**PROJECT REPORT**

Zinema – Management Software

Distributed System

**Students**

Andrei Cioanca – 266105

Claudiu Rediu – 266129

Dominika Kubicz – 266148

Nikita Roskovs – 266900

Stefan Harabagiu – 266116

**Supervisors**

Jakob Knop Rasmussen

Jan Otto Munch Pedersen

**Abstract**

The following report will present the process of introducing the Zinema – Cinema Management Distributed Software System to the market and the conclusions that have been drawn in the end.

The project followed the Unified Software Development Process and SCRUM. At the start, the focus was on setting up why the system should be created. As it advanced, the functional and non-functional requirements were defined and use cases were created accordingly.

A Domain Model was used to analyse the problem and understand how different objects would interact with each other. Next, the design of the system was defined using Design Class diagrams based on the existing Domain model.

The Design Class diagrams acted as blueprints for the implementation of a 3-tier architecture system. In the last phases of the project, the application was tested based on the Use Cases.

Once finalized, the results were subject to discussion and analysis to bring further improvements in the future.

**Table of contents**

[1. Introduction 1](#_Toc526199517)

[2. Requirements 2](#_Toc526199518)

[3. System Analysis 2](#_Toc526199519)

[4. System Design 3](#_Toc526199520)

[5. System Implementation 3](#_Toc526199521)

[6. Testing 4](#_Toc526199522)

[7. Results and Discussion 5](#_Toc526199523)

[8. Conclusion 6](#_Toc526199524)

9. Project Future

10. List of Appendices

11. Sources of Information

# Introduction

The film industry has experienced a steady growth in recent years, as reported by the Motion Picture Association of America (Motion Picture Association of America, 2017), with the global box office in 2016 for all films increasing by one percent from 2015. In 2016 the number of cinema screens has also increased by eight percent showing an upward trend in what concerns the cinema business. In Denmark there have been reported 11.9 million admissions to movies in 2017 alone (Danish Film Institute, 2017). As the average ticket price for a movie can be estimated at about 9.6 euros the Danish box office revenue for 2017 was approximately 114 million euros. The annual budget of the Danish Film Institute (DFI), for that same year, was 66.9 million euros, the annual accounts reaching a total of 7.4 million euros. The operating costs for the DFI was roughly 1.6 million euros, 26 percent of which was attributed to properties, IT systems, etc.

Over the years, Danish cinemas have experienced a steady decrease in activity and popularity. While the total number of cinemas and digital screens within them has increased since 2011, overall, there has been a decline in the total number of tickets sold, despite the constant sprouting of new cinemas all over the country.

In 2011 there were 12.4m tickets sold in Danish cinemas (155 cinemas) (Danish Film Institute, 2012), in 2016 (Danish Film Institute, 2017) there were 13m tickets sold (163 cinemas) while in 2017, 11.9M (Danish Film Institute, 2018) (166 cinemas). The number of tickets sold per capita fluctuates between 2.1 and 2.4 with 2017 coming in at 2.1.

This overall decline in recent years (Nielsen, 2018) can be attributed to many reasons but the rise in popularity of streaming platforms such as Netflix and HBOGo that offer, subscribed users, movies and tv shows on demand has been one of the main reasons why Danes no longer go to the cinema as often as they did. Netflix launched in 2012 (media.netflix.com, 2012) in Denmark and since then it quickly became the most popular entertainment streaming platform in the country.

There is a new cinema company on the market. Zinema is an up and coming business in need of help. It wants to have alleviated some of the pressure brought by starting in a declining market in Denmark. Its main purpose is to revive the Danish films in a new modern way. Some of the biggest maintenance costs go into the IT systems. Their software has an old interface that is costly to maintain and built in a way that no further features can be added to it. The customers and administrators have reported that it’s hard to handle and crashes often, without having saved what the user has done. This has resulted in a lot of money lost for the starting company.

POStive Cinema, a company that offers management solutions to cinemas, states that their product helps their clients “achieve better results and improve operational efficiency” (POSitive Cinema, 2018). Knowing that software has a big influence on the future success, Zinema wants to provide their clients with the best possible software that will not only “enhance the cinema experience” (Vista Cinema, 2018), but also emphasize Zinema’s unique features.

The cinema business is in search of a way to improve the user experience and the functionality while also lowering the costs of maintenance.

# Requirements

1. **Functional Requirements**

**Movie Creator**

1. A user should be able to create movies and store them in the database.
2. A movie consists of a title, year of creation, release date, price for renting, name of the studio, director name, description, main actor name, rented status.

**Movie Manager**

1. A user should be able to retrieve a list of movies that are not rented from the database.
2. A user should be able to rent a movie from the acquired list.

**Movie Scheduler**

1. A user should be able to make a schedule and store it in the database.
2. A schedule consists of a list of scheduled movies.
3. A schedule movie consists of a movie, a room, a time and day at which it’s booked and a list of seats.
4. A user should be able to get a list of rooms.
5. A user should be able to get a list of rented movies.
6. A room consists of number of seats and a description.
7. A user should be able to add and remove rooms from the database.

**Ticket Booker**

1. A user should be able to book tickets for the scheduled movies.

**Advertiser**

1. A user should be able to create an advertisement and store it in the database.
2. An advertisement consists of advertiser name, scheduled movie title and duration.
3. A user should be able to register.
4. A user should be able to login with email and password.

\*Rent = change the rented status to true

1. **Non-Functional Requirements**
2. The system will have a 3-tier architecture
3. The system will be written in C# and Java
4. The system will have a GUI for each client
5. The system will use a protocol for sockets

# System Analysis

The system makes use of several different components communicating with each other to achieve some sort of ecosystem. These components are being controlled by the employees of Zinema and their customers are able to interact with the system through their official website.

