МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

WEB-приложение «Счетчик калорий»

Выполнил студент Костюкова Анна Олеговна

(Ф.И.О.)

Руководитель проекта ассистент Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассистент Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассистент Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

Реферат

Содержание

Введение

Правильное питание является важнейшим фактором, обеспечивающим здоровье человека, его работоспособность и устойчивость к внешним неблагоприятным воздействиям. В настоящее время существует ряд проблем, с которыми сталкивается большая часть общества: малоподвижный образ жизни, употребление высококалорийной пищи, повышенное потребление сахара и соли, замена чистой питьевой воды газированными напитками. Для поддержания здоровья организма с помощью правильного питания необходимо, чтобы энергетическая ценность рациона соответствовала суточным энергетическим затратам организма и рацион был сбалансирован по белкам, жирам, углеводам.

Целью курсовой работы является разработка приложения «Счетчик калорий». Данное приложение позволит рассчитать то количество энергии, которое необходимо ежедневно получать организму в зависимости от вашего роста, веса, возраста и степени физической активности.

Приложение должно быть разработано с помощью языка JavaScript и программной платформы Node.js реализуя клиент-серверную архитектуру. В качестве СУБД для базы данных была выбрана PostgreSQL. Для корректного выполнения операций над данными, с которыми работает приложение, необходимо чётко и правильно составить реляционную структуру.

Основными задачами курсовой работы являются:

* Определить необходимости сущности;
* Создать таблицы и установить связи между ними;
* Разработать приложение с использованием многоуровневой архитектуры;
* Провести тестирование.

1 Постановка задачи

Основной задачей курсового проекта является разработка web-приложения «Счетчик калорий». Идея приложения заключается в том, что пользователь может добавлять продукты, съеденные за день, для подсчета употребленных калорий и рассчитать свою дневную норму калорий по своим параметрам.

Задачи курсового проекта:

* Определить необходимости сущности;
* Создать таблицы и установить связи между ними;
* Разработать приложение с использованием многоуровневой архитектуры;
* Провести тестирование.

Функционально должны быть выполнены следующие задачи:

* Возможность регистрации и авторизации;
* Возможность добавлять продукты в базу данных;
* Возможность создавать дни для занесения приемов пищи;
* Возможность добавлять употребленные продукты в прием пищи;
* Возможность рассчитать дневную норму калорий по своим параметрам;
* Возможность отправлять уведомления на почту;
* Возможность просматривать статистику.

1.1 Обзор прототипов

FatSecret – это кроссплатформенный счётчик калорий. После стандартной процедуры регистрации и проверки e-mail пользователь попадает в подготовительный раздел сервиса под названием «Быстрый старт» (рисунок 1.1). Здесь необходимо ввести данные: рост и вес, уровень повседневной активности, цель – набор веса, его сохранение или похудение, а также указать желаемый вес в килограммах.

