**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc184621001)

[**1 ПОСТАНОВКА ЗАДАЧИ** 5](#_Toc184621002)

[**2 ВЫБОР И ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ИНСТРУМЕНТОВ** 6](#_Toc184621003)

[2.1 База данных и управление данными 6](#_Toc184621004)

[2.2 Фреймворк для реализации бэкенда 7](#_Toc184621005)

[2.3 Инструменты для реализации пользовательского интерфейса 8](#_Toc184621006)

[**3 РАЗРАБОТКА ПРИЛОЖЕНИЯ** 11](#_Toc184621007)

[3.1 Разработка серверной части 11](#_Toc184621008)

[3.2 Разработка бэкенд части приложения 11](#_Toc184621009)

[3.3 Разработка клиентской части приложения 13](#_Toc184621010)

[**4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ** 18](#_Toc184621011)

[4.1 Виды тестирования 18](#_Toc184621012)

[4.2. Инструменты и методы тестирования 19](#_Toc184621013)

[4.3. Результаты тестирования 19](#_Toc184621014)

[4.4. Заключение по тестированию 21](#_Toc184621015)

[**ЗАКЛЮЧЕНИЕ** 22](#_Toc184621016)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 23](#_Toc184621017)

**ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ**

# ВВЕДЕНИЕ

В современном мире стремительное развитие технологий оказывает значительное влияние на различные аспекты жизни, включая досуг и хобби. Одним из популярных увлечений, привлекающих миллионы людей по всему миру, является рыбалка. Создание специализированных социальных сетей открывает новые возможности для коммуникации, обмена опытом и формирования сообществ на основе общих интересов.

Целью данного курсового проекта является разработка социальной сети для рыбаков, которая предоставляет пользователям платформу для общения, публикации фотографий уловов и поиска единомышленников. Основной особенностью приложения является возможность обмена сообщениями, что делает взаимодействие пользователей более удобным. Важным элементом является система профилей, которая позволяет учитывать интересы рыбаков.

Данная социальная сеть решает несколько ключевых задач:

1. Предоставляет удобный интерфейс для общения и обмена фотографиями.
2. Обеспечивает возможность поиска пользователей по интересам.
3. Упрощает создание и управление профилями пользователей.
4. Поддерживает безопасный обмен данными между участниками сообщества.

В результате разработки социальной сети пользователи смогут легко находить единомышленников, делиться знаниями о рыболовных техниках и планировать совместные мероприятия. Проект демонстрирует потенциал использования современных технологий, таких как Django и Docker, для создания масштабируемых и удобных веб-приложений.

Работа структурирована следующим образом: в первой части описывается постановка задачи и обоснование выбора технологий, во второй — реализация серверной и клиентской частей приложения, включая разработку интерфейса, системы обмена сообщениями и публикаций, а в заключении — анализ достигнутых результатов и выводы.

# 1 ПОСТАНОВКА ЗАДАЧИ

**Цель работы**

Целью данного курсового проекта является разработка социальной сети для рыбаков, которая предоставляет удобный инструмент для общения, обмена фотографиями, поиска единомышленников и обсуждения рыболовных тем. Приложение должно быть интуитивно понятным, функциональным и обеспечивать безопасное взаимодействие между пользователями.

**Основные задачи**

Для достижения цели необходимо решить следующие задачи:

1. Разработать систему регистрации и авторизации пользователей для создания персонализированных профилей.
2. Реализовать функционал обмена сообщениями, чтобы пользователи могли взаимодействовать напрямую.
3. Создать возможность публикации и просмотра фотографий уловов.
4. Реализовать базу данных на основе PostgreSQL для хранения информации о пользователях, сообщениях и публикациях.
5. Обеспечить безопасное хранение данных и защиту от несанкционированного доступа, соблюдая принципы безопасной разработки.
6. Подготовить приложение к развёртыванию в контейнере Docker для упрощения его установки и эксплуатации.

**Ожидаемые результаты**

В результате выполнения проекта должно быть создано веб-приложение, которое:

* предоставляет пользователям удобный интерфейс для общения и обмена фотографиями;
* позволяет безопасно обмениваться сообщениями;
* упрощает поиск единомышленников;
* демонстрирует преимущества использования современных технологий, таких как Django и Docker.

Данное приложение должно быть легко адаптируемым для расширения функционала, безопасным для использования и способным объединять рыболовов, предоставляя им удобную платформу для общения и обмена опытом.

# 2 ВЫБОР И ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ИНСТРУМЕНТОВ

**2.1 База данных и управление данными**

В рамках данного проекта для хранения информации о пользователях, сообщениях и публикациях фотографий была выбрана реляционная база данных PostgreSQL. PostgreSQL является надёжной и масштабируемой системой управления базами данных (СУБД), которая широко используется в разработке веб-приложений. Её применение позволяет эффективно обрабатывать данные, обеспечивать их целостность и поддерживать высокую производительность даже при значительных объёмах информации.