Starting from the bottom, the Movie Creator creates the movies and stores them in the database.

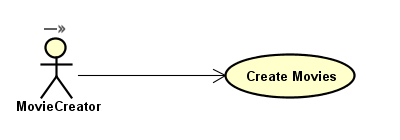


Figure - Movie Creator Use Case

[Activity Diagram For Creating Movie]

The employee that operates the application can also list the current movies from the database in order to ensure that his actions were successful without requiring relying on a different component.

The Movie Manager is the next component in the chain. This one can rent movies that have been created by the previous link, as long as they have not already been rented. In broad terms, it functions quite similarly to the Movie Creator but has a more restricted access to the data it works with.

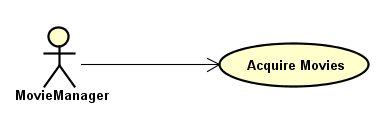


Figure - Movie Manager Use Case

An employee is able to retrieve a list of the currently unrented movies from the database and then simply insert the id of the movie he wants to rent in the application and the system will store it as rented. Once this is complete, the list will be automatically updated and the rented movie removed from the list.

[Activity Diagram for Movie Manager]

The third component is the Movie Scheduler. It is definitely the most complex one since it can do a lot more than the others combined. The scheduler can create and manage rooms for the cinema as well as take the movies that the previous component has rented and schedule them for screening using the available rooms.

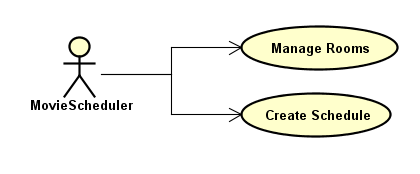


Figure - Movie Scheduler Use Case

The employee working as the scheduler can also send the scheduled movies available for the week to the next and last component of the system, the one that handles customer interaction. This part of the system can also retrieve and manipulate information from the database such as getting the list of available movies to be scheduled or even deleting rooms from the system.

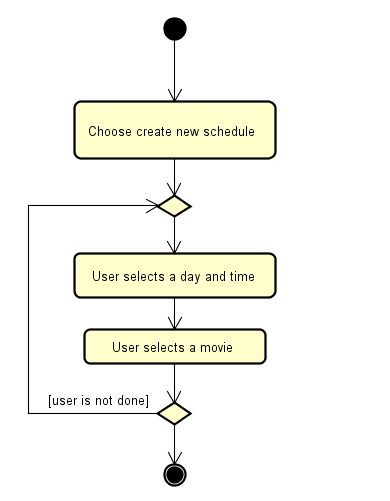


Figure - Create Schedule Diagram

The last component is called the Ticket Booker and is the only component in the system that handles and allows for customer interaction. Using the information from the previous component, the component allows the customer to input data through the cinema’s official website.

The customer can see this week’s scheduled movies and can select one from the list. Once selected he will be able to see all available seats in the room as well as the ones that have been taken. He can then pick the number of a seat and with a single click of a button the system will register his booking for that seat.

All that is left to do for the customer is to give his personal e-mail and telephone number so that the information can be sent to him and then he needs to show up thirty minutes before the screening to receive his ticket.

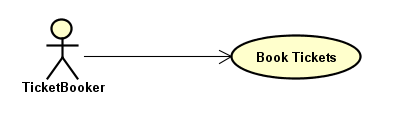


Figure - Ticket Booker Use Case

[Activity diagram for booking ticket]

As seen from these figures, the system is easy to understand and more importantly, easy to use for both customers and employees of the cinema, a trait highly appreciated by both consumers and stakeholders.

# System Design

The first step in designing the system was making an overall view of the system and its components and then expanding on that with details by using UML diagrams.

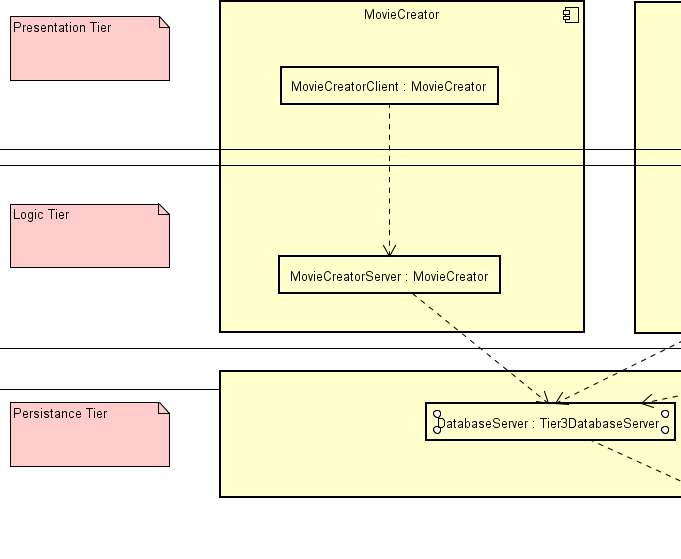


Figure - The three tiers

The design of the system follows a 3-tier architecture that helps separate the user interface, business logic and data storage layers.

