



Individual Project  
**Software Architecture Document**

Date: 16.05.2024

Version: Version 1.2

# Version History

Version	Date	Author	Changes	State
1.0	28.03.2024	Claudiu Badea	Initial Document	Complete
1.1	19.04.2024	Claudiu Badea	Updated technology choices and design decisions	Complete
1.2	16.05.2024	Claudiu Badea	Updated Diagrams	Complete

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose	4
1.2	Scope	4
<b>2</b>	<b>System Context</b>	<b>4</b>
2.1	Business Context	4
2.2	System Overview	4
<b>3</b>	<b>Containers and Technology Choices</b>	<b>6</b>
3.1	Backend Container	6
3.2	Frontend Container	6
<b>4</b>	<b>Components</b>	<b>10</b>
4.1	Backend Components	10
4.2	Frontend Components	12
<b>5</b>	<b>CI Pipeline</b>	<b>12</b>
5.1	Build Phase	12
5.2	Test Phase	12
5.3	Sonar Check Phase	12
<b>6</b>	<b>Security and Authentication</b>	<b>14</b>
<b>7</b>	<b>Conclusion</b>	<b>14</b>



# 1 Introduction

## 1.1 Purpose

This Software Architecture Document (SAD) offers a detailed overview of the architectural framework for **QWEST**. It delineates the high-level design choices, structural components, and technologies that underpin the development and functionality of the system.

## 1.2 Scope

The primary objective of the **QWEST** is to simplify and personalize the travel planning process. By offering an adaptive and user-friendly platform, it aims to cater to the unique needs and preferences of travellers worldwide, making travel planning an enjoyable and hassle-free experience.

