, the main problem statement sounds like that:

How to deliver a system which makes booking of temporary employees more effective and efficient?

We have also decided to divide the main problem statement into separate smaller question that would eventually lead us to making a fully functioning and user friendly system:

- Who will be the main users of the program?
- How to design the program so that it would be easy to navigate in?
- What kind of requirements does the program has to meet?
- What kind of challenges we might meet while constructing the system?
- What deadlines do we have to set for ourselves so that the program will be delivered in time?

1.6 Risk Management (Gregory)

TOWS – analyses:

Strengths:

- well established
- strong in the market
- good reputation
- efficiency
- increasing numbers of customers

Weakness

- high staff turnover
- dealing with good logistic management
- lack of time
- crowded market sector (big competition)

Opportunities:

- high quality service
- good customer service is company's top priority
- reasonable prices
- strong management team

S-O

- W: Competitors O Good Customer Service
- In fact of the big competition of the market
- company focus es on good customer service.
- Happy customer comes back
-

W-O

- W: Competitors O: Reasonable prices
- Dealing with big competition on the market to reach the customer. The high quality service is not enough. Price is still the biggest market driver.

FK DISTRIBUTION	**Opportunities** High quality service Good customer service is company's top priority Reasonable prices Strong management team	**Threats** New cheaper competitors on market Luck of good system management of employees and fast training of new once global economic crisis
Strengths Well established Strong in the market Good reputation Efficiency Increasing numbers of customers	**S-O** S: Strong market position O: High quality service High quality service as well as great contact with customer building the strong position company on the market S: Efficiency O: Strong management team The efficiency of the company is depend of the strong well skilled management team. Just this kind of team will be able to deal with fast training of new employees	**S-T** S: Strong T: Competitors There is always treats of competitors. especially those companies located in foreign countries where labor force is cheaper S: Efficiency T: Good system management. System management is understood as set of tools which are to help in providing more efficient work organization.
Weakness High staff turnover Dealing with good logistic management Luck of time Crowded market sector (big competition)	**W-O** W: Competitors O: Good Customer Service In fact of the big competition of the market company focus on good customers service. Happy customer comes back. W:Competitors . O: Reasonable prices Dealing with big competition on the market. to reach customer. the high quality service is not enough. Price is the still biggest market driver.	**W-T** W: High staff turn. T: Luck of good system management well organized work eliminate chaos and help to save time. or reduce of cost new employees management W: Luck of time T: Luck of good system management Good time management is the key driver for distribution company. Therefore company must have a great system management.

2. Software Design

2.1. Supplementary Specification (Tadas)

Introduction

This project is a mandatory assignment for a second semester student, to test the knowledge of students and grade their knowledge in a grade system. Project was made by four students, maximum number of a group. Each group was given the set of requirements from each class to fulfill. The product of this project is a software prototype, that will be given to the specific company to work with it. All the software in this project is created by the group members.

Purpose

The purpose of this Supplementary specification is to briefly review the project that we made.

Scope

This supplementary specification is associated only with this project. No other applications or projects are related to this supplementary specification.

Overview

Further on this supplementary specification will be described functionality, usability, reliability, performance, supportability, interfaces, legal issues. The program starts at Login class.

Functionality

This project has to meet specified requirements in four subjects, which include Software Construction, Software Design, ITO and Operating Systems and Computer Architecture. This program is featuring a graphical user interface with JavaFx and MySQL database. This program is capable to

interact between the objects created in the source code and data in database, it is able to create, update and delete the information in it's database. For a security reasons it requires a user authentication.

Usability

This program is easy to use, and does not require high computing skills to use it. A basic level computer user can be productive by using this application. Regular task are easy to complete and it does not differ a lot from the other java applications. Colors and other components are selected to ease the using of program.

Reliability

It is planned that the program will work continuously during the day, and failure occurrence should be as low as possible. Short maintenance could be made at any time, if needed. Assuming that this is a prototype, so bugs and defects may occur in the very beginning, but expected to be fixed quickly after users feedback.

Performance

The system performance is highly dependent on network speed, because there is a network connection between JavaFx application and database, the faster is network - the faster system response. Application itself is very quick and takes less than a second to respond to user's input, because there is no complex and big components in the system. System is capable to accommodate one user at one device, but multiple users can use it on multiple devices. System requires a basic computer resources as memory, disc, network connection, etc…

Supportability

To enhance the support and maintaining of the system, there should be met all coding standards, naming conventions, class libraries, ect.. Communication between user and developer will happen on their own agreed methods (phone, emails..).

Interfaces

User interface: user interface is implemented by JavaFx application.

Hardware interface: system is running on the personal computer or other computing device that has memory, processor, display, input methods(keyboard, touchpad, ect..) and network.

Software interface: system can run on any software that supports java. Can be one of the operating systems of MS Windows, Mac or Linux.

Legal issues

This system is a prototype, so all the copyright notices are standard like the other prototypes. Developer will provide a warranty for the system for the agreed period of time. Developer will seed patent notice if they desire it.

What is the purpose of the program? (Edgaras)

FK Distribution is a relatively big company that is working within the whole East Denmark region. The company is sorting various advertisement papers or journals and distributing them to people all around the region.

As a lot of companies in Denmark they do not only have the permanent employees, but also sometimes they would book temporary workers which do not have a fixed schedule. This is because some of the times during the year there is more work than during the others. Therefore FK Distribution is working with some agencies which are providing this temporary work - force. However, sometimes it might be difficult to keep a track when and who is working therefore the company needed some kind of tool that would make this job easier.

In short, the purpose of our program is to make the booking of people for work easier for both the managers of the company and for the actual employees. Our program is a simple tool which is supposed to be used by both the managers and the employees. With the help of this system the manager can every day see how many temporary workers are available for every particular day. In other words, the manager is able to book the necessary amount of employees every single day and for every single shift. It

is important to mention, that the company has divided it's working day into three different shifts according to the time of the day: night shift, day shift and evening shift. And the amount of needed employees can vary from shift to shift and from day to day. For example, there can be situations that one day only 2 additional (temporary) employees will be booked for a shift, but sometimes the amount of the employees can reach 10 or even more.

Together with workers name and post code the manager will always see the phone number so whenever the manager will decide to contact the particular person he will not have to search for the contact information anywhere else.

Another important aspect that needs to be mentioned is that our program is keeping an archive in the database with the information of all the employees (also of the former workers). Thanks to that, FK Distribution will not have any problems in situations when they will need to retrieve any kind of information about one of their former employees.

It will also be a very useful tool for the temporary employees themselves, because they will be able to log in to the system and mark their availability for every single day. Moreover, they will also be able to see for which days the manager had booked them to work. The employees themselves will not be the ones who will be creating their own accounts. The managers of the company will do this job. After that, they will just give the username and the password to the worker. Every employee will have an unique username and password.

While working with the project one of our main goals was to make this system to be as user friendly as possible. We knew that some of the employees might not have good IT skills therefore we were focusing on making this program to be very easy to navigate. The goal was to make just enough of options to choose from for both the managers and the employees.

2.2. Use cases (Edgaras, Gregory, Jakub)

Brief use case description:

1. Create manager account:

A manager logs in into his account. He provides data necessary to create a new manager account which includes new user name and new

password. The account is created. The manager hands the new user name and the new password to the substitute manager.

2. Create employee's account:

A manager logs into his account. The he accesses employee's account creation form. He enters all the required data. The form is filled with the information. The manager confirms that the information entered is correct and the account is created.

3. Edit employee's account:

A manager logs into his account. Then he accesses a window which contains all the employee's information. The manager chooses the employee whose personal information he wants to edit. He edits the information and saves the changes.

Fully dressed use case description:

1. Use case: Create ultimate schedule

Scope: FK Distribution Production Department

Level: User Goal

Primary actor: Manager

Stakeholders and interests:

1. Manager: Wants to create the working schedule in order to see a list of people which are coming to work during the week. He also wants to have possibility to join or separate certain people from others in order to create more friendly and efficient working environment.
2. Employee: Wants to be able to check his/hers work schedule – remind himself when is he starting his shift, plan his time.

Preconditions:

1. All the employees have access to their accounts.
2. Employees have marked their availability.

3. The Manager has an account and full access to it.
4. Manager is successfully logged into his account.

Post-conditions:

1. The Ultimate Schedule is updated – the employees are booked and both employees and the manager can access and see the schedule.

Main success scenario:

1. Manager chooses the particular week.
2. Manager selects type of shift from particular day.
3. List of available employees appears.
4. Manager selects desired employees.
5. The set of employees is submitted.
6. The ultimate schedule is populated with approved employees related to certain date and type of shift.
7. Shift is accessible and changes are visible to all logged in users.

Extension:

*a At any time the system breaks down:
   a. Manager exits the program.
   b. Manager logs in again.
   c. Manager starts the process again.

If these procedures fail the manager contacts the IT support team.

2. Use case: Edit ultimate schedule

Scope: FK Distribution Production department

Level: User goal

Primary Actor: Manager

Stakeholders and Interests:

Manager: Wants to edit the Ultimate schedule so he can book more employees and unmark those who are not available to work anymore.

Employee: The employee is responsible to noticing the manager (by phone) that he cannot come to work anymore so that the manager can edit the ultimate schedule.

Preconditions:

1. The manager was noticed by the employees about the changes.
2. The manager has an access to his account.
3. The ultimate schedule was confirmed previously.

Postconditions:

1. New employees are booked.
2. Unavailable employees are canceled for the shift.
3. Changes are saved.

Main success scenario:

1. The manager successfully logs into his account.
2. The he chooses the particular week, the day and the shift he wants to edit.
3. List of already chosen employees and the rest of available employees appear.
4. The manager marks additional employees and unmark those who became unavailable.
5. The manager confirms and saves the changes.
6. The Ultimate schedule is updated.

Extensions:

*a. At any time systems breaks down:

1. The manager exits the program.
2. The manager logs in again.
3. The manager repeats the changes.
4. The manager saves the changes.

If these procedures fail the manager contacts the IT support team.

Special requirements:

- The computer has to be able to run Java applications.
- The computer has to have internet access.

Frequency of Occurrence: Could be nearly continuous.

3. **Use case: Mark employee's availability**

Name: Mark availability

Scope: FK Distribution Production department

Primary actor: Employee

Stakeholders and Interests:

- Employee: Wants to inform the manager about his/hers availability.
- Manager: Want to know who can he assign to a shift at particular day and time.

Preconditions:

1. Employee has to have an account.
2. Employee must be logged in.

Success Guarantee (Postconditions):

- Manager can see the list of available workers in the Managers table.

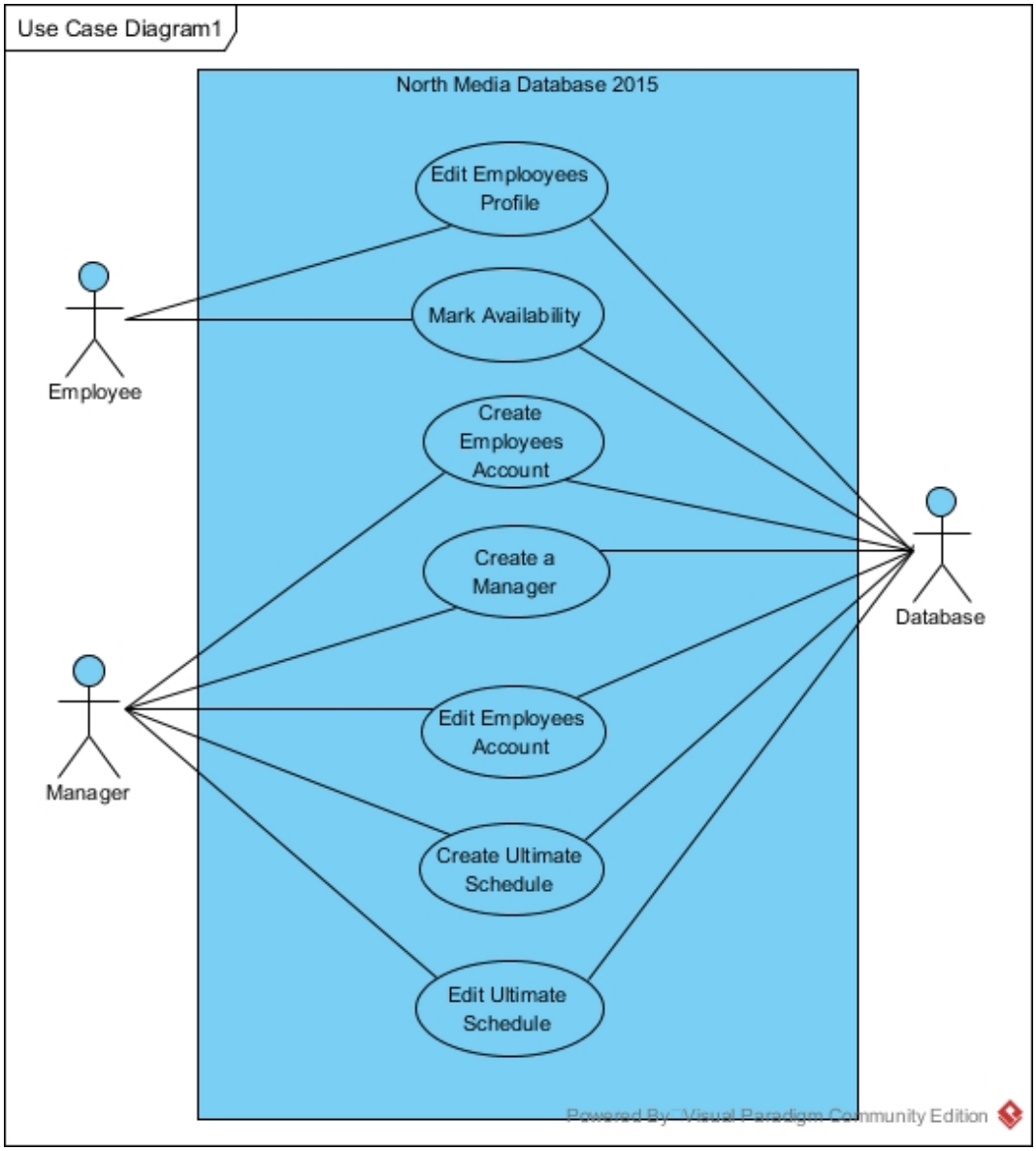
Main success scenario:

1. Employee picks the week of interest.
2. Employee selects the type of shift from a particular day.
3. All marks made by employee are saved into Managers table.
4. Message that the marks were saved appears.

