Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____**ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**_____

КАФЕДРА _____**КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ**_____

НАПРАВЛЕНИЕ ПОДГОТОВКИ _____**09.03.01 Информатика и вычислительная техника**_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

_____Сайт для размещения_____

_____электронных объявлений_____

Студент ____ИУ6-51Б____ _____ ____Д.Ю. Воронин___
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ ____Д.А. Миков_____
(Подпись, дата) (И.О. Фамилия)

Количество баллов:
за программный продукт –
за расчетно-пояснительную записку –
за доклад и ответы на вопросы –
Итого:
Оценка:

2025 г.

РЕФЕРАТ

Расчетно-пояснительная записка состоит из 23 страниц и включает в себя 14 рисунков, 2 приложения, 5 источников.

Целью данной курсовой работы является проектирование, реализация и документирование системы «Омега» в соответствии с требованиями технического задания и современными практиками разработки программного обеспечения. В процессе выполнения работы применяются принципы объектно-ориентированного программирования и клиент-серверной архитектуры. Результатом работы станет функциональный программный продукт, сопровождаемый расчётно-пояснительной запиской и необходимой программной документацией.

Для реализации используются реляционная база данных PostgreSQL 17, язык программирования Golang 1.23, библиотека React 13.3, среды разработки PgAdmin 4 и Visual Studio Code.

СОДЕРЖАНИЕ

Введение

В условиях стремительного развития цифровых технологий и роста онлайн-торговли всё большее значение приобретают платформы, позволяющие частным лицам и представителям малого бизнеса легко и быстро размещать и находить объявления о продаже товаров и услуг. Существующие крупные маркетплейсы, такие как Avito или Юла, несмотря на широкую популярность, зачастую создают барьеры для новых участников рынка: высокая конкуренция и значительные комиссии делают их использование не всегда выгодным для продавцов и покупателей.

Актуальность разработки собственной платформы «Омега» обусловлена необходимостью предоставления упрощённого и экономически выгодного решения. Проект ориентирован на пользователей, которые ценят простоту, прозрачность и прямое взаимодействие без посредников. Основные преимущества системы – минимальные комиссии, прямые контакты между покупателями и продавцами, а также лаконичный пользовательский интерфейс, не перегруженный избыточной функциональностью.

Разрабатываемая система представляет собой полноценное веб-приложение с разделением на клиентскую (frontend) и серверную (backend) части, реализующее ключевые функции: регистрацию и авторизацию пользователей, размещение, редактирование и удаление объявлений, поиск по заголовку, просмотр объявлений, загрузку изображений и взаимодействие через контактные данные. Особое внимание уделено надёжности, безопасности вводимых данных и целостности хранения информации в базе данных.

1 Анализ требований и уточнение спецификаций

1.1 Анализ задания и выбор технологии, языка и среды разработки

На основе анализа технического задания к проекту «Омега» были определены следующие ключевые требования к системе:

- поддержка двух типов пользователей (авторизованных и неавторизованных);
- реализация CRUD-операций для объявлений (создание, чтение, редактирование, удаление);
- возможность поиска объявлений по заголовку;
- загрузка изображений;
- обеспечение надёжности и контроля вводимых данных;
- поддержка основных веб-браузеров;
- простота пользовательского интерфейса.

Для реализации веб-приложения с такими характеристиками была выбрана клиент-серверная архитектура с разделением на frontend и backend. Такой подход обеспечивает гибкость, масштабируемость и удобство сопровождения.

Frontend разработан с использованием React – современной JavaScript-библиотеки для построения пользовательских интерфейсов. React позволяет создавать модульные, переиспользуемые компоненты и обеспечивает минимально необходимое изменение DOM при изменении состояния [1]. Стилизация выполнена с использованием библиотеки Bootstrap, что позволяет разрабатывать интерфейсы быстрее, чем на чистом CSS, сохраняя при этом современный и аккуратный вид приложения.

Backend реализован на языке Golang с использованием библиотеки echo и дополняющих пакетов, таких как x/crypto для хеширования паролей и gorm.io для взаимодействия с PostgreSQL посредством технологии ORM, которая связывает объекты языка с реляционной базой данных[2]. Go был выбран благодаря своей производительности, простоте синтаксиса, встроенной поддержке конкурентности и сильной типизации, что снижает

вероятность ошибок на этапе разработки. Архитектура backend построена по принципам чистой архитектуры (Clean Architecture): чёткое разделение на слои handlers, service, repository и models обеспечивает тестируемость и независимость от внешних зависимостей. Для управления схемой базы данных используются миграции с помощью утилиты migrate, что гарантирует воспроизводимость и контроль версий структуры БД.

В качестве СУБД выбрана PostgreSQL – надёжная, производительная и полнофункциональная реляционная база данных с отличной поддержкой в экосистеме Go. Её использование позволяет эффективно работать со сложными запросами и обеспечивать целостность данных благодаря реализации в PostgreSQL концепции ACID [3].

Выбранный технологический стек (Go + React + PostgreSQL) полностью соответствует требованиям курсовой работы: он современен, поддерживает разработку программного продукта средней сложности, обеспечивает развитый пользовательский интерфейс и позволяет применять инструменты автоматизации и контроля версий, такие как Git. Кроме того, совместимость этих технологий между собой и их активные сообщества упрощают поиск решений типовых задач и способствуют быстрому устранению возникающих проблем.

1.2 Разработка диаграммы вариантов использования

Для формализации функциональных требований к веб-приложению «Омега» была построена диаграмма вариантов использования в соответствии с методологией UML – рисунок 1. В системе выделены два типа пользователей, указанных в техническом задании. Незарегистрированный пользователь может только просматривать и искать объявления. Зарегистрированный пользователь же обладает полным набором функций: создание, редактирование и удаление собственных объявлений, а также просмотр и поиск.

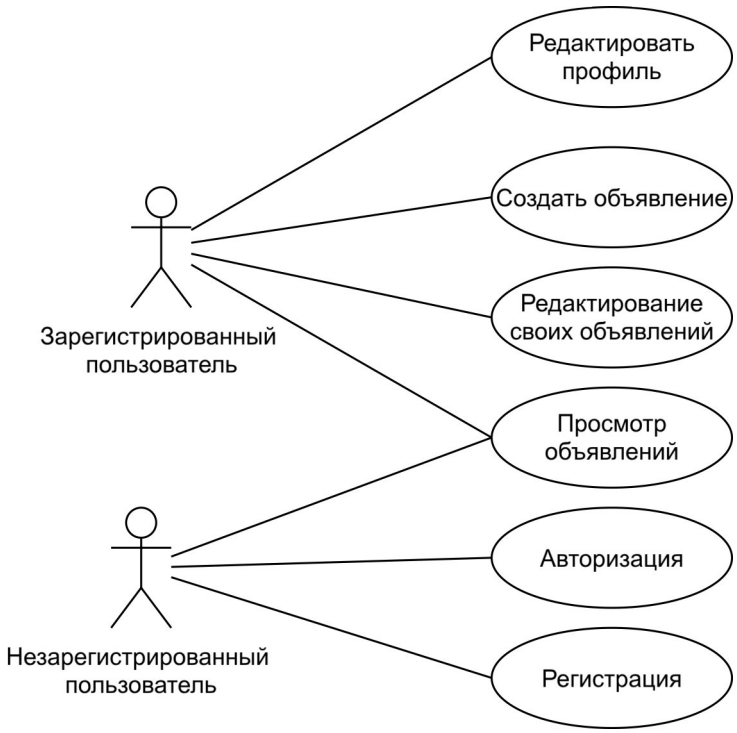

Рисунок 1 – Диаграмма вариантов использования

1.3 Выбор методов решения задачи

Для аутентификация используется JSON Web Token. После проверки логина и хэша пароля (с помощью bcrypt) сервер выдаёт токен с user_id и сроком действия 24 часа. Все защищённые маршруты проверяют наличие и валидность токена через middleware.

Все пользовательские данные проходят валидацию:

- имя пользователя от 3 до 100 символов;
- логин имеет корректный формат электронной почты;
- номер телефона является корректным;
- пароль от 6 символов;
- заголовок объявления от 3 до 100 символов;
- описание объявления от 20 до 5000 символов;
- изображения размером не более 10 МБ в форматах JPEG, JPG, PNG или WEBP.

Поиск объявлений реализован как регистронезависимый подстроковый поиск с использованием SQL-оператора ILIKE в PostgreSQL.

2 Проектирование структуры и компонентов программного продукта

2.1 Разработка интерфейса пользователя

2.1.1 Построение графа (диаграммы) состояний интерфейса

Интерфейс веб-приложения «Омега» реализован в виде одностраничного приложения (SPA) на основе библиотеки React. В соответствии с требованиями технического задания была разработана диаграмма состояний интерфейса зарегистрированного и незарегистрированного пользователей – рисунок 2.