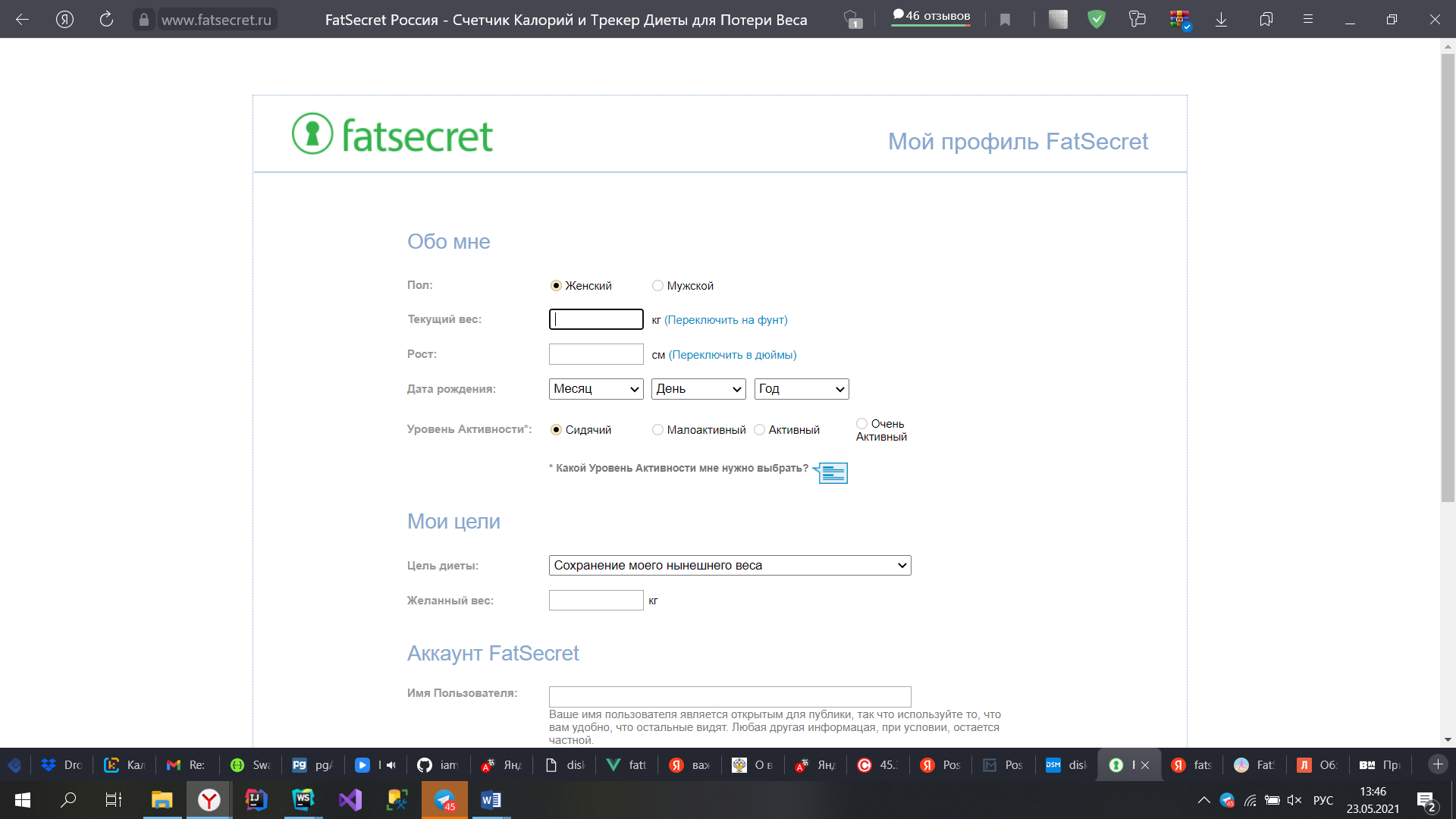


Рисунок 1.1 – Ввод необходимой информации FatSecret

Пройдя подготовительный этап, пользователь попадает на главную страницу с лентой новостей. Наверху располагается меню с пятью вкладками (рисунок 1.2):

* Мой FatSecret – здесь вы ведёте свои дневники и отслеживаете результаты;
* Продукты – здесь можно посмотреть состав любого продукта в базе сервиса или добавить свой продукт;
* Рецепты – здесь вы можете поделиться с сообществом своими рецептами или посмотреть уже существующие;
* Фитнес – здесь можно посмотреть энергоёмкость упражнений и другой активной деятельности с учётом вашего собственного веса;
* Общество – здесь публикуются записи из дневников и результаты других пользователей.

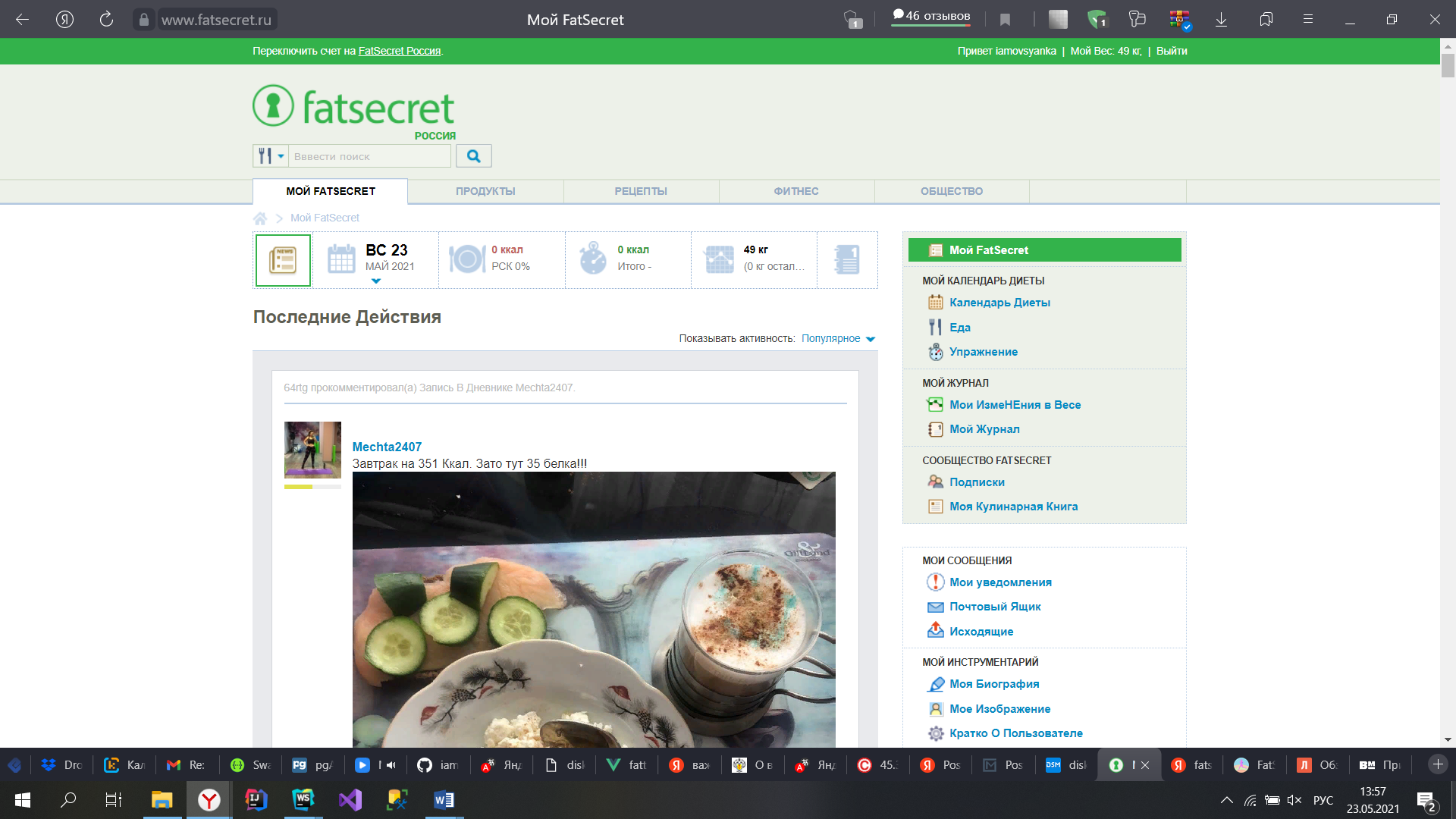


Рисунок 1.2 – Меню на главной странице FatSecret

Достоинства:

* быстрая обработка данных;
* возможность запоминать часто употребляемые продукты.