Alternative flows:

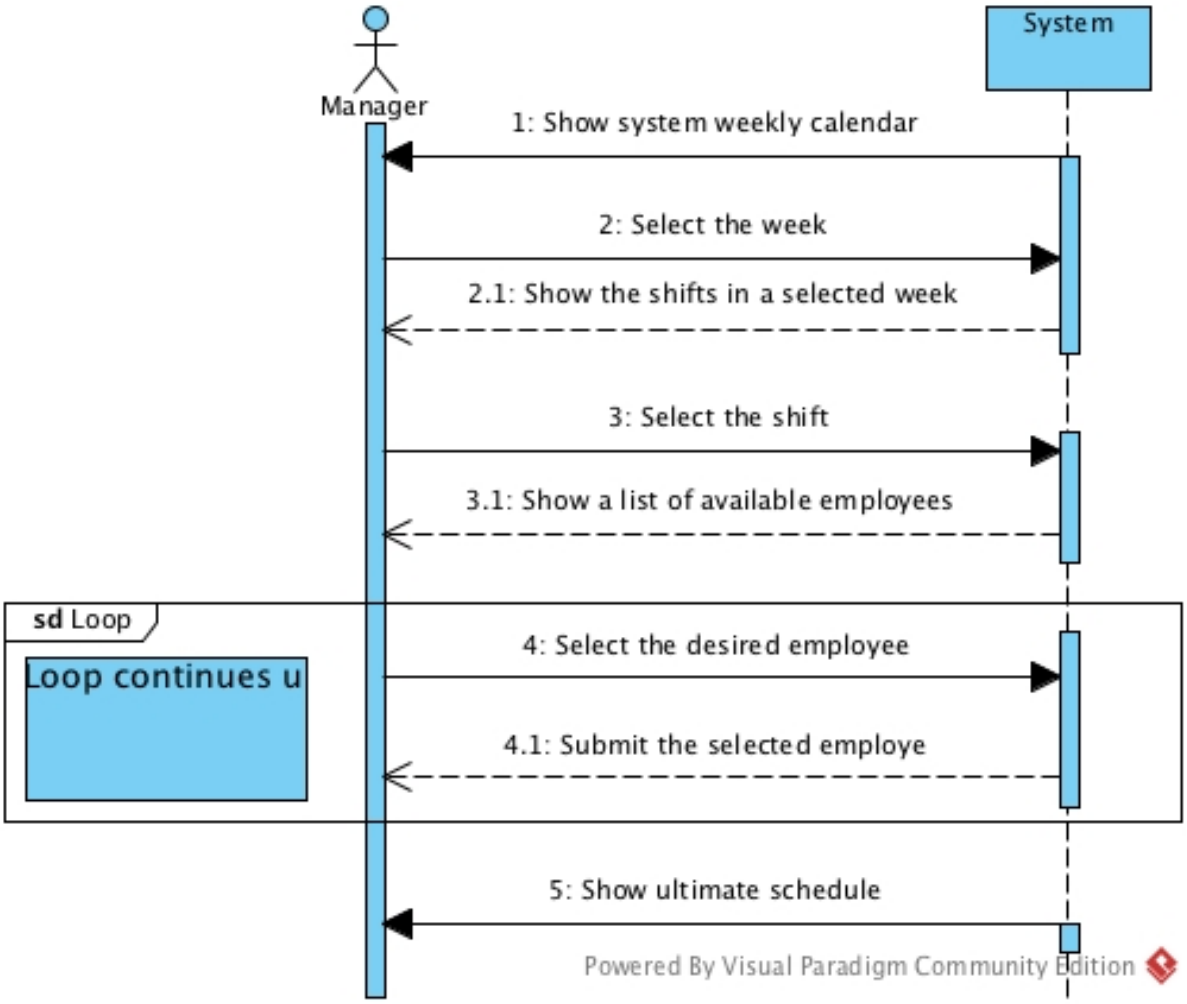
1. Massage saying that marks were not saved in the system appeared.
    a) Check your internet connection.
    b) Call the IT department.
    c) Call the manager.

In the following picture there is **Use Case Diagram**, that shows all use cases together:

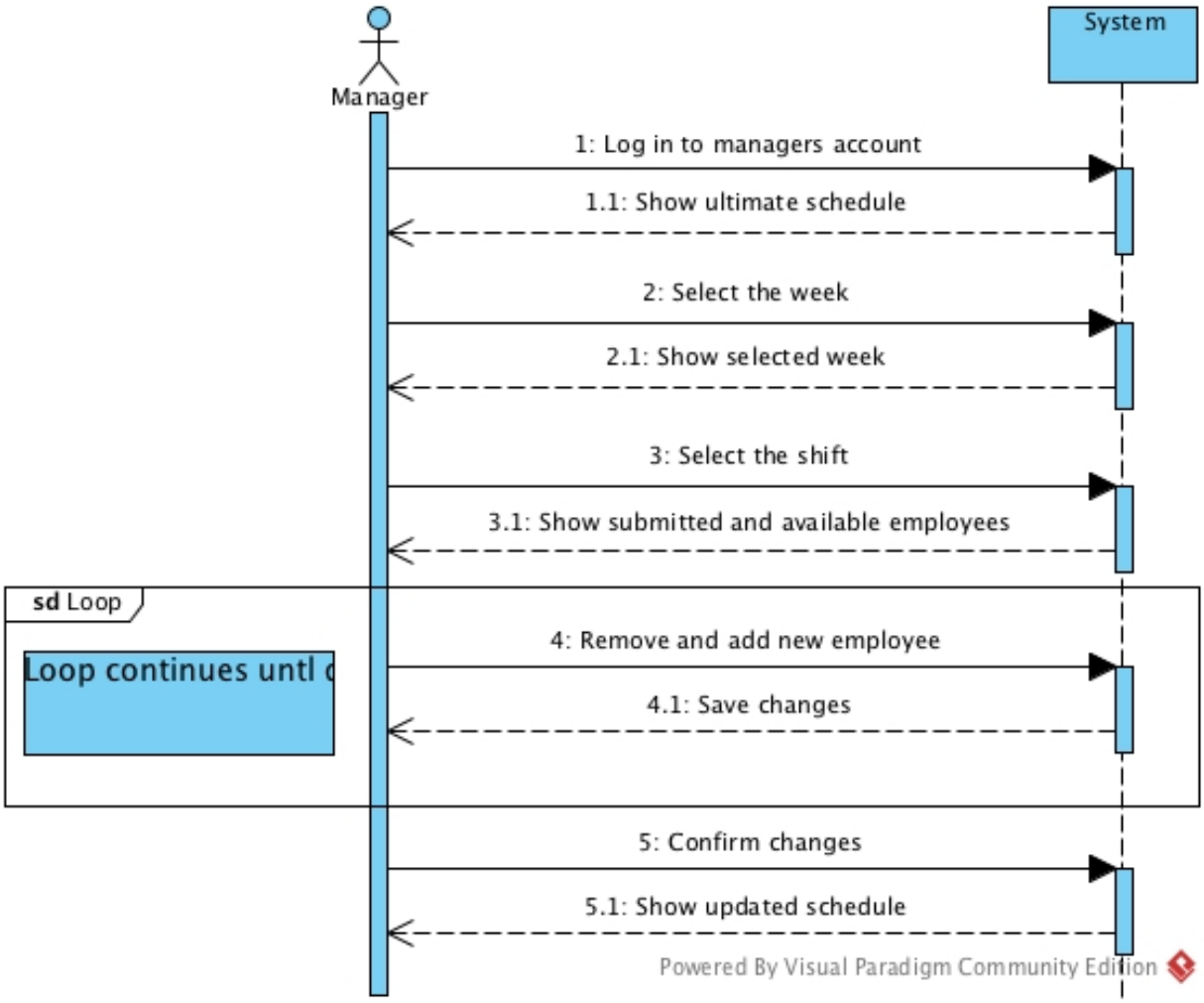


2.3. System Sequence Diagrams (SSD) (Tadas)

Following diagram is for use case Create Ultimate Schedule, it shows the events between manager and a system:



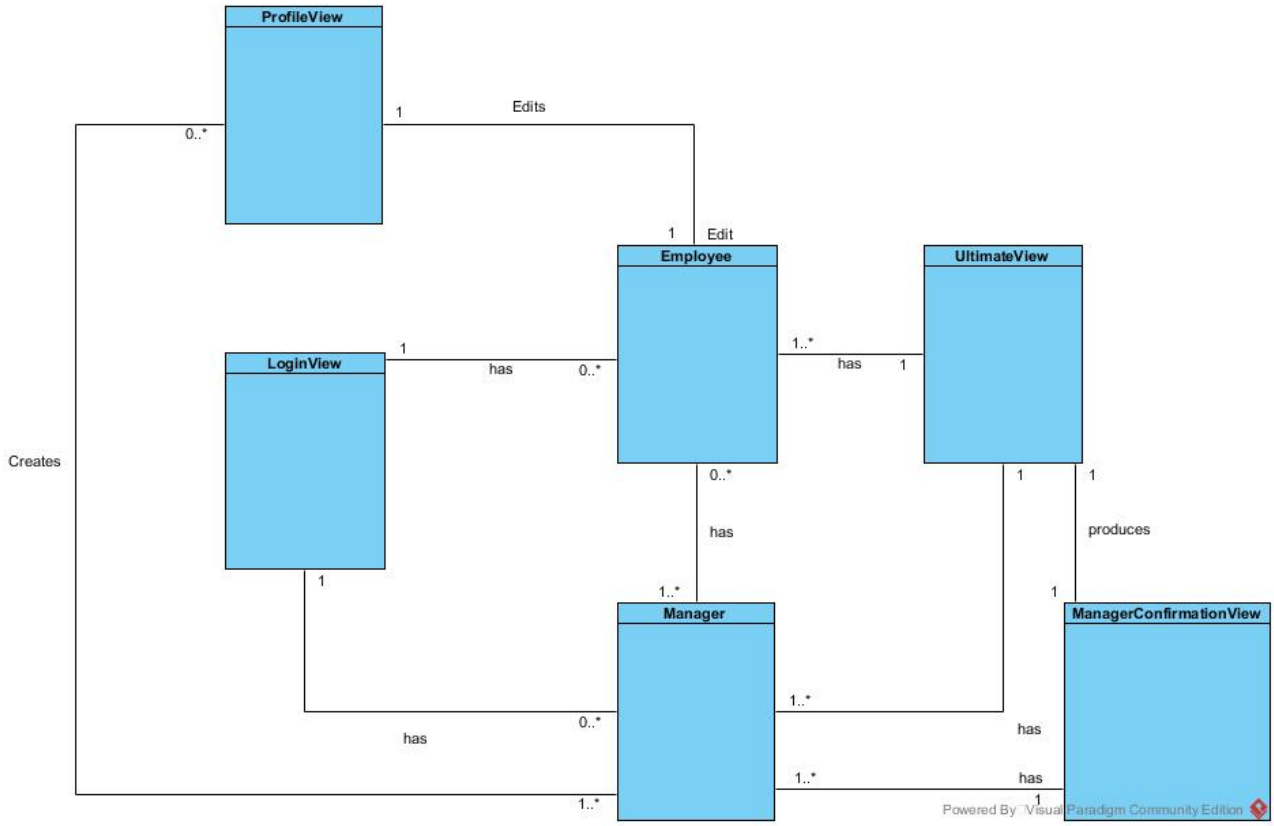
Following diagram is for Edit Ultimate Schedule use case, it also shows events between manager and a system, but in a different order:



2.4. Domain model (Gregory)

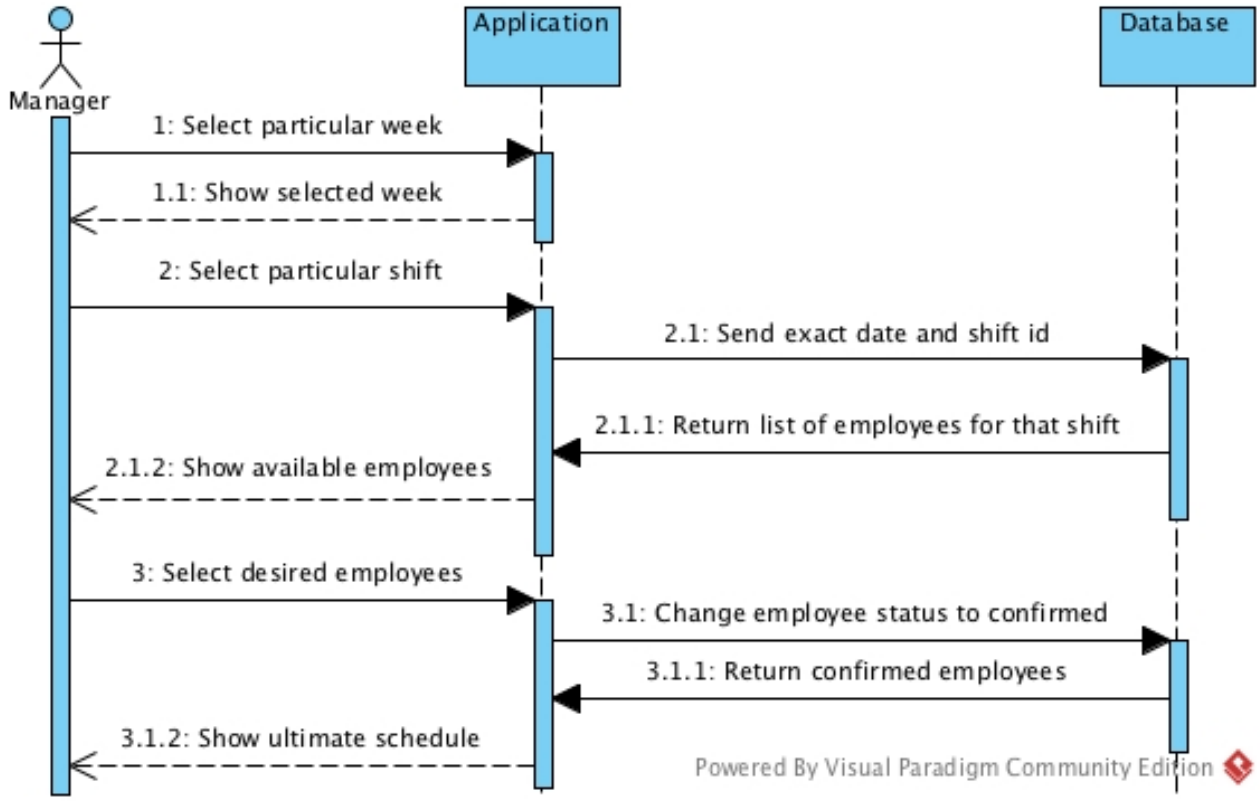
During ongoing collaboration we had to establish the scope as well as the requirements, list of features of our app. The domain model was very helpful to us to see and understand the main relations between our vision and reality.

I personally enjoy creating diagrams. I feel as it was the connector between me and my team, as well as between the whole team and our client.

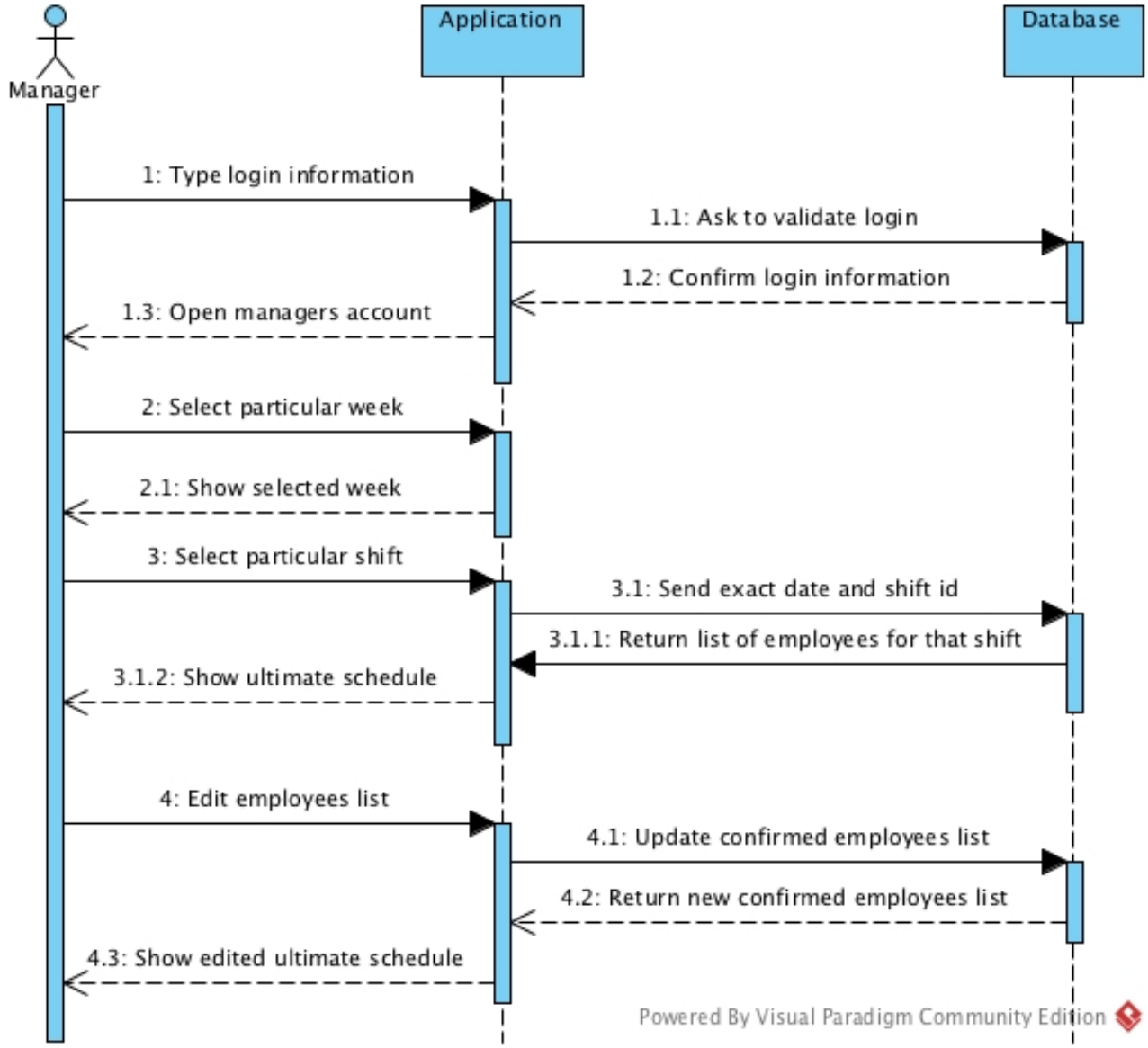


2.5. Sequence Diagrams (SD) (Tadas)

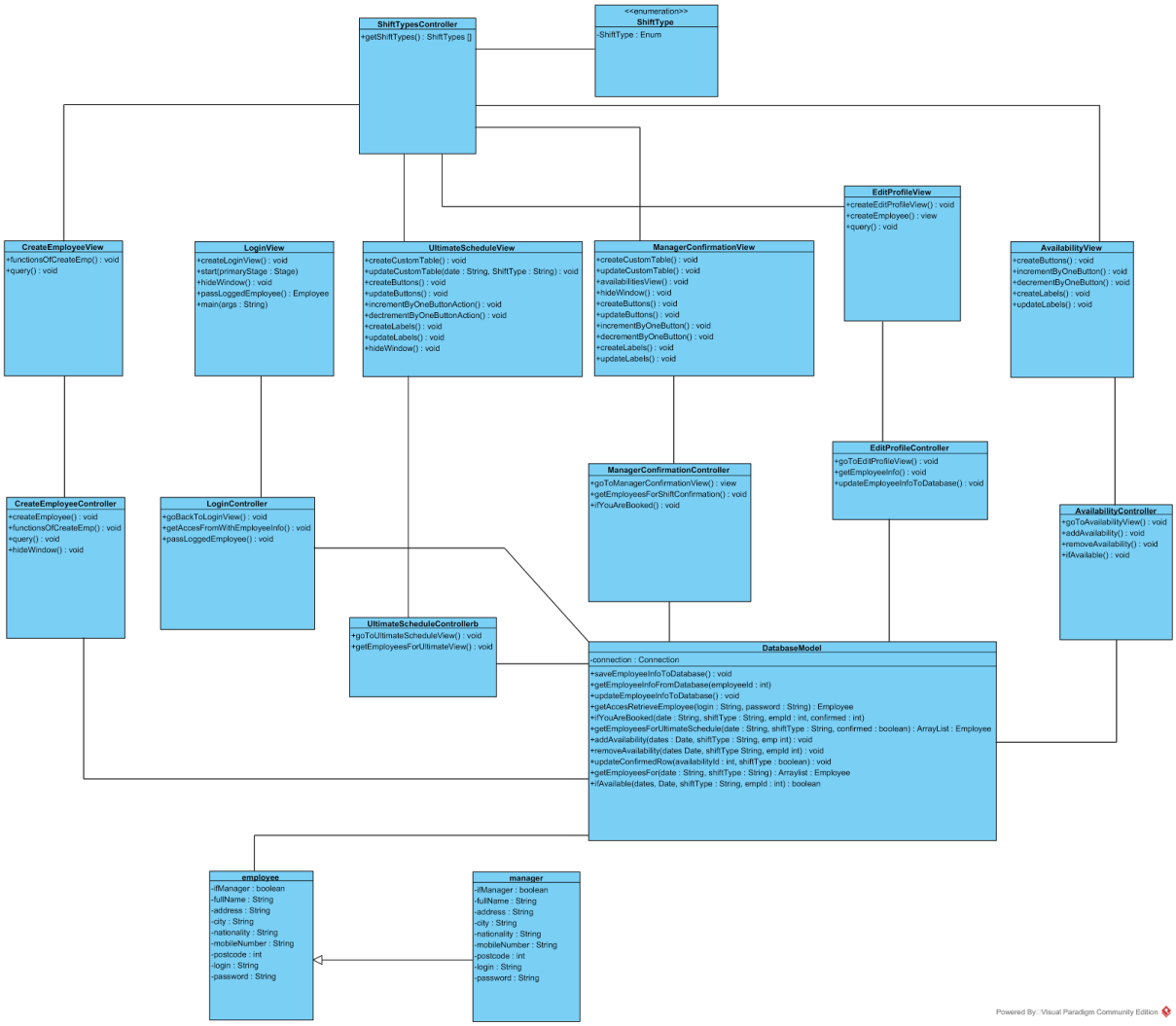
Following diagram shows the sequence between the manager and other system objects at use case of Create Ultimate Schedule:



Following diagram shows the sequence between the manager and other system objects at use case of Edit Ultimate Schedule:



2.6. Design Class Diagram (DCD) (Gregory, Edgaras)



2.7. GRASP patterns (Gregory)

1. Controller

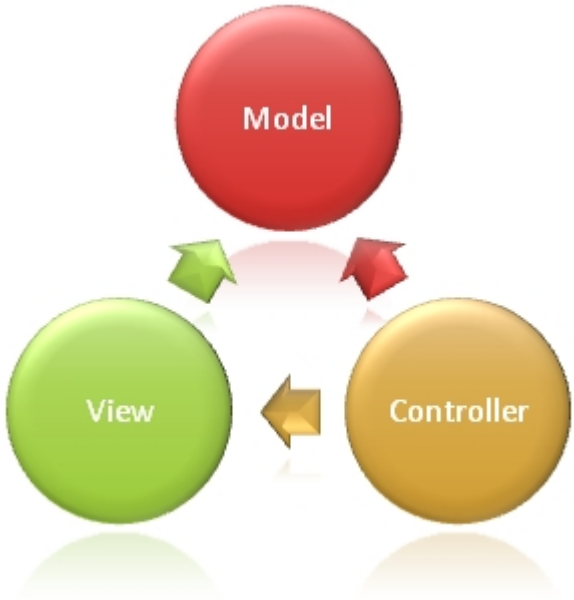
In a project of such a scope as that it would be hard to notice, that applying General Responsibility Assignment Software Patterns – GRASP is inevitable. Therefore from the very beginning have we planned on using the one, which I have to admit was the most innovatory for our barely second semester team – software architectural pattern known widely as MVC. It

divides our application into three parts (Source: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller):

- **Model** – which holds the data – connected to the database (DB). Model is not connected directly to users interface (UI), instead data is swapped between it and Controller.