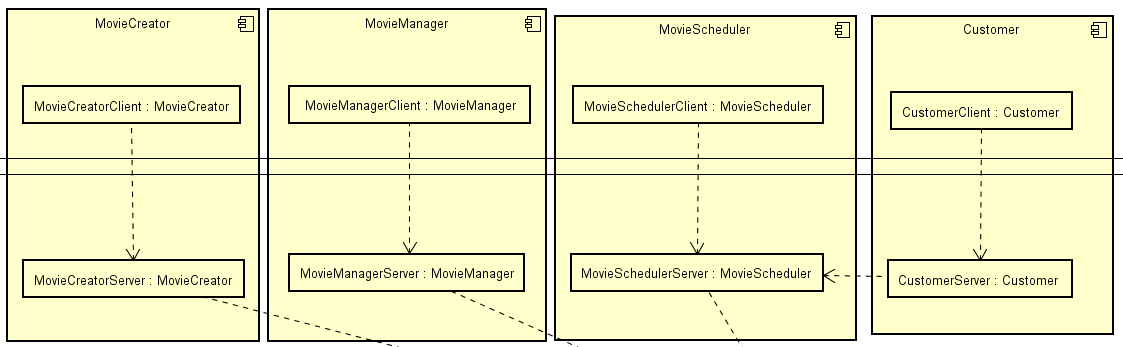


Figure – Overview of all four components

The system is structured into four different components that communicate with each other; every component having specific responsibilities. For the first three components the first and second tiers are implemented in Java and for the last component tier 1 and 2 in C#.

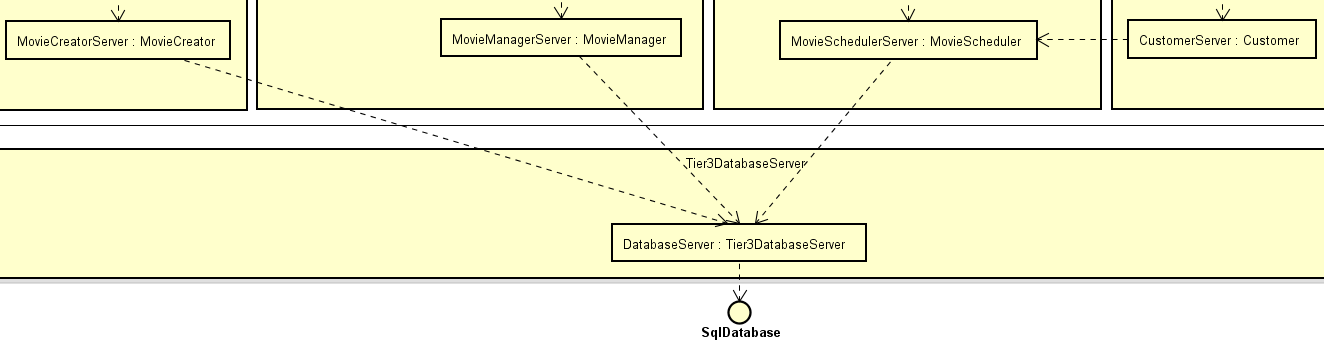


Figure - Tier3 overview

The chosen method of data storage is a database, therefore, the third tier was implemented in C# as well, in order to benefit from the Entity Framework Core data access technology.

The communication between tiers is made through sockets using the TCP protocol. In both Figure 7 and Figure 8 communication implemented by the team through sockets is represented by the dotted arrows. In order to easily send data that can be interpreted by programs written in both Java and C#, the data sent between tiers is serialized with Json.

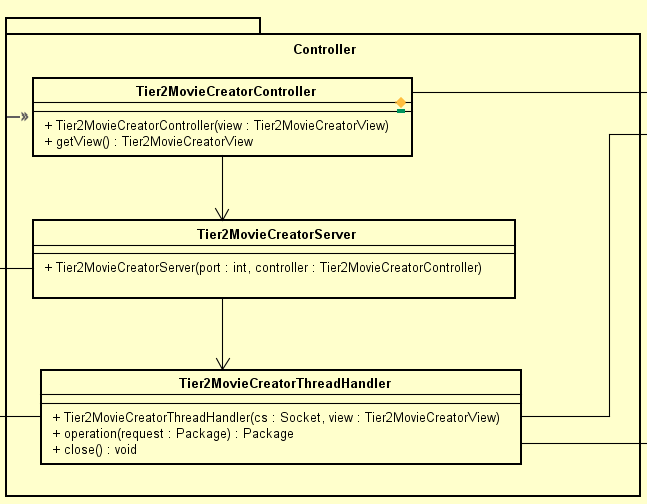


Figure - Server Diagram

All servers in the system follow the structure shown in Figure 9. When the controller is instantiated through its constructor it will instantiate a Tier2MovieCreatorServer by giving it a port and injecting the controller. The server class will then open a socket on the given port and listen for connections. When a connection is made the client socket will be injected in the ThreadHandler class, which runs on a thread, that handles the communication between the client and the server.

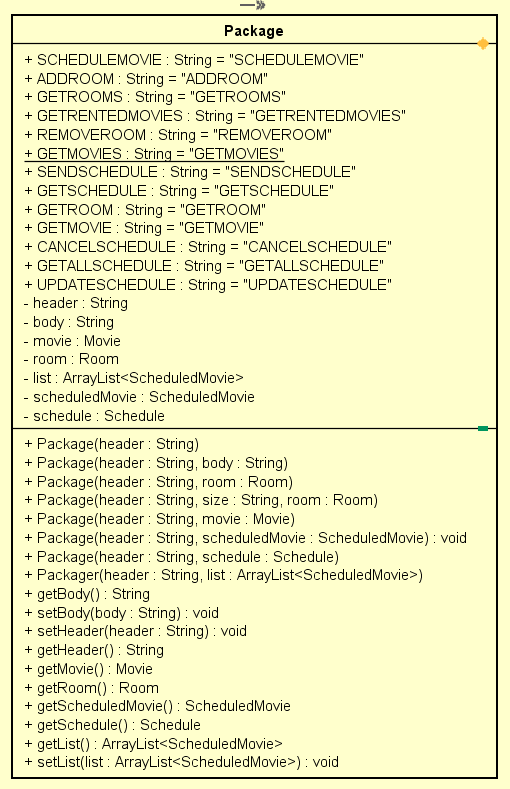


Figure - Package class for sending data

The data sent between tiers is formatted by using a custom made package. The Package class contains a header field so that the client or server receiving can identify what it needs to do with the data inside the package. This way, all objects that need to be sent through a TCP connection will be encapsulated in a package and then serialized into Json.

