2015

University College of Northern Denmark

Ticket booking console

In the following report, there is an analysis of a modern business and the approach taken to boost their working process through code along with argumentation for each decision. The ticket booking console handles different types of employees with different access simultaneously and saves data regarding ticket bookings for different events. The motivation behind the project is to apply the knowledge acquired for the past year and take notes of the results – in brief – form a better perspective.

University College of Northern Denmark

Nikolay Dinev, Cosmin Gabinat, Iliyan Iliev, Boril Milanov
Group 2, dmai0914, Computer Science
2/6/2015

Table of Contents

1. Introduction

Technology and software take a big part of the personal and the business life of anyone. A large amount of modern businesses feel the need of software, which can optimize and accompany their development. Eventually, it leads to faster integration on the market and fiercer competitiveness among established companies.
To begin, the following is the definition of software:

> *"Organized information in the form of operating systems, utilities, programs, and applications that enable computers to work."* [1]

From a business point of view, it can optimize the workflow; improve product quality, boost information travel, increase security and etc. All this taken together in many cases would lead to faster and bigger profit in the face of less used resources.
In order take advantage of the latest technologies available on the market, each company needs to have access to specialized software.

Aalbornification is a start-up running in the event planning field. Opened in 2009, the company had a limited number of employees and not a lot of exposure.

Over time the company has developed, increasing also their work amount. With an increased number of events to be handled, the need of organizing the internal information has gained their focus, likewise the need of a personalized platform that can give them control over their ticket sales.

Even though the company is already using an online platform for selling tickets, they should be able to benefit of personalized software that can ease their daily work and is also shaped to fit their needs.

[1] http://www.businessdictionary.com/definition/software.html#ixzz3Z5DzcrAB

1.1. Problem statement

Considering the above mentioned information, this project is aiming on formulating and answering a problem statement that will improve the issue of software use in relation to the business activity in Aalbornification.

The problem statement is formulated as follows: "How to design a ticketing platform that will improve the working environment and the ticket booking process for Aalbornification?"

1.2. Methodology

The project follows the unified process (UP) together with the SCRUM approach in order to simplify and divide the whole project as well as to have an organized flux of tasks.

The unified process approach is taking the form of a project plan[2] created in Microsoft Project. However, the actual periods the tasks will be taken might meet a slight difference from what is initially planned, due to unexpected events that might occur during the process.

Also due to the lack of a permanent physical working space, the tasks list specific to the SCRUM model will be held on an online platform called Trello. This will help the team keep track of the tasks and deadlines that are to be met.

The programming language used for the development of this software is Java, as well as MS SQL used for the relational database.

The first step of this report is the collection of data and information from the company manager and the employees. Following data collection and analysis, the project can proceed if it proves to be feasible.

The second step is to design the domain model and the related diagrams prior to beginning the implementation of the code.
Meanwhile the first set of mock-ups will be created and a think aloud test will be carried in order to test the learnability and accessibility of the design.

2. Company overview

The Business Factory is an umbrella company opened in 2009 that has no field of activity, but as the name suggests, it creates new startup concepts for possible future companies. The company has a limited number of permanent employees, as it does not have a constant flux of events. When new concepts for events are created, the company hires the required amount of part time personnel in order to meet the workloads.

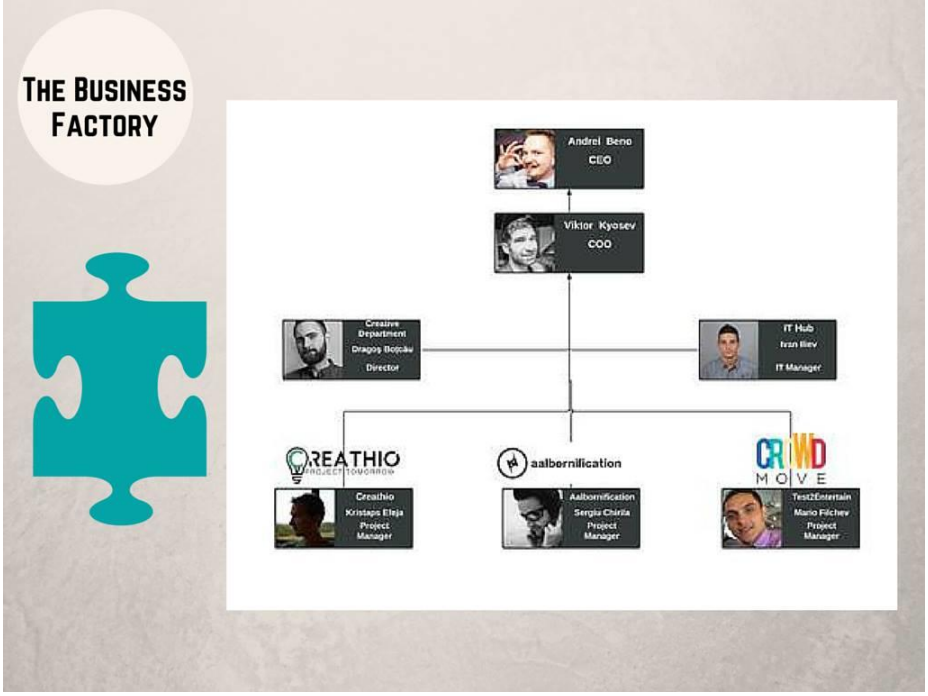
Currently the umbrella company has three fully functional companies (Aalbornification opened in 2011, Creathio -2013 and Crowd Move in 2014), and a total number of 7 permanent employees. Two of them are the owners and creators of The Business Factory, Viktor and

[2] See last page of the appendix

Andrei. One employee is representing the IT department and the other - the creative department, which handles the video editing (FX. Creating promotional videos or after movies of the events) and other designs. From the remaining employees one is assigned for each of the companies. In this way only one of them is assigned as an employee of Aalbornification.

2.1. Description of the organizational structure

"An effective structure facilitates management and clarifies relationships, roles and responsibilities, levels of authority, and supervisory or reporting lines."[3]



The three sub companies are considered as departments of the main company. In the graphical representation, the three persons running the sub companies are called "Project managers", meaning that they are responsible for the projects running in each company. Thus, the main company is considered to have a *simple, functional structure*, because of being divided into smaller groups based on specialized functional areas - the IT, the creative departments, as well as the other three different sub companies. Each of these can hire part time staff for the

[3] http://www.pathfinder.org/publications-tools/pdfs/Strengthening-You-Organization-A-Series-of-Modules-and-Reference-Materials-for-NGO-and-CBO-Managers-and-Policy-Makers-Organizational-Structure.pdf

running projects, while the CEO and COO are also taking part actively in the projects of each of the companies.

2.2. Evaluation of organizational structure and problems

> *"By reviewing an organization's structure, a manager will be able to determine which human, financial, and technical resources are available, how they should be allocated, and which resources are lacking."*[4]

Evaluating the organizational structure is an important step in discovering the problems that currently occur, or may occur in the company. Having this information can shape the way the solutions and suggestions will be developed in this report.

Considering the limited amount of permanent employees, the organizational structure fits the company, as it allows it to work on different projects at the same time, each department being specialized in their own field. However, if the sub companies develop enough, they will need to increase the number of their permanent employees, thus the structure will require a change to Mintzbert's divisionalised form of organizational structure.

> *"When a company creates an organizational structure by grouping positions into departments based on geography, product or customer, the company is using a divisional structure. Divisions function independently of one other. They also govern themselves with much autonomy, relying on the parent company in only limited ways, such as for resources."*[5]

This might happen if one of the company's demands exceeds the current power of delivering. In this case:

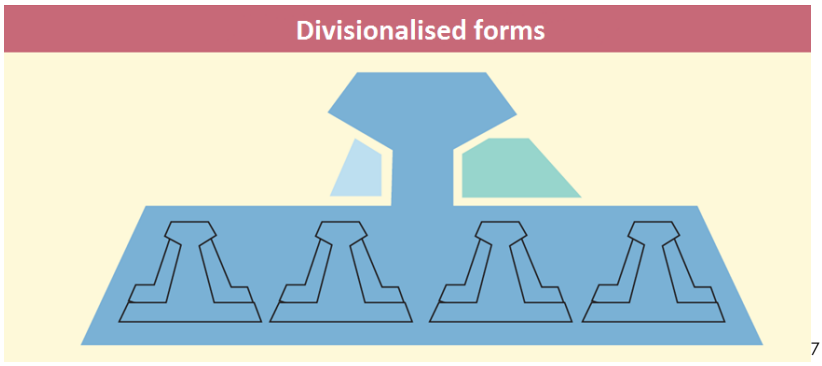
---

[4] http://www.pathfinder.org/publications-tools/pdfs/Strengthening-You-Organization-A-Series-of-Modules-and-Reference-Materials-for-NGO-and-CBO-Managers-and-Policy-Makers-Organizational-Structure.pdf

[5&6] http://yourbusiness.azcentral.com/divisionalized-organizational-structure-4924.html

"*This autonomy means that each division must include people able to perform different business functions, all concentrating exclusively on the objectives of their own division*"[6]

. The advantage is that the company can focus better on their specific demands.



[7]

2.3. Style of leadership

Each individual is unique, this is why it is important to analyze each style of leadership. Their styles can influence the way a company is run and the way the employees perform their work. Analyzing and understanding their leadership styles can facilitate the process of shaping a solution specific to their needs.

With a CEO and a COO, the main company has two leaders, both being involved in managing and leading the sub-companies. However, when Aalbornification is hiring extra staff for their projects, the new employees have one more manager, which is the project manager of the sub company.

Andrei has a previous BA in International Hospitality Management and is currently taking an MSc in entrepreneurial engineering. His main focus is on the business part and the entrepreneurship. He is in charge of business development, finance, funding and defining strategies due to his education and position of CEO.

On the other hand Viktor has also a BA in International Hospitality Management, and is currently studying MSc in Culture, Communication and Globalization, so he is more people focused. He is COO (chief operating officer) and as he says:

"*I deal with HR, creating pipeline of talent (influencing attraction and retention), designing a framework to implement strategy into operations, coaching and mentoring. Anyway, both me and Andrei deal with all sorts of things, I am also in charge of B2G and to large extend B2B and many other things such as Social Media marketing, sponsorships, fund-raising and so on*".