- **View** – which represents the plane of communication between the user and the rest of the program (supporting therefore the „black box" principle) and keeping the usage of application easy as well as supporting the secured access to the program (user has no way of interaction between him and database.

- **Controller** – „Middle Man" between the View and Model. Responsible for inputs and outputs (IO) to and from the database. Accepts the commands from the user via graphical user interface (GUI).



Although the team put maximum effort to separate the program to the required layers (MVC), therefore supporting the Indirection principle, I would call our work a well done attempt and continue towards that direction in the future iterations of that project.

2. **Indirection, Low Coupling and High Cohesion.**

We have successfully managed to assign the adequate requirements to its classes. For example, particular classes of each single package enclose corresponding content such as:

- Package: View: AvailabilityView contains availability GUI etc.
- Package: Controller: availabilityController which corresponds with AvailabilityView.
- Package: Model: ShiftTypes is an Enum of type of Shifts.

3. **Inheritance**

During early phases of collaboration we all agreed that a manager is a function of employee in the structure of FK Distribution, therefore the class class Model.Manager class would extend class Employee.Manager was supposed to extend the Employee class. It didn't happened, because we agreed, that the difference between Employee and a Manager should occur only within database, in table called Employee. There the system administrator can find the column named ifManager therefore creating another object called Manager in Java lost in our eyes its importance.

In addition we applied inheritance between classes CreateEmployeesView and EditingProfileView. LoginView class extends the Application class because it is the Main class.

2.8. Phase plan (Edgaras)

One of the first things our group has done was to decide how we are going to organize our daily work with the project. We decided that the group will mostly work together and the meeting will be organized 5 times per week. Our work was organized according to the best practices of UP.

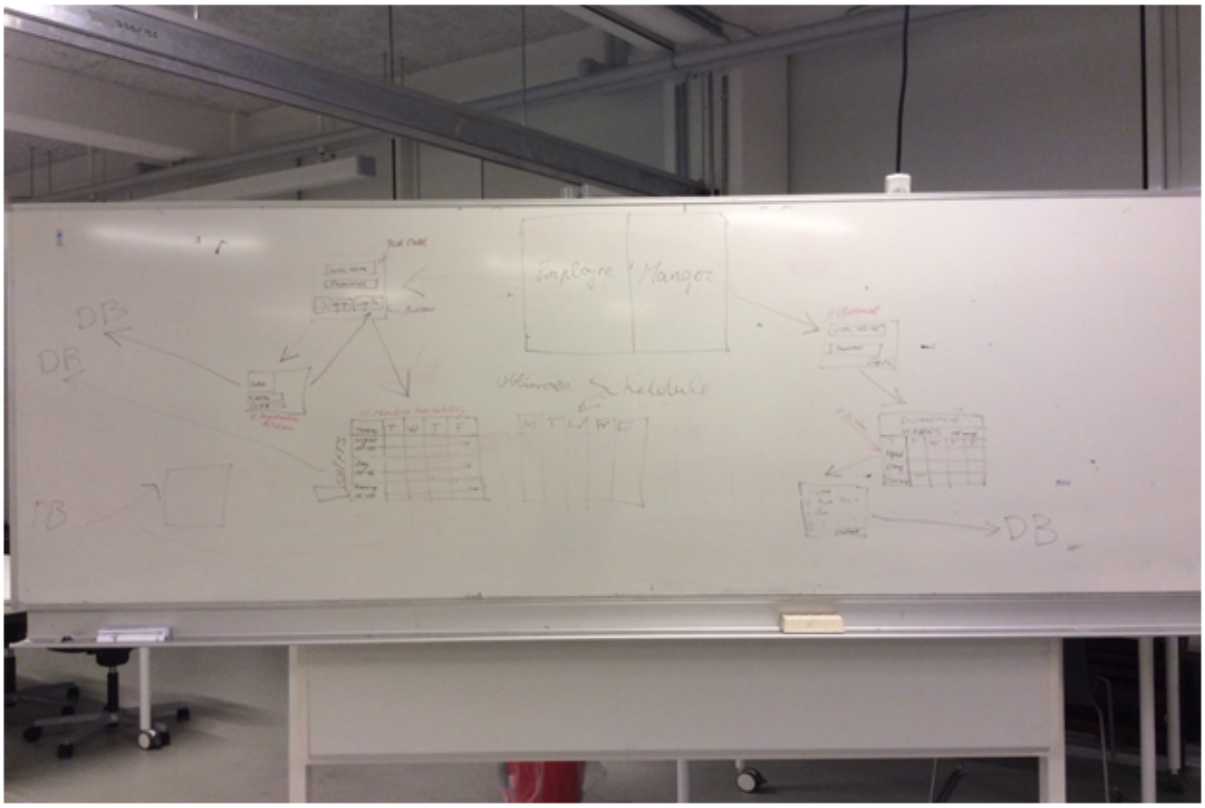
First week (Inception and start of Elaboration) 21th of November – 28th of November

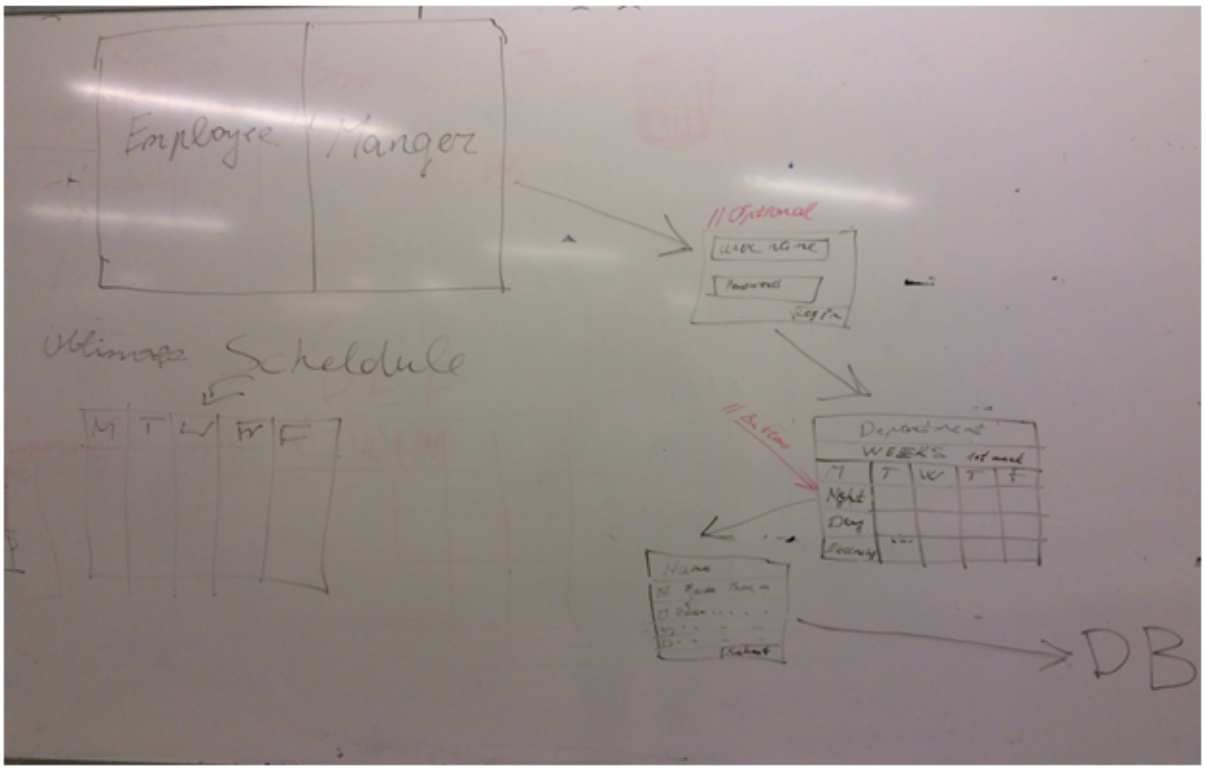
The first week was started with the Inception part. It had only 1 iteration and lasted less than 1 week. During this phase we organized our first

meetings in order to get a vision of our project. We wanted to understand is our system going to be beneficial to the company we are going to work for. For this reason during the inception phase there was a meeting with an employee of FK Distribution during which we changed opinions on how should the program look like in terms of some main functionalities. However, only the main functionalities were discussed at the meeting.

After that we decided to clarify the vision and validate some of the main technical ideas. For this purpose we were making user interface prototypes in order to get a brief vision of how our program might look like. SWOT, feasibility and stakeholders analyses were also made during the inception phase. The group was also working on non-functional requirements(supplementary specification).

During the Inception phase most of the actors and use cases were named and briefly discussed. At the end of the inception part we made a plan for the first iteration in the Elaboration phase.





At the end of the first week we entered the Elaboration phase with the first iteration. Our group decided to tackle the most important requirements during the early stage of the elaboration phase. Therefore, during the first iteration we have fully written most of the use cases so the requirements for the program will be even more visible. We also started coding at the first iteration, even though it was just basic UI views with some MVC details.

Second week (Elaboration, end of 1th iteration – 2nd iteration) 29th of November – 6th of December

The second week was fully dedicated to the second iteration of the elaboration phase. During this iteration our group have managed to stabilize our programs architecture and made sure that it will support the main expected functionalities of our system. Our goal was to make sure that we will not have to make any serious changes during the construction phase. Use case diagrams and conceptual diagrams were fully created.

At the end of the second iteration we have made a plan for the construction phase. This is the biggest and most important part of the whole process therefore we addressed major risk factors and tried to discuss what kind of delays we might have because of them. We have also decided that the

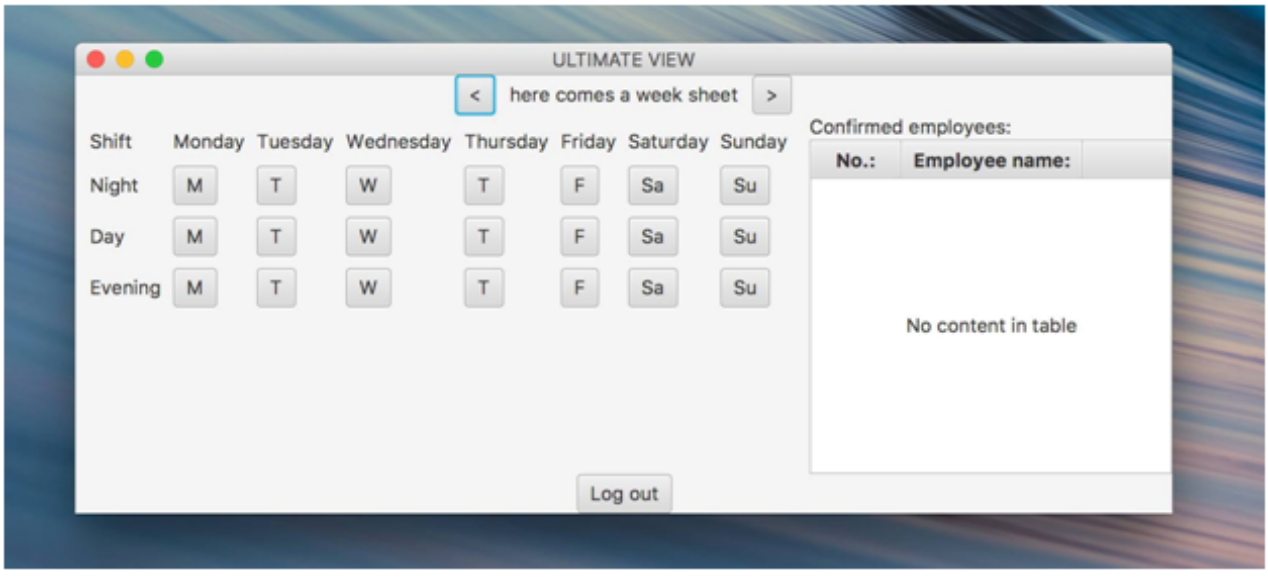
construction phase will consist out of 3 iterations. Our plan was based on previous experience with the inception and elaboration phases.

Third week (Construction phase, 1st – 2nd iteration) 7th of December – 13th of December

The construction part consisted of 3 short time limited iterations. The third week of our work was divided into first two iterations.

The first step in the first iteration was dividing coding tasks between all 4 members of our group. Because of the common methods and views that will be used in our system we made it sure that there will not be situations when two members of our group are doing the same or very similar job. At the end of the first iteration the executable version of the program was released and tested. Some decisions related to GUI design and the looks of the program were also made.

