Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторным работам

Дисциплина: «Технологии сетевого программирования»

Тема «Проектирование приложения»

Выполнили: Шашков Н.А. Поверинов А.Д

Группа: 6304-010302D

Самара, 2025

Содержание

[Введение 3](#_Toc199681468)

[1 Проектирование приложения 4](#_Toc199681469)

[1.1 Схема взаимодействия компонентов 4](#_Toc199681470)

[1.1.1 Возможности пользователя 4](#_Toc199681471)

[1.1.2 Возможности администратора 4](#_Toc199681472)

[1.2 Логическая схема БД 4](#_Toc199681473)

[2 Описание API 5](#_Toc199681474)

[3 Тестирование API 6](#_Toc199681475)

[4 Аутентификация и авторизация 7](#_Toc199681476)

[4.1 Аутентифиакция 7](#_Toc199681477)

[4.2 Безопасность 7](#_Toc199681478)

[5 Структура пользовательского интерфейса 8](#_Toc199681479)

[6 Демонстрация работы приложения 9](#_Toc199681480)

[6.1 Взаимодействие с приложением 9](#_Toc199681481)

[Приложение А 11](#_Toc199681482)

Введение

Наше приложение помогает пользователям контролировать правильное питание, подсчитывать калории и макронутриенты (белки, жиры, углеводы). Оно разработано с использованием технологий Java 21, Spring Boot и PostgreSQL, а также использует Thymeleaf для серверного рендеринга. Вся система должна быть контейнеризирована с помощью Docker (не успели дореализовать).

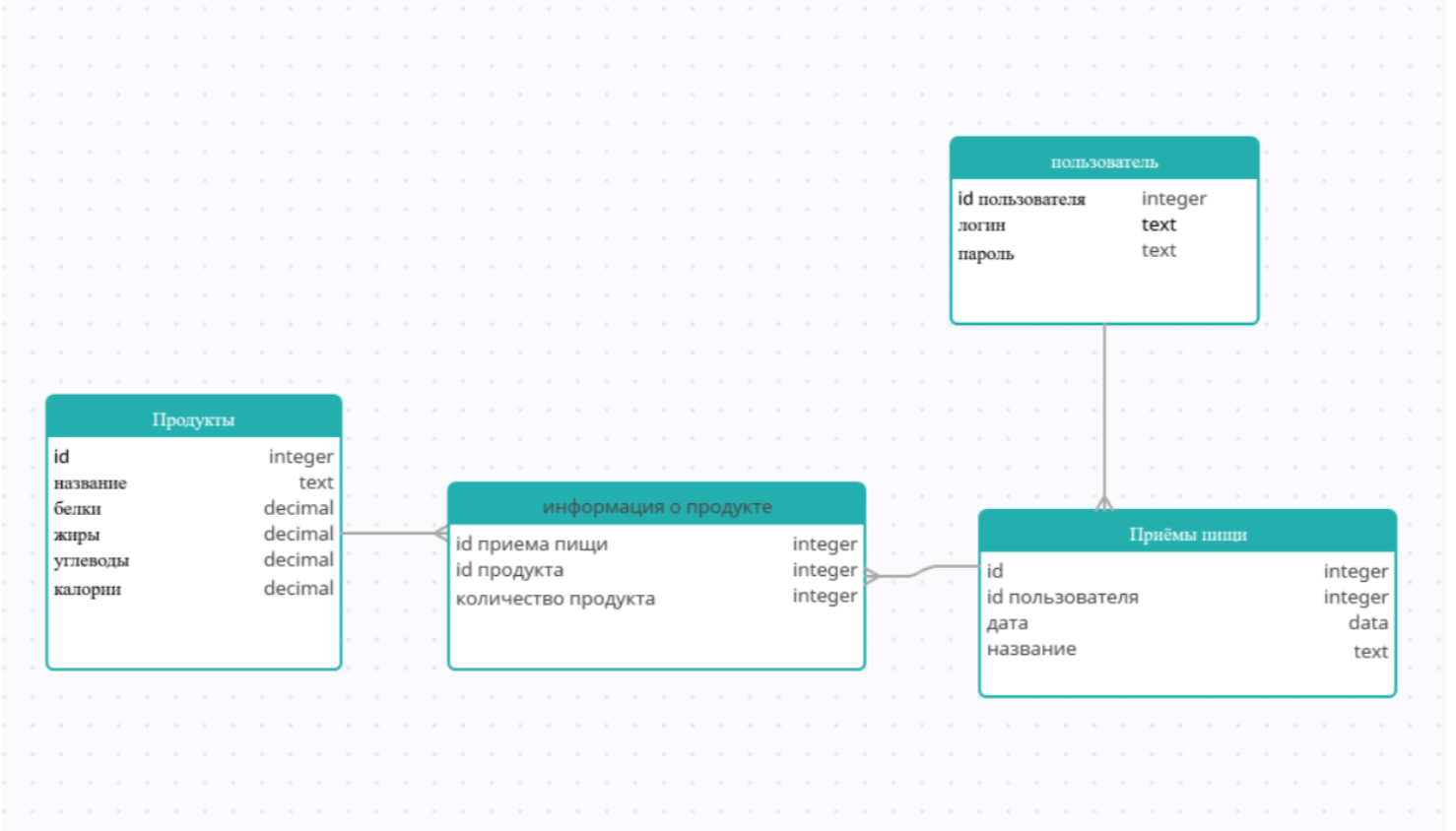
Проект создавался в учебных целях для демонстрации навыков в серверной разработке, проектировании архитектуры приложений, настройке безопасной аутентификации, работе с базами данных и интеграции компонентов в контейнеризированную среду.