Недостатки:

* можно добавлять продукты только тех брендов, которые присутствуют в базе данных, что не дает возможность вносить иные продукты;
* невозможно удалить аккаунт;
* часть функционала доступна только за деньги.

Дневник питания – web-приложение для подсчета калорий. Для расчета своей нормы калорий на день необходимо ввести данные (рисунок 1.3).

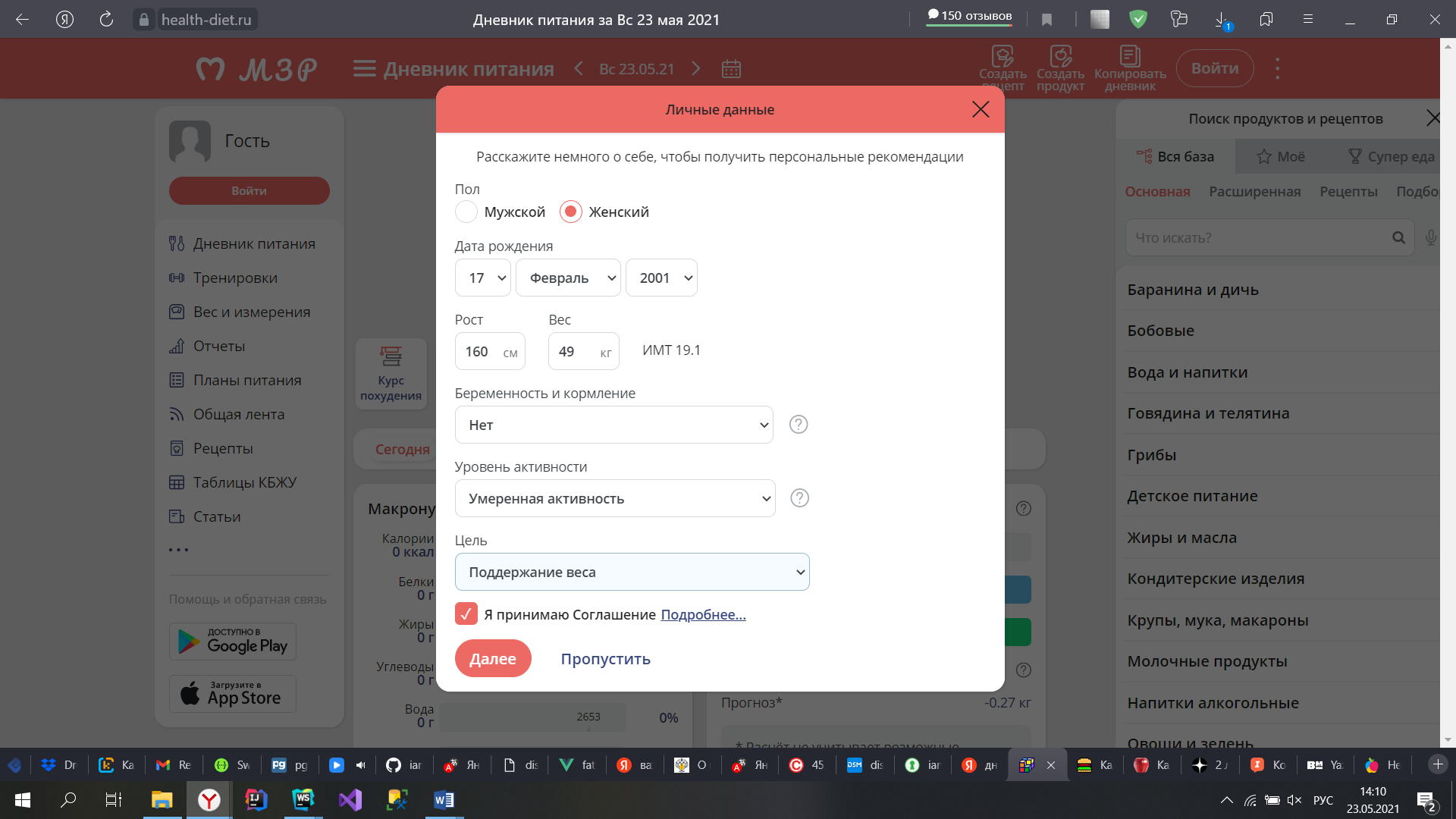


Рисунок 1.3 – Ввод необходимой информации «Дневник питания»

После регистрации пользователь может вносить продукты в свой прием пищи (рисунок 1.4).

