**DESIGN DOCUMENT**

**Ticketpass**

**Date:** 10/11/2023

**Version:** 3.0

**Made by:** Hristo Ganchev

**Table of Contents**

[1. Introduction 3](#_Toc149995011)

[2. Architecture Constraints and Design Decisions 3](#_Toc149995012)

[3. C4 Model Diagrams 4](#_Toc149995013)

[**Level 1: System Context Diagram** 4](#_Toc149995014)

[**Level 2: Container Diagram** 5](#_Toc149995015)

[**Level 3: Component Diagram (Backend)** 6](#_Toc149995016)

[**Level 3: Component Diagram (Frontend)** 7](#_Toc149995017)

[**Level 4: Class Diagram (Backend)** 8](#_Toc149995018)

[4. Explanations and Rationale 10](#_Toc149995019)

[5. Sprints Progress 10](#_Toc149995020)

[**Sprint 1:** 10](#_Toc149995021)

[**Sprint 2:** 11](#_Toc149995022)

# **Introduction**

The purpose of this document is to describe the constraints and decisions that the platform is going to be developed under and the flow of the implemented functions by presenting a C4 Model Diagram.

Ticketpass is a platform that provides the user with a wide range of concerts. They can select a concert they would like to attend, purchase tickets for the selected event and gain access to it at the entrance of the venue.

# **Architecture Constraints and Design Decisions**

Based on customer requirements, Spring Boot will be used for the Backend, React for the frontend, and MySQL for Data Management.

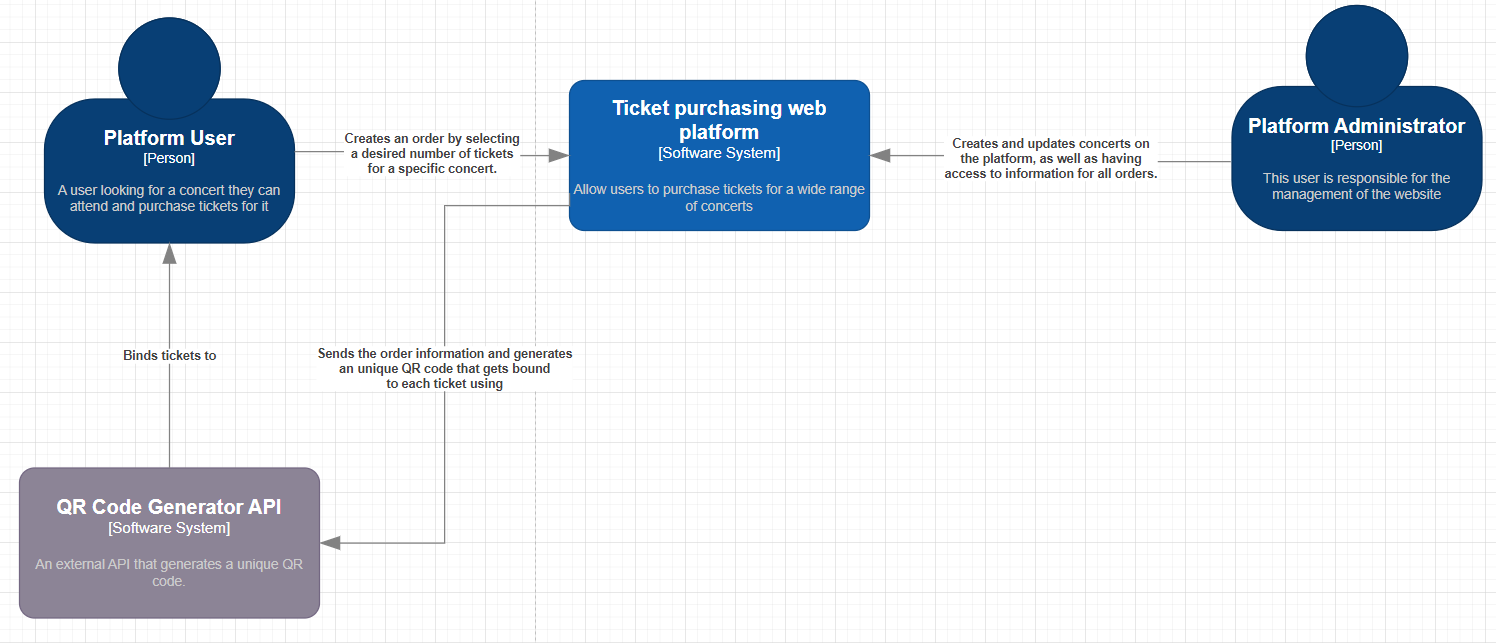
Spring Boot is a framework that provides the developer with a quick compilation of their product and thus the development of the application is conducted more efficiently. What is more, Spring Boot has a large and active community, as well as a wide ecosystem, which will significantly reduce development time. By using this framework, the Dependency Inversion Principle will be followed, promoting loose coupling and avoids the need of continuously and repetitively re-declaring services/repositories throughout the application. The Single-Responsibility Principle and the YAGNI (You Aren’t Gonna Need It) Principle will also be followed.

