Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет Математики и информатики
Кафедра Информационных и управляющих систем
Направление подготовки 09.03.02 – Информационные системы и технологии
Направленность (профиль) образовательной программы Информационные системы и технологии

ДОПУСТИТЬ К ЗАЩИТЕ

Зав. кафедрой

_____ А.В. Бушманов

«_____»_____2023 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка веб-приложения для аренды и продажи недвижимости

Исполнитель
студент группы 955об _____ (подпись, дата) С.И. Костенко

Руководитель
доцент, канд. физ.-мат. наук _____ (подпись, дата) В.В. Еремина

Консультант:
по безопасности и экологичности
доцент, канд. техн. наук _____ (подпись, дата) А.Б. Булгаков

Нормоконтроль
инженер кафедры _____ (подпись, дата) В.Н. Адаменко

Благовещенск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

(ФГБОУ ВО «АмГУ»)

Факультет Математики и информатики

Кафедра Информационных и управляющих систем

УТВЕРЖДАЮ

Зав. кафедрой

_____ А.В. Бушманов

«_____»_____2023 г.

ЗАДАНИЕ

К выпускной квалификационной работе студента Костенко С.И.

1. Тема выпускной квалификационной работы: Разработка веб-приложения для аренды и продажи недвижимости

(утверждено приказом от 24.04.2023 №974-уч)

2. Срок сдачи студентом законченной работы (проекта) 20.06.23

3. Исходные данные к выпускной квалификационной работе: нормативные документы, учебная литература, Интернет-ресурсы.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов):

5. Перечень материалов приложения: (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.):

6. Консультанты по выпускной квалификационной работе (с указанием относящихся к ним разделов): Булгаков А.Б., раздел Безопасность и экологичность

7. Дата выдачи задания 30.01.2023

Руководитель выпускной квалификационной работы: В.В. Еремина, доцент, канд. физ.-мат. наук

(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Задание принял к исполнению (30.01.2023)_____

(подпись студента)

РЕФЕРАТ

Дипломная (бакалаврская) работа содержит 72 страницы, 29 рисунков, 21 таблицу, 15 источников и 1 приложение.

WEB-ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, НЕДВИЖИМОСТЬ, ЭФФЕКТИВНОСТЬ, ЯЗЫКИ ПРОГРАММИРОВАНИЯ

В работе выполнена разработка веб приложения для аренды и продажи недвижимости.

Цель работы – проектирование и разработка веб-приложения для аренды и продажи недвижимости.

Объект исследования – ООО «ФЛЭТС»

Предмет исследования – процесс продажи и аренды недвижимости.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных

СУБД – система управления базами данных

SQL – (Structured Query Language) структурированный язык запросов

ПК – персональный компьютер

ПО – программное обеспечение

ПП – программный продукт

ВВЕДЕНИЕ

В современном обществе недвижимость является одной из самых значимых и востребованных сфер деятельности. Купля, продажа и аренда недвижимости требуют от участников рынка эффективных инструментов для поиска, выбора и оформления сделок. В этой связи разработка веб-приложения для продажи и аренды недвижимости является актуальной и перспективной задачей, которая позволит упростить и автоматизировать процесс взаимодействия между продавцами, арендодателями и потенциальными клиентами.

Эффективность интернет-рекламы растет с каждым годом. Буквально несколько лет назад у многих владельцев агентств не было четкого понимания - для чего им нужно веб-приложение, как оно должен работать и какие функции выполнять. Сейчас же картина изменилась, многие владельцы агентств понимают, что собственный сайт станет хорошей площадкой для привлечения клиентов. Конечно при том условии, что он будет грамотно построен будет привлекать целевых посетителей. Также сайт свидетельствует о процветании компании, о ее солидности и современности.

С развитием интернет-технологий и распространением доступа к сети Интернет, все больше людей предпочитают искать и предлагать недвижимость онлайн. Однако, существующие решения на рынке не всегда полностью удовлетворяют потребности пользователей и могут быть ограничены в функциональности. В связи с этим разработка собственного веб-приложения, адаптированного под конкретные потребности пользователей, представляет интерес и актуальность для рынка недвижимости.

Целью данной работы является разработка веб-приложения, предоставляющего удобный и эффективный инструмент для продажи и аренды недвижимости. Для достижения данной цели поставлены следующие задачи:

– изучение предметной области и анализ существующих решений в сфере продажи и аренды недвижимости;

– формулирование функциональных требований к разрабатываемому веб-приложению;

– разработка макета и проектирование веб-приложения, включая архитектуру и структуру базы данных;

– реализация и тестирование веб-приложения с учетом поставленных требований и ожиданий пользователей.

Таким образом, данная работа представляет собой комплексный подход к разработке веб-приложения для продажи и аренды недвижимости, объединяющий в себе теоретические и практические аспекты данной проблемы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Агентство недвижимости предоставляет собой профессиональное сопровождение всех операций, которые возможны на рынке недвижимости. В первую очередь это продажа и покупка жилой и коммерческой недвижимости, и недвижимости в новостройках, а также аренда квартир, комнат, земельных участков. Преимуществом агентства является объёмная база вариантов недвижимости и земельных участков, выставленных на продажу или предлагаемых для сдачи в аренду. Разрабатываемый сайт позволит разместить информацию об агентстве недвижимости и услугах, которые оно предлагает, различные объявления о продаже или аренде недвижимости, а также позволит пользователю оставлять заявки и предоставит возможность связаться с агентством.

При создании сайта, посвященного купле-продаже квартир и офисов нужно позаботиться об удобстве навигации и поиска необходимой информации.

Потенциальный клиент, который заходит на сайт агентства недвижимости, должен иметь возможность в считанные минуты найти то, что ему нужно, иначе он может воспользоваться услугами конкурентов. Поиск объектов на сайте агентства недвижимости должен производиться одновременно по нескольким параметрам. Во-первых, он должен осуществляться по стоимости недвижимости и её площади. Для домов и квартир также вводится функция поиска по количеству комнат. Еще один важный момент, на который необходимо обратить внимание, это местоположение объекта.

Потенциальному покупателю или арендатору необходимо иметь возможность выбора недвижимости в определенных районах и улицах.

Структура сайта агентства недвижимости состоит из двух главных разделов – продажи и аренды. Каждый из них для удобства поиска необходимо разбить на подразделы, посвященные жилой, офисной и торговой недвижимости.

После того, как база сайта агентства недвижимости создана, необходимо позаботиться о том, чтобы пользователи имели доступ к исчерпывающей информации по тому или иному объекту. Общая площадь, место расположения, описание инфраструктуры и интерьера существенно повысят шансы агентства продать или же сдать в аренду объект.

Немаловажное значение играет и наличие фотографий, которые дают представление о том, что же выставлено на сайте.

Любой человек, собирающийся приобретать или арендовать недвижимость, всегда интересуется правовыми аспектами сделки. В следующем параграфе будут рассмотрены сведения об организации предоставляющей данные услуги.

1.2 Общие сведения о предприятии

ООО ФЛЭТС (Фирма Лидеров Эксклюзивных Транзакций по Сделкам с недвижимостью) - это организация, специализирующаяся на продаже и аренде недвижимости.

ООО ФЛЭТС предлагает широкий спектр услуг, связанных с недвижимостью, для клиентов различных категорий, включая частных лиц, инвесторов, разработчиков и корпоративных клиентов. Основная цель компании - удовлетворение потребностей клиентов в поиске и приобретении или аренде качественной недвижимости.

