QWEST

Individual Project
Software Architecture Document

Date: 19.04.2024

Version: Version 1.1

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

Contents

1 Introduction 4

1.1 Purpose 4
1.2 Scope 4

2 System Context 4

2.1 Business Context 4
2.2 System Overview 4

3 Containers and Technology Choices 6

3.1 Backend Container 6
3.2 Frontend Container 6

4 Components 8

4.1 Backend Components 8
4.2 Frontend Components 10

5 CI Pipeline 10

5.1 Build Phase 10
5.2 Test Phase 10
5.3 Sonar Check Phase 10

6 Security and Authentication 12

7 Conclusion 12

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.

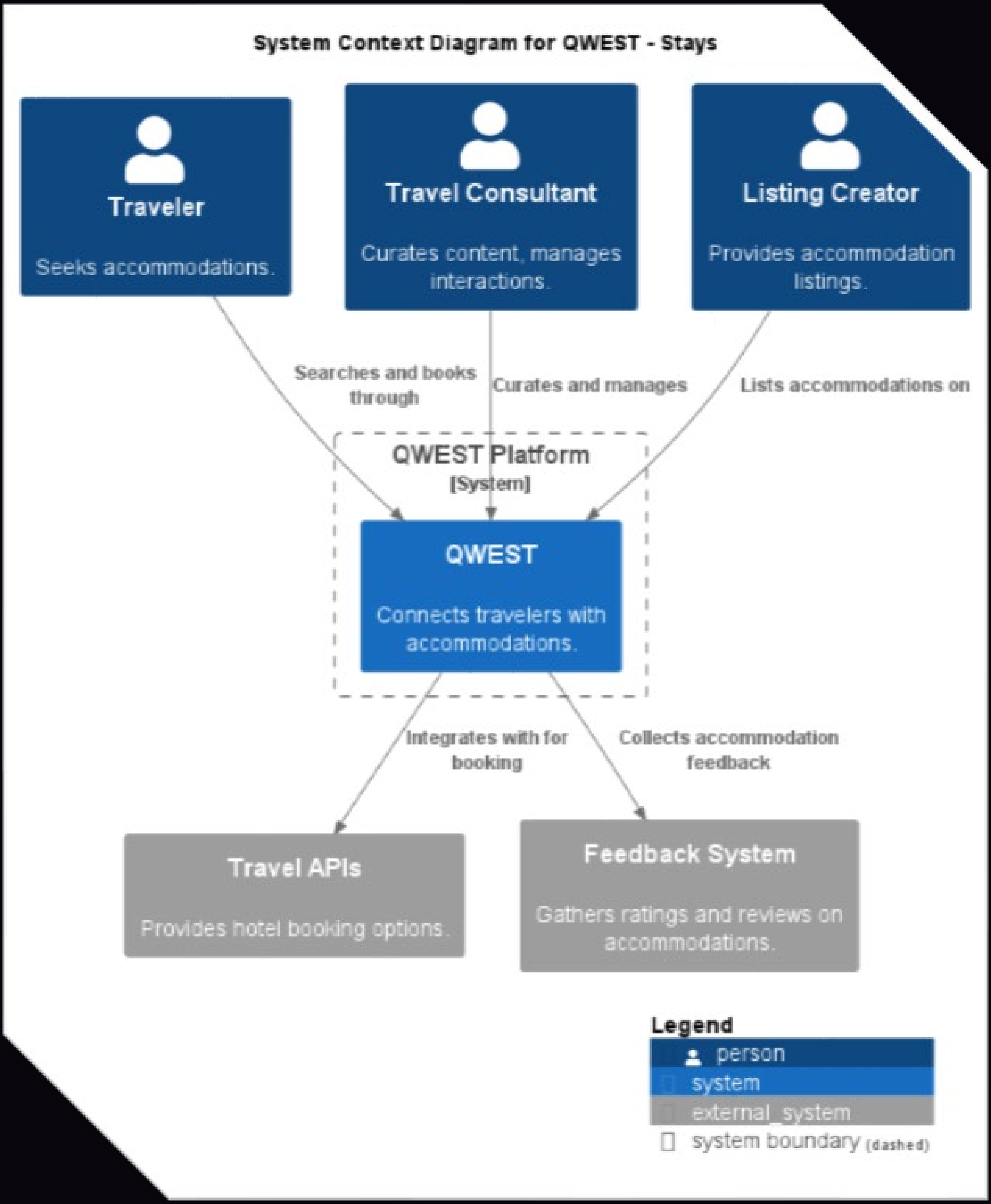
2 System Context

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
Traveler
Seeks accommodations.
Travel Consultant
Curates content, manages interactions.
Listing Creator
Provides accommodation listings.
Searches and books through
Curates and manages
Lists accommodations on
QWEST Platform
[System]
QWEST
Connects travelers with accommodations.
Integrates with for booking
Collects accommodation feedback
Travel APIs
Provides hotel booking options.
Feedback System
Gathers ratings and reviews on accommodations.
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:

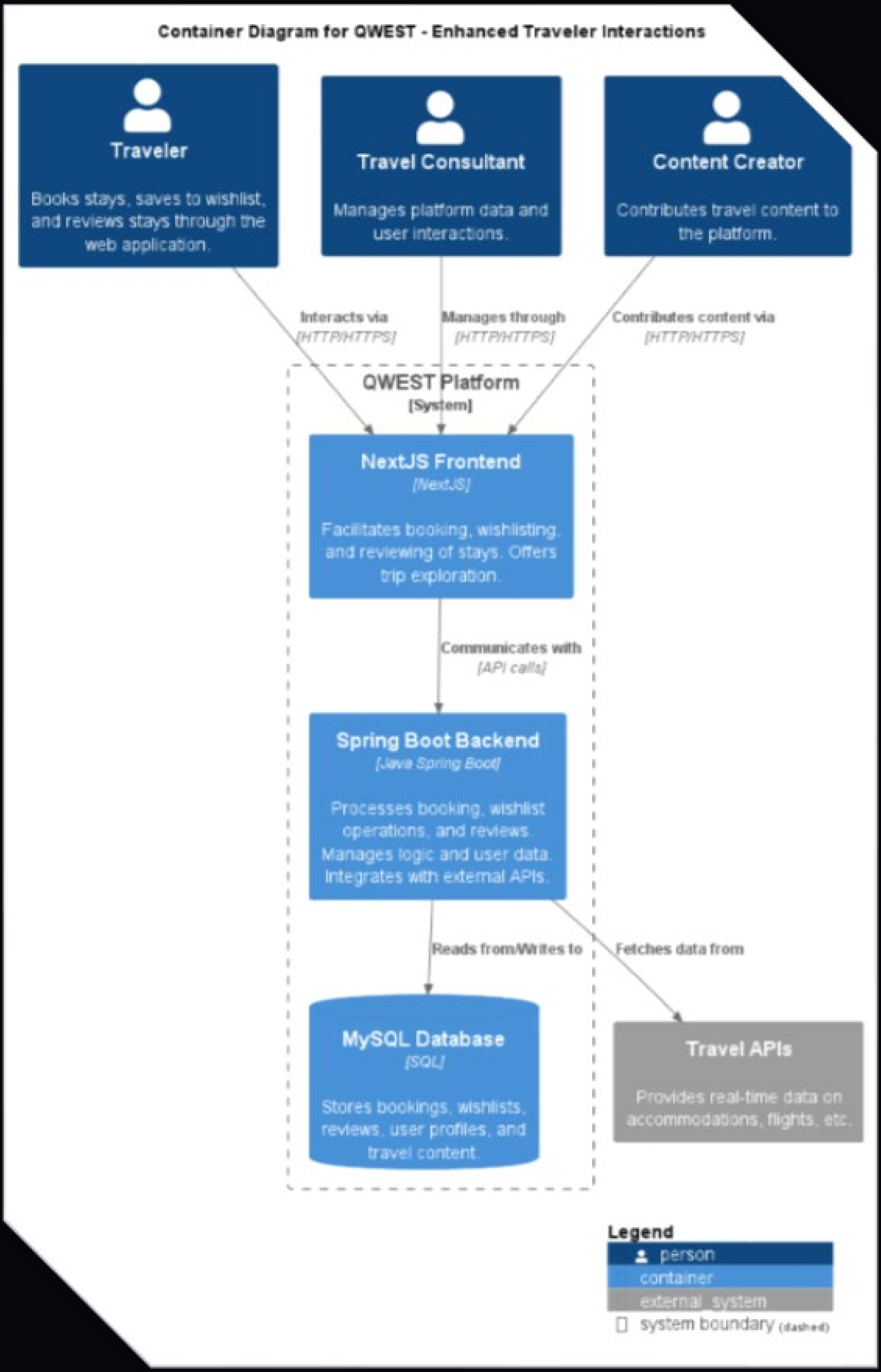
- Spring Boot: Chosen for its efficiency in development speed, automatic configuration abilities, and comprehensive support within the Spring ecosystem, which is perfect for building microservices.
- RESTful API: Provides a stateless, consistent interface that simplifies the integration and interaction 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
Traveler
Books stays, saves to wishlist, and reviews stays through the web application.
Travel Consultant
Manages platform data and user interactions.
Content Creator
Contributes travel content to the platform.
Interacts via
[HTTP/HTTPS]
Manages through
[HTTP/HTTPS]
Contributes content via
[HTTP/HTTPS]
QWEST Platform
[System]
NextJS Frontend
[NextJS]
Facilitates booking, wishlisting, and reviewing of stays. Offers trip exploration.
Communicates with
[API calls]
Spring Boot Backend
[Java Spring Boot]
Processes booking, wishlist operations, and reviews. Manages logic and user data. Integrates with external APIs.
Reads from/Writes to
Fetches data from
MySQL Database
[SQL]
Stores bookings, wishlists, reviews, user profiles, and travel content.
Travel APIs
Provides real-time data on accommodations, flights, etc.
Legend
person
container
external_system
system boundary (dashed)