1. Проектирование приложения
   1. Схема взаимодействия компонентов

Клиент отправляет запросы к API (например, вход в систему, добавление продуктов).

База данных (PostgreSQL) хранит список продуктов, пользователей и их пароли, приемы пищи и обеспечивает связь между ними.

* + 1. Возможности пользователя
* Добавление продуктов в прием пищи;
* Просмотр всех приемов пищи;
* Просмотр БЖУ приема.
  + 1. Возможности администратора
* Просмотр всех пользователей и приемов;
* Удаление любых приемов и пользователей.
  1. Логическая схема БД



1. Описание API

В таблице 1 представлены основные эндпоинты API.

Таблица 1 – Описание API

|  |  |  |
| --- | --- | --- |
| Метод | URL | Описание |
| POST | /users/login | Аутентификация пользователя |
| POST | /users/register | Регистрация пользователя |
| POST | /users | Создание пользователя |
| GET | /users/getMeals | Получение списка всех приёмов пищи пользователя |
| GET | /users/{id} | Получить пользователя по id |
| GET | /users/getAll | Вывести всех пользователей |
| GET | /products | Вывести все продукиы |
| POST | /products/selectProduct | Добавить продукт в приём пищи |
| POST | /products/deselectProduct | Удалить продукт из приёма пищи |
| POST | /products/saveMeal | Сохранение текущего приёма пищи |
| POST | /meals/{id} | Получение приёма пищи по id |

1. Тестирование API

Для проверки правильности работы серверной части веб-приложения было проведено ручное тестирование API с использованием Postman. Хотя клиентский интерфейс реализован с помощью серверного рендеринга на Thymeleaf, каждый ключевой функционал также был дополнительно протестирован через соответствующие HTTP-запросы. Процесс тестирования включал следующие аспекты:

* Проверка регистрации и авторизации пользователей;
* Добавление и удаление продуктов в прием пищи;
* Получение списков пользователей и приемов их пищи;
* Обработка ошибок (например, неправильный формат данных, отсутствие прав доступа и др.).

На каждом этапе проверялись:

* Корректность ответа (HTTP-статус, тело ответа, сообщения об ошибках);
* Защита маршрутов с использованием JWT-токенов.

1. Аутентификация и авторизация

Веб-приложение использует механизм JWT для управления доступом пользователей. Аутентификация и авторизация реализованы на стороне backend-приложения с использованием Spring Security.

* 1. Аутентифиакция

При попытке входа в систему пользователь отправляет POST-запрос на адрес /users/login, в котором указывает свои учетные данные (логин и пароль). Если аутентификация проходит успешно, сервер создает JWT-токен и возвращает его в теле ответа. Этот токен представляет собой закодированную строку, содержащую информацию о пользователе, такую как его ID, роль и срок действия токена. Он необходим для доступа к защищенным ресурсам.

При последующих запросах клиент должен добавлять токен в заголовок Authorization в формате: Bearer <токен>. Без действительного токена доступ к защищенным маршрутам будет закрыт.

* 1. Безопасность

Внедрены следующие меры безопасности:

• Проверка уникальности логина при регистрации;

• Хэширование паролей

1. Структура пользовательского интерфейса

Пользовательский интерфейс веб-приложения реализован с использованием Thymeleaf. Интерфейс построен по классической MVC-модели и рендерится на стороне сервера, что обеспечивает простоту интеграции с Spring Boot и улучшает контроль над безопасностью.

Приложение разделено на логически обособленные пользовательские интерфейсы. Основные страницы и компоненты:

* Регистрация и авторизация – отдельные формы для клиентов с соответствующей валидацией и отображением ошибок.
* Основная страница – здесь пользователь добавляет и удаляет продукты из приема пищи, смотрит БЖУ и подписывает прием пищи.
* Страница ошибки – если неправильно введен пароль или логин при входе.
* Страница всех приемов пищи – список всех приемов пищи, который клиент когда-либо добавлял.

