MMA TourSafe 2

Analysis and Design Document

Student: Nagy Laura - Beatrix

**Group: 30432**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Design and implement an application that helps MMA tournaments manage their scheduled fights better while ensuring covid safety standards.

# Functional Requirements

## A manager should be able to create a tournament and invite fighters to sign up for a venue. Each tournament requires weekly or monthly matches to generate traction and revenue within the tournament period.

## After they sign up, in order to ensure proper safety standards, fighters are required to present themselves at a tournament isolation bubble with a test which will be recorded and they will immediately be tested again on site. If the “arrival” test is positive the fighter is then moved to quarantine until the test results are negative again.

## In order for a fighter to take part in a tournament they require at least 3 weeks of negative test history after which they can be matched up with similar fighters of their caliber.

## A manager should be able to see in real time the tournament schedule being populated with eligible fighters (at least bi-weekly).

# Non-functional Requirements

-database generation

-scalability

-testability

2. Use-Case Model

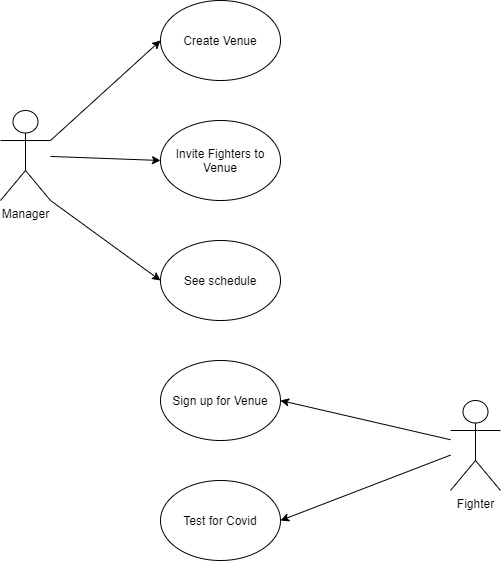
Use case: create venue

Level: user-goal

Primary actor: manager

Main success scenario: create venue

Extensions: error creating venue

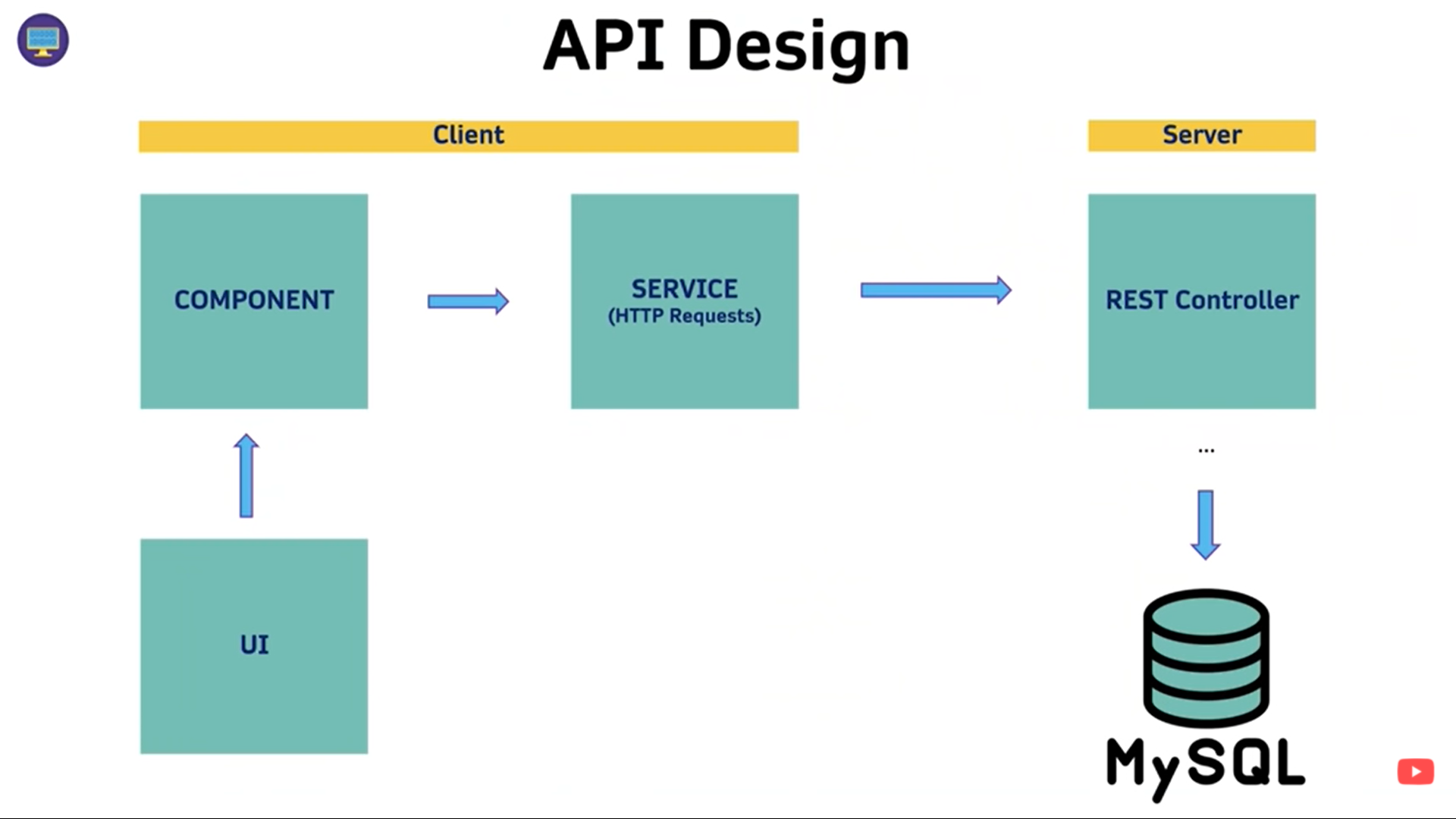
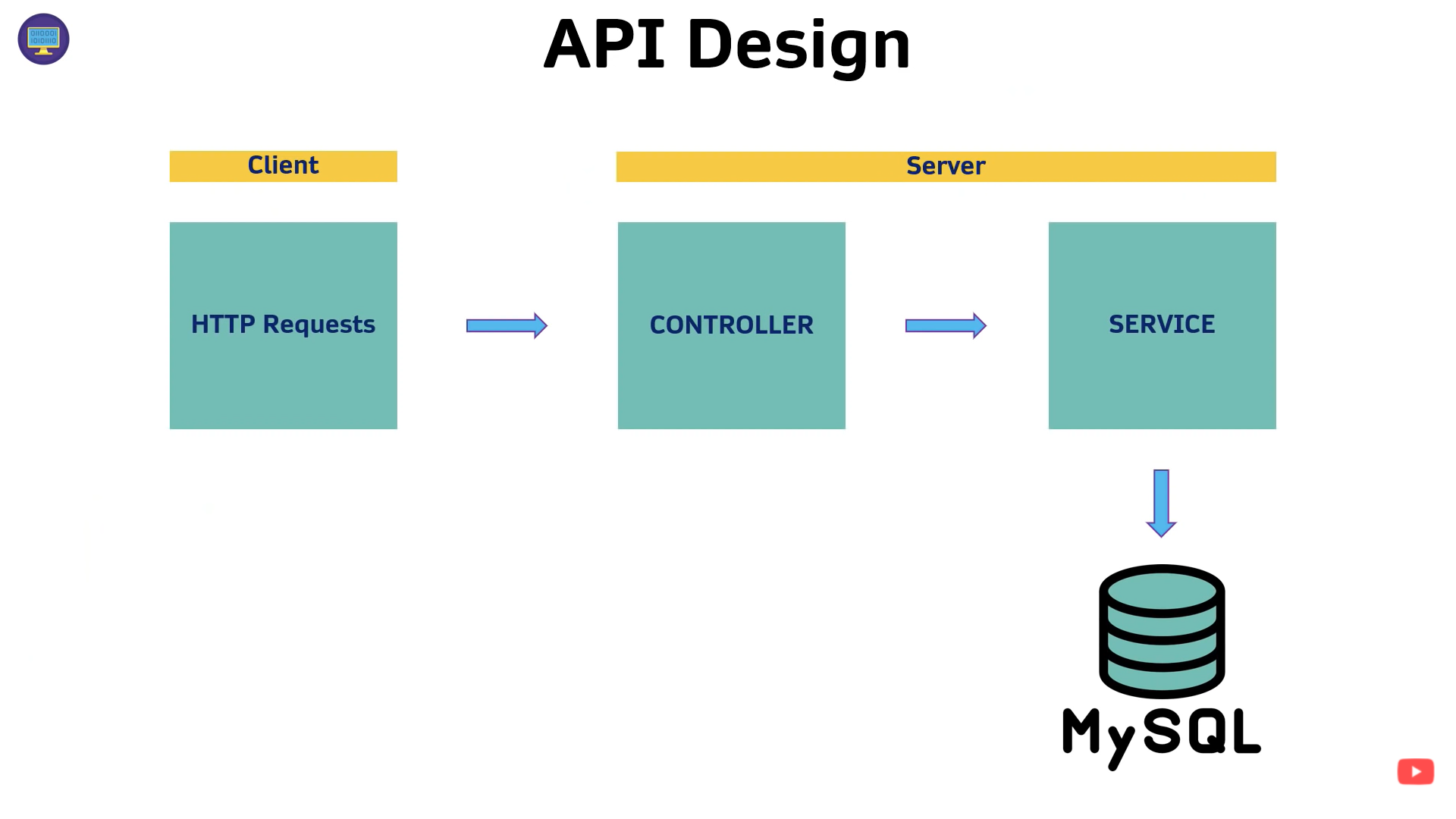
**