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:

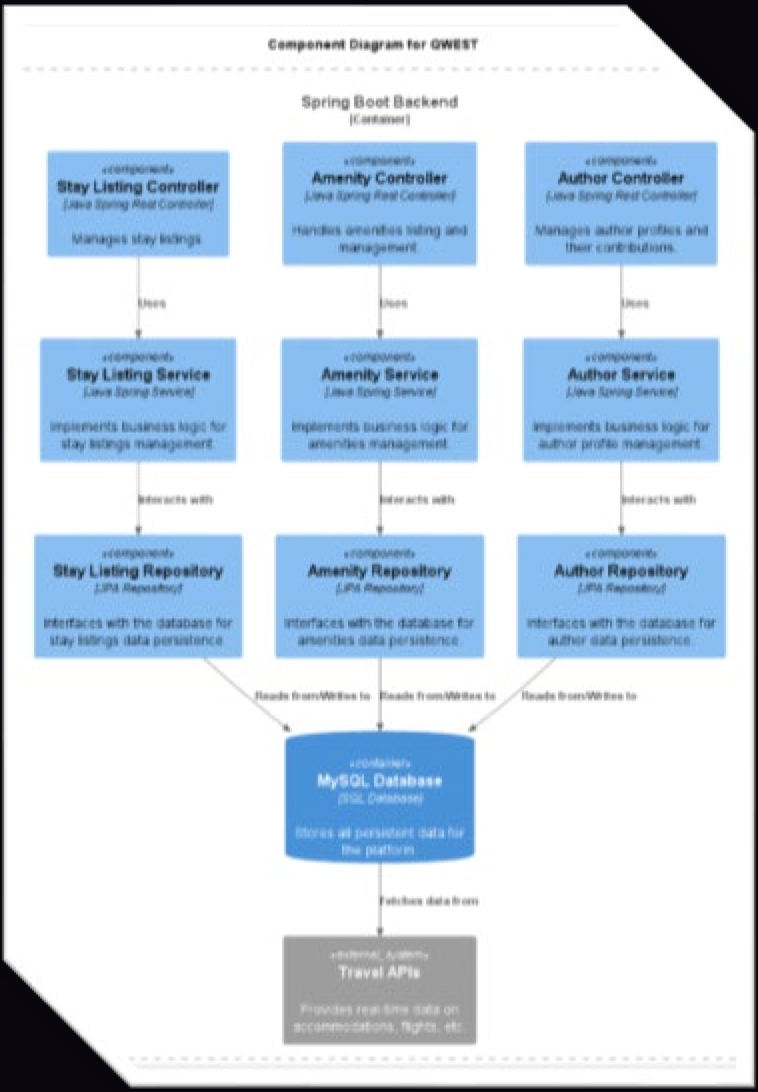
- Entity Classes: Correspond to MySQL tables, serving as data models.
- Repositories: Utilize JpaRepository for object-relational mapping, streamlining database operations.