# 2 System Context

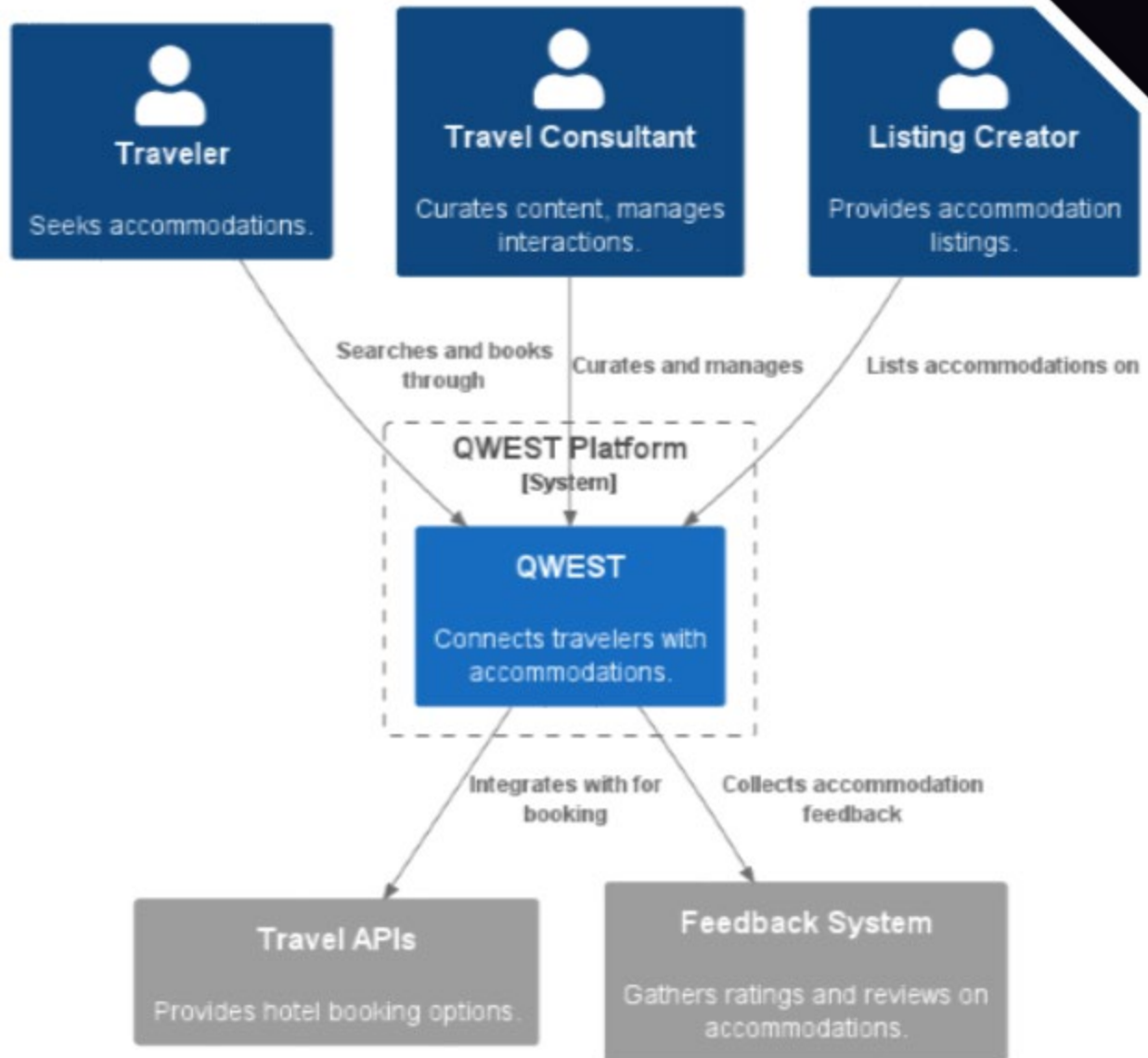
## 2.1 Business Context

Travelers currently face the challenge of using multiple platforms to plan trips. **QWEST** aims to unify these aspects into a single, streamlined service, enhancing the user experience by offering a platform that adapts to their personal travel preferences and simplifies the entire planning process.

## 2.2 System Overview

**QWEST** consists of a backend powered by Java Spring Boot, which provides RESTful APIs, and a frontend developed with NextJS, facilitating a dynamic and engaging user interface.

### System Context Diagram for QWEST - Stays



#### Legend

	person
	system
	external system
	system boundary (dashed)

## 3 Containers and Technology Choices

### 3.1 Backend Container

The contains the core backend infrastructure of QWEST, equipped with powerful API capabilities and server-side processing logic.

Technology Stack:

- **Spring Boot**: Chosen for its efficiency in development speed, automatic configuration abilities, and comprehensive support within the Spring ecosystem, which is perfect for building microservices.
- **RESTful API**: Provides a stateless, consistent interface that simplifies the integration and interaction between the frontend and backend elements.

