

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

09.04.01 «Информатика и вычислительная техника»

Проектирование и реализация веб-приложения
«Рекрутинговый портал ITFinder»

КУРСОВАЯ РАБОТА

по дисциплине:
Технологии разработки ПО

Исполнитель:

студент группы

8BM22

Ямкин Н.Н.

18.05.2023

Руководитель:

преподаватель

Поляков А.Н.

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ	2
ВВЕДЕНИЕ.....	3
ТРЕБОВАНИЯ К ПРОГРАММЕ	6
Назначение программы.....	6
Область применения	6
Задачи, решаемые программой	6
Функциональные требования.....	6
Требования к надежности.....	7
Требования к эргономике и технической эстетике	8
Требования к программной документации.....	9
Стадии и этапы разработки	9
Диаграмма вариантов использования	10
Создание проекта	13
Создание навыка.....	14
АНАЛИЗ	15
Диаграмма классов.....	16
Диаграмма состояний	17
ПРОЕКТИРОВАНИЕ	19
Диаграмма проектных классов	19
Диаграмма пакетов.....	21
Диаграмма последовательностей.....	22
РЕАЛИЗАЦИЯ	26
Разработка приложения	26
Тестирование	27
Модульные тесты	28
Интеграционные тесты	29
Построение и выполнение тестов.....	30
Покрытие кода.....	30
Запуск приложения для тестирования.....	31
ДОКУМЕНТАЦИЯ	32
Назначение программы.....	32
Условия запуска программы	32
Выполнение программы	33
ЗАКЛЮЧЕНИЕ	42
СПИСОК ИСТОЧНИКОВ.....	43

ВВЕДЕНИЕ

Сегодня веб-приложения крайне важны для бизнеса, так как способствуют его развитию, а следовательно, и увеличению прибыли.

Собственное веб-приложение может помочь во многих аспектах предпринимательской деятельности. Ниже приведены некоторые из них:

1. Увеличение узнаваемости бренда: с помощью веб-приложения можно расширить свое онлайн-присутствие и увеличить узнаваемость своего бренда. Пользователи могут быстро и легко найти информацию о компании, её товарах и услугах, а также связаться с офисом.

2. Улучшение клиентского опыта: хороший сайт предоставляет удобный и интуитивно понятный интерфейс, который позволяет пользователям быстро находить нужную информацию и выполнять целевые действия, такие как покупка товаров или заказ услуг. Таким образом, можно многократно увеличить продажи.

3. Автоматизация бизнес-процессов: благодаря собственному интернет-ресурсу можно автоматизировать многие бизнес-процессы, такие как обработка заказов, управление складом и учет финансов. Это позволяет сотрудникам работать более эффективно и повышает производительность бизнеса в целом.

4. Расширение аудитории: веб-приложения доступны для использования из любой точки мира, что позволяет компании привлекать новых клиентов и расширять свою аудиторию.

5. Сбор и анализ данных: онлайн-платформы могут собирать данные о пользователях и их поведении, что позволяет предприятию анализировать эти данные и использовать полученную информацию для улучшения своих продуктов и услуг.

В результате, веб-приложение может помочь бизнесу, причем неважно малый он, средний или крупный, повысить эффективность работы, улучшить клиентский опыт и увеличить продажи, что в итоге приведет к росту прибыли.

В частности, рассмотрим основные преимущества рекрутинга сотрудников онлайн:

1. Размещение вакансий в интернете.
2. Онлайн-тестирование и собеседования. Это может упростить и ускорить процесс найма, а также позволить компании работать с кандидатами из разных географических регионов.
3. Автоматизация процессов отбора: рекрутинговые платформы могут использоваться для автоматизации процессов отбора кандидатов. Например, они могут использоваться для сбора данных о кандидатах, анализа резюме и определения наиболее подходящих кандидатов.
4. Снижение затрат на рекрутинг благодаря экономии на печати и распространении объявлений о вакансиях, организацию собеседований и тестирования.
5. Сохранение информации о кандидатах. Данное преимущество может быть полезно на более длительном периоде времени и использоваться в случае появления новых вакансий, на которые могут быть рассмотрены сохраненные ранее резюме кандидатов.

Цель работы: разработка веб-приложения для рекрутинга «ITFinder», удовлетворяющего исходным требованиям.

Объект исследования: рекрутинг сотрудников.

Предмет исследования: онлайн-рекрутинг сотрудников.

Данный проект был разработан на языке Python. Python – интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой.

Для написания кода приложения используется среда разработки PyCharm.

Для создания приложения использовалась платформа Django. С помощью простой и надежной платформы Django, основанной на языке Python, можно создавать мощные веб-решения, написав всего несколько строк

кода. Ее использование позволяет обеспечить высокую скорость и гибкость разработки, а также решать широкий спектр прикладных задач.

По умолчанию Django в качестве базы данных использует SQLite. Она очень проста в использовании и не требует запущенного сервера. Все файлы базы данных могут легко переноситься с одного компьютера на другой. Однако при необходимости мы можем использовать в Django большинство распространенных СУБД.

ТРЕБОВАНИЯ К ПРОГРАММЕ

Назначение программы

Сайт «ITFinder» предназначен поиска и найма специалистов в сфере IT. Система поиска позволяет подобрать нужного специалиста по резюме, портфолио или описанию проектов; рейтинг проекта поможет оценить квалификацию, а мессенджер – послать оффер.

Область применения

Разрабатываемый программный продукт предназначен для частных лиц и организаций желающих найти и привлечь кандидатов на работу.

Задачи, решаемые программой

Для обеспечения возможности решать требуемые задачи с помощью данного приложения, необходимо соблюдать следующие требования:

1. Язык реализации – Python с интерпретатором версии 3.9;
2. Фреймворк Django версии 4.0.6;
3. СУБД – SQLite 3;
4. Основной язык программы – русский;
5. Система должна производить авторизацию пользователя не более, чем в течение трех секунд;
6. Данная система должна разрабатываться и функционировать в рамках существующего законодательства Российской Федерации.
7. Функционал системы должен отвечать соответствующим заявленным требованиям.

Функциональные требования

1. Программа «ITFinder» (далее система) должна предоставлять полный список кандидатов, добавленных в данную систему, с отображением

следующих параметров: изображение самого кандидата, информация о его/её опыте и навыках, портфолио из выполненных проектов;

2. Система должна предоставлять полный список проектов, добавленных в данную систему, с отображением следующих параметров: логотип проекта, информация о его разработчике, описание проекта, отзывы, количество положительных отзывов, общее количество отзывов, стек используемых технологий (теги);

3. Система должна предоставлять возможность регистрации новых пользователей;

4. Система должна предоставлять возможность аутентификации пользователя;

5. Система должна предоставлять возможность авторизации пользователя;

6. Система должна предоставлять возможность поиска по профилям, навыкам, тегам, описаниям проектов;

7. Система должна предоставлять возможность производить операции CRUD создания, редактирования, удаления профилей, проектов, тегов проекта и навыков разработчика.

8. Система должна предоставлять возможность отправки сообщений другим пользователям;

9. Система должна предоставлять возможность оценивать проекты и оставлять отзывы к ним.

Требования к надежности

Надежность веб-приложения является важным качественным показателем, который определяет, насколько хорошо приложение функционирует и сохраняет работоспособность в различных условиях. Основные требования к надежности веб-приложения включают:

1. Доступность: система должна быть доступна для пользователей в любое время суток. Для этого необходимо обеспечить стабильную работу

серверов, на которых запущено приложение, и оптимизировать производительность приложения, чтобы оно могло обрабатывать большое количество запросов без сбоев.

2. Устойчивость к ошибкам: система должна быть способна обрабатывать ошибки и сбои в работе, чтобы не останавливаться полностью при возникновении каких-либо проблем.

3. Безопасность: система должна обеспечивать безопасность пользователей, защищая их данные от несанкционированного доступа или взлома. Для этого необходимо внедрить аутентификацию и авторизацию пользователей.

4. Масштабируемость: система должна обеспечивать масштабируемость, то есть способность обрабатывать большое количество запросов при росте числа пользователей или объема данных, без снижения производительности. Для этого необходимо использовать современные технологии и архитектуру, которые позволяют горизонтально масштабировать приложение.

5. Поддержка: Веб-приложение должно иметь достаточную техническую поддержку, чтобы быстро реагировать на проблемы, которые могут возникнуть у пользователей.

6. Интерфейс должен быть интуитивно понятен и приветлив. Он не должен изобиловать большой палитрой цветов, отвлекать пользователя.

Требования к эргономике и технической эстетике

Взаимодействие пользователя с программой должно осуществляться посредством графического интерфейса программной системы, с помощью браузера. Интерфейс программы должен быть выполнен в едином стиле, на главном экране должно находиться меню навигации по страницам приложения.

При вводе некорректных данных, должны появляться сообщения о том, что данные введены не корректно.

Требования к программной документации

Документация представлена в виде отчета по курсовой работе, который состоит из следующих частей:

1. Титульный лист, номинальный объем – 1 страница;
2. Содержание, номинальный объем – 1 страница;
3. Введение, номинальный объем – 2 страницы;
4. Требования к программе, номинальный объем – 8 страниц;
5. Анализ, номинальный объем – 8 страниц;
6. Проектирование, номинальный объем – 8 страниц;
7. Реализация, номинальный объем – 8 страниц;
8. Документация, номинальный объем – 8 страниц;
9. Заключение, номинальный объем – 1 страница;
10. Список использованных источников, номинальный объем – 1 страница.

Стадии и этапы разработки

Основные этапы разработки программы:

1. Описание требований к системе;
2. Выявление классов. Построение и описание диаграммы классов анализа;
3. Построение и описание диаграммы состояний;
4. Построение и описание проектных классов;
5. Построение и описание диаграмм последовательности для операций проектных классов;
6. Построение и описание диаграммы пакетов;
7. Разработка программы;
8. Модульное тестирование;
9. Документация.

Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) – это один из типов диаграмм UML, используемых для моделирования поведения системы. Она помогает описать функциональные требования к системе и процессы взаимодействия между ее компонентами и пользователями [1].

Диаграмма вариантов использования состоит из актеров (пользователей системы) и вариантов использования (use case), которые представляют сценарии использования системы.

Каждый вариант использования описывает действия пользователя и системы, которые происходят при выполнении определенной функции. Варианты использования могут быть связаны друг с другом, образуя более сложные процессы.

Диаграмма вариантов использования является полезным инструментом для понимания требований к системе и дает возможность легко визуализировать все возможные сценарии ее использования.

Ниже представлены спецификации некоторых вариантов использования, изображенных на рисунке 1 и 2.