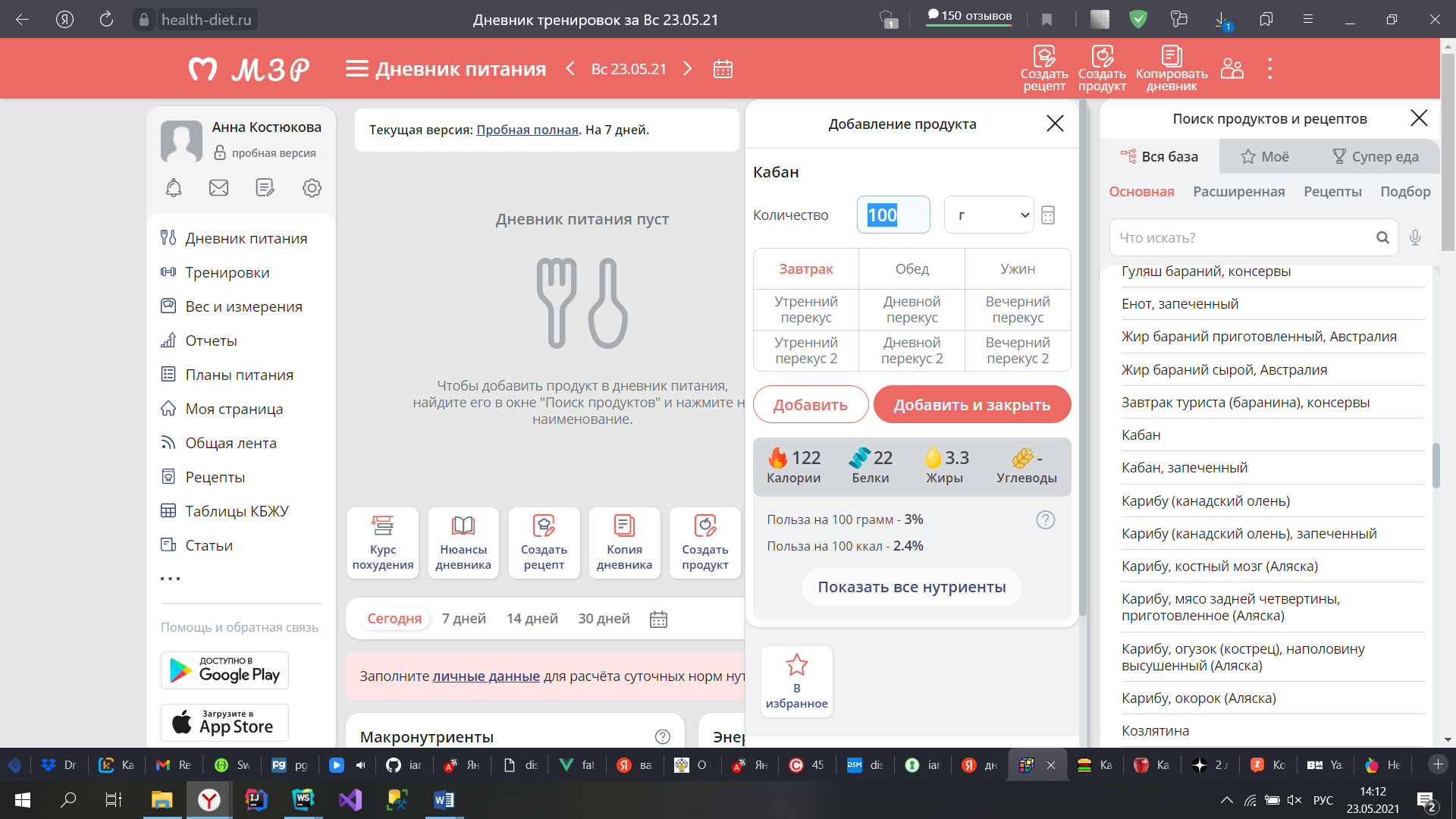


Рисунок 1.3 – Добавление продукта в прием пищи «Дневник питания»

Достоинства:

* возможность общаться на форуме;
* возможность создавать отчеты за определенный период;
* возможность создавать планы питания;
* возможность читать статьи о здоровье.

Недостатки:

* большая часть функционала доступна только за деньги.

2 Проектирование приложения

Общая структура приложения представлена на рисунке 2.1.