The third tier is common for all for components and it acts as a server that receives requests. According to those requests it runs queries on the database to either retrieve or save data. This part of the system uses EF Core to easily store objects into the database.

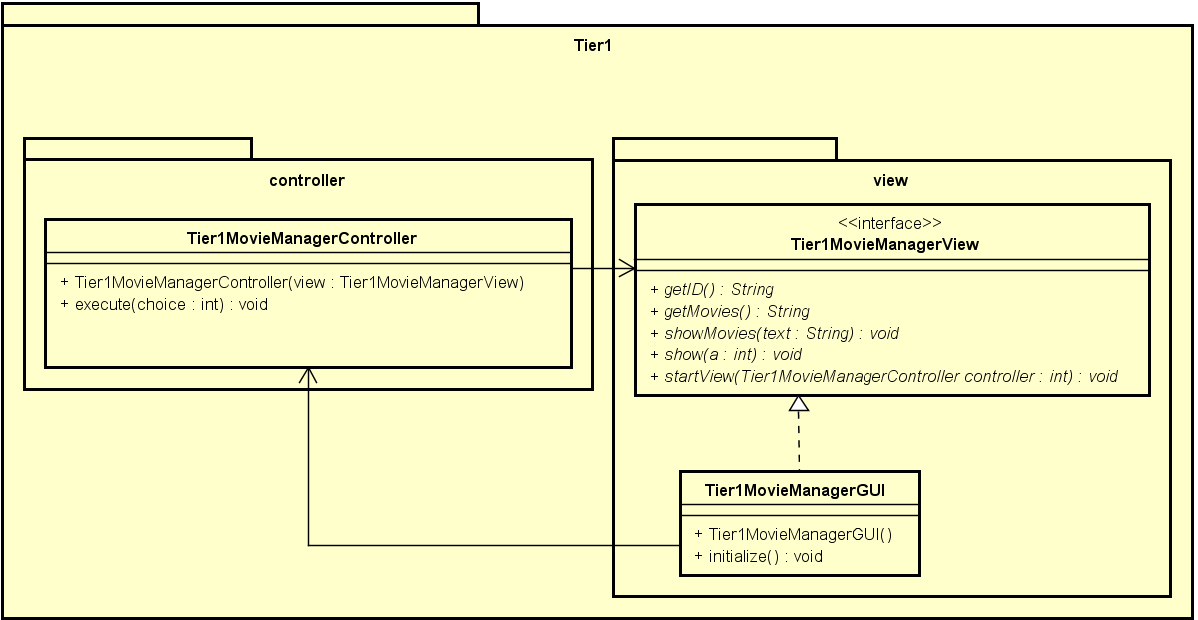


Figure - Tier1 First Component

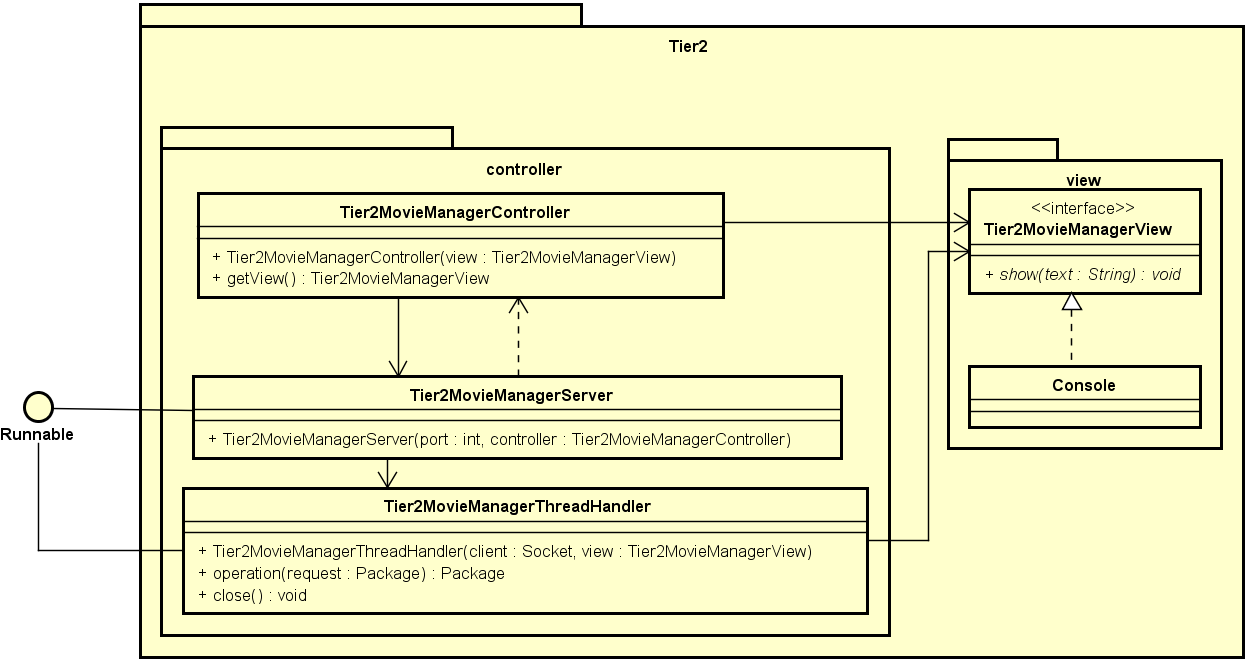


Figure - Tier2 First Component

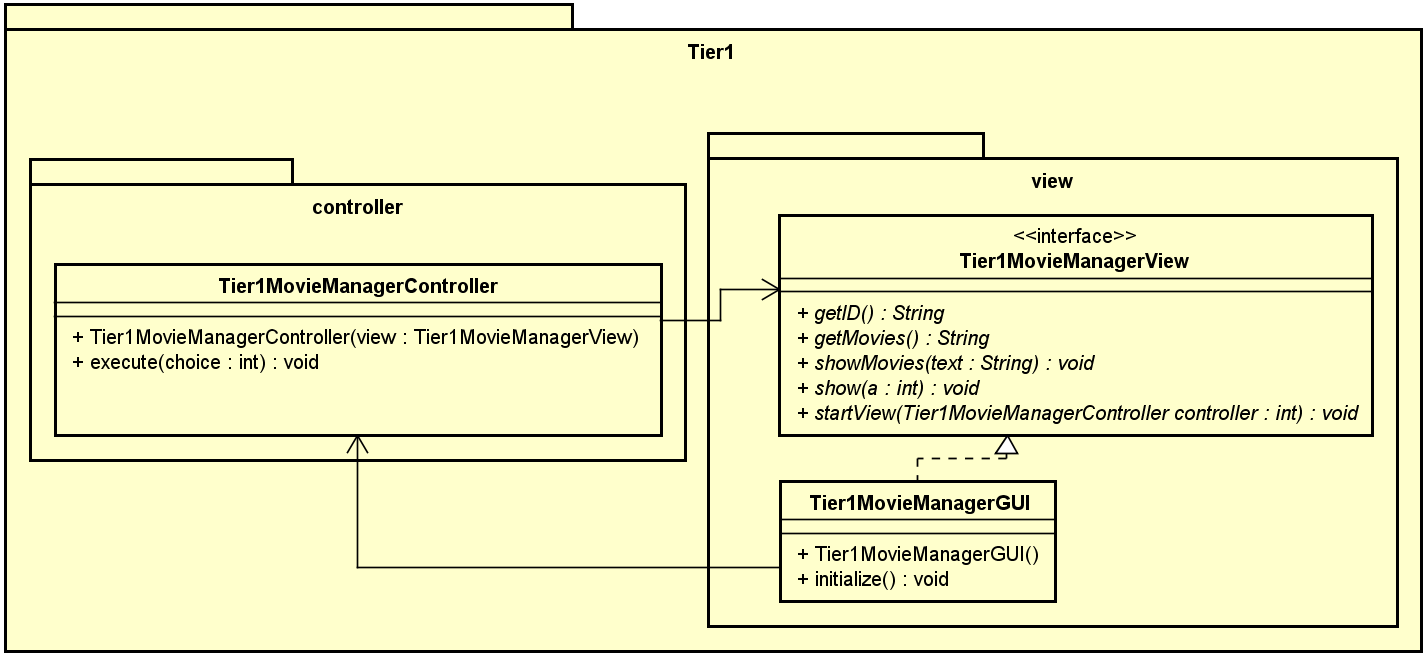


Figure - Tier1 Second Component

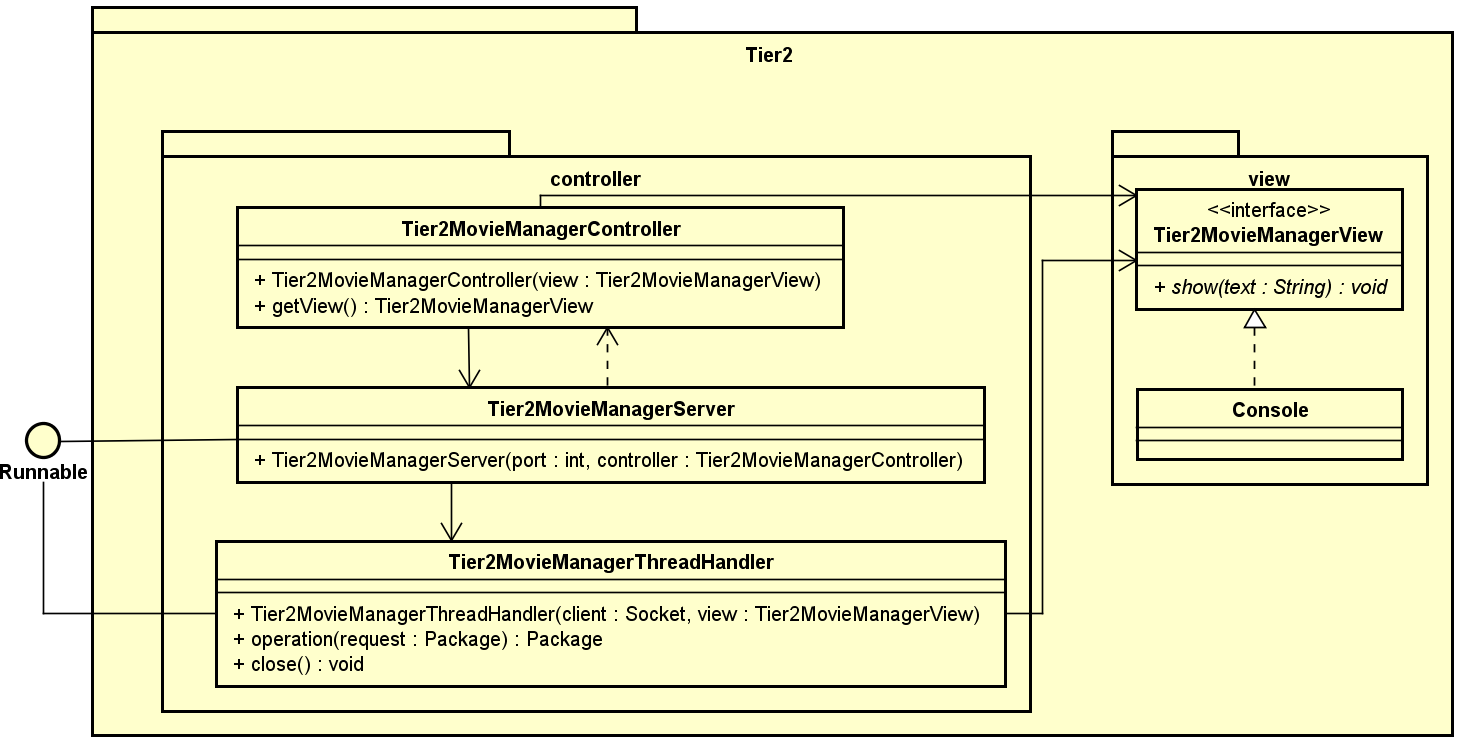


Figure - Tier2 Second Component