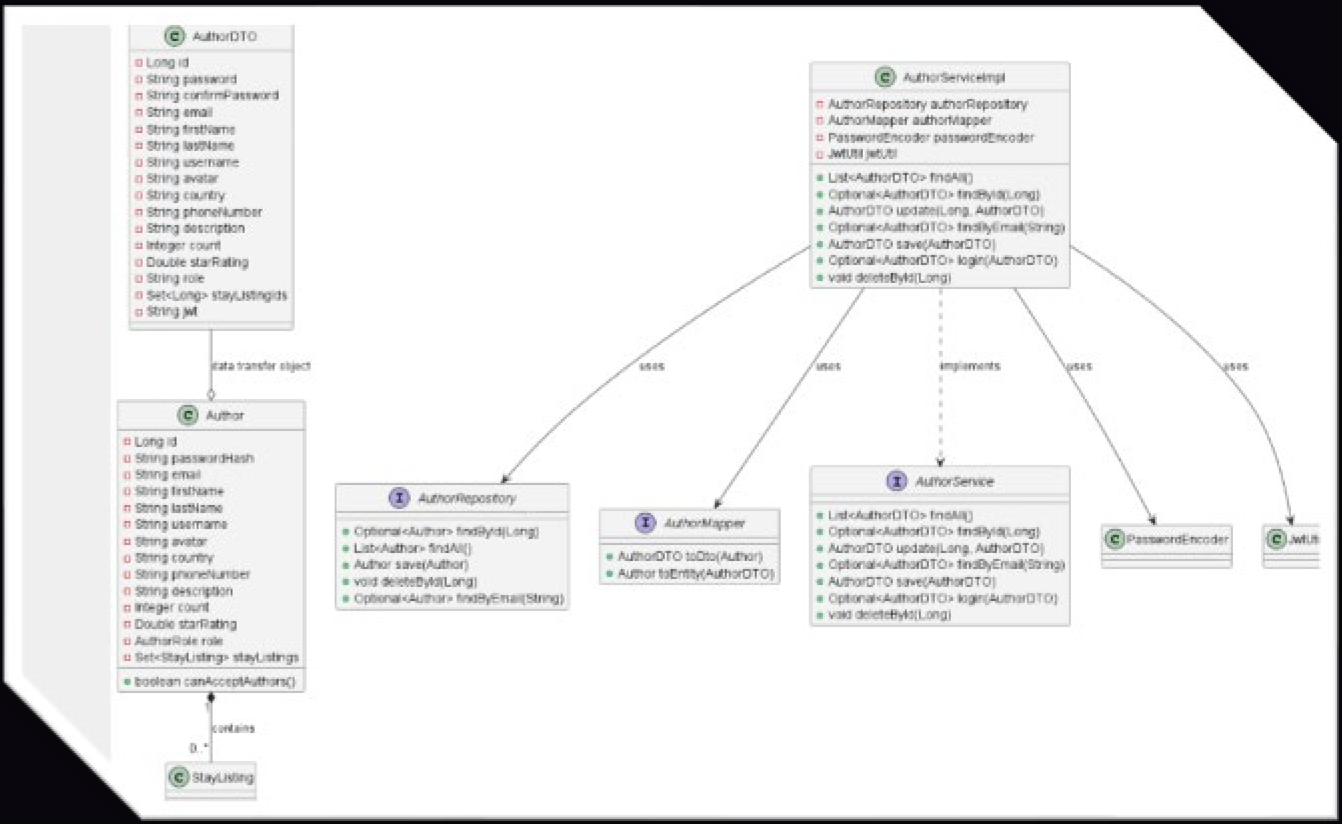
Business: Encapsulates the application's primary logic, modifying data received from the Persistence layer for application consumption.

