Software Architecture Document

Fit buddy

Semester 3

Stoyan Grozdev

Version Table

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 21.03.2023 | Stoyan Grozdev | Document Created |
|  |  |  |  |
|  |  |  |  |

Table on contents

1. Document overview

2. Architecture overview

2.1 Context

2.2 Containers

2.3 Components

3. Architecture decisions

Document overview

This article aims to provide insights into the technology choices and software architecture for a fitness training management system. Its goal is to describe the high-level structure of the system. This document can be used by technical instructors to check whether the project complies with the requirements and to understand how the system works.

System context diagram

The Fitness App will function alone, not requiring any assistance from outside systems. The system supports three different user roles: administrator, trainer, and customer. Each has certain objectives and associated privileges. The software system is intended to be self-sufficient, offering all required functionality inside without requiring integration with external systems, in order to meet the project's objectives.

Diagram

Description automatically generated

Container diagram

Reasons for the Two Different Uses

Two distinct containers will make up the fitness software system. A Java Spring Boot application will manage business logic, retrieve and write data to the database, and act as a REST API on the backend. On the other hand, React, a JavaScript-based library, will be used to create the frontend. Via HTTP requests, these two programs will communicate, exchanging data in JSON format.

The flexibility of the project is the main justification for using two separate apps or containers. The backend can support several frontend interfaces by keeping the frontend apart from it. A solid basis for future scalability and extensibility is laid by this separation.

Explanation: Rationale for Choosing Java as the Backend Programming Language

This software system's backend application was chosen to use Java based on a number of considerations. Its resemblance to C# firstly enables a quicker learning curve. Second, Java is one of the most extensively used programming languages in use today, thus there is a wealth of online documentation and tools to aid in problem-solving. Third, because the code can be run on any platform, its platform independence allows the freedom to transition between operating systems without restrictions.

Last but not least, a variety of premade tools and libraries are available to hasten and streamline the development process, including Lombok, a framework that minimizes boilerplate code, and Gradle, a build automation tool. Java is a good option for this project's backend application because of these benefits.

Explanation: Selection of Backend Framework

For the backend application, the Java Spring Boot framework was selected for a number of reasons. First of all, it takes care of numerous configurations and dependencies automatically, freeing developers to concentrate more on the project itself. Second, Spring Boot comes with an embedded server called Tomcat that makes it easier to test the application while it is still in development.

Explanation: Selection of Backend Framework

For the backend application, the Java Spring Boot framework was selected for a number of reasons. First of all, it takes care of numerous configurations and dependencies automatically, freeing developers to concentrate more on the project itself. Second, Spring Boot comes with an embedded server called Tomcat that makes it easier to test the application while it is still in development.

Explanation: Database Selection

A relational database has many benefits, chief among them the simple SQL queries that can be used to retrieve data and the relational database's natural data architecture. Due to prior experience, MySQL was selected as the relational database management system (RDBMS) for this project, making it a practical and effective option.

Diagram

Description automatically generated

Component diagram

Diagram

Description automatically generated

The User Controller and Exercise Controller are just a few of the REST controllers that will make up the backend application. The REST API endpoints will be specified by these controllers. Users, and exercises will be the application's main resources.

Architecture decisions

Client-Server Architecture:

The frontend React application (the client) will communicate with the REST API (the server), achieving separation of concerns and modularity. This approach ensures that the system is not tightly coupled or directly dependent.

Key benefits include:

-Modularity

-Maintainability

-Extensibility

Dependency Inversion:

Dependency inversion will be implemented through the use of interfaces, which will be injected into the classes utilizing them. This approach ensures that classes rely on abstractions rather than concrete implementations.

Key benefits include:

Adherence to the SOLID principle of Dependency Inversion

Flexibility

Extensibility

Layered Architecture:

The backend application will utilize a layered architecture, consisting of the Controllers Layer, Business Layer, and Data Access Layer. This approach achieves separation of concerns, as each layer is responsible only for its designated functions. It also reduces dependencies, since the operation of each layer is isolated from the others. This separation simplifies testing, as each component can be evaluated individually.

Key benefits include:

Separation of concerns

Readability

Adherence to the SOLID principle of Single Responsibility