Команда ФЛЭТС состоит из опытных и квалифицированных специалистов в области недвижимости, включая риэлторов, юристов и экспертов по оценке. Каждый сотрудник обладает профессиональными знаниями и пониманием рынка недвижимости, что позволяет организации предоставлять высококачественные услуги своим клиентам.

ООО ФЛЭТС обладает обширной базой данных недвижимости, включающей различные типы объектов, такие как квартиры, дома, коммерческая недвижимость и земельные участки. Благодаря использованию современных тех-

нологий и эффективным инструментам маркетинга, компания может эффективно продвигать и предлагать свои объекты клиентам.

ООО ФЛЭТС придерживается принципов профессионализма, этики и конфиденциальности при работе с клиентами. Компания стремится установить долгосрочные партнерские отношения с каждым клиентом, предлагая индивидуальный подход и учитывая их уникальные потребности и требования.

ФЛЭТС также активно участвует в общественных и социальных инициативах, направленных на развитие сферы недвижимости и поддержку социальных программ в области жилищного строительства.

В целом, ООО ФЛЭТС является уважаемой и успешной организацией, занимающейся продажей и арендой недвижимости, и стремится быть лидером в своей области, предоставляя высококачественные услуги и создавая долгосрочные отношения с клиентами.

1.3 Организационная структура предприятия

Руководство:

– генеральный директор: отвечает за общее руководство предприятием и принятие стратегических решений;

– заместитель генерального директора: поддерживает работу генерального директора и может замещать его в его отсутствие.

Отдел продаж:

– менеджеры по продажам: отвечают за поиск и привлечение клиентов, продажу недвижимости и заключение договоров;

– агенты по продажам: работают в тесном контакте с менеджерами по продажам и обеспечивают выполнение всех необходимых операций в процессе продажи недвижимости.

Отдел аренды:

– менеджеры по аренде: занимаются поиском арендаторов, заключением договоров аренды и контролем выполнения условий договора;

– агенты по аренде: выполняют задачи, связанные с поиском арендаторов и подготовкой объектов к аренде.

Отдел маркетинга и рекламы:

– менеджеры по маркетингу: занимаются исследованием рынка недвижимости, разработкой стратегий продвижения и рекламы услуг предприятия;

– специалисты по рекламе: отвечают за создание рекламных материалов и их распространение.

Отдел управления объектами:

– управляющие объектами: отвечают за обслуживание объектов недвижимости и контроль их технического состояния;

– технические специалисты: обеспечивают ремонт и обслуживание объектов недвижимости.

Бухгалтерия и финансовый отдел:

– бухгалтер: отвечает за ведение бухгалтерского учета и отчетности;

– финансовый аналитик: занимается анализом финансовой деятельности предприятия и разработкой стратегий для улучшения ее финансового состояния.

Такая структура позволит организовать работу предприятия по аренде и продаже недвижимости наиболее эффективно и обеспечит качественное обслуживание клиентов.

Структура также представлена на рисунке 1.1.

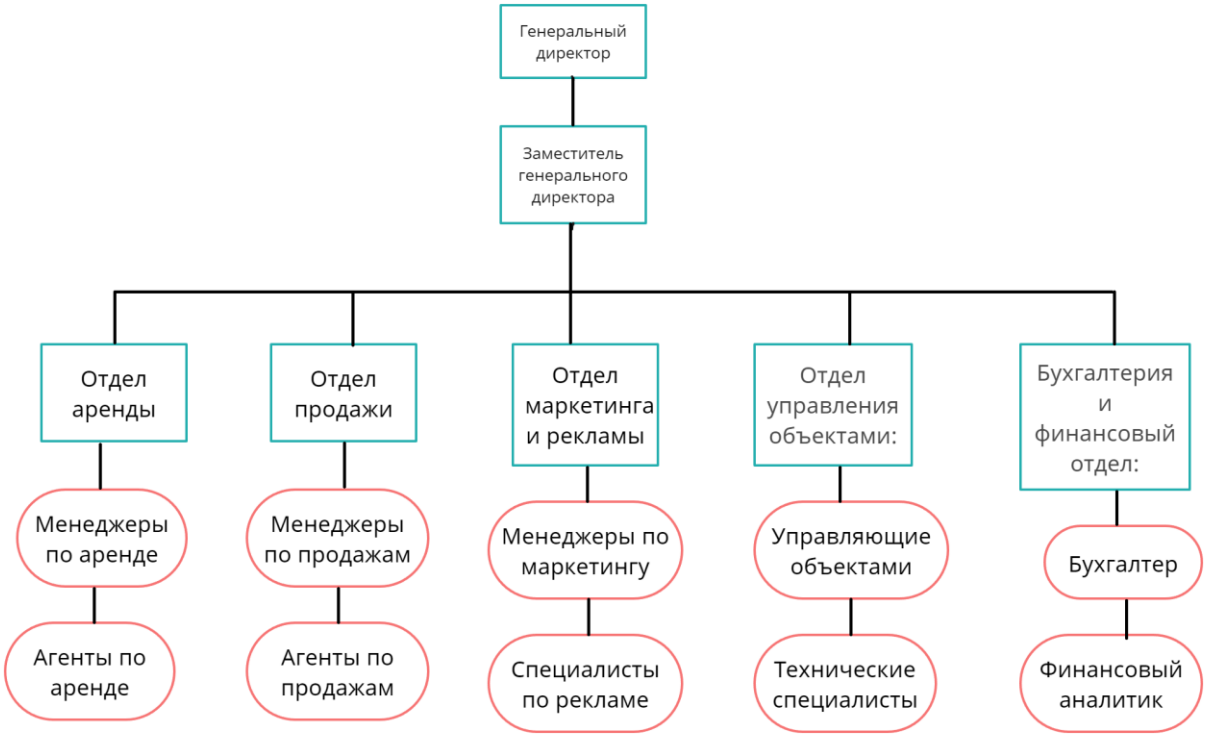


Рисунок 1.1 – Организационная структура ООО «ФЛЭТС»

1.4 Анализ и выбор основного языка программирования

Разработка качественного веб приложения — это очень трудоемкий процесс. Для реализации задуманного необходимо определить основной язык программирования, на котором будет реализовано приложение.

В настоящее время в мире существует множество языков программирования, каждый из которых обладает своими особенностями, преимуществами и недостатками. В рамках данного исследования был проведен анализ двух самых популярных языков для веб-программирования, Python и PHP.

Python является интерпретируемым высокоуровневым языком программирования, который отличается простым и понятным синтаксисом. Одним из главных преимуществ Python является его универсальность и широкий спектр применения. Он часто используется в научных исследованиях, веб-разработке, анализе данных и автоматизации задач. Python также известен своим обширным сообществом разработчиков, которое активно поддерживает и дополняет его функциональность. Однако, недостатком Python является его относительная

медлительность в сравнении с некоторыми компилируемыми языками, что может сказываться на производительности при выполнении некоторых задач.

PHP, в свою очередь, является языком программирования, специализирующимся на веб-разработке. Он широко применяется для создания динамических веб-сайтов и веб-приложений. Одним из основных преимуществ PHP является его интеграция с базами данных и простота разработки динамических страниц. Большое количество готовых решений и расширений также делают PHP популярным выбором для разработчиков веб-приложений. Однако, PHP имеет свои недостатки, включая сложности в поддержке и относительную низкую производительность по сравнению с некоторыми другими языками программирования.