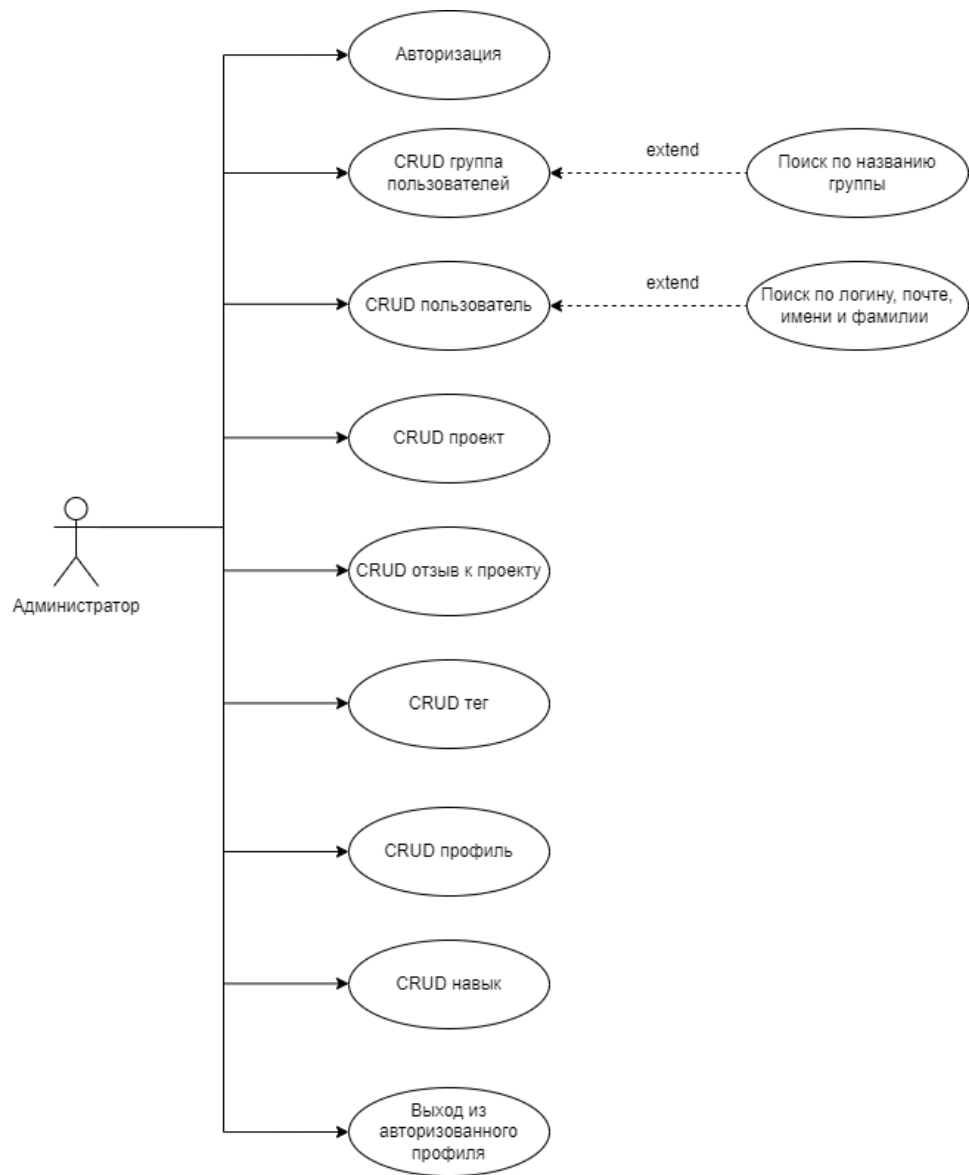


Рисунок 1 – Use Case Diagram для пользователя Администратор

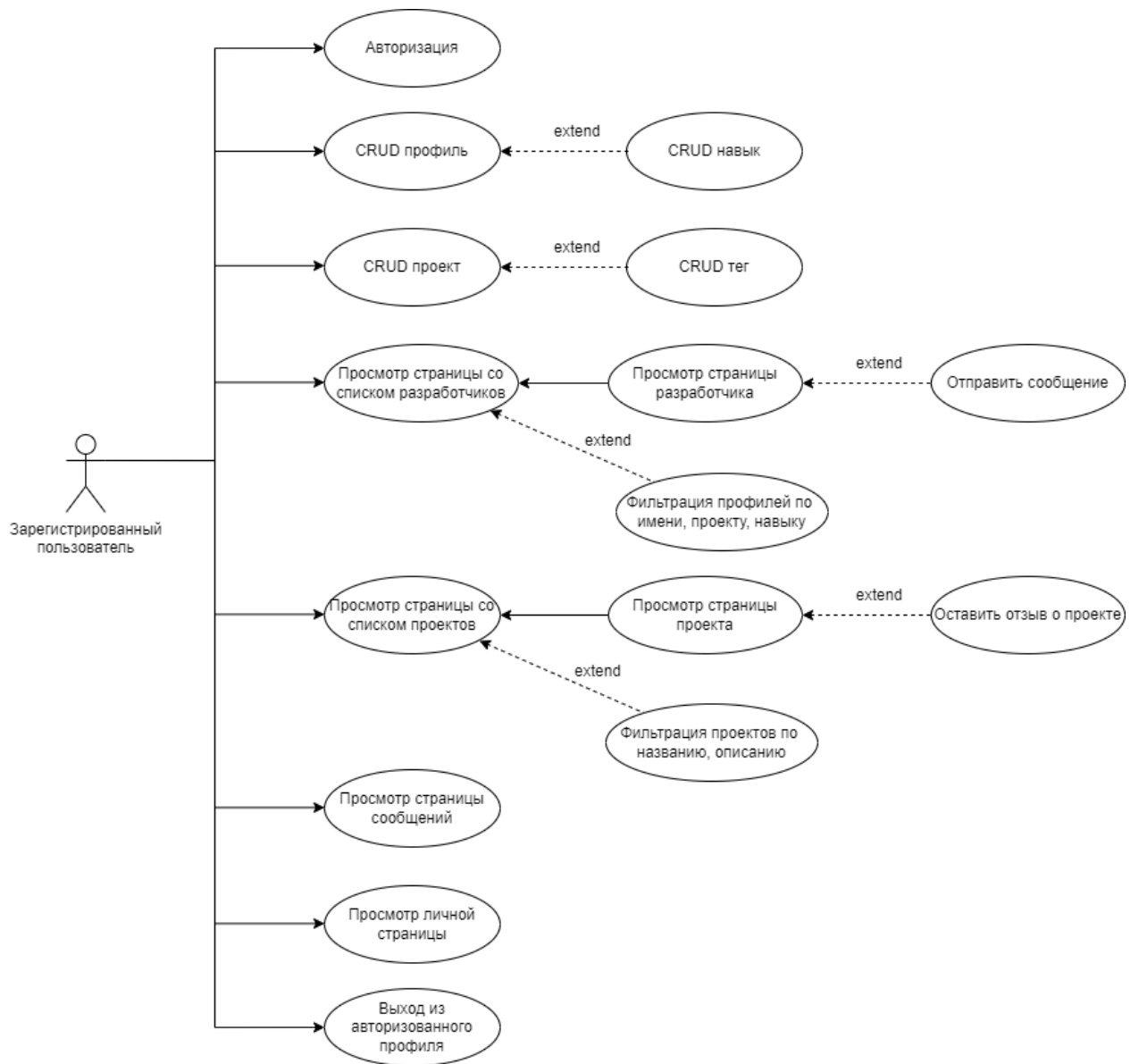


Рисунок 2 – Use Case Diagram для зарегистрированного пользователя

Создание проекта

Роль	Зарегистрированный пользователь (П.)
Пре-условие	П. авторизовался в системе (С.) и находится на странице «Мой аккаунт».
Основной сценарий	<ol style="list-style-type: none"> 1. П. выбирает опцию добавления проекта; 2. П. видит форму в отдельном окне, заполняет её: <ol style="list-style-type: none"> a. Название проекта; b. Слаг; c. Скриншот проекта; d. Теги (предоставленные по умолчанию); e. Описание проекта; f. Ссылка на GitHub; g. Теги (собственные) 3. П. выбирает опцию сохранения; 4. С. переводит пользователя обратно на страницу «Мой аккаунт».
Ожидаемый результат	П. разместил проект на личной странице и на общей странице со всеми проектами.
Альтернатива 1	Если П. пытается добавить проект с незаполненными обязательными полями, то отображается ошибка – «Заполните это поле», а опция добавления становится недоступной.
Альтернатива 2	Если П. превысил максимально допустимое количество символов в обязательных полях, то отображается ошибка – «Превышено максимальное количество допустимых символов», а опция добавления становится недоступной.
Альтернатива 3	Если П. выбирает опцию отмены добавления проекта, то он находится на том же месте, откуда была вызвана эта опция (см. пре-условие).
Пост-условие	П. находится на странице «Мой аккаунт».

Создание навыка

Роль	Администратор (А.)
Пре-условие	А. авторизовался в системе (С.) и находится на странице «Django administration».
Основной сценарий	<ol style="list-style-type: none"> 1. А. переходит на страницу «Skills»; 2. А. выбирает опцию добавления навыка; 3. А. видит форму в отдельном окне, заполняет её: <ol style="list-style-type: none"> а. Название навыка; б. Слаг; с. Описание навыка; 4. А. выбирает опцию сохранения; 5. А. видит сообщение «The skill was added successfully. You may edit it again below». 6. С. переводит пользователя на страницу «Change skill».
Ожидаемый результат	А. добавил навык на страницу «Skills»;
Альтернатива 1	Если А. пытается добавить навык с незаполненными обязательными полями, то отображается ошибка – «This field is required», а опция добавления становится недоступной.
Альтернатива 2	Если А. выбирает опцию отмены добавления навыка, то он находится на том же месте, откуда была вызвана эта опция (см. пре-условие).
Пост-условие	А. находится на странице «Django administration».

АНАЛИЗ

При разработке программного обеспечения одной из основных методологий является анализ. На данном этапе создается аналитическая модель предметной области.

Этот процесс необходим для того, чтобы понять, как должна работать система, какие функции она должна выполнять и как организовать ее разработку и реализацию.

Анализ предметной области важен, поскольку он позволяет:

1. Разработать наиболее подходящую модель для создаваемой системы, учесть все требования и особенности, которые нужны для решения задачи.

2. Сформировать правильное представление о том, как работают процессы в предметной области, чтобы сделать правильный выбор инструментов и технологий на стадии проектирования и разработки.

3. Спланировать процесс разработки ПО и выделить количества времени и ресурсов, необходимых для реализации проекта.

Анализ предметной области помогает сформулировать базовые требования и цели, определить ожидания пользователей и выделить ключевые функциональные и нефункциональные требования, что в итоге существенно сокращает количество ошибок и повышает качество программного продукта.

Существует три уровня моделирования данных:

1. Концептуальный уровень моделирования данных – это высокоуровневое описание сущностей предметной области, которые требуются для разработки системы. Он является абстрактным уровнем моделирования и описывает ключевые объекты и их атрибуты. На этом уровне моделирования используются сущности, которые существуют в предметной области, а не в программном коде. В роли инструмента на данном уровне используется ER-диаграмма и/или концептуальная диаграмма классов [5].

2. Логический уровень моделирования данных – это более детальное описание сущностей, их свойств и отношений между ними. На этом уровне уже определяются те сущности, которые будут использоваться в программном коде, и определяются требования к базе данных [5].

3. Физический уровень моделирования данных – это уровень, на котором определяются технические детали реализации базы данных, такие как формат хранения данных, индексы и т.д. В роли инструментов на данном уровне используются DDL-скрипты или физические ER-диаграммы [5].

Каждый уровень является важным компонентом процесса разработки базы данных. Он представляет собой значимый шаг к полному описанию структуры данных системы и дает возможность разработчикам лучше понимать всю необходимую функциональность, требуемую от базы данных.

Диаграмма классов

Воспользуемся диаграммой классов UML, для концептуального моделирования системы.

Диаграмма классов UML (Unified Modeling Language) представляет собой графическое представление классов, интерфейсов, объектов, атрибутов и методов, которые используются для описания объектно-ориентированных систем.

Данная нотация является одной из наиболее универсальных диаграмм UML, используемых в процессе разработки ПО, и часто представляется как первая диаграмма, которую следует создавать при моделировании системы. Она позволяет визуализировать структуру системы вместе с ее ключевыми атрибутами и методами, что очень полезно для выявления потенциальных ошибок и необходимых улучшений в ранних стадиях разработки [2].

В разрабатываемом приложении будет несколько классов, связанных друг с другом различными способами:

Один к одному – у одного пользователя может быть только один профиль.

Один ко многим – один пользователь (профиль) может быть автором множества проектов; один проект может получить множество отзывов / (комментариев и оценок).

Многие ко многим – у одного проекта может быть множество тегов, а у пользователя – несколько навыков. Один и тот же тег (навык) может использоваться во множестве проектов (или профилей).