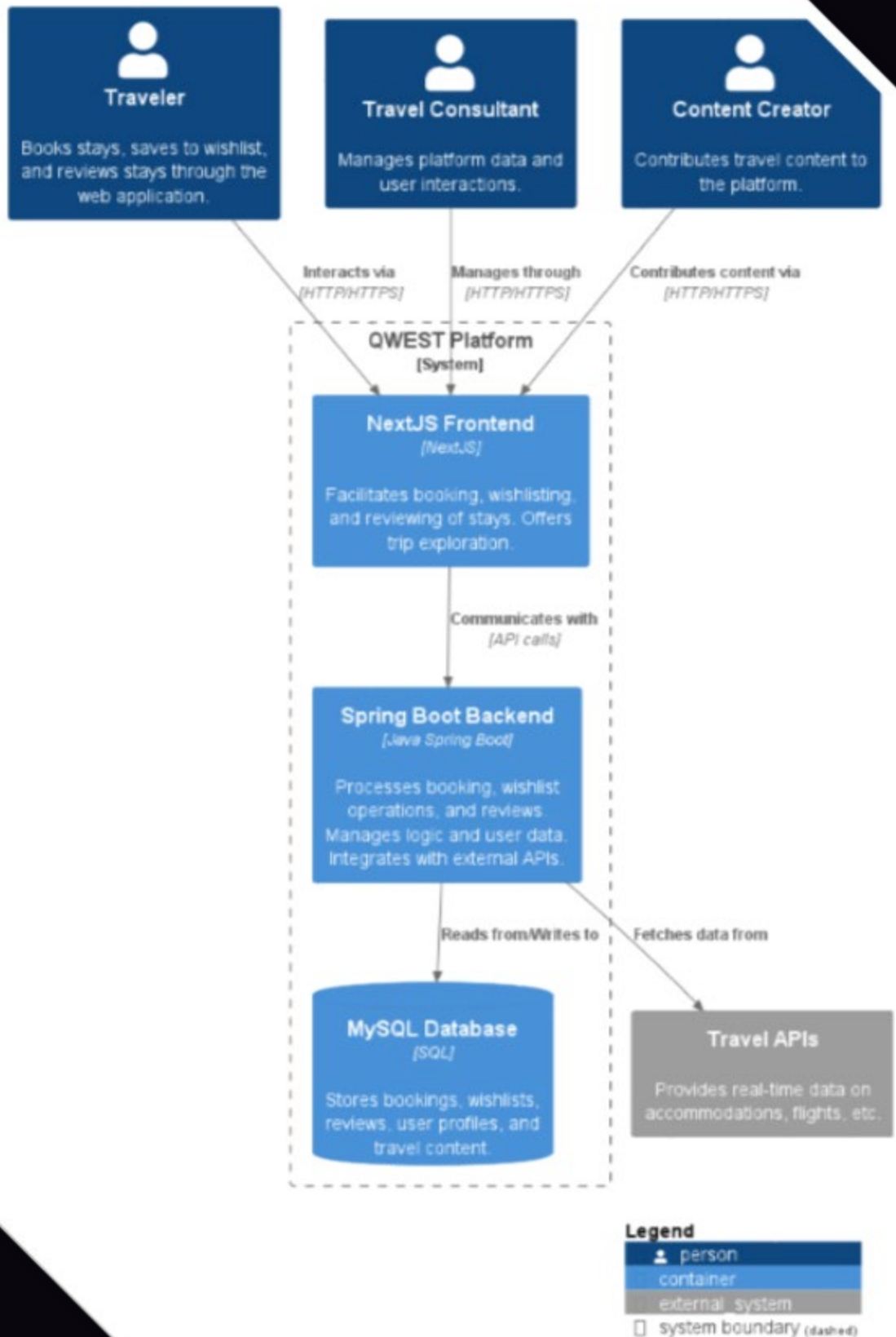
### 3.2 Frontend Container

The contains the core frontend infrastructure of QWEST, which enables a seamless and responsive user experience.

Technology Stack:

- **Next.js**: Adopted for its capabilities in server-side rendering and generating static websites, crucial for enhancing the platform's performance and SEO. Its automatic routing system and support for React's component-based architecture also streamline development, making it ideal for building interactive web applications.

Container Diagram for QWEST - Enhanced Traveler Interactions





## 4 Components

### 4.1 Backend Components

The backend framework is structured into three principal layers:

**Persistence**, **Business**, and **Controller**, detailed as follows:

**Persistence**: Oversees data storage and access, working with the MySQL database to guarantee effective and secure data management. Elements:

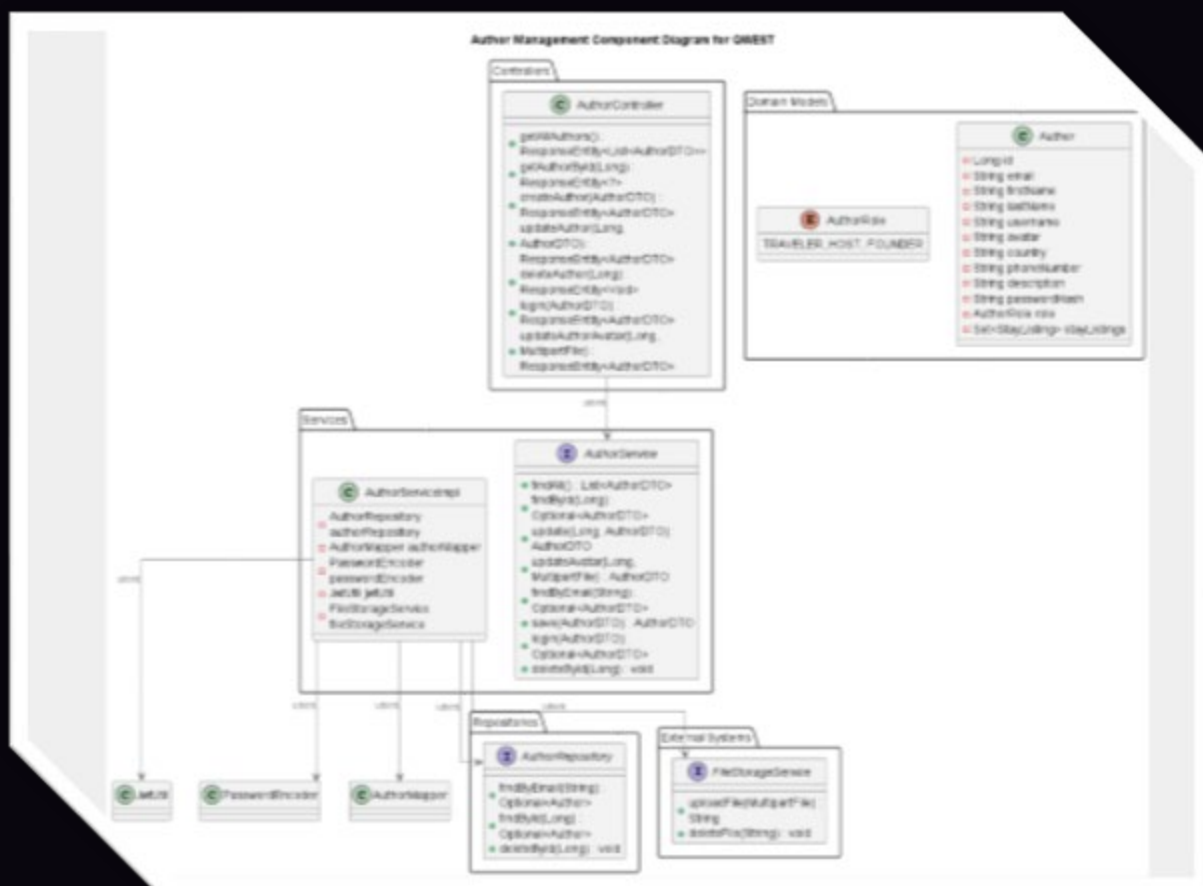
- **Entity Classes**: Correspond to MySQL tables, serving as data models.
- **Repositories**: Utilize JpaRepository for object-relational mapping, streamlining database operations.

**Business**: Encapsulates the application's primary logic, modifying data received from the Persistence layer for application consumption.