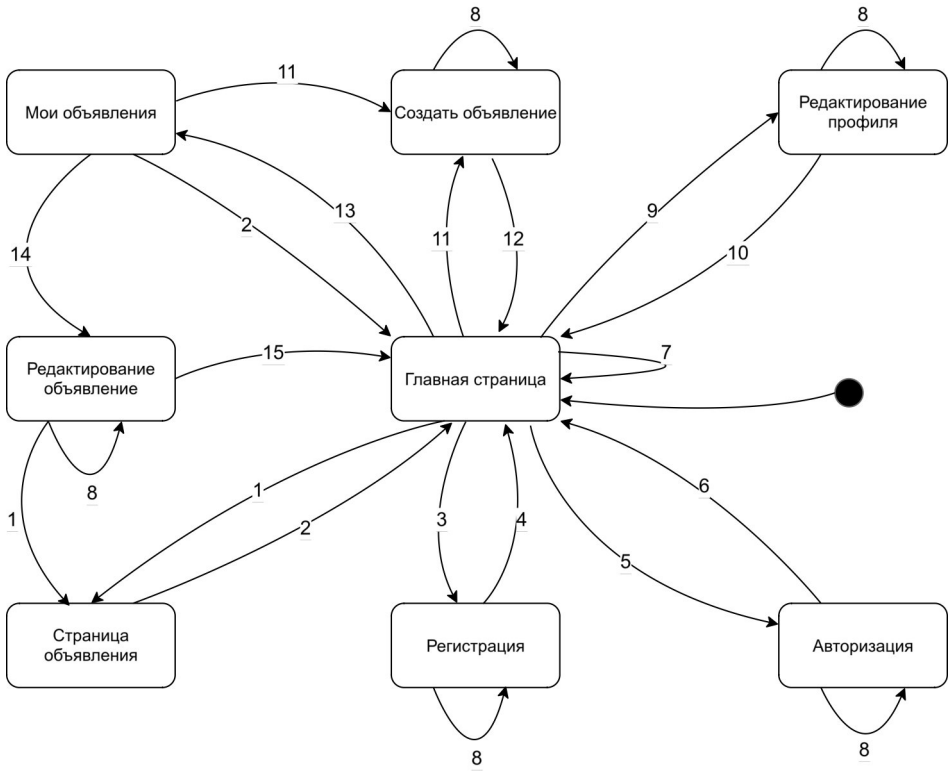

Рисунок 2 – Диаграмма состояний интерфейса

Ниже представлены условия переходов между состояниями:

- 1 – нажатие на кнопку «Просмотр» у объявления;
- 2 – возврат на главную страницу;
- 3 – нажатие на кнопку «Регистрация»;
- 4 – возврат на главную страницу или успешная регистрация;
- 5 – нажатие на кнопку «Войти»;
- 6 – возврат на главную страницу или успешная авторизация;
- 7 – поиск или фильтр по категориям;

- 8 – ввод некорректных данных;
- 9 – нажатие на кнопку «Профиль» у выпадающего меню профиля;
- 10 – успешное редактирование профиля;
- 11 – нажатие на кнопку «Создать объявление»;
- 12 – возврат на главную страницу или успешное создание объявление;
- 13 – нажатие на кнопку «мои объявления»;
- 14 – нажатие на кнопку «Редактировать» у объявления;
- 15 – возврат на главную или успешное редактирование объявления.

2.1.2 Разработка форм ввода-вывода информации

Были разработаны следующие формы интерфейса – рисунки 3-10.