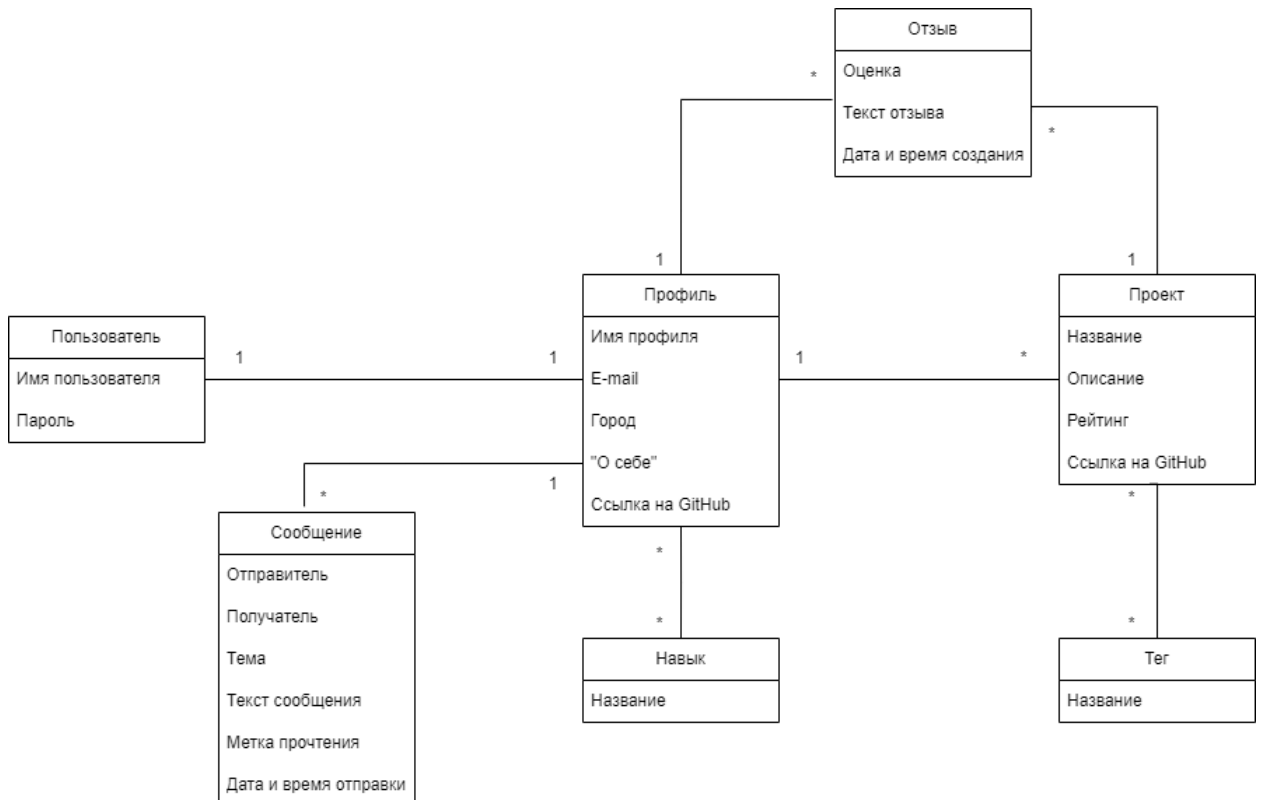


Рисунок 3 – Концептуальная диаграмма классов

Диаграмма состояний

Для большинства физических систем, кроме самых простых и тривиальных, статических представлений совершенно недостаточно для моделирования процессов функционирования подобных систем как в целом, так и их отдельных подсистем и элементов.

Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра

определенного класса – одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события [1].

На рисунке 4 представлена диаграмма конечных автоматов для класса «Сообщение» данной системы.

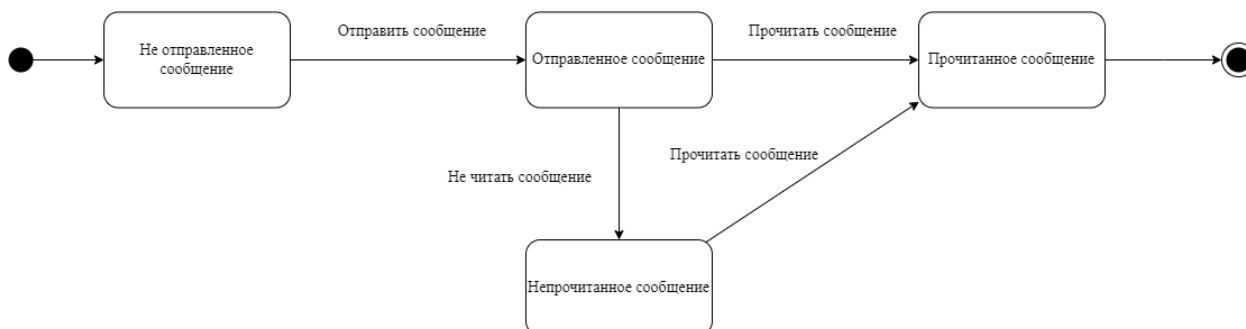


Рисунок 4 – Диаграмма состояний для класса «Сообщение»

При запросе пользователя происходит создание экземпляра класса «Сообщение», вызывается форма создания сообщения, после заполнения которой экземпляр класса «Сообщение» автоматически переходит в состояние не отправленного.

При нажатии кнопки отправки сообщения, объект переходит в состояние «Отправленное сообщение».

При игнорировании полученного сообщения, объект переходит в состояние «Непрочитанное сообщение».

При прочтении полученного сообщения, объект переходит в состояние «Прочитанное сообщение».

Созданные на данном этапе диаграммы подготовили основу для проектирования системы с учетом выбранных языков программирования, платформ и средств разработки. Данный этап является одним из самых важных.

ПРОЕКТИРОВАНИЕ

На этапе проектирования, опираясь на ранее написанную концептуальную и логическую модель, создается физическая модель, которая определяет технические детали реализации системы.

Диаграмма проектных классов

Диаграмма проектных классов UML – это структурная диаграмма, которая отображает структуру объектной модели в виде классов, интерфейсов и их связей. Физическая диаграмма классов UML показывает, как элементы объектной модели будут представлены в реальном коде, на сервере или клиенте.

Диаграмма классов UML физического уровня моделирования помогает программистам понимать и контролировать физическую структуру проекта. Данная диаграмма может быть очень полезной при разработке и реализации крупных систем, где необходимо учитывать сложные связи между компонентами [5].

На рисунке 5 приведена диаграмма проектных классов разрабатываемого приложения.

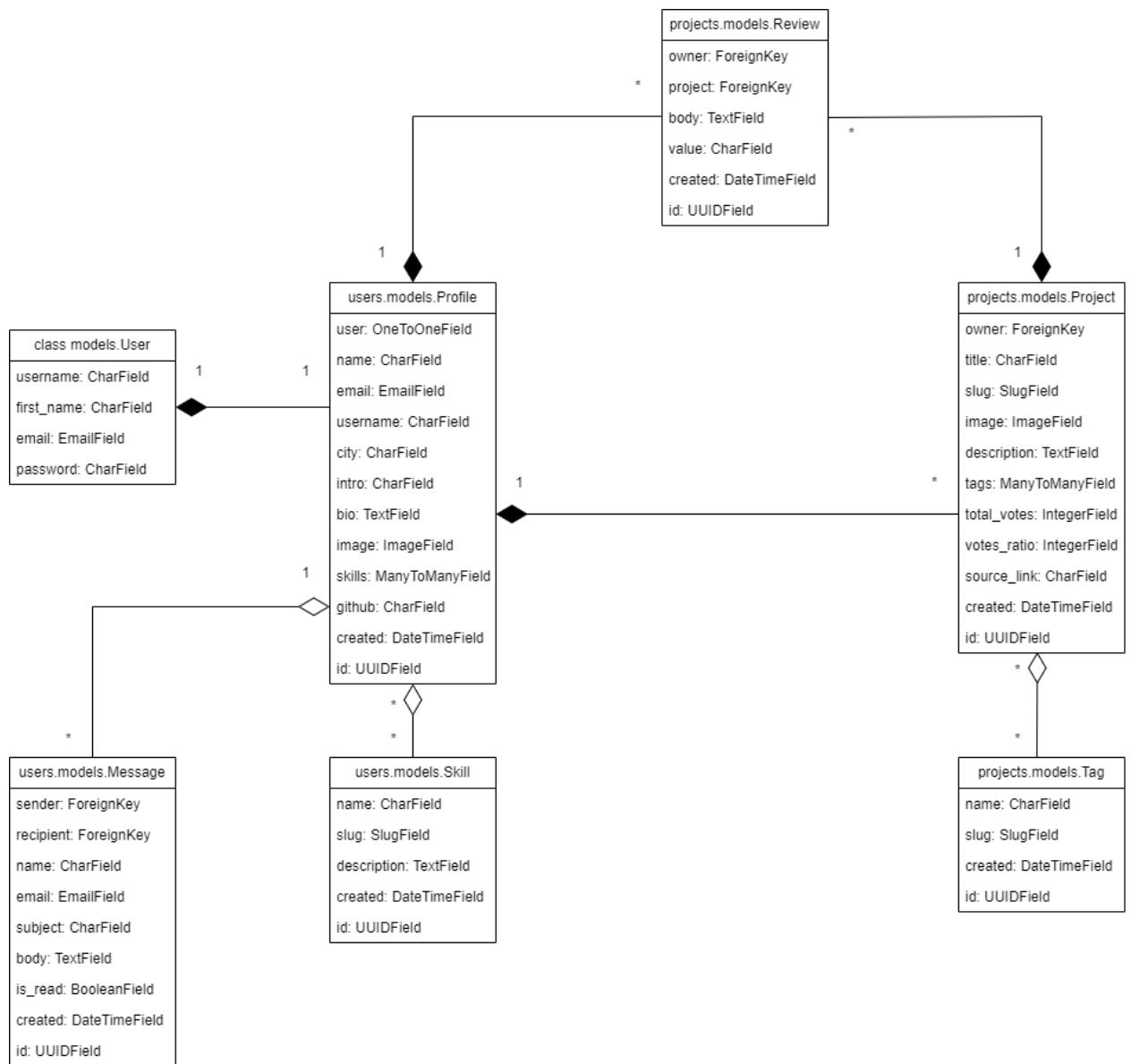


Рисунок 5 – Диаграмма проектных классов

По сравнению с концептуальной диаграммой классов, на физической диаграмме были указаны типы данных для всех атрибутов, а также были добавлены новые атрибуты в каждый из классов.

На диаграмме также проиллюстрированы отношения между сущностями. Так, например класс «User» связан композицией с классом «Profile», а класс «Project» с классом «Tag» – через агрегацию.

Диаграмма пакетов

Диаграмма пакетов UML – это диаграмма, которая показывает, как компоненты, классы или другие элементы системы организованы в логическую иерархическую структуру. Эта диаграмма помогает организовать элементы в модули, пакеты или подсистемы, что облегчает управление и поддержку проекта. Каждый пакет может содержать вложенные пакеты и элементы, а также отображать зависимости между различными пакетами. Диаграмма пакетов UML расширяет диаграмму классов, предоставляя дополнительную информацию об организации классов в проекте [3].

Пакет – это способ организации элементов модели в блоки, которыми можно распоряжаться как единым целым. Хорошо структурированный пакет объединяет семантически и концептуально близкие элементы, которые имеют тенденцию изменяться совместно [1].

Диаграмма пакетов представлена на рисунке 6. Она состоит из пакетов, содержащих в себе классы для формирования интерфейса пользователя.

