Solution Architecture Document

1. Project Description

RecipeBook is a web application for storing and managing recipes.

Its purpose is to provide users with a centralized platform to view, manage, and share cooking recipes in a user-friendly environment.

Scope: The main capabilities of the system are:

- to display all available recipes and their details to users;
- to search for recipes by title;
- to add, edit, and delete recipes for authenticated users.

Key features include:

- viewing all recipes stored in the database;
- searching for recipes by title;
- viewing detailed information about a selected recipe;
- user authentication via registration and login;
- adding, editing, and deleting personal recipes (for authenticated users only).

The **target users** are people who enjoy cooking and want a convenient way to store, manage, and discover recipes. The platform can be used by both unauthorized and authorized users, but only registered users can create, edit, or delete their own recipes.

2. Project Architecture

Architecture style. The project follows a client-server architecture built with a monolithic approach, that means the entire application — frontend, backend logic, and data access — is developed as a single unified application.

The project consists of:

- frontend:
  - *programming language:* TypeScript;
  - *libraries:* React, MUI;
- backend:
  - *programming language:* TypeScript;

    - *platform:* Node.js;
    - *framework:* NestJS;
- database: PostgreSQL;
- API communication: RESTful APIs;
- documentation tools: Swagger, TypeDoc, Storybook.

The **interaction** between the main components is presented in fig. 2.1. It includes:

- frontend (written using React), that runs on port 5173, and interacts with the backend via HTTP requests;
- backend (written using NestJS), that runs on port 5000, processes requests, interacts with the database, and returns JSON responses;
- database (PostgreSQL DBMS), that runs on port 5432, stores and returns structured data.

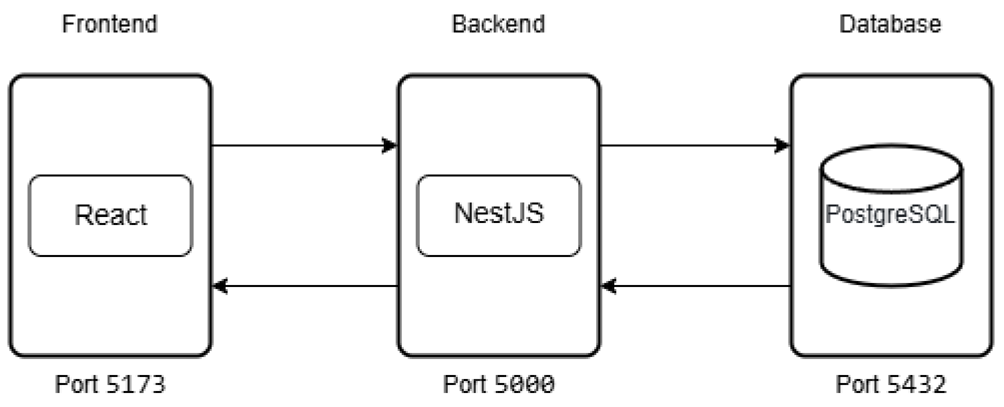

Figure 2.1 – Component Interaction Diagram

The frontend interacts with the backend and transfers data to it using the REST API.

The database consists of four main tables:

- user – stores information about registered users, including their authentication details (username, email, and password hash) and admin status;
- product – contains ingredients or food items that can later be used in recipes;
- recipe – represents cooking recipes, including their titles, descriptions and user who created it;

- recipe_ingredient – a join table that links recipes to products, indicating which ingredients are used in which recipes and in what quantities.

The database structure, its tables, and the relationships between them are presented in more detail in fig. 2.2.

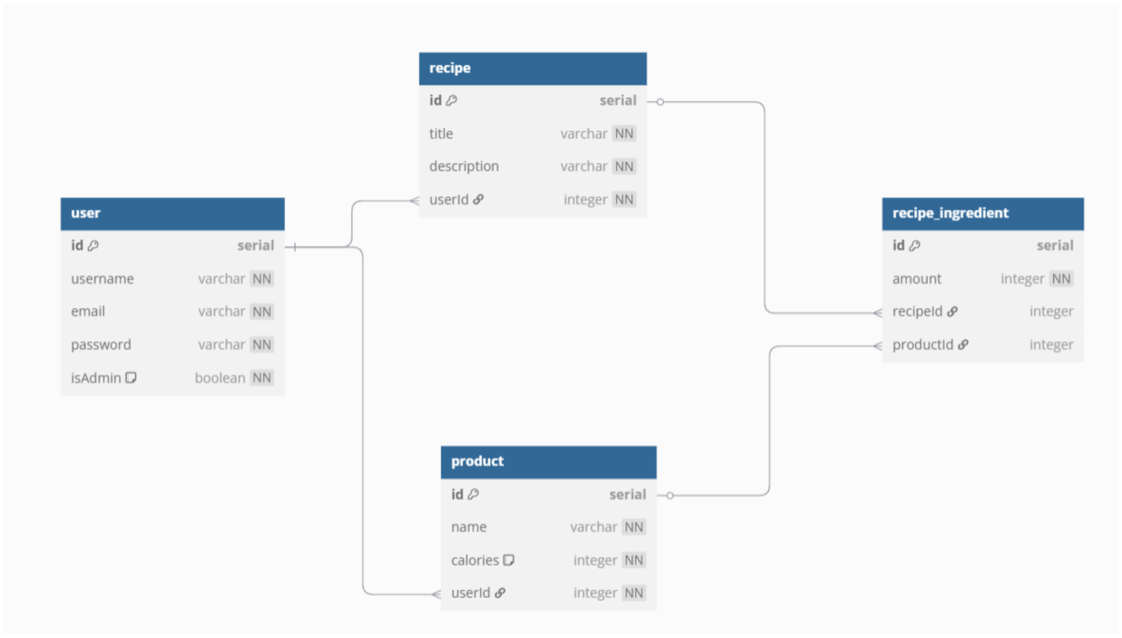

Figure 2.2 – Database Diagram

3. Infrastructure Requirements

Currently, the application runs only on localhost. There is no cloud deployment or public production environment set up.

To run, develop and maintain the application, the following environment setup is required:

Versions:

- Node.js: v18+;
- NestJS: v10+;
- PostgreSQL: v17+;
- npm: v10+;
- React: v18+;
- TypeScript: v5+.

General tools:

- Git;

- Visual Studio Code.

Environment configuration:

- A .env file is used to manage environment-specific settings, including:
  - the backend server port;
  - PostgreSQL database name;
  - PostgreSQL username and password.

CI/CD: No CI/CD pipeline is configured at this stage.

Monitoring & Logging: No external monitoring or logging tools are used. Errors and logs are currently handled via built-in logging and error handling mechanisms provided by NestJS during development.

Security: The following security measures have been added to the project:

- JWT-based user authentication;
- password encryption using bcrypt (with a salt rounds count of 10) for user authentication.