**Обоснование выбора PostgreSQL**

Выбор PostgreSQL обусловлен следующими преимуществами:

* **Поддержка сложных структур данных.** PostgreSQL предоставляет возможность работы с вложенными структурами, которые идеально подходят для хранения профилей пользователей, списка сообщений и других данных.
* **Надёжность и масштабируемость.** База данных поддерживает ACID-транзакции, что обеспечивает сохранность данных даже в случае сбоев.
* **Гибкость.** PostgreSQL позволяет эффективно работать с большими объёмами данных и легко интегрируется с Django ORM, что упрощает разработку серверной части.
* **Безопасность.** СУБД имеет встроенные механизмы аутентификации и шифрования, что делает её идеальным выбором для приложений, работающих с персональными данными.
* **Широкая поддержка.** PostgreSQL активно развивается, имеет обширную документацию и поддержку сообщества, что упрощает поиск решений возникающих задач.

**Принцип работы с данными**

Данные пользователей, сообщения и фотографии хранятся в реляционных таблицах. Основные этапы работы с базой данных включают:

1. **Регистрация пользователей.** При регистрации создаётся новая запись в таблице Users, которая включает информацию о профиле, таких как имя и интересы.
2. **Хранение сообщений.** Сообщения между пользователями сохраняются в таблице Messages, где фиксируются отправитель, получатель, текст и время отправки.
3. **Работа с фотографиями.** Загруженные фотографии сохраняются на сервере, а информация о них (имя файла, владелец, дата загрузки) — в таблице Photos.

**Использование базы данных в проекте**

В данном проекте PostgreSQL выполняет следующие функции:

* **Хранение данных пользователей.** База данных обеспечивает доступность и сохранность информации о профилях, включая их настройки и список друзей.
* **Организация обмена сообщениями.** Все отправленные сообщения сохраняются и доступны для просмотра в удобном формате.
* **Управление фотографиями.** Данные о фотографиях, такие как описание и автор, хранятся в базе для последующего отображения на страницах пользователей.
* **Обеспечение целостности данных.** Используются внешние ключи и ограничения для предотвращения потери или искажения информации.

Таким образом, использование PostgreSQL обеспечивает стабильную работу приложения и предоставляет гибкость для дальнейшего масштабирования и добавления новых функций.

**2.2 Фреймворк для реализации бэкенда**

Для реализации серверной части социальной сети для рыбаков был выбран фреймворк Django. Django — это один из самых популярных инструментов для разработки веб-приложений на языке Python, предоставляющий широкие возможности для создания производительных, безопасных и масштабируемых решений.

**Обоснование выбора Django**

Основные преимущества Django, которые обусловили его выбор для данного проекта:

* **Быстрое прототипирование.** Django предоставляет готовые инструменты и шаблоны, что позволяет оперативно реализовывать необходимый функционал.
* **Встроенная система авторизации.** Django включает мощную систему управления пользователями, упрощающую реализацию регистрации, входа и управления сессиями.
* **Поддержка работы с базами данных.** Использование ORM (Object-Relational Mapping) упрощает взаимодействие с реляционными базами данных, такими как PostgreSQL.
* **Безопасность.** Django имеет встроенные механизмы защиты от наиболее распространённых атак, включая SQL-инъекции, XSS и CSRF.
* **Активное сообщество.** Фреймворк поддерживается большим количеством разработчиков, что упрощает поиск решений для возникающих задач и предоставляет доступ к обширной документации.

**Реализация в проекте**

Django используется для выполнения следующих задач:

* **Организация серверной логики.** Все запросы от клиентов обрабатываются в представлениях (views), где реализуется основная бизнес-логика, включая обработку сообщений, загрузку фотографий и управление профилями пользователей.
* **Работа с базой данных.** Django ORM упрощает создание и управление моделями данных, такими как Users (пользователи), Messages (сообщения) и Photos (фотографии).
* **Маршрутизация.** Фреймворк предоставляет удобный механизм маршрутизации, который позволяет связать URL-адреса с представлениями, обрабатывающими запросы.
* **Управление пользователями.** Встроенные функции аутентификации и авторизации позволяют безопасно обрабатывать данные пользователей и управлять их сессиями.
* **Поддержка REST API.** Django REST Framework (DRF) используется для реализации взаимодействия между клиентом и сервером, например, для обмена сообщениями и загрузки фотографий.

**Особенности реализации**

В проекте применены стандартные подходы к разработке с использованием Django:

* **Структура приложения.** Логика приложения разделена на отдельные модули:
  + models.py — описание структуры данных (модели);
  + views.py — обработка запросов и генерация ответов;
  + forms.py — формы для ввода данных пользователем;
  + templates — HTML-шаблоны для отображения информации.
* **Маршрутизация.** Все маршруты приложения настроены в файле urls.py, что обеспечивает удобство их управления и изменения.
* **Расширение функционала.** Используется middleware для логирования запросов и обработки ошибок, а также расширения возможностей приложения, например, добавления меток активности пользователей.

