### Введение

В современном мире информационные технологии проникли во все сферы

деятельности, включая управление человеческими ресурсами. Компании,

стремящиеся к устойчивому развитию и конкурентоспособности, вынуждены

искать новые способы оптимизации своих бизнес-процессов, включая найм и

отбор персонала. Эффективное управление процессами найма, особенно в

условиях высокой конкуренции на рынке труда, становится ключевым

фактором успеха для любой организации. В этом контексте разработка и

внедрение специализированных веб-сервисов для приема заявок на

стажировку представляется не просто актуальной задачей, но и стратегически

важным шагом.

Стажировка является важным этапом для молодых специалистов, которые

стремятся получить первый опыт работы в профессиональной среде.

Компании, предлагающие стажировки, получают уникальную возможность

привлечь талантливых выпускников, инвестируя в их обучение и развитие.

Однако традиционные методы обработки заявок на стажировку, основанные

на бумажных носителях или простых электронных формах, становятся все

менее эффективными. Это связано с увеличением объемов данных, которые

необходимо обрабатывать, а также с ростом ожиданий кандидатов в

отношении удобства и скорости подачи заявок.

В данной дипломной работе рассматривается разработка веб-сервиса для

приема заявок на стажировку в компании Reviro. Основной целью проекта

является создание удобного и эффективного инструмента, который позволит

автоматизировать процесс приема и первичной обработки заявок, обеспечив

высокий уровень удобства как для кандидатов, так и для HR-специалистов.

Веб-сервис должен обеспечить возможность подачи заявок в любое время и

из любого места, хранение и защиту личных данных, а также интеграцию с

внутренними системами компании для дальнейшей обработки и анализа.

Разработка такого веб-сервиса требует проведения всестороннего анализа

требований, включая изучение существующих решений, выявление

потребностей пользователей, а также определение технических и

функциональных характеристик системы. Важно учитывать не только

текущие потребности компании, но и перспективы развития, чтобы

созданный сервис мог быть легко адаптирован и расширен в будущем.

Настоящая работа состоит из нескольких разделов, в которых

последовательно рассматриваются этапы анализа, проектирования,

разработки и тестирования веб-сервиса. В первой главе проводится обзор и

анализ требований, обосновывается актуальность разработки и описываются

основные функции будущей системы. В последующих главах освещаются

процесс проектирования архитектуры сервиса, выбор технологий и

инструментов, а также этапы реализации и тестирования.

Результатом выполнения дипломного проекта станет веб-сервис, который

позволит компании Reviro существенно повысить эффективность процесса

найма стажеров, улучшить качество отбора кандидатов и сократить

временные и финансовые затраты на обработку заявок. Кроме того,

внедрение такого решения будет способствовать укреплению имиджа

компании как инновационного и привлекательного работодателя для молодых

специалистов.

Таким образом, разработка веб-сервиса для приема заявок на стажировку

является важной и актуальной задачей, решение которой требует

комплексного подхода и учета множества факторов. Этот проект не только

способствует улучшению внутренних процессов компании, но и открывает

новые возможности для привлечения и развития молодых талантов, что в

конечном итоге способствует долгосрочному успеху и процветанию

организации.

### Глава 1 Обзор и анализ требований

#### 1.1 Актуальность разработки ПО

В современном мире информационные технологии играют ключевую роль в

различных аспектах бизнеса и общественной жизни. Одним из таких

аспектов является процесс найма и отбора персонала, который становится все

более сложным и многоуровневым. В условиях высокой конкуренции на

рынке труда компании вынуждены искать инновационные решения для

оптимизации процессов поиска и отбора кандидатов. В этом контексте

разработка программного обеспечения для приема заявок на стажировку в

компании приобретает особую актуальность.

Компании, стремящиеся привлекать молодых специалистов и выпускников,

сталкиваются с рядом проблем и вызовов. Одной из ключевых проблем

является необходимость эффективного управления большими объемами

данных, поступающих от кандидатов. Вручную обрабатывать заявки

становится нецелесообразно и трудозатратно, особенно в крупных

организациях, где количество потенциальных стажеров может исчисляться

сотнями и тысячами. Использование специализированного программного

обеспечения позволяет автоматизировать этот процесс, что существенно

сокращает временные и финансовые затраты.

Кроме того, современная молодежь, являющаяся основной целевой

аудиторией для стажировок, активно использует цифровые технологии. Для

них важно, чтобы процесс подачи заявок был максимально удобным и

понятным. Веб-сервис для приема заявок обеспечивает интуитивно понятный

интерфейс и доступность в любое время и из любого места, что повышает

лояльность кандидатов и способствует увеличению числа поданных заявок.

Важным аспектом является также возможность оперативного анализа и

обработки данных. Программное обеспечение позволяет автоматизировать не

только процесс приема заявок, но и их первичный отбор на основе заданных

критериев. Это позволяет HR-специалистам концентрироваться на более

значимых этапах отбора, таких как интервью и оценка профессиональных

навыков, тем самым повышая общую эффективность процесса найма.

Не менее значимым является вопрос безопасности данных. Веб-сервис для

приема заявок должен обеспечивать надежное хранение и защиту личных

данных кандидатов, соответствуя современным стандартам и

законодательным требованиям в области защиты информации.

Использование специализированного ПО позволяет минимизировать риски

утечек и несанкционированного доступа к данным.

Компания Reviro, являющаяся прогрессивной и динамично развивающейся

организацией, стремится применять современные технологии во всех

аспектах своей деятельности. Разработка веб-сервиса для приема заявок на

стажировку соответствует стратегии компании по внедрению инноваций и

цифровых решений. Это позволит компании не только оптимизировать

внутренние процессы, но и укрепить свой имидж как работодателя, идущего

в ногу со временем и предлагающего молодым специалистам удобные и

современные инструменты для взаимодействия.

Таким образом, разработка программного обеспечения для приема заявок на

стажировку является актуальной задачей, обусловленной современными

тенденциями на рынке труда и потребностями компаний в эффективном

управлении процессами найма. Это решение способствует повышению

конкурентоспособности организации, улучшению качества отбора

кандидатов и оптимизации внутренних бизнес-процессов. В условиях

стремительного развития технологий и повышения требований к управлению

персоналом, внедрение такого программного обеспечения становится не

просто желательным, а необходимым шагом на пути к успешной и

эффективной работе компании.

**1.2 Выбор программных средств разработки**

В этом пункте рассмотрены программные средства, которые

понадобились для разработки сайта: HTML5, язык разметки CSS3, Twitter

Bootstrap, JavaScript, фреймворк для разработки сайтов Python Django и среда

разработки языка Python — Pycharm.

Основное преимущество такого подхода заключается в разделении

написания клиентской части сайта от серверной. Следовательно, каждая

отдельная часть приложения, созданного с помощью Python, имеет одно

назначение и может быть изменена независимо, т.е., без влияния на

остальные

компоненты. Дизайнер может изменить HTML страницы без внесения

изменений в код, который отображает страницу. Администратор базы данных

может переименовать таблицу и определить эти изменения в одном месте,

вместо того, чтобы искать и вносить изменения во множество файлов.

Для подготовки гипертекстовых документов в данной работе

используется HTML5 (Hyper Text Markup Language — язык разметки

гипертекстовых документов), предоставляющий широкие возможности по

форматированию и структурной разметке документов, организации связей

между различными документами, средства включения графической и

мультимедийной информации.

В пятой версии HTML разработчики постарались объединить все

инструменты, необходимые для создания профессиональных, современных и

динамичных сайтов, использующих наиболее распространенные технологии

и соответствующих современным стандартам.

Первое, что стоит отметить, это использование в работе Twitter Bootstrap

— свободного набора инструмента работающего с HTML и CSS. В данном

проекте с Bootstrap было сформировано всё оформление и шаблоны сайта.

Во-вторых — использование JavaScript — это инструмент,

предназначенный для придания динамичности HTML-страницам. В работе

JavaScript используется для динамического изменения данных на странице и

отрисовки графиков. Для рисования графиков используется библиотека

Chart.js.

Следующее что стоит отметить — это использование фреймворка для

разработки сайтов Django.

Django включает в себя Model View Controller — инструмент,

позволяющий разделить общую архитектуру на отдельные части. При этом

управляющая логика разделена на три отдельных компонента так, что

модификация одного из них оказывает минимальное воздействие на другие

части. К таким компонентам относят разделяемые данные, логику и слои

визуализации. В общем случае такая концепция позволяет разделить

разработку информационного наполнения на уровне базы данных и

разработку web-страниц.

**Python**

Python — высокоуровневый язык программирования, построенный на

идея императивного, объектно-ориентированного и функционального

программирования. Язык создан Гвидо ван Россумом в 1989 году и с тех пор

непрерывно совершенствуется.

Преимущества Python:

– открытая разработка;

– язык довольно прост в изучении, особенно на начальном этапе;

– особенности синтаксиса стимулируют программиста писать хорошо

читаемый код;

– предоставляет средства быстрого прототипирования и динамической

семантики;

– имеет большое сообщество, позитивно настроенное по отношению к

новичкам;

– множество полезных библиотек и расширений языка можно легко

использовать в своих проектах благодаря предельно унифицированному

механизму импорта и программным интерфейсам;

– механизмы модульности хорошо продуманы и могут быть легко

использованы;

– абсолютно всё в Python является объектами в смысле ООП, но при

этом объектный подход не навязывается программисту;

– python предлагает инструменты для простой и функциональной

разработки web-сайтов.

Исходя из преимуществ, язык программирования Python подходит для

решения поставленной задачи, т.е. разработки серверной и клиентской части

сайта.

**HTML**

HTML — это язык гипертекстовой разметки, с помощью которого

верстальщики создают структуру web-страниц, приложений и email-писем.

HTML формирует структуру веб-страницы посредством заголовков,

списков и других подобных элементов, начиная от начала страницы —

хедера(header), и до ее конца — футера(footer).

Структура web-страницы должна быть прописана на web-основе с

помощью определенных команд.

В данном языке оперирует понятие тег. Тег это своего рода указатель на

какую-либо информацию. Теги бывают парные и одиночные. Для парного

тега

необходимо определить начало(<b>) и конец(</b>), для одиночного же тега

присуще не закрытие формы.

Теги имеют так же еще и параметры, которыми можно указать размер

объекта: style="width: 445px; height: 445px;", шрифт текста:

style="font-family: 'Fjalla One', sans-serif;" и т.д.

Свободные места внутри тегов называют контейнерами, теги

определяют действие только для содержимого своего контейнера [11].

Простейший html-документ выглядит следующим образом:

<!DOCTYPE html>

<html>

<head>

<title>Заголовок</title>

</head>

<body>

<h1>Тут будет размещен заголовок</h1>

<p>Первый абзац</p>

</body>

</html>

Представленный небольшой вариант простого HTML-кода, содержит

несколько элементов, состоящих, в свою очередь, из таких тэгов как:

– <head></head> — открывающий и закрывающий тэг, предназначен для

хранения элементов, цель которых — помочь браузеру в работе с данными.