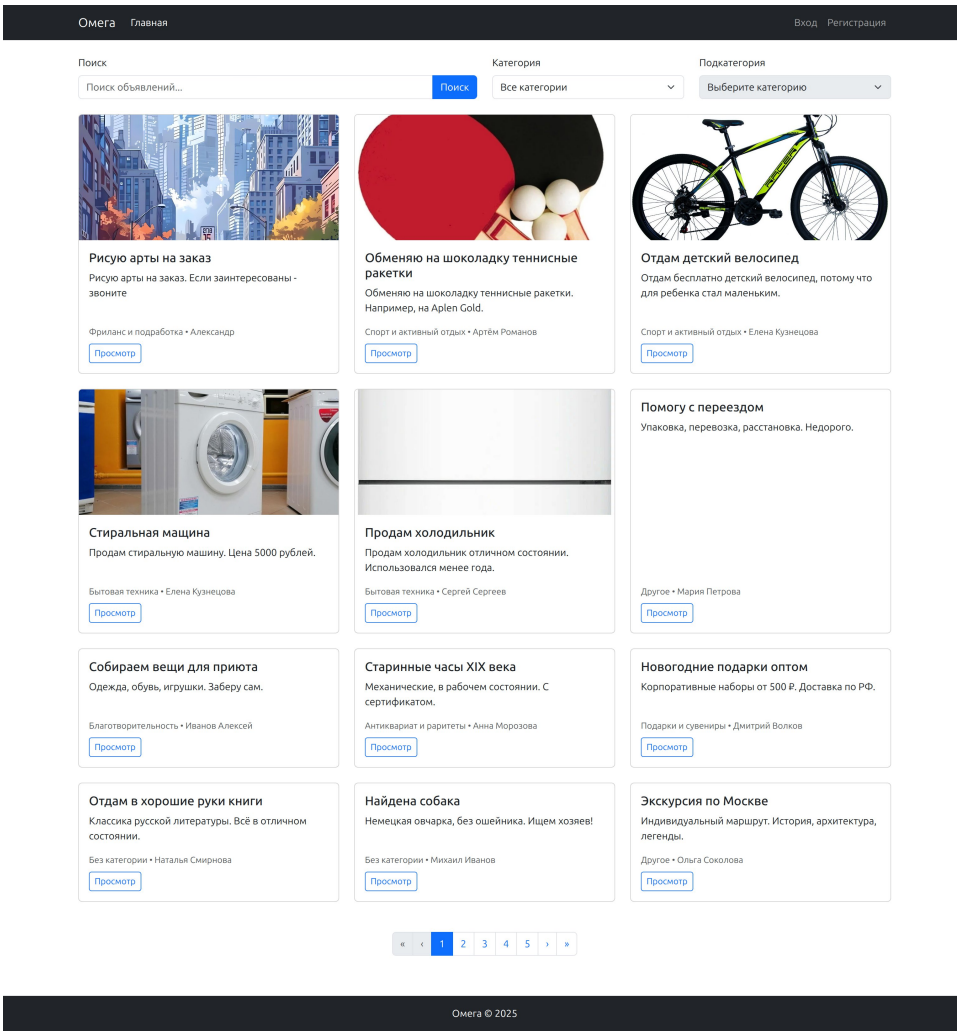

Рисунок 3 – Форма главной страницы

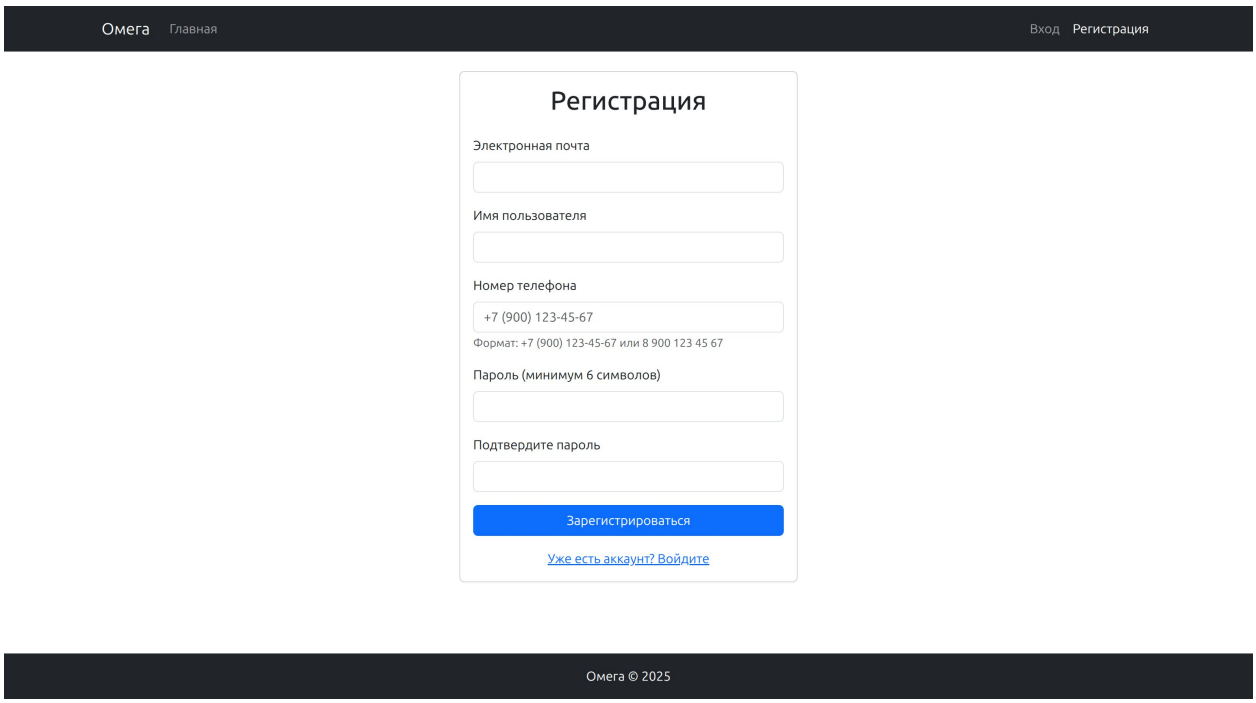

Рисунок 4 – Форма регистрации

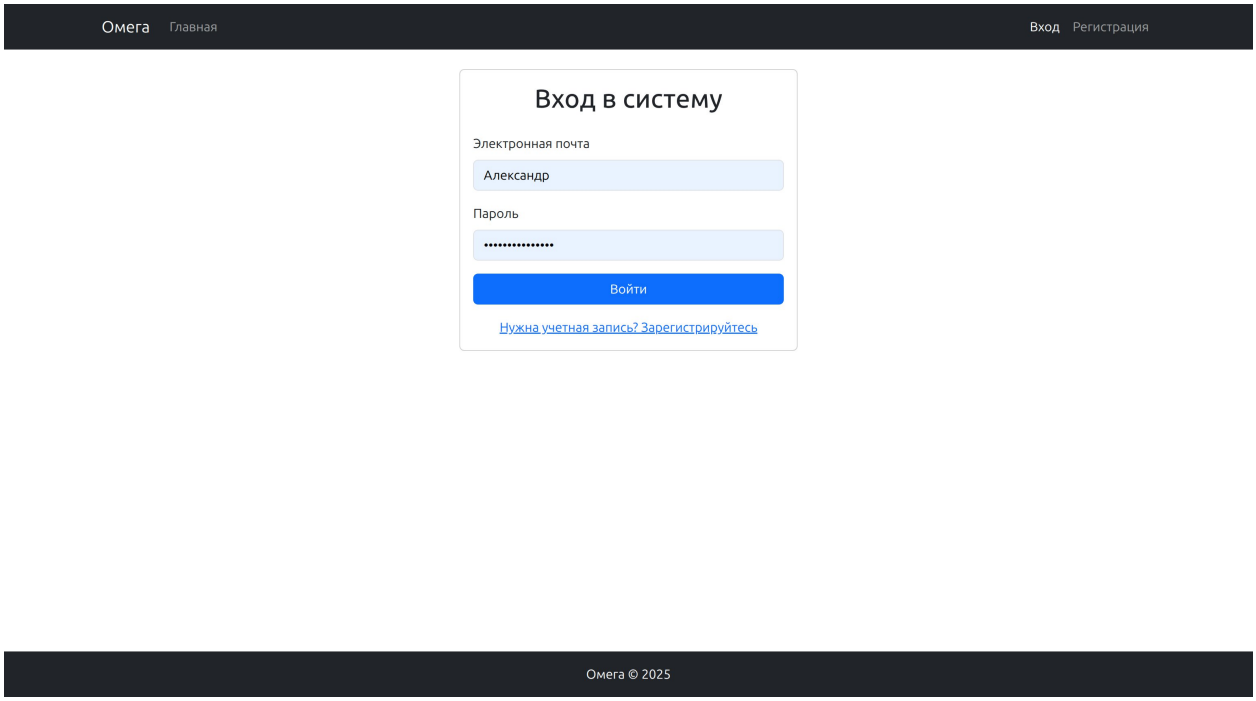

Рисунок 5 – Форма входа в систему

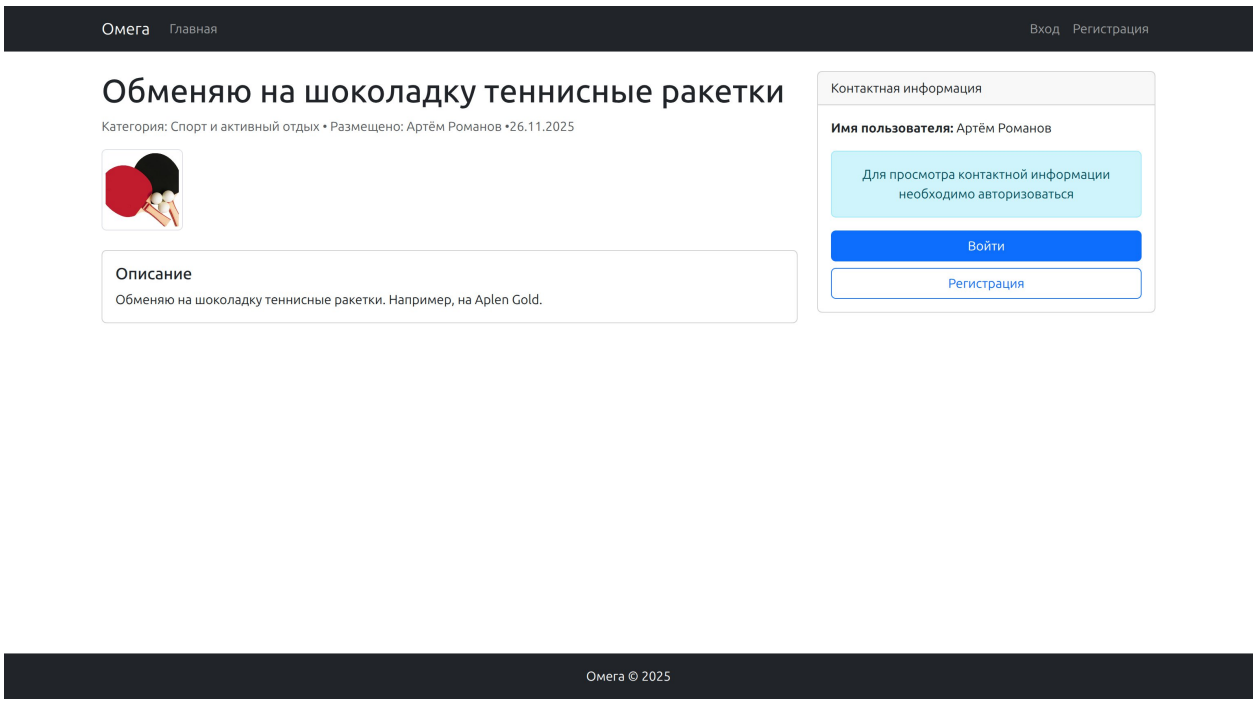

Рисунок 6 – Форма просмотра объявления

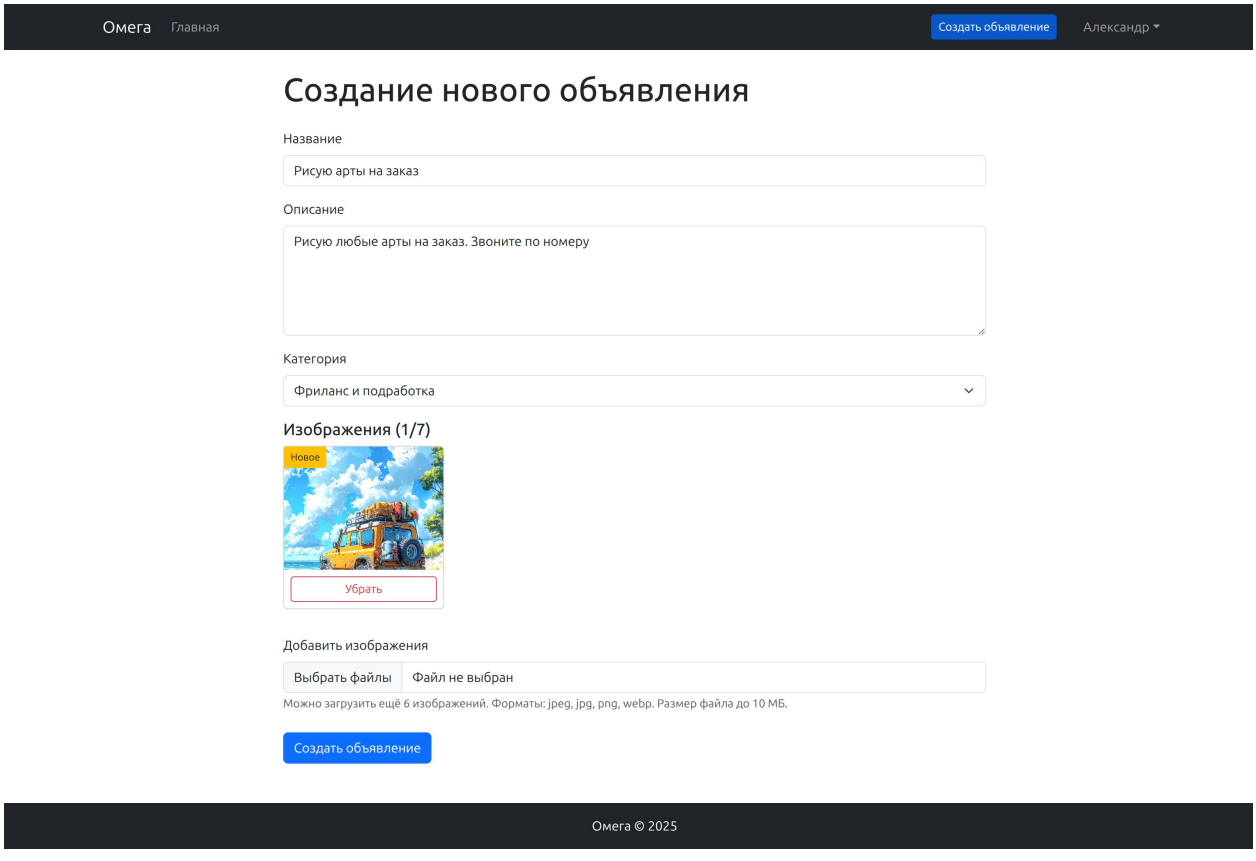

Рисунок 7 – Форма создания объявления

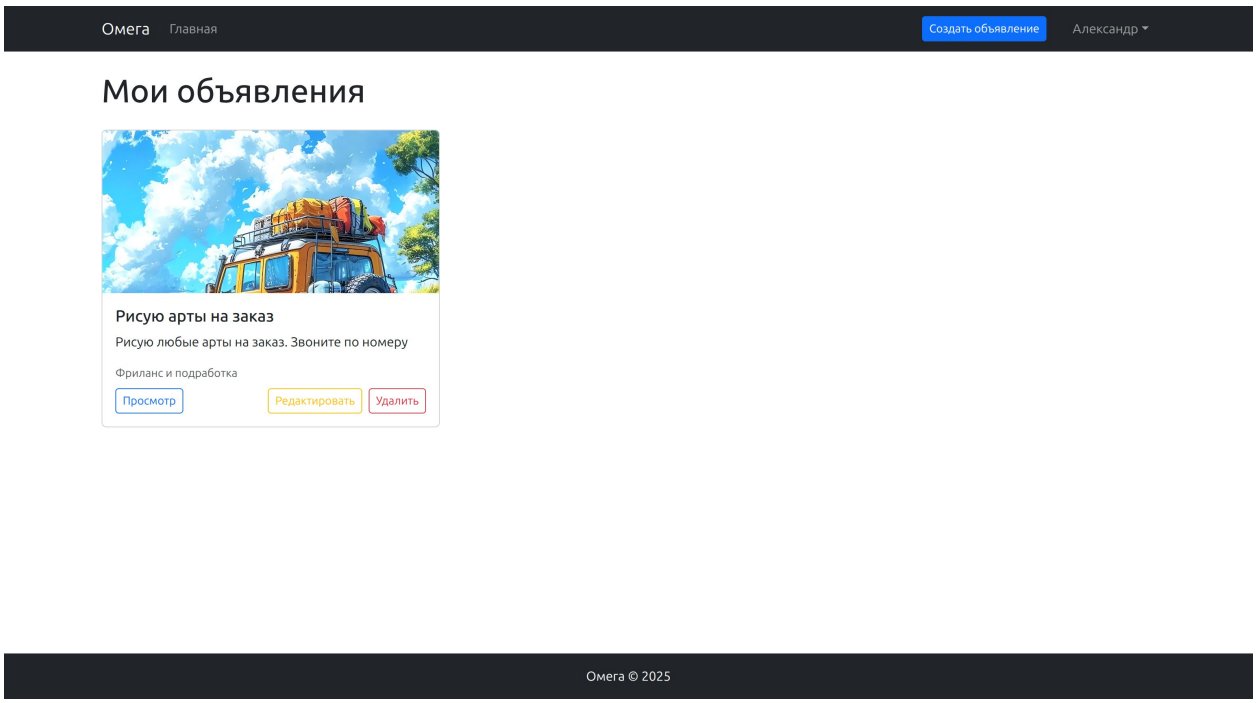

Рисунок 8 – Форма объявлений авторизованного пользователя

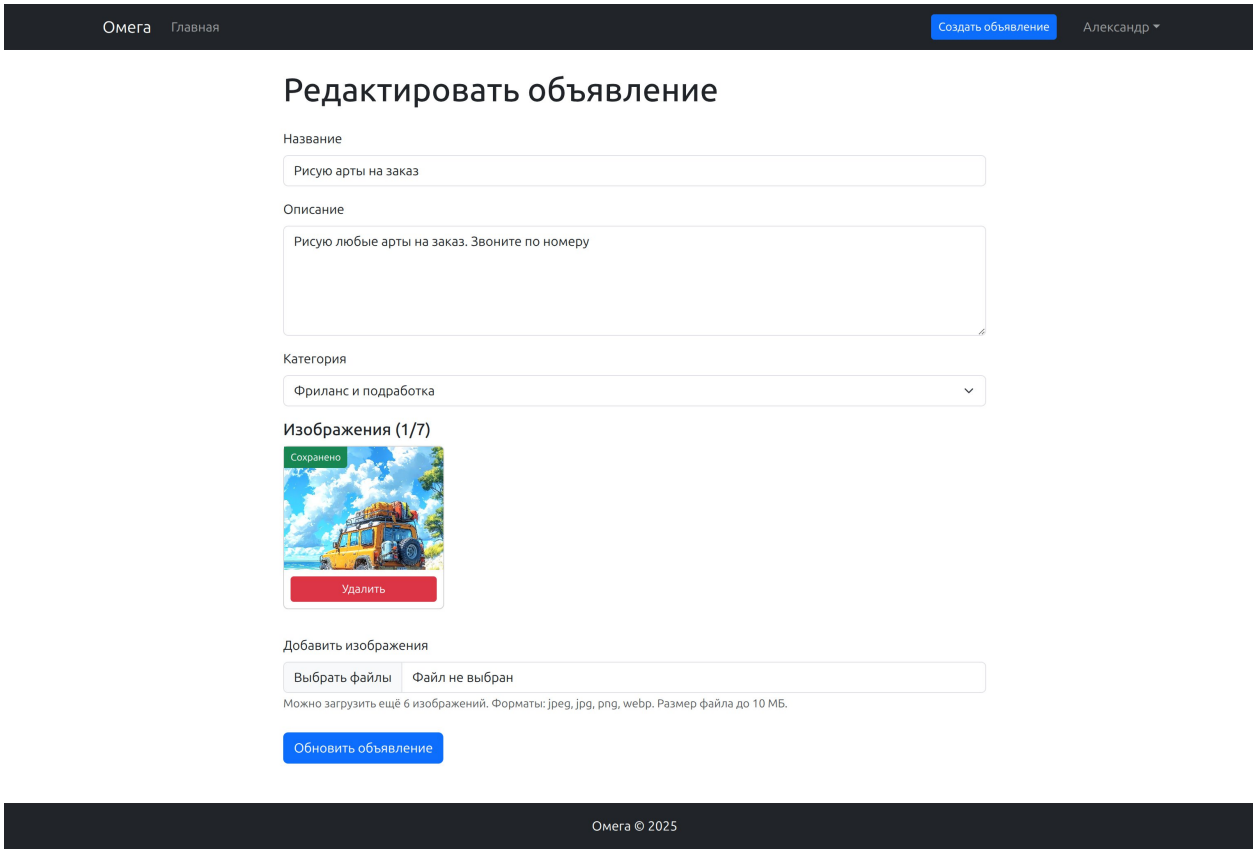

Рисунок 9 – Форма редактирования объявления

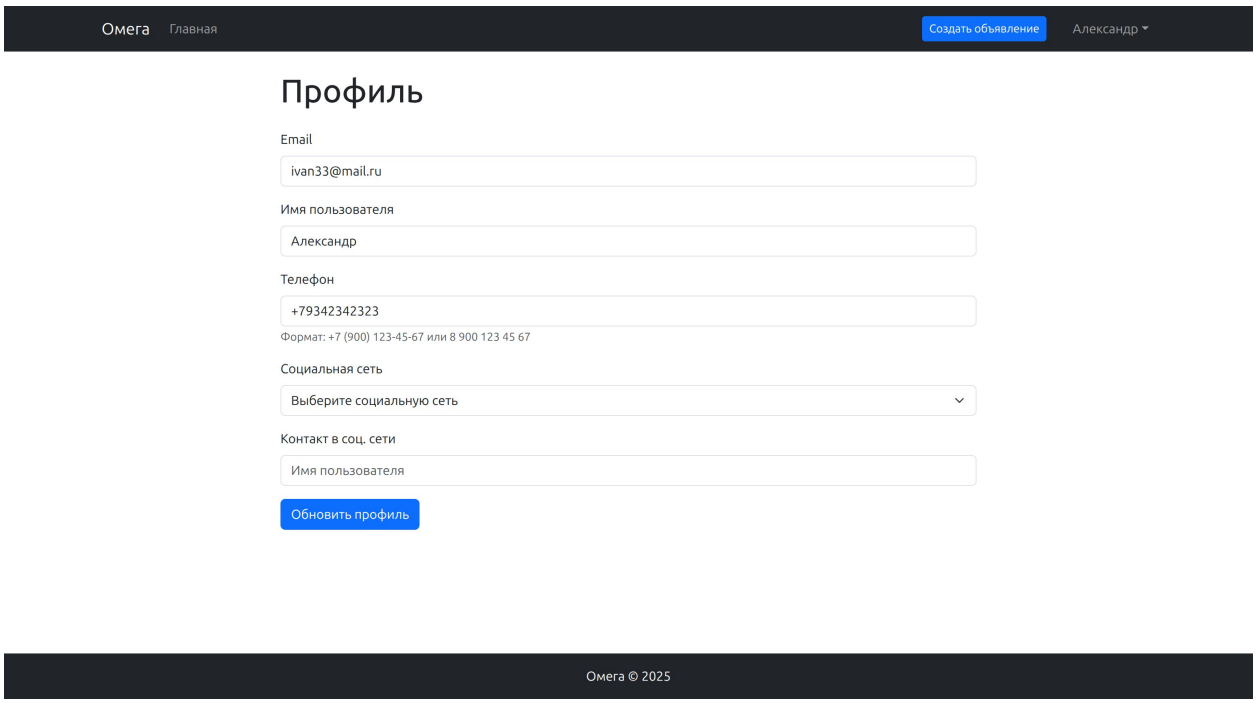

Рисунок 10 – Форма редактирования профиля

2.3 Разработка структурной схемы программного продукта

Структурная схема программного продукта отражает состав и взаимодействие компонентов системы. В проекте «Омега» реализована клиент-серверная архитектура, в которой выделены две основные части: клиентская (frontend) и серверная (backend).

Серверная часть реализована на языке Go и построена по принципу чистой архитектуры (Clean Architecture), что позволяет отделить бизнес-логику от деталей реализации (HTTP, база данных). Выделены следующие слои:

- handlers – обрабатывают входящие HTTP-запросы, выполняют первичную валидацию, вызывают сервисы и формируют ответ;
- services – содержат бизнес-логику приложения. Например, логика создания объявления, проверки прав доступа;
- repository – обеспечивают доступ к данным, инкапсулируя взаимодействие с PostgreSQL.

Все зависимости направлены внутрь: Handlers зависят от Services, Services – от Repository.

Клиентская компонента реализована с использованием библиотеки React и состоит из следующих логических блоков:

- pages – компоненты верхнего уровня, представляющие собой отображаемые страницы;
- components – переиспользуемые элементы интерфейса;
- services – модули для взаимодействия с API;
- contexts – управление глобальным состоянием, в частности, AuthContext для хранения токена и данных пользователя.