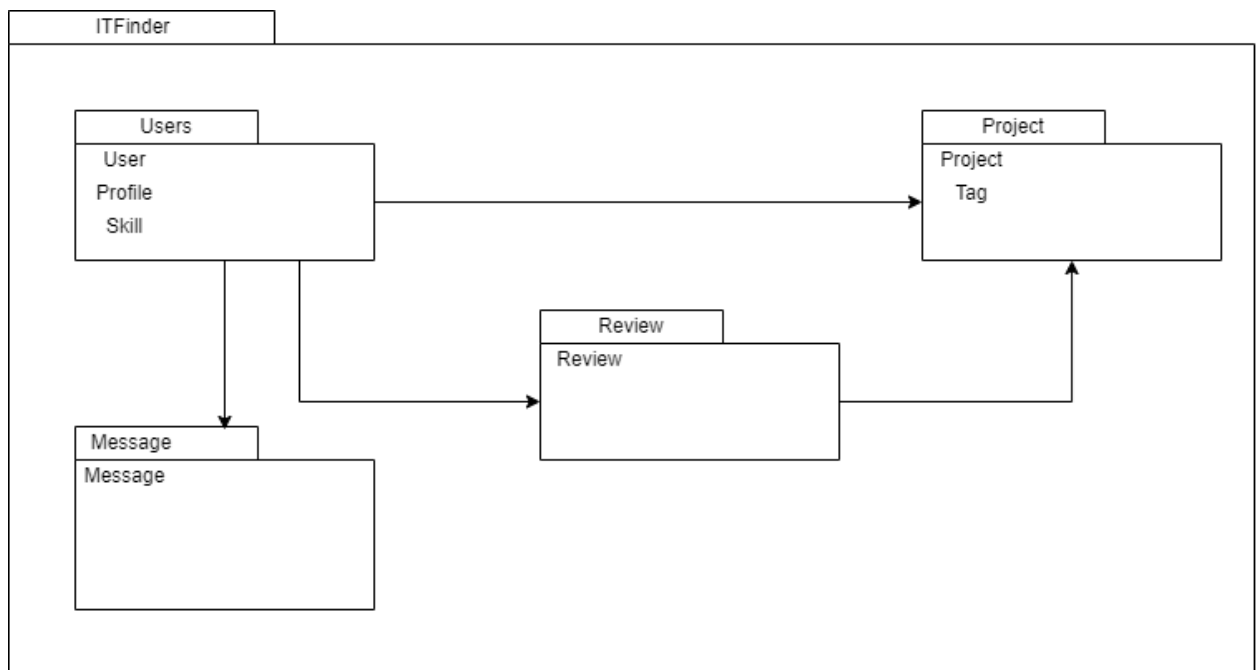


Рисунок 6 – Диаграмма пакетов

Проект «ITFiner» содержит в себе пакеты: «Users», «Message», «Review» и «Project».

Пакет «Users» состоит из концептуально близких классов «User», «Profile» и «Skill».

Пользователи могут добавлять в свой профиль проекты. Пакет «Project» содержит список проектов.

Пакет «Review» содержит в себе отзывы к проектам, а пакет «Message» – сообщения, которые отправляют и получают пользователи системы.

Диаграмма последовательностей

Диаграмма последовательностей UML – это графическое представление взаимодействия между объектами или компонентами в системе на основе временной последовательности. Этот тип диаграммы UML, который показывает, как объекты взаимодействуют друг с другом через сообщения, отображая порядок выполнения действий во времени [1].

Диаграмма последовательностей может помочь разработчикам программного обеспечения понять, как компоненты взаимодействуют друг с другом в системе, и помочь в определении возможных узких мест в проектировании. Для создания диаграммы последовательности используются следующие элементы: объекты, линии жизни, сообщения, фрагменты и хронометраж.

Были спроектированы диаграммы последовательностей для нескольких вариантов использования. На рисунке 7 представлена диаграмма последовательности для операции «Авторизация».

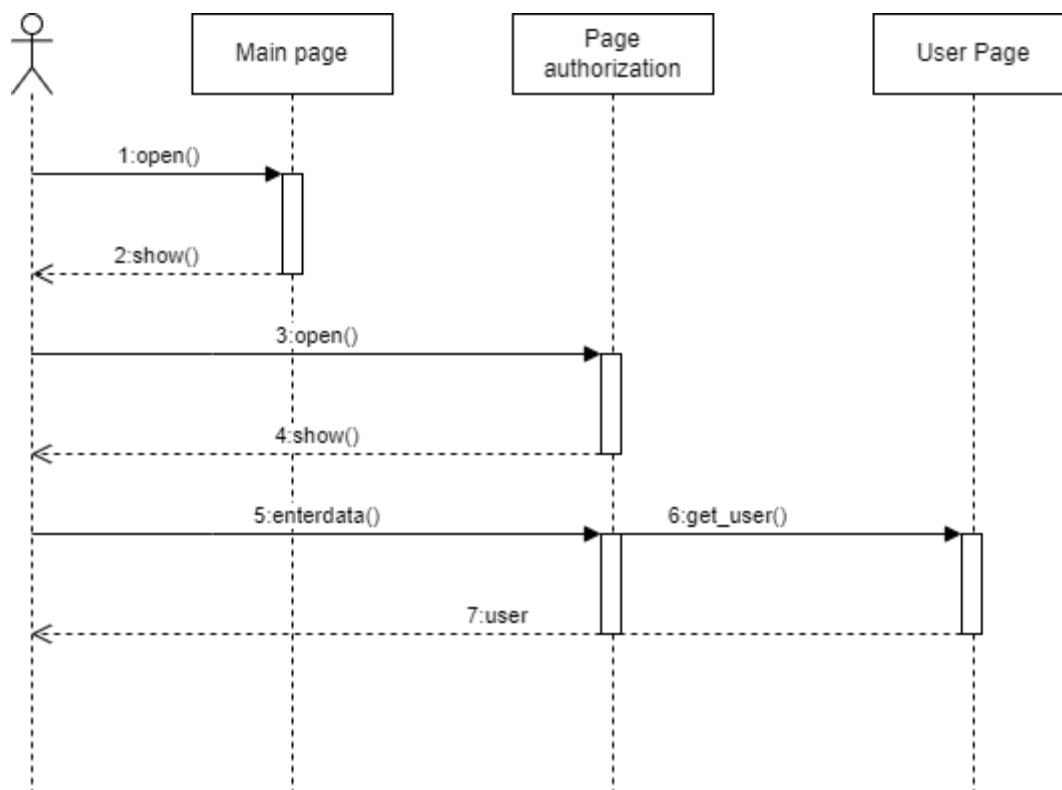


Рисунок 7 - Диаграмма последовательности для операции «Авторизация»

Для авторизации пользователю необходимо открыть главную страницу веб-приложения, на которой нужно нажать на кнопку авторизации. После вывода формы авторизации, необходимо заполнить необходимые поля. Если данные были введены корректно, вы войдем в систему. После авторизации пользователь попадает на личную страницу.

На рисунке 8 представлена диаграмма последовательности для операции «Добавление проекта».

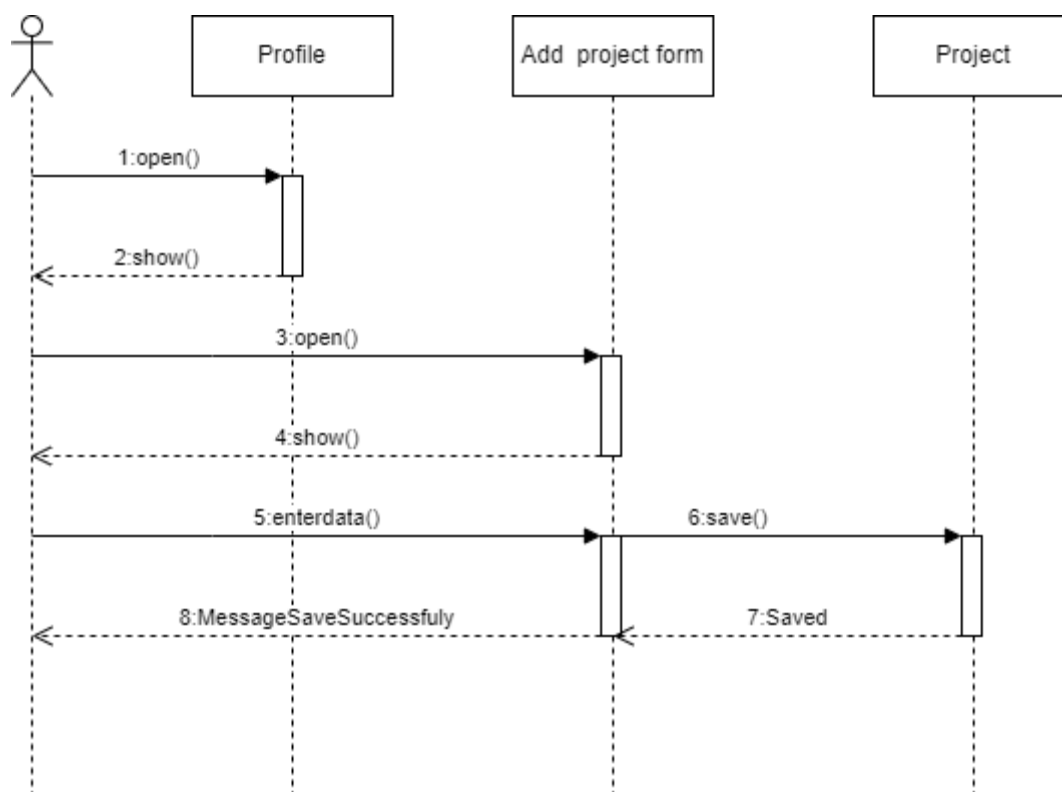


Рисунок 8 – Диаграмма последовательности для операции «Добавление проекта»

Для добавления проекта авторизованный пользователь переходит в свой профиль и жмет кнопку «Добавить проект». Затем он заполняет форму и сохраняет введенные данные. После чего выводится сообщение об успешном добавлении нового автомобиля.

На рисунке 9 представлена диаграмма последовательности для операции «Поиск разработчика».

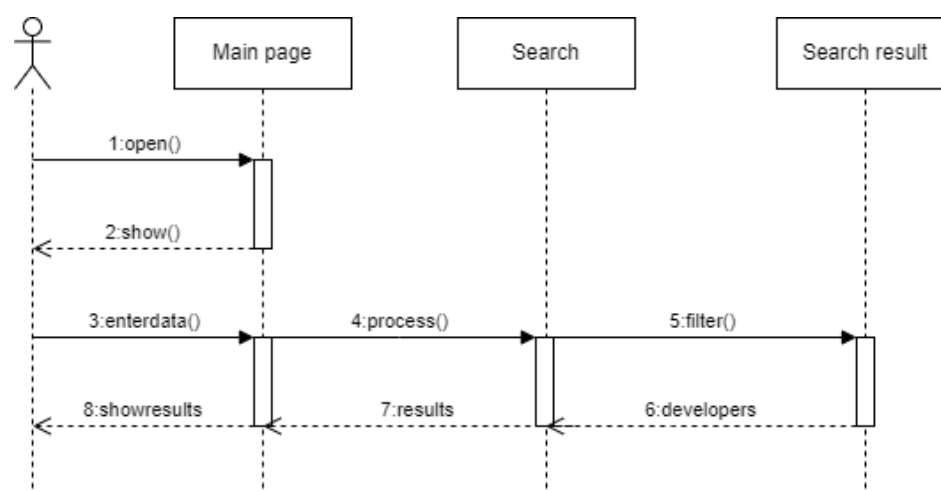


Рисунок 9 – Диаграмма последовательности для операции «Поиск разработчика»

Для операции поиска пользователю необходимо открыть главную страницу и в строке поиска ввести параметры, по которым он будет осуществляться. После нажатия на соответствующую кнопку, нам выведется список в соответствии с заданными параметрами.

На рисунке 10 представлена диаграмма последовательности для операции «Добавление отзыва».