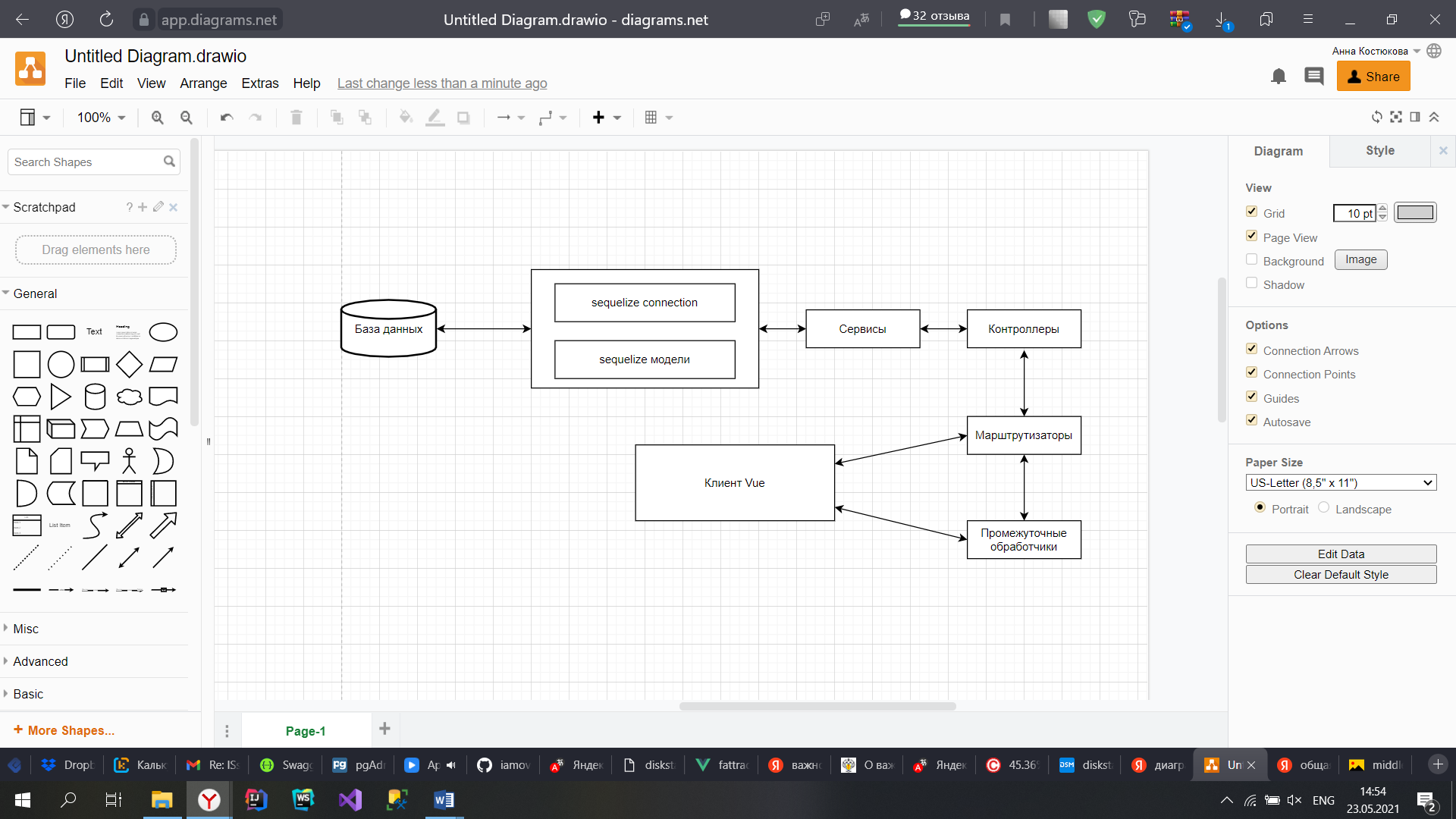


Рисунок 2.1 – Общая структура приложения

В приложении используется паттерн MVC, который является одним из распространенных паттернов, применяемых в веб-приложениях. В том числе он применяется и в приложениях на Node.js.

Паттерн MVC включает ряд компонентов:

* модели определяют структуру и логику используемых данных;
* представления определяет визуальную часть, как данные будут отображаться;
* контроллеры обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляет в ответ клиенту некоторый результат обработки.
* система маршрутизация как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер [1].

Проект разбит на отдельные логические модули. Клиентская часть приложения взаимодействует с маршрутизаторами и промежуточными обработчиками, которые проверяют авторизирован ли пользователь. Контроллер проверяет данные от пользователя и передают запрос определенному сервису. Сервисы посредством sequelize моделей взаимодействуют с базой данной. Данная архитектура выбрана из-за ее гибкости, расширяемости и легкой взаимозаменяемости компонентов.

2.1 Проектирование структуры базы данных

Взаимодействие серверной части приложения с реляционной базой данных осуществляется с помощью ORM библиотеки Sequelize [2]. Благодаря данному подходу осуществлено моделирование данных приложения на основе объектных схем, включая приведение типов, проверку, построение запросов и многое другое.

Для реализации курсового проекта было принято решение о создании базы данных состоящей из 5 таблиц. Диаграмма базы данных представлена в приложении А.

Таблица Users содержит информацию о пользователях и содержит 15 столбцов:

* Столбец id является первичным ключом, тип данных integer, инкрементируется при добавлении пользователя;
* Столбец email хранит почту пользователя, тип данных character varying, уникальный, ненулевой;
* Столбец name хранит имя пользователя, тип данных character varying, уникальный, ненулевой;
* Столбец password хранит пароль в зашифрованном виде, тип данных character varying, ненулевой;
* Столбец verified хранит значение истина, если пользователь подтвердил свою почту, и ложь, если нет, ненулевой;
* Столбец height хранит рост пользователя, тип данных integer;
* Столбец weight хранит вес пользователя, тип данных integer;
* Столбец role хранит роль пользователя, тип данных character varying, ненулевой;
* Столбец birthDay хранит дату рождения пользователя, тип данных date;
* Столбец activity хранит коэффициент активности пользователя, тип данных double precision;
* Столбец requiredCalories хранит дневную норму калорий пользователя, тип данных double precision;
* Столбец createdAt хранит временную метку создания пользователя;
* Столбец updatedAt хранит временную метку обновления пользователя;
* Столбец deleteAt хранит временную метку удаления пользователя.