Для удобного сравнения характеристик этих языков была составлена следующая сравнительная таблица:

Таблица 1.1 – Сравнительная таблица языков программирования

Характеристики	**Python**	**PHP**
Простой синтаксис	+	-
Универсальность	+	-
Широкий спектр применения	+	-
Интеграция с базами данных	-	+
Простота разработки	-	+
Большое количество готовых решений	-	+
Относительная медлительность	+	-
Подходит для разработки низкоуровневых систем	-	-

Продолжение таблицы 1.1

Ограниченные возможности для многопоточности	+	-
Сложности в поддержке и обновлении	-	+
Сложности с масштабированием и поддержкой больших проектов	-	+

Исходя из проведенного анализа и данных представленных в таблице 1.1 в качестве основного языка в данном проекте был выбран Python.

1.5 Анализ дополнительных средств и компонентов необходимых для разработки программного продукта

В реализации веб-приложения также будут задействованы следующие дополнительные средства и компоненты:

– HTML5;

– таблица стилей CSS;

– JavaScript;

– jQuery;

– библиотека Flask;

– Bootstrap5.

Ниже приведено краткое описание каждого из этих компонентов.

HTML – гипертекстовый язык разметки, который используется для вёрстки и создания веб страниц. HTML документ состоит из разметочных тегов и обычного текста, такие документы строят каркас страницы и автоматически распознаются, и обрабатываются современными браузерами. Разметочные теги служат ориентиром для браузера, чтобы браузер правильно отображал контент на странице [1].

CSS – это язык описания внешнего вида документа. Служит для стилизации контента на веб-страницах, сверстанных с помощью HTML. В CSS документе прописывается как именно отображает то или иное содержимое тега, добавить цвета фона, изменить размер шрифта или сам шрифт, все это делается с помощью таблицы стилей CSS [2].

JavaScript – язык программирования, который позволяет вам создать динамически обновляемый контент. Поскольку не всякие события и действия необходимо обрабатывать на стороне сервера, к примеру, такие как: подтверждение ввода в отдельном окне, анимация форм, нестандартное поведение кнопок и т.д., то в качестве «локального» обработчика этих действий будет применяться язык JavaScript. Этот язык обрабатывается также любым из современных браузеров, и для работы программ, написанных на нем, клиенту нет необходимости устанавливать отдельное программное обеспечение [3].

jQuery – набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Такой подход очень удобен, поскольку рутинные операции по взаимодействию с DOM можно осуществлять с помощью встроенных функций, дальнейшее проектирование заключается лишь в абстрагировании этого функционала [4].

Flask – библиотека python для разработки веб-приложений альтернативной которой является Django. Django поставляется с огромным количеством встроенного функционала в то время как Flask содержит лишь минимальный набор необходимый в основном для маршрутизации GET\POST запросов. В данном проекте выбор пал на Flask поскольку важно контролировать и понимать весь процесс разработки, а также, потому что продукт, который рассматривается в данной работе, нельзя «шаблонизировать» и разработка его функционала лежит на плечах программиста [5].

Bootstrap5 – набор инструментов для создания сайта, он включается в себя HTML и CSS шаблоны оформления типографики, форм и другого контента. Он облегчает разработку поскольку предоставляет уже готовые варианты этих форм с уже описанными стилями CSS. Сетка Bootstrap также делает разработанные с помощью него макеты адаптивными, т.е. страница будет корректно отображается на любом устройстве вне зависимости от разрешения и соотношения экрана дисплея. Не смотря на всю мощь и, казалось бы, простоту, bootstrap предоставляет лишь шаблоны, для тонкой настройки и обеспечения всем необходимым функционалам основную часть HTML и CSS придется писать самому разработчику.

1.6 Анализ и выбор СУБД

Существует достаточно большое количество систем управления базами данных. Из всего их разнообразия, в основном, можно выделить MySQL, Microsoft SQL server и SQLITE. Рассмотрим их подробнее.

MySQL представляет собой свободную реляционную СУБД. Она имеет открытый исходный код и позволяет поддерживать табличные базы данных с простой структурой и сложными условиями запросов. MySQL отличается гибкостью и высокой скоростью обработки информации, а также способностью синхронизации с другими базами данных.

Преимущества MySQL:

– бесплатное и свободное программное обеспечение;

– прекрасно документирована;

– предлагает много функций, даже в бесплатной версии;

– пакет MySQL включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы Linux, что позволяет устанавливать её элементарно просто;

– поддерживает набор пользовательских интерфейсов.

Недостатки MySQL:

– для бесплатной версии СУБД доступна только платная поддержка;

– недостаточная надежность. В вопросах надежности некоторых процессов по работе с данными MySQL уступает некоторым другим СУБД;

– низкая скорость разработки.

Microsoft SQL Server – система управления базами данных, которая позволяет эффективно управлять базами данных различных масштабов. Данная СУБД была разработана компанией Microsoft, вследствие чего имеет ее поддержку. Система позволяет обрабатывать запросы Transact-SQL.

Преимущества MSSQL:

– достаточно простой интерфейс;

– упрощенное развертывание, передача и интеграция больших данных;

– высокая производительность;

– поддержка постоянной памяти, что хранит данные после отключения питания

– самая быстрая интеграция в проекты Visual Studio;

Недостатки MSSQL:

– коммерческая версия для юридических лиц имеет высокую стоимость

– требует больше ресурсов, чем другие СУБД

SQLite - это компактная и легковесная реляционная база данных, которая подходит для использования в различных типах приложений, включая веб-сайты, мобильные приложения и настольные приложения.

Основной преимуществом SQLite является его простота и легковесность, которые обеспечивают быструю и эффективную работу даже на устройствах с ограниченными ресурсами. База данных SQLite не требует отдельного сервера и может быть встроена непосредственно в приложение, что упрощает ее использование и уменьшает нагрузку на систему.

SQLite поддерживает большинство основных функций, которые можно ожидать от реляционной базы данных, таких как поддержка транзакций, индексация данных, ограничения целостности и SQL-запросы. Она также поддержи-

вает расширения, которые позволяют расширять ее функциональность с помощью написания собственных модулей на языке Си.

SQLite имеет высокую степень надежности и безопасности, а также хорошо масштабируется. База данных может обрабатывать большое количество одновременных запросов и может использоваться для хранения большого объема данных.

В целом, SQLite - это простой, но мощный инструмент для хранения и управления данными, который может быть использован в различных типах приложений и средах разработки.

Достоинства SQLite:

– компактность и легковесность: SQLite является небольшой и легковесной базой данных, что обеспечивает ее быструю и эффективную работу даже на устройствах с ограниченными ресурсами;

– не требует отдельного сервера: SQLite может быть встроена в приложение, что упрощает ее использование и уменьшает нагрузку на систему;

– поддержка большинства основных функций: SQLite поддерживает большинство основных функций, которые можно ожидать от реляционной базы данных, таких как поддержка транзакций, индексация данных, ограничения целостности и SQL-запросы;

– высокая надежность и безопасность: SQLite обладает высокой степенью надежности и безопасности, и может быть использована для хранения конфиденциальной информации;

– хорошая масштабируемость: SQLite может обрабатывать большое количество одновременных запросов и может использоваться для хранения большого объема данных;

– простая интегрируемость с библиотекой Flask.