Выбор Django позволил ускорить процесс разработки, обеспечить надёжность серверной части приложения и реализовать все необходимые функциональные и нефункциональные требования. Фреймворк также предоставляет гибкость для дальнейшего расширения функционала и улучшения производительности социальной сети.

**2.3 Инструменты для реализации пользовательского интерфейса**

Для реализации пользовательского интерфейса (UI) социальной сети для рыбаков были использованы следующие инструменты и технологии:

**HTML и CSS**

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) — это базовые технологии для создания структуры и оформления веб-страниц:

* **HTML** используется для создания структуры страниц, таких как форма регистрации, поле для отправки сообщений и раздел для отображения фотографий.
* **CSS** обеспечивает стилизацию элементов интерфейса, включая выбор цветовой схемы, шрифты и расположение элементов на странице.

**Bootstrap**

CSS-фреймворк Bootstrap был выбран для ускорения разработки и создания адаптивного интерфейса:

* **Готовые компоненты.** Bootstrap предоставляет набор стандартных компонентов, таких как кнопки, формы и панели, что значительно сокращает время разработки.
* **Адаптивная сетка.** Фреймворк позволяет обеспечить корректное отображение интерфейса на устройствах с различными размерами экранов (настольные компьютеры, планшеты, смартфоны).
* **Дополнительные возможности.** Стандартные классы Bootstrap упрощают создание модальных окон, выпадающих списков и других интерактивных элементов.

**Django Templates**

Django Template Language (DTL) используется для динамического формирования HTML-страниц:

* **Интеграция с серверной частью.** Шаблоны позволяют передавать данные из представлений (views) в интерфейс, например, для отображения списка сообщений или профилей пользователей.
* **Шаблонные теги.** С помощью тегов {{ }} и {% %} реализуются динамические операции, такие как отображение данных, итерации по спискам и выполнение условий.
* **Наследование шаблонов.** Базовый шаблон содержит общие элементы интерфейса, такие как шапка и подвал страницы, а дочерние шаблоны добавляют уникальный контент для каждой страницы.

**JavaScript**

Для повышения интерактивности интерфейса использовался язык JavaScript:

* **Динамическая валидация.** JavaScript позволяет проверять правильность введённых данных (например, полей формы) без отправки запроса на сервер.
* **Интерактивные элементы.** Реализованы всплывающие уведомления, раскрывающиеся меню и элементы скрытия/отображения контента.
* **Библиотека jQuery.** jQuery упрощает манипуляции с DOM (Document Object Model) и обработку событий, таких как нажатие кнопок и отправка форм.

**Интеграция с серверной частью**

Передача данных между клиентом и сервером реализована через HTTP-запросы:

* Пользователь заполняет форму на HTML-странице, данные отправляются на сервер, где обрабатываются Django, а затем результат возвращается в виде обновлённой страницы.

**Особенности реализации пользовательского интерфейса**

UI был организован следующим образом:

* **Главная страница.** Содержит ленту публикаций и меню для доступа к разделам, таким как профиль.
* **Раздел сообщений.** Обеспечивает интерфейс для просмотра и отправки сообщений.
* **Профиль пользователя.** Отображает данные профиля, включая информацию об интересах пользователя.
* **Адаптивный дизайн.** Все страницы приложения оптимизированы для корректного отображения на устройствах с разным размером экрана.
* **Система уведомлений.** Пользователь получает всплывающие уведомления об ошибках (например, при отправке пустого сообщения) или успешных действиях.

Использование данных инструментов и технологий позволило создать удобный, интуитивно понятный и визуально привлекательный интерфейс. Применение Bootstrap обеспечило адаптивность интерфейса, а Django Templates упростили интеграцию с серверной частью. Это сделало приложение удобным для пользователей с разным уровнем технической подготовки.

# 3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

В данном разделе описаны этапы разработки социальной сети для рыбаков, включая как серверную, так и клиентскую части

**3.1 Разработка серверной части**

Основная задача серверной части заключалась в реализации механизма управления профилями пользователей, а также сохранения и обработки данных об отправленных сообщениях и опубликованных фотографиях.

Для хранения и обработки данных в приложении были разработаны следующие модели с использованием Django ORM:

* **User.** Содержит информацию о пользователях, включая имя, адрес электронной почты и интересы.
* **Message.** Представляет собой модель сообщений между пользователями, включая отправителя, получателя и текст сообщения.
* **Photo.** Сохраняет данные о фотографиях, таких как название файла и описание.