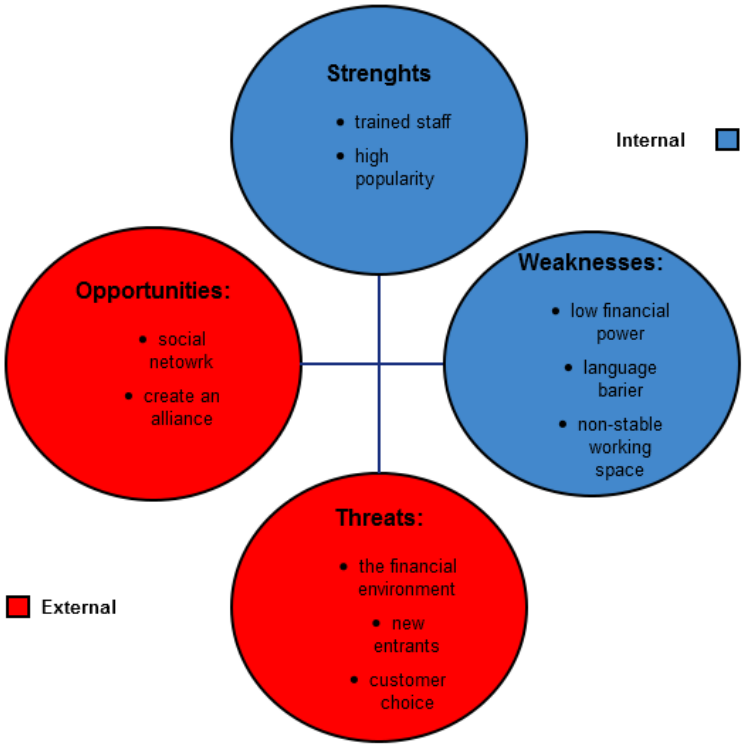
7

2.4. S.W.O.T. analysis

S.W.O.T. analysis is a strategic tool to identify a business. Each category has its pro's and con's.

The S.W.O.T. categories are divided into internal (strengths and weaknesses) and external (opportunities and threads). "Strengths" is the category, which makes the company understand what keeps them stable and running. "Weaknesses" is not necessarily weaknesses, as it aids tracking down the fragile points, which can be turned around as strengths as long as precautions are made. It can be even a type of lure for opposing companies. "Opportunities" is strengths that have not yet been reached. "Threats" are inevitable problems that cannot be dealt with fully unlike "Weaknesses" - they are constants and are outside the company's reach.

Below is presented and described the SWOT analysis of Aalbornification:



2.4.1. Strengths

The main strength of the company is the industry related training of the staff. All of the employees are currently or have been previously enrolled in a management or another similar education, most of them at UCN Aalborg. As mentioned in the style of leadership paragraph, the CEO and COO have a previous education in International Hospitality Management, and chose to continue their professional development in a related field of studies.

Since the opening of Aalbornification in 2011, the company quickly became very popular among the international students due to the student parties they were organizing. Later they developed and the events they were organizing became bigger and bigger, catching the interest of the locals. Nowadays their events are targeting both locals and internationals attracted by their distinct events organized in and around Aalborg.

2.4.2. Weaknesses

As most of the full time employees at Aalbornification, including the CEO and COO, are currently students. This means their financial possibilities are limited, which impacts the ability to invest in the company. This is why they must rely a lot on sponsorships to manage to organize their events.

The next weakness is the language barrier. As mentioned above, the company relies on sponsorships, and considering the language barrier, this can sometimes be hard to obtain in the conditions that none of their employees can carry a presentation in Danish for the possible sponsors.

Due to the low financial power, the company did not have a stable office where they can carry their daily work. Over the last years, the company changed the location of their office a few times, every time their employees were faced with change and required to reorganize their working environment. This represents a weakness for the company, until they find a stable working place.

2.4.3. Opportunities

Considering the weaknesses, an opportunity that Aalbornification could benefit of is to create an alliance with a local company. Hence they can facilitate their access to the local market and can diversify their range of offered events.

The company managed to turn the use of social network into an opportunity as they promote their events online and most of their event guests are getting informed from there. Thus, the use of social media is one of the main ways of promoting their events.

2.4.4. Threats

The threats that Aalbornification is facing are the threat of an unfavorable customer choice, meaning that the customers might choose to attend other events instead of the events organized by them. Another threat could be represented by new entries on the market.

The current situation of the financial environment could represent a threat for the company as the financial crisis has impacted negatively the buying power of the people, as well as the investment power of possible sponsors, on which the company is highly dependent.

2.5. Mission, vision & values

Having formulated a mission and vision statements helps the company to clearly determine their purpose to both the public and their own employees. Aalbornification has already clearly defined their mission and vision.

As found on their official website, it is cited:

> *"MISSION - Aalbornification is an event planning agency that delivers tailored services to our clients. We transform corporate and social events into experiences, by creating a fusion between the exclusive and the entertaining."*[8]

Furthermore they have defined their vision in the following statement:

> *"VISION – Our vision is to become the leading provider of entertainment on the Danish market by enriching the social and music culture through international influences."*[9]

They present a very specific set of values that are important in the company. These values are also present on their website in the following graphical representation.

10

8&9 http://aalbornification.dk/about-us/

3. Business case

3.1. Introduction and background

The problem description of this project has been thoroughly shaped in the beginning of this document, giving a clear goal for this project.

3.2. Management summary

As the problem statement mentions, this projects focuses on the development of computer software that is meant to improve the ticket handling and booking processes for Aalbornification. Having a clear company overview makes it easier to begin the project.

It is focusing on developing a solution to the problem statement, in form of software fitted to the company's needs. The software is meant to include all the functionalities required in connection to the handling and booking of tickets, for different events that the company is organizing.

The proposed software is intended to give a better overview and more control over the information regarding the tickets, events, employees and other relevant areas. At the same time it will aim to increase the speed and simplify the booking process. The software will be available only to the employees, and is not intended for customer use.

In terms of business, the implementation of such software is expected to benefit the company by raising the employee satisfaction rates and simultaneously influencing the sales and revenue.

3.3. Description of problem

The subject of this project is not necessarily a problem for the company, but actually a suggested improvement in the way they carry out their work. Currently the company uses the online environment for the booking process and this is not meant to be replaced. Through this project, the company is offered an opportunity to improve their work, allowing their employees to have access to software fitted to their daily work, making their work easier. Beside the online booking option that is already in use, they can explore the opportunity of having personalized software that gives control over the data they work with, as well as allowing them to also make bookings for their customers (taken by phone or email).

---

10 http://aalbornification.dk/about-us/

3.4. Cost/benefit analysis

Even though the access to the current financial situation of the company is limited, this analysis will present only the project related costs and benefits, divided into tangible and intangible.

The tangible costs resume to the value of the physical systems required to run the software. This will include the cost of a server that will host the database entries. The company has to pay the initial cost as well as the maintenance of the system. Alternatively the database could be hosted at an online service, the cost being reduced, but at the same time also weakens the control over the data.

Intangible costs may consist of the employees' possible resistance to the new software. Regarding the benefits of the project, it can be mentioned the employee satisfaction, expected to be immediate. This can automatically help bring in an increase in sales, expected to be a long-term benefit.

3.5. Risks

Despite the project aiming to develop user-friendly software meant to ease the work of the employees, certain risks still exists. One specific risk may be the inability of some of the employees to adapt to the new software, or for this matter the resistance they may present when required to adopt the new software.

3.6. Conclusion

All in all, the benefits of the proposed investment are expected to weight more on long term than the value of the investment. Furthermore immediate changes should be noticeable as soon as the proposed software will be implemented.

Considering this, this project proves to be feasible and the company is encouraged to pursue its development and the implementation.

3.7. Use case diagram

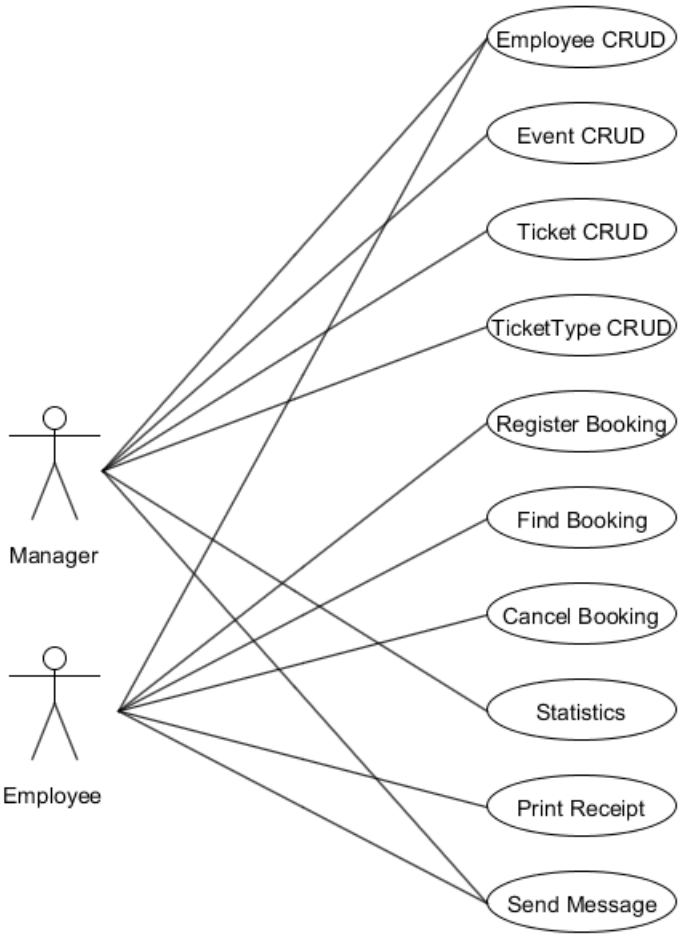
> *"A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. "*[11]

The importance of the use case diagram is quite big, allowing the developers to understand the actual functional requirements of the software to be developed.

Below is represented in UML (Unified Modeling Language), the use case diagram for the present project.

As seen in the diagram, the proposed software includes eight use cases. Two of them represent the CRUD functionality of the employee and event, objects that are required to register a booking.

After a review, two more use cases have been added to the diagram. These include the ticket and ticket type CRUDs. These objects are also included in the registration of a booking.



[11] http://whatis.techtarget.com/definition/use-case-diagram

4. Quality and Acceptance Criteria

When a system is going to be accepted, it is important that requirements set by the customer are met. Accept/done criteria can be set up for the Functional/Non Functional requirements in order for the developer to know when a requirement is covered[12].