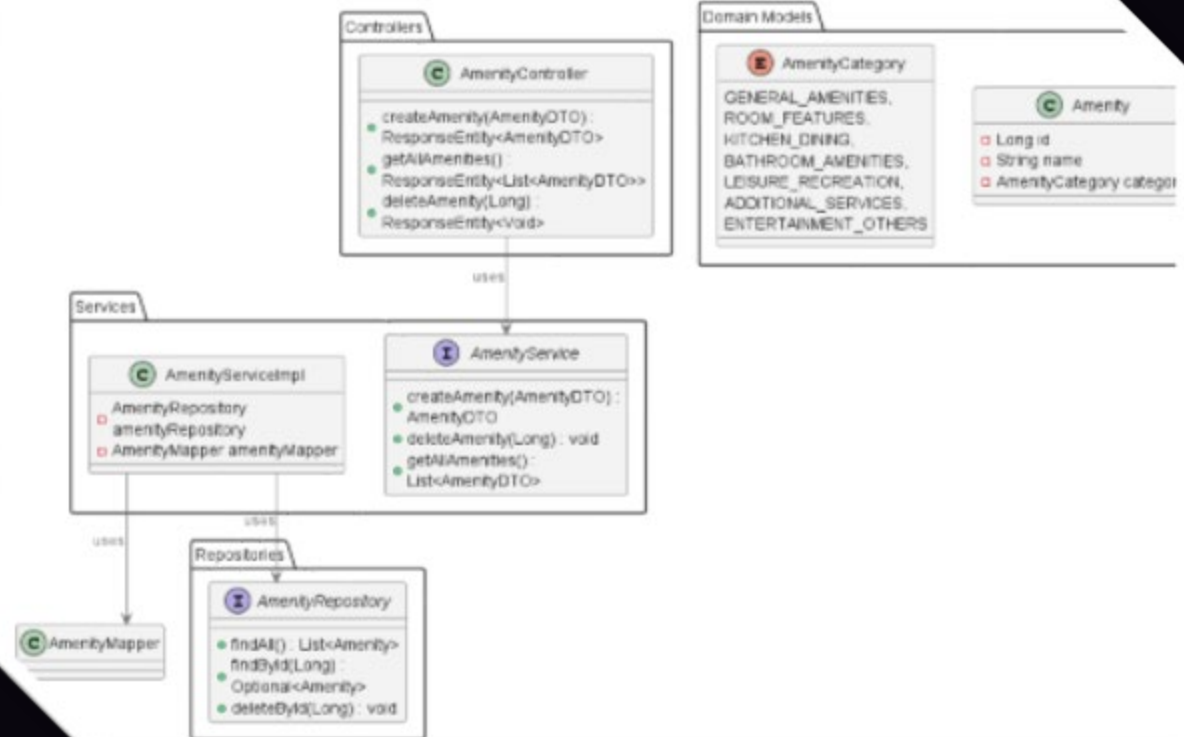
- **Service Classes**: Embed business logic, preparing data for the Controller layer.
- **DTOs (Data Transfer Objects)**: Support in-application data movement, embodying the YAGNI (You Ain't Gonna Need It) principle by omitting superfluous base classes.

**Controller**: Directs the flow of data between the user interface and the business logic, interpreting user inputs and delivering suitable responses.

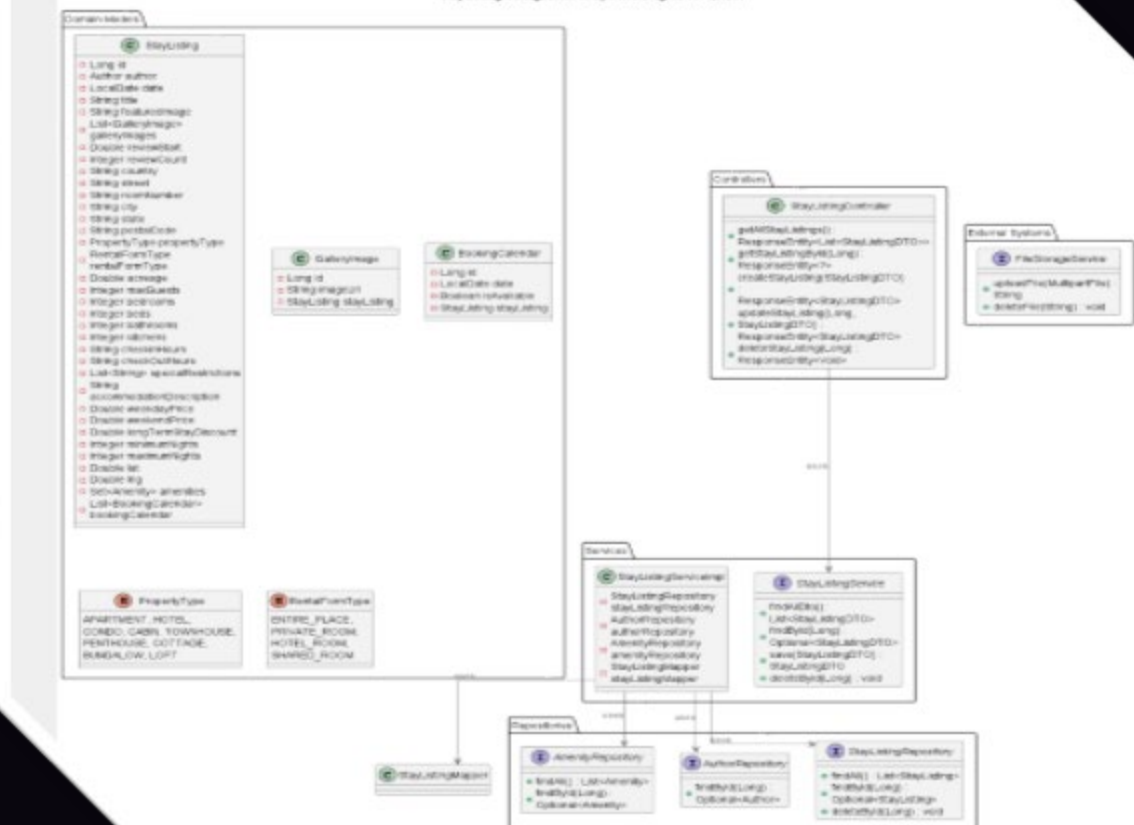
- **Controllers**: Connect the business layer with the frontend, directing the application's response to user interactions.



