Technical Report - Project specifications

# 2SeeONot? 2SeeOrNot? ???

| | |
|---|---|
| Course: | IES - Introdução à Engenharia de Software |
| Date: | Aveiro, 02-12-2020 |
| Students: | 93147: David Morais<br>92964: Mariana Ladeiro<br>89318: Pedro Iglésias<br>93442: Wei Ye |
| Project abstract: | Web app that provides a fast and enjoyable way of browsing and acquiring tickets. The goal is to develop an app that lists available movies and cinemas to the user, so that they can check movies plot and information, as well as schedules and be able to buy tickets for various movie sessions. In addition, it also provides a clean and user-friendly interface that allows users to manage their bought tickets via a developed API. Furthermore a cinema owner can also create a page for its own cinema. |

Table of contents:

# 1 Introduction

This project is going to develop on the scope of IES, and the main objective is:

- ➔ Development of the product specification, from use cases to technical design.
- ➔ Propose, justify and implement a software architecture based on business structures.

<insert project name here>, a web / mobile app to allow and facilitate the marketing of movie tickets.

We created a git repository to develop our product and PivotalTracker backlog so that we could create and track our objectives.

//Quando estiver completo eleminar estas linhas

<Note: you may prepare the report in English or Portuguese, but don't mix languages. Adapt the template if using Portuguese.>

<background info with respect to the objectives of this project assignment in the scope of the IES course,...>

# 2 Product concept

## Vision statement

Our product is based on showing detailed information about each movie.

So, it is possible for each person (client) using the app or web page:

- Search for a movie;
- Search for popular movies;
- Search for recent movies;
- Check movie info (it will not show current ticket sales stats);
- Search for cinemas selling a movie ticket;
- Search for cinemas;
- Add favourite movie;
- Enable notifications (which notifies you when a movie ticket is being sold by a cinema);

Also, each cinema will be another type of client that will be able to:

- Search for a movie;
- Search for popular movies;
- Search for recent movies;
- Check movie info (it will show current ticket sales stats);
- Add a movie to the ticket sales;
- Check cinema currently selling movie tickets (maybe a graph of the sales?);

- Check top movie sales (maybe a graph?);

There is also an admin user type that will be able to:

- Search for a movie;
- Search for popular movies;
- Search for recent movies;
- Check movie info (it will show current ticket sales stats);
- Check top movie sales (maybe a graph?);
- Add or remove a movie;
- Check cinema sales(maybe a graph?);

This product will facilitate the life of the users that will join a large number of movies and cinemas so that it can be easier to buy and sell tickets.

//Quando estiver tudo eleminar estas linhas

<functional (black-box) description of the application: what will you system be used for? Which is the high-level/business problem being solved by your system?>

<if needed, clarify what was planned/expected to be included but was changed to a different approach/concept >

<optional: how is your system different or similar to other well-known products?>

<optional: you may include a UML Use Case diagram to support the explanation>

<optional: additional details on the process for the requirements gathering and selection (how did we developed the concept? Who helped us with the requirements? etc)>

## Personas

### Marta

Marta is the owner of an independent cinema in downtown Lisbon. Due to the business being slow she wants to add visibility to the business without losing the independent soul of the cinema. So, she's looking for a platform where she can add her cinema and sell tickets with all the gains being reverted to her and her staff.

### Lídia

Lídia is 75 years old and lives in Aveiro. Because she is retired, she has a lot of free time and likes to spend it at the movies, especially with her grandchildren. Since Lidia isn't very familiar with technologies, she is looking to find an app with a simple and easy interface.

### Samuel

Samuel is a 32 year old engineer living in Coimbra. Samuel is currently working on a mobile app that requires the use of an API with movie details. He's looking for a simple, perceptible and well documented API in order to make his job easier.

## Main scenarios

First Scenario

Marta decides to use the app/website to add her independent cinema. To do this,

she fills out a form giving the cinema information and submits it. The request is sent to the admin and, if accepted, she can choose the movies playing in her cinema every week based on the movies available as well as schedules, room and seat info.

Second Scenario

Lídia's grandchildren decided last minute they would like to go the movies. Lídia opens the website to see some animation movie options. The movie they decide to watch is almost starting, so, to make the process quicker, Lídias uses the website to purchase the tickets and get in time to the cinema. After the movie, Lídia and her grandchildren give it a rating.

Third Scenario

Samuel is in charge of developing an app that shows movie details, like actors, production and plot. He's now taking care of passing the data from an API to the app. While researching, he finds the website API and decides to use it. In order to do this, he uses the method GET from the API receiving the data he wanted in JSON format.

# 3 Architecture notebook

For this project, we used Frontend Backend Separation Architecture. Backend and frontend components were developed separately.