- 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.


Component Diagram for QWEST
Spring Boot Backend
[Container]
«component»
Stay Listing Controller
[Java Spring Rest Controller]
Manages stay listings
«component»
Amenity Controller
[Java Spring Rest Controller]
Handles amenities listing and management
«component»
Author Controller
[Java Spring Rest Controller]
Manages author profiles and their contributions
Uses
Uses
Uses
«component»
Stay Listing Service
[Java Spring Service]
Implements business logic for stay listings management
«component»
Amenity Service
[Java Spring Service]
Implements business logic for amenities management
«component»
Author Service
[Java Spring Service]
Implements business logic for author profile management
Interacts with
Interacts with
Interacts with
«component»
Stay Listing Repository
[JPA Repository]
Interfaces with the database for stay listings data persistence
«component»
Amenity Repository
[JPA Repository]
Interfaces with the database for amenities data persistence
«component»
Author Repository
[JPA Repository]
Interfaces with the database for author data persistence
Reads from/Writes to
Reads from/Writes to
Reads from/Writes to
«container»
MySQL Database
[SQL Database]
Stores all persistent data for the platform
Fetches data from
«external_system»
Travel APIs
Provides real-time data on accommodations, flights, etc.


AuthorDTO
Long id
String password
String confirmPassword
String email
String firstName
String lastName
String username
String avatar
String country
String phoneNumber
String description
Integer count
Double starRating
String role
Set<Long> stayListingIds
String jwt
data transfer object
Author
Long id
String passwordHash
String email
String firstName
String lastName
String username
String avatar
String country
String phoneNumber
String description
Integer count
Double starRating
AuthorRole role
Set<StayListing> stayListings
boolean canAcceptAuthors()
1
contains
0..*
StayListing
AuthorServiceImpl
AuthorRepository authorRepository
AuthorMapper authorMapper
PasswordEncoder passwordEncoder
JwtUtil jwtUtil
List<AuthorDTO> findAll()
Optional<AuthorDTO> findById(Long)
AuthorDTO update(Long, AuthorDTO)
Optional<AuthorDTO> findByEmail(String)
AuthorDTO save(AuthorDTO)
Optional<AuthorDTO> login(AuthorDTO)
void deleteById(Long)
uses
uses
implements
uses
uses
AuthorRepository
Optional<Author> findById(Long)
List<Author> findAll()
Author save(Author)
void deleteById(Long)
Optional<Author> findByEmail(String)
AuthorMapper
AuthorDTO toDto(Author)
Author toEntity(AuthorDTO)
AuthorService
List<AuthorDTO> findAll()
Optional<AuthorDTO> findById(Long)
AuthorDTO update(Long, AuthorDTO)
Optional<AuthorDTO> findByEmail(String)
AuthorDTO save(AuthorDTO)
Optional<AuthorDTO> login(AuthorDTO)
void deleteById(Long)
PasswordEncoder
JwtUti