Недостатки SQLite:

– медленная скорость работы с большим объемом данных: SQLite может работать медленно при обработке большого объема данных, особенно если используются сложные запросы;

– ограниченная функциональность: SQLite не имеет некоторых функций, которые доступны в более мощных базах данных, таких как PostgreSQL или MySQL;

– ограниченная масштабируемость: SQLite может иметь проблемы с масштабированием на очень больших проектах;

– ограниченная поддержка: SQLite не имеет такой широкой поддержки, как более популярные базы данных, что может привести к проблемам с устареванием и безопасностью.

На основании проведенного анализа также была составлена сравнительная таблица:

Таблица 1.2 – Сравнительная таблица различных СУБД

Характеристики	**MySQL**	**Microsoft SQL Server**	**SQLite**
Бесплатное и свободное программное обеспечение	+	-	-
Прекрасно документирована	+	-	-
Предлагает много функций, даже в бесплатной версии	+	-	-
Пакет включен в стандартные репозитории распространенных дистрибутивов Linux	+	-	-

Продолжение таблицы 1.2

Поддерживает набор пользовательских интерфейсов	+	-	-
Доступна только платная поддержка	-	+	-
Недостаточная надежность	-	+	-
Низкая скорость разработки	-	+	-
Простой интерфейс	-	+	-
Упрощенное развертывание, передача и интеграция больших данных	-	+	-
Высокая производительность	-	+	-
Поддержка постоянной памяти	-	+	-
Быстрая интеграция в проекты Visual Studio	-	+	-
Не требует отдельного сервера	-	-	+
Поддержка большинства основных функций	-	-	+

Продолжение таблицы 1.2

Высокая надежность и безопасность	-	-	+
Хорошая масштабируемость	-	-	+
Простая интегрируемость с библиотекой Flask	-	-	+

В качестве СУБД в данном дипломном проекте была выбрана СУБД SQLITE, основной причиной этого является простая интегрируемость с библиотекой Flask.

1.7 Анализ и выбор среды разработки

Для разработки приложения, рассматриваемого в данной работе, была выбрана среда разработки PyCharm

PyCharm – это интегрированная среда разработки для python. Это очень мощный инструмент, который подойдет как для опытных, так и начинающих разработчиков.

Из всех возможностей, выделяются следующие. Отладка кода с помощью встроенного дебаггера PyDev. Позволяет найти ошибки в логике программы следуя шаг за шагом за процессом выполнения программы. Полезно, когда возникает глубокая логическая ошибка отследить которую очень сложно.

Подсветка синтаксиса. Основные ключевые слова и конструкции python выделяются на фоне остального кода, что позволяет избежать синтаксических ошибок в написании программы

Автодополнение кода. Программа понимает какие ключевые слова могут следовать после текущего ввода и предлагает автоматически вставить их. Очень

сильно ускоряет разработку программы и опять же уменьшает количество синтаксических ошибок

Анализ кода на соответствие PEP8. PEP8 – это соглашение разработчиков python по написанию программ на этом ЯВУ. PEP8 создан на основе рекомендации самого разработчика python Гуидо ван Россума. Согласованность с этим стандартом позволяет писать программы, которые понятны другим разработчикам, повышает читабельность кода и позволяет работать над проектами вместе с другими разработчиками без затруднений в понимании конструкций и логики программ.

PyCharm поставляется в двух версиях Community Edition и Professional Edition.

PyCharm Community Edition – это бесплатно распространяемая версия с открытым исходным кодом. Поддерживает все основные функции и возможности PyCharm.

Professional Edition – платная версия PyCharm, которая помимо всех базовых функций предоставляет дополнительный функционал в том числе и встроенную поддержку разработки с помощью Flask. Автоматическое развертывание проекта Flask, запуск сервера, отладка. Все это включено в пакет.

Для экономии средств в данном проекте будет использоваться бесплатная версия, а весь процесс развертывания приложения Flask и работы с ним будет проводиться разработчиком вручную.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

2.1 Инфологическое проектирование базы данных

Инфологическое проектирование базы данных - это процесс, который включает в себя разработку концептуальной модели данных, определение сущностей, атрибутов и связей между ними. Данное проектирование выполняется на этапе анализа и планирования базы данных и позволяет определить структуру базы данных, а также описать основные характеристики, которые будут использоваться в дальнейшей разработке системы.

Для создания приложения по продаже и аренды недвижимости необходимо провести инфологическое проектирование базы данных. В данном случае основными сущностями являются объекты недвижимости и клиенты портала. Каждый объект недвижимости имеет свой уникальный идентификатор, а также характеристики, такие как площадь, цена аренды или покупки, географическое расположение и другие. Каждый клиент портала имеет также свой уникальный идентификатор, а также персональные данные, такие как имя, адрес электронной почты и пароль.

Для связи между сущностями используются связи «один к многим». Так, каждый пользователь может купить или арендовать несколько объектов, и каждый объект недвижимости может быть куплен или арендован несколькими пользователями.

Таким образом, инфологическое проектирование базы данных является важным этапом в создании приложения для аренды и покупке недвижимости. Оно позволяет определить структуру базы данных, а также описать основные сущности, атрибуты и связи между ними. Правильное проектирование базы данных позволит создать эффективную систему, которая будет удобной и доступной для пользователей.

В результате проведённого инфологического проектирования было выделено 6 сущностей. В таблицах 2.1 – 2.6 представлен набор атрибутов каждой из них.

Таблица 2.1 – Сущность «Пользователь»

Атрибут	**Тип данных**
Идентификатор	Целочисленный
Адрес электронной почты	Строка
Пароль	Строка

Таблица 2.2 – Сущность «Роль»

Атрибут	**Тип данных**
Идентификатор роли	Целочисленный
Название роли	Строка

Таблица 2.3 – Сущность «Клиент»

Атрибут	**Тип данных**
Идентификатор клиента	Целочисленный
Серия паспорта	Строка
Номер паспорта	Строка
Адрес	Строка
Идентификатор пользователя	Целочисленный

Таблица 2.4 – Сущность «Недвижимость»

Атрибут	**Тип данных**
Идентификатор недвижимости	Целочисленный
Площадь	Число
Цена	Число
Географическое расположение	Строка
Дополнительные характеристики	Строка

Продолжение таблицы 2.4

Тип	Строка

Таблица 2.5 – Сущность «Договор»

Атрибут	**Тип данных**
Идентификатор договора	Целочисленный
Идентификатор клиента	Целочисленный
Идентификатор недвижимости	Целочисленный
Дата начала аренды	Дата
Дата окончания аренды	Дата

Таблица 2.6 – Сущность «Чек об оплате»

Атрибут	**Тип данных**
Идентификатор чека	Целочисленный
Идентификатор договора	Целочисленный
Сумма оплаты	Число
Дата оплаты	Дата и время

Помимо сущностей были также определены связи между ними (табл. 2.7).

Таблица 2.7 – Связи между сущностями

Сущность 1	**Сущность 2**	**Тип связи**
Недвижимость	Договор	Один к одному
Договор	Чек об оплате	Один к одному
Клиент	Пользователь	Один к одному
Пользователь	Роль	Многие ко многим
Клиент	Договор	Один ко многим

2.2 Логическое проектирование базы данных

Логическое проектирование базы данных - это процесс, в ходе которого концептуальная модель данных, разработанная на предыдущем этапе инфологического проектирования, преобразуется в специфические структуры данных, соответствующие выбранной СУБД (системе управления базами данных). Даталогическое проектирование учитывает особенности конкретной СУБД и включает в себя определение таблиц, полей, ключей, индексов и других объектов базы данных.