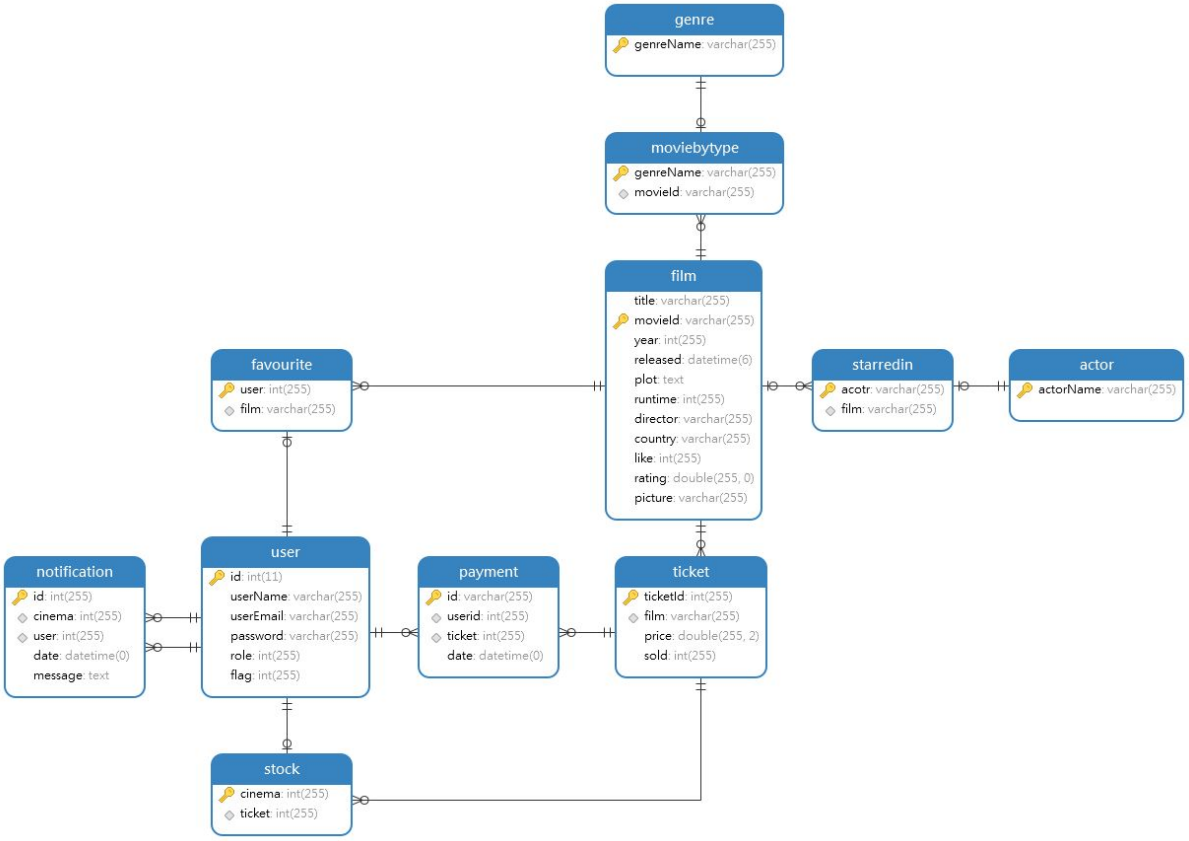
## Key requirements and limitations

- The system must provide a user authentication system.
- Different types of users should have different API access. If a user doesn't have access, the request should be refused.
- The system should avoid, as much as possible, the access to the database (use the cache engine).
- The system should guarantee data consistency (example: failure while buying tickets can't cause data inconsistency).
- The system should implement every use case.
- The server should communicate with the frontend through the http protocol.
- The system should analyse the acquired data.
- The system should use a message queue to optimize a few time-consuming operations.

## Architetural view

- **Technologies**
  - **Database**
    - Mysql: It's one of the most common relational dbs on the market that we are more familiarized with. It serves to persist the main data of the app.
    - Redis: A NoSql db of the type Key-Value, extremely efficient, provides more data structures than other Key-Value dbs (ex: memcache). The cost of accessing a db (mysql) is expensive, we should avoid it, as much as possible, so we used Redis as a cache of the db queries results.
  - **Backend**
    - Springboot
  - **Message Queue**
    - RabbitMQ. When a client buys a ticket, the system should respond immediately whether the purchase was made or not. Since the purchase can be a simple operation, sending emails or billing can take time.
  - **Authentication**
    - Spring Security + JWT. Used for the user authentication and authorization of the received requests (we have to verify if the user that sends the request has access to a certain API or not).
  - **Frontend**
    - Angular