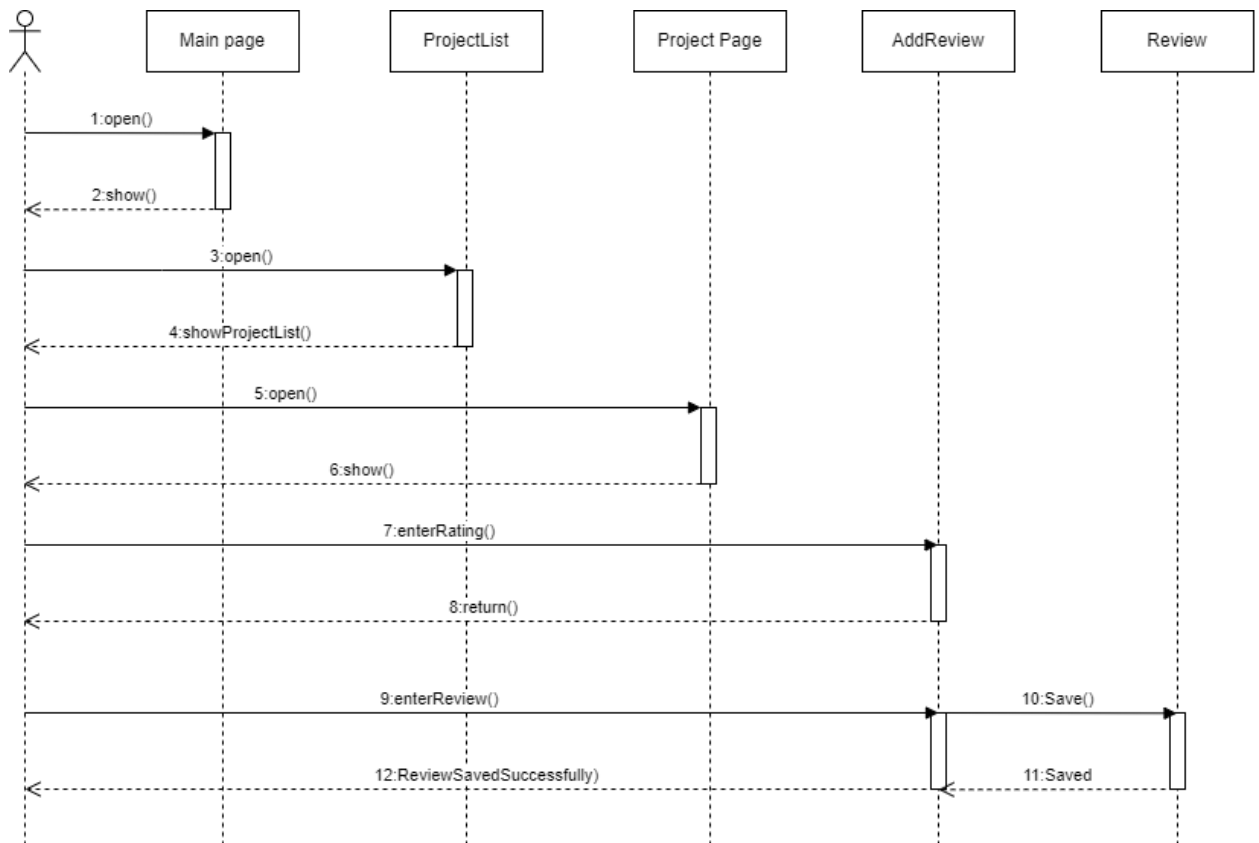


Рисунок 10 – Диаграмма последовательности для операции «Добавление
ОТЗЫВА»

Для того, чтобы оставить отзыв авторизованному пользователю необходимо открыть страницу со списком проектов и перейти на страницу любого из них. Затем ему нужно поставить оценку (положительную или отрицательную), написать текст отзыва и нажать кнопку сохранения.

РЕАЛИЗАЦИЯ

Разработка приложения

На рисунках ниже представлены некоторые части интерфейса. Ссылка на GitHub https://github.com/yamichnikita/Software_development_technologies.

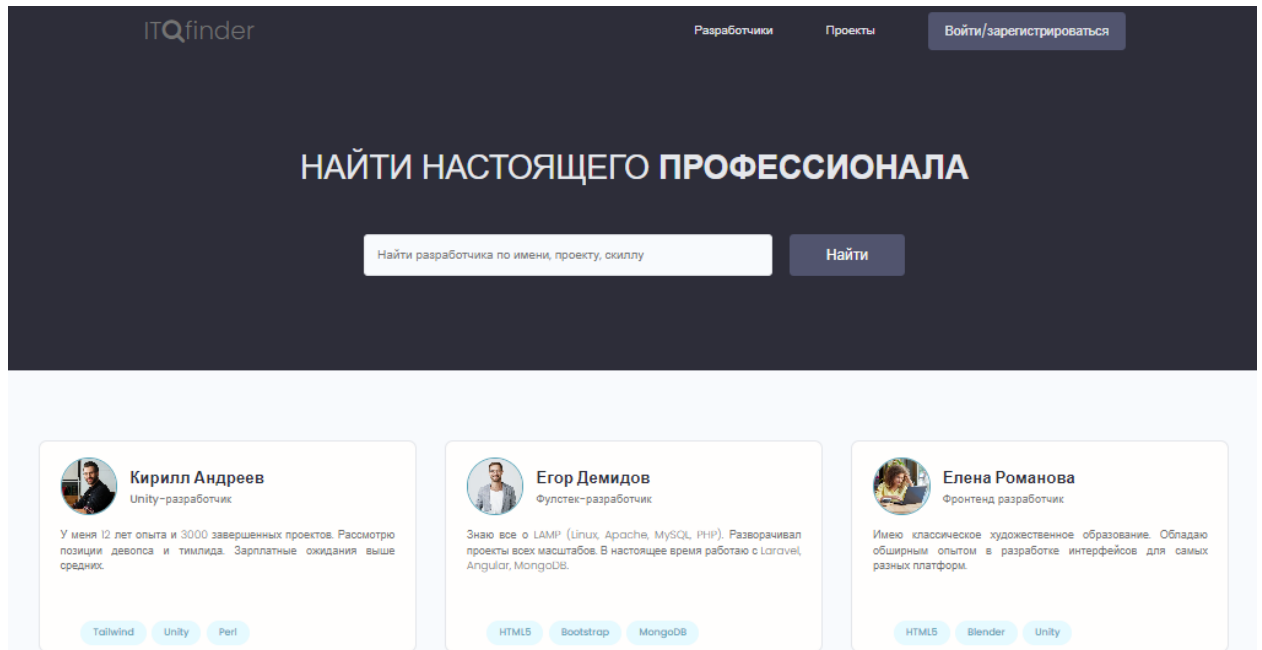


Рисунок 11 – Главная страница сайта

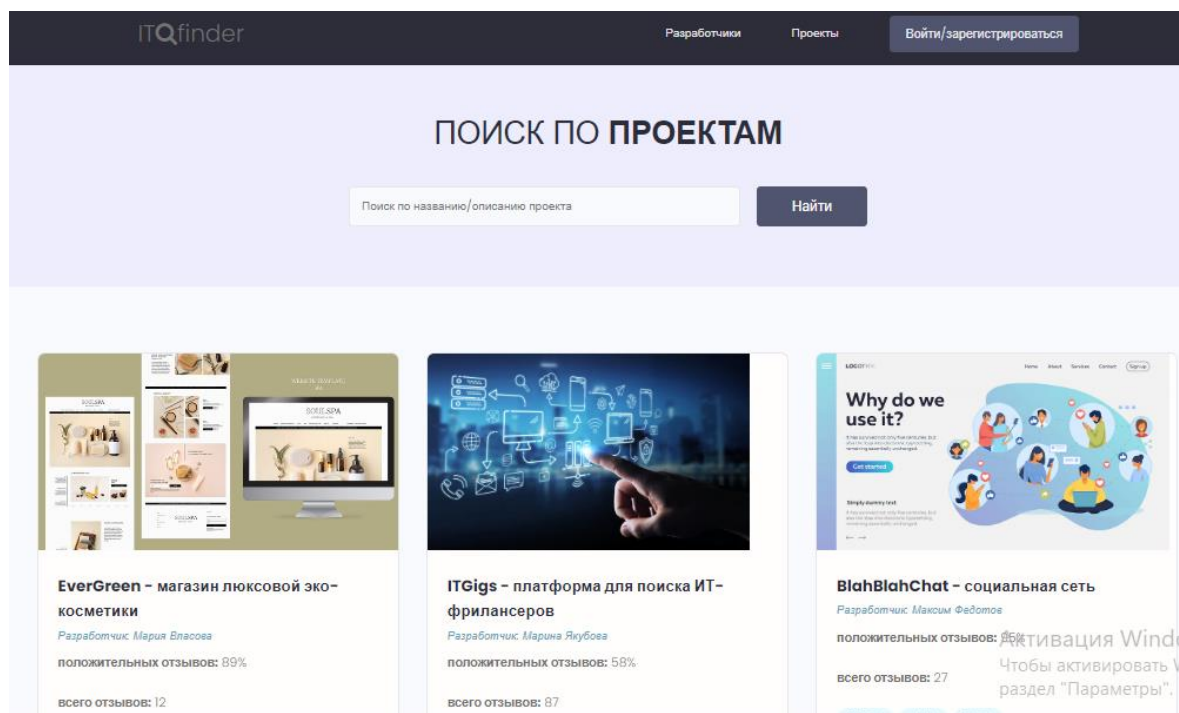


Рисунок 12 – Страница с проектами

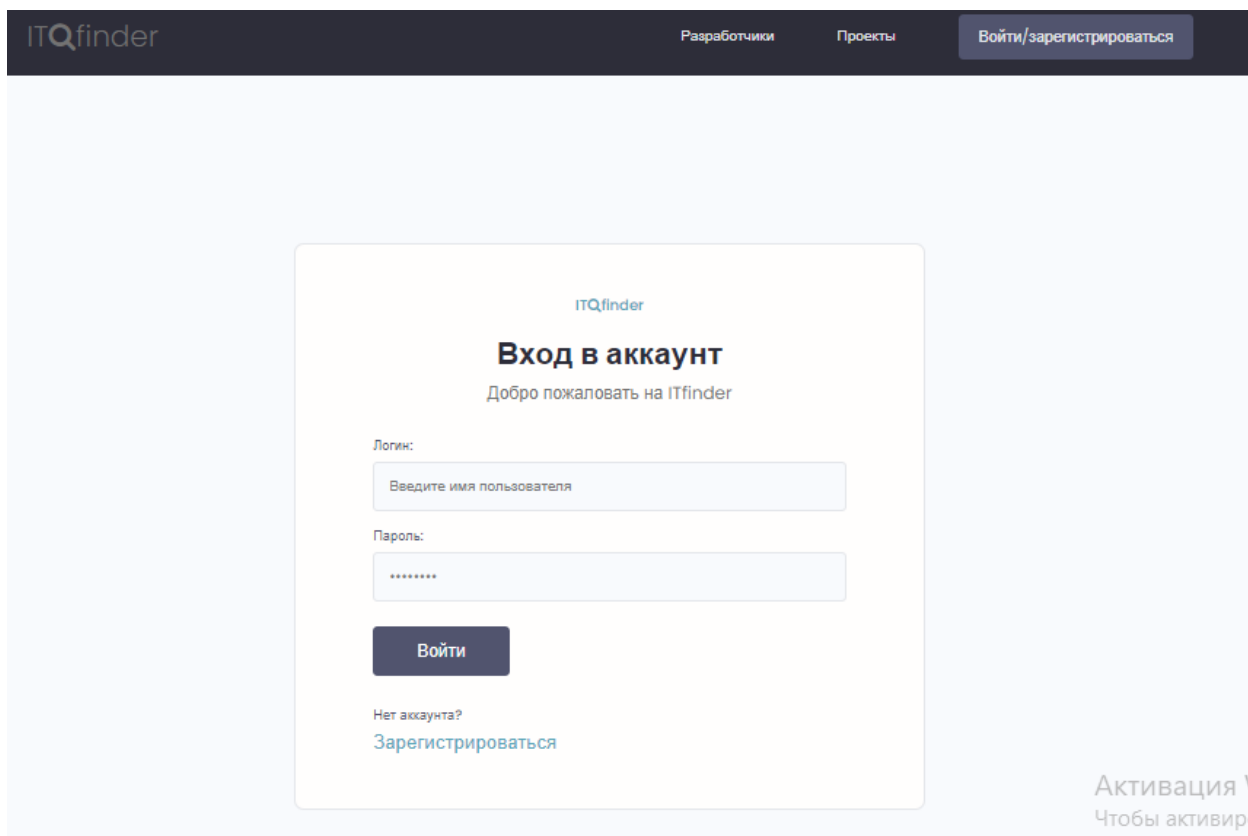


Рисунок 13 – Страница авторизации

Тестирование

Основная цель тестирования программного обеспечения заключается в обнаружении ошибок и дефектов в программе, чтобы они могли быть исправлены и готовое ПО было более надежным и качественным. В общем смысле, цель тестирования заключается в проверке того, что ПО соответствует заданным требованиям.