Содержимое тега <head> не отображается на html-странице, за исключением

тега <title> который устанавливает окна web-страницы;

– <body></body> — открывающий и закрывающий тэг, указывающий на

начало и окончание основного блока html-документа, в котором будет

содержаться ее контент;

– <h1></h1> — открывающий и закрывающий тэг, указывающий на

начало и окончание заголовка. Всего таких тегов может быть 6, и отличаются

они величиной шрифта — чем выше числовой порядок заголовка, тем

меньшими буквами он будет отражаться. Вместе с тегом <p>, <h1>

формирует структуру самого контента: в частности метки <p></p>

определяют начало и конец абзацев текста.

Таким образом, один за другим формируются разные элементы, которые

впоследствии будут представлены на выходной странице.

**Django**

Django — свободный фреймворк для веб-приложений на языке Python,

использующий шаблон проектирования MVC. Проект поддерживается

организацией Django Software Foundation.

Сайт на Django строится из одного или нескольких приложений,

которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из

существенных архитектурных отличий этого фреймворка от некоторых

других (например, Ruby on Rails). Один из основных принципов фреймворка

— DRY (Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django

конфигурируются явно при помощи регулярных выражений, а не выводятся

автоматически из структуры моделей контроллеров.

Для работы с базой данных Django использует собственный ORM, в

котором модель данных описывается классами Python, и по ней генерируется

схема базы данных.

Архитектура приложения Django основана на шаблоне модель–шаблон–

представление (Model-Template-View) в том смысле, что в ней разделяются

уровни логики приложения, пользовательского интерфейса (User Interface) и

доступа к данным с целью дать возможность изменять каждый из этих

уровней независимо от других уровней. Модель можно рассматривать как

уровень доступа к данным, на котором приложение взаимодействует с

любыми базами данных и источниками информации. Шаблон — это уровень,

определяющий то, как данные должны быть представлены пользователю. В

шаблоне модель – шаблон–представление (далее MTV) уровень

представления описывает, какие данные должны быть представлены

пользователю. Что же до уровня контроллера шаблона модель–

представление–контроллер (Model-View-

Controller), считается, что в Django его роль выполняет сама инфраструктура,

которая в соответствии с конфигурацией адресов URL определяет, какому

представлению следует послать тот или иной запрос.

Помимо моделей, шаблонов и представлений, Django предлагает

изначально такую функциональность, как конфигурация адресов URL,

автоматический интерфейс администрирования базы данных, кэширование и

прочее, что значительно облегчает работу web-разработчика. Так же как и в

Python, Django предоставляет совместимость с большой стандартной

библиотекой дополнительных пакетов, которые можно использовать в своих

приложениях без необходимости дополнительной загрузки чего-либо.

Уровень модели приложения Django обрабатывается в инфраструктуре

Django уровнем доступа к данным. Внутри этого уровня находится все,

относящееся к данным: настройки соединения, параметры валидации,

отношения и т.д. Изначально в Django включена поддержка PostgreSQL (базы

данных, предпочитаемой создателями Django), MySQL, SQLite и Oracle.

Информация о том, какую базу данных следует использовать, храниться в

файле настроек, и уровень модели будет одним и тем же независимо от того,

какую именно базу данных выбрать.

Модели в Django можно рассматривать как описание схемы таблиц базы

данных, представленное в виде кода на Python. Django использует модель для

генерации и выполнения в базе данных инструкций SQL. База данных в свою

очередь возвращает результат, который Django переводит в структуру данных

Python, доступную для использования приложением Django. Очевидным

преимуществом такого подхода является то, что можно переключаться между

различными СУБД, например, перейти с MySql на PostgreSQL, не меняя при

этом модели приложения [5].

Структура web-приложения фреймворка Django выглядит следующим

образом:

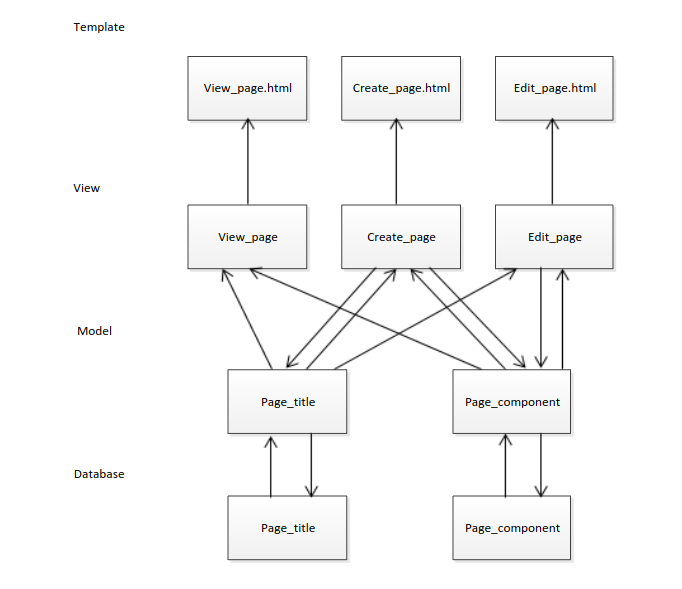
 Рисунок 1

Рисунок 1 — Структура web-приложения Django

Таким образом архитектура проекта Django согласно модели MVC

разделяет приложение на три составляющих: модель данных приложения,

интерфейс конечного пользователя и средства управления.

**PostgreSQL**

PostgreSQL. (обычно также называемая Postgres) - это мощная и расширяемая реляционная база данных с открытым исходным кодом. Она предлагает множество функций и возможностей для хранения, организации и управления данными. Вот некоторые ключевые особенности PostgreSQL:

Реляционная структура: PostgreSQL основана на реляционной модели данных, что означает, что данные организованы в таблицы с явными связями между ними.

Многопользовательская поддержка: PostgreSQL предоставляет возможность многопользовательского доступа к базе данных, позволяя нескольким пользователям одновременно выполнять операции чтения и записи.

Расширяемость: PostgreSQL обладает мощным механизмом расширений, который позволяет разработчикам создавать и интегрировать собственные функции, типы данных и другие расширения в базу данных.

Поддержка SQL: PostgreSQL полностью поддерживает язык SQL, включая множество расширенных возможностей, таких как подзапросы, представления, триггеры и хранимые процедуры.

Транзакционность и целостность данных: PostgreSQL. обеспечивает ACID-свойства (атомарность, согласованность, изолированность долговечность) для транзакций, гарантируя целостность и надежность данных.

Поддержка JSON и других нереляционных возможностей: PostgreSQL предлагает возможности работы с нереляционными данными, включая Хранение и манипуляцию JSON-данными с помощью встроенных функций и операторов.

Репликация и отказоустойчивость: PostgreSQL, поддерживает различные методы репликации, позволяющие создавать резервные копии данных и обеспечивать отказоустойчивость системы.

Масштабируемость: PostgreSQL. предлагает различные горизонтального и вертикального масштабирования, позволяя управлять ростом объема данных и обеспечивать производительность при высоких нагрузках.

PostgreSQL является одной из наиболее популярных реляционных баз данных в мире благодаря своей надежности, производительности и богатому набору функций. Она широко используется как для небольших проектов, так и для крупнейших предприятий и веб-приложений. Ее активное сообщество разработчиков и поддержка открытого исходного кода обеспечивают постоянное развитие и улучшение базы данных.

Одной из привлекательных особенностей PostgreSQL является его расширяемость. Благодаря механизму расширений, разработчики могут создавать собственные функции, типы данных и другие расширения, которые интегрируются в базу данных. Это позволяет адаптировать PostgreSQL под конкретные потребности проекта и расширять его возможности.

PostgreSQL также предлагает мощную систему индексации, позволяющую улучшить производительность запросов и поиск данных. Она поддерживает различные типы индексов, включая В-деревья, хэши, GIN Generalized Inverted Index) и другие. Выбор подходящего типа индекса зависит от конкретных требований приложения.

Другой важной возможностью PostgreSQL является поддержка полнотекстового поиска. База данных предоставляет инструменты для выполнения поисковых запросов по текстовым данным, включая возможность использования словарей, настройки весов терминов и выполнения расширенных поисковых операций.

PostgreSQL также обладает возможностями для работы с геоданными. Она поддерживает географические объекты и операции над ними, что делает ее полезной для приложений, связанных с геолокацией, картографией и анализом данных на основе местоположения.

В целом, PostgreSQL. предлагает широкий набор возможностей, которые делают ее привлекательным выбором для разработки веб-приложений и управления данными. Ее надежность, производительность

и гибкость делают се популярным решением для широкого спектра проектов.

Вывод: благодаря выше описанным системам можно разработать интернет-магазин.

# **Глава 2 Проектная часть**

## **2.1 Разработка модели бизнес-процессов**

##### 2.1.1 Описание процесса создания профиля кандидата

1. Данные пользователя:
   * Получение личных данных пользователя для создания профиля

кандидата.

1. Процесс создания профиля кандидата:
   * Ввод данных о пользователе, таких как имя, фамилия,
   * электронная почта, пароль и другие необходимые данные для
   * создания профиля кандидата.

##### 2.1.2 Описание процесса подачи заявки кандидатом

1. Информация о вакансии:
   * Получение информации о доступных вакансиях.
2. Профиль кандидата:
   * Использование данных из профиля кандидата для подачи заявки.
3. Процесс подачи заявки кандидатом:
   * Заполнение и подача заявки на выбранную вакансию.

##### 2.1.3 Описание процесса добавления комментария к заявке

1. Заявка:
   * Обработка поданной заявки.
2. Комментарий:
   * Добавление комментариев к заявке для дополнительной
   * информации или обратной связи.
3. Процесс добавления комментария к заявке:
   * Интеграция комментариев в процесс рассмотрения заявки.

##### 2.1.4 Разработка ERD (Диаграмма сущностей и связей)

1. Сущность "User":
   * Атрибуты: id, username, password, email, first\_name, last\_name.
2. Сущность "CandidateProfile":
   * Атрибуты: id, name, last\_name, education, experience, photo,

phone\_number, address.

* + Связь: один к одному с "User".

1. Сущность "Application":
   * Атрибуты: id, resume, cover\_letter, comments, position,

date\_submitted, interview\_date, feedback, rating, portfolio\_link,

github\_profile.

* + Связь: многие к одному с "CandidateProfile".

1. Сущность "Comment":
   * Атрибуты: id, text.
   * Связь: многие к одному с "Application".