3. System Architectural Design

**3.1 Architectural Pattern Description**

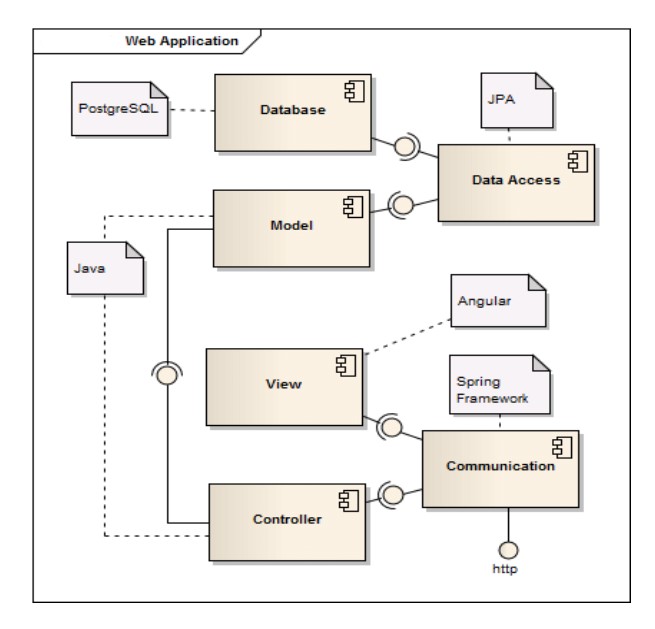
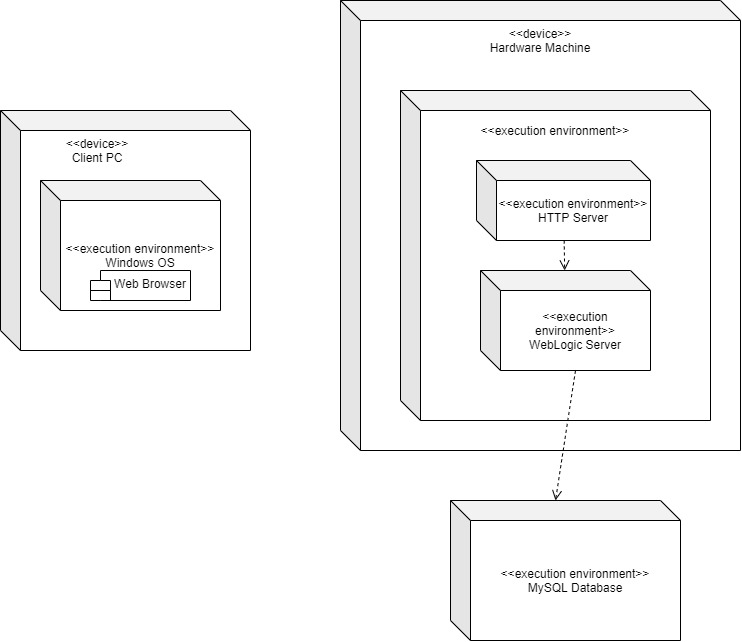
The client-server architecture is a computing model, where the server hosts, distributes and controls the majority of the resources as well as services to be used by the client. Such structural designs are made up of one or more client systems connected to central or main servers through a network, which we usually know as an Internet connection. All such systems associated with it share computing resources.

**3.2 Diagrams**



The server side of the application is represented by the backend application, which contains the Rest Controller and the Data layer. Controller layer will access the Service and the Data layer, and then respond to requests coming in from any client. The requests will come through the Controller. The controller is supposed to be taking the requests and passing them over to the Service. The Service along with other utility classes and access the data from the service. The Controller is the part of the server that will be exposed to the world.

The client side of the application is represented by the frontend application. The frontend is composed of other parts than the backend application. The UI part is what the user is able to see on the screen. The UI will have access to the Service through the Component, and that Service will be what will have access to the backend by the means of making HTTP requests.



4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*

5. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

6. Data Model

The conceptual data model is used to organize and define concepts and rules. It consists of the static business structures and concepts.

The logical and physical data models are used to demonstrate the implementation of a system using MySQL database.

7. System Testing

To test the correctness of the CRUD operations, I have used Postman. Postman is a popular API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses.

8. Bibliography

1. <https://spring.io/projects/spring-boot>
2. <https://www.baeldung.com/learn-jpa-hibernate>
3. <https://restfulapi.net/rest-architectural-constraints/>
4. <https://angular.io/guide/observables>
5. <https://learning.postman.com/docs/getting-started/introduction/>