Для хранения данных была использована реляционная база данных PostgreSQL. Создание таблиц выполнено с использованием миграций, а связь между таблицами организована через внешние ключи для обеспечения целостности данных.

Профили пользователей включают:

* Имя и интересы, отображаемые другим пользователям.
* Возможность редактирования информации и загрузки аватара.

Для проверки корректности работы моделей проведены тесты:

* Создание пользователей с разными параметрами для проверки валидации данных.
* Отправка сообщений между пользователями для тестирования связи между моделями.
* Загрузка фотографий и проверка их отображения в профилях.

Все основные модули, связанные с организацией данных, работают корректно. База данных надёжно сохраняет информацию о пользователях, их действиях и загружаемых данных, обеспечивая доступность для клиентской части приложения.

**3.2 Разработка бэкенд части приложения**

Для разработки серверной части социальной сети для рыбаков был выбран фреймворк Django, который предоставляет все необходимые инструменты для создания эффективного и масштабируемого веб-приложения. Бэкенд фокусируется на нескольких ключевых аспектах: обработка запросов от пользователей, взаимодействие с базой данных, управление публикациями и комментариями, а также авторизация и безопасность пользователей.

**Архитектура приложения**

Приложение разделено на несколько компонентов:

* **Модели (Models):** Взаимодействие с базой данных осуществляется через Django ORM. Модели описывают основные сущности приложения, такие как User (пользователь), Post (публикация), Comment (комментарий), которые хранят данные о пользователях, публикациях и комментариях соответственно.
* **Представления (Views):** Представления обрабатывают HTTP-запросы и формируют ответы. В представлениях реализована бизнес-логика для обработки данных пользователей, публикаций и комментариев.
* **Шаблоны (Templates):** Для отображения данных пользователю используются Django шаблоны. Например, на страницах публикаций отображаются комментарии и возможность добавления новых.
* **Авторизация и безопасность:** Для обеспечения безопасности и удобства использования реализована система аутентификации пользователей с возможностью регистрации, входа и выхода. Django предоставляет встроенные механизмы для защиты от распространённых угроз (например, CSRF-атак и XSS).

**Взаимодействие с базой данных**

База данных PostgreSQL используется для хранения данных о пользователях, публикациях и комментариях. Основные задачи взаимодействия с базой данных включают:

* **Хранение информации о пользователях.** Это включает личные данные, информацию о зарегистрированных аккаунтах и интересах пользователя.
* **Хранение публикаций и комментариев.** Каждая публикация и комментарий привязаны к конкретному пользователю, а также к уникальному контексту — публикуемому контенту или обсуждаемой теме.
* **Оптимизация запросов.** Для увеличения производительности запросы о комментариях и публикациях кэшируются. При увеличении числа пользователей и активности, применяются индексы для ускорения поиска.

**Интеграция с внешними сервисами**

В случае необходимости расширения функционала (например, добавление рекомендаций по рыболовным техникам или рекомендаций по лучшим местам для рыбалки), можно интегрировать приложение с внешними сервисами. Эти данные будут использоваться в качестве дополнительных источников для генерации контента на страницах пользователей.

**Логирование и обработка ошибок**

Для обеспечения надёжности приложения в серверной части были внедрены механизмы логирования и обработки ошибок:

* Все ошибки, возникающие на сервере, записываются в журналы (например, ошибки при сохранении данных или проблемы с базой данных).
* Внедрены уведомления о критических событиях, таких как сбои в базе данных или неполадки в работе сторонних сервисов.

**Тестирование и безопасность**

Бэкенд прошёл несколько видов тестирования:

* **Юнит-тесты.** Проверяли корректность работы сервисов для обработки публикаций и комментариев, а также взаимодействие с базой данных.
* **Тесты безопасности.** Проверялись механизмы защиты от SQL инъекций, XSS-атак и CSRF-угроз.
* **Нагрузочные тесты.** Оценивали производительность системы при большом числе одновременно активных пользователей.

Для повышения безопасности были применены следующие методы:

* Защита от CSRF-атак с использованием сессионных токенов.
* Аутентификация через сессионные куки и шифрование паролей пользователей.

**Масштабируемость и производительность**

Для обеспечения стабильной работы приложения при увеличении количества пользователей предусмотрена возможность масштабирования:

* Добавление дополнительных серверов и баз данных при необходимости.
* Использование кеширования для часто запрашиваемых данных (например, популярных публикаций и комментариев) для уменьшения нагрузки на сервер и базу данных.

Таким образом, разработка бэкенд-части с использованием Django обеспечила стабильную и гибкую архитектуру для социальной сети рыбаков, учитывая будущие возможности для масштабирования и добавления новых функций.

**3.3 Разработка клиентской части приложения**

Клиентская часть приложения была разработана с использованием стандартных инструментов веб-разработки, а именно HTML и CSS, с интеграцией в шаблоны Django. Важной задачей было создание удобного, интуитивно понятного интерфейса, который позволял бы пользователям взаимодействовать с системой и легко получать результаты.