Кроме того, тестирование программного обеспечения имеет следующие цели:

1. Улучшение качества ПО.
2. Сокращение рисков: тестирование позволяет предотвращать потенциальные проблемы, которые могут привести к сбоям или нежелательным результатам.

3. Экономия ресурсов: тестирование может предотвратить затраты на ошибки, которые будут стоить дорого, если они произойдут после выпуска продукта.

Тестирование является важной частью цикла разработки ПО и помогает осуществлять контроль качества.

Основные принципы организации тестирования:

1. Необходимой частью каждого теста должно являться описание ожидаемых результатов работы программы;

2. Программе не должна тестироваться её автором;

3. Организация – разработчик программного обеспечения не должна "единолично " его тестировать;

4. Необходимо подбирать тесты не только для правильных (предусмотренных) входных данных, но и для неправильных (непредусмотренных);

5. При анализе результатов каждого теста необходимо проверять, не делает ли программа того, что она не должна делать;

6. "Принцип скопления ошибок" – вероятность наличия не обнаруженных ошибок в некоторой части программы прямо пропорциональна числу ошибок, уже обнаруженных в этой части.

Модульные тесты

Каждая сложная программная система состоит из отдельных частей – модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе системы в целом, необходимо вначале протестировать каждый модуль системы в отдельности. В случае возникновения проблем это позволит проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них. Такое тестирование модулей по отдельности получило название модульного тестирования (unit testing) [6].

Для каждого модуля, подвергаемого тестированию, разрабатывается тестовое окружение, включающее в себя драйвер и заглушки, готовятся тест-требования и тест-планы, описывающие конкретные тестовые примеры.

Основная цель модульного тестирования – удостовериться в соответствии требованиям каждого отдельного модуля системы перед тем, как будет произведена его интеграция в состав системы. Для создания unit-теста используется класс `TestCase`, от которого наследуются все созданные разработчиком классы с тестами [6].

Ниже приведен тест входа пользователя в систему.

```
class SigninTest(TestCase):
    def setUp(self):
        self.user = Profile.objects.create(
            user= 'test',
            username='test1',
            email='test@nail.ru',
            name='tester',
        )
        self.user.save()
    def tearDown(self):
        self.user.delete()
    def test_correct(self):
        user = authenticate(username='test', password='12test12')
        self.assertTrue((user is not None) and user.is_authenticated)
    def test_wrong_username(self):
        user = authenticate(username='wrong', password='12test12')
        self.assertFalse(user is not None and user.is_authenticated)
    def test_wrong_pssword(self):
        user = authenticate(username='test', password='wrong')
        self.assertFalse(user is not None and user.is_authenticated)
```

Интеграционные тесты

Интеграционное тестирование – это метод тестирования программного обеспечения, который проверяет работу системы, когда отдельные компоненты системы объединены. Цель интеграционного тестирования заключается в том, чтобы обнаружить ошибки взаимодействия между отдельными компонентами системы, сценарии использования и функциональность системы в целом.

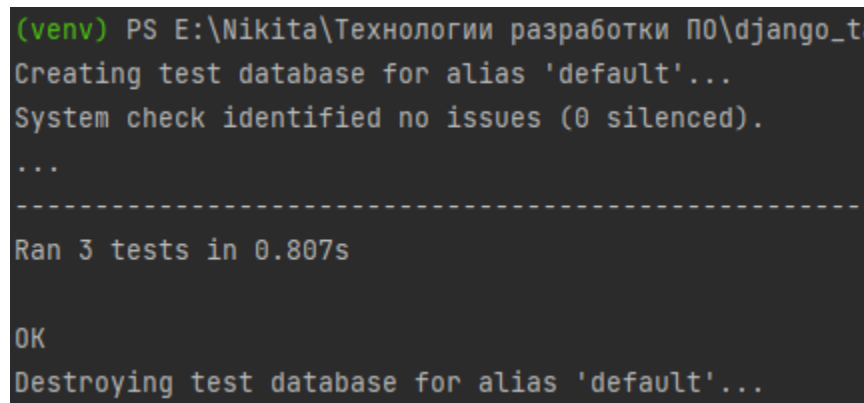
Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует

их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования [8].

Построение и выполнение тестов

Для выполнения всех тестов для необходимо использовать команду `manage.py test`. Чтобы обратиться к какому – то определенному приложению или проверки теста, необходимо использовать `python manage.py test <application_name>`.

Для проверки работоспособности приложения в Django в оперативной памяти создается база данных специально для тестов.



```
(venv) PS E:\Nikita\Технологии разработки ПО\django_t
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
-----
Ran 3 tests in 0.807s

OK
Destroying test database for alias 'default'...
```

Рисунок 14 – Результат тестирования

На рисунке 14 видим, что тестирование входа пользователя в систему прошло успешно.

Покрывтие кода

Тестирование покрытия кода – это метод тестирования программного обеспечения, который измеряет, сколько строк кода было выполнено или протестировано в ходе запуска тестов. Цель такой проверки – обнаружение недостаточно протестированных участков кода.

Проверка осуществляется с помощью специальных инструментов, которые анализируют код и определяют, какие участки были выполнены во время тестов, а какие участки кода остались непроверенными.

Различают несколько способов измерения покрытия кода. В Википедии представлены следующие определения [9]:

- Покрытие операторов – каждая ли строка исходного кода была выполнена и протестирована?
- Покрытие условий – каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована?
- Покрытие путей – все ли возможные пути через заданную часть кода были выполнены и протестированы?
- Покрытие функций – каждая ли функция программы была выполнена?
- Покрытие вход/выход – все ли вызовы функций и возвраты из них были выполнены?

Jenkins CI позволяет оценить степень покрытия кода тестами.

Запуск приложения для тестирования

Запуск и отладка проекта производится на локальном тестовом сервере (IP 127.0.0.1:8000), который предназначен для разработки и выявления правильности работы приложения, отладки кода программы, тестирования пользовательского интерфейса.

ДОКУМЕНТАЦИЯ

Назначение программы

Сайт «ITFinder» предназначен поиска и найма специалистов в сфере IT. Система поиска позволяет подобрать нужного специалиста по резюме, портфолио или описанию проектов; рейтинг проекта поможет оценить квалификацию, а мессенджер – послать оффер.

Разработанный сайт предоставляет следующий функционал:

1. Отображение полного списка кандидатов, добавленных в данную систему, с отображением следующих параметров: изображение самого кандидата, информация о его/её опыте и навыках, портфолио из выполненных проектов;
2. Отображение полного списка проектов, с отображением следующих параметров: логотип проекта, информация о его разработчике, описание проекта, отзывы, количество положительных отзывов, общее количество отзывов, стек используемых технологий (теги);
3. Регистрация новых пользователей;
4. Аутентификация пользователя;
5. Авторизации пользователя;
6. Поиск по профилям, навыкам, тегам, описаниям проектов;
7. Операции CRUD профиль, проект, тег проекта и навык разработчика.
8. Отправка сообщений другим пользователям;
9. Оценка проектов и возможность написания отзывов к ним.

Разработанный программный продукт предназначен для частных лиц и организаций желающих найти и привлечь кандидатов на работу.

Условия запуска программы

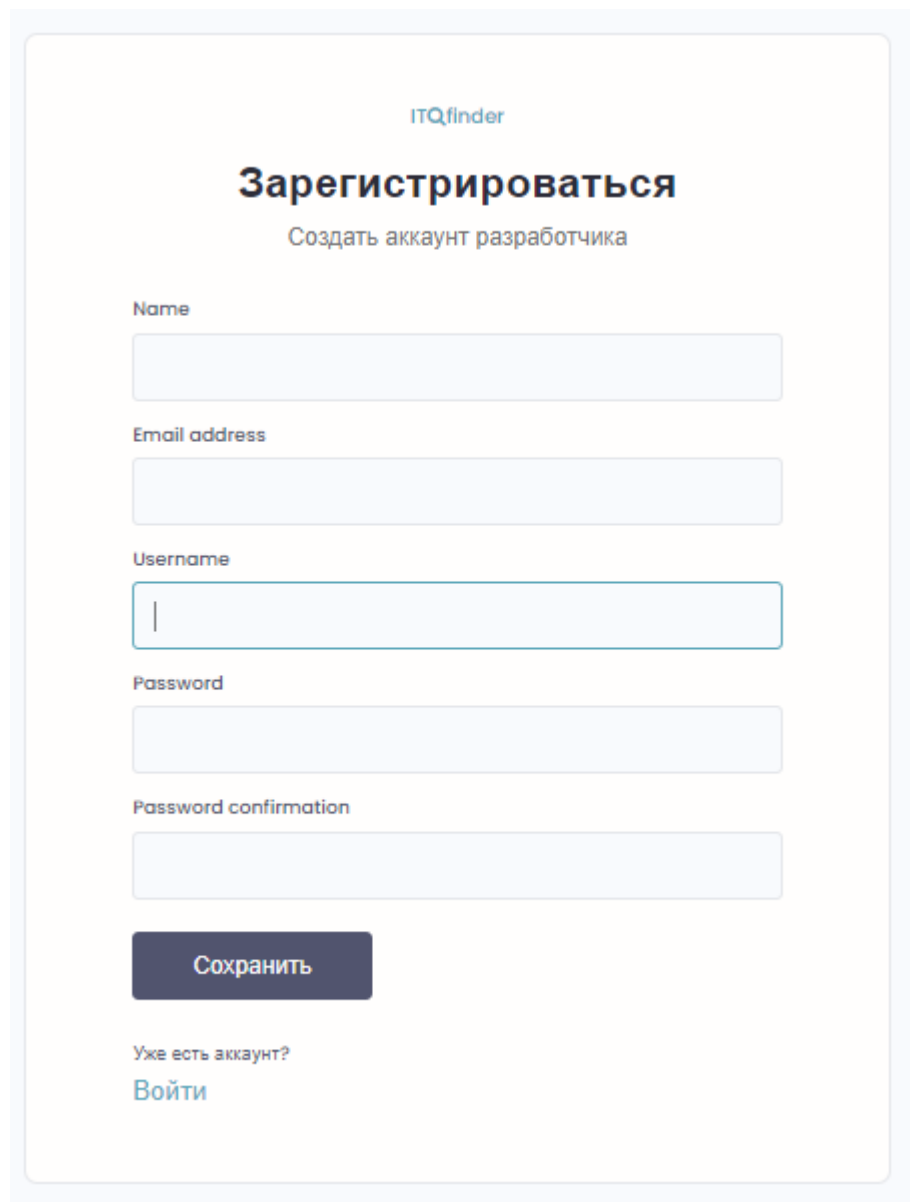
Для пользования сайтом пользователю необходимо иметь персональный компьютер с возможностью выхода в интернет со скоростью не менее 256 КБ/с. Также необходим установленный браузер и устройство ввода.

Выполнение программы

Сервис выполняет следующие функции:

- **Регистрация пользователя**

Для полноценной работы с сервисом необходимо зарегистрироваться. Для этого достаточно нажать на кнопку регистрации в шапке главной страницы и заполнить необходимые поля на появившейся странице регистрации.



The image shows a registration form for a service called ITQfinder. The form is titled "Зарегистрироваться" (Register) with the subtitle "Создать аккаунт разработчика" (Create developer account). It contains five input fields: "Name", "Email address", "Username", "Password", and "Password confirmation". Below the fields is a dark blue button labeled "Сохранить" (Save). At the bottom, there is a link "Войти" (Login) preceded by the text "Уже есть аккаунт?" (Already have an account?).

ITQfinder

Зарегистрироваться

Создать аккаунт разработчика

Name

Email address

Username

Password

Password confirmation

Сохранить

Уже есть аккаунт?

[Войти](#)

Рисунок 15 – Форма регистрации пользователей

- **Авторизация пользователя**

Для авторизации достаточно нажать на кнопку входа в шапке главной страницы и заполнить необходимые поля на появившейся странице авторизации.