Переходы между страницами осуществляются с помощью стандартных маршрутов. Этот метод обеспечивает простой и эффективный пользовательский опыт, снижая зависимость от сторонних библиотек и упрощая процесс.

1. Демонстрация работы приложения
   1. Взаимодействие с приложением

Пользователь может перейти по адресу http://localhost:8080, где доступен web-интерфейс приложения, реализованный с использованием Thymeleaf.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Страница входа.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Страница регистрации.

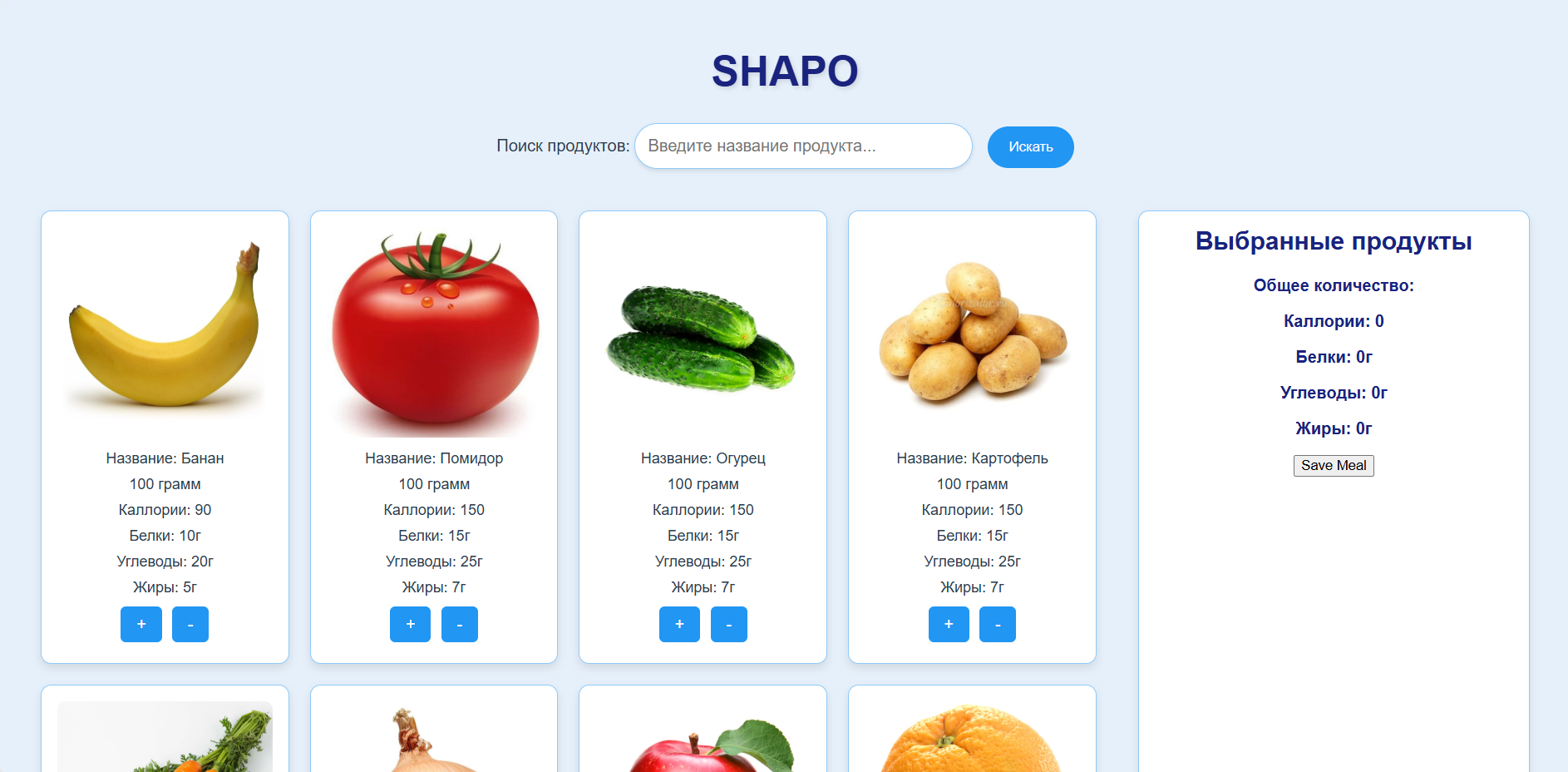


Рисунок 3 – Основная страница.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Страница всех приемов пищи пользователя.

Приложение А

Ссылка на GitHub со всем проектом:

[https://github.com/](https://github.com/afatin/social-network)[[PoverinovAlex/demo](https://github.com/afatin/social-network)](https://github.com/PoverinovAlex/demo)