Таблица Categories содержит информацию о категориях продуктов и содержит 5 столбцов:

* Столбец id является первичным ключом, тип данных integer, инкрементируется при добавлении категории;
* Столбец name хранит название категории, тип данных character varying, уникальный, ненулевой;
* Столбец description хранит описание категории, тип данных character varying, ненулевой;
* Столбец createdAt хранит временную метку создания категории;
* Столбец updatedAt хранит временную метку обновления категории.

Таблица Products содержит информацию о продуктах и содержит 10 столбцов:

* Столбец id является первичным ключом, тип данных integer, инкрементируется при добавлении продукта;
* Столбец name хранит название продукта, тип данных character varying, уникальный, ненулевой;
* Столбец calories хранит калорийность продукта, тип данных double precision, ненулевой;
* Столбец fats хранит количество жиров продукта, тип данных double precision, ненулевой;
* Столбец protein количество хранит белков продукта, тип данных double precision, ненулевой;
* Столбец carbs хранит количество углеводов продукта, тип данных double precision, ненулевой;
* Столбец categoryId является внешним ключом, идентификатор категории, тип данных integer, ненулевой;
* Столбец userId является внешним ключом, идентификатор пользователя, который добавил данный продукт, тип данных integer, ненулевой;
* Столбец createdAt хранит временную метку создания продукта;
* Столбец updatedAt хранит временную метку обновления продукта.

Таблица Days содержит информацию о категориях продуктов и содержит 5 столбцов:

* Столбец id является первичным ключом, тип данных integer, инкрементируется при добавлении дня;
* Столбец date хранит дату дня, тип данных date n, ненулевой;
* Столбец userId является внешним ключом, идентификатор пользователя, тип данных integer, ненулевой;
* Столбец createdAt хранит временную метку создания дня;
* Столбец updatedAt хранит временную метку обновления дня.

Таблица Meals содержит информацию о категориях продуктов и содержит 7 столбцов:

* Столбец id является первичным ключом, тип данных integer, инкрементируется при добавлении приема пищи;
* Столбец dayId является внешним ключом, идентификатор дня, тип данных date, ненулевой;
* Столбец productId является внешним ключом, идентификатор продукта, тип данных integer, ненулевой;
* Столбец weight хранит вес продукта, тип данных integer, ненулевой;
* Столбец meal хранит название приема пищи, тип данных character varying, ненулевой;
* Столбец createdAt хранит временную метку создания приема пищи;
* Столбец updatedAt хранит временную метку обновления приема пищи.

2.2 Анализ требований к программному средству и разработка функциональных требований

Приложение имеет 3 роли: не аутентифицированный пользователь, зарегистрированный пользователь, администратор. При первом открытии сайта пользователь не аутентифицирован. Диаграмма вариантов использования не аутентифицированного пользователя представлена на рисунке 2.2.

Рисунок 2.1 – Диаграмма использования не аутентифицированного пользователя

Не аутентифицированный пользователь имеет доступ только к авторизации и регистрации. При попытке перехода на любую другую страницу пользователь получит сообщение о том, что страница не найдена.

Зарегистрировавшись и войдя в аккаунт, пользователь получает роль аутентифицированного пользователя. В данном приложении аутентифицированные пользователи делятся на две роли – это обычные пользователи и администраторы. Диаграмма вариантов использования приложения в зависимости от роли изображена на рисунке 2.3.

1. Разработка приложения

3.1 Проектирование структуры серверной части

На рисунке 3.1 представлена структура разрабатываемого программного средства, включающая в себя директории, конфигурационные файлы и файлы скриптов.