- **Database model**




| | |
|---|---|
| User | Table that stores the system users. A user can be a normal client, a cinema owner or the admin. |
| Film | Table that stores all the information about each movie. |
| Favourite | Table that stores each user's favourite movies. |
| Stock | Table that stores the ticket stock of each cinema. |
| Payment | Table that stores all the ticket |

| | |
|---|---|
| | purchases made. |
| Ticket | Table that stores all the tickets. |
| Genre | Table that stores movie genres. |
| MoviebyType | Table relating the film to its type |
| Notification | Table that stores all the notifications sent from a cinema to a client. |
| Actor | Table that stores all the actors. |
| StarredIn | Table that relates the actor with the film |

## Module interactions

- **Frontend communication**
  - The communication is made through HTTP. The frontend sends a http request to the server, this request can be rejected due to authentication/ authorization failure. If not, it enters one of the backend endpoints. AFter processing the request, the backend returns a serialized JSON object (that contains the response, http status code and other necessary information) to the frontend.

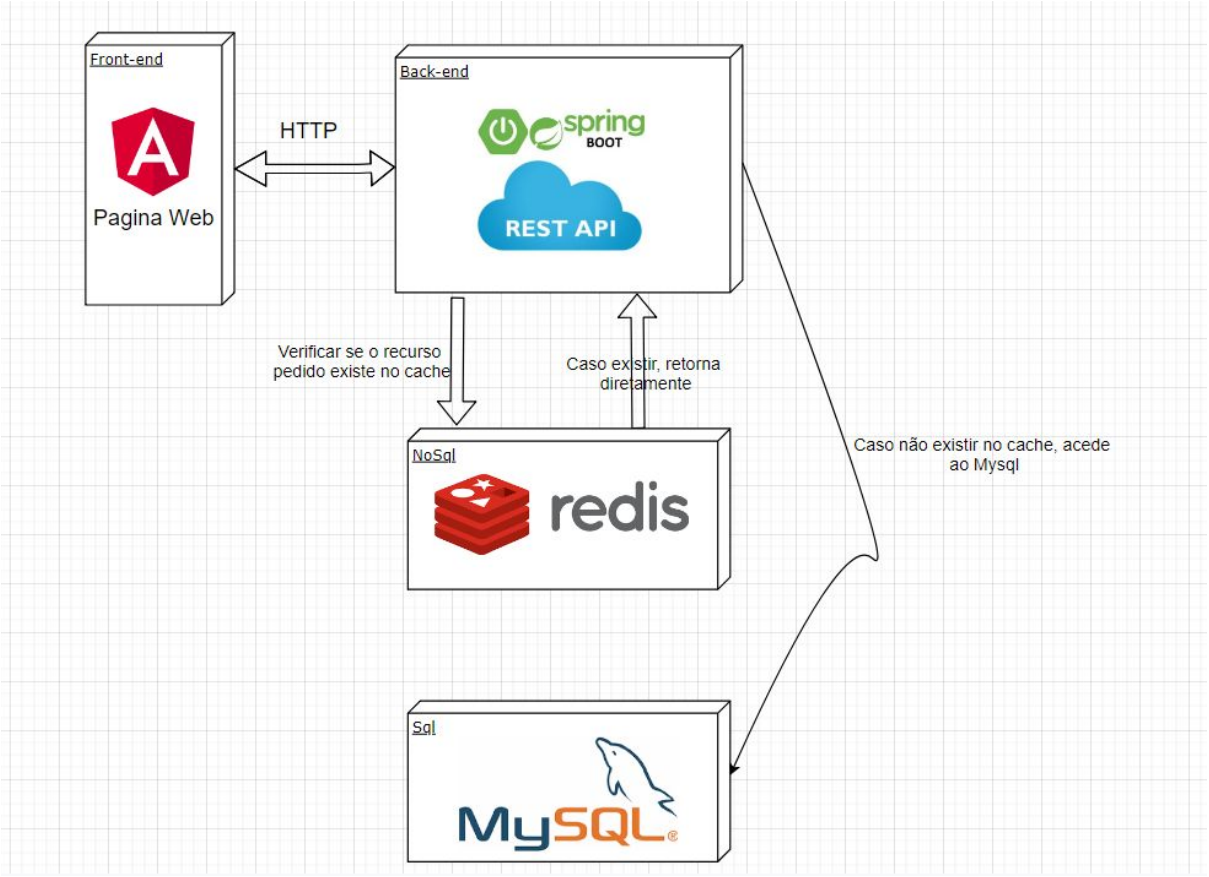
- **Protocolo Json**
  - {

    “statusCode”: xxx (http status code defined by us),

    “message”: xxx (message that describes an answer, ex:success, authentication failed, etc.),

    “data”: xxx (data returned from the server)

    }




# 4 Information perspetive

<which concepts will be managed in this domain? How are they related?>

<use a logical model (UML classes) to explain the concepts of the domain and their attributes>

# 5 References and resources

API used for collecting movie info;

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>