During the second iteration most of the methods and queries were established. This iteration also ended with the release of the functioning program. However, this time the program had all the necessary functions fully working, therefore only some last design adjustments were left for the last iteration of the construction part.



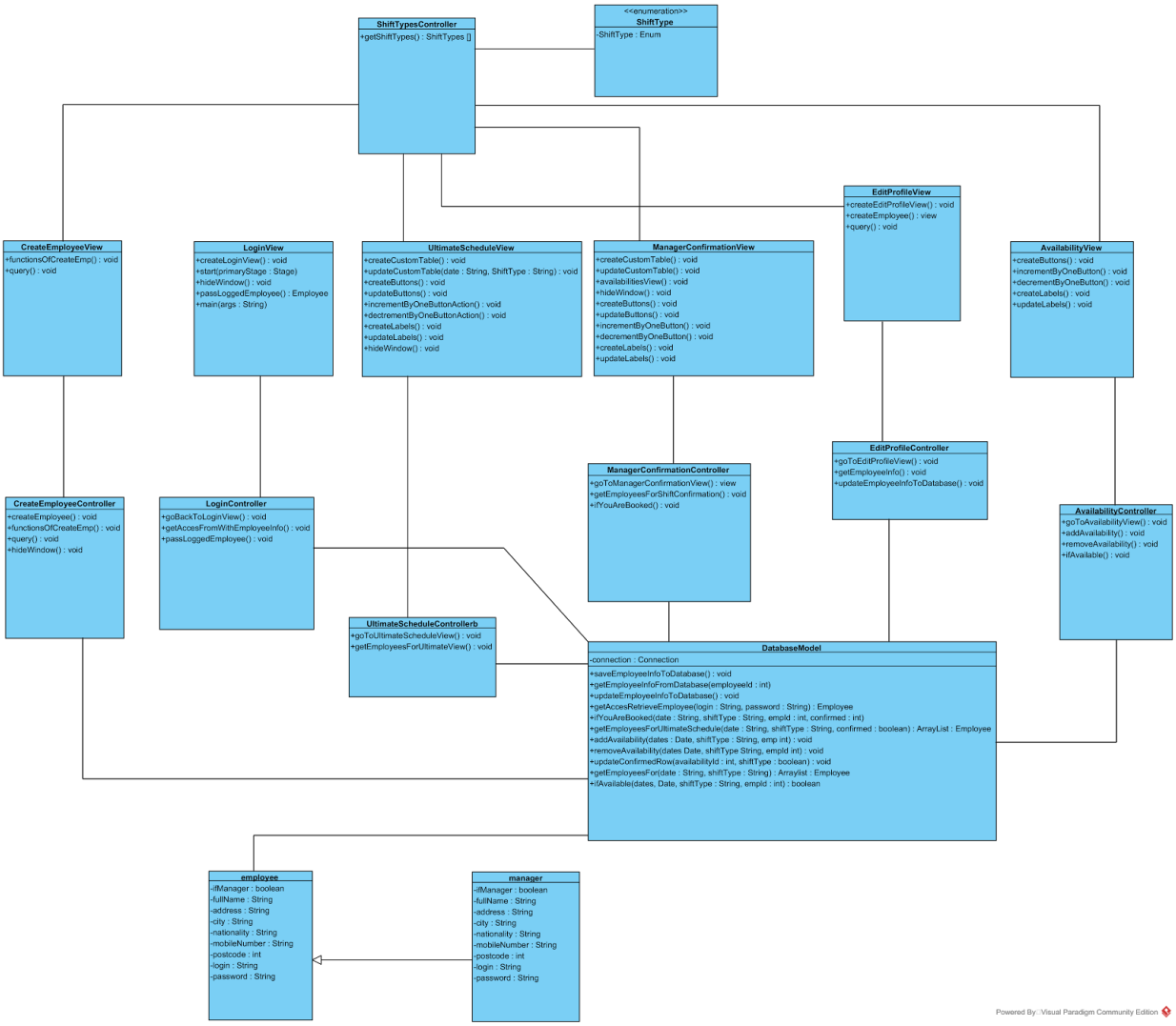
Fourth week (3rd Construction iteration – Transition) 13th of December – 17th December

During the third iteration of the construction part our group was concentrating on some very last functionalities that we decided to add to our program. Most of them were done in order to make the GUI more user friendly and simpler to navigate. We also spent a lot of time adjusting a background image so that it would fit to our views. After this iteration our system was fully done with all the functionalities working properly. The last testing of the system was also done in order to spot any kind of bugs.

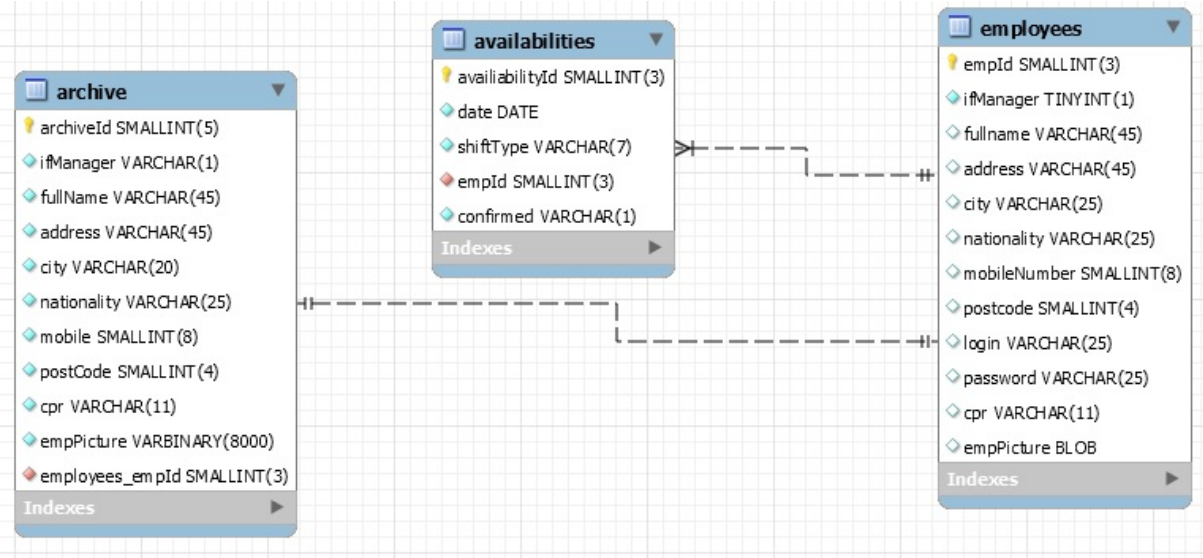
The fourth week ended with the transition phase which consisted from 1 iteration. During it we handed in our project with all the necessary documentation.

3. Software Construction

3.1. Class diagram, including methods (Gregory, Edgaras)



3.2. Diagram of database tables (Tadas, Edgaras)



Description:

In this ER diagram of our database we have 3 tables. Employees table has functional dependency with Archive table with a relation one to one. That means that every time there is a employee saved in the Employees table, so a copy of employee information is stored also in a Archive table, in order to keep whole history of employees data.

Other functional dependency is between tables Employees and Availabilities, where relation is one to many.

3.3. Scope of the SWC part (Tadas)

This developed “Employee management system” is a prototype, not a final product for market. We was prototyping and building software application which display the functionality of the possible system, but not the exact working application.

The motive to do the prototype came, because it is very popular software development model in a today's world, as it enables to understand customers requirements in early stage. In this way, we get valuable feedback from the customer, which helps us, as a developers, to understand about what exactly is expected from the product under development.

This system fully suits a software prototype definition because it is a working model with a basic functionality, we leave some room for improvement for later modifications after customer feedback. We think, that it is a good solution, because the customer can evaluate our proposals and try them out. It also helps understand specific customers requirements which may not been considered by us during product design.

Our prototyping was approached by these basic steps:

- Basic requirement identification;
- Developing the initial prototype;
- Reviewing the developing product;
- Revising and enhancing the prototype in the future;

This system is a vertical prototype, which is a detailed elaboration of a specific functions inside the software model, rather than just a user interface and look. For example, there was more concentration on database requirements, software interaction with a data and general system flow.

We choosed software prototyping, because it is most useful in development of systems having high level of user interactions.

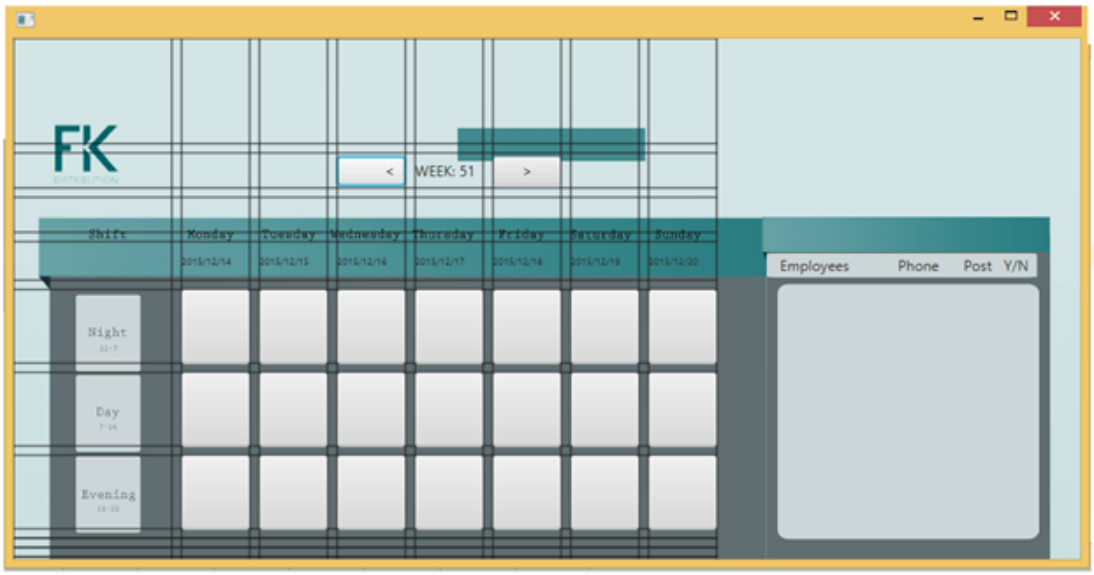
3.4. What is working and what is not (Edgaras)

Before the start of the project we have carefully chosen what functionalities our program should have. The program starts with the main *log-in* window where a manager or an temporary employee can log in. Even though the log-in is the same for everybody the person's position (manager or temporary employee) determines to which window after logging in the person will be sent to. For example, if the persons who is logging in is a manager he will be sent to managers confirmation view where he can see all the available employees for each day and for each type of shift. Here the manager with the help of fully working checkboxes can check in temporary workers for work whenever he needs them. After that, the information about confirmed employees is going to the database. Then the manager can click on an *ultimate schedule view* button and the *ultimate schedule view* window appears. Here the manager can see the full schedule with all the employees booked for work and their personal information including *name*, *telephone number* and *post code*. Post code is necessary to see which employees are

living close to the workplace so that they can appear at work quickly. The manager view also contains *create an employee* and *edit an employee* buttons. With the help of these two functions the manager can create an account for a temporary employee and also to edit his/hers personal information. The information that was typed in and saved is sent to the database to the *employees* table and to the *archive* table where we are saving all the information about existing and former employees. The archive table was created in order to be able to retrieve the personal information about the employees even though they might not be working for the company any more.

When the manager created an account for the employee the personal log-in information is given to the temporary employee. With the *log in* and the *password* he can enter the system. After logging in to the system the employee is sent to a view where he can mark his availability for work for certain days and types of shift. This is done in a way that the employee just simply clicks on a button which is set for the necessary day and the button turns green which then signalize for the employee that the information about his availability is sent to the database and now is update in the *managers confirmation view window*.In this way, the manager is able to see who is able to work for which day and for which shift. The temporary employee has an ability to mark his availability for all the days and all the shift types he wants, but the manager can book him for work just for one shift per day. Also the employee is able to access the ultimate schedule view window where he can see for which days he is booked for work. We made it easier to spot for which day the temporary employee is booked by making the buttons assigned for the particular shift to appear in green color. In this way the temporary employee will instantly see for which days he is booked to work.

Another thing that was made in our project is applying a custom made background image for all of our views. In this case, all the views have a nice looking background image. This was made with a help of CSS. However, we have spent a lot of time in order to adjust everything to our needs. While doing that we have learned how to work with such tools like *gridPane* or *borderPane*.