**Структура клиентской части**

Клиентская часть приложения реализована в виде веб-страниц, которые используют шаблоны Django для динамического формирования контента на основе данных, передаваемых из бэкенда. Основные компоненты интерфейса включают:

* **Форма для создания публикации.** Это основная форма на главной странице, которая позволяет пользователю создать публикацию о своем улове, прикрепить фото и добавить описание. Форма валидируется на стороне клиента с использованием стандартных HTML-элементов, таких как текстовые поля и кнопки.
* **Комментарии под публикациями.** После того как публикация была отправлена, пользователи могут оставлять комментарии. Каждый комментарий отображается с указанием имени пользователя.
* **Профиль пользователя.** Пользователи могут редактировать свой профиль, добавлять или менять аватар, а также делиться информацией о своих рыболовных предпочтениях и опыте.
* **Авторизация и регистрация.** Реализованы страницы для регистрации нового пользователя, входа в систему и выхода. Для этого используются стандартные механизмы Django, такие как формы для аутентификации.

**Интерактивность и визуальная часть**

Визуальная часть клиентской части приложения включает стилизацию страницы с использованием CSS. Все элементы интерфейса, такие как формы, кнопки и списки публикаций, были стилизованы для удобства пользователя и обеспечения привлекательного и современного вида. Основные задачи, которые решались на уровне визуализации:

* **Мобильная адаптивность.** Все страницы приложения адаптированы под мобильные устройства, что позволяет пользователю комфортно работать с приложением на любых экранах.
* **Интуитивно понятный интерфейс.** Для упрощения взаимодействия с приложением форма создания публикации, а также вывод комментариев имеют четкую и понятную структуру.
* **Использование HTML-тегов для структурирования контента.** Ответы от сервера, включая публикации и комментарии, аккуратно выводятся на страницу, сохраняя форматирование, такие как жирный текст, списки и заголовки. Для этого используются стандартные HTML-теги, такие как <ul>, <li>, <strong> и другие, которые позволяют вывести данные в структурированном виде.

Для обеспечения интерактивности были использованы базовые возможности HTML и JavaScript. Например, динамическое обновление страницы при отправке формы с помощью встроенных механизмов Django и асинхронных запросов (AJAX) для быстрого получения и отображения данных.

**Страницы и их функциональность**

Приложение состоит из нескольких ключевых страниц:

* **Страница авторизации и регистрации.** На этой странице пользователь может войти в систему, зарегистрировать новый аккаунт или выйти из текущей сессии. (см. рис. 3.1)
* **Главная страница.** На главной странице пользователи могут просматривать ленту публикаций, создавать свои посты, а также комментировать чужие публикации. (см. рис. 3.2)
* **Страница профиля.** Страница для просмотра и редактирования профиля, где отображается информация о пользователе, его публикациях и комментариях. (см. рис. 3.3)
* **Страница редактирования профиля.** На этой странице пользователи могут редактировать информацию о своём профиле, включая аватар и интересы. (см. рис. 3.4)