В контексте создания приложения для продажи и аренды недвижимости даталогическое проектирование базы данных направлено на перевод концептуальной модели, созданной на предыдущем этапе, в структуру, подходящую для выбранной СУБД. На этом этапе определяются таблицы, которые будут использоваться для хранения информации о объектах недвижимости, пользователях, договорах и других сущностях.

Важным аспектом даталогического проектирования является определение связей между таблицами. Например, связь "один к одному" между таблицами "Пользователи" и "Клиент" будет отражать возможность того, что один пользователь может иметь один профиль клиента. Аналогично, связь "один к многим" между таблицами "Клиент" и "Договор" будет показывать, что один клиент может быть связан со многими договорами.

В результате даталогического проектирования базы данных получается структура, состоящая из таблиц, полей, ключей и связей, которая будет использоваться для эффективного хранения и управления данными, связанными с арендой земельных участков.

Для предложенной ранее инфологической модели была разработана следующая даталогическая модель:

Таблица 2.8 - User

Поле	**Тип данных**
ID	Целочисленный

Продолжение таблицы 2.8

Email	Строка
Password	Строка

Таблица 2.9 - Role

Поле	**Тип данных**
ID	Целочисленный
Name	Строка

Таблица 2.10 - Client

Поле	**Тип данных**
ID	Целочисленный
PassportSeries	Строка
PassportNumber	Строка
Address	Строка
UserID	Целочисленный

Таблица 2.11 – Property

Поле	**Тип данных**
ID	Целочисленный
Area	Число
Price	Число
Location	Строка
Description	Строка

Таблица 2.12 – Contract

Поле	**Тип данных**
ID	Целочисленный
ClientID	Целочисленный

Продолжение таблицы 2.12

PropertyID	Целочисленный
StartDate	Дата
EndDate	Дата

Таблица 2.13 – PaymentReceipt

Поле	**Тип данных**
ID	Целочисленный
ContractID	Целочисленный
Amount	Число
DateTime	Дата и время

Помимо определения структуры хранения данных необходимо также определить следующие ограничения целостности:

Ограничение первичного ключа (Primary Key Constraint):

– в таблице "User" поле "ID" должно быть уникальным и не может содержать значений NULL;

– в таблице "Role" поле "ID" должно быть уникальным и не может содержать значений NULL;

– в таблице "Client" поле "ID" должно быть уникальным и не может содержать значений NULL;

– в таблице "Property" поле "ID" должно быть уникальным и не может содержать значений NULL;

– в таблице "Contract" поле "ID" должно быть уникальным и не может содержать значений NULL;

– в таблице "PaymentReceipt" поле "ID" должно быть уникальным и не может содержать значений NULL.

Ограничение внешнего ключа (Foreign Key Constraint):

– в таблице "Contract" поле "ClientID" является внешним ключом, который ссылается на поле "ID" в таблице "Client". Это ограничение гарантирует, что каждый договор связан с существующим клиентом;

– в таблице "Contract" поле "PropertyID" является внешним ключом, который ссылается на поле "ID" в таблице "Property". Это ограничение гарантирует, что каждый договор аренды связан с существующим объектом недвижимости;

– в таблице "PaymentReceipt" поле "ContractID" является внешним ключом, который ссылается на поле "ID" в таблице "Contract". Это ограничение гарантирует, что каждый чек об оплате связан с существующим договором.

Ограничение NOT NULL (NOT NULL Constraint):

– в таблице "Client" поле "PassportSeries" и "PassportNumber " не могут содержать значения NULL, так как они обязательны для каждого арендатора;

– в таблице "PaymentReceipt" поле "Amount" и "DateTime" не могут содержать значения NULL, так как они обязательны для каждого чека об оплате.

2.3 Физическое проектирование базы данных

Физическое проектирование баз данных является одной из важнейших частей разработки информационных систем. Этот этап предшествует непосредственной реализации БД и включает в себя определение структуры БД, выбор методов хранения данных, оптимизацию запросов и многое другое.

На первом этапе физического проектирования определяется структура базы данных. Этот этап включает в себя определение типов данных, которые будут использоваться для хранения информации, а также определение связей между различными таблицами. Также на этом этапе проводится анализ производительности БД, так как структура БД напрямую влияет на ее эффективность.

На втором этапе выбираются методы хранения данных. В определении методов хранения данных учитываются критерии, такие как скорость доступа к данным, размеры БД, требуемый уровень безопасности и т. д. Одним из наиболее распространенных методов хранения данных является использование реляционной СУБД.

На третьем этапе проводится оптимизация запросов к БД. Для получения эффективной работы базы данных необходимо использовать правильные запросы, а также оптимизировать их. Оптимизация запросов включает в себя проектирование индексов, использование правильных типов соединений между таблицами и т. д.

Таким образом, физическое проектирование БД является важным этапом создания информационных систем. В определении структуры БД, выборе методов хранения данных и оптимизации запросов к БД учитываются многие факторы, которые влияют на эффективность и безопасность БД.

В результате проведенного физическое проектирования были получены следующие наборы полей, типов данных и ограничений:

Таблица 2.14 – User

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
Login	VARCHAR(50)	NOT NULL
Password	VARCHAR(50)	NOT NULL
Role	INT	FOREIGN KEY

Таблица 2.15 – Role

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
Name	VARCHAR(50)	NOT NULL

Таблица 2.16 – Client

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
PassportSeries	VARCHAR(10)	NOT NULL
PassportNumber	VARCHAR(20)	NOT NULL
Address	VARCHAR(100)	NOT NULL

Продолжение таблицы 2.16

UserID	INT	FOREIGN KEY

Таблица 2.17 – Property

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
Area	DECIMAL(10,2)	NOT NULL
RentalPrice	DECIMAL(10,2)	NOT NULL
Location	VARCHAR(100)	NOT NULL
Description	VARCHAR(100)	NOT NULL

Таблица 2.18 – Contract

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
ClientID	INT	FOREIGN KEY
PropertyID	INT	FOREIGN KEY
StartDate	DATE	NOT NULL
EndDate	DATE	NOT NULL

Таблица 2.19 – PaymentReceipt

Поле	**Тип данных**	**Ограничения**
ID	INT	PRIMARY KEY
ContractID	INT	FOREIGN KEY
Amount	DECIMAL(10,2)	NOT NULL
DateTime	DATE	NOT NULL

На рисунке 2.1 представлена ER-диаграмма разработанной базы данных.