React uses a virtual DOM and updates only the necessary elements to the UI (User Interface). This clearly follows the KISS (Keep It Simple, Stupid) Principle, making the website application easier to compile and maintain, and promotes a better user experience. Moreover, the framework is perfect for the DRY (Don’t Repeat Yourself) principle, since a component is declared only once and then can be reused in the entire application, without the need of copy-pasting code. The Single-Responsibility Principle will also be followed.

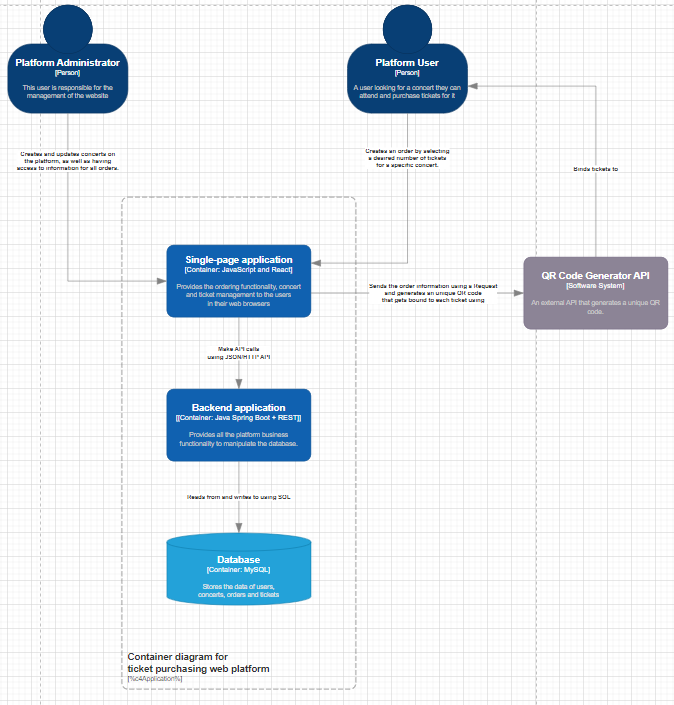
MySQL will be used due to the wide ecosystem of tools and libraries. This will make the database much more adaptable to the development environment. MySQL also has a track record for reliability and performance.