4.1. Functional Requirements

The functional requirements should be measurable, the accept or done criteria should be set up for each use case step[13]:

- System actions
- System reactions from user input

4.1.1. Employee CRUD use case

Considering the above, the following functional requirements criteria were made up for the Employee CRUD use case :

1. Use Case Step : Create Employee
   - Must hold the following information : fname (First name), lname (Last name), email, phone, cpr, password, tokens, isManager (whether the employee is Manager or not), company
   - As minimum required fields to be filled : fname, lname, cpr.
   - fname – numbers are not allowed, lname – numbers are not allowed, cpr – must contain no less and up to ten 10 digits.
2. Use Case Step : Read Employee (find)
   - Must hold the following information : fname, lname, email, phone, cpr, tokens, isManager, company
   - As minimum required fields to be filled : fname
   - fname – numbers are not allowed, must be a valid record
3. Use Case Step : Update Employee (This step of the use case is available only for employees who are not managers)

[12]&[13] See Ann's slide “Quality Criteria and Review” on ecampus

- ❖ Must hold the following information : fname, lname, email, phone, password
- ❖ As minimum required fields to be filled : At least one of the mentioned above
- ❖ fname – numbers are not allowed, lname – numbers are not allowed, email – must follow the required email pattern

4. Use Case Step : Delete Employee (This step of the use case is available only for employees who are managers)
    - ❖ Must hold the following information : fname, lname, email, cpr
    - ❖ As minimum required fields to be filled : fname
    - ❖ fname – numbers are not allowed, must be a valid record

4.1.2. Booking handling use cases

1. Register Booking
    - ❖ Must hold the following information : bdate (Booking Date), emp (Employee object), tt (TicketType object), quantity, total, gname (Guest name), gphone (Guest phone), gemail (Guest email), payType, status.
    - ❖ As minimum required fields to be filled : bdate, emp, tt, quantity, gphone, payType, status.
    - ❖ bdate – can't be invalid since it is taken from the system, quantity – must be in the range from 1 to 10, gphone – must contain only digits, payType – must choose from the presented types “credit card, cash, tokens”.

2. Find Booking:
    - ❖ Must hold the following information : bdate, emp, tt, quantity, total, , gphone, payType, status.
    - ❖ As minimum required fields to be filled : gphone
    - ❖ gphone – only digits are allowed, must be a valid record

3. Cancel Booking (only available after Find Booking is executed).
    - ❖ Must hold the following information : bdate, emp, tt, quantity, total, , gphone, payType, status.
    - ❖ As minimum required fields to be filled : gphone
    - ❖ gphone – only digits are allowed, must be a valid record

4.2. Nonfunctional requirements

Can be classified as critical (If Non Functional Requirements are not met, the system might be unusable), hard to make measurable, often related to the system as a whole instead of a single functionality.

After consultation with the customer, the demanded nonfunctional requirements for the whole system were clearly set as follows:

- Performance – Multiple concurrent users are using the system, while it is on normal operation. Response time is less than 2sec.
- Usability – Using the heuristics in order to group similar fields, makes the interface really easy to use. Tooltips were added to guide the user if he/she finds it difficult to understand the next step which has to be done. Messages after every action performed allow seeing if certain operation went well or something went wrong, also preventing crashes of the software.
- Security – Security is held by using Prepared Statements in order to protect the data that is being manipulated.

5. Domain Model

> "*The Domain Model is a representation of real-world conceptual classes, not of software components. It is not a set of diagrams describing software classes, or software objects with responsibilities.* "[14]

Built by the "Candidates for Classes" table, a well composed Domain Model consists of:

- Domain Classes – *"Each domain class denotes a type of object*"[15]
- Attributes – "*An attribute is the description of a named slot of a specified type in a domain class. Each instance of the class separately holds a value*"[16]

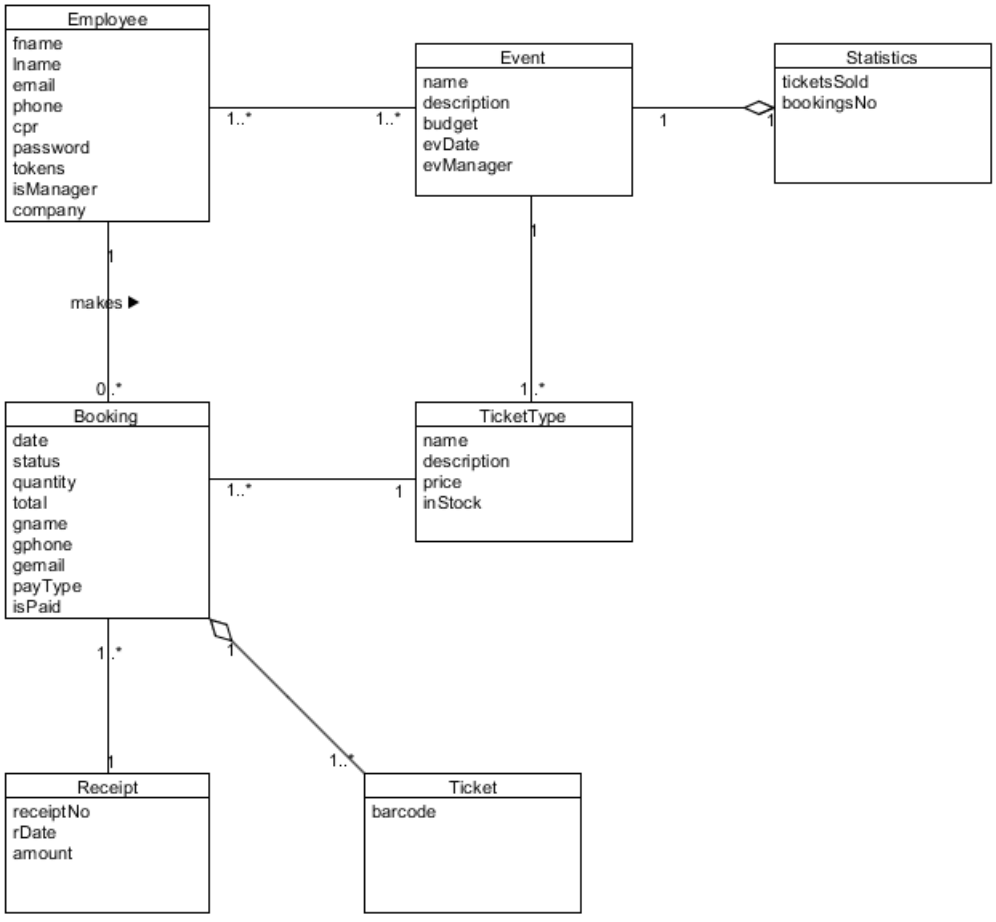
[14] Applying UML and Patterns Craig Larman

[15] Business Modeling with UML - Penker

[16] Business Modeling with UML - Penker

- Associations – "*An association is a relationship between two (or more) domain classes that describes links between their object instances. Associations can have roles, describing the multiplicity and participation of a class in the relationship*"[17].

The domain model is an essential part of the Software Development Cycle, like it is mentioned above, it represents classes in the problem domain, shows information about these classes in the form of attributes, and shows the connection between them. Widely used by software developers to communicate with the user in order to show the business logic behind the future software.



Considering the "Candidates for Classes" table, the domain classes were made. Following the agreement pattern the associations were considered and attributes were given to each class. For the project's most important use case is "Register Booking", the group is aware that, there is no

[17] Business Modeling with UML - Penker

need in including “BookingLine” class (which will allow in the future to add more than 1 ticket type in a single booking), because of the idea to keep everything of the process realistic.

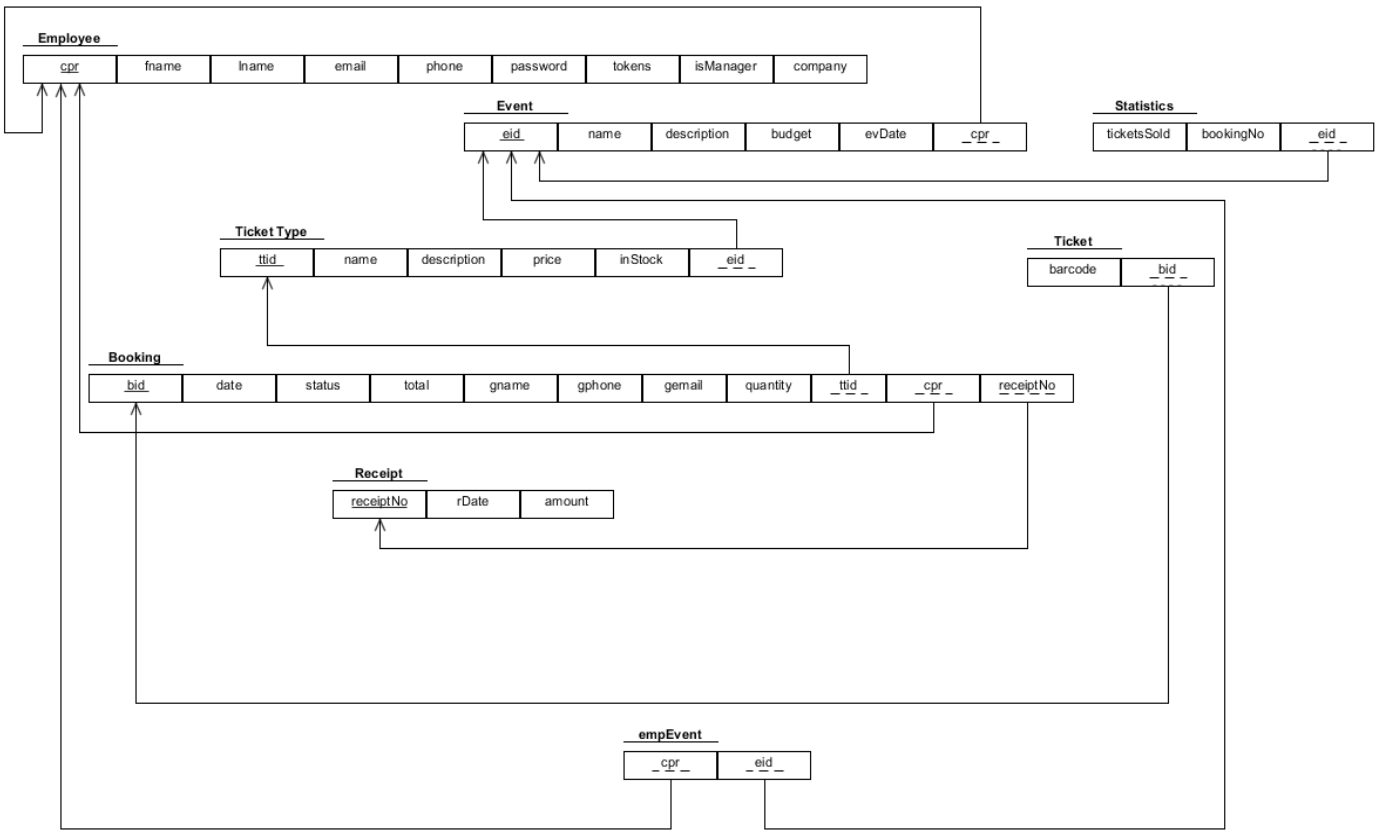
6. Relational Model

The Relational Model is a formal model of a relational database, based on a set theory, and represents the database as a collection of relations.[18]

Made by transforming the Domain Model, the Relational Model is used in order to build functioning databases.[19] In order to transform the Domain Model in Relational Model a few steps must be done:

1. Each class must be mapped to a relation schema
2. Class name is used as relation name
3. Attributes are columns

- ❖ For each attribute consider: Domain(type), NULLs, Uniqueness(key)
- ❖ Primary and Foreign Keys



[18] & [19] Ann's Slides, Transformation to RDB, Session 5

Following the Domain Model, attributes became columns and primary keys were considered for each table, where in some cases "Dummy Keys" were used, because the lack of uniqueness of the attributes.