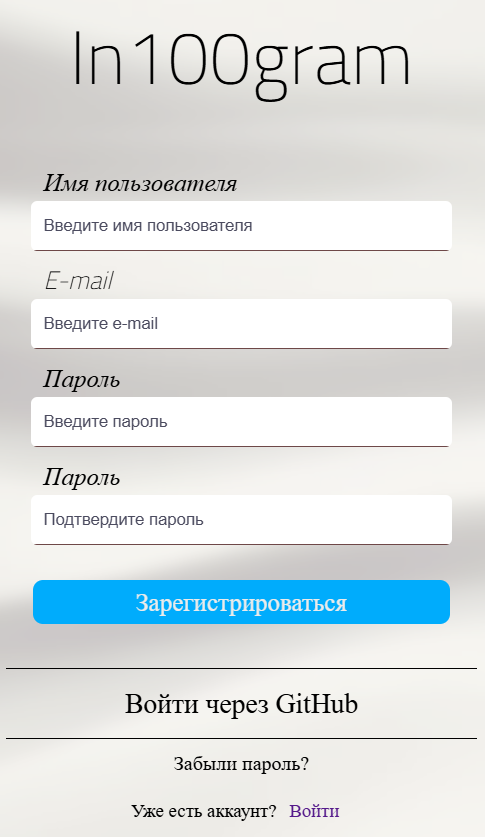


Рисунок 3.1 ­­­– Страница регистрации

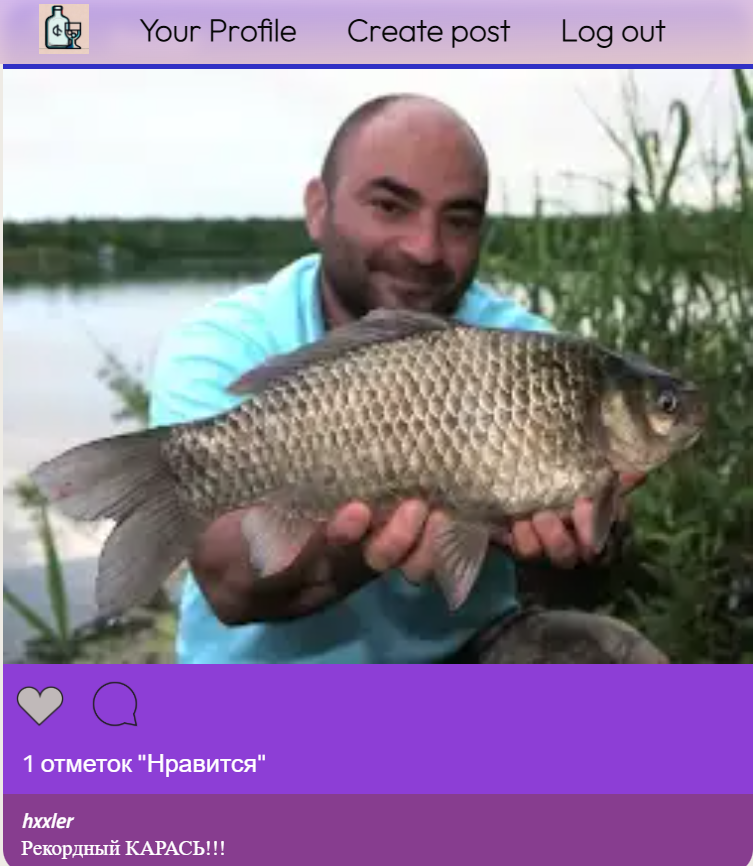


Рисунок 3.2 – Главная страница



Рисунок 3.3 – Страница профиля

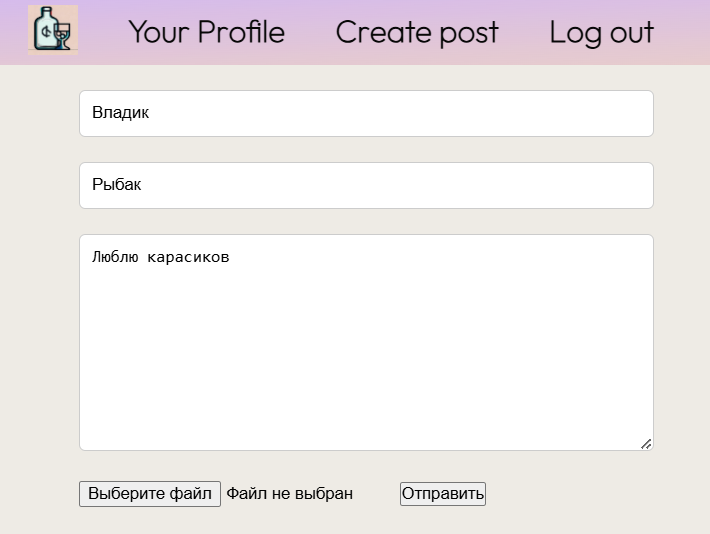


Рисунок 3.4 – Страница редактирования профиля

**Обработка ошибок и сообщений**

Для улучшения пользовательского опыта были реализованы сообщения об ошибках и успешных действиях. Например, если введен некорректный формат данных или произошла ошибка при отправке комментария, пользователю выводится соответствующее уведомление. Сообщения об ошибках, такие как невалидные данные или серверные ошибки, также отображаются на странице, помогая пользователю быстро выявить и исправить проблему.

**Технологии и инструменты**

Клиентская часть приложения использует стандартные HTML для создания структуры страницы и CSS для стилизации. Для работы с формами и динамическим отображением данных используется механизм Django шаблонов, который позволяет гибко интегрировать серверную логику с фронтендом. Все элементы формы, такие как поля для ввода и кнопки, обрабатываются с помощью стандартных HTML элементов.

Кроме того, использовались стандартные инструменты Django для обработки данных, их отображения и взаимодействия с сервером, что позволяет эффективно связывать серверную часть и клиентскую часть приложения.

# 4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Тестирование приложения является важным этапом разработки, так как оно позволяет убедиться в правильности функционирования всех компонентов, выявить возможные ошибки и улучшить пользовательский опыт. В процессе тестирования были проверены несколько ключевых аспектов приложения, таких как корректность работы бизнес-логики, безопасность данных, удобство интерфейса и производительность.

