МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

# РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ С ФУНКЦИОНАЛОМ СОЦСЕТИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Данилина Даниила Ивановича

Научный руководитель
зав. каф., к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

# ВВЕДЕНИЕ

В наше время цифровых технологий социальные сети стали неотъемлемой частью повседневной жизни, предоставляя средства связи, обмена информацией и формирования виртуальных сообществ. Создание веб-приложения с функционалом социальной сети — уникальное исследование, объединяющее технические аспекты разработки с удовлетворением социальных потребностей пользователей. Процесс разработки серверной части предоставляет возможности для инноваций в области обработки данных, бизнес-логики и обеспечения эффективного взаимодействия с клиентской частью.

Цель исследования заключается не только в создании эффективного серверного приложения, но и в понимании того, как выбор методов решения и инструментов может повлиять на качество и функциональность конечного продукта. Наша команда разработчиков, работая в рамках программы «ВКР как стартап», стремится не просто создать социальную сеть, а инновационное веб-приложение, способное удовлетворить разнообразные потребности пользователей и предоставить им новые возможности в сфере виртуального взаимодействия и обмена контентом.

Цель данной дипломной работы — разработка серверной части приложения с функционалом социальной сети, направленного на монетизацию контента через платные подписки и денежных пожертвований, а также обеспечения поддержки начинающих творческих личностей.

Для достижения поставленной цели требуется выполнение следующих задач:

1. создать и сконфигурировать приложение на Ruby on Rails;
2. реализовать безопасную регистрацию и аутентификации пользователей;
3. реализовать базовые функции социальной сети с использованием фреймворка GraphQL, такие как создание и удаление постов, комментариев и т.д;
4. реализовать механику работы личных сообщений;
5. реализовать систему рекомендаций постов.

**Структура и объем работы**. Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 2 основные главы, заключение, список использованных источников из 20 наименований и 4 приложения. Работа изложена на 55 страницах, содержит 6 рисунков.

Первая глава имеет название «Анализ предметной области» и содержит основную информацию о задаче и реализуемом продукте, имеющихся аналогов и доступных на текущий момент технологий разработки серверной части веб-приложения.

Вторая глава имеет название «Реализация работы приложения», данная глава содержит подробное описание процесса выполнения работы.

Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложения с кодом А-Г.

**Основное содержание работы**

**Постановка задачи**. Задача заключается в реализации клиент-серверного приложения. Клиент предоставляет интерфейс пользователя, в то время как сервер обеспечивает обработку запросов и управление данными. Эта архитектура обеспечивает легкость масштабирования и поддерживает разделение обязанностей между клиентом и сервером.

Таким образом, основная цель этого исследования заключается в разработке серверной части приложения с функционалом социальной сети. Мы стремимся к тому, чтобы это приложение было современным в используемых технологиях. Кроме того, наша цель — удовлетворить потребности пользователей, предоставив им удобную и многофункциональную платформу для обмена информацией. И осуществить поддержку начинающих талантов и социально-значимых проектов.

**Описание продукта**. Основной целью нашего проекта является создание возможности для творческих личностей в России зарабатывать на своем таланте через различные способы распространения контента и взаимодействия с поклонниками. Наша платформа, «Ghosty», представит собой веб-решение, объединяющее элементы социальной сети, аукционной площадки и маркетплейса, что позволит пользователям максимально разнообразно взаимодействовать и монетизировать свой контент.

Основным источником дохода для нас будет комиссия за обслуживание сервиса, которая будет взиматься при оформлении платной подписки на контент, а также при совершении покупок и продаж на платформе. Это обеспечит стабильное финансирование для дальнейшего развития и улучшения «Ghosty».

**Имеющиеся аналоги**. Boosty представляет собой платформу, которая также ориентирована на творческих людей и предоставляет возможность заработка на контенте. Они предлагают инструменты для сбора пожертвований и подписок, а также обеспечивают пользователей возможностью продавать свои работы.

Patreon также является сервисом, позволяющим творческим личностям зарабатывать на своем контенте через подписки и пожертвования от поклонников. Ключевая особенность Patreon заключается в том, что она позволяет создателям контента получать стабильный доход от своей аудитории, обеспечивая им возможность создавать и делиться эксклюзивным контентом, доступным

только для своих поклонников-патронов. Этот контент может включать в себя ранний доступ к видео, уникальные фотографии, эссе, подкасты, веб-комиксы и многое другое.