Example: Table Event – "eid" was added as a Primary Key, because none of the other attributes guaranteed uniqueness. The whole purpose of such an attribute is to give each Event (in this case) unique identifier, which later on will be used as a parameter in some operations of the software.

Considering the multiplicity and associations/aggregations, foreign keys were added to each table.

- One-to-many (1-n): Include the primary key from the one-side on the many-side as foreign key
- Many-to-many (n-m): Create a new table with the primary keys from both sides as foreign keys. The combination of the two foreign keys becomes primary key in the new table.
- One-to-one (1-1): Include the primary key from one of the sides on the other side as foreign key.

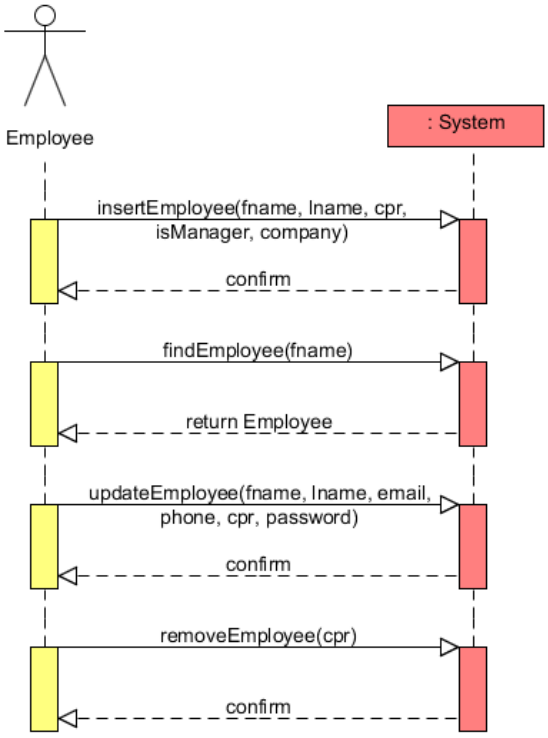
The only one-to-one association (Event – Statistics) , has been decided to give the foreign key from Event to Statistics (eid), in order to minimize the NULL values afterward.

7. Employee CRUD

Following the UP process, the project handles the system development, implementation and testing for one use case until finalisation, before proceeding with the next use case. The first handled case has been the Employee CRUD.

7.1. System Sequence Diagram

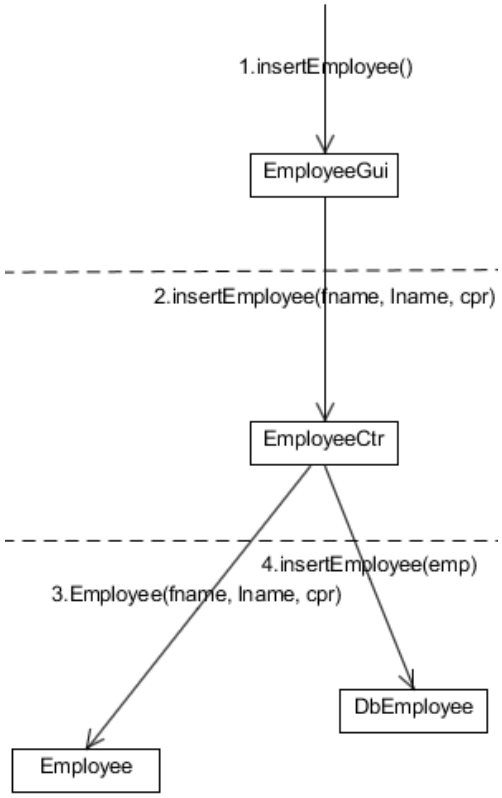
The system sequence diagram (SSD) gives a better understanding of how the Employee CRUD is intended to work. In each of the four operations, the user is expected to provide one or more attributes, and a confirmation is provided in return. In the case of finding an employee, the confirmation is actually the required employee object. If the employee with the specified name is not found in the database, the return of the method will be null.



The operation contracts explain more detailed each operation, providing pre and post conditions required for the operation to be successful.

7.2. Interaction diagram

The below interaction diagram shows how the insertion of the employee takes place. In an earlier version of the program, from the manager tab, the user has to fill in the first name, last name and cpr number in order to create a new employee. Later, in order to create an employee, also a company, a project and a Boolean value are required. In the final version, in the GUI layer, the register employee method is calling the **insertEmployee** method passing **fname**, **lname**, **cpr**, **company**, **project** and a **boolean isManager**.



The **EmployeeCtr** has the duty of creating an employee object with the attributes that were passed from the GUI layer.

Finally after creating the object, a transaction is started, and the **insertEmployee** from the **dbLayer** is called passing the new employee object as a parameter. An integer "res" is returned from dbLayer to ctrLayer and up to the guiLayer, in order to verify if the transaction has been successful or not. According to the value of the integer, a message is displayed to the user.

Finding an employee has a *similar structure*, with the difference that from the guiLayer, what is passed is a String "fname" which is used in the query in DbEmployee to find the employee with the specified name. Considering that the program is intended for a number of approximately *10 users*, there is a *low risk of running into duplicates* of fname that might cause problems. *In case of a larger number of users*, a method that finds the employees by cpr would be more appropriate in order to avoid any confusion.

Once created and inserted in the database, the employee can log in with the cpr number and the default password, and then he/she can update their contact information from the User Tab, adding an email address, a phone number and other projects that he/she might have worked on.

The removal of an employee takes place also in the manager tab, where the employee is removed by the first name. Again, an object is created in the controller, which is passed to the DbEmployee.

7.3. Unit testing

The JUnit test class EmployeeTest is aiming to test the insert employee method. One problem that arose here was the event required for the registration of a new employee. When the "many to many" relation between the employee and the event had been implemented, the full constructor of the employee required also an event, to add to the EmpEvent table, in the database. In order to avoid creating a new event when testing the insertEmployee method, a conditional statement has been implemented in the dbEmployee class that checks if the event associated with the new employee is null. In this case, the association will not be introduced in the database. Below is the block of code that takes care of the condition mentioned above.

```
if (res != -1 && emp.getEv() != null) {
    DbEvent dbEv = new DbEvent();

    res = dbEv.insertEmpEvent(emp.getEv(), emp);
}
```

8. Booking handling

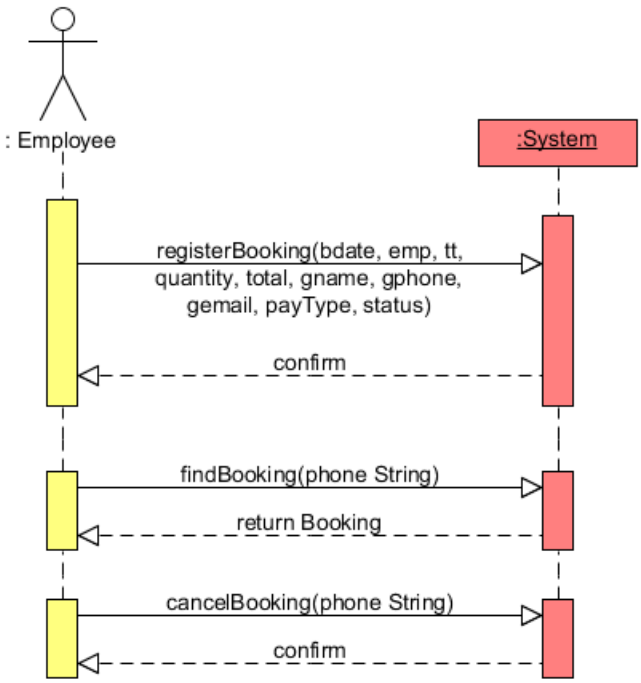
The booking handling group of use cases (registerBooking, findBooking and cancelBooking) are the core of this project. The reason why it is the most complex use cases is due to the amount of data being transferred in and out and the amount of classes in each layer that are required in order to achieve its purpose. Despite this low coupling and high cohesion were achieved.

8.1. System Sequence Diagram

When taking in mind the previous experiences from the first semester until now and the SSD diagram, there is nothing that is out of the ordinary in Booking handling.

The register, find and cancel methods are there with their respective attributes and objects being passed. Briefly after each command done by the user, there is a confirmation, which is either a message or a result.

In this program, there is no update booking, because the user can always cancel / delete it and make a new booking. Likewise the information needed as input by the user is not a lot, therefore there was no use for it as it makes no difference in time consumption.



8.2. Interaction diagram & code implementation

The register booking command starts by being called by the user on the BookingTab, from there it passes the two objects (employee and ticket type) plus the bundle of attributes (bdate, quantity, total, gname, gphone, gemail, payType and status) to the BookingCtr. (1. - interaction diagram)

The validation of the name, phone and e-mail of the given guest is done with numerous if and else statements along with the aid of the **HelperClass**, which has boolean validation methods with their own specific patterns. The rest of the statements check the amount of stock and whether the amount is more or an invalid amount. Below you will see an example of the collaboration between the two classes.

```java
if(gemail.trim().length()>0 && !HelperClass.emailIsValid(gemail)) {
    JOptionPane.showMessageDialog(null, "Email is not valid. \nPlease enter a valid email");
}
else {
    int quantity = Integer.parseInt(btTicketQuantCB.getSelectedItem().toString());
```

```java
public static boolean emailIsValid(String email) {
    Pattern p = Pattern.compile("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}");
    Matcher m = p.matcher(email);

    return m.find();
}
```