Фронтенд взаимодействует с бэкендом посредством RESTful API, передавая данные по HTTP в формате JSON и multipart/form-data в случае загрузки файлов.

Для сохранения и получения данных серверная часть обращается к СУБД PostgreSQL.

Полученная схема представлена на рисунке 11.

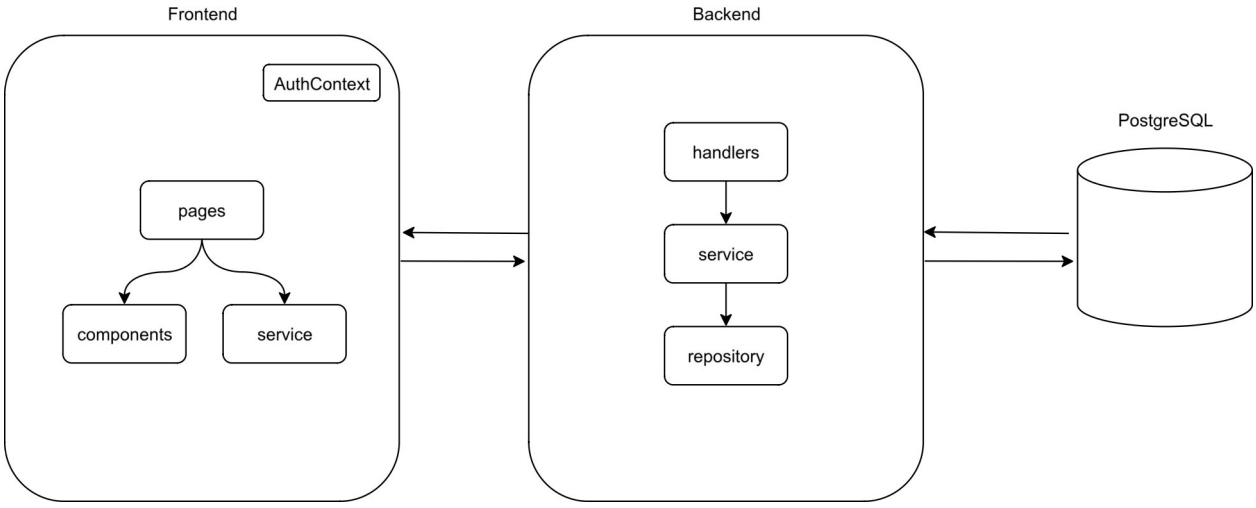

Рисунок 11 – Структурная схема программного продукта

2.4 Проектирование даталогической модели базы данных

Проектирование даталогической моделей базы данных является этапом, на котором структура данных формализуется с учетом требований предметной области. Даталогическая модель конкретизирует сущности, их атрибуты и взаимосвязи в виде таблиц, типов данных и ограничений. Полученная модель представлена на рисунке 12.

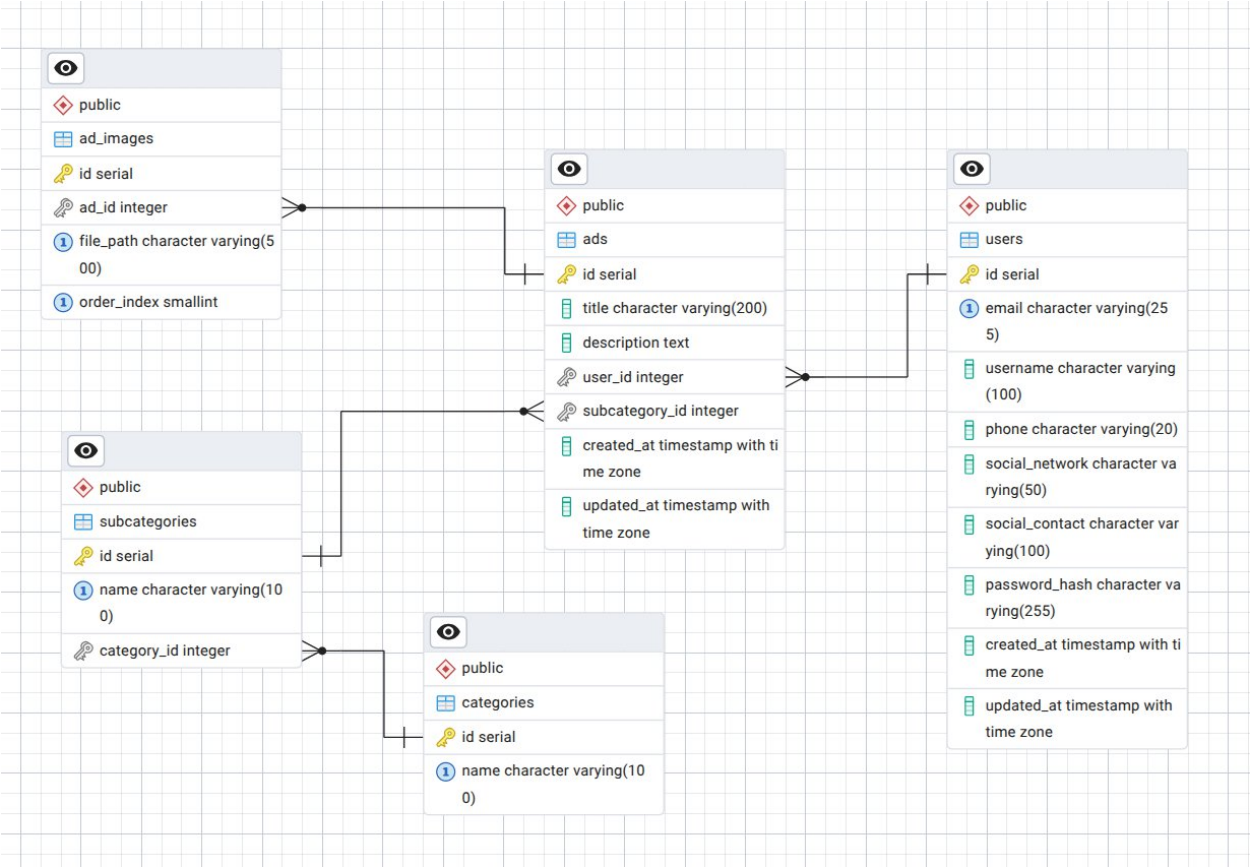

Рисунок 12 – Даталогическая модель базы данных

Главной сущностью системы является объявление (ads). Оно объединяет информацию о пользователе, категории и прикреплённых изображениях, формируя основной контент платформы. Сущности users, categories и subcategories выступают в качестве справочных: они обеспечивают контекст для объявлений (кто разместил, к какой тематике относится). Второстепенная сущность ad_images детализирует визуальное представление объявления, позволяя прикреплять несколько изображений с учётом порядка отображения.

Нормализация данных достигнута за счёт чёткого разделения ответственности между таблицами и устранения дублирования. Контактные данные (email, телефон, соцсети) хранятся только в users, а не дублируются в каждом объявлении. Категории и подкатегории вынесены отдельно, что позволяет гибко управлять иерархией тематик. Изображения хранятся как отдельные записи в ad_images, что упрощает их управление и масштабирование.

Ограничения целостности обеспечивают надёжность системы: в каждой таблице определён первичный ключ (PRIMARY KEY), гарантирующий уникальность записей; внешние ключи (FOREIGN KEY) с каскадным удалением автоматически удаляют связанные данные, предотвращая появление «висячих» ссылок; кроме того, наложены уникальные ограничения (UNIQUE) на поля email в таблице users, name в таблицах categories и subcategories, а также на пару полей file_path и order_index в таблице ad_images, что исключает дублирование критически важной информации [4].

Таким образом, спроектированная даталогическая модель обеспечивает эффективное хранение, целостность и масштабируемость данных, полностью соответствующих функциональным и надёжностным требованиям, предъявляемым к веб-приложению «Омега».

2.5 Проектирование классов для реализации интерфейса и предметной области

В соответствии с требованиями технического задания, в рамках разработки веб-приложения «Омега» была выполнена детальная декомпозиция системы на компоненты с использованием принципов объектно-ориентированного проектирования. Несмотря на то, что язык Go не поддерживает классы в традиционном смысле, он предоставляет мощные инструменты для реализации ООП-концепций через структуры, методы и интерфейсы [5]. В проекте применена архитектура, основанная на паттерне «Чистая архитектура» (Clean Architecture), где взаимодействие между слоями строго определяется через абстракции, представленные в виде интерфейсов.

На диаграмме (рисунок 13) отражена иерархия компонентов приложения, начиная с главного модуля App, который инициализирует и запускает все основные обработчики.. Каждый обработчик зависит от своего сервиса, например, AuthHandler зависит от AuthService. AuthService в свою очередь, зависит от репозитория, который используется для доступа к данным.