Рисунок 16 – Форма авторизации пользователя

Если введённые данные корректны, то пользователь попадает на личную страницу.

- **Отображение списка кандидатов**

На главной странице сервиса отображается список кандидатов. Пользователь (как авторизованный, так и не авторизованный) может перейти на страницу любого из них.

На личной странице разработчика доступна следующая информация: изображение самого кандидата, информация о его/её опыте и навыках, портфолио из выполненных проектов.

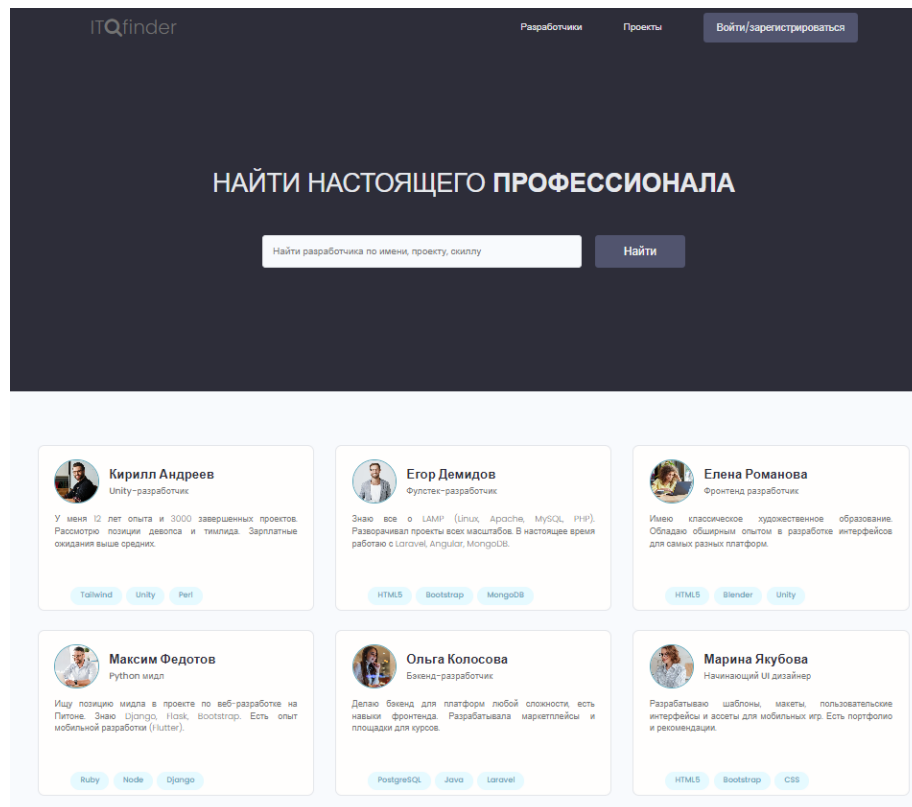


Рисунок 17 – Главная страница со списком разработчиков

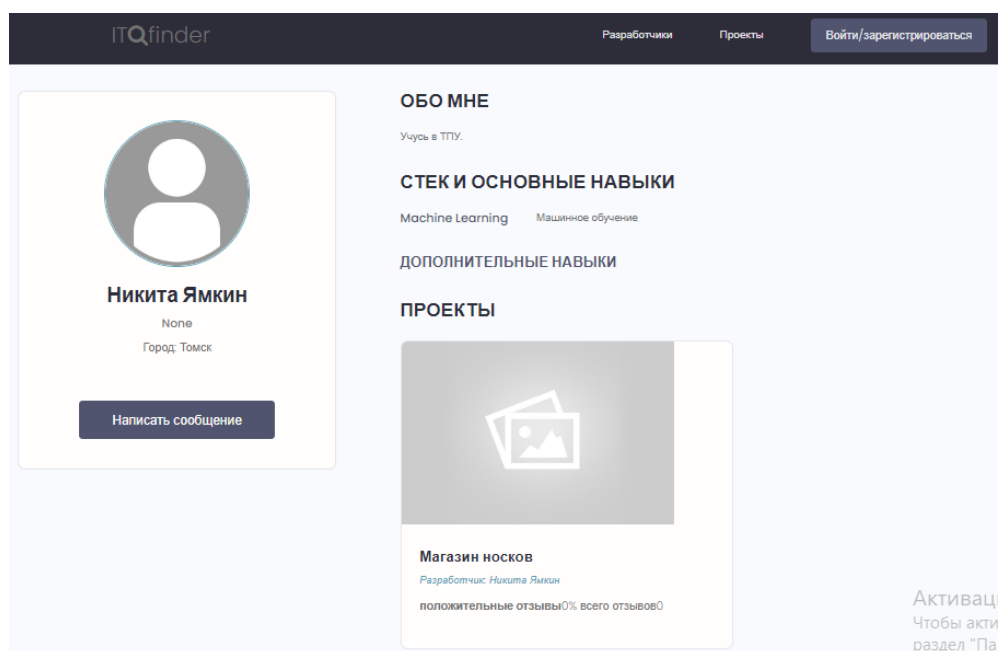


Рисунок 18 – Личная страница разработчика

- **Отображение списка проектов**

На сайте имеется страница со списком всех проектов. Пользователь (как авторизованный, так и не авторизованный) также может посмотреть подробную информацию о любом проекте. А именно: логотип проекта, информация о его разработчике, описание проекта, отзывы, количество положительных отзывов, общее количество отзывов, стек используемых технологий (теги).