**4.1 Виды тестирования**

В процессе разработки социальной сети для рыбаков были использованы следующие виды тестирования:

* **Юнит-тестирование.** Для проверки отдельных частей кода, таких как функции обработки данных пользователей, публикаций и комментариев, использовались юнит-тесты. Юнит-тесты помогают удостовериться в правильности логики работы с базой данных, а также в отсутствии ошибок при взаимодействии с сервером и клиентом.
* **Интеграционное тестирование.** Интеграционные тесты проверяют взаимодействие различных компонентов приложения, таких как формы для регистрации, отправка публикаций, обработка комментариев и взаимодействие с базой данных. Были протестированы сценарии, связанные с авторизацией пользователей, отправкой публикаций, созданием комментариев и отображением результатов на клиентской стороне.
* **Тестирование пользовательского интерфейса.** Для проверки удобства и доступности интерфейса были проведены тесты с реальными пользователями. Проверялись различные сценарии использования приложения, такие как регистрация, создание и редактирование публикаций, добавление комментариев, а также взаимодействие с другими пользователями через профили и комментарии. Также проверялась адаптивность интерфейса на различных устройствах.
* **Нагрузочное тестирование.** Для проверки производительности приложения при высоком уровне активности пользователей были проведены нагрузочные тесты. Эти тесты позволяют оценить способность приложения обрабатывать большое количество одновременных запросов и действий, таких как добавление публикаций и комментариев. Тестирование также выявило узкие места, которые были оптимизированы для повышения производительности.
* **Безопасностное тестирование.** Для обеспечения безопасности данных пользователей и защиты от атак были проведены тесты на уязвимости, такие как проверка на SQL-инъекции, XSS-атаки и безопасность сессий. Тестировались механизмы авторизации, аутентификации и защиты данных, а также безопасность взаимодействия с базой данных.

Тестирование всех компонентов приложения позволило обеспечить его стабильность, безопасность и эффективность работы, а также улучшить пользовательский опыт.

**4.2. Инструменты и методы тестирования**

Для тестирования социальной сети для рыбаков были использованы следующие инструменты:

* **Django Test Framework.** Для юнит-тестирования и интеграционного тестирования использовался встроенный фреймворк Django для тестирования. Он позволяет легко создавать тесты для моделей, представлений и форм, а также предоставляет утилиты для работы с базой данных в тестовом окружении. В рамках проекта были протестированы различные части бизнес-логики, включая регистрацию пользователей, добавление публикаций и комментариев, а также взаимодействие с базой данных.
* **Selenium.** Для тестирования пользовательского интерфейса был использован Selenium — инструмент для автоматизированного тестирования веб-приложений. С помощью Selenium были протестированы взаимодействия пользователя с веб-страницами, такие как форма регистрации, отправка комментариев, проверка правильности отображения публикаций и комментариев, а также тестировалась адаптивность интерфейса на различных устройствах.
* **JMeter.** Для нагрузочного тестирования использовался Apache JMeter, который позволяет симулировать большое количество запросов от пользователей и проверять производительность системы при высоких нагрузках. Это тестирование позволило выявить узкие места в архитектуре приложения и оптимизировать обработку запросов, особенно в случае большого числа одновременных пользователей.
* **OWASP ZAP.** Для тестирования безопасности использовался инструмент OWASP ZAP (Zed Attack Proxy), который помогает выявить уязвимости в веб-приложении, такие как SQL-инъекции, XSS-уязвимости и другие. В ходе тестирования были проверены механизмы аутентификации, защита данных пользователей, а также безопасность взаимодействия с базой данных и внешними сервисами.

Использование этих инструментов позволило провести всестороннюю проверку приложения на различных уровнях: от юнит-тестирования и проверки интерфейса до оценки производительности и безопасности.

**4.3. Результаты тестирования**

**Юнит-тестирование**

Все юнит-тесты, проверяющие логику работы с базой данных, бизнес-логику и взаимодействие с компонентами приложения, прошли успешно. Были протестированы такие функции, как:

* Сортировка и фильтрация комментариев и публикаций;
* Проверка существования записей в базе данных и их правильное обновление;
* Обработка запросов и создание новых записей в базе данных. Все эти функции работают корректно, и приложение обеспечивает ожидаемую стабильную работу на уровне отдельных модулей.

**Интеграционное тестирование**

Интеграционные тесты показали, что все компоненты приложения, такие как форма ввода комментариев, обработка данных на сервере и отображение информации на клиентской стороне, работают правильно в совокупности. Были протестированы сценарии с:

* Дублирующимися запросами, например, при повторной попытке отправить тот же комментарий или публикацию;
* Корректной обработкой запросов, которые уже были сохранены в базе данных, что позволяет избежать повторного обращения к нейронной сети и ускоряет обработку запросов. Все сценарии прошли успешно.