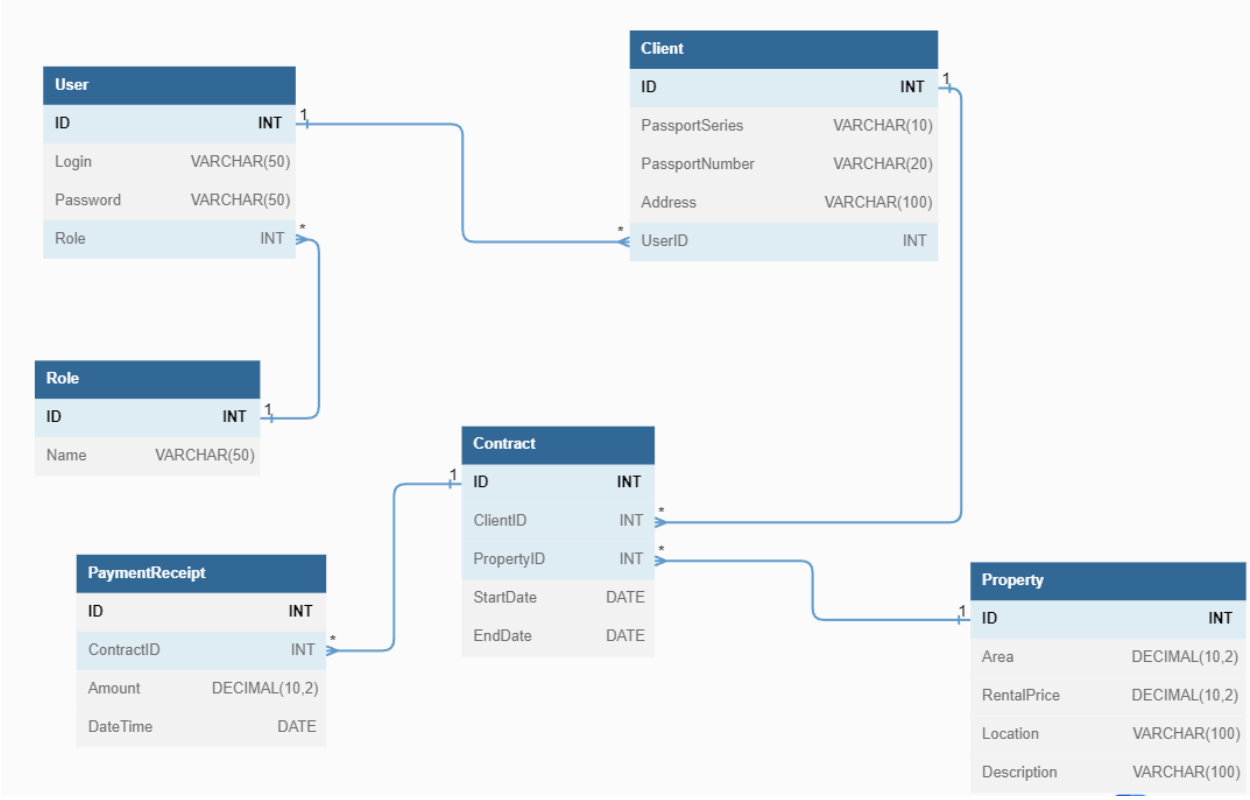


Рисунок 2.1 – ER-диаграмма

Спроектированная база данных находится в 3ей нормальной форме, так как значения всех атрибутов всех сущностей атомарны; все атрибуты, не входящие в первичный ключ, связаны с ним полной функциональной зависимостью; ни один ключевой атрибут функционально не зависит от любого другого не ключевого атрибута.

SQL код создания всех таблиц с ограничениями расположен в приложении Б.

2.4 Архитектура приложения

Архитектура приложений — это структурный принцип, по которому создано приложение. Каждому архитектурному виду свойственны свои характеристики, свойства и отношения между компонентами. Компонент — мелкая или крупная логическая и независимая часть архитектурной системы приложения.

Архитектуру приложений важно продумывать перед стартом разработки. Четко продуманная архитектура — это залог работоспособности и удобства бу-

дущей программы. Реакция программы на действия пользователя, возможность справляться с высокими нагрузками, зависания — все это и другие потенциальные проблемы зависят от того, насколько качественно изначально продумали архитектуру приложения.

В разрабатываемом приложении используется многослойная архитектура.

Многослойная архитектура разделяет программу на следующие слои:

– слой представления отвечает за интерфейс пользователя;

– слой бизнес-логики отвечает за функционал и логику приложения и не отвечает за интерфейс;

– слой передачи данных отвечает за работу с базами данных и обработку информации.

Каждый элемент в приложении «проходит» через все слои. Такая архитектура считается:

– Простой в реализации;

– Абстрактной;

– Защищенной за счет изолированности каждого слоя;

– Легко управляемой и масштабируемой.

– Частным паттерном реализации такой архитектуры является MVC.

Model View Controller (MVC) - это схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер, таким образом, что модификация каждого компонента может осуществляться независимо. Это очень удобно поскольку если необходимо изменить один из бизнес-процессов, то нет необходимости лезть во все участки кода, а лишь отредактировать изолированный участок. Причем это касается также и двух других компонентов, при современном развитии приложений дизайн устаревает буквально каждый день. И дизайнеры, разработчики могут вносить новые элементы дизайна и обновлять внешний вид страниц, не опасаясь за работоспособность других модулей.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Как уже ранее отмечалось основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;

Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;

Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

На рисунке 2.1 представлена схема работы MVC паттерна с точки зрения пользователя.

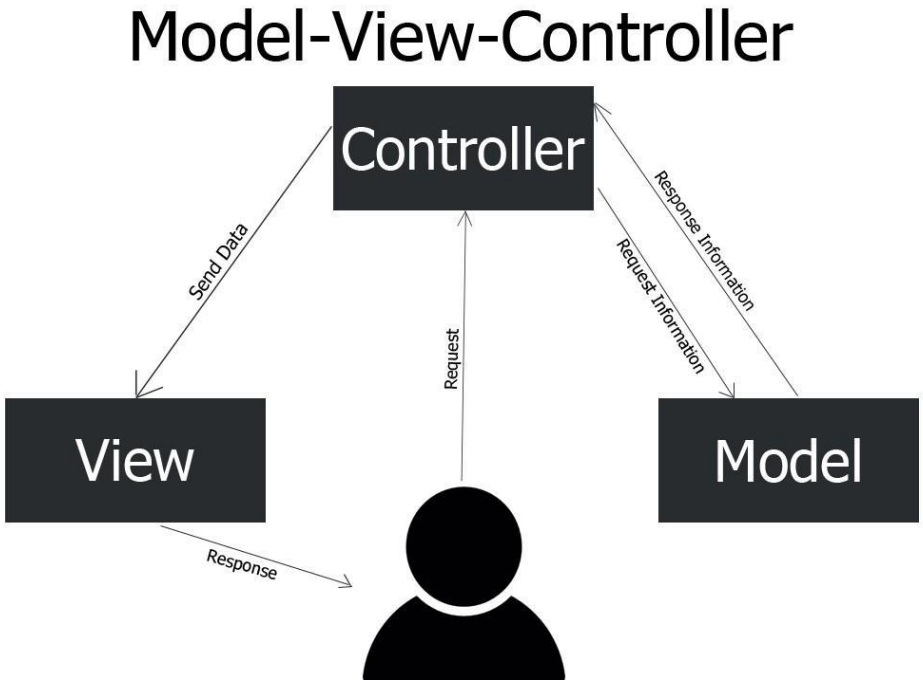


Рисунок 2.2 – Схема работы MVC

Разрабатываемый проект состоит из двух компонент: клиента и сервера.

На стороне клиента выступает браузер пользователя способны интерпретировать и выполнять JavaScript код, а также обрабатывать и отображать визуальную составляющую с помощью тегов HTML и CSS.

На стороне сервера используется один из множества различных языков программирования, которые вкупе с веб-фреймворками представляют полноценный веб-сервер способный обрабатывать запросы пользователей и определяет логику работы приложения.

Для обмена данными между клиентской и серверной частями проекта используется HTTP протокол. Клиентский браузер отправляет запросы на сервер с использованием методов GET и POST. Метод GET используется для получения данных с сервера, например, при запросе страницы или изображений. Метод POST позволяет отправлять данные на сервер для их обработки, например, при регистрации пользователя или отправке формы.

На рисунке 3.2 представлена схема общения клиента с сервером.