During the design part of our program we had a new challenge which was to learn how to work with custom made background images. This was a big issue for all the members of our team, because none of us had worked with them before. The biggest challenges was to learn how to position our labels, buttons and tables so that they would fit perfectly with the custom made background image. One of the tools that helped us a lot was *gridPane* lines which can be set to be visible. With the help of these lines it was much easier to set the necessary constraints for our buttons and labels. We were also able to set different spaces between these lines which helped us a lot while adding buttons or labels that had different sizes.

```
gridPane.setConstraints(ultimateScheduleButton, 6, 12,4,1, HPos.RIGHT, VPos.CENTER);
gridPane.setConstraints(logOutButton, 0, 12,3,1);
gridPane.setConstraints(createEmployeeButton, 3, 12,3,1);

// gridPane.getChildren().addAll(nightShiftLabel, dayShiftLabel, eveningShiftLabel, logOutButton,
gridPane.getChildren().addAll(logOutButton, createEmployeeButton,ultimateScheduleButton);
//gridPane.setGridLinesVisible(true);
gridPane.getColumnConstraints().add(new ColumnConstraints(130));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
gridPane.getColumnConstraints().add(new ColumnConstraints(56));
// gridPane.getColumnConstraints().add(new ColumnConstraints());

customTableGridPane.getColumnConstraints().add(new ColumnConstraints(100));
customTableGridPane.getColumnConstraints().add(new ColumnConstraints(60));
customTableGridPane.getColumnConstraints().add(new ColumnConstraints(30));
customTableGridPane.getColumnConstraints().add(new ColumnConstraints(5));
```

Another important aspect that we had agreed before actually starting the project was to have a common system for naming our variables or other items. Whenever a group of software developers are working in a team it is vitally important to agree on various names and words that will be used in the code. It makes a lot of sense to name variables in names that would clearly show for what reason it is being used. In other words, we were trying to make our code as easy understandable as possible in order not to have any problems or misunderstandings. Another thing that was important for our group was to comment the code wherever it is needed. Quite often software developers are having problems reading each others or even their own code. Therefore, commenting the code is a simple, but very effective way to make it easy readable not only for fellow developers, but also for yourself.

```
/***
 * buttons rely on labels text
 */
int indexButton = 0; // initializing the reference to ArrayList<Button> buttons of the main scope
```

Technical problems during the project (Edgaras)

Group Room

At the start of the project we have decided to look for some tools that would make it easier to share various files and also to hold some discussions when we are not together. Therefore we have chosen to use GroupRoom. GroupRoom is a free online collaboration tool which is supposed to help people who are working in groups. Because this tool looked to be promising and was free we decided to use it. GroupRoom have both computer and mobile versions and we were trying to use both of them.

However, already at the start of the project we were starting to face various problems while using GroupRoom. Quite early we have noticed that the user interface is not that comfortable as it could be. First of all, navigating through the files we had uploaded is not that easy and sometimes it takes some time before finding the necessary documents. Secondly, there were cases when some of our files just got lost. Therefore, after these and some

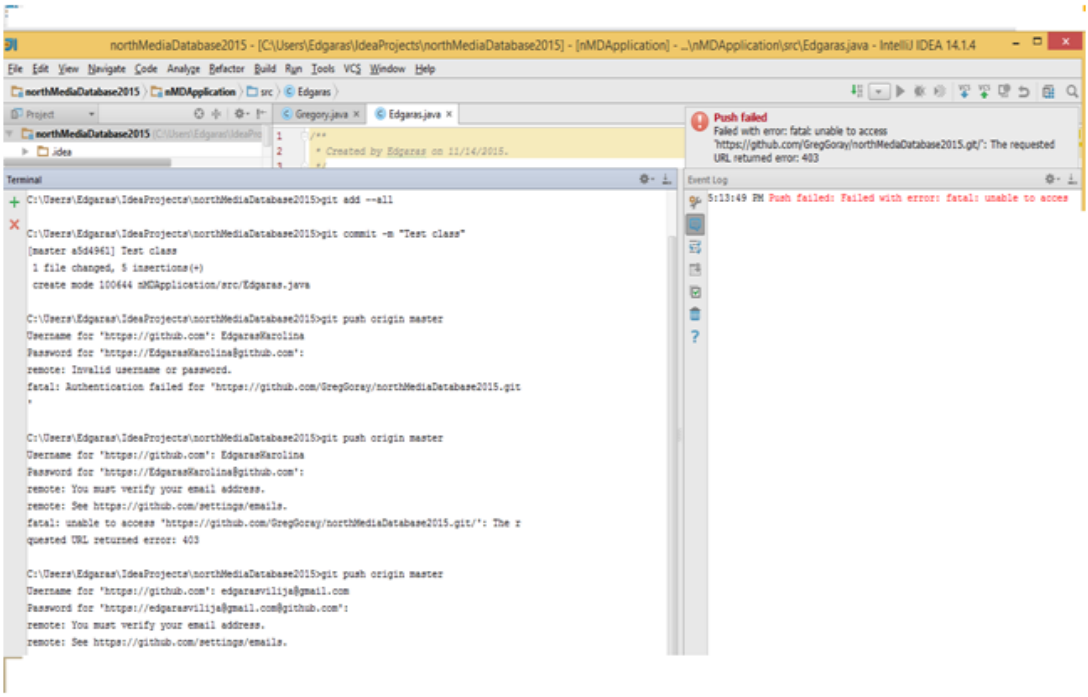
others accidents we decided not to use GroupRoom that much and instead for sharing some documents we were using Facebook which helped us a lot during the whole project.

Facebook (Edgaras)

Facebook was one of the most important tools for helping us to share various documents and files. It was also the main place where we had discussions during the time we were not together. When having problems with Git Facebook was also the place where we were sharing our code and putting everything together into one piece.

GitHub (Edgaras)

During the project we also had numerous problems with Git. Even though it is a really useful tool for software developers who are working in teams sometimes the amount of problems was overwhelming. Sometimes while wanting to execute *push* command the program was not recognizing user's *username* or *password*. There was also a case, when the entire class just disappeared. However, despite all the problems with Git we were still sometimes using this tool during particular periods of the development.



3.5. Description of GUI construction (Jakub)

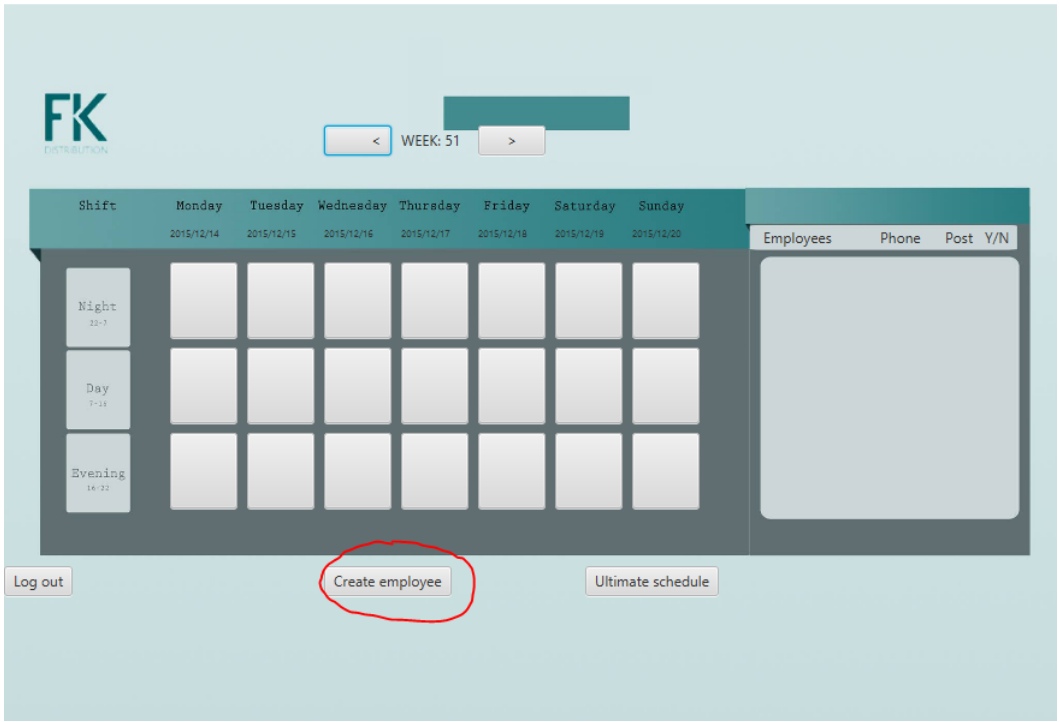
Graphical user interface was made with JavaFX and CSS. We used a lot of build in JavaFX nodes like Buttons , TextFields, Labels, ImageViews but also decided to make some of our own. For example instead of using TableView and DatePicker we build our own interfaces using Buttons, Labels, CheckBoxes. We realised that they would exactly meet our needs and they would be more appropriate. For organising the views we used all different kind of layouts like VBoxes, Hboxes, BorderPanes and GridPanes which we found the most efficient and after some training the easiest to use. We asked a graphic designer to make images for the background of our windows and later we applied those graphics in CSS.

3.6. Manual of application/GUI (Jakub)

1. Creating a profile:

Possibility of creating a user is reserved to the manager only. To register:

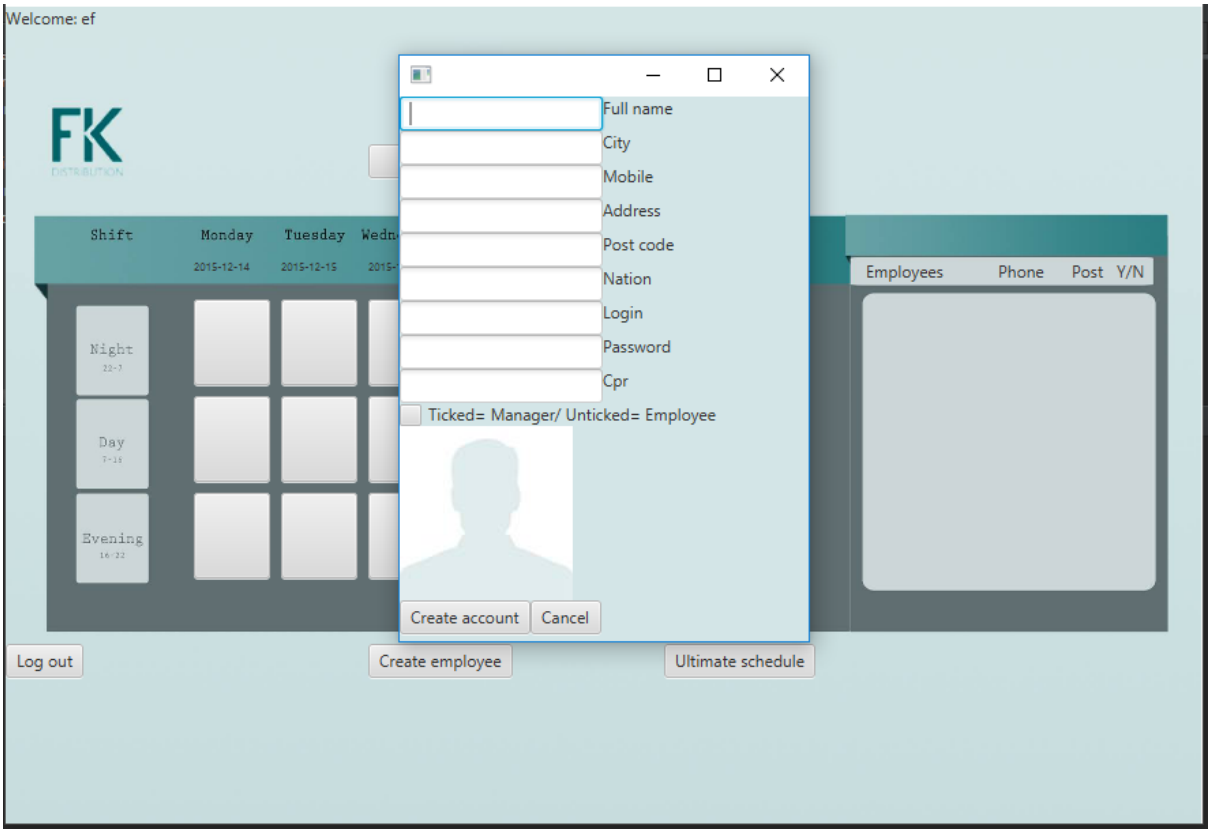
A. Log in with your unique manager credentials supplied by our team.
B. You will be redirected to the *Manager Confirmation Window* where you will find a *Create employee* button in the bottom centre.



C. You will be take to the *Create Employee Window.*

D. Fill out all the empty spaces with the new employee's information.

-To add the employee's photo click on the picture with the person's outline.