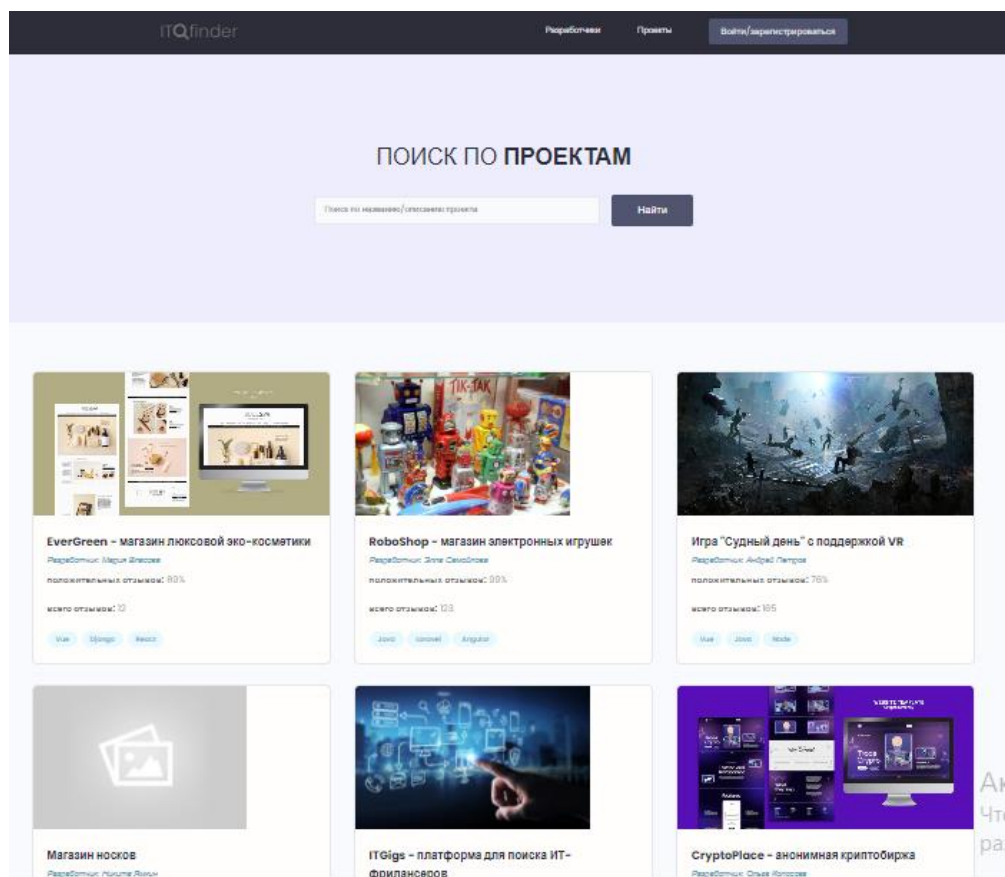


Рисунок 19 – Страница со списком проектов

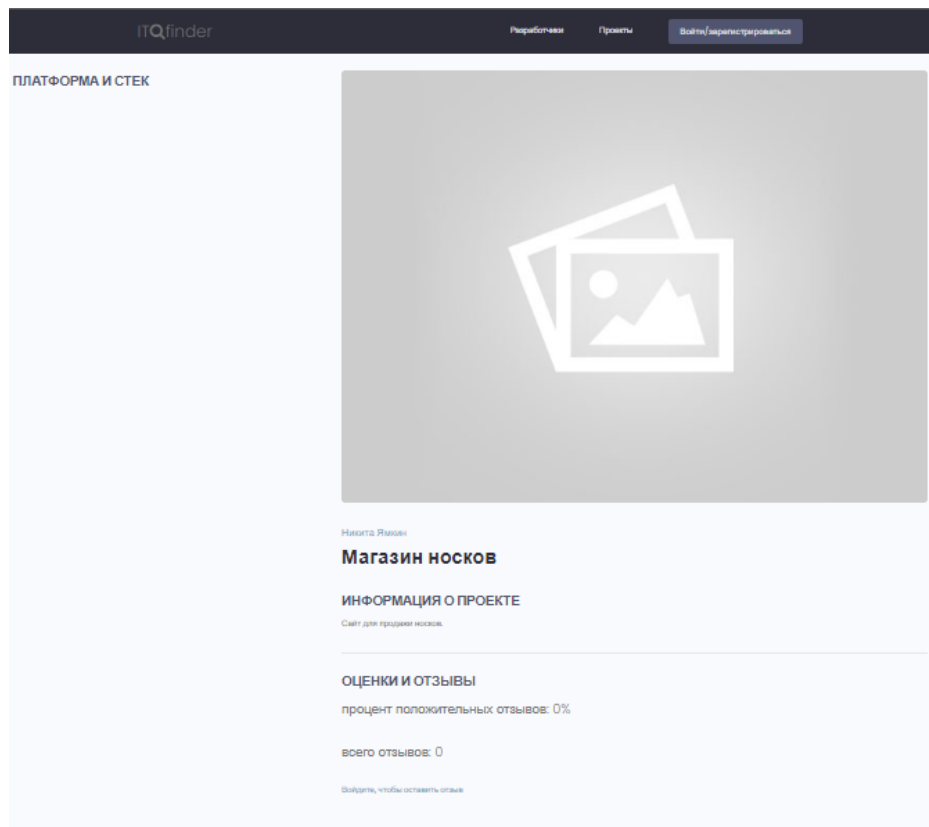


Рисунок 20 – Страница с информацией о конкретном проекте

• Поиск

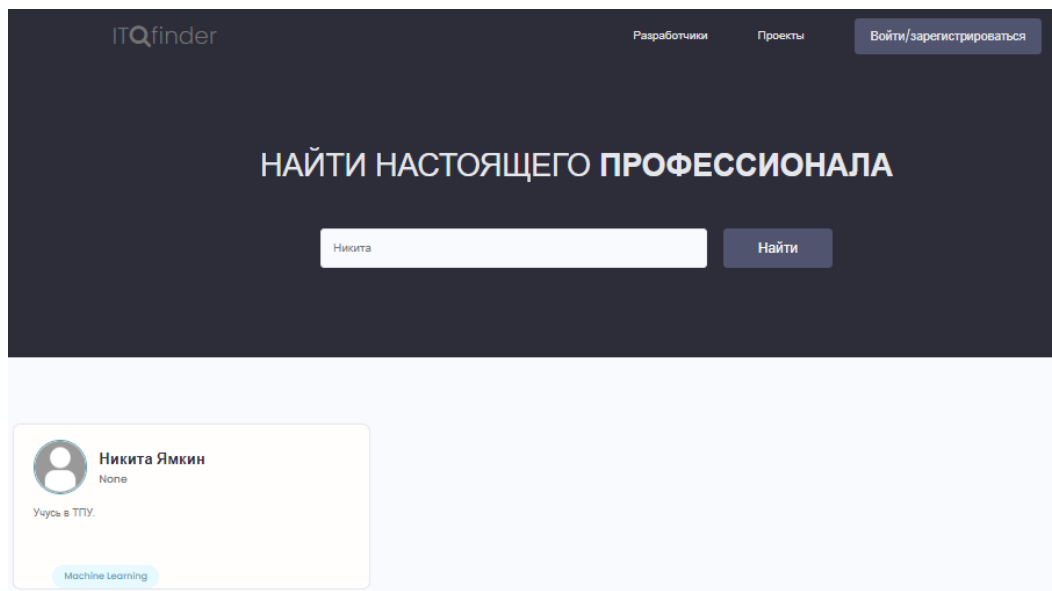


Рисунок 21 – Результат поиска по имени

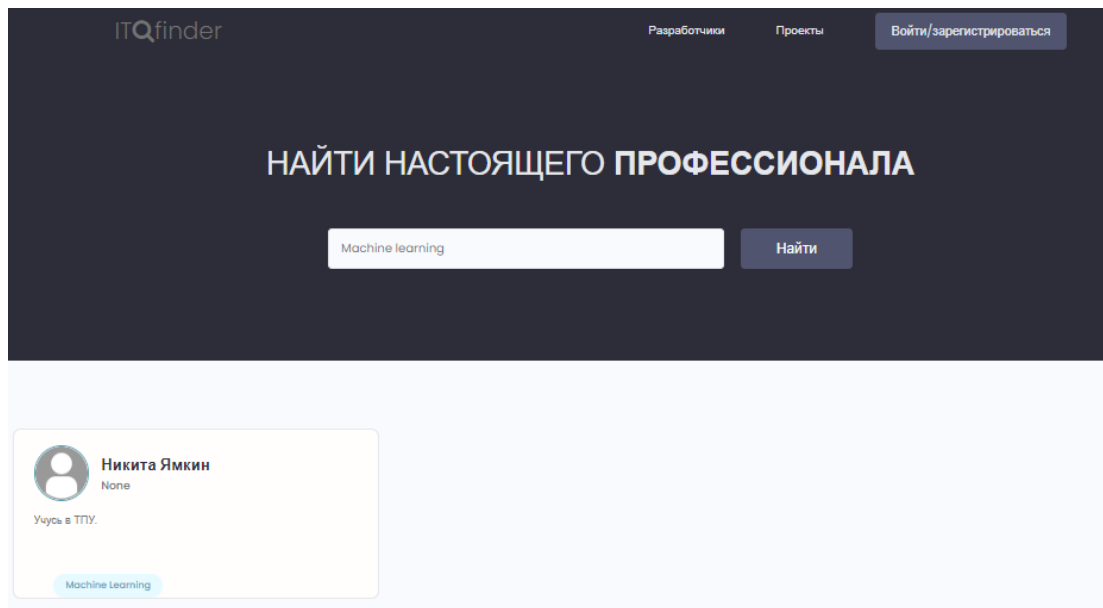


Рисунок 22 – Результат поиска по навыку

- **CRUD проект, тег проекта и навык разработчика**

Авторизованный пользователь может добавлять в свое портфолио новые проекты, создавать навыки и теги.

Имя проекта

Слэг

Сервис проекта

Выберите файл

Файл не выбран

Тип

☐ React

☐ Django

☐ Vue

☐ Angular

☐ Laravel

☐ Node

☐ Java

☐ Tailwind

☐ PHP

☐ MySQL

☐ MariaDB

☐ Flask

☐ Wordpress

☐ Flutter

☐ Golang

☐ Kotlin

☐ Go

☐ Unity

☐ Zbrush

☐ IC

Описание проекта

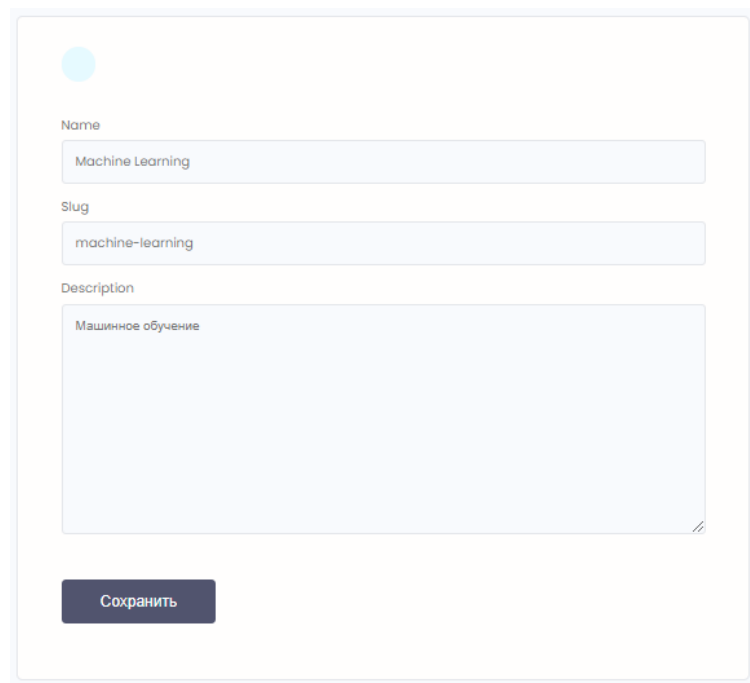
Исходный код на GitHub

Тип

Добавьте собственный тип, если его нет в списке

Сохранить

Рисунок 23 – Страница создания проекта



Machine Learning

machine-learning

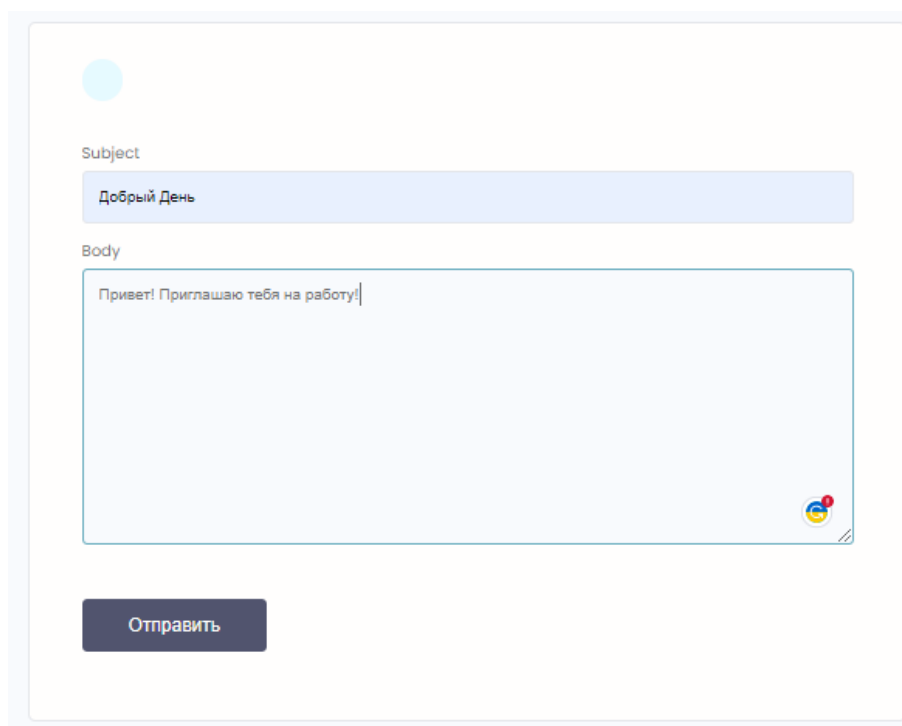
Машинное обучение

Сохранить

Рисунок 24 – Страница изменения навыка

- **Отправка сообщений другим пользователям**

Авторизованный пользователь может отправить сообщение другому пользователю, например для того, чтобы пригласить на работу.



Добрый День

Привет! Приглашаю тебя на работу!

Отправить

Рисунок 25 – Окно для отправки сообщения

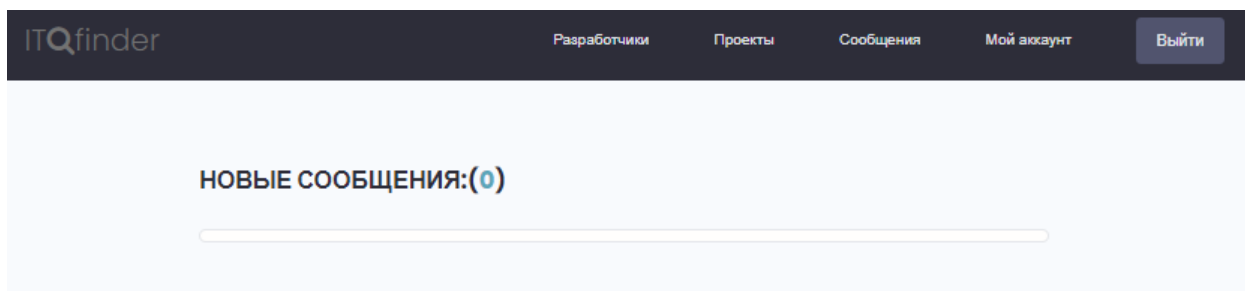


Рисунок 26 – Страница с входящими сообщениями

- **Оценка проектов и возможность написания отзывов к ним**

Разработчик также может оставить отзыв к проекту другого разработчика.

Рисунок 27 – Отправка отзыва

ЗАКЛЮЧЕНИЕ

Результатом данной курсовой работы является созданный сайт для рекрутинга разработчиков «ITFinder», разработанный с использованием языка программирования Python и платформы для создания веб-приложений Django.

Создание данного приложения включало создание концептуальных, аналитических и физических моделей данных, что облегчило разработку и помогло соблюсти требования. Поэтому данный подход был оправдан.

В будущем планируется модификация приложения и работа над существующими багами. При доработке и модификации программного продукта, возможно, его внедрения для практического применения, так как рассматриваемая область имеет большой спрос в современном мире. Кроме того, преимущество использования Django, которое заключается в возможности создания API для работы с сервисом с помощью любого устройства без изменений в исходном коде программы.

СПИСОК ИСТОЧНИКОВ

1. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя: Пер.с англ.-М: ДМК, 2000. – 432 с.,ил
2. UML Диаграмма Классов (UML Class Diagram) // youtube.com [Электронный ресурс] – URL: <https://www.youtube.com/watch?v=sVVJp5a41o4&t=316s>, свободный (дата обращения: 18.05.2023).
3. UML Диаграмма Пакетов (UML Package Diagram) // youtube.com [Электронный ресурс] – URL: <https://www.youtube.com/watch?v=237BWanM4Ak>, свободный (дата обращения: 18.05.2023).
4. UML: что такое диаграмма пакетов? Как это использовать? // Кибермедиана [Электронный ресурс] – URL: <https://www.cybermedian.com/ru/uml-what-is-package-diagram-how-to-use-it/>, свободный (дата обращения: 18.05.2023).
5. Моделирование данных: зачем нужно и как реализовывать // Хабр [Электронный ресурс] – URL: <https://habr.com/ru/articles/554388/>, свободный (дата обращения: 18.05.2023).
6. Как покрыть приложение на Django модульными тестами // mkdev [Электронный ресурс] – URL: <https://mkdev.me/ru/posts/kak-pokryt-prilozhenie-na-django-modulnymi-testami>, свободный (дата обращения: 18.05.2023).
7. Модульное тестирование: что это? Типы, инструменты // Logrocon [Электронный ресурс] – URL: https://logrocon.ru/news/unit_testing, свободный (дата обращения: 18.05.2023).
8. Интеграционное тестирование // Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Интеграционное_тестирование, свободный (дата обращения: 18.05.2023).
9. Что такое покрытие кода? // Atlassian [Электронный ресурс] – URL: <https://www.atlassian.com/ru/continuous-delivery/software-testing/code-coverage>, свободный (дата обращения: 18.05.2023).