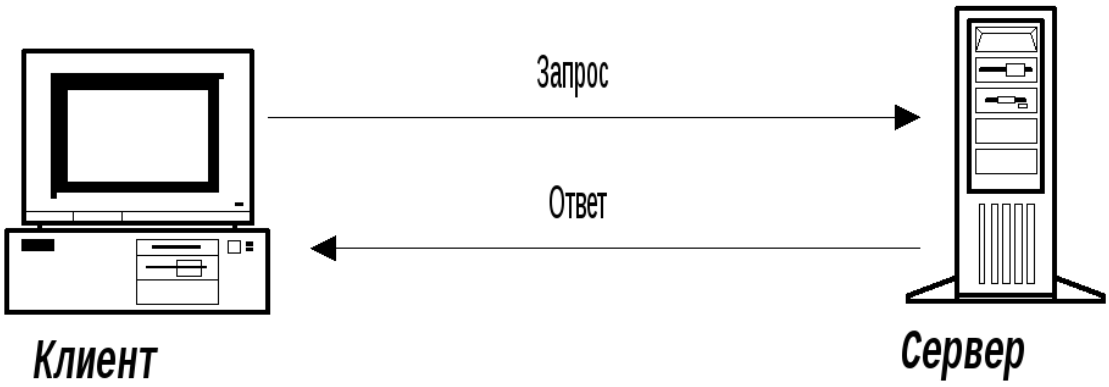


Рисунок 2.3 – Общение между сервером и клиентом

Использование HTTP протокола позволяет обеспечить взаимодействие между клиентом и сервером.

2.5 Работа с базой данных с помощью технологии ORM

ORM (Object-Relational Mapping) – это технология программирования которая связывает базы данных с концепциями объектно-ориентированных программ, создавая «виртуальную объектную БД».

Например, для работы с таблицой «Пользователь» создается отдельных класс User, где поля класса соответствуют атрибутам этой таблицы, и у класса должны существовать методы для получения значений из этих полей.

Достоинства технологии ORM:

– возможность оперировать элементами языка программирования, а не элементами реляционной модели данных;

– автоматическое создание SQL запросов (те самые методы для получения значений полей класса);

– ORM избавляет от необходимости работы с SQL и проработки тонны SQL кода который однообразен и подвержен ошибкам;

– ORM изолирует код программы от данных.

Недостатки технологии ORM:

– дополнительная нагрузка на программиста, поскольку ему необходимо изучать этот дополнительный уровень абстракции;

– отсутствие явных SQL запросов мешает в некоторых случаях оптимизировать получение данных.

Если в реализации ORM присутствуют ошибки, то отлавливать их очень трудно. Механизм работы ORM схематично изображен на рисунке 2.4.

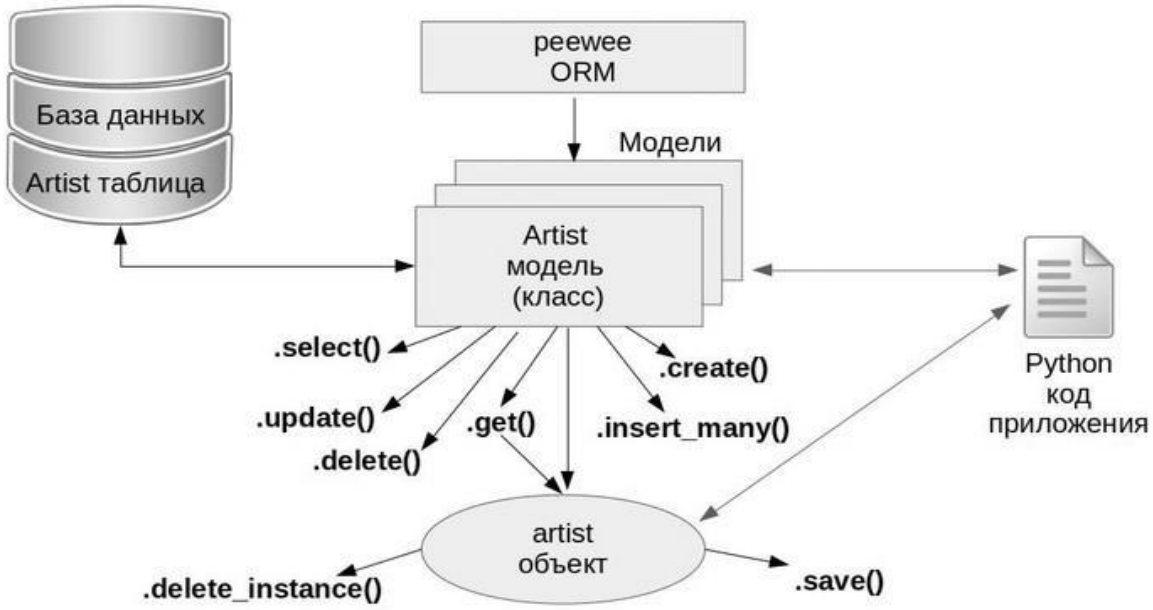


Рисунок 2.4 – Иллюстрация работы механизма ORM

2.6 Информационная безопасность

Для обеспечения конфиденциальности в проекте будет применено транспортное шифрование.

Транспортное шифрование – это способ передачи данных, в котором информация предварительно шифруется перед отправкой на сервер, после чего дешифрируется сервером и вновь шифруется для отправки получателю. Такое шифрование защищает личные данных от перехвата в момент передачи и обеспечивает высокий уровень безопасности. Доступ к информации имеет лишь обладатель «ключа».

Для шифрования будет применятся алгоритм RSA.

RSA – это криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых

чисел. Это ассиметричный алгоритм т.е. для шифрования\дешифрования данных необходимо владеть сразу двумя ключами «открытым» и «закрытым».

Рисунок 2.5 – Иллюстрация работы алгоритма RSA

На рисунке 2.5 изображен пример использования RSA. Python поставляется со стандартной библиотекой RSA которая реализует алгоритм шифрования RSA.

```
import rsa
(pubkey, privkey) = rsa.newkeys(512) # генерация ключей

message = 'edu'.encode('utf8') # шифрование
message = rsa.decrypt(crypto, privkey) # дешифрование
```

Рисунок 2.6 – Пример использования RSA в Python

Помимо шифрования также важна защита пароля и прочих аутентификационных данных пользователя непосредственно от «кражи», для обеспечения этого будет применено хеширование.

Хеширование отличается от шифрования тем, что восстановить исходные данные из хеш-строки невозможно, но для правильной работы необходимо гарантировать однозначное хеширование т.е. что при одинаковом входном потоке байтов гарантируется одинаковая хеш-строка. Пароли пользователей будут храниться в базе данных в хешированном виде, так что даже при хакерской атаке и краже БД злоумышленники не смогут получить доступ к пользовательским страницам.

В качестве алгоритма хеширования будет использоваться алгоритм MD5 – это 128 битный алгоритм шифрования, разработанный профессором Рональдом Л.

Алгоритм получает на вход строки и возвращает хеш который состоит из 128 бит и обычно представляется как 32-значное шестнадцатеричное значение, для каждых входных данных это значение уникально и вероятность совпадения очень мала. Такой алгоритм обеспечивает высокий уровень защиты данных. На рисунке 2.6 представлена иллюстрация принципов работы алгоритмов хеширования.

Рисунок 2.7 – Иллюстрация принципа хеширования

2.7 Проектирование клиентской части веб-приложения

В современной веб-разработке существует несколько подходов к разработке визуальной части интерфейса.