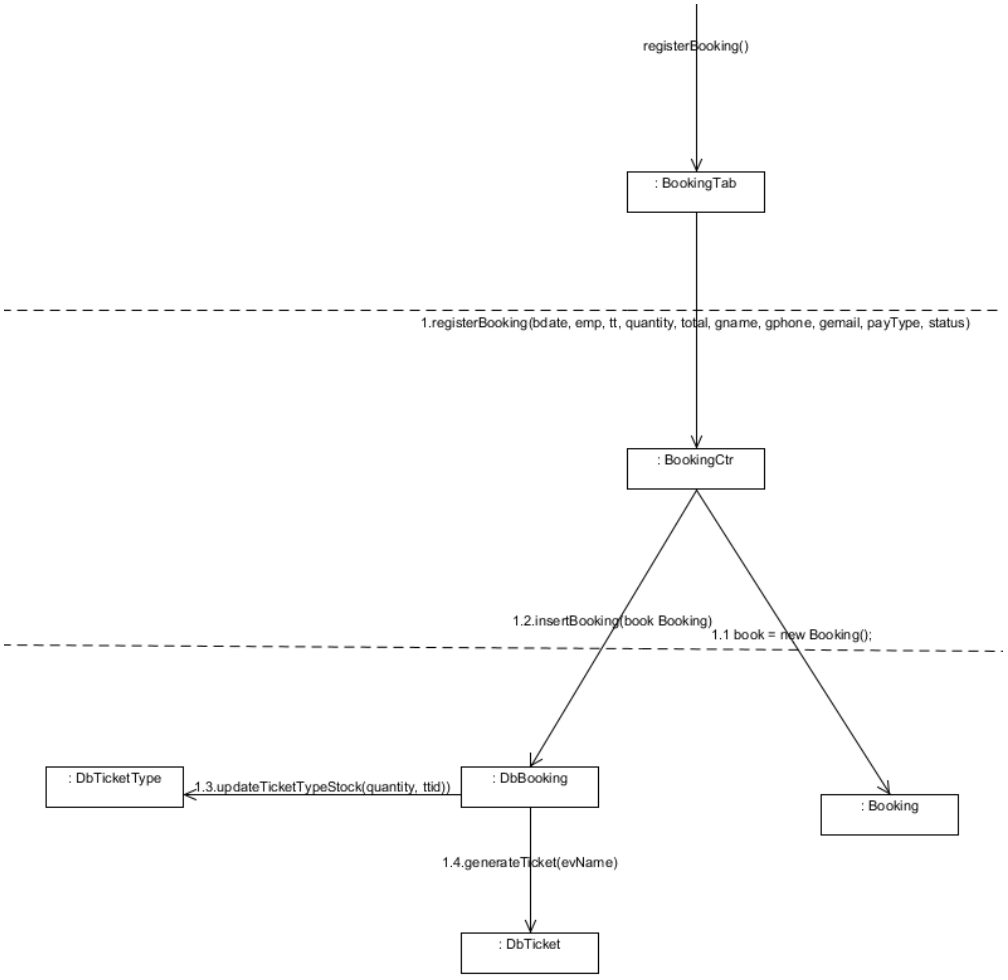
If everything has passed smoothly the program will show a message, notifying the user of its success in registering the booking, if not – vice versa.

After reaching the control layer successfully, a new booking object is created. It takes what is been passed on by the GUI with a bunch of set methods. (1.1.)

Then the program tries to establish a connection with the database, so it can start the transaction (1.2).

The passing of the booking happens with a prepared statement for security purposes, so there is no risk of a memory leak.

Finally, the stock gets updated accordingly (1.3) and the ticket(s) receives a randomly generated barcode (1.4).



9. Design & Implementation

> *"A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP)."*[20]

Due to the large size, it has not been possible to include the whole design class diagram at this point. In this chapter the diagram will be broken into pieces and explained. However, this diagram can be found in full attached in the appendix.

First of all, the program is implementing the three layered architecture, divided into four packages as follows: the graphical user interface (GUI), the control layer (ctr) and the last layer is shared by the model and database packages. The ticketing software includes more CRUD use cases, but in this chapter, for the purpose of explaining the design of the program, only the employee CRUD will be included.

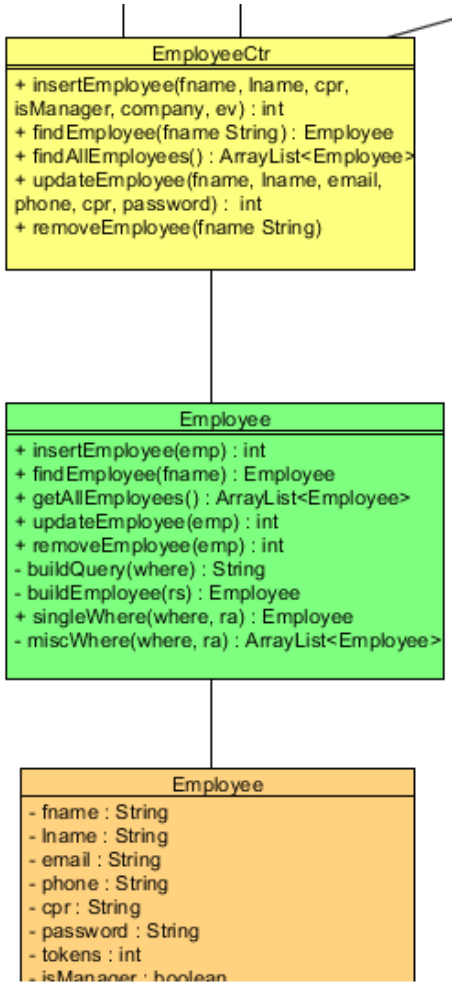
On the right side is represented a piece of the design class diagram. In this graphical representation, it can be seen that the employee CRUD functionality is divided between two classes of the GUI. These are the user information tab, which is making use of the findEmployee method, in order to display the user information for the logged user, and also it gives access to the update functionality, if the user chooses to update his/her information. The second tab that is part of the employee CRUD, is the manager tab, which allows the user(manager), to register a new user, or remove an existing one.

Another class that has access to the employee controller class, is the MainWindow class. However, this only happens due to the lack of the implementation of a login window that can hold the name of the logged user.

---

[20] http://searchsoa.techtarget.com/definition/class-diagram

The second part of this diagram is even simpler. As seen, from the employee controller the classes connect only within the specified use case. As such, the controller communicates with the database class (green), which communicates with the model class.

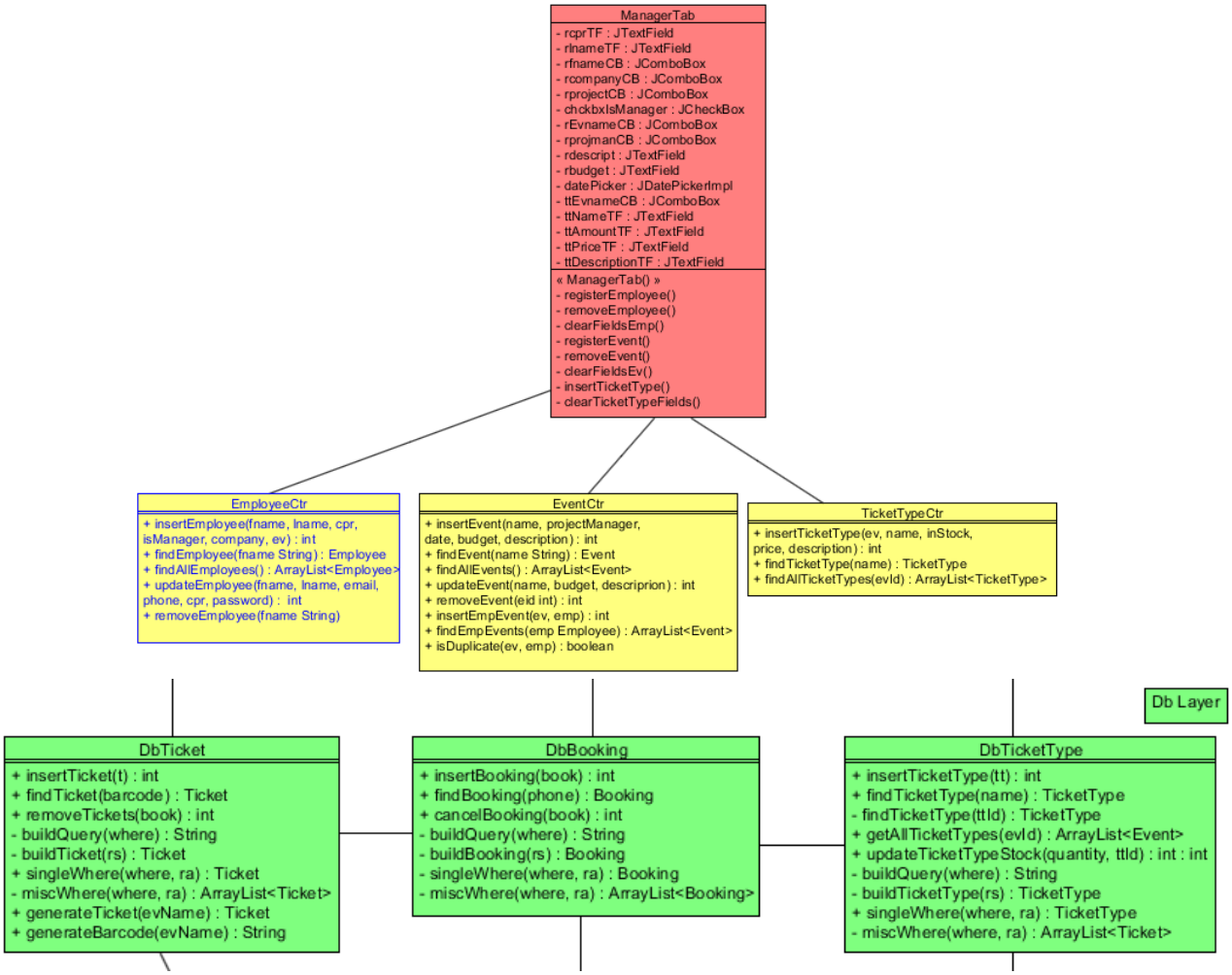


So far, even if the connection between the GUI and controller layer is not singular, the design is still implementing a low coupling as well as a high cohesion within the employee CRUD.

There are two particular cases in the design of the program that must be specified here. The first one is referring to the connections that exist between the manager tab class and the controller layer. In this particular case, the manager tab connects to three different classes from the controller layer (EmployeeCtr, EventCtr, TicketTypeCtr). Even though it raises the level of the coupling, these connections are required. The manager tab, which is only available to the use of logged in managers, is meant to give more functionality in one window. In this way, this tab is the place where most of the "create" and "delete" functionalities from CRUDs take place.

The second case, is taking place in the database package. Here the booking class is connecting to DbTicket and DbTicketType. It is the only place in this program where the database classes communicate within the package. This connection is justified by the fact that in the moment of creating a booking, the program is designed to update the number of tickets in stock, process that takes place in the DbTicketType. Also in the moment of creating a booking, new tickets with unique barcode are created in the DbTicket, and then assigned to the booking.

The graphical representation of the mentioned cases is included below.



Last thing to mention about the design of the program is the existing of two unconnected classes, existing in the GUI layer. These classes are a helper and a wrapper class.