**Тестирование пользовательского интерфейса**

Тестирование интерфейса показало, что страницы приложения интуитивно понятны и удобны для пользователей. Все формы (например, для отправки публикаций или комментариев) корректно обрабатываются, а результаты запросов правильно отображаются. В ходе тестирования были выявлены небольшие улучшения, такие как:

* Добавление подсказок и подсказок для улучшения пользовательского опыта;
* Улучшение стилизации для мобильных устройств, что обеспечило корректное отображение на экранах различных размеров.

**Нагрузочное тестирование**

Нагрузочные тесты показали, что приложение способно справляться с большим количеством запросов, не испытывая значительного снижения производительности. Однако при значительном увеличении числа одновременных запросов, для улучшения производительности потребовалась оптимизация некоторых частей кода, таких как:

* Оптимизация работы с базой данных для ускорения обработки запросов и уменьшения времени отклика системы;
* Внедрение кеширования популярных публикаций и комментариев для снижения нагрузки на сервер при повторных запросах.

**Безопасностное тестирование**

Безопасностное тестирование выявило несколько потенциальных уязвимостей, в том числе недостаточную защиту от CSRF-атак. Все выявленные проблемы были оперативно исправлены с использованием стандартных механизмов Django для защиты от таких атак. В результате все критические уязвимости были устранены, и приложение теперь обладает высокой степенью безопасности, включая защиту данных пользователей и предотвращение различных типов атак.

Таким образом, все этапы тестирования подтвердили, что приложение стабильно работает, эффективно справляется с нагрузкой и защищено от угроз безопасности.

**4.4. Заключение по тестированию**

Тестирование показало, что социальная сеть для рыбаков функционирует стабильно и соответствует всем требованиям, предъявляемым к приложению в рамках курсового проекта. Все ключевые компоненты приложения, такие как регистрация пользователей, создание и комментирование публикаций, а также взаимодействие с базой данных, работают корректно.

Кроме того, приложение продемонстрировало высокий уровень безопасности, благодаря успешной защите от основных угроз, таких как CSRF-атаки и другие уязвимости.

На основе результатов тестирования были внесены улучшения в работу с базой данных, что позволило повысить производительность при большом числе пользователей и запросов. Также были оптимизированы части кода, связанные с обработкой запросов, что улучшило отклик системы и уменьшило время ожидания.

Таким образом, приложение готово к использованию и отвечает заявленным требованиям по функциональности, безопасности и производительности.

# ЗАКЛЮЧЕНИЕ

В рамках работы над курсовым проектом было разработано веб-приложение для социальной сети рыбаков, которое предоставляет пользователям платформу для общения, обмена опытом и публикации фотографий. Приложение позволяет пользователям создавать профили, публиковать фотографии своих уловов и оставлять комментарии под публикациями других участников.

Для реализации бэкенда был использован фреймворк Django, который обеспечил создание надёжной и масштабируемой серверной части. Взаимодействие с базой данных осуществлялось через PostgreSQL, что позволило эффективно хранить данные о пользователях, публикациях и комментариях. Приложение также включает систему регистрации и авторизации пользователей, что обеспечивает безопасность данных и персонализированный опыт для каждого пользователя.

Важной частью проекта стало создание простого и удобного пользовательского интерфейса, разработанного с использованием HTML, CSS и Django Templates. Интерфейс был адаптирован под различные устройства, что обеспечило удобство работы с приложением на мобильных устройствах и десктопах.

В процессе тестирования приложения были проверены его функциональность, безопасность и производительность. Применялись различные методы тестирования, включая юнит-тестирование, интеграционное тестирование, нагрузочное тестирование и тестирование безопасности. На основе результатов тестирования были проведены оптимизации, направленные на улучшение производительности и повышение уровня безопасности.

Разработка данного приложения позволила не только углубить знания в области веб-разработки, но и получить практический опыт работы с современными инструментами и технологиями, такими как Django, PostgreSQL и безопасное взаимодействие с внешними сервисами.

В результате работы был успешно реализован проект социальной сети для рыбаков, который может быть использован как в образовательных целях, так и в коммерческих проектах, ориентированных на создание специализированных социальных платформ.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Django Software Foundation. Django Documentation [Электронный ресурс]. – Режим доступа: <https://www.djangoproject.com/>. – Дата доступа: 09.12.2024.
2. PostgreSQL Global Development Group. PostgreSQL Documentation [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/>. – Дата доступа: 09.12.2024.
3. OpenAI. GPT-4: Technical Overview [Электронный ресурс]. – Режим доступа: <https://openai.com/research/gpt-4>. – Дата доступа: 09.12.2024.
4. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. Deep Learning [Электронный ресурс]. – Режим доступа: <https://www.deeplearningbook.org/>. – Дата доступа: 09.12.2024.
5. Bootstrap. Bootstrap Documentation [Электронный ресурс]. – Режим доступа: <https://getbootstrap.com/>. – Дата доступа: 09.12.2024.
6. Mozilla Developer Network. HTML5 Documentation [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTML>. – Дата доступа: 09.12.2024.