The first two components are quite similar to each other in terms of functionality so they are presented together. The first tier is responsible for displaying a GUI to the user and taking input from them.

The second tier acts as a server that receives requests from the first tier, interprets them and then it replies. The second tier also acts as a client for the third tier, as it sends data to tier 3 so it can save it into the database. Furthermore, on this level the input from the user is validated before sending it to the third tier.

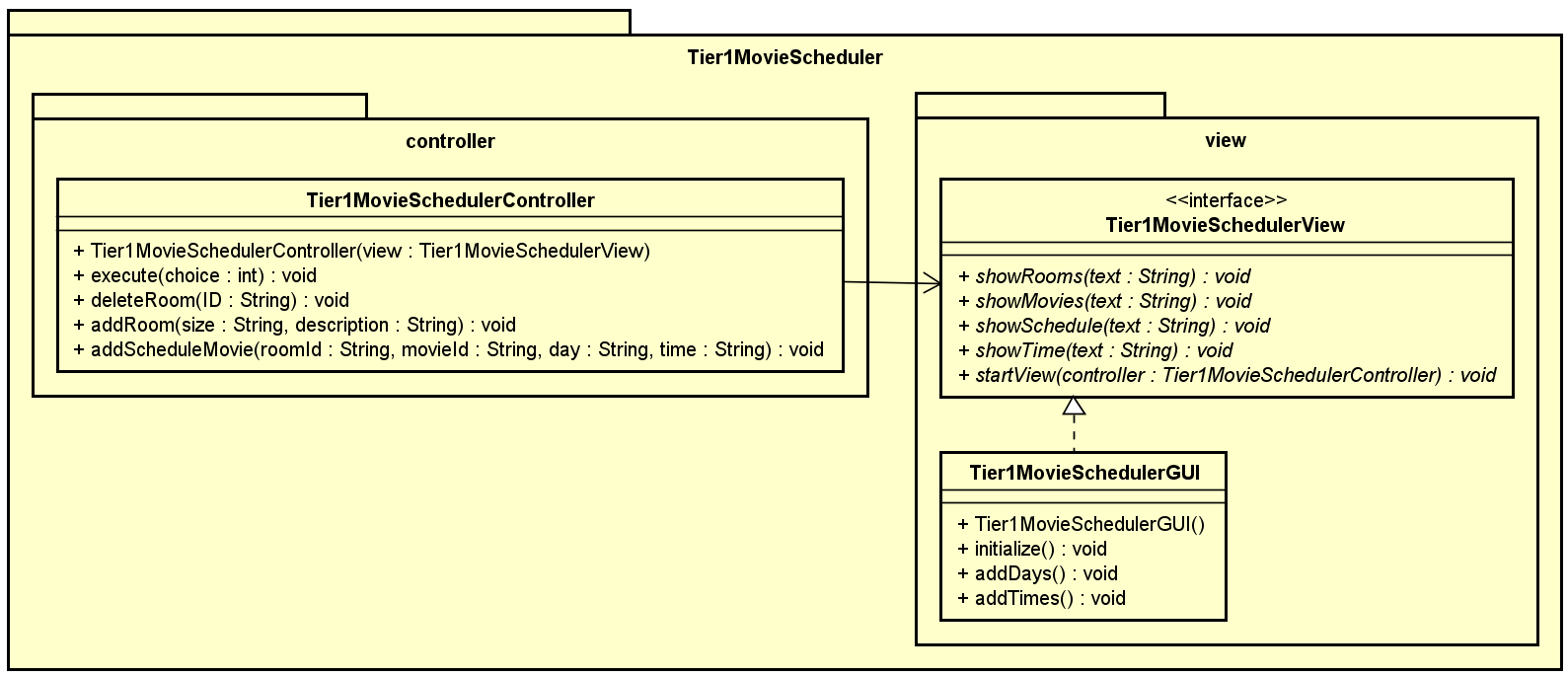


Figure - Tier1 third component

The third component is responsible for creating rooms in which movies can be displayed and for creating the schedule of the cinema.

