Shop system technical specification

Strzechowski Konrad

June 2024

version 1.0

Contents

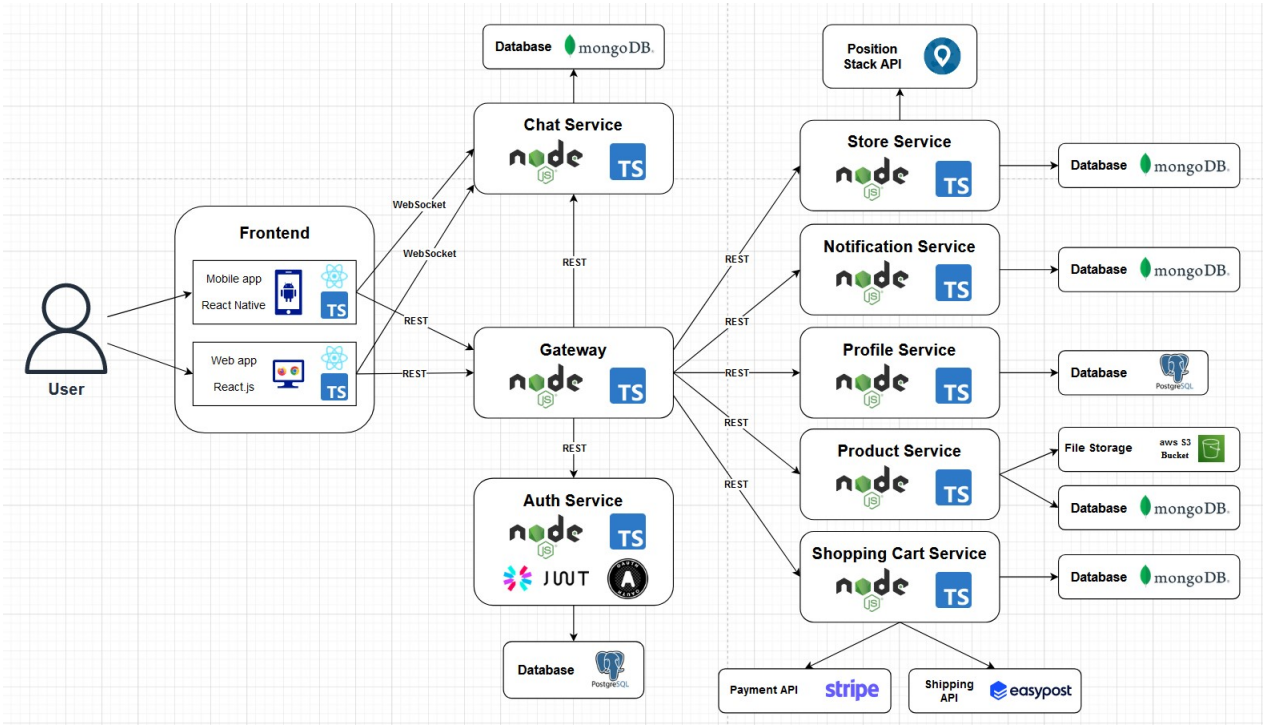
1 Abstract

This document contains the technical specification of a large-scale enterprise store system. The specification consists of a detailed discussion of the system architecture, highlighting its layers, nodes, modules, and main components. Each system module is thoroughly presented, including a description of its tasks and the technological requirements for implementation. Besides, the document presents the dependencies occurring between the modules and precisely specifies the ways and types of communication occurring in the system. Additionally, specification contains details about external interfaces. Finally, this document provides an approximation of the planned appearance of the application through detailed mockups of both the mobile and web versions. These mockups offer a visual representation of the user interface and user experience, illustrating the design, layout, and functionality that users can expect.

2 System architecture

The system consists of a backend built with a microservices architecture and two client applications: web and mobile. The backend is developed using the Node.js runtime and TypeScript language, but thanks to the microservices architecture, it is possible to use different languages and runtimes for future development as well. The web application is created using the React library, while the mobile app is implemented with React Native. This approach leverages the similarities between React and React Native, simplifying the development process and ensuring a consistent user experience across both platforms. All communication between services, except for chat, is implemented using standard REST APIs. Chats utilize WebSockets for real-time communication.

Figure 1: Overview diagram of the Store system architecture



2.1 Gateway

The Gateway is a single point of access for all clients, ensuring proper communication with each service and concealing implementation specifics from the frontend. It acts as a bridge between services, performing the necessary operations to efficiently fulfill requests. The Gateway API can easily combine data from different services to create the appropriate result needed by the user. It also provides security and performs the necessary authorization through close communication with the Auth service. It should also be noted that the gateway has no direct connection to any database and can be easily duplicated and used with a load balancer under heavy load.

2.2 Auth Service

The Auth API ensures a secure authentication and authorization process by utilizing OAuth 2.0 and JWT tokens. This enables an easy and secure connection between clients and other backend services. The Auth service is fully responsible for issuing, validating, and renewing tokens. It also connects to PostgreSQL datbase, which is essential for the proper management of user credentials.

2.3 Store Service

The Store service manages local stores and their locations by communicating with the external API, Position Stack. Position Stack provides a fast and straightforward geocoding solution, allowing us to map each address to precise coordinates, which enables users to view store locations on a map. Given the variability in location data, the Store service uses a non-relational MongoDB database. This choice allows for flexible data

management, accommodating changes and documents with different fields within a single collection.

2.4 Notification Service

The Notification Service is responsible for sending notifications to users. Its primary function is to distribute new marketing ads to email addresses. Additionally, it sends receipts or invoices after each purchase. The service also integrates closely with the mobile app through the gateway to deliver push notifications in real-time to users who have installed the app. To accommodate the evolving requirements, we have again chosen a MongoDB database for its flexibility and scalability.

2.5 Profile Service

The Profile service manages all user details, including data for customers, employees, and administrators. The Profile API utilizes a PostgreSQL database, as the clearly specified user account requirements make a relational database the best fit for ensuring data integrity and efficient querying.

2.6 Product Service

The Product service is currently the largest and most widely used API, responsible for managing all product data down to the smallest details. Due to the variability in product information, it utilizes a MongoDB database. The service employs text indexes and standard compound indexes to offer fast and accurate search and sorting capabilities, ensuring an efficient and responsive user experience.

2.7 Shopping Cart Service

The Shopping Cart Service is responsible for processing both online and local purchases. It works closely with the external Payment API, Stripe, and the Shipping API, EasyPost, to provide customers with various purchasing and shipping methods. Due to the differing details of each purchase, the Shopping Cart Service uses a MongoDB database to handle it efficiently.

2.8 Chat Service

The Chat service is currently final planned API. It provides users with ability to start a chat with automated bots, but also real human assistants. It also saves the entire chat history, allowing for data analysis to further optimize app and user experience.