##### 2.1.5 Описание процесса взаимодействия

1. Создание профиля:
   * Пользователь создает профиль, который сохраняется как

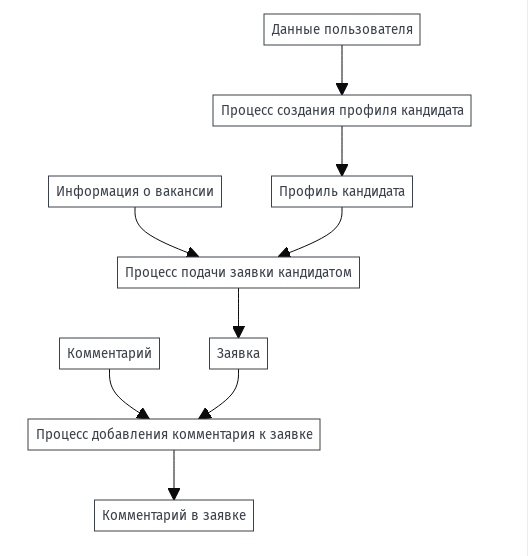
сущность "CandidateProfile" и связывается с сущностью "User".

1. Подача заявки:
   * Кандидат использует свой профиль для подачи заявки на

вакансию, создавая сущность "Application".

1. Получение комментария:
   * Комментарии добавляются к заявкам, создавая сущность

"Comment", которая связывается с сущностью "Application".



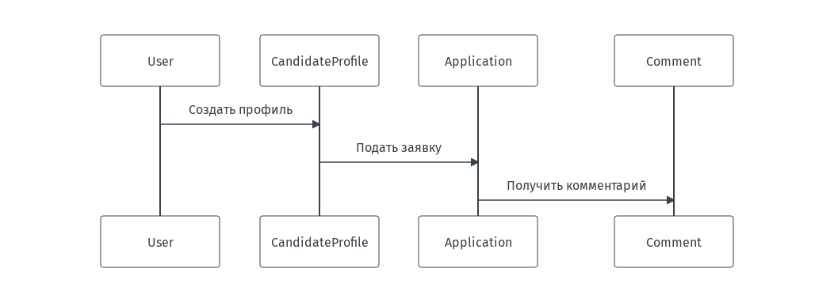


Рисунок 2 диаграмма IDEF0

Рисунок 3 диаграмма последовательности

# Рисунок 4 диаграмма классов

# 

# Рисунок 5 ERD (Диаграмма сущностей и связей)

# Глава 3 Разработка документации

## 3.1 Руководство пользователю

#### Основные функции веб-сервиса:

1. Регистрация и аутентификация пользователей
   * JWT (JSON Web Token): Для обеспечения безопасности и

удобства использования веб-сервис поддерживает

аутентификацию с помощью JWT. Это позволяет пользователям

регистрироваться и входить в систему с использованием токенов,

которые предоставляют безопасный и эффективный способ

управления сессиями.

* + Сессии: В дополнение к JWT, сервис поддерживает

классическую аутентификацию с использованием сессий. Это

позволяет гибко настраивать безопасность и сохранять

информацию о пользователе в течение его взаимодействия с

сервисом.

1. Личный профиль пользователя
   * Просмотр и редактирование профиля: Каждый пользователь

имеет доступ к личному профилю, где можно просматривать и

редактировать персональные данные. Профиль включает

информацию, такую как имя, контактные данные, образование и

опыт работы.

* + История заявок: В личном профиле также хранится история

поданных заявок на стажировку. Это позволяет пользователям

отслеживать статус своих заявок и получать уведомления о

любом изменении их статуса.

1. Отправка заявок на стажировку
   * Аутентификация обязательна: Для обеспечения безопасности и

защиты от спама, отправка заявок возможна только для

аутентифицированных пользователей. Это гарантирует, что все

поданные заявки поступают от реальных кандидатов,

заинтересованных в стажировке.

* + Форма заявки: Пользователи могут заполнить онлайн-форму,

включающую информацию о личных данных, образовании, опыте

работы и мотивационное письмо. Форма может быть дополнена

файлами, такими как резюме или портфолио, для более полного

представления кандидата.

#### Используемые технологии

**Редактор кода: Visual studio code**

##### Back-end

1. Django: Основной фреймворк для разработки веб-приложения, предоставляющий мощные и гибкие инструменты для создания надежных веб-сервисов.
2. PostgreSQL: Высокопроизводительная реляционная база данных, используемая для хранения и управления данными пользователей и заявок.
3. Django REST Framework (DRF): Библиотека для создания API, обеспечивающая удобные средства для разработки RESTful веб-сервисов.
4. GraphQL: Используется для создания гибкого и эффективного API, позволяющего клиентам запрашивать только те данные, которые им необходимы.
5. gRPC: Протокол для эффективного взаимодействия между микросервисами, обеспечивающий высокую производительность и низкую задержку при передаче данных.

##### Front-end

1. Django Template: Для визуализации проекта используется встроенная в Django система шаблонов, которая позволяет создавать динамические и интерактивные веб-страницы. Django Templates обеспечивают гибкость и простоту в создании пользовательского интерфейса, интегрируясь с серверной частью приложения.

### Архитектура веб-сервиса

### Веб-сервис является отдельным микросервисом в архитектуре главного сайта компании "Ревиро". Такая модульная архитектура обеспечивает гибкость, масштабируемость и независимость каждого компонента системы. Рассмотрим основные элементы и взаимодействия внутри этой архитектуры.

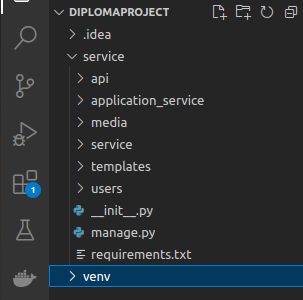


Рис 6. Файловая структура проекта

Для удобства разработки и поддержки веб-сервиса мы используем Django, разделяя проект на несколько приложений (apps) как показано на рисунке 9. В нашем случае это три приложения: api, users и application\_service, а также директория media для хранения пользовательских файлов, таких как резюме и портфолио.

Давайте рассмотрим каждый директорий по отдельности:

diploma\_project/

│

├── service/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py

│ ├── wsgi.py

│ ├── asgi.py

│ └── ...

1. service/

* settings.py: Файл конфигурации проекта, содержащий настройки для Django.
* urls.py: Главный файл маршрутизации проекта, связывающий URL-адреса с соответствующими view-функциями.
* wsgi.py и asgi.py: Файлы конфигурации для развертывания проекта с использованием WSGI и ASGI.

│

├── api/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── serializers.py

│ ├── urls.py

│ ├── views.py

│ ├── tests.py

│ └── migrations/

│ └── \_\_init\_\_.py

2. api/

* models.py: Определение моделей данных, используемых в API.
* serializers.py: Определение сериализаторов для преобразования данных моделей в форматы JSON или XML и обратно.
* urls.py: Файл маршрутизации для API.
* views.py: View-функции или классы для обработки запросов API.
* tests.py: Тесты для проверки работы API.
* migrations/: Директория для хранения миграций базы данных.

│

├── users/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── serializers.py

│ ├── urls.py

│ ├── views.py

│ ├── tests.py

│ ├── forms.py

│ ├── signals.py

│ └── \_\_init\_\_.py

3. users/

* models.py: Определение моделей данных для пользователей.
* serializers.py: Определение сериализаторов для пользователей.
* urls.py: Файл маршрутизации для пользователей.
* views.py: View-функции или классы для обработки запросов, связанных с пользователями.
* forms.py: Определение форм для регистрации, аутентификации и других операций с пользователями.
* signals.py: Определение сигналов для выполнения дополнительных действий при определённых событиях (например, при регистрации пользователя).
* templates/registration/: Шаблоны для страниц регистрации, входа, сброса пароля и других страниц, связанных с пользователями.
* migrations/: Директория для хранения миграций базы данных.

│

├── application\_service/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── serializers.py

│ ├── urls.py

│ ├── views.py

│ ├── tests.py

│ ├── forms.py

│ ├── signals.py

│ └── migrations/

│ └── \_\_init\_\_.py

4. application\_service/

* models.py: Определение моделей данных для заявок на стажировку.
* serializers.py: Определение сериализаторов для заявок.
* urls.py: Файл маршрутизации для заявок на стажировку.
* views.py: View-функции или классы для обработки запросов, связанных с заявками на стажировку.
* forms.py: Определение форм для подачи заявок на стажировку.
* signals.py: Определение сигналов для выполнения дополнительных действий при определённых событиях (например, при подаче заявки).
* templates/application\_service/: Шаблоны для страниц подачи заявок и просмотра списка заявок.
* migrations/: Директория для хранения миграций базы данных.

│

├── media/

│ ├── resumes/

│ │ └── ...

│ ├── portfolios/

│ │ └── ...

│ └── ...

5. media/

* resumes/: Директория для хранения загруженных резюме пользователей.
* portfolios/: Директория для хранения загруженных портфолио пользователей.

│

├── manage.py

├── requirements.txt

└── README.md

6. manage.py: Основной файл управления проектом Django, используемый для выполнения команд (например, запуска сервера, создания миграций и т.д.).

1. requirements.txt: Файл, содержащий список зависимостей проекта.
2. README.md: Файл с описанием проекта, инструкциями по установке и использованию.

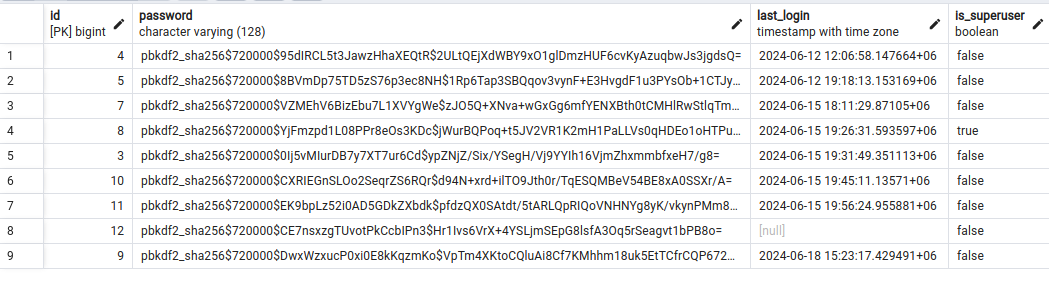
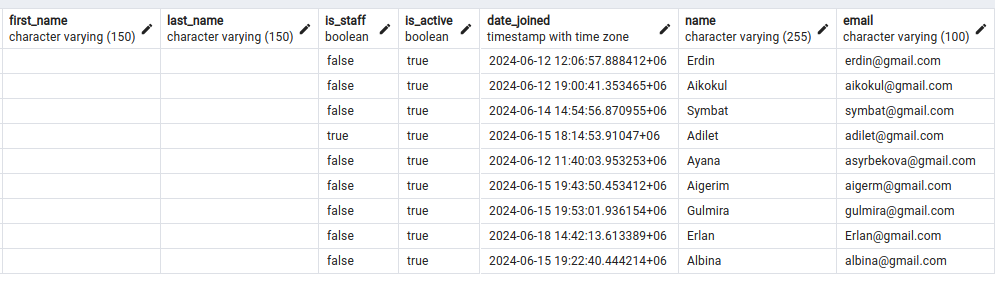
Эта структура обеспечивает организованное разделение кода, что облегчает разработку, тестирование и поддержку сервиса.