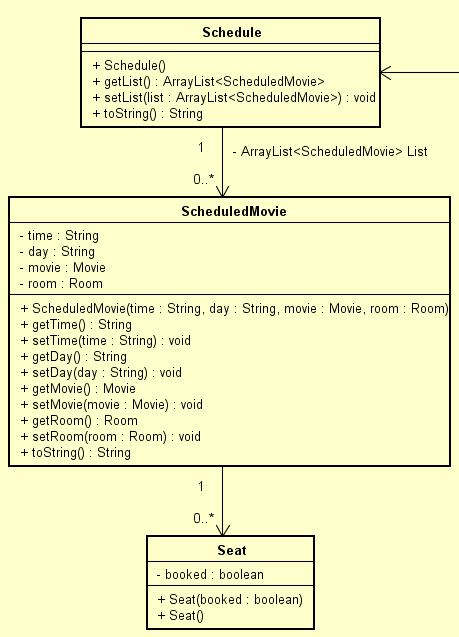


Figure - MovieScheduler Schedule Model

A schedule consists of multiple Movie – Time – Day – Room – Seats pairings. The ScheduledMovie class is used to represent one pairing and the Schedule class contains a list of ScheduledMovie objects. The size of the Seats array is dependent on the size attribute of the Room the movie is displayed in.