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

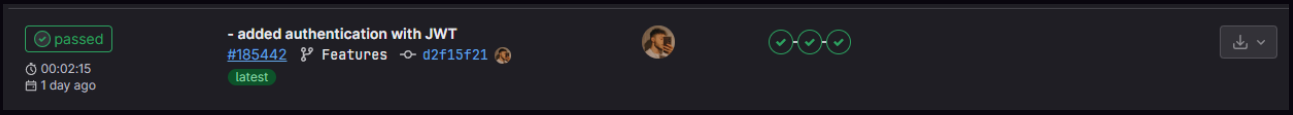
- Utilizes Gradle to compile and assemble the project components.

Test Phase

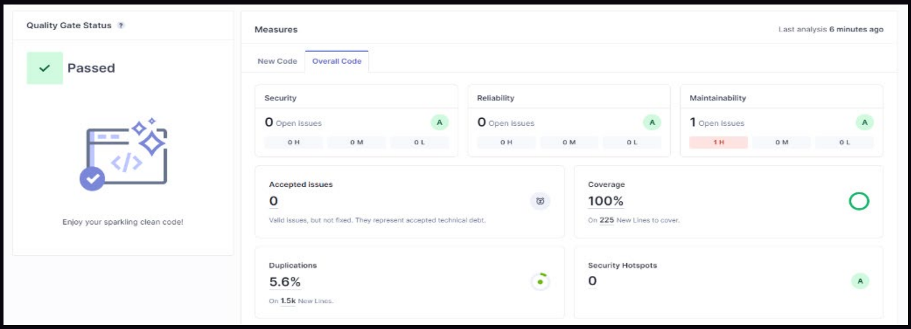
- Employs Gradle again, this time to execute the project's tests, ensuring functionality and reliability.

Sonar Check Phase

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



passed
00:02:15
1 day ago
- added authentication with JWT
#185442 Features d2f15f21
latest



Quality Gate Status ?

Passed

Enjoy your sparkling clean code!

Measures

Last analysis **6 minutes ago**

New Code | Overall Code

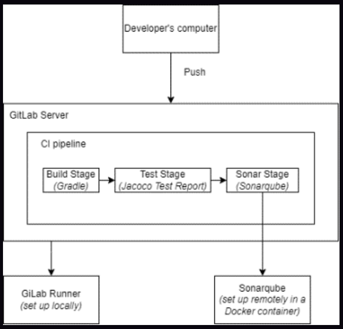
Security	Reliability	Maintainability
0 Open issues A	0 Open issues A	1 Open issues A
0 H 0 M 0 L	0 H 0 M 0 L	1 H 0 M 0 L