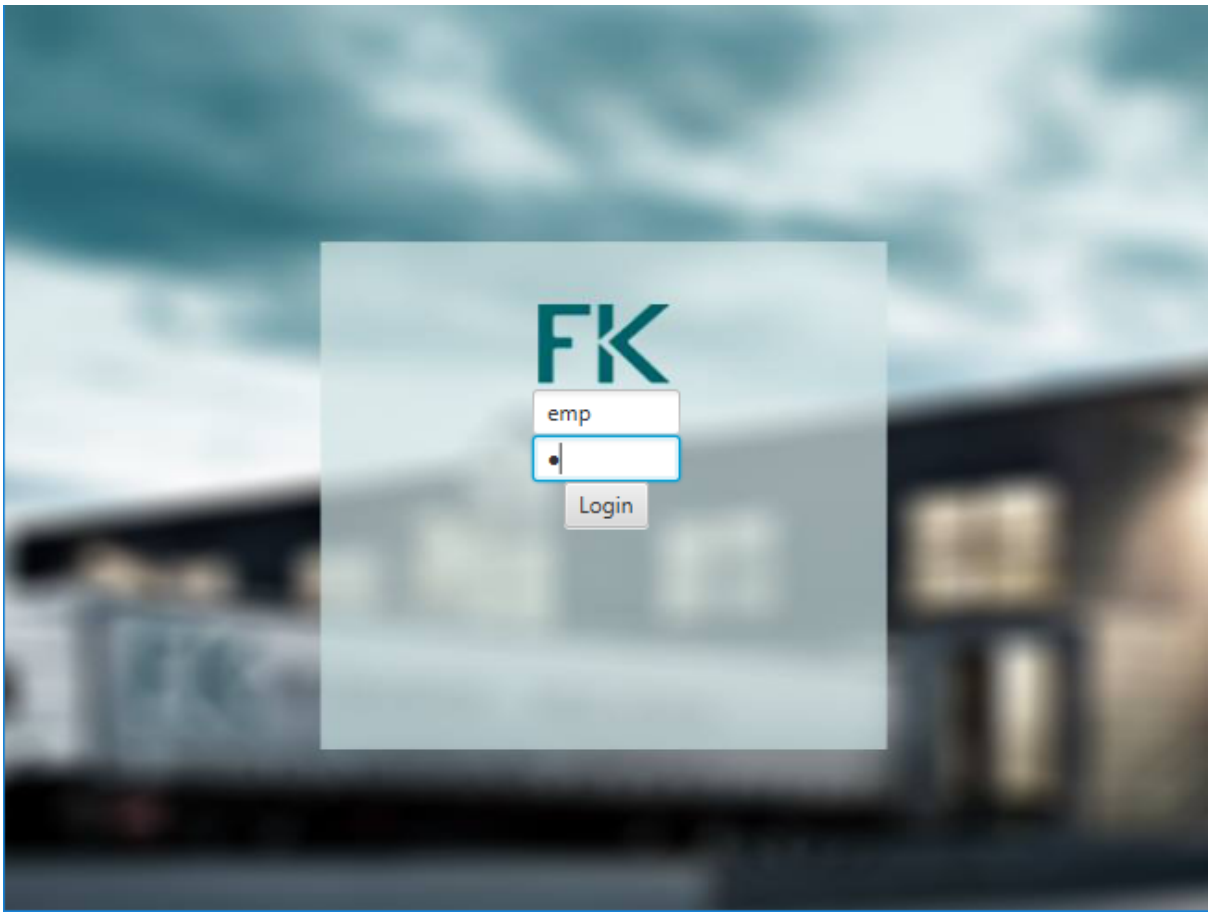
E Click *Create account* button to complete the process.

-Click *Cancel* button to exit the *Create Employee* window. Note that none of the information will be saved.

*To see about editing profile to go section ??

2) Logging in:

a) Type in your login credentials supplied to you by either our team(if you are the first user) or by your manager.

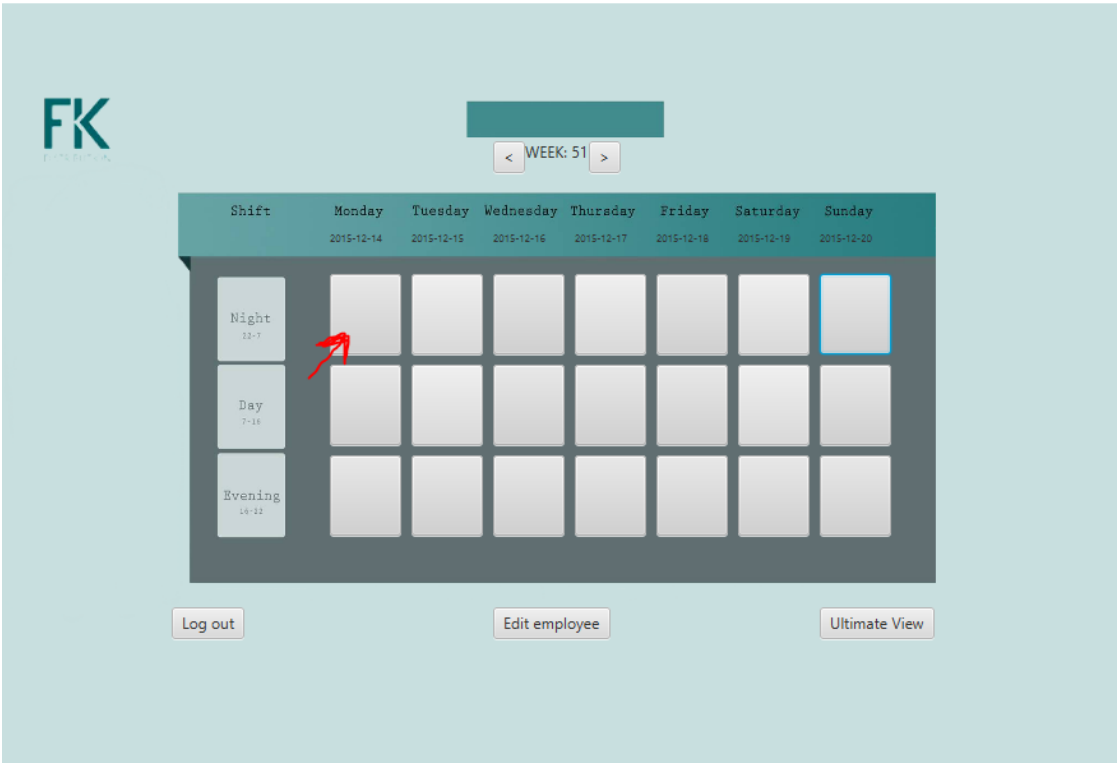


b) Press *Login* button.
c) If your login information was confirmed in the system you will be taken to:

- if you are an employee: *Availability Window*
- if you are a manager: *Manager Confirmation Window*

3) Marking availability in the system:
When you want to apply for a shift at a specific date.

a) Log in as an employee
b) You will be redirected to *Availability Window*

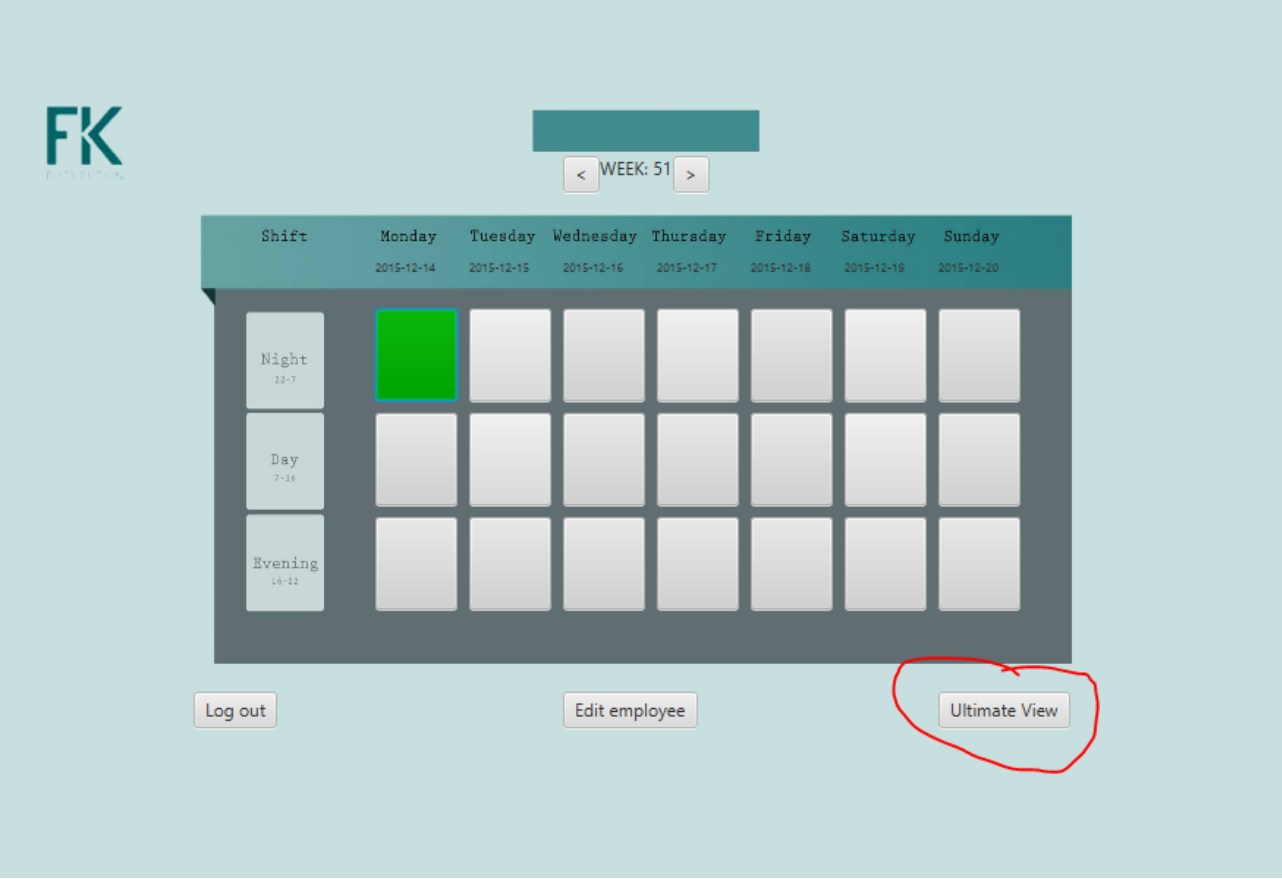


c) Click on one of the buttons representing a given shift.
d) The button will turn green which means you have successfully applied for a shift.

4. Check your schedule:

When you want to see whether your application(s) for a shift were confirmed by a manager or to check who is working on at a given time you need to open *Ultimate Schedule Window*.

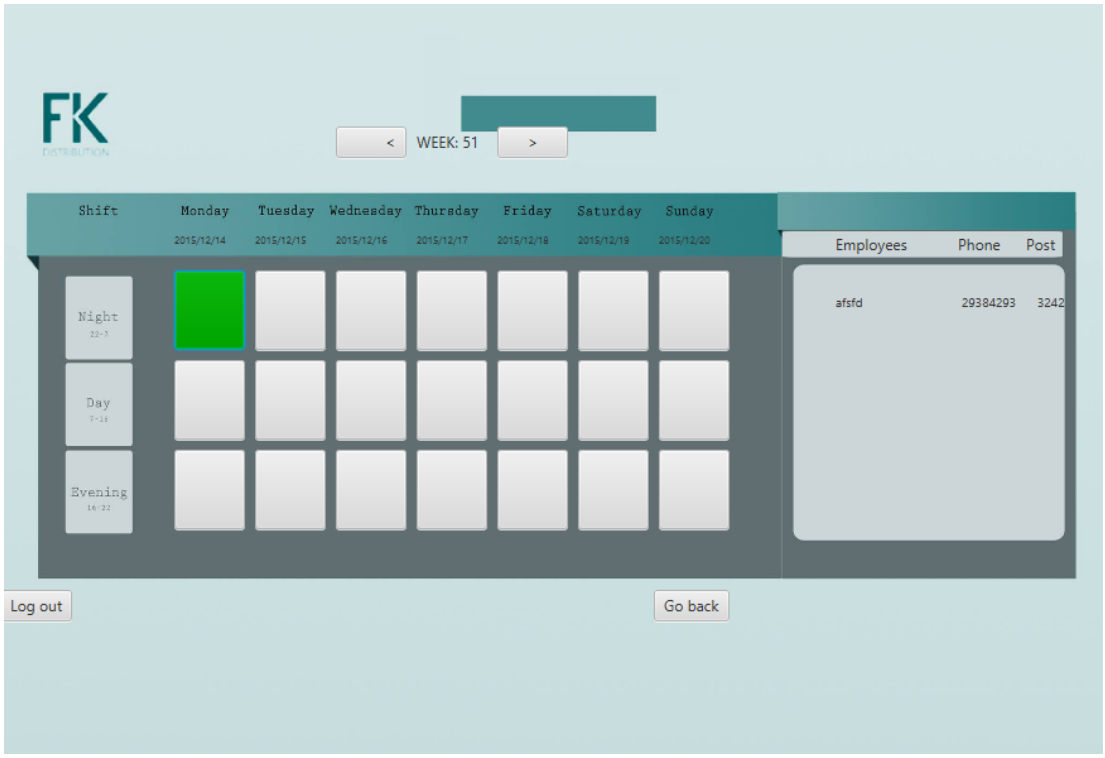
a) Log in
b) You will be redirected to *Availability Window*
c) - if you are an employee click *Ultimate View* button



d) You will be redirected to *Ultimate Schedule Window*

e) - If you were not confirmed on any of the shifts you will only see gray buttons.