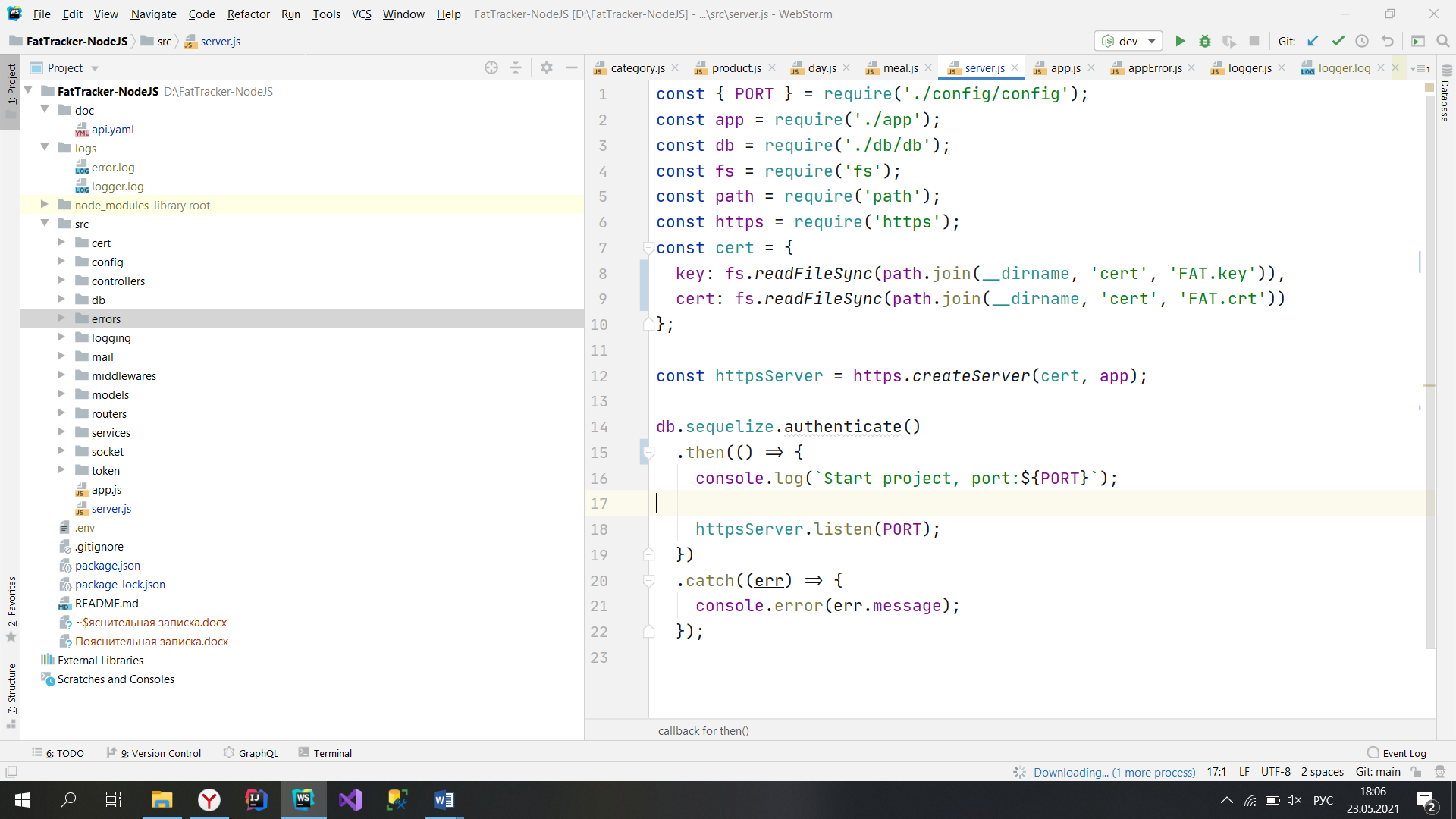


Рисунок 3.1 – Структура проекта сервера

Входной точкой в приложение является файл server.js (рисунок 3.2) в нем создается веб-сервер. Для использования протокола HTTPS необходимо иметь сертификат и секретный ключ. В данном проекте использовались самоподписываемые сертификаты, для которых было необходимо сгенерировать секретные ключи, сертификаты и запрос на сертификацию.