**Реализация Back-end части:**

### Структура базы данных:

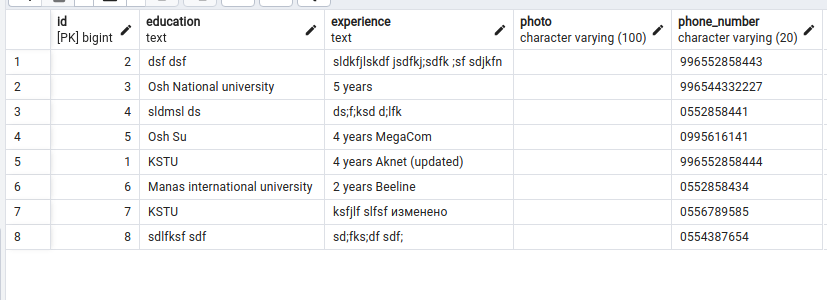
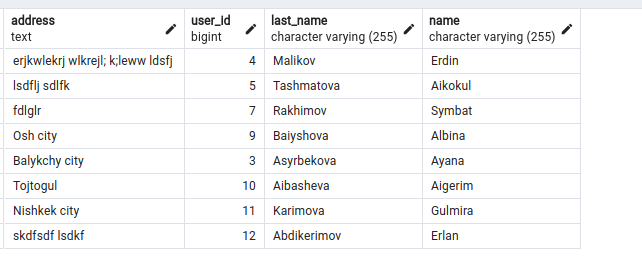
#### Таблица users\_user (модель User из приложения users) на рисунке 10 и 11:

* id: Primary Key, автоматически генерируемый Django
* password: Хэшированный пароль пользователя
* last\_login: Дата последнего входа
* is\_superuser: Флаг суперпользователя
* name: Имя пользователя
* email: Уникальный адрес электронной почты пользователя
* is\_staff: Флаг персонала (для доступа к административной панели Django)
* is\_active: Флаг активности учетной записи
* date\_joined: Дата создания учетной записи

 **Рис 7 и 8**

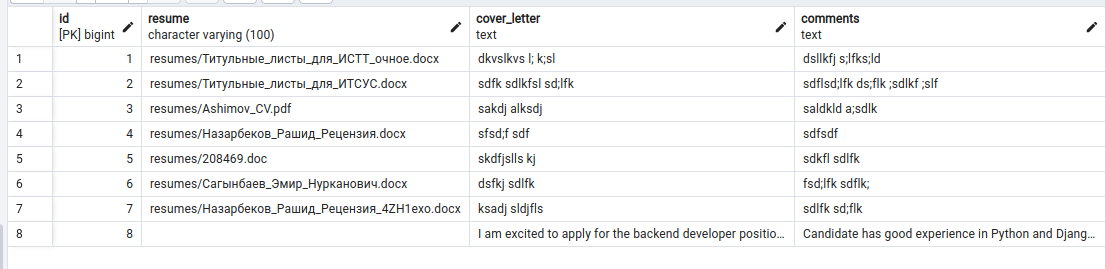
#### Таблица application\_service\_candidateprofile (модель CandidateProfile из приложения application\_service) на рисунке 7 и 8:

* id: Primary Key, автоматически генерируемый Django
* user\_id: Foreign Key, связь с таблицей users\_user через поле user
* name: Имя кандидата
* last\_name: Фамилия кандидата
* education: Текстовое поле с описанием образования кандидата
* experience: Текстовое поле с описанием опыта работы кандидата
* photo: Фотография кандидата (хранится как путь к файлу в системе)
* phone\_number: Номер телефона кандидата
* address: Адрес кандидата

 Рисунок 10 и 11

#### Таблица application\_service\_application (модель Application из приложения application\_service) на рисунке 14,15,16:

* id: Primary Key, автоматически генерируемый Django
* candidate\_id: Foreign Key, связь с таблицей application\_service\_candidateprofile через поле candidate
* resume: Резюме кандидата (хранится как путь к файлу в системе)
* cover\_letter: Сопроводительное письмо кандидата
* comments: Комментарии к заявке
* position: Позиция кандидата (выбор из предопределенных вариантов)
* date\_submitted: Дата подачи заявки (автоматически добавляется при создании)
* interview\_date: Дата собеседования (может быть пустой)
* feedback: Обратная связь по результатам собеседования
* rating: Оценка кандидата
* portfolio\_link: Ссылка на портфолио кандидата
* github\_profile: Ссылка на профиль кандидата на GitHub



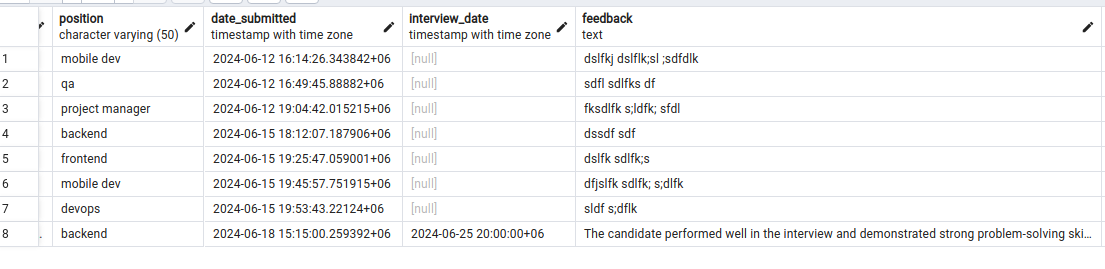
 Рисунок 14

Рисунок 15

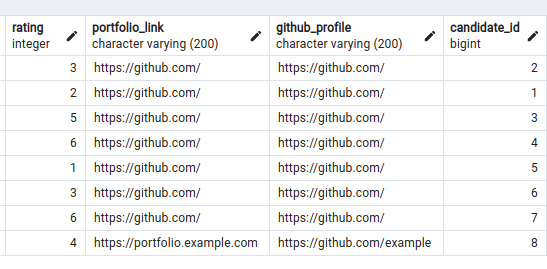
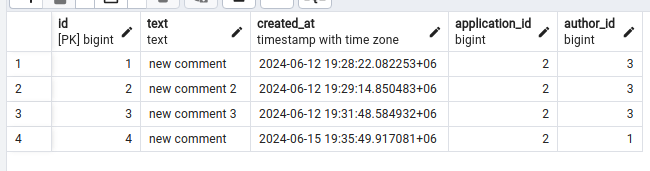


Рисунок 16

#### Таблица application\_service\_comment (модель Comment из приложения application\_service) на рисунке 17:

* id: Primary Key, автоматически генерируемый Django
* application\_id: Foreign Key, связь с таблицей application\_service\_application через поле application
* author\_id: Foreign Key, связь с таблицей application\_service\_candidateprofile через поле author
* text: Текст комментария
* created\_at: Дата создания комментария (автоматически добавляется при создании)

 Рисунок 17

### Связи между таблицами

* CandidateProfile связана с User через user\_id (OneToOneField), предоставляя дополнительную информацию о пользователе.
* Application связана с CandidateProfile через candidate\_id (ForeignKey), что позволяет отслеживать заявки, поданные конкретным кандидатом.
* Comment связана с Application через application\_id (ForeignKey), обеспечивая возможность добавления комментариев к конкретной заявке.
* Comment также связана с CandidateProfile через author\_id (ForeignKey), указывая автора комментария.

Эта структура базы данных соответствует представленным моделям и обеспечивает эффективное хранение и связывание данных для работы с заявками на стажировку и профилями кандидатов.

**Описание эндпоинтов:**

1. admin/**:**

* + Описание: Эндпоинт для доступа к административной панели Django.

1. auth/:
   * Описание: Эндпоинты для аутентификации и управления пользователями через пакет djoser. Включает стандартные маршруты для регистрации, восстановления пароля и подтверждения email.
2. auth/ (JWT):
   * Описание: Эндпоинты для работы с JWT-токенами через пакет djoser, такие как получение и обновление токенов.
3. accounts/:
   * Описание: Эндпоинты для аутентификации и управления пользователями через пакет django-allauth. Включает маршруты для регистрации, входа в систему, выхода и управления учетной записью.
4. swagger<format>/:
   * Описание: Эндпоинт для получения схемы API в различных форматах (например, JSON).
5. swagger/:
   * Описание: Эндпоинт для интерактивной документации API с использованием Swagger UI.
6. redoc/:
   * Описание: Эндпоинт для интерактивной документации API с использованием ReDoc.
7. home/:
   * Описание: Эндпоинты для основного функционала сервиса, включающие маршруты из приложения application\_service.
8. users/:
   * Описание: Эндпоинты для работы с пользователями, включающие маршруты из приложения users.
9. api/:
   * Описание: Эндпоинты для API, включающие маршруты из приложения api.
10. graphql/:
    * Описание: Эндпоинт для доступа к GraphQL API.

### Шаги для выполнения запроса к GraphQL API

1. Определение запроса: Вам нужно написать GraphQL-запрос, который соответствует структуре вашей схемы.