The first class includes four methods that return array lists of wrappers, which are used when populating and retrieving objects from the combo boxes used across the program. Below is included one of the actual methods mentioned above.

```
public static ArrayList<Wrapper<Employee>> getEmployees() {
    EmployeeCtr empCtr = new EmployeeCtr();
    ArrayList<Employee> empList = empCtr.findAllEmployee();
    ArrayList<Wrapper<Employee>> wrapEmps = new ArrayList<>();
    for(Employee emp : empList) {
        wrapEmps.add(new Wrapper<Employee>(emp, () -> emp.getFname()));
    }
    return wrapEmps;
}
```

The need of a generic wrapper type can be simply explained. This class overrides the toString() method which is designed to return the name attribute of the object type. Beside this, the class includes another method which simply returns the object itself, which has been used for

creating the wrapper. As seen here, the constructor of the wrapper class takes as parameters, one object, and a lambda expression.

```
public Wrapper(T obj, Supplier<String> f) {
    this.obj = obj;
    this.f = f;
}
```

Another thing to mention from the implementation is the use of the "int res" in most of the classes. This attribute is always given a value of -1, and after executing an SQL statement in the database classes, the attribute is meant to change the value according to the number of rows that have been modified in the database following the execution of the statement. This happens in the methods where the following line is present: **"res = s.executeUpdate();"** . The value of res is then returned through layers, all the way up to the GUI, which displays a message to the user. If the value remains unchanged (-1), it means that an error occurred and the SQL statement has not been executed successfully (in which case such a message will be also displayed to the user).

An interesting functionality is implemented also in the DbTicket class. Beside the basic methods that can be found in all the other database classes within the program, this class includes two other methods. The first method called "**generateTicket(String evName)**" is calling the second method "**generateBarcode(String evName)**" which returns a string composed from the first two letters of the event name, combined with a random number between 1000 and 10000. Before returning the string, the method also checks if the generated barcode already exists in the database, in which case a new barcode will be generated. This process is repeated until a new barcode is generated, in order to avoid duplicate barcodes in the database. This barcode is returned to the first method, which assigns it to a newly created ticket, which will be returned further until it is added to the database.

In order to make it easier to follow the above explanations, below are added also the actual blocks of code that represent this explanation.

```
public Ticket generateTicket(String evName) {
    String barcode = generateBarcode(evName);
    //int res = -1;
    Ticket t = new Ticket();
    t.setBarcode(barcode);

    return t;
}

private String generateBarcode(String evName) {
    Random rand = new Random();
    int nr = rand.nextInt(10000)+1000;
    String barcode = evName.substring(0, 2) + nr;

    //TicketCtr tCtr = new TicketCtr();
    Ticket t = findTicket(barcode);
    if( t == null ) {
        System.out.println("if = true");
        return barcode;
    }
    else {
        System.out.println("if = false");
        generateBarcode(evName);
        return null;
    }
}
```

10. Conclusion

This project aimed to develop a computer program for Aalbornification, which could improve their daily activity in connection to bookings and other data. Even though the actual problem formulation is referring to designing a ticketing platform, this document included two different aspects/parts of the subject.

Beside the design and implementation, the business environment has been also analyzed in order to get more information on the importance of the project in terms of business. Following the business chapter, the new system proved to be a feasible investment and the project has proceeded with the actual design of the system.

The project has successfully addressed the problem of designing the system. All the use cases have been taken into consideration, in order to include all the required functionality that the company requires, in relation to registering bookings, and objects associated to bookings. However the implementation of the design has not been completed and further work is still required.

In conclusion, the program carries out successfully the basic functionality so far, as well as the most important use case (registering a booking), which proves that the formulated problem statement has been answered and the project met its goals and requirements.

11. Working process

The development of this project in a team of four has been quite challenging at times. This happened due to the difference in the level of involvement which the team members showed during the semester.

However improvements have been made from the last performance in the same group, consisting in a more organized working environment, better performance in clearly establishing and assigning tasks to team members, and a better compliance with the deadlines that have been set.

As an example, during this project the earlier problems represented by conflicts on the svn folder has been almost reduced to zero. The team members worked more organized in contributing to the svn folder, while in previous projects this has represented a big problem.

Another improvement in the team work is represented by the way the tasks have been assigned to the team members. This has been a weakness in the past, as the tasks were assigned according to ones strengths. This resulted in the same member dealing with most of the code, another member dealing with the design of diagrams, and others dealing with the written report.

However, in this project tasks have been assigned in a mixed manner, in such a way that each team member gets to work within all the parts of the project.

The overall working process has improved, resulting in better team work and better results.

Revision Number: 42

Repository:

https://github.com/IliyanStoev/dmai0914_2sem_2FinalProject/

12. Bibliography

- What is software? Definition and meaning. Retrieved from http://www.businessdictionary.com/definition/software.html#ixzz3Z5DzcrAB
- Strengthening your organization. A series of modules and reference materials for NGO and CBO Managers and Policy Makers – Organizational Structure. Retrived from http://www.pathfinder.org/publications-tools/pdfs/Strengthening-You-Organization-A-Series-of-Modules-and-Reference-Materials-for-NGO-and-CBO-Managers-and-Policy-Makers-Organizational-Structure.pdf
- Divisionalised oranisational structure. Retrieved from http://yourbusiness.azcentral.com/divisionalized-organizational-structure-4924.html
- About us. Retreived from http://aalbornification.dk/about-us/
- Use case diagram. Retreived from *http://whatis.techtarget.com/definition/use-case-diagram*
- Quality Criteria and Review. Retrieved from ecampus
- Applying UML and Patterns by Craig Larman
- Business Modeling with UML by Penker
- Transformation to RDB, Session 5. Retrieved from ecampus
- Class diagram. Retrieved from http://searchsoa.techtarget.com/definition/class-diagram

13. Appendix

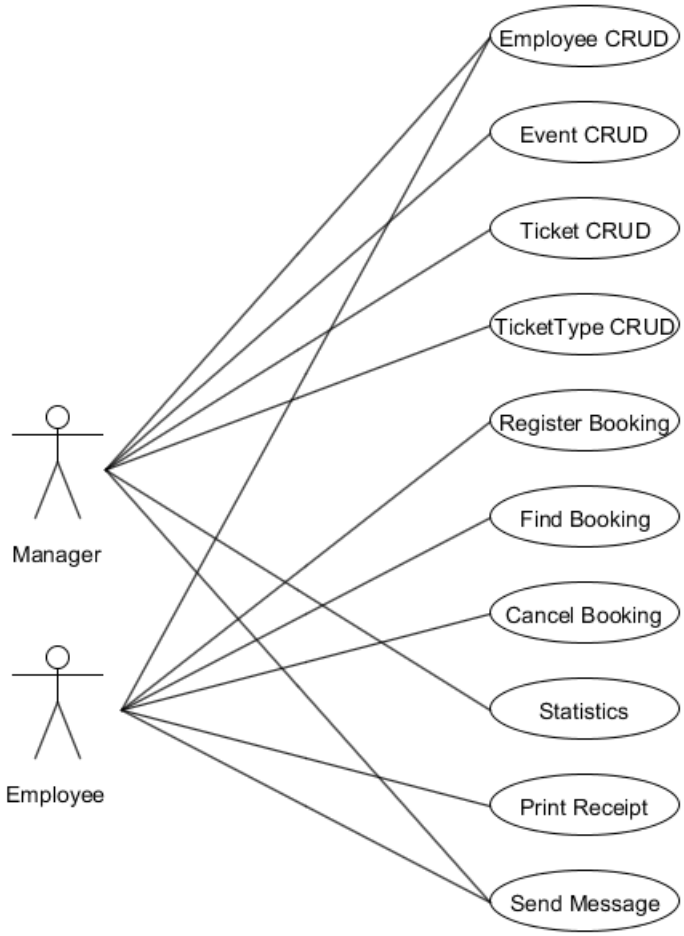


Fig.1 Use-Case Diagram

Fig.2 Employee CRUD Brief Description

Employee CRUD use case
The Employee CRUD functionality is meant to give the user the possibility to create new users, read and update the information, as well as removing the user completely from the database.
The functionality will however be divided. In this way, only a manager will be able to create, read and remove a user, while only the user itself will be able to update his/her employee information.
A manager can create a user by filling in the first name, last name, the phone and the cpr numbers. The newly created user will then be able to log in using his Cpr. no with the default password. Only after the login will he/she be able to fill in the remaining information in the employee information section. Also the password change process can take place after the first login.

Fig.2 Employee CRUD Brief Description

Event CRUD use case
The Event CRUD functionality is meant to give the user the possibility to create new Events, read and update the information, as well as removing them completely from the database.
The functionality will however be divided. In this way, only a manager will be able to create, read and remove an event, while the employee itself will only be able to read and update information about it.
A manager can create an event by filling in the name, and add some description. The employees will then be able to see (read) information about the event and add information if it is necessary.

Fig.3 Event CRUD Brief Description

Register/Find/Cancel Booking use cases
The Register Booking functionality is meant to give the user the possibility to create new Bookings. The function will be available only for the employees. Booking can be created by filling in the date, status, total, guest name, guest phone, guest email, quantity, ticket type id, cpr number and receipt number.
The Find Booking functionality is meant to show the user information about a previously created booking. Booking can be found by providing the guest phone
The Cancel Booking functionality is meant to allow the user to cancel already created booking.
Booking can be canceled by clicking the Cancel Button.

Fig.4 Register/Find/Cancel Booking Brief Descriptions

Fig.5 Fully Dressed Use Case Description Register Booking

Use case name	Register Booking	
Actor(s)	Employee	
Preconditions	Employee, ticket type and event already exist.	
Post conditions	The booking has been made.	
Frequency	Many times per day	
Main success scenario / Flow of events	**Actor**	**System**
	1. A customer calls to book tickets.	
	2. The ticket type, the amount and the event get selected.	3. The system calculates the total price.
	4. The employee adds the customer's information.	
	5. The payment method gets selected.	6. The booking is created and the user gets notified.
Alternative flow	3a. There may not be enough tickets – the user will be notified	

Candidate for classes	Evaluation	In/Out
Customer	Not needed to create a booking	Out
Employee	Needed to create a booking	In
Booking	Needed to book tickets	In
Ticket	Needed to create a booking	In
Ticket Type	Needed to create a booking	In
Event	Needed to create a booking	In
Receipt	Needed to print details about a booking	In
Statistics	Needed in order to make statistics over an employee	In