Однако, мы считаем, что наша платформа, «Ghosty», имеет ряд преимуществ, таких как более разнообразные возможности взаимодействия с аудиторией и более гибкие инструменты монетизации контента. И будет предоставлять больше функциональности, включая элементы социальной сети и аукционной площадки, что позволит пользователям более полноценно взаимодействовать и монетизировать свое творчество.

**Обзор инструментальных средств**

**Веб-фреймворки**.

— .NET Core — это кроссплатформенный, высокопроизводительный фреймворк, разработанный Microsoft. Он поддерживает разработку и развертывание приложений на различных операционных системах, включая Windows, Linux и macOS. Основные преимущества .NET Core включают модульность, легковесность, открытый исходный код, а также возможность развертывания в контейнерах Docker и облачных средах.

— Django — это высокоуровневый веб-фреймворк для Python, который способствует быстрой и эффективной разработке веб-приложений. Он включает множество встроенных функций, таких как административная панель, ORM (Object-Relational Mapping), система маршрутизации URL и многое другое. Принципы Django позволяют разработчикам сосредоточиться на написании кода без необходимости заботиться о низкоуровневых деталях.

**Архитектура приложения**.

1. REST API (Representational State Transfer Application Programming Interface) — это архитектурный стиль взаимодействия между компонентами распределенной системы в сети. REST API используется для создания веб-сервисов, где клиенты и серверы взаимодействуют через HTTP запросы и ответы. Основные принципы REST включают использование стандартных методов HTTP, таких как GET, POST, PUT и DELETE для выполнения операций над ресурсами.
2. MVC (Model-View-Controller) — это архитектурный паттерн, разделяющий приложение на три взаимосвязанных компонента: модель, представ-

ление и контроллер. Модель управляет данными и логикой приложения, представление отвечает за отображение данных пользователю, а контроллер обрабатывает пользовательские вводы и взаимодействует с моделью для выполнения действий. MVC способствует организованному и модульному подходу к разработке приложений.

**Системы управления базами данных**.

— MySQL —– это система управления базами данных (СУБД) с открытым исходным кодом, основанная на языке SQL (Structured Query Language). Она широко используется для хранения и управления данными в веб-приложениях и корпоративных системах. MySQL поддерживает транзакции, репликацию и кластеризацию, что делает её популярным выбором для масштабируемых и надежных приложений.

— SQLite —– это легковесная СУБД, которая хранит всю базу данных в одном файле. Она не требует настройки сервера и идеально подходит для небольших и встраиваемых приложений. SQLite поддерживает основные возможности SQL, такие как транзакции и ограничения целостности данных, что делает её удобной для использования в различных проектах.

**Инструменты разработки**. В результате для выполнения ВКР были выбраны следующие инструменты:

1. Ruby on Rails (RoR) — это мощный веб-фреймворк с открытым исходным кодом, написанный на языке Ruby. Он обеспечивает быстрое и эффективное создание веб-приложений благодаря принципу «соглашение больше, чем конфигурация». Это означает, что разработчики могут следовать стандартным соглашениям по умолчанию, минимизируя необходимость в ручной настройке. Rails предлагает обширную библиотеку встроенных модулей, таких как Active Record, который упрощает работу с базами данных и управление данными приложения.
2. GraphQL — это язык запросов для API и среда выполнения для выполнения этих запросов с существующими данными. Он позволяет клиентам точно запрашивать данные, которые им нужны, и ничего больше. Это делает обмен данными между клиентом и сервером более эффективным и гибким. GraphQL отлично подходит для работы с сложными структурами данных и является мощным инструментом для разработки API.
3. PostgreSQL — это мощная объектно-реляционная система управления

базами данных с открытым исходным кодом. Она обеспечивает надежное хранение и управление данными, поддерживает ACID-транзакции и предоставляет расширенные возможности для работы с данными, включая индексы, подзапросы и многое другое. PostgreSQL является популярным выбором для веб-приложений благодаря своей стабильности и производительности.

**Реализация работы приложения**. Реализация серверной части веб-приложения с функционалом социальной сети включает в себя несколько ключевых этапов. Предварительная конфигурация проекта. Реализация функциональности авторизации пользователей. Разработка базового функционала социальной сети. Разработка функционала для обмена личными сообщениями. Разработка системы рекомендаций.