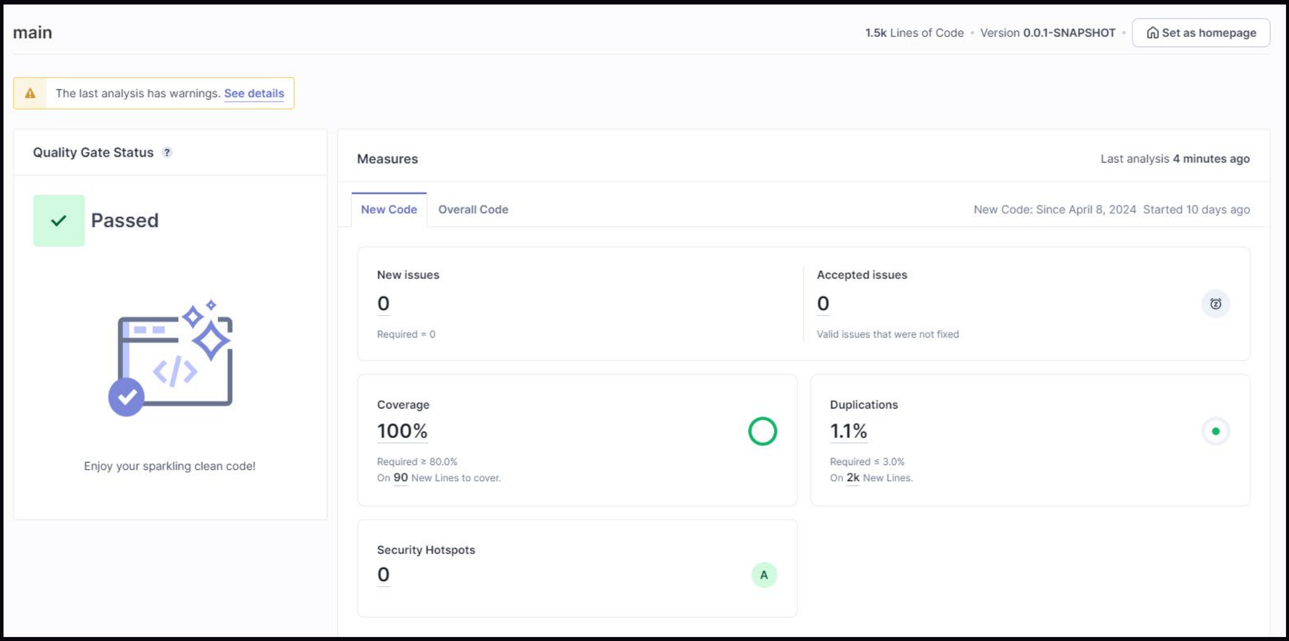
Accepted issues
0
Valid issues, but not fixed. They represent accepted technical debt.

Coverage
100%
On **225** New Lines to cover.

Duplications
5.6%
On **1.5k** New Lines.

Security Hotspots
0
A





main

1.5k Lines of Code • Version **0.0.1-SNAPSHOT** • Set as homepage

The last analysis has warnings. See details

Quality Gate Status ?

Passed

Enjoy your sparkling clean code!

Measures

Last analysis **4 minutes ago**

New Code | Overall Code

New Code: Since April 8, 2024 Started 10 days ago

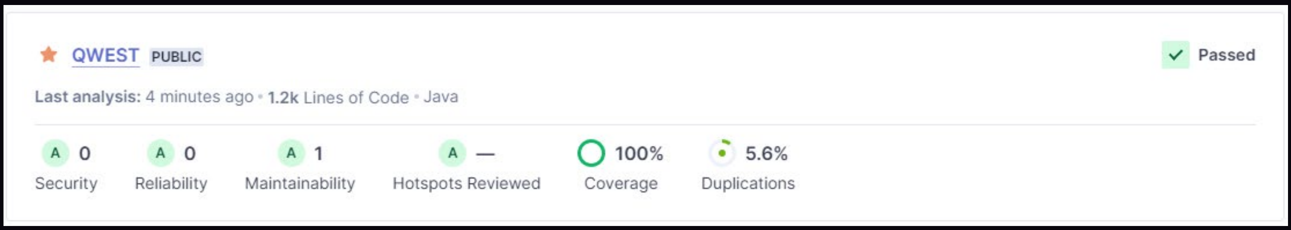
New issues
0
Required = 0

Accepted issues
0
Valid issues that were not fixed

Coverage
100%
Required ≥ 80.0%
On **90** New Lines to cover.

Duplications
1.1%
Required ≤ 3.0%
On **2k** New Lines.

Security Hotspots
0
A



QWEST PUBLIC

Passed

Last analysis: 4 minutes ago • **1.2k** Lines of Code • Java

A 0	A 0	A 1	A —	100%	5.6%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

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.