Fig.6 Candidates For Classes Table

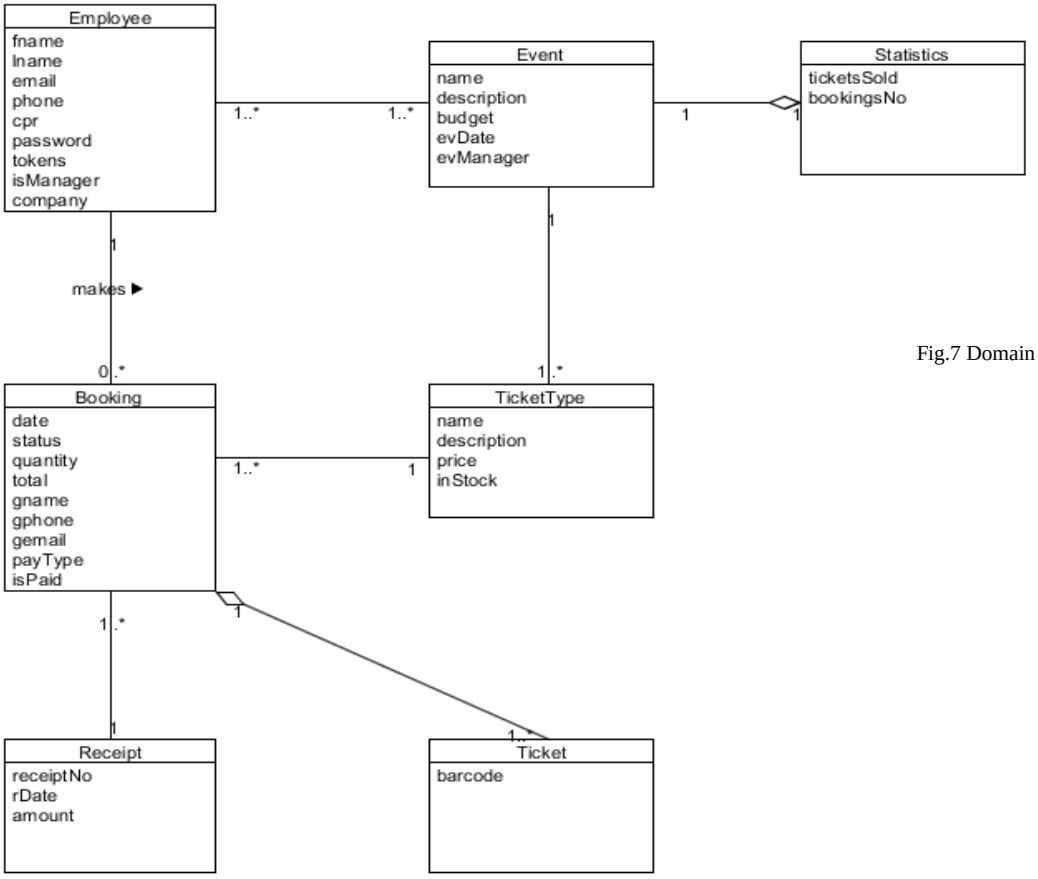


Fig.7 Domain Model

Fig.8 Relational Model

Use case	Priority	No.
Employee CRUD	Medium	3
Event CRUD	Medium	4
Register Booking	High	1
Find Booking	High	2
Cancel Booking	Medium	5
Send message	Low	8
Print receipt	Low	6
Statistics	Low	7

Fig.9 Prioritized List

Operation : insertEmployee(fname, lname, phone, cpr)

Use Case: Employee CRUD

Pre-condition: database connection is setup and employee table exists

Post-condition:
- An employee object was created
- The object information has been added to the employee database
- emp.fname became fname
- emp.lname became lname
- emp.phone became phone
- emp.cpr became cpr
-

Figure.10 Operation Contracts Employee CRUD

Operation: findEmployee(fname)

Use Case: Employee CRUD

Pre-condition: the desired employee has been already been inserted into the employee database

Post-condition:
- the fname is found in the database
- the employee object is built using the values from the database
- the employee object is retrieved to the gui
- the employee information is displayed in the window

Operation: updateEmployee(fname, lname, email, phone, cpr, password)

Use Case: Employee CRUD

Pre-condition: the employee with the specified CPR, has been already inserted into employee database

Post-condition: - the values in the database are replaced with the newly inserted values, where the CPR is the same as the specified CPR

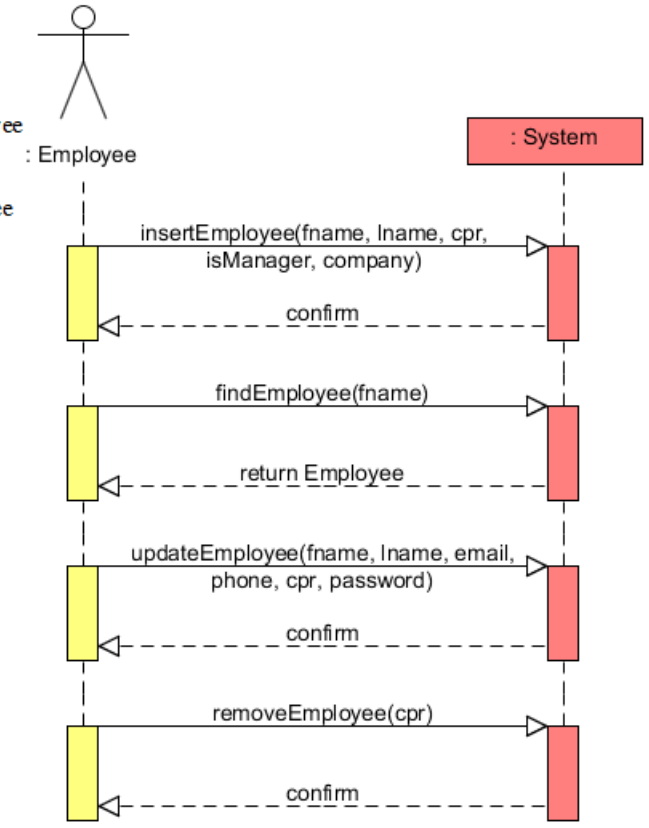
Fig.11 System Sequence Diagram Employee CRUD

Operation: removeEmployee(cpr)

Use Case: Employee CRUD

Pre-condition: the employee with the specified CPR, has been already inserted into employee database

Post-condition: - the employee with the specified CPR is removed from the employee database



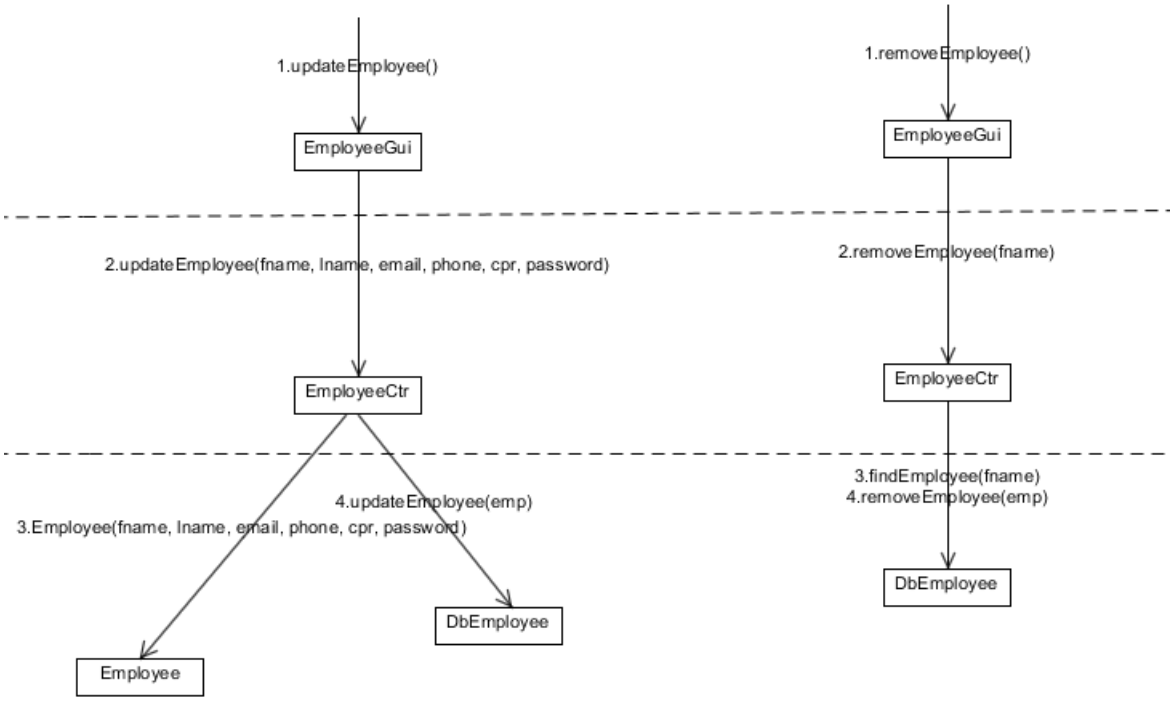


Fig.12 Interaction Diagram Employee CRUD (Update and Delete Employee)

Fig.13 Interaction Diagram Employee CRUD (Create and Read Employee)

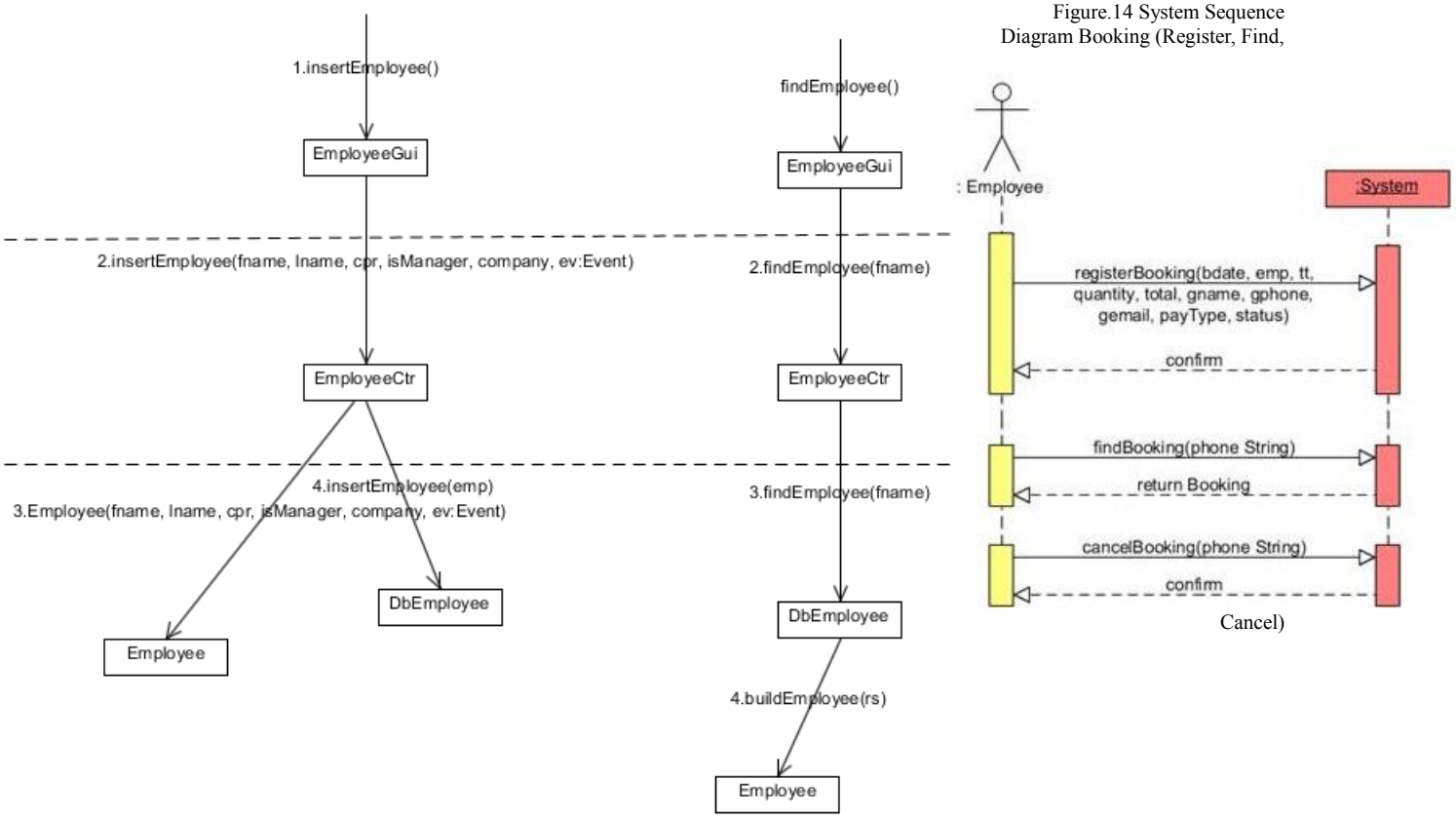


Figure.14 System Sequence Diagram Booking (Register, Find, Cancel)