**Предварительная настройка проекта**.

1. **Создание rails приложения**. Перед началом работы с Ruby on Rails необходимо создать новое приложение с помощью консольной команды `rails new`. Эта команда формирует базовую структуру проекта, включая каталоги, файлы конфигурации и шаблоны кода.
2. **Расширение проекта с помощью GraphQL**. Выполним команду `rails generate graphql:install` в консоли. В результате сгенерируется контроллер и основная директория обработки запросов — `graphql`. Внутри этой директории расположен файл `api_scheme.rb`, который отвечает за определение корневых типов операций мутации, запросов и подписок, обработку ошибок, поддержку подписок и DataLoader, разрешение Union и Interface, а также обеспечивает идентификацию объектов в стиле Relay. Кроме того, в этой же директории имеются поддиректории `mutations`, `types` и `queries`, отвечающие за мутации, определенные типы и запросы соответственно.
3. **Настройка GraphQL**. GraphQL автоматически управляет маршрутизацией, поэтому в приложении определен лишь один маршрут `post "/graphql"`, который направляет все запросы в `graphql_controller.rb`. Этот контроллер оркестрирует все остальные процессы. Кроме того, определены еще два вспомогательных маршрута для отображения схемы GraphQL и статистики по подключениям Action Cable в веб-интерфейсе. Представленный код является завершенной версией маршрутизации в приложении,

поскольку библиотека GraphQL берет на себя остальную работу.

**Авторизация пользователей**. Авторизация пользователей является одним из наиболее важных аспектов приложения, поскольку именно ее реализация обеспечивает безопасность данных. В процессе разработки не удалось найти подходящего готового решения для GraphQL, которое бы удовлетворило все требования, поэтому было принято решение реализовать собственное. Процесс авторизации в приложении основан на использовании JWT (JSON Web Token) — открытого стандарта для создания токенов доступа, основанного на формате JSON. В ходе работы были реализованы мутации для регистрации, входа и выхода из системы, сброса пароля, а также для авторизации через сторонние сервисы, такие как ВКонтакте.

**Базовая функциональность**.

1. **Создание постов**. Пользователи могут создавать новые посты, каждый пост будет привязан к определенному пользователю, что позволит отслеживать авторство контента. Кроме того, настроены связи между постами и их комментариями, а также лайками. Каждый пост может иметь несколько комментариев и лайков, что обогащает пользовательский опыт. Также существует возможность прикреплять изображения к постам для улучшения визуального представления. Автор поста может создавать, удалять и редактировать свои посты, а другие пользователи могут комментировать и оценивать их.
2. **Создание комментариев**. Пользователи так же могут оставлять комментарии к постам. Модель комментариев связана с постом, пользователем и родителем через отношение `belongs_to`. Кроме того, комментарии могут комментироваться и оцениваться лайками. В данном контексте «родитель» представляет собой связь с другим комментарием, который может быть прокомментирован, что позволяет реализовать иерархию вложенных комментариев.
3. **Подписки на пользователей**. Добавление функционала подписок на пользователей позволяет пользователям отслеживать активность интересных авторов и формировать персонализированную ленту новостей. Путем подписки на других пользователей, пользователь отслеживает их активность, такую как новые публикации. Это расширяет возможности взаимодействия пользователей в приложении, делая его более социальным и удоб-

ным для использования.

4. **Лайки**. Функция лайков, предоставляемая нашим приложением, позволяет пользователям выражать свою оценку контенту, такому как комментарии и посты, размещенные другими пользователями. Это не только обогащает взаимодействие пользователей, но и способствует формированию социальной оценки контента. Этот механизм способствует продвижению идей, позволяя пользователям лучше понимать, какой контент наиболее интересен и актуален в их сообществе.
5. **Обновление аватара и шапки профиля**. Обновление аватара и шапки профиля предоставляет пользователям возможность персонализировать свой профиль и делать его более уникальным. Пользователи могут загрузить изображения, которые наилучшим образом отражают их личность, интересы или стиль жизни, и использовать их в качестве аватара или шапки профиля.

**Личные сообщения**. Для обмена личными сообщениями между пользователями создан отдельный компонент. Пользователи могут отправлять и получать сообщения в реальном времени, используя подписку на изменения данных для их мгновенной передачи. Это обеспечивает приватное и удобное общение внутри социальной сети. Реализация функционала личных сообщений базируется на моделях комнат, сообщений и их участников. Каждое сообщение и комната связаны с пользователем отношением «один-к-одному». Комната может содержать множество пользователей через записи об участниках комнаты и сообщениях.