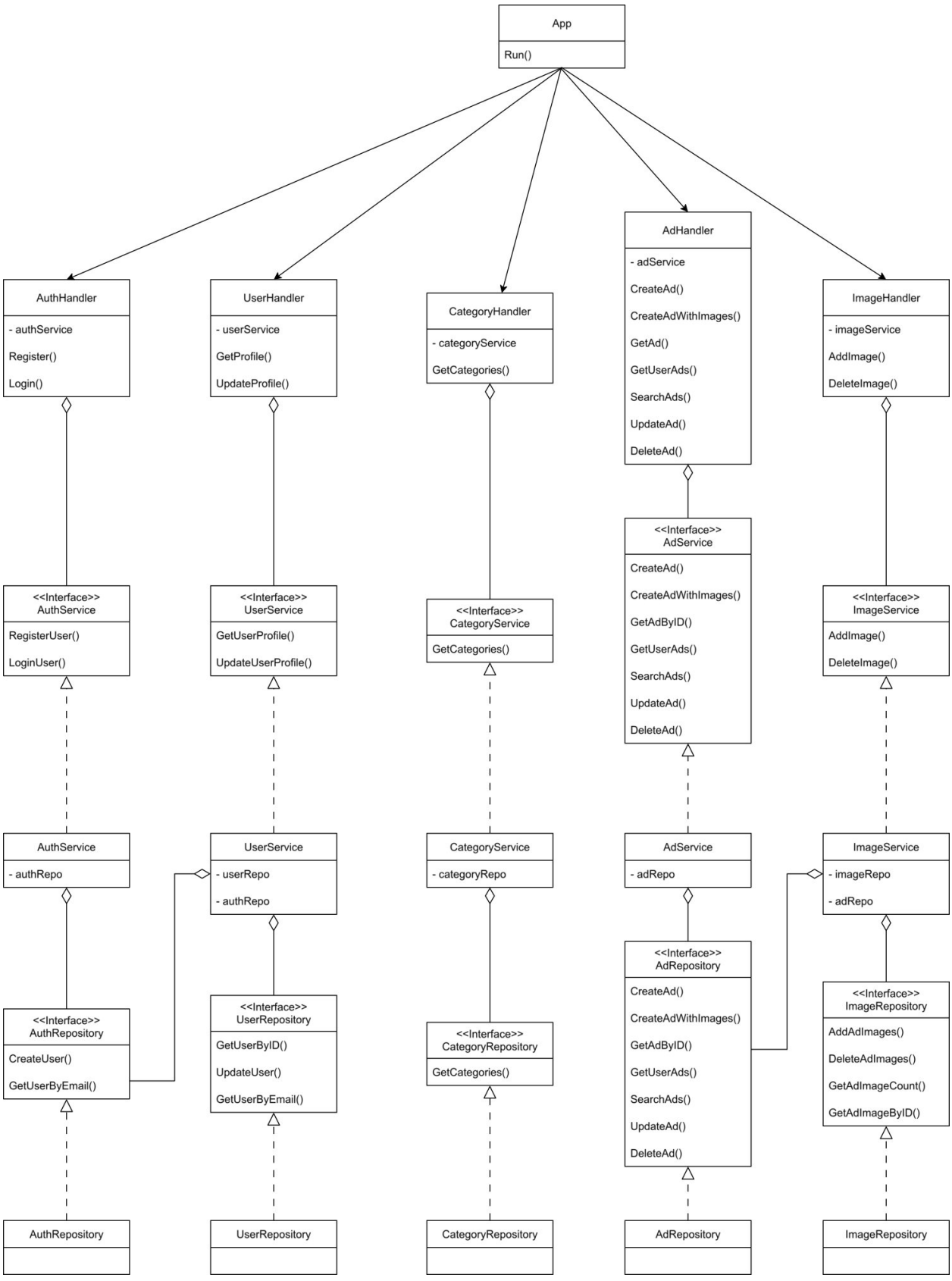

Рисунок 13 – Диаграмма классов

Слой обработчиков отвечает за прием HTTP-запросов, парсинг входных данных, вызов соответствующих методов сервиса и формирование ответа.

Например, AdHandler.CreateAd() получает данные из запроса, передаёт их в AdService.CreateAd(), а затем возвращает клиенту JSON-ответ или ошибку.

Слой сервисов реализует бизнес-логику приложения. Здесь происходит валидация данных, проверка прав доступа, логика создания объявления с изображениями и т.д.

Слой репозиториев осуществляет работу с базой данных. Каждый репозиторий реализует свой интерфейс и содержит методы для выполнения CRUD-операций и сложных запросов.

Все зависимости между слоями определены через интерфейсы, что обеспечивает слабую связанность и упрощает тестирование. Например, AdService зависит не от конкретной реализации AdRepository, а от интерфейса AdRepository, что позволяет легко заменить реализацию.

Модуль App служит точкой входа и координатором всех компонентов. Он создаёт экземпляры сервисов и репозиториев, внедряет зависимости и запускает HTTP-сервер.

Таким образом, данная архитектура обеспечивает:

- масштабируемость – добавление новых функций не требует изменения существующего кода;
- тестируемость – возможность изолированного тестирования каждого слоя;
- поддерживаемость – чёткое разделение ответственности и слабая связанность между компонентами.

2.6 Разработка диаграммы последовательности действий

Для уточнения взаимодействия компонентов системы при выполнении успешной операции создания объявления с прикреплёнными изображениями была разработана диаграмма последовательности действий.

Выбор именно этого сценария обусловлен его комплексностью: он включает в себя аутентификацию пользователя, валидацию входных данных, загрузку файлов, сохранение метаданных в базу данных и формирование

ответа клиенту. Это позволяет продемонстрировать взаимодействие всех основных слоёв архитектуры. Диаграмма представлена на рисунке 14.

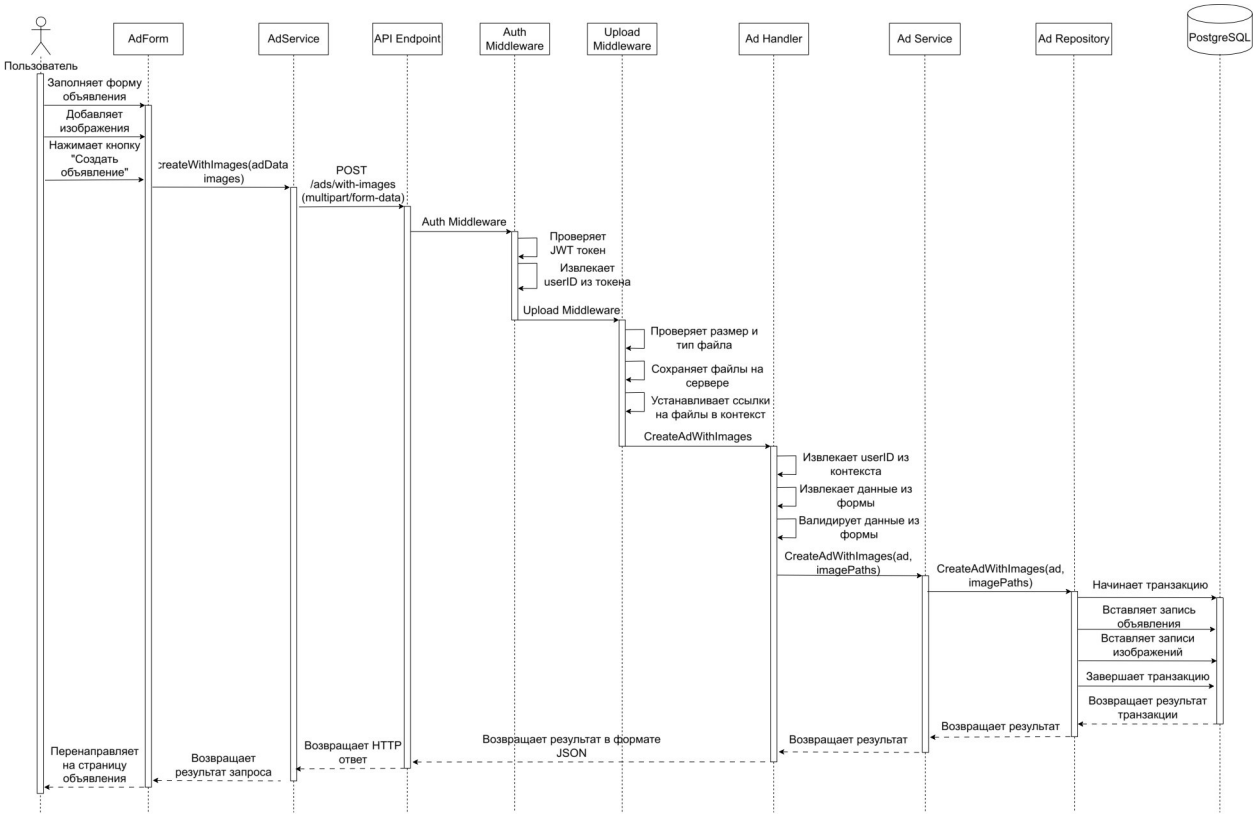

Рисунок 14 – Диаграмма последовательности действий при создании объявления с изображениями

Диаграмма иллюстрирует следующую логику. Пользователь, находясь на странице «Создать объявление», заполняет форму, прикрепляет файлы и нажимает кнопку «Создать объявление». Далее запрос отправляется к сервису, который отправляет HTTP запрос серверу. На сервере запрос проходит через промежуточные обработчики, которые проверяют авторизацию и валидность файлов. Далее в обработчике происходит валидация данных и их передача в слой бизнес логики, которая в свою очередь передаёт запрос слою работы с базой данных. Он формирует транзакцию и вставляет данные. После этого в обратном порядке возвращаются ответы. При этом обработчик напрямую возвращает ответ API Endpoint, минуя промежуточные обработчики (middleware).

3 Выбор стратегии тестирования и разработка тестов

Для проверки соответствия реализации функциональным требованиям ТЗ была выбрана стратегия функционального (чёрного ящика) тестирования. Цель – убедиться, что система корректно реагирует на входные данные и выполняет заявленные функции независимо от внутренней реализации.

В таблице 1 приведены тестовые сценарии, охватывающие ключевые пользовательские действия.