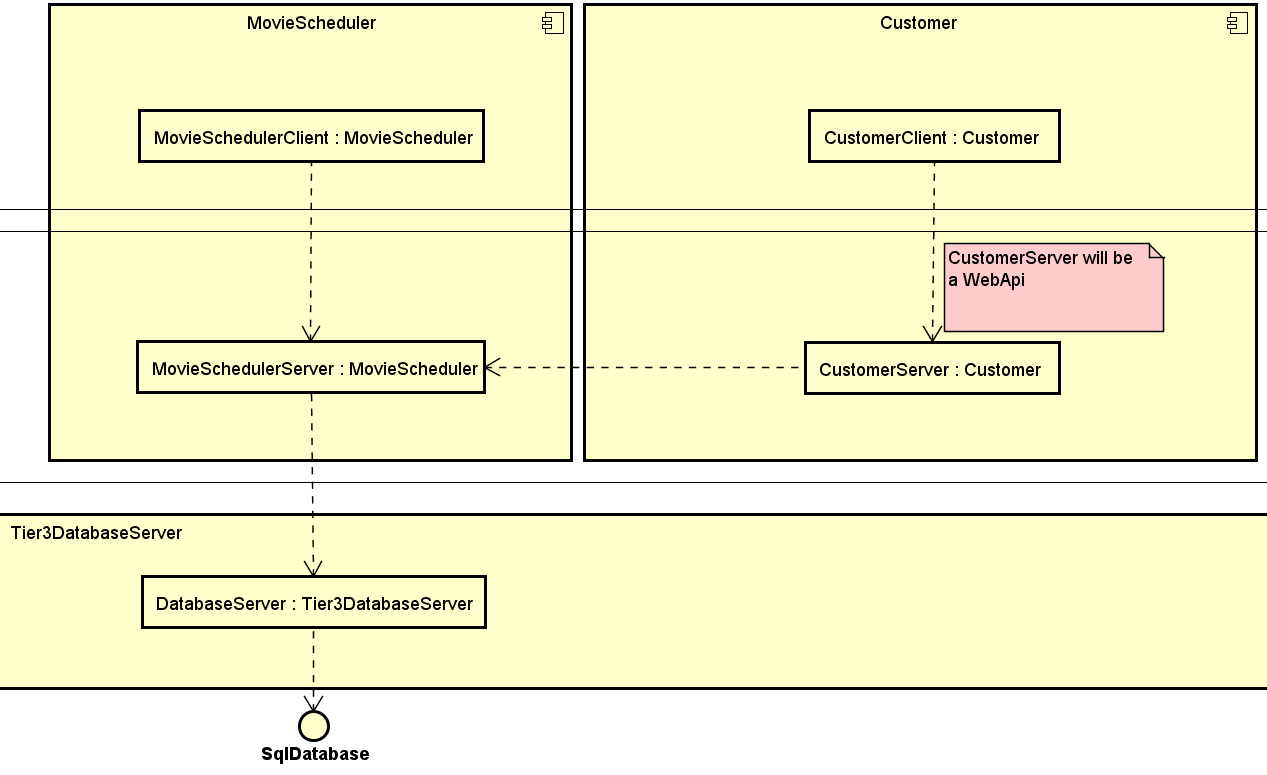


Figure - Communication between last two components

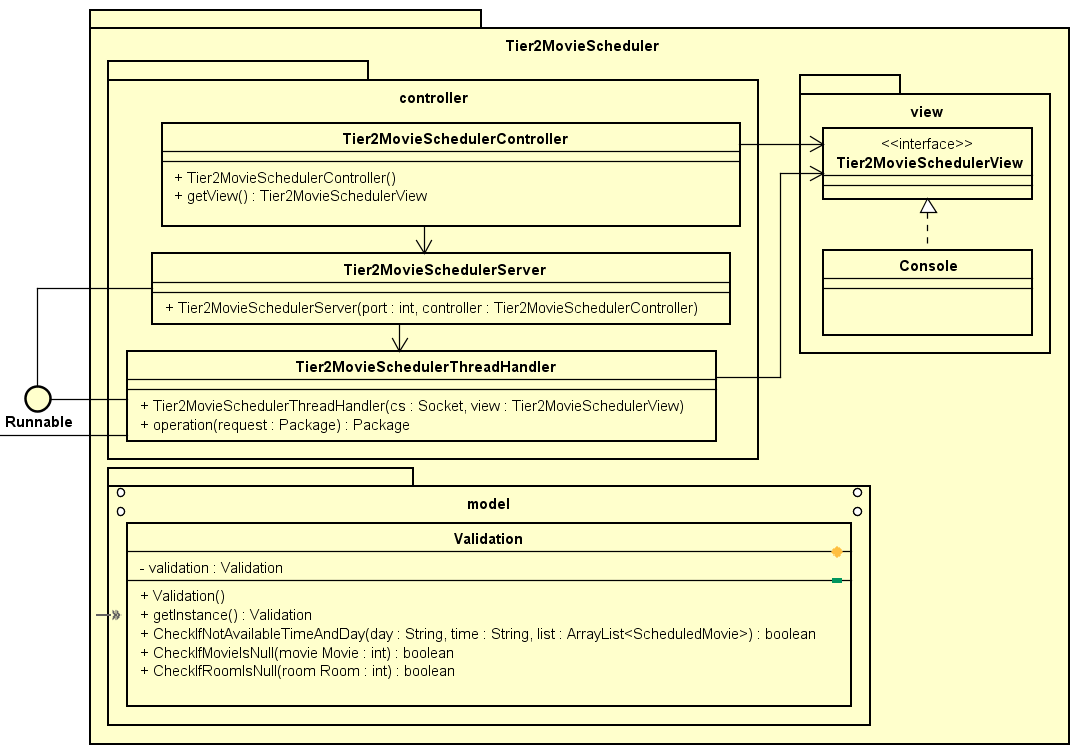


Figure - Tier2 third component

First of all, the tier 2 in this component acts as a client to the third tier just like the other components, however, the server on tier 2 can receive requests from the tier 1 in the MovieScheduler component, which is implemented in Java, and from the WebApi from the fourth component, implemented in C#. This design makes it so that the WebApi does not have direct access to the data storage layer. Secondly, the input from the user on tier 1 is validated on this level through the use of the Validation class. The methods in this class are used to check for mistakes the user could have made when writing.

The last component was designed to be used by the clients of Zinema, therefore, it uses a web application to grant ease of access to the users. The WebApi once it starts will send a request to the server on the second tier of the previous component, in order to get the movie schedule for the cinema. The user will interact with this component through a web page designed with Razor Pages, that depending on the user’s input will send HTTP requests to the WebApi. The WebApi can return the schedule to the user on a GET request and can book a seat for a specific screening on a PUT request.

**Security**

Taking into consideration the design of the system, not too many security mechanisms are required to be implemented as potential losses are not particularly high. The last component uses a website and Webapi and this does expose the system to certain risks. After the client chooses what seat to book they will be asked for the email address and phone number in order to be able to receive the ticket. Even though that data is not stored by our system, the message sent by the client to the Webapi could be intercepted and thus, information disclosure would be achieved by the attacker. As this scenario would best be avoided, SSL certificates are used as a proactive measure to deter such attacks. This grants the data both confidentiality and authenticity, but because anyone can access the website and there are no authentication protocols set in place this last component is still vulnerable to DoS attacks and it would be preferable to implement some prevention mechanisms.

The first three components send data that is not protected between tiers and this makes it vulnerable to attacks. However, the information sent by the system is not that confidential so disclosing it does not pose much of an issue. Tampering, though, is a more dangerous threat as it is not desirable for corrupted data to reach either tier 2 or 3. It would be preferable that digital signatures would be used on the messages sent from tier 1 to ensure authenticity. Coupling that with public key encryption of the messages would give a good layer of security to the communication between the first two tiers.