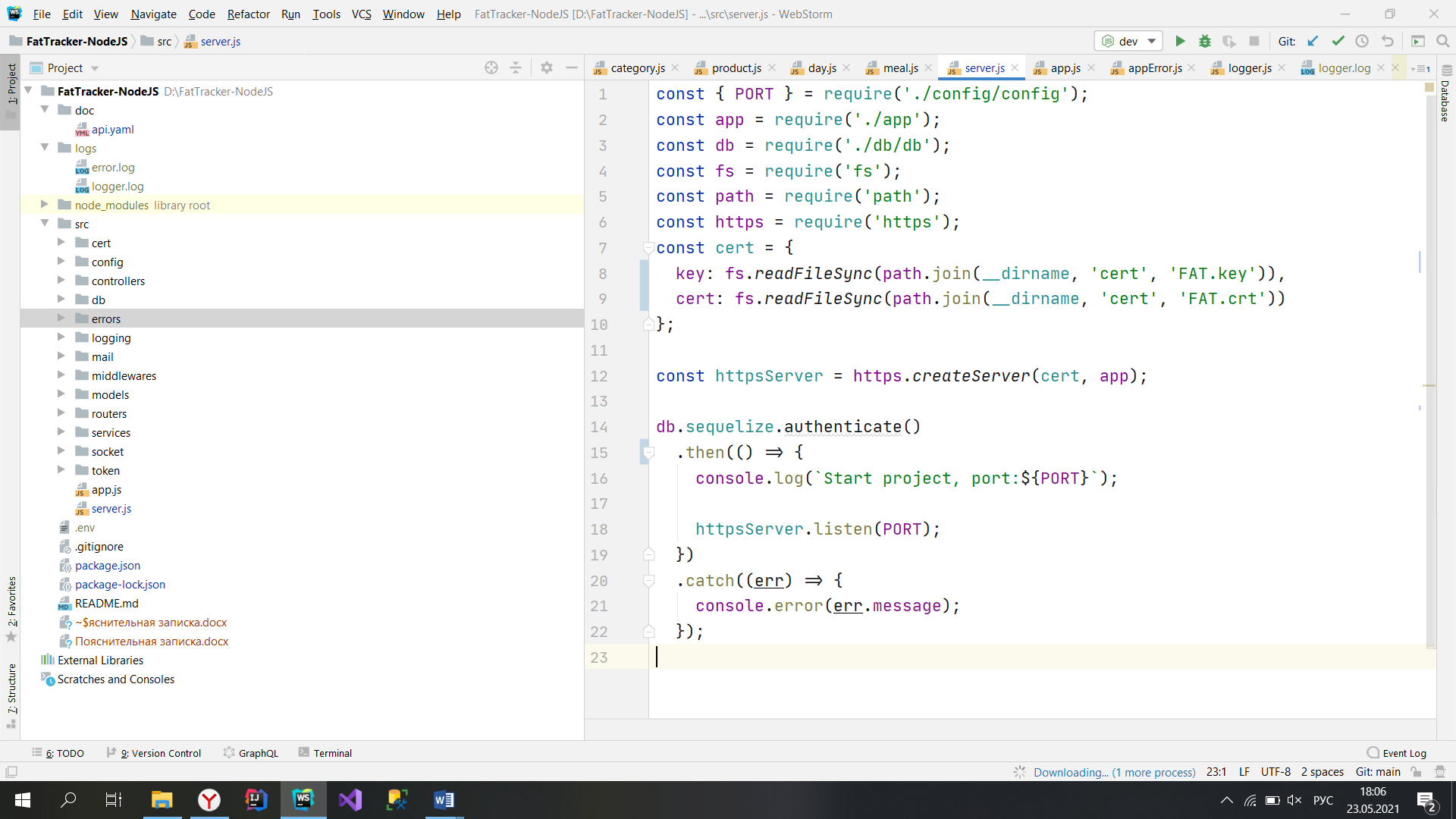


Рисунок 3.2 – Содержание файла server.js

В server.js подключается файл app.js, в котором все необходимые для работы сервера модули, выполняется начальная инициализация приложения, настройка ограничений, вызванных CORS, и регистрация промежуточных обработчиков и маршрутизаторов.

В директории «config» находится файл config.js (рисунок 3.3), в котором экспортируются переменные окружения.

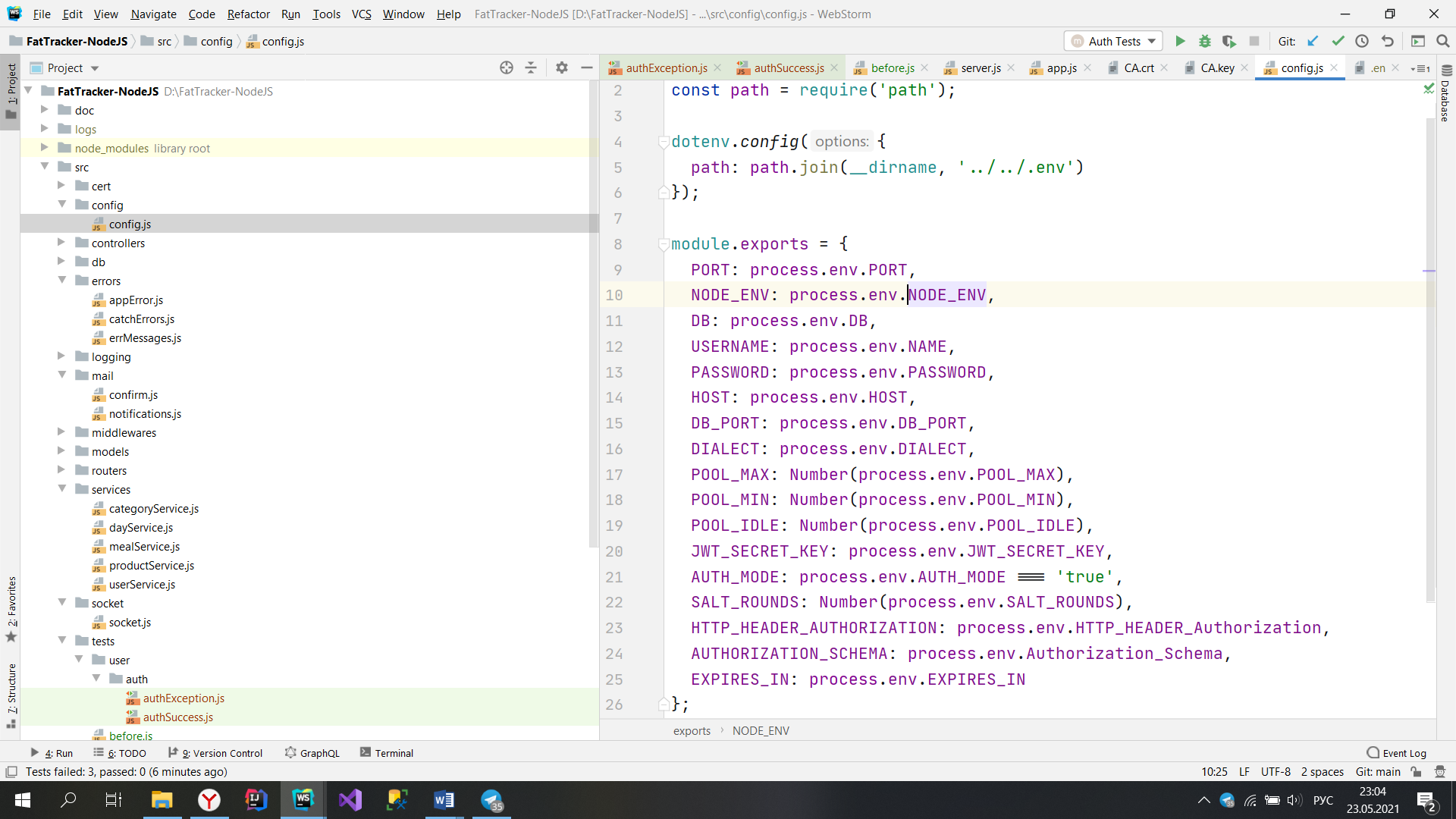


Рисунок 3.3 – Содержание файла config.js

В директории «controllers» находятся контроллеры (рисунок 3.4), которые проверяют входные данные и вызывают соответствующие сервисы.