Таблица 1 – Таблица функциональных тестов

№	Условие теста	Значение исходных данных	Ожидаемый результат	Полученный результат	Статус теста
1	Тестирование регистрации пользователя с валидными данными	Логин: user123@mail.ru, имя: Иван, номер телефона: +79523423423 пароль: password123	Успешная регистрация	Успешная регистрация	Тест пройден
2	Тестирование регистрации пользователя с ошибкой в данных	Логин: user123@mail.ru, имя: Иван, номер телефона: +795234234 пароль: password123	Ошибка валидации данных и сообщение об этом	Ошибка валидации и отображение ошибки «Неверный формат номера телефона (пример: +7 (900) 123-45-67)»	Тест пройден
3	Тестирование корректной авторизации пользователя	Логин: user123@mail.ru, пароль: password123	Успешная авторизация	Успешная авторизация	Тест пройден
4	Нажатие на кнопку «Просмотр» у объявления	Объявление с названием «Стиральная машина», заданным описанием и картинкой	Открытие страницы объявления	Открытие страницы объявления и отображением соответствующей ему информации	Тест пройден

Продолжение таблицы 1

5	Тестирование создания объявления с картинками	Название: Наручные часы, описание: Стильные мужские часы на металлическом браслете, Категория: Аксессуары Файл: watches.jpg (размер 131,4 КБ)	Успешное создание объявления	Успешное создание объявления и открытие страницы созданного объявления с заданными данными, включая картинку	Тест пройден
6	Тестирование редактирования объявления	Изменение название объявления на «Продаю наручные часы» и добавление файла watches1.jpg (размер 61 КБ)	Успешное редактирование объявления	Успешное редактирование объявления и открытие страницы отредактированного объявления с заданными данными, включая новую картинку	Тест пройден
7	Тестирование поиска объявления	Значение поиска: Продаю наручные часы	Найдено ранее созданное объявление	Найдено единственное объявление, которое было создано раннее	Тест пройден
8	Тестирование удаления объявления	Удаления объявления с названием «Продаю наручные часы»	Объявление удалено	Объявление успешно удалено	Тест пройден
9	Тестирование редактирования профиля	Социальная сеть: Telegram, Контакт в соц. сети: @abacaba	Профиль успешно обновлён	Данные профиля успешно обновлены	Тест пройден

Заключение

В ходе выполнения курсовой работы был спроектирован и реализован веб-сайт «Омега», предназначенный для размещения и поиска объявлений. Были реализованы все функции, указанные в техническом задании: регистрация, авторизация, создание, редактирование и удаление объявлений, поиск по заголовку, загрузка изображений. Применена клиент-серверная архитектура, обеспечена надёжность ввода, целостность данных и безопасность доступа. Проведено функциональное тестирование, подтвердившее корректность работы системы. Все этапы работы сопровождались диаграммами для наглядного представления архитектуры, логики взаимодействия компонентов и структуры пользовательского интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация к React [Электронный ресурс]. URL: https://react.dev/learn (дата обращения: 20.11.2025)

2. Документация к библиотеке GORM [Электронный ресурс]. URL: https://gorm.io (дата обращения: 20.11.2025)

3. Фомин М.М. Реляционные базы данных. Учебное пособие для бакалавров: МГТУ имени Н.Э. Баумана, 2023. 161 с.

4. Документация PostgreSQL и Postgres Pro [Электронный ресурс]. – URL: https://postgrespro.ru/docs (дата обращения 20.11.2025).

5. Effective Go: официальное руководство по написанию идиоматического кода на языке Go [Электронный ресурс]. URL: https://go.dev/doc/effective_go (дата обращения: 20.11.2025)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ – ИУ6**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

САЙТ ДЛЯ РАЗМЕЩЕНИЯ ЭЛЕКТРОННЫХ ОБЪЯВЛЕНИЙ

Техническое задание на курсовую работу
по дисциплине Технология разработки программных систем

Листов 7

Студент _гр. ИУ6-51Б__ _____ ___Д.Ю. Воронин__
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы,
(Кандидат технических наук, доцент) _____ ____Д.А. Миков____
(Подпись, дата) (И.О. Фамилия)

Москва, 2025

1 ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку сайта «Омега», используемого для размещения и поиска товаров и услуг и предназначенной для частных лиц и предпринимателей, желающий осуществлять онлайн-продажи на локальном рынке.

Актуальность данной разработки обусловлена растущим спросом на онлайн-платформы для торговли и обмена товарами. Существующие аналоги (такие как Avito, Юла) имеют большую внутреннюю конкуренцию и высокие комиссии, что создает барьер для малого бизнеса. Разрабатываемый маркетплейс предлагает простую систему поиска с фильтрами и минимальные комиссии. Отличительными особенностями являются: упрощенный интерфейс и система прямых контактов между покупателями и продавцами.

2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Сайт для размещения электронных объявлений «Омега» разрабатывается в соответствии с тематикой кафедры «Компьютерные системы и сети» (ИУ6).

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Основное назначение сайта для размещения электронных объявлений «Омега» заключается в предоставлении пользователям платформы для размещения, поиска и просмотра электронных объявлений о продаже товаров и услуг.

4 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1 Требования к функциональным характеристикам

4.1.1 Выполняемые функции

4.1.1.1 Для зарегистрированного пользователя:

- просмотр объявления;
- создание объявления;

−редактирования объявления;

−удаление объявления;

−поиск объявления по его заголовку.

4.1.1.2 Для незарегистрированного пользователя:

−просмотр объявления;

−поиск объявления по его заголовку.

4.1.2 Исходные данные:

−логин (электронная почта);

−заголовок объявления;

−фотография объявления;

−контактные данные (номер телефона, контакт в социальной сети).

4.2 Требования к надежности

4.2.1 Предусмотреть контроль вводимой информации.

4.2.2 Предусмотреть блокировку некорректных действий пользователя.

4.2.3 Обеспечить целостность информации в базе данных.

4.3 Условия эксплуатации

4.3.1 Условия эксплуатации в соответствии с СанПиН 2.2.2/2.4.1340-03.

4.3.2 Обслуживание

4.3.2.1 В случае какого-либо сбоя работы откатить сервис к последней рабочей версии.

4.3.3 Обслуживающий персонал

4.3.3.1 Программист, который будет следить за стабильностью работы веб-приложения, а также в случае сбоев оказывать техническую поддержку.

4.4 Требования к составу и параметрам технических средств

4.4.1 Программное обеспечение должно функционировать на IBM-совместимых персональных компьютерах.

4.4.2 Минимальная конфигурация технических средств:

4.4.2.1 Тип процессора ..... Intel Core i5

4.4.2.2 Объем ОЗУ ..... 8 Гб.

4.5 Требования к информационной и программной совместимости

4.5.1 Программное обеспечение должно работать под управлением браузеров, таких как «Яндекс», «Google Chrome», «Opera», «Mozilla Firefox».

4.5.2 Входные данные представлены в следующем формате: текст, изображения.

4.5.3 Программное обеспечение должно иметь вид: начальную страницу с возможностью авторизации, регистрации, просмотра размещённых объявлений и их поиска, форму для создания и редактирования объявлений, страницу объявления.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

5.2 Разрабатываемое программное обеспечение должно включать справочную систему.

5.3 В состав сопровождающей документации должны входить:

5.3.1 Расчетно-пояснительная записка на 25-30 листах формата А4 (без приложений 5.3.2, 5.3.3 и 5.3.4).

5.3.2 Техническое задание (Приложение А).

5.3.3 Руководство пользователя (Приложение Б).

5.4 Графическая часть должна быть включена в расчетно-пояснительную записку в качестве иллюстраций:

5.4.1 Диаграмма вариантов использования.

5.4.2 Граф состояний интерфейса.

5.4.3 Формы интерфейса.

5.4.4 Схема структурная программного обеспечения.

5.4.5 Даталогическая модель базы данных.

5.4.6 Диаграмма классов.

5.4.7 Диаграмма последовательности действий.

5.4.8 Таблицы тестов.

6 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Этап	**Содержание этапа**	**Сроки и объем**	**Представляемые результаты**	
			Спецификации и программный продукт	**Документы**
1.	Выбор темы, составление задания, решение организационных вопросов	1..2 недели (10 %)	-	**Заполненный бланк задания на курсовую работу – вывешивается на сайт кафедры для получения утверждающей подписи заведующего кафедрой**
2.	Анализ предметной области, разработка ТЗ. Исследование методов решения, выбор основных проектных решений	3..4 недели	Результаты декомпозиции предметной области. Эскизный проект: интерфейс, схемы, возможно, часть программы (выбранные готовые решения).	Фрагмент расчетно-пояснительной записки с обоснованием выбора средств и подходов к разработке
3.	**Сдача ТЗ**	**4 неделя (25 %)**	**-**	**Техническое задание – утверждается руководителем**
4.	Проектирование и реализация основных компонентов – ядра программы	5..7 недели	Технический проект основной части: структура программы, алгоритмы программ, описания структур данных, диаграмма классов – в зависимости от выбранной технологии разработки. Программный продукт, реализующий основные функции (демонстрируется руководителю)	Фрагмент расчетно-пояснительной записки с обоснованием разработанных спецификаций Тексты части программного продукта, реализующего основные функции.

Этап	Содержание этапа	Сроки и объем	Представляемые результаты	
			Спецификации и программный продукт	**Документы**
5.	**Сдача прототипа программного продукта**	**7 неделя (50 %)**	**Прототип программного продукта – демонстрируется руководителю**	
6.	Разработка компонентов, обеспечивающих функциональную полноту	8..10	Рабочий проект программы. Готовая программа	Черновик расчетно-пояснительной записки. Тексты программного продукта.
7.	**Сдача программного продукта**	**11 неделя (75 %)**	**Готовая программа – оценивается руководителем в баллах**	**-**
8.	Тестирование программы и подготовка документации	12..14	Тесты и результаты тестирования.	РПЗ и Руководство пользователя.
9.	**Оформление и сдача документации**	**14 неделя (90 %)**	**–**	**Расчетно-пояснительная записка и Руководство пользователя – проверяются и подписываются руководителем**
10.	Защита курсовой работы	15..16 недели (100%)	–	Доклад (3-5 минут). Защита курсовой работы. Подписанная документация – вывешивается на сайт кафедры

7 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

7.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

7.2 Порядок защиты

Защита осуществляется комиссии преподавателей кафедры.