graphql

Copy code

query {

allProfiles {

id

name

lastName

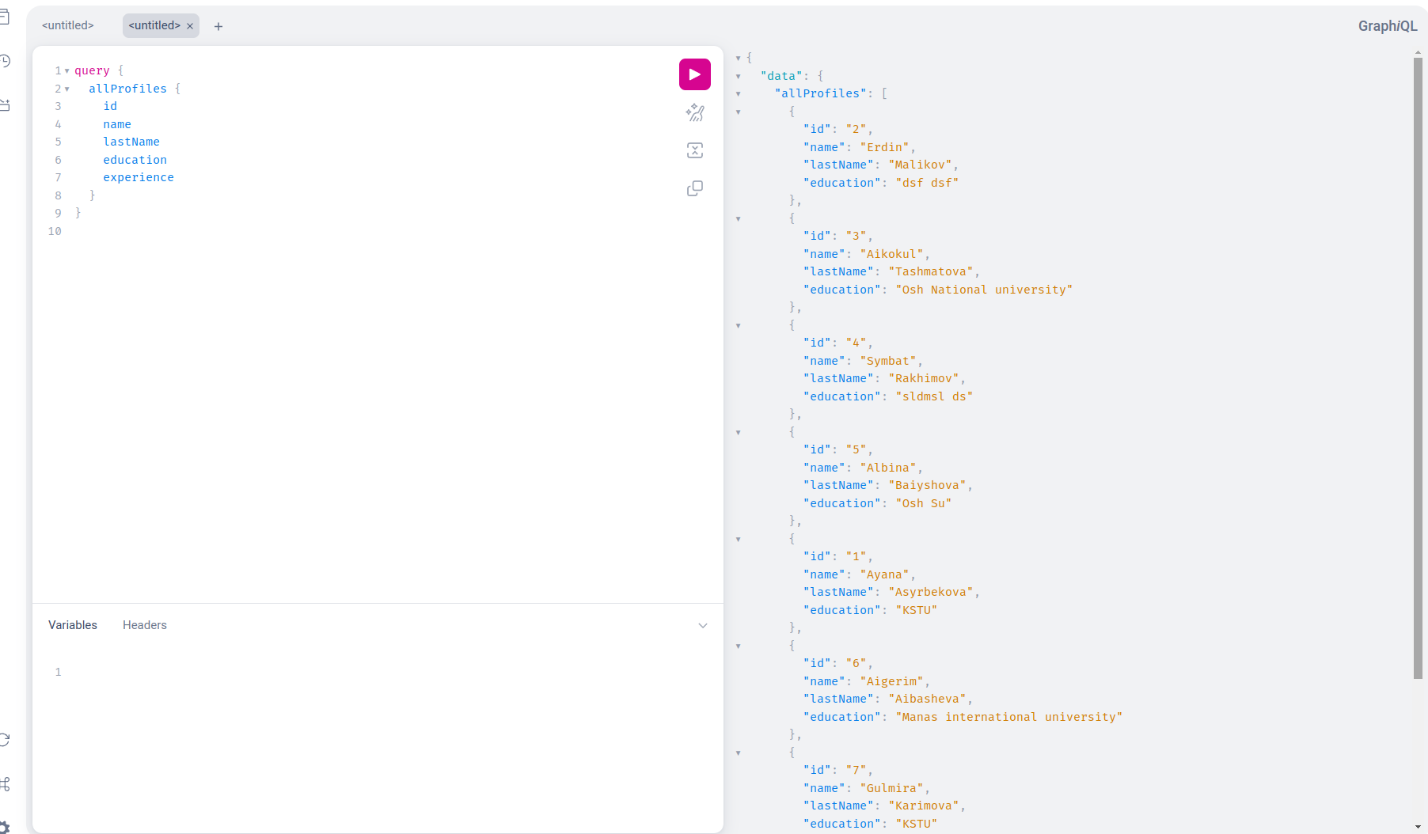
education

experience

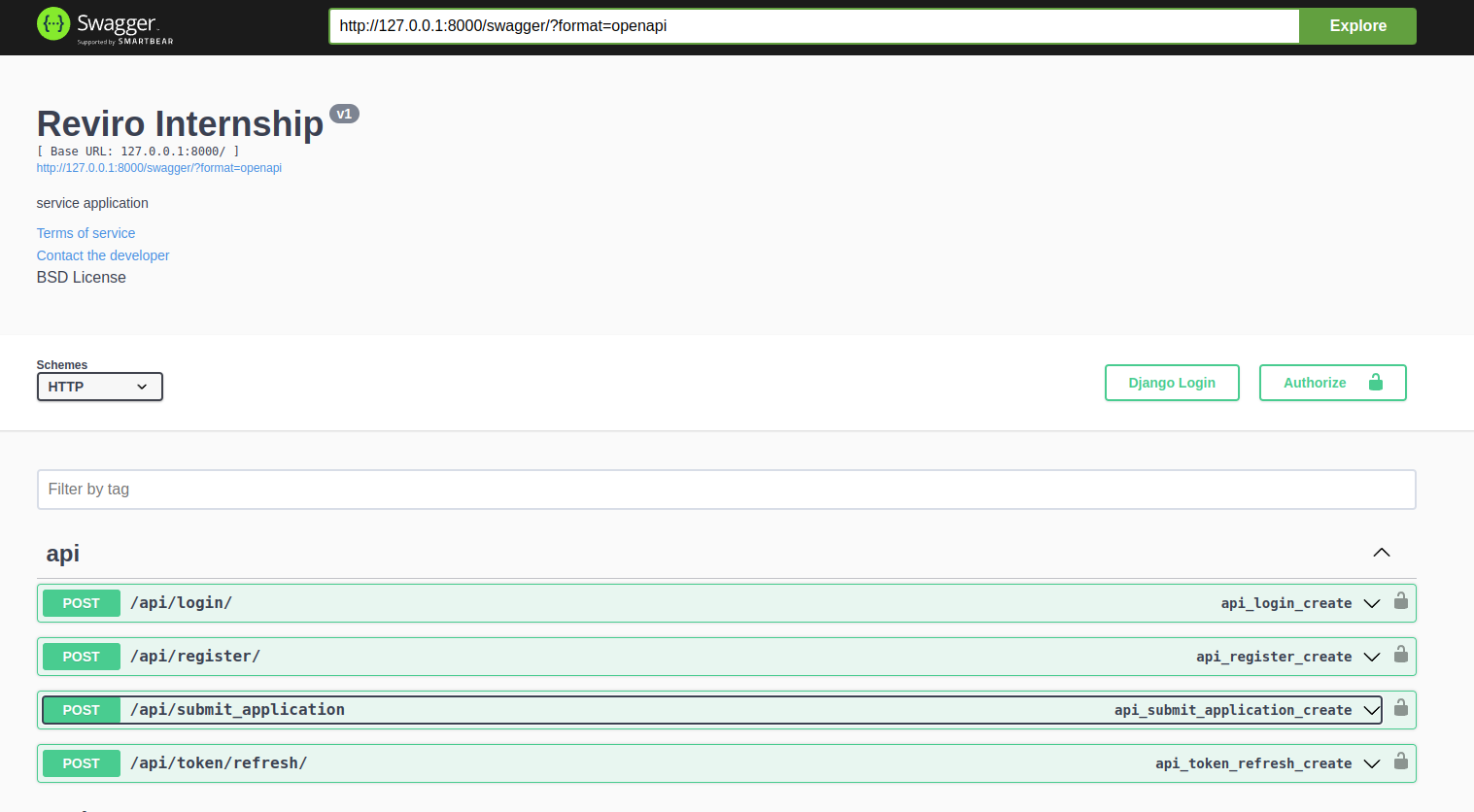
}

}

Этот запрос запрашивает поля id, name, lastName, education и experience для всех профилей кандидатов как показано на рисунке 18.

 Рисунок 18

**Использование Swagger для документирования API позволяет создавать удобные и понятные интерфейсы для взаимодействия с вашим сервисом. Это облегчает работу разработчиков, тестировщиков и других участников процесса разработки. Интеграция Swagger с вашим проектом Django предоставляет интерактивную документацию, которая всегда актуальна и отражает текущую структуру и функциональность API.**

 Рисунок 19

**Описание эндпоинтов Api рисунок 19:**

api/register/**:** Регистрация новых пользователей.

api/login/: Аутентификация пользователей и получение JWT-токенов.

api/token/refresh/: Обновление JWT-токенов.

api/submit\_application/: Подача заявок на стажировку.

**Описание эндпоинтов users:**

users/register**:** Регистрация новых пользователей через форму**.**

users/login: Вход пользователей через форму.

users/logout/: Выход из системы.

**Описание эндпоинтов application\_service:**

home/submit\_application/**:** Подача заявок на стажировку через форму.

home/confirmation/: Подтверждение успешной подачи заявки.

home/edit\_profile/: Редактирование профиля пользователя.

home/application/<int:application\_id>/: Просмотр деталей конкретной заявки.

home/candidate\_list/: Просмотр списка кандидатов.

Пример аутентификации через jwt токену приведены на рисунке 20 и 21:

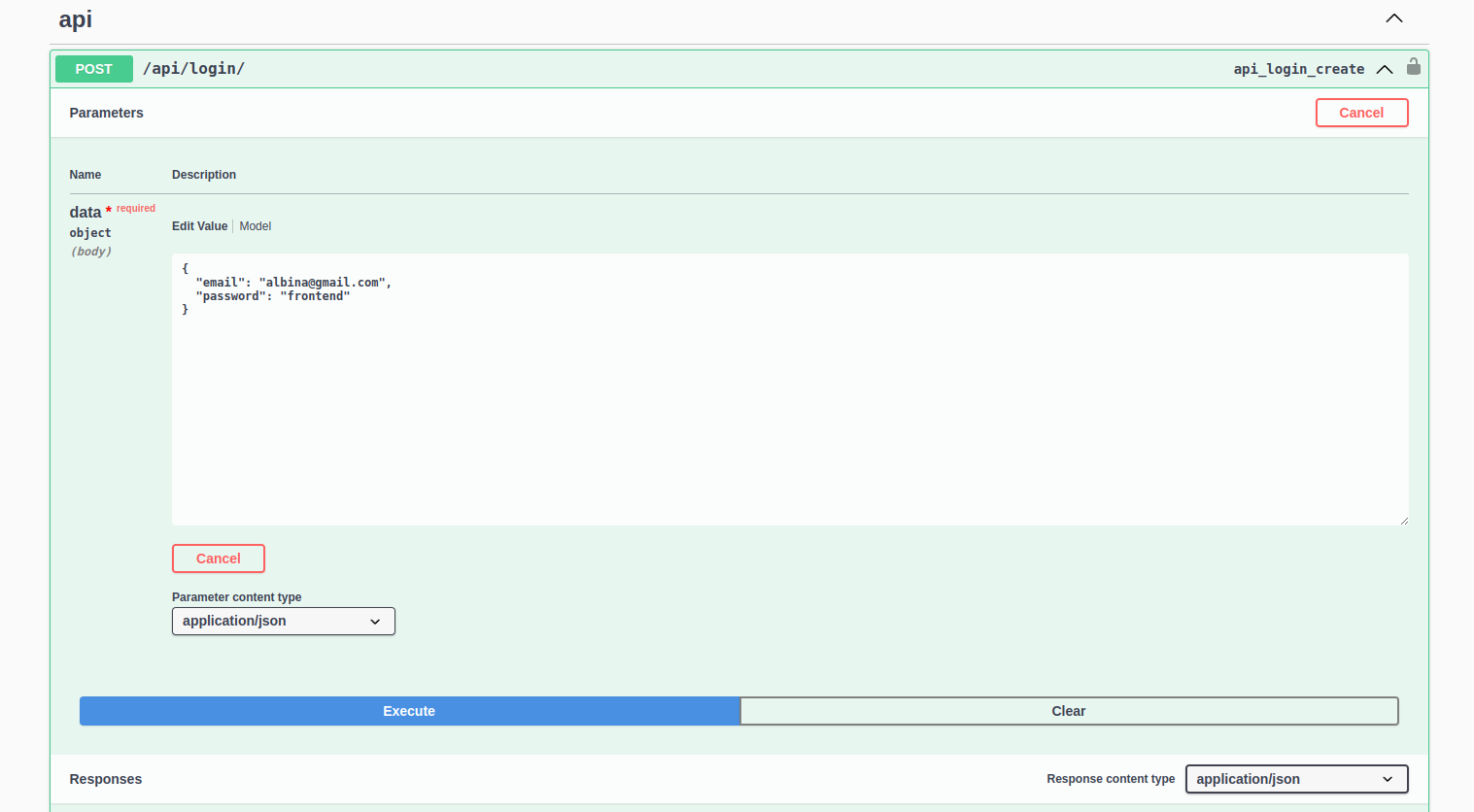
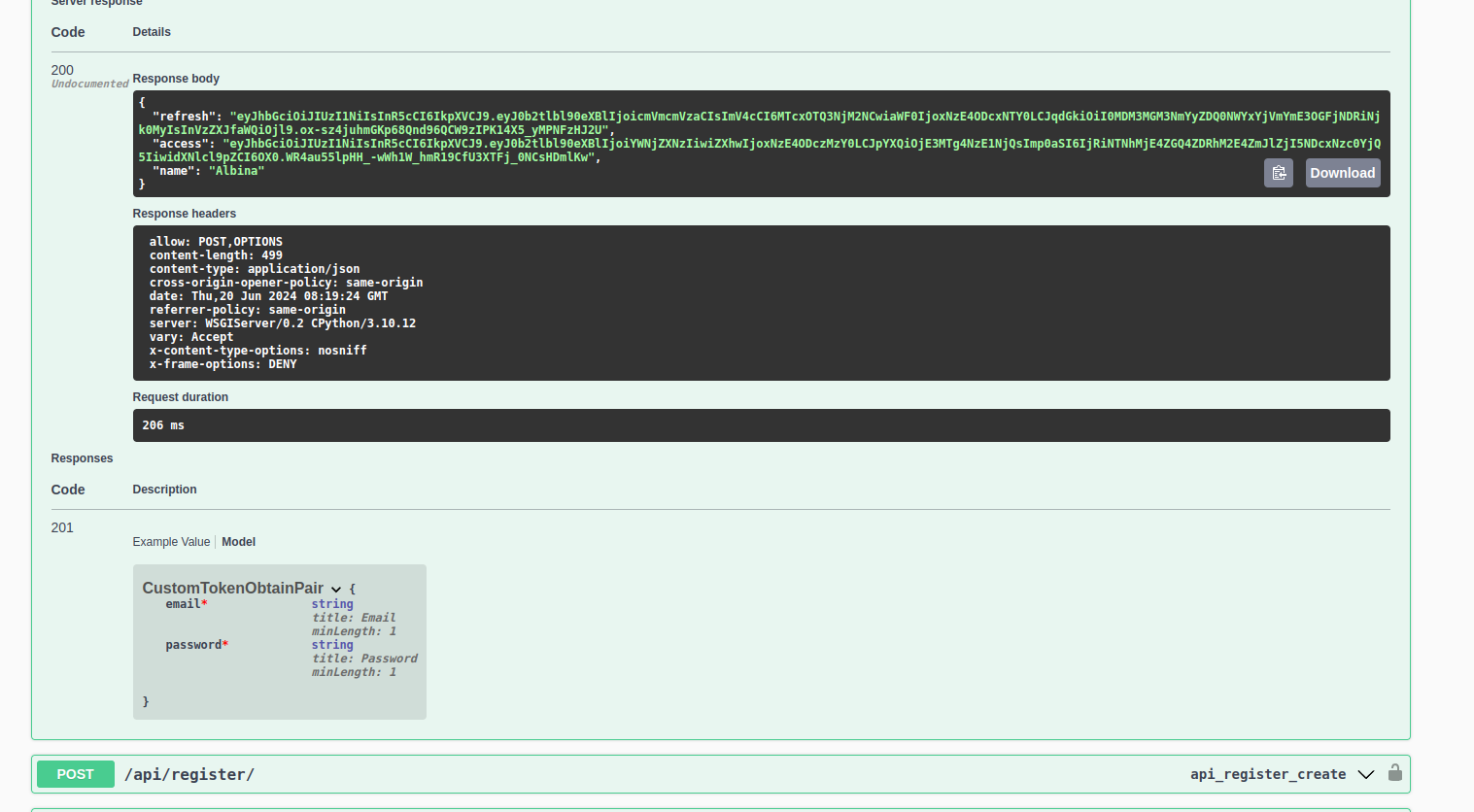
 Рисунок 20

Рисунок 21

На рисунке 21 показано что при успешной аутентификации пользователю выдается токен и сервер возвращает http ответ 201.

**Реализация Front-end:**

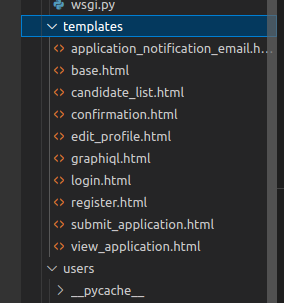
**Для реализации фронтенда веб-сервиса по приему заявок на стажировку в компанию Ревиро использованы Django Templates. Это встроенная система шаблонов, которая позволяет создавать динамически генерируемые HTML-страницы с помощью данных, передаваемых из backend на Django. Django Templates предоставляет мощный и гибкий способ создания пользовательского интерфейса.**

#### Преимущества использования Django Templates:

1. Интеграция с Django: Django Templates тесно интегрируются с самим

фреймворком Django, что позволяет легко передавать данные из представлений (views) в шаблоны.

1. Безопасность: Django автоматически экранирует данные для предотвращения атак типа XSS (межсайтовый скриптинг).
2. Простота использования: Система шаблонов Django использует простой синтаксис, который легко изучить и использовать.
3. Расширяемость: Возможность наследования шаблонов и использования блоков позволяет создавать сложные макеты страниц с минимальными усилиями.

****

**Рисунок 22**

### Основные шаблоны, используемые в проекте:

1. application\_notification\_email.html:
   * Описание: Этот шаблон используется для создания уведомлений по электронной почте, отправляемых кандидатам. Он содержит информацию о статусе их заявки или другой важной информации.
2. base.html:
   * Описание: Основной шаблон, который служит базовым для всех других шаблонов. Он содержит общие элементы, такие как заголовок, навигационная панель и подвал, которые могут быть унаследованы и расширены другими шаблонами.
3. candidate\_list.html:
   * Описание: Шаблон для отображения списка кандидатов. Он используется для визуализации данных о всех кандидатах, подавших заявки на стажировку.
4. confirmation.html:
   * Описание: Шаблон страницы подтверждения успешной подачи заявки. Он информирует пользователя о том, что его заявка была успешно отправлена.
5. edit\_profile.html:
   * Описание: Шаблон для редактирования профиля пользователя. Он содержит форму для обновления личной информации, такой как имя, контактные данные и т.д.
6. graphiql.html:
   * Описание: Шаблон интерфейса GraphiQL. GraphiQL предоставляет встроенную среду для разработки и тестирования запросов к GraphQL API.
7. login.html:
   * Описание: Шаблон для страницы входа. Он содержит форму для аутентификации пользователей.
8. register.html:
   * Описание: Шаблон для страницы регистрации. Он содержит форму для создания нового аккаунта пользователя.
9. submit\_application.html:
   * Описание: Шаблон для подачи заявки на стажировку. Он содержит форму, в которой кандидаты могут вводить необходимые данные, такие как резюме, сопроводительное письмо и выбранную позицию.
10. view\_application.html:
    * Описание: Шаблон для просмотра подробной информации о конкретной заявке. Он отображает данные кандидата и документы, приложенные к заявке.

**Использование Django Templates для фронтенда обеспечивает эффективное и удобное создание динамических веб-страниц, интегрированных с серверной частью приложения на Django. Эта система шаблонов позволяет быстро и безопасно разрабатывать пользовательские интерфейсы, что делает её идеальным выбором для проектов, требующих тесной интеграции между фронтендом и бекендом.**

### 3.2 Руководство пользователя

#### Описание проекта "Система приема заявок на стажировку компании Ревиро"

Проект "Система приема заявок на стажировку компании Ревиро" представляет собой веб-сервис, предназначенный для автоматизации процессов приема и управления заявками на стажировку в компании Ревиро. Система позволяет кандидатам регистрироваться, заполнять профили и подавать заявки на стажировку, а администраторам — управлять кандидатами и их заявками.

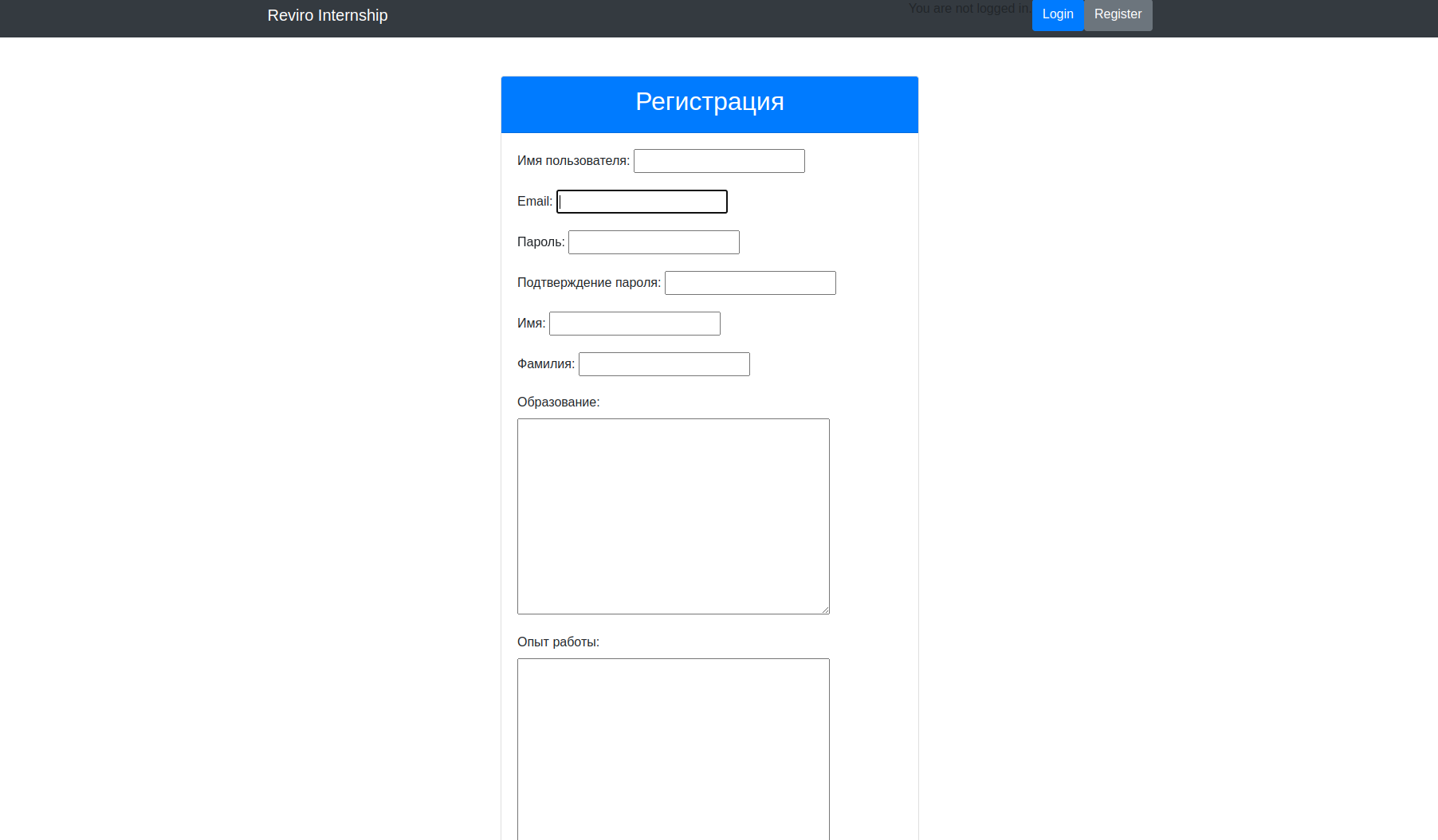
#### Цель руководства пользователя

Настоящее руководство предназначено для пользователей системы "Система приема заявок на стажировку компании Ревиро". В нем описываются основные функции системы и способы их использования.