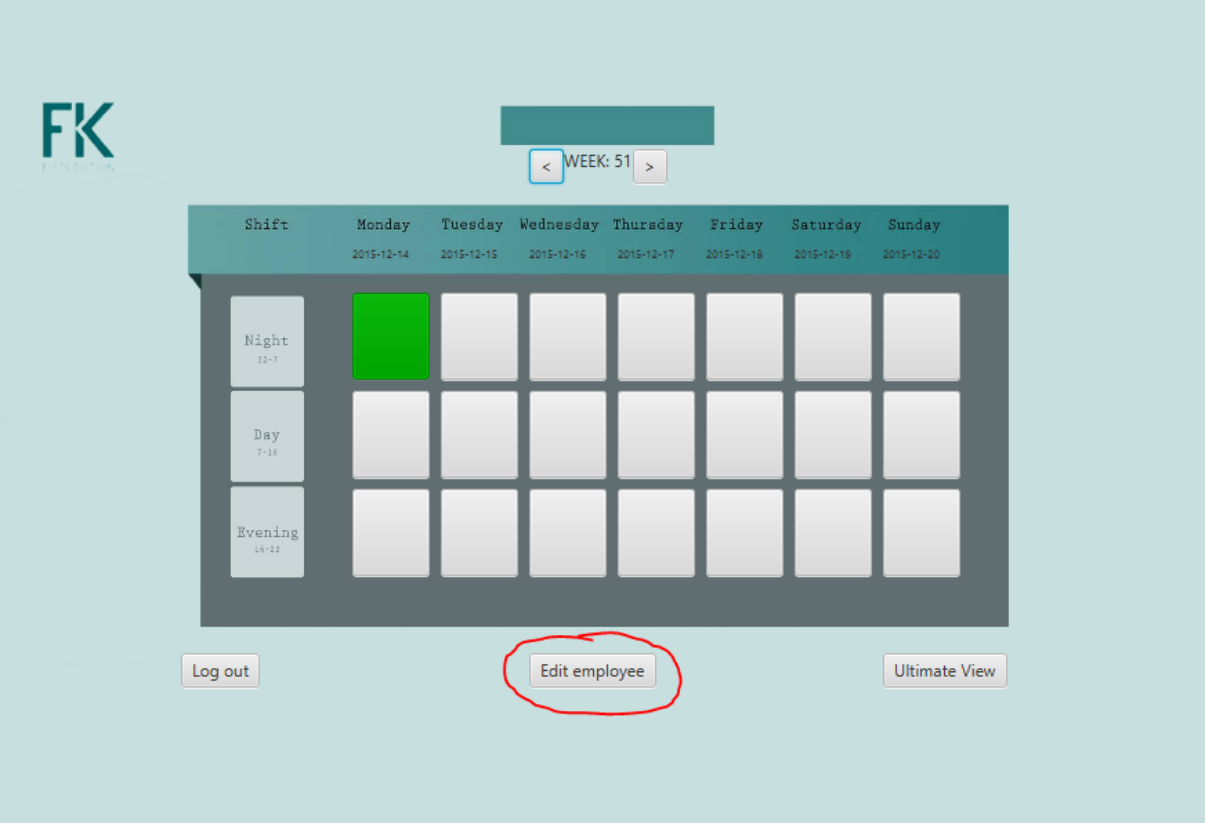
- If you were confirmed the button with your shift will become green.



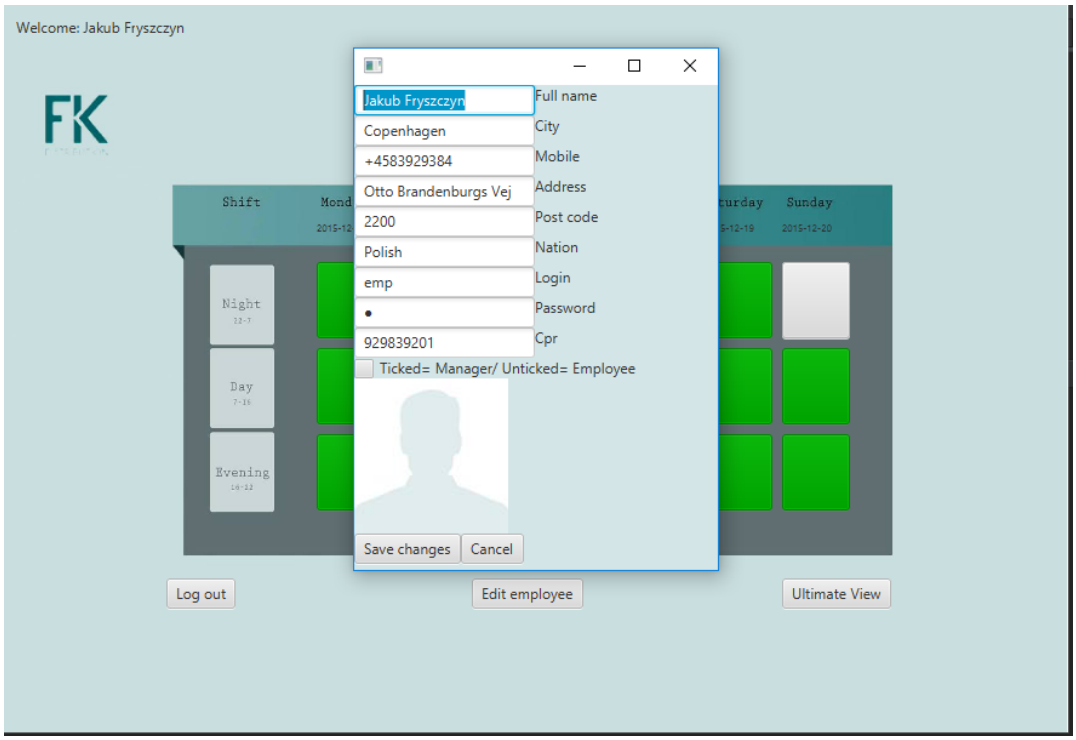
*To see who is available on a given day click the button corresponding to a shift. A list of employees will appear on the right.

5) Editing your profile

a) Log in
b) You will be redirected to *Availability Window*
c) Click *Edit profile* button



d) You will be redirected to *Edit Profile Window*



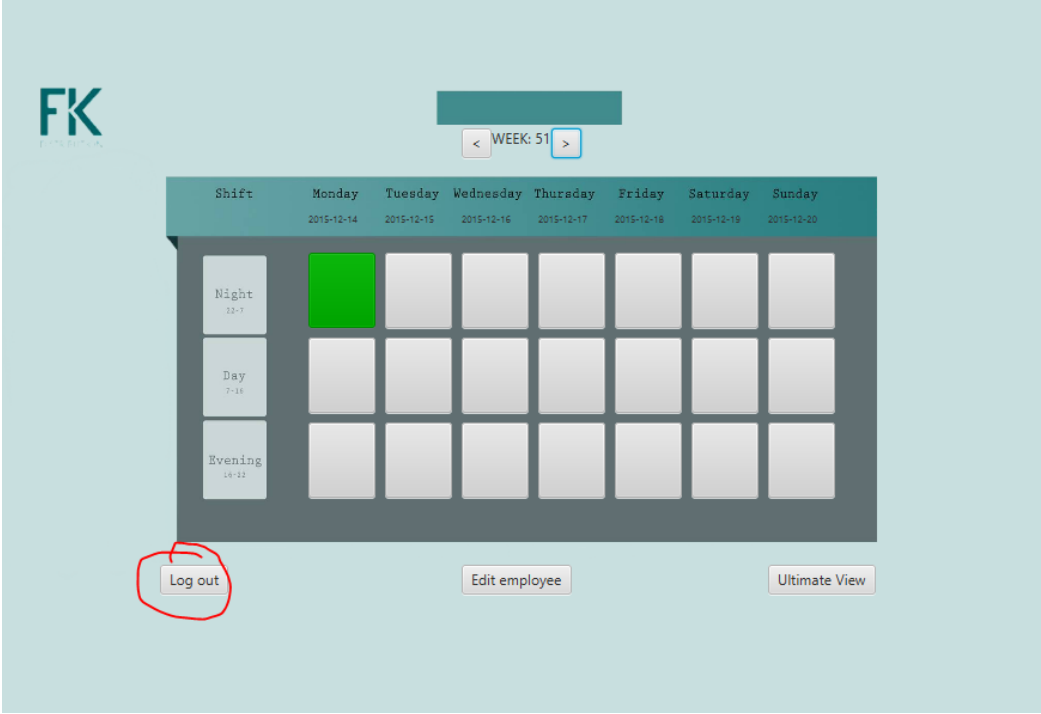
e) Update your personal information.
f) Click *Save changes* button to commit changes.
6. Using the calendar

To change the date use the arrows on both sides on *Week* label

The dates are displayed under the days of the week.

7. Logging out

To log out press the *Logout* button located in any of the windows containing calendar.



4. Osca

4.1. Data types (Tadas, Edgaras)

The following tables are showing all data types and their characteristics according to our database.

Table: “Archive”:

Name:	Type:	Range:	Minimum Bits:	Minimum Bytes:	Maximum Bytes	Reason for choosing:	Data example:
archivedId	Smalint(5)	0- 99999	40	1	5	Best for numbering the row	10256
ifManager	Varchar(1)	0, 1	8	1	1	Fast and simple	1
fullName	Varchar(45)	Letters and spaces: A-Z, a-z, space	360	1	45	Best option for a name	Michael Jordan
address	Varchar(45)	Letters, numbers, spaces and symbols: A-Z, a-z, 0-9, (. - : ,)	360	1	45	Best option for an address	Folehaven 80, st th
city	Varchar(20)	Letters and spaces: A-Z, a-z, space	160	1	20	Best option for a city	Copenhagen
nationality	Varchar(25)	Letters: A-Z, a-z	200	1	25	Best option for a nationality	Danish
mobile	Smallint(8)	0-99999999	64	1	8	Suitable for a Danish phone number	45826514
postCode	Smallint(4)	0-9999	32	1	4	Best suitable for a post code	2200
cpr	Varchar(11)	From 000000-0000 til 999999-9999	88	11	11	Fits best to use for a CPR number	141185-3271
empPicture	Varbinary(8000)	From 1 bit to 8000 bytes	1	1	8000	BLOB is used to store the large objects such as pictures	image.jpg

Table: “Employee”:

Name:	Type:	Range:	Minimum Bits:	Minimum Bytes:	Maximum Bytes	Reason for choosing:	Data example:
archivedId	Smalint(5)	0- 99999	40	1	5	Best for numbering the row	10256
ifManager	Varchar(1)	0, 1	8	1	1	Fast and simple	1
fullName	Varchar(45)	Letters and spaces: A-Z, a-z, space	360	1	45	Best option for a name	Michael Jordan
address	Varchar(45)	Letters, numbers, spaces and symbols: A-Z, a-z, 0-9, (. - : ,)	360	1	45	Best option for an address	Folehaven 80, st th
city	Varchar(20)	Letters and spaces: A-Z, a-z, space	160	1	20	Best option for a city	Copenhagen
nationality	Varchar(25)	Letters: A-Z, a-z	200	1	25	Best option for a nationality	Danish
mobile	Smallint(8)	0-99999999	64	1	8	Suitable for a Danish phone number	45826514
postCode	Smallint(4)	0-9999	32	1	4	Best suitable for a post code	2200
login	Varchar(25)	Letters and numbers: 0-9, A-Z, a-z	200	1	25	Best option for a login name	twillightsparkle
password	Varchar(25)	Letters and numbers: 0-9, A-Z, a-z	200	1	25	Best option for a password	liverpool
cpr	Varchar(11)	From 000000-0000 til 999999-9999	88	11	11	Fits best to use for a CPR number	141185-3271
empPicture	Varbinary(8000)	From 1 bit to 8000 bytes	1	1	8000	BLOB is used to store the large objects such as pictures	image.jpg

Table: “Availabilities”:

Name:	Type:	Range:	Minimum Bits:	Minimum Bytes:	Maximum Bytes	Reason for choosing:	Data example:
availabilityId	Smalint(3)	0-999	24	1	3	Best option for numbering the row	254
date	date	From 1000-01-01 till 9999-12-31	80	10	10	Best suitable for choosing a date	2015-06-22
shiftType	Varchar(7)	Letters: A-Z, a-z	56	1	7	Best option for storing a shift type	Night
empid	Smallint(3)	-999	24	1	3	Best for numbering employees	332
confirmed	Varchar(1)	0, 1	8	1	1	Fast and simple	1