Figure.15 Interaction Diagram Register Booking

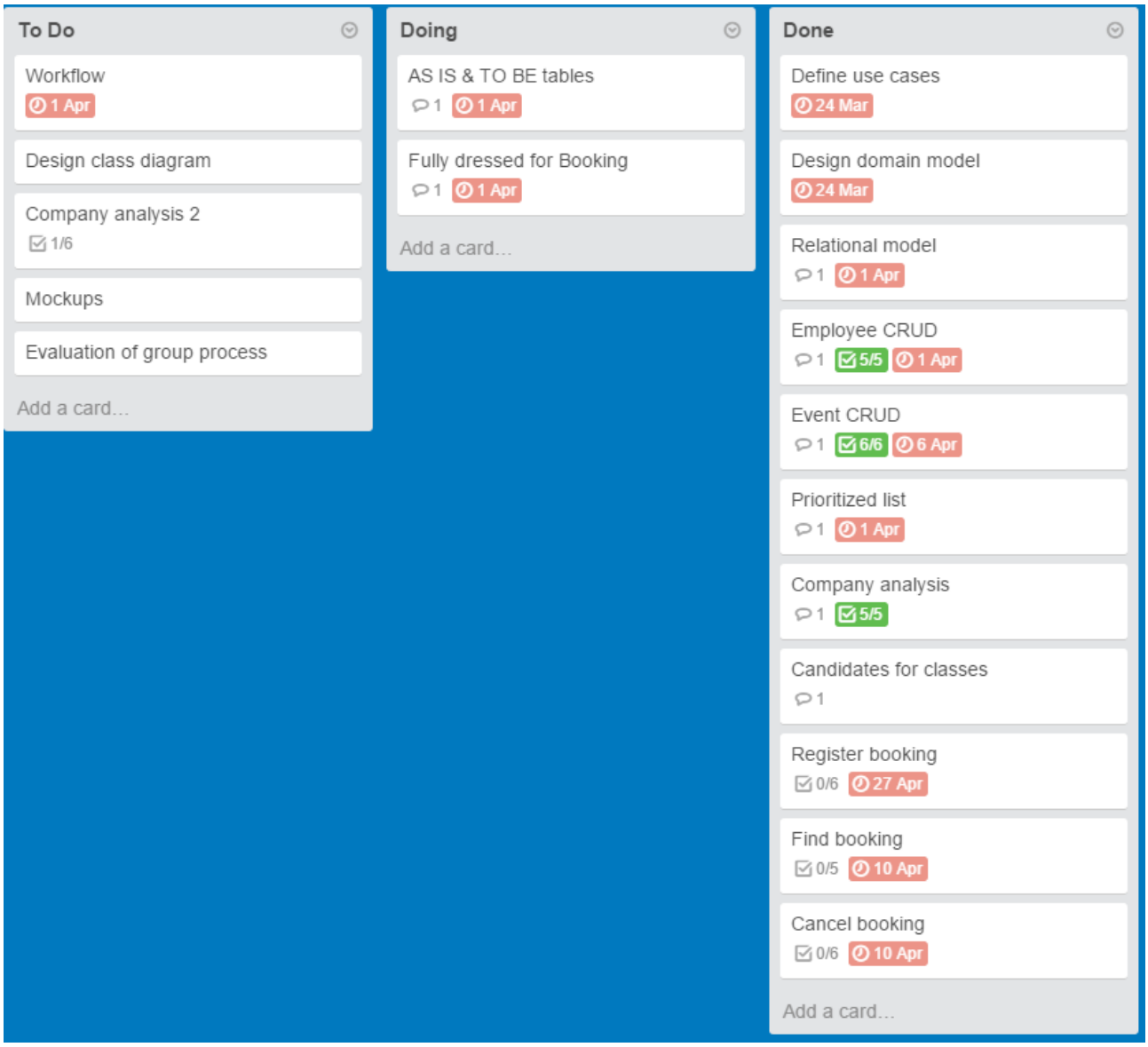
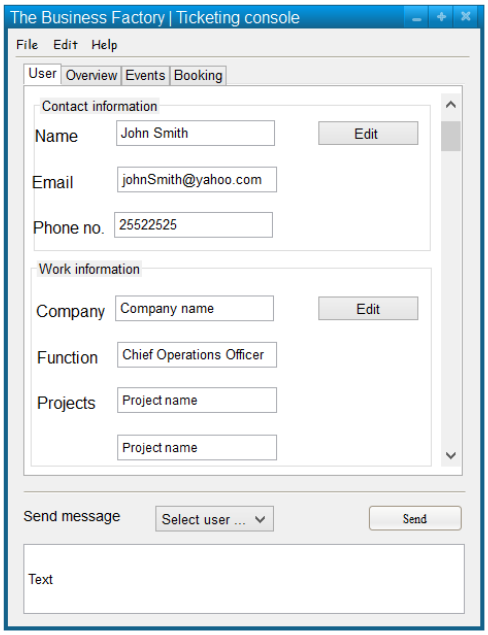
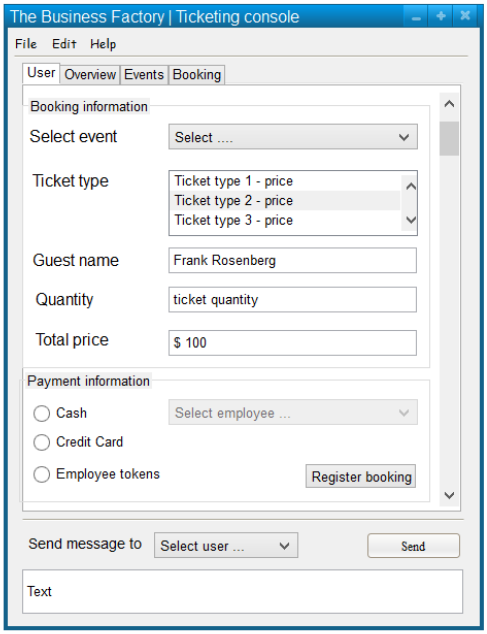
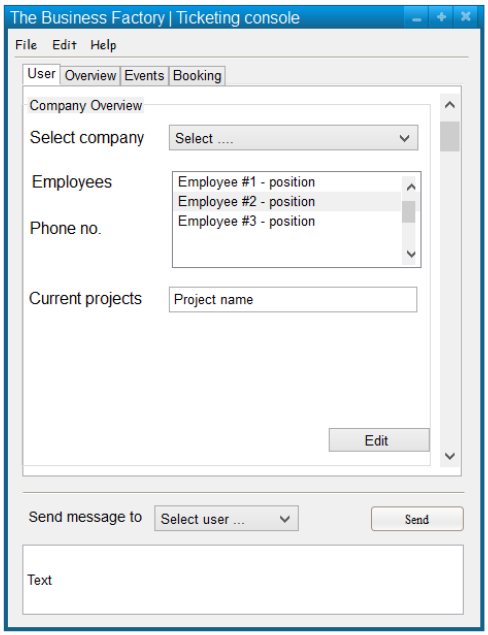


Figure.16 SCRUM in Action







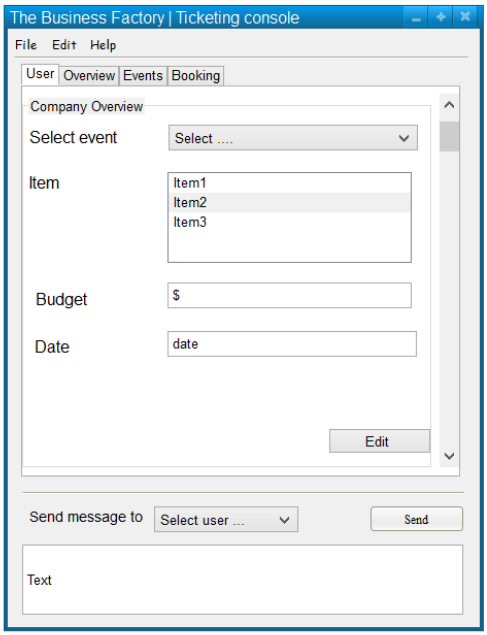


Figure 17 Mock-up

	START		Mon 2.3.15	
	Phase 1# Collect And Analyse Data	**30 days**	**Mon 2.3.15**	**Fri 10.4.15**
	Description of the organisational structure	1 day	Mon 2.3.15	Mon 2.3.15
	Evaluation of organisational structure and problems	1 day	Mon 2.3.15	Mon 2.3.15
	Organisational culture and description of the managers behaviour (style of leadership)	1 day	Mon 23.3.15	Mon 23.3.15
	Make an analysis of the project stakeholders and their expectations to a new system	1 day	Mon 23.3.15	Mon 23.3.15
	Make a SWOT analysis	1 day	Wed 1.4.15	Wed 1.4.15
	Business Case	1 day	Fri 10.4.15	Fri 10.4.15
	Phase 2# System Development	**24 days**	**Mon 27.4.15**	**Thu 28.5.15**
	Feasibility Study and Requirements	1 day	Mon 27.4.15	Mon 27.4.15
	Analysis and Design	3 days	Mon 11.5.15	Wed 13.5.15
	Phase 3# Implementation and Testing	**11 days**	**Mon 18.5.15**	**Mon 1.6.15**
	Implementation and Testing	11 days	Mon 18.5.15	Mon 1.6.15
	FINISH		Tue 2.6.15	

Fig. 18 Project Plan in MS Project