# **C4 Model Diagrams**

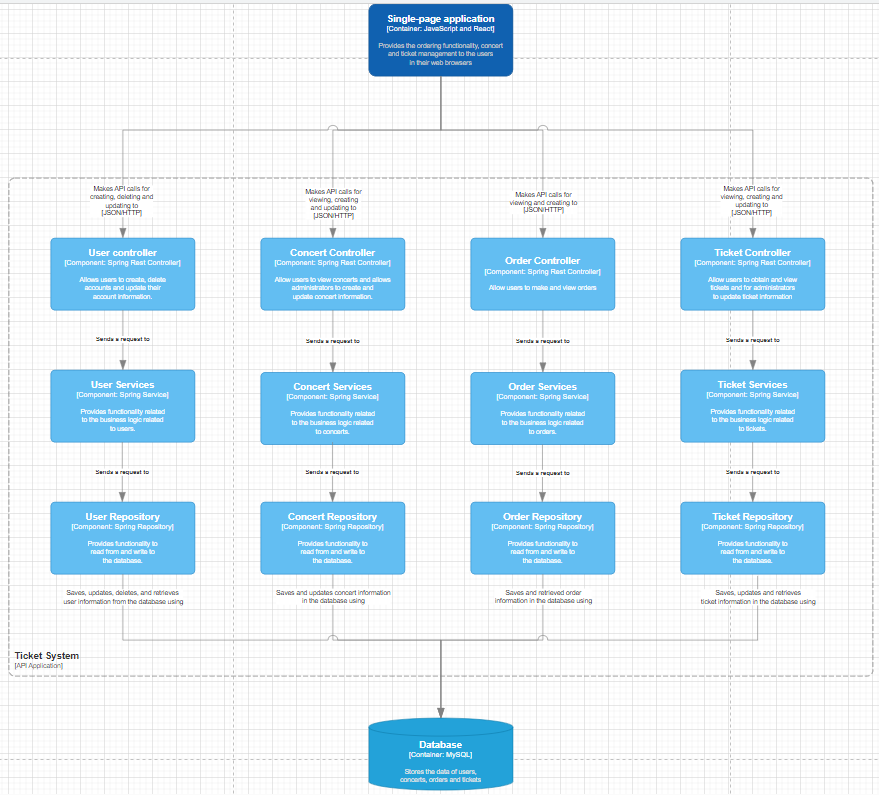
## **Level 1: System Context Diagram**

****

## **Level 2: Container Diagram**



## **Level 3: Component Diagram (Backend)**



## **Level 3: Component Diagram (Frontend)**

**Картина, която съдържа текст, екранна снимка, диаграма, Паралелен

Описанието е генерирано автоматично**

## **Level 4: Class Diagram (Backend)**

**Картина, която съдържа текст, екранна снимка, номер, Шрифт

Описанието е генерирано автоматично**

**Картина, която съдържа текст, екранна снимка, номер, диаграма

Описанието е генерирано автоматично**

# **Explanations and Rationale**

The administrator is responsible for managing the platform. They can add and update concerts from the MySQL Database, can review orders, and overall manage the flow of the platform.

The user is a ticketholder. They select a concert, purchase the desired number of tickets, which get bound to their account.

The project is divided into three parts: a single-page application, a backend application, and a database.

The single-page application is updated accordingly, and only necessary UI elements are displayed. This makes the platform much easier to compile and provides the user with a fast and easy experience.

The backend application is implemented using Java Spring Boot. It is responsible for all the logic behind the application. It is separated into 4 different categories (User, Ticket, Order, Concert). Each category consists of a REST Controller that is responsible for managing the HTTP requests, a Service layer where all of the logic is managed, and a Repository that assists with the retrieval and updating of information to/from the database.

The database is where all of the information is stored. Measures will be taken in order to keep sensitive information (e.g. passwords) hidden in case of data leaks.

# **Sprints Progress**

## **Sprint 1:**

1. As an administrator, I want to add concerts to the database with the necessary information so that they can be displayed to the users of the platform.

* A database that stores concerts should be created.
* Each concert should have a unique ID, Artist Name, Music Genre, Venue, City, Date, Description, Photo URL, Ticket Price and Tickets Remaining.
* A method that adds concerts to the database should be implemented

## **Sprint 2:**

1. As a user, I want to see all concerts in one place, so that I have a wide selection of different concerts of different genres available in a glance.

* A page where concerts are retrieved from a database should be created
* Each concert should be displayed with their name, photo, and description in the “Concerts” page