### Основные функции системы

#### 1. Регистрация и аутентификация

Пользователи могут регистрироваться (рис 23 и24) в системе и проходить аутентификацию для получения доступа к функционалу. Регистрация и аутентификация реализованы с использованием JWT и сессий для обеспечения безопасности.

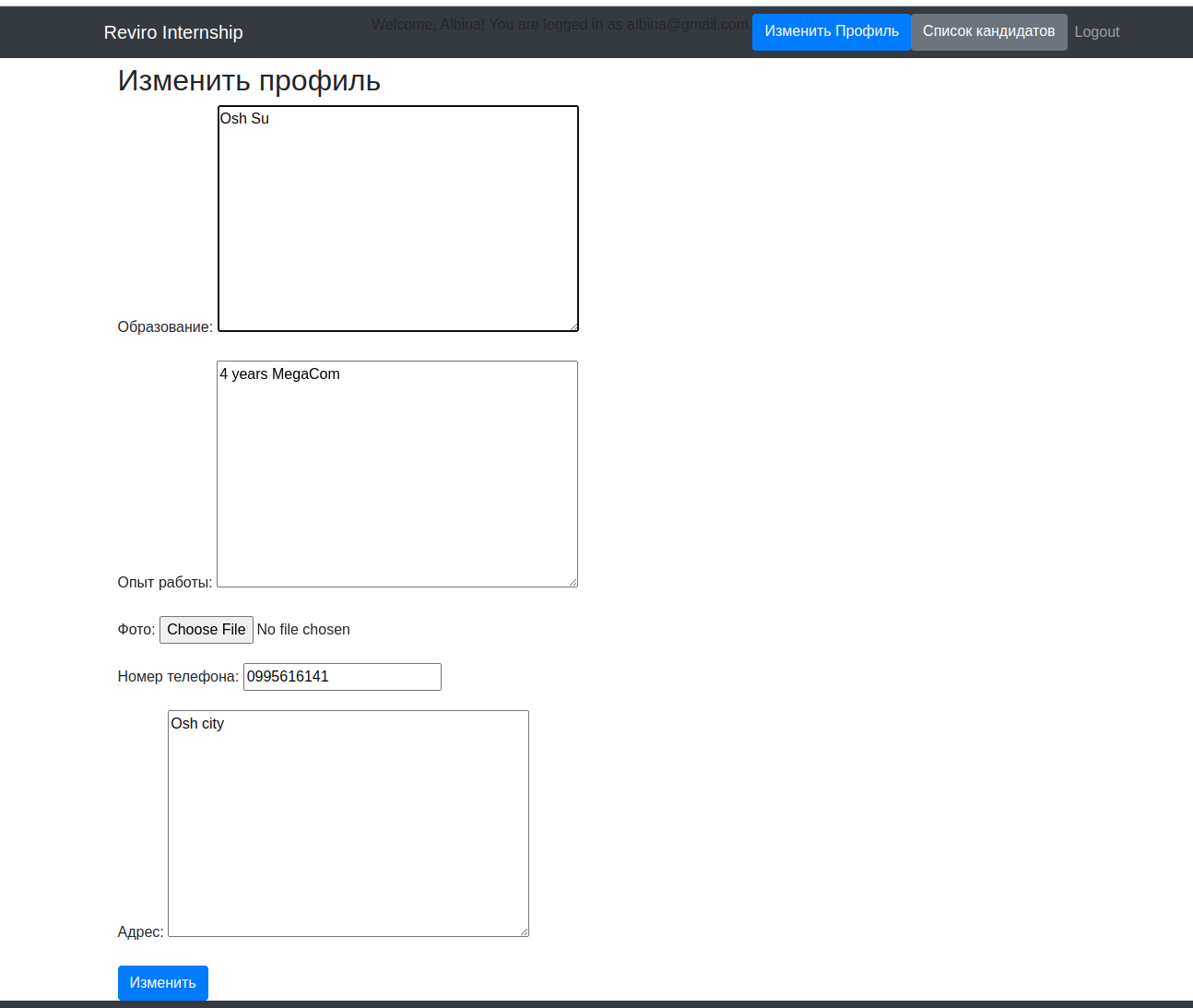
* Регистрация: Пользователи могут создать аккаунт, заполнив форму регистрации. Необходимо указать имя, адрес электронной почты и пароль.
* Аутентификация: Пользователи могут войти в систему, используя свой адрес электронной почты и пароль. После успешной аутентификации пользователь получает доступ к личному профилю и другим функциям системы.

#### 

#### Рисунок 23 и24

#### 2. Личный профиль

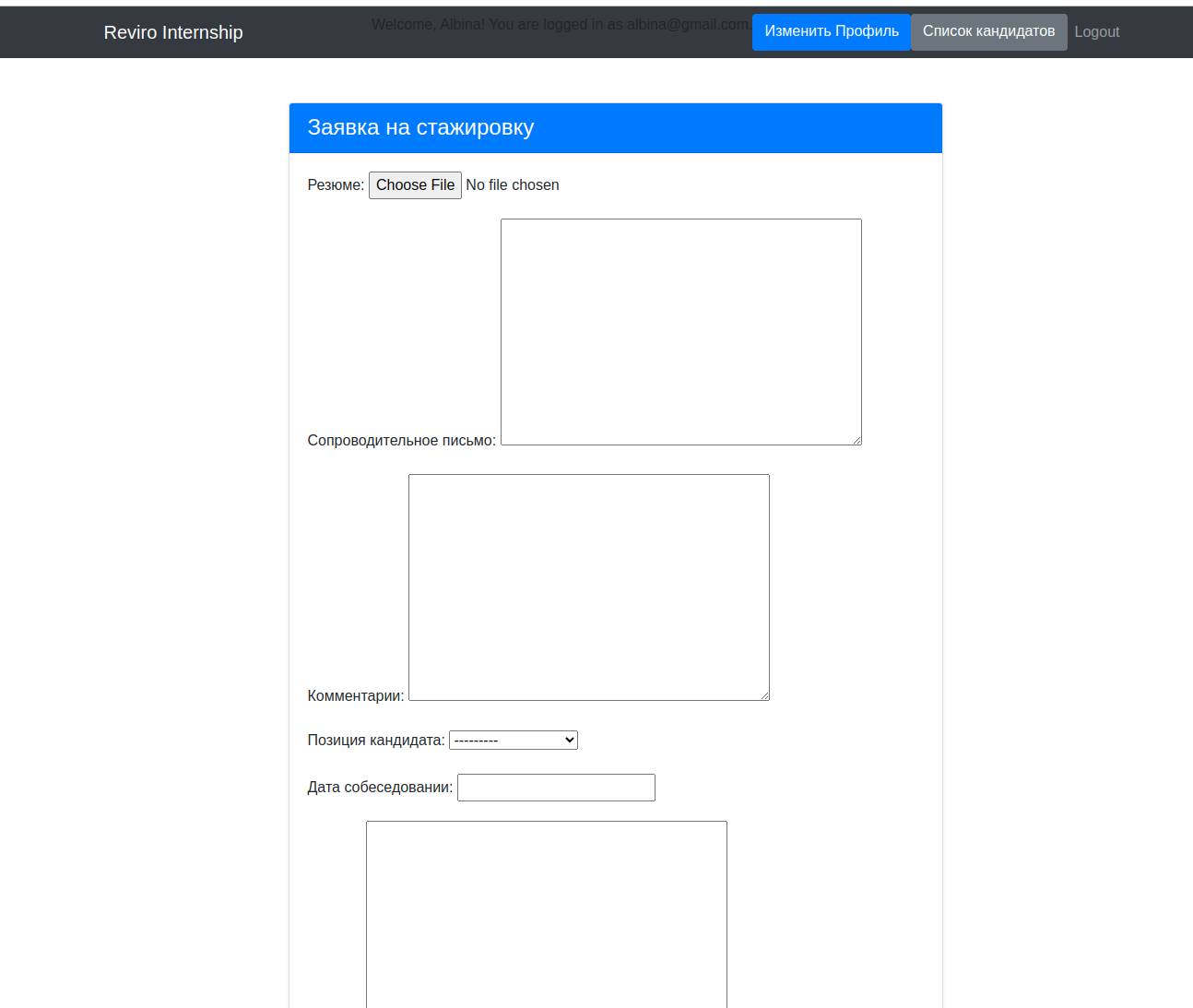
После аутентификации пользователи могут заполнить и редактировать свой личный профиль. Профиль содержит следующую информацию:

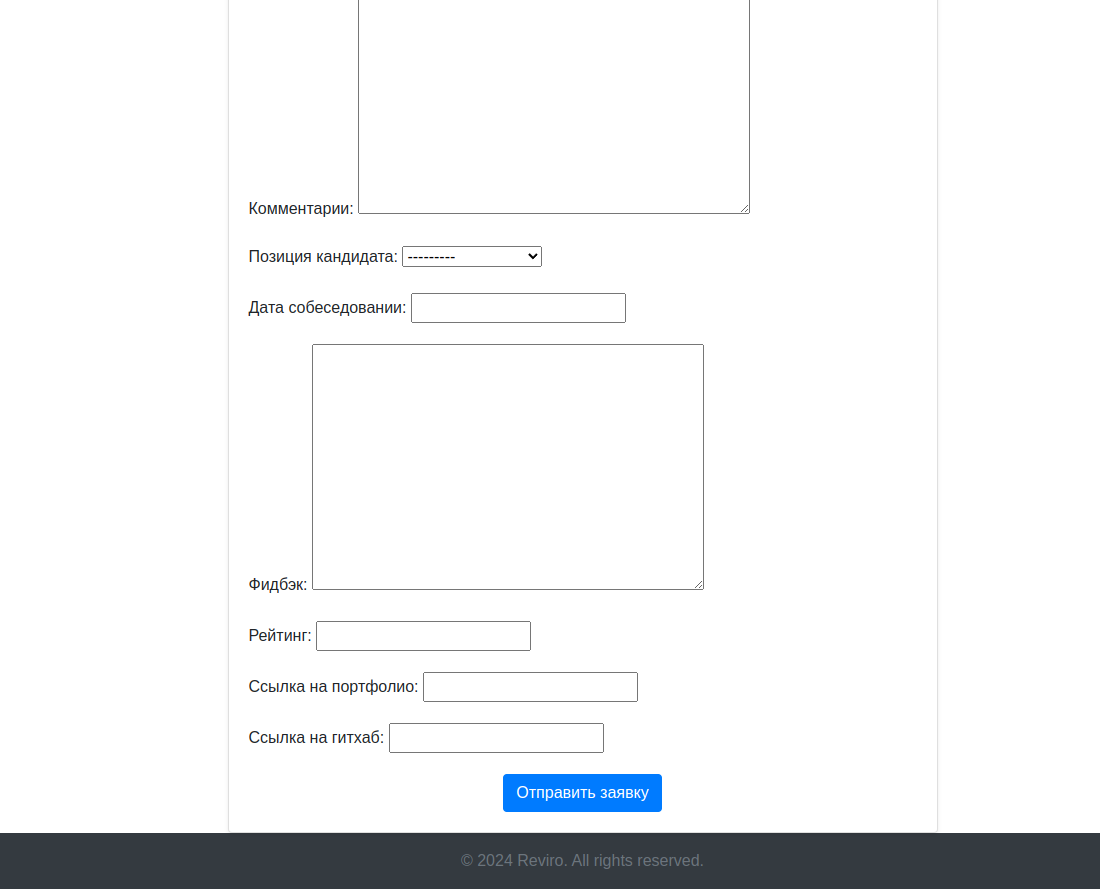
* Имя
* Фамилия
* Образование
* Опыт работы
* Фото
* Номер телефона
* Адрес

#### 3. Подача заявок на стажировку

Аутентифицированные пользователи могут подавать заявки на стажировку, заполняя соответствующую форму. В форме необходимо указать следующие данные:

* Резюме (загрузка файла)
* Сопроводительное письмо
* Желаемая позиция (Backend, Frontend, Mobile Dev, DevOps, QA, Project Manager)
* Ссылка на портфолио
* Ссылка на GitHub
* Дополнительные комментарии





### Ответ после успешной отправки заявки

После успешной отправки заявки на стажировку в компании Ревиро, клиент получает следующее уведомление как показано на картинке 25:

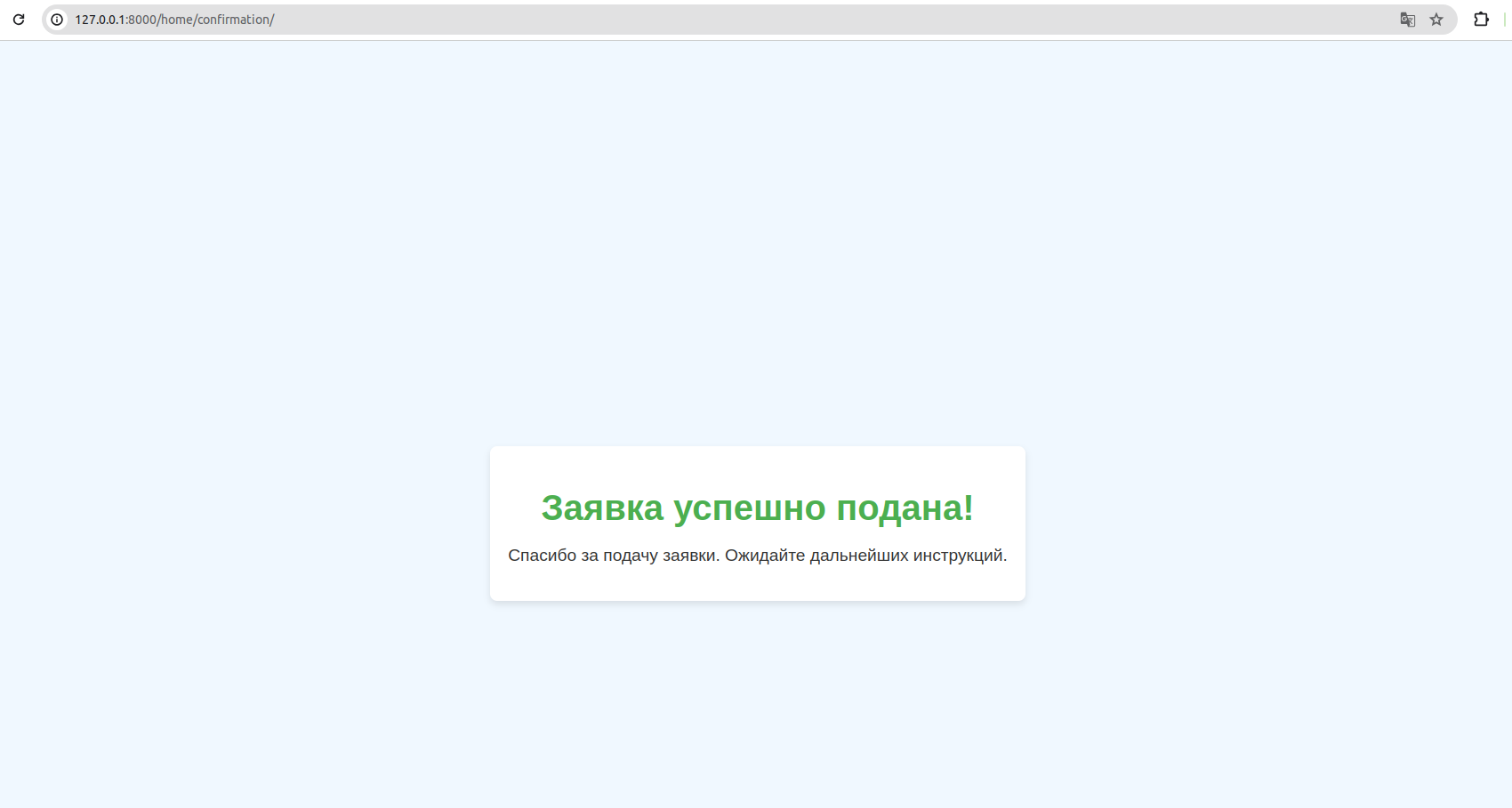
Заявка успешно подана!

Сообщение выглядит следующим образом:

* Текст на экране гласит: "Заявка успешно подана! Спасибо за подачу заявки. Ожидайте дальнейших инструкций."
* Этот текст отображается в центре экрана на белом фоне, что делает его легко заметным и читаемым.
* Дизайн уведомления прост и лаконичен, что подчеркивает успешность операции и не отвлекает внимание пользователя от основного сообщения.

### Пояснение

Это уведомление служит для подтверждения того, что заявка была успешно получена и находится в обработке. Пользователь информируется о том, что дальнейшие инструкции будут предоставлены позднее, что помогает ему понять последующие шаги и ожидания от процесса подачи заявки.

 Рисунок 25

#### 4. Управление заявками (для администраторов)

Администраторы системы имеют возможность просматривать, редактировать и управлять заявками кандидатов. Они могут просматривать детали каждой заявки, оставлять комментарии, назначать собеседования и предоставлять обратную связь.

### Инструкция по использованию системы

#### Регистрация и вход в систему

1. Перейдите на страницу регистрации.
2. Заполните форму, указав имя, адрес электронной почты и пароль.
3. Нажмите кнопку "Зарегистрироваться".
4. Для входа в систему перейдите на страницу входа.
5. Введите свой адрес электронной почты и пароль.
6. Нажмите кнопку "Войти".

#### Заполнение и редактирование профиля

1. После входа в систему перейдите в раздел "Профиль".
2. Заполните все необходимые поля: имя, фамилия, образование, опыт работы, фото, номер телефона и адрес.
3. Нажмите кнопку "Сохранить" для сохранения изменений.

#### Подача заявки на стажировку

1. Перейдите в раздел "Подать заявку".
2. Заполните форму, указав все необходимые данные.
3. Загрузите резюме и заполните сопроводительное письмо.
4. Выберите желаемую позицию.
5. Укажите ссылку на портфолио и GitHub.
6. Нажмите кнопку "Отправить заявку".

#### Просмотр и управление заявками (для администраторов)

1. Войдите в систему как администратор.
2. Перейдите в раздел "Список заявок".
3. Выберите заявку для просмотра деталей.
4. Просмотрите информацию о кандидате, резюме, сопроводительное письмо и другие данные.
5. Оставьте комментарий, назначьте собеседование или предоставьте обратную связь.

# **Приложения**

## **Приложение 1.** Листинг программы

Структура файлов и директорий в Django проекте организована таким

образом, чтобы обеспечить удобство разработки, сопровождения и расширяемости.

Django следует принципу разделения ответственности (Separation of Concerns), что означает разбиение проекта на логические части, каждая из которых отвечает за свою собственную задачу. Это помогает сделать код более модульным и легким в управлении.

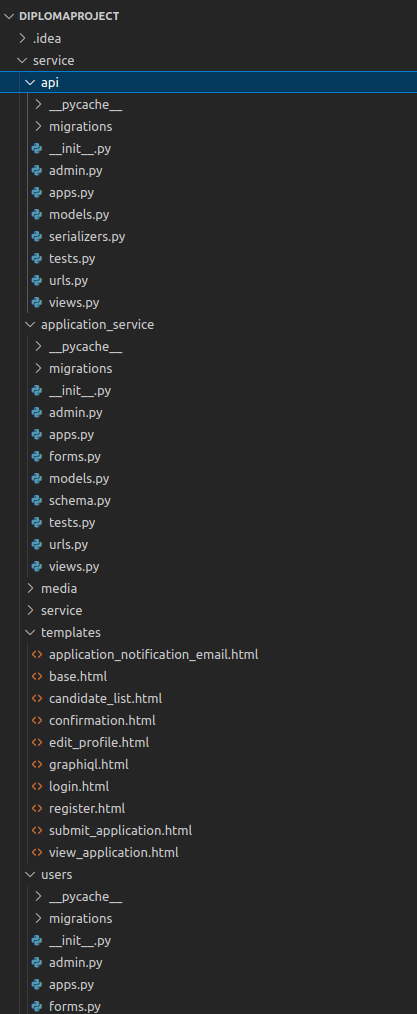


Рис 26 Структура веб-сервиса

1. models.py: определение моделей данных.
2. views.py: определение представлений, которые обрабатывают запросы и возвращают ответы.
3. forms.py: определение форм для ввода данных пользователем.
4. urls.py: маршрутизация URL-адресов к соответствующим представлениям.
5. admin.py: конфигурация административного интерфейса.
6. apps.py: конфигурация приложения.

В папке users хранится код для работы с аккаунтами, профилями, сессиями и пользователями.

В папке service хранится все настройки сервера для его работы и запуска.