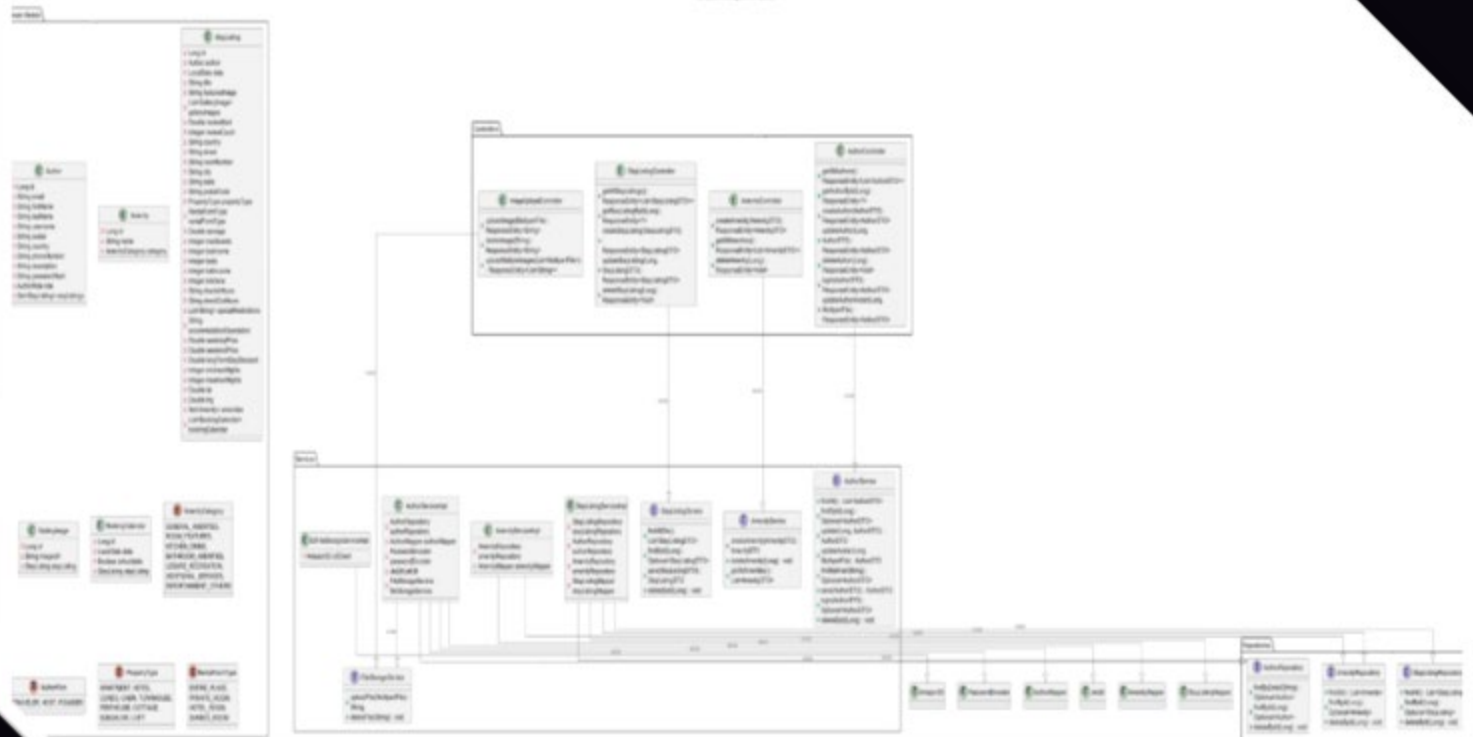
Amenity Management Component Diagram for QWEST