Логика обработки личных сообщений основана на взаимодействии между двумя пользователями. Для создания диалога и создания комнаты для общения необходимо, чтобы один из пользователей отправил сообщение другому через страницу его профиля. Затем с помощью мутации создается комната для этих двух пользователей.

**Система рекомендаций**. Система рекомендаций основывается на социальных связях пользователя и его интересах. Для формирования рекомендательной ленты пользователя была разработана логика подбора постов с использованием SQL-выражения, состоящего из нескольких подзапросов.

1. Запрос `subscriptions_posts` возвращает все посты пользователей, на которых подписан текущий пользователь, и тех пользователей, которые

подписаны на текущего пользователя.

2. Запрос `liked_posts` возвращает все посты, которые понравились текущему пользователю.
3. Запрос `user_posts_whos_posts_was_liked` выбирает из всех постов пользователей те, которые были лайкнуты за последние 5 дней.

Затем результаты предыдущих подзапросов объединяются и возвращаются все записи, упорядоченные по дате создания, от новых к старым. Чтобы результат запроса соответствовал схеме данных GraphQL, результат его выполнения декорируется, и для каждого найденного поста возвращаются комментарии.

**Результаты разработки** Результатом разработки серверной части является не только реализация бизнес-логики приложения, но и создание надежного моста между клиентскими приложениями и базой данных. Эта часть приложения отвечает за обработку запросов, аутентификацию пользователей, выполнение операций с данными и обеспечение безопасности. Она является ключевым компонентом, который обеспечивает взаимодействие между пользовательским интерфейсом и хранилищем данных.

Для развертывания приложения необходимо скачать исходные файлы с удаленного репозитория или с приложенного CD-диска. Для первичной инициализации контейнеров требуется выполнить команду `make preinstall`. В процессе выполнения команды будет скачачен образ виртуальной машины, в которой впоследствии будет развернут сервер, и запускается терминал внутри этого контейнера. После установки откроется терминал внутри созданного контейнера. Затем для установки зависимостей, создания базы данных и запуска миграций необходимо выполнить `make install`. Для запуска сервера необходимо выполнить `make start`.

Иерархия серверной части приложения состоит из следующих элементов:

— папка `app` — основная директория приложения, где находится большинство кода:
  - папка `channels` — содержит реализацию кода с подпиской на события;
  - папка `controllers` — содержит исходный код контроллеров;
  - папка `graphql` — содержит исходный код мутаций, GraphQL конфигураций, запросов и структур ответов;

  - папка `models` — содержит исходный код моделей;
  - папка `services` — содержит исходный код модуля аутентификации.
- папка `config` — содержит файлы конфигурации приложения;
- папка `db` — содержит файлы миграции базы данных и файлы сидов для заполнения базы данных начальными данными;
- файл `routes.rb` — содержит код маршрутов приложения.

# ЗАКЛЮЧЕНИЕ

В ходе данной работы была реализована серверная часть приложения с функционалом социальной сети. Приложение разработано на базе фреймворка Ruby on Rails с использованием GraphQL для обеспечения эффективного взаимодействия между клиентской и серверной частями.

Для достижения поставленной цели в рамках работы:

1. было создано и сконфигурировано приложение на Ruby on Rails;
2. реализована безопасная регистрация и аутентификация пользователей;
3. реализованы базовые функции социальной сети с использованием фреймворка GraphQL, такие как создание и удаление постов, комментариев и т.д;
4. реализована механика работы личных сообщений;
5. реализована системы рекомендаций постов.

Таким образом, все поставленные в рамках работы задачи были выполнены.

Российский рынок информационных технологий проявляет высокую динамику и стремительный рост, создавая благоприятное окружение для предпринимательской активности. При наличии адекватной мотивации и разработке инновационных, удобных для пользователей решений, открывается реальная возможность занять устойчивую рыночную нишу.

В свете планов по дальнейшему развитию нашего проекта мы стремимся привлечь квалифицированных специалистов в области социального медиамаркетинга (SMM). Этот шаг является неотъемлемой частью нашей бизнес-стратегии, направленной на успешную реализацию наших целей и задач.

Участие в программе «Стартап как диплом» в Саратовском государственном университете предоставило нам уникальную возможность не только получить ценные обратные связи и рекомендации от ведущих экспертов, но и расширить наши знания и понимание в области стартап-инициатив и инновационного предпринимательства.

Благодаря эффективному выполнению поставленных задач нами было создано функциональное и безопасное приложение, ориентированное на удовлетворение потребностей пользователей в общении, обмене информацией и поиске интересного контента и единомышленников.