7.3 Срок защиты

Срок защиты: 15-16 недели.

8 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ – ИУ6**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

САЙТ ДЛЯ РАЗМЕЩЕНИЯ ЭЛЕКТРОННЫХ ОБЪЯВЛЕНИЙ

Руководство пользователя

Листов 5

Студент _гр. ИУ6-51Б__ _____ ___Д.Ю. Воронин__
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы,
(Кандидат технических наук, доцент) _____ ____Д.А. Миков____
(Подпись, дата) (И.О. Фамилия)

Москва, 2025

1. Назначение программы

Веб-приложение «Омега» предназначено для размещения, поиска и просмотра объявлений о продаже товаров и предоставлении услуг. Оно позволяет:

- просматривать объявления;
- создавать объявления;
- редактировать свои объявления;
- удалять свои объявления;
- искать объявления по заголовку или категориям.

2. Начало работы

2.1 Регистрация нового аккаунта:

- перейдите на главную страницу приложения;
- нажмите кнопку «Регистрация»;
- заполните поля: электронная почта (логин), имя пользователя, номер телефона, пароль;
- нажмите кнопку «Зарегистрироваться».

2.2 Вход в существующий аккаунт:

- перейдите на главную страницу приложения;
- нажмите кнопку «Вход»;
- заполните поля: электронная почта (логин) и пароль;
- нажмите кнопку «Войти».

3. Основной функционал

3.1 Просмотр и поиск объявлений:

- все пользователи могут просматривать список объявлений на главной странице;
- для поиска по названию введите часть заголовка в поле «Поиск» и нажмите клавишу Enter или на кнопку «Поиск»;

3.2 Фильтрация категорий объявлений

Для поиска по тематике воспользуйтесь фильтром «Категория»:

- выберите основную категорию;

- при необходимости уточните выбор подкатегорией;
- система отобразит только те объявления, которые относятся к выбранной категории и подкатегории.

3.3 Создание объявления:

- войдите в систему;
- нажмите кнопку «Создать объявление»;
- заполните поля: название и описание;
- выберите категорию объявления из предложенных;
- при необходимости загрузите до 7 изображений (поддерживаются форматы JPEG, JPG, PNG и WEBP, размер файла — до 10 МБ);
- нажмите кнопку «Создать»;
- после публикации вы будете автоматически перенаправлены на страницу созданного объявления.

3.4 Редактирование и удаление объявления:

- перейдите в раздел «Мои объявления» (доступен только авторизованным пользователям);
- найдите нужное объявление и нажмите «Редактировать» или «Удалить»;
- при редактировании объявления можно изменить название, описание, категорию и изображения;
- удаление объявления необратимо: оно будет полностью удалено из базы данных.

4 Работа с изображениями:

- при создании или редактировании объявления нажмите «Выбрать файлы»;
- вы можете загрузить до 7 изображений за один раз;
- неподдерживаемые форматы (например, PDF, EXE) будут отклонены;
- изображения сохраняются на сервере и отображаются в карточке объявления в виде галереи;

5 Управление учётной записью

После входа в систему в правом верхнем углу интерфейса отображается имя текущего пользователя. При нажатии на него появляется выпадающее меню со следующими пунктами:

- профиль – переход на страницу редактирования профиля.
- мои объявления – отображение списка всех объявлений, созданных текущим пользователем, с возможностью редактирования и удаления.
- создать объявление – переход к форме публикации нового объявления.
- выход – завершение сеанса и возврат на главную страницу в роли незарегистрированного пользователя.

5.1 Профиль

Содержит информацию о профиле текущего пользователя с возможностью редактирования. Для изменения:

- заполните поля, которые хотите изменить: email, имя пользователя, телефон, контакт в социальной сети;
- нажмите кнопку «Обновить профиль».

6 Примеры использования

Пример 1. Продажа подержанного ноутбука с контактом @user123 в Telegram

- зарегистрируйтесь в системе;
- создайте объявление с заголовком «Продам MacBook Air 2020»;
- зайдите в профиль, выберите социальную сеть Telegram и укажите контакт для связи @user123;
- загрузите 3 фотографии устройства;
- объявление станет доступно всем пользователям сразу после публикации.

Пример 2. Поиск недвижимости

- перейдите на главную страницу (можно без авторизации);
- выберите «Недвижимость» в категориях;

- просмотрите все актуальные объявления с этим словом в заголовке;
- свяжитесь с продавцом напрямую по указанным контактам.

Листинг 1 – Исходный текст модуля App

```
package app

import (
	"fmt"
	"log"

	"adboard/internal/config"
	"adboard/internal/handlers"
	"adboard/internal/middleware"
	"adboard/internal/models"
	"adboard/internal/repository"
	"adboard/internal/service"

	"github.com/labstack/echo/v4"
	echomiddleware "github.com/labstack/echo/v4/middleware"
	"gorm.io/driver/postgres"
	"gorm.io/gorm"
)

type App struct {
	cfg  *config.Config
	echo *echo.Echo
	db   *gorm.DB
}

func NewApp(cfg *config.Config) *App {
	return &App{cfg: cfg}
}

func (a *App) InitDB() error {
	log.Printf("Attempting to connect to database with config:
host=%s port=%d user=%s dbname=%s",
		a.cfg.DBHost, a.cfg.DBPort, a.cfg.DBUser,
a.cfg.DBName)
```

Продолжение листинга 1

```
	dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s
dbname=%s sslmode=disable",
		a.cfg.DBHost, a.cfg.DBPort, a.cfg.DBUser,
a.cfg.DBPassword, a.cfg.DBName)

	db, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})
	if err != nil {
		log.Printf("Failed to connect to database: %v", err)
		return fmt.Errorf("failed to connect to
database: %w", err)
	}
	a.db = db
	log.Println("Successfully connected to database")
	return nil
}

func (a *App) InitializeData() error {
	// Только проверяем наличие категорий, миграции
выполняются отдельно
	err := a.checkCategoriesExist()
	if err != nil {
		return fmt.Errorf("failed to check categories: %w",
err)
	}
	return nil
}

func (a *App) InitEcho() {
	a.echo = echo.New()
	a.echo.Use(echomiddleware.Logger())
	a.echo.Use(echomiddleware.Recover())
	a.echo.Use(echomiddleware.CORS())
```

Продолжение листинга 1

```
	a.echo.Static("/images", a.cfg.UploadDir)
	a.SetupValidator()
}

func (a *App) RegisterRoutes() {
	authRepo := repository.NewAuthRepository(a.db)
	userRepo := repository.NewUserRepository(a.db)
	categoryRepo := repository.NewCategoryRepository(a.db)
	adRepo := repository.NewAdRepository(a.db)
	imageRepo := repository.NewImageRepository(a.db)

	authService := service.NewAuthService(authRepo, a.cfg)
	userService := service.NewUserService(userRepo, authRepo)
	categoryService :=
service.NewCategoryService(categoryRepo)
	adService := service.NewAdService(adRepo)
	imageService := service.NewImageService(imageRepo, adRepo)

	authHandler := handlers.NewAuthHandler(authService)
	userHandler := handlers.NewUserHandler(userService)
	categoryHandler :=
handlers.NewCategoryHandler(categoryService)
	adHandler := handlers.NewAdHandler(adService)
	imageHandler := handlers.NewImageHandler(imageService)

	a.echo.POST("/register", authHandler.Register)
	a.echo.POST("/login", authHandler.Login)
	a.echo.GET("/categories", categoryHandler.GetCategories)
	a.echo.GET("/ads", adHandler.SearchAds)
	a.echo.GET("/ads/:id", adHandler.GetAd)

	protected := a.echo.Group("")
	protected.Use(middleware.JWTAuth(a.cfg))
```

Продолжение листинга 1

```
	protected.GET("/profile", userHandler.GetProfile)
	protected.PUT("/profile", userHandler.UpdateProfile)

	protected.POST("/ads", adHandler.CreateAd)
	protected.POST("/ads/with-images",
adHandler.CreateAdWithImages, middleware.UploadFiles)
	protected.GET("/my-ads", adHandler.GetUserAds)
	protected.PUT("/ads/:id", adHandler.UpdateAd)
	protected.DELETE("/ads/:id", adHandler.DeleteAd)

	protected.POST("/ads/:id/images", imageHandler.AddImages,
middleware.UploadFiles)
	protected.DELETE("/images/:imageId",
imageHandler.DeleteImage)
}

func (a *App) Start() error {
	addr := fmt.Sprintf(":%s", a.cfg.ServerPort)
	log.Printf("Server starting on %s", addr)
	return a.echo.Start(addr)
}

func (a *App) checkCategoriesExist() error {
	// Проверяем, есть ли уже категории
	var count int64
	err := a.db.Model(&models.Category{}).Count(&count).Error
	if err != nil {
		return err
	}

	// Если категорий нет, то предупреждаем (не ошибка, так
как это может быть первый запуск)
```

Продолжение листинга 1

```
	if count == 0 {
		log.Println("Внимание: Категории не найдены.
Убедитесь, что выполнены все миграции: make db-up")
	}

	return nil
}
```

Ссылка на местоположение полного комплекта исходных модулей программы – https://github.com/cod1ng-space/avito-clone