The second tier sends data to the third tier to be stored in the database. Because there would already be some mechanisms implemented on the first tier, some less tight security mechanisms could be implemented just in case the previous mechanisms fail. Because there is a fixed number of servers on tier 2 and only one server on tier 3, symmetric key encryption could be used to communicate between those two tiers. As there are only three components that communicate with the third tier, the server would only need to store three keys. In order to also provide some form of authentication but not in the shape of digital signatures, the messages sent could also contain a MAC that the tier 3 server would verify when received.

# System Implementation

# Testing

# Results and Discussion

# Conclusion

# Project Future

# List of Appendices

# Sources of Information

# **Bibliography**

Danish Film Institute, 2012. *dfi.dk.* [Online]   
Available at: https://www.dfi.dk/files/docs/2018-02/f\_and\_f\_2012\_screen\_singlepages%20%281%29.pdf  
[Accessed 20 September 2018].

Danish Film Institute, 2017. *dfi.dk.* [Online]   
Available at: https://www.dfi.dk/files/docs/2018-02/Facts%20and%20Figures%202017%20%281%29.pdf  
[Accessed 20 September 2018].

Danish Film Institute, 2018. *dfi.dk/.* [Online]   
Available at: https://www.dfi.dk/files/docs/2018-04/Facts-and-figures-2018\_0.pdf  
[Accessed 20 September 2018].

http://www.upedu.org/, 2018. *http://www.upedu.org/.* [Online]   
Available at: http://www.upedu.org/  
[Accessed 20 September 2018].

media.netflix.com, 2012. *media.netflix.com.* [Online]   
Available at: https://media.netflix.com/en/press-releases/netflix-launches-in-sweden-denmark-norway-and-finland-migration-1  
[Accessed 20 September 2018].

Motion Picture Association of America, 2017. *mpaa.org.* [Online]   
Available at: https://www.mpaa.org/wp-content/uploads/2017/03/MPAA-Theatrical-Market-Statistics-2016\_Final.pdf  
[Accessed 20 September 2018].

Nielsen, M. B., 2018. *dst.dk.* [Online]   
Available at: https://www.dst.dk/en/Statistik/emner/kultur-og-kirke/film-boeger-og-medier/biografer-og-film  
[Accessed 20 September 2018].

POSitive Cinema, 2018. *http://positivecinema.com/.* [Online]   
Available at: http://positivecinema.com/  
[Accessed 20 September 2018].

Scrum.org, 2018. *Scrum.org.* [Online]   
Available at: https://www.scrum.org/resources/what-is-scrum  
[Accessed 20 September 2018].

Vista Cinema, 2018. *Vista.* [Online]   
Available at: https://www.vista.co/en/vista-products/vista-products/cinema/  
[Accessed 20 September 2018].