Stay Listing Management Component Diagram for QWEST



# Image Upload Management Component Diagram for QWEST



## 4.1 Frontend Components

### API

- Manages backend API requests, using HTTP methods for server communication.

### Components

- Acts as modular elements for the user interface, enabling the development of dynamic and engaging web pages.

### Pages

- Delivers the application's diverse visual representations, employing components to present information and interact with users.

## 5 CI Pipeline

### Continuous Integration (CI) Pipeline Stages

This pipeline is structured into two main phases: build and test.

#### Build Phase

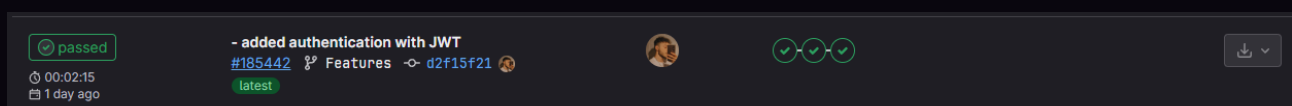
- Utilizes Gradle to compile and assemble the project components.

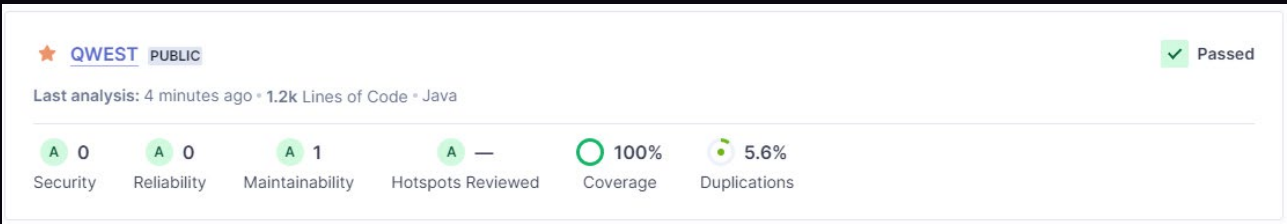
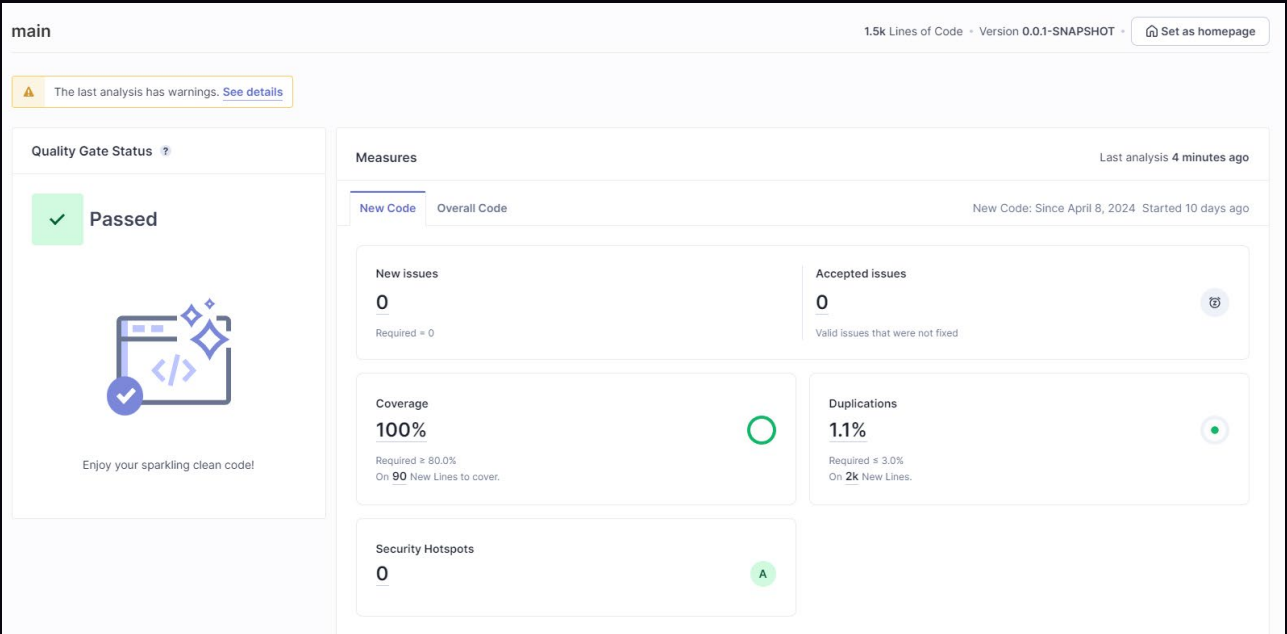
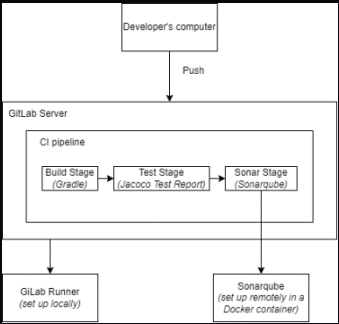
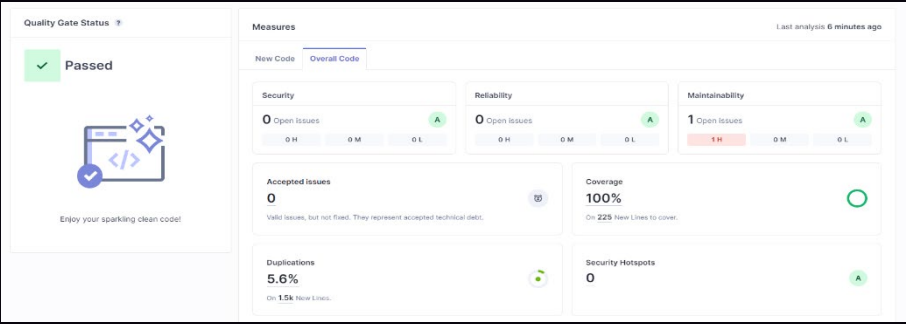
#### Test Phase

- Employs Gradle again, this time to execute the project's tests, ensuring functionality and reliability.

#### Sonar Check Phase

- Executes Gradle tasks (“test” and “jacocoTestReport”) and then triggers SonarQube analysis to ensure code quality and security standards are met.





## 6 Security and Authentication

Detailed mechanisms for securing the application and managing user authentication, including the use of Spring Security for robust, stateless API security.

## 7 Conclusion

This **Software Architecture Document** outlines the foundational architecture, design motivations, and technical foundations of the **QWEST** project. Through the adoption of methodologies such as SOLID and YAGNI, and the use of cutting-edge, effective technologies like Spring Boot and Next.js, **QWEST** is designed to provide a scalable, sustainable, and engaging platform for travelers.