Самым простым вариантом является выдача пользователю готовых и статичных html файлов, однако такой подход устарел и невозможен при разработке веб-приложений с динамическим изменением контента или реакцией на действия пользователя.

Следующий способ, который является новым подходом и используется повсеместно, генерация html страниц на стороне клиента. В таком случае необходимо использовать один из frontend фреймворков, которые будут общаться с

сервером и генерировать контент на страницах «на месте». У такого подхода есть свои плюсы, однако он избыточен и не подходит при разработке локальных и небольших приложений где основная нагрузка идет на серверную часть.

В разрабатываемом приложении будет применяться технология Server Side Rendering.

Server Side Rendering (SSR) – это подход к построению пользовательского интерфейса, при котором страницы динамически генерируются на стороне сервера, и поставляются со всеми необходимыми формами клиенту. Для реализации подобной технологии используют «шаблонизаторы».

Шаблонизатор – это библиотека\программный модуль который генерирует странички с html содержимым (и любым другим) на основе своего внутреннего синтаксиса, который может отличаться от синтаксиса языка программирования.

Необходимо также отметить что такой подход более безопасен поскольку данные формируется на стороне сервера, и клиент никак не участвует в этом процессе.

В данной работе используется шаблонизатор Jinja2.

Jinja2 – это шаблонизатор для языка программирования python. Flask использует его для обеспечения динамичности веб страницам. Библиотека Flask содержит метод render_template(template) в качестве аргументов этот метод получает html-страницу после чего обрабатывает страницу в соотв. с синтаксисом Jinja.

Принцип работы этого шаблонизатора заключается в том, что внутри html документа размещаются специальные теги, и с помощью этих тегов flask обрабатывает страницу и вставляет контент там, где это необходимо. С помощью этого можно добавлять различные счетчики, или отобразить текстовую метку с именем пользователя, которая для каждого пользователя будет разная, поскольку документ будет обработан шаблонизатором и в нужную часть будет

вставлено имя пользователя, который отправил запрос на просмотр этой страницы.

Также с помощью Jinja2 можно оптимизировать код и искоренить его дублирование. К примеру, тег <head>, который почти в каждой странице должен быть одинаков, можно поместить в отдельный документ, назовем его base.html после чего во всех следующих страницах с помощью тега extends автоматически встраивать нужный фрагмент кода.

Работа шаблонизатора Jinja представлена на примере страницы login.html (рис. 2.7).

```
{% extends "base.html" %}
{% block body %}
<div class="container bg-light rounded mt-2 shadow p-4">
    <h1>Пожалуйста авторизуйтесь</h1>
    {% if message %}
    <div class="h3 text-danger">
        {{message}}
    </div>
    {% endif %}
    <form class="form-control p-3" action="/login" method="post">
        <div class="mb-3">
            <input class="form-control" type="text" name="username" placeholder="Логин" required/>
        </div>
        <div class="mb-3">
            <input class="form-control" type="password" name="password" placeholder="Пароль" required/>
        </div>
        <div class="mb-3">
            <button class="btn btn-outline-primary" type="submit">Войти</button>
            <a href="/registration" class="btn btn-outline-primary">Зарегистрироваться</a>
        </div>
    </form>
</div>
{% endblock %}
```

Рисунок 2.8 – login.html

В начале документа размещена конструкция {% extends <template> %} с помощью этой конструкции мы сообщаем шаблонизатору что фрагмент кода, который заключен в тегах {% block content %} и {% endblock %}, будет встраиваться в страницу template. С помощью подобного механизма наследования, можно не описывать элементы страницы, которые повторяются на всех страницах. Одним из таких элементов является блок <head>.

Необходимо еще раз отметить что сама html страница со всеми нужными элементами формируется на стороне сервера, пользователь лишь получает готовый html код.

2.8 Проектирование серверной части веб-приложения

Как отмечалось ранее в данном проекте используется библиотека Flask которая отвечает за серверную логику приложения.

На рисунке 2.8 представлены компоненты серверной части приложения, а также взаимодействие между ними.

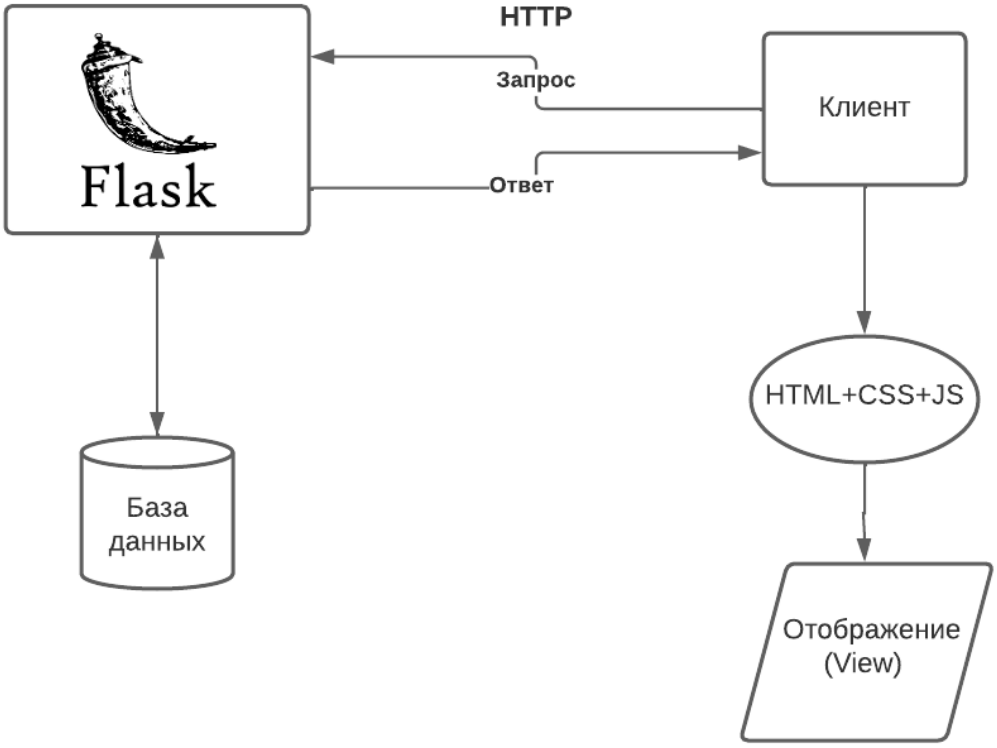


Рисунок 2.9 – Схема работы серверной части приложения

Для маршрутизации запросов Flask использует обычные функции python.

Рисунок 2.10 – Пример функции для маршрутизации запросов

На рисунке 2.10 изображен фрагмент кода(функции) отвечающий за обработку запросов по адресу /login, и показывающая соотв. страницу авторизации.

Как видно на рисунке функция «обернута» специальным декоратором, который сообщает по какому URL следует ожидать запрос.

Декораторы в python — это "обёртки", которые дают возможность изменить поведение функции, не изменяя напрямую её код. Разработчики flask описали несколько таких декораторов [6].

Декоратор @app.route управляет маршрутизацией и сообщает функции по какому адресу необходимо ожидать запрос. Имеет дополнительный параметр methods который может принимать значения GET и POST в зависимости от типа, запроса, который ожидается.

Все функции для маршрутизации располагаются в файле index.py. Структура директории проекта, а также описание основных файлов и папок представлены на рисунке 2.10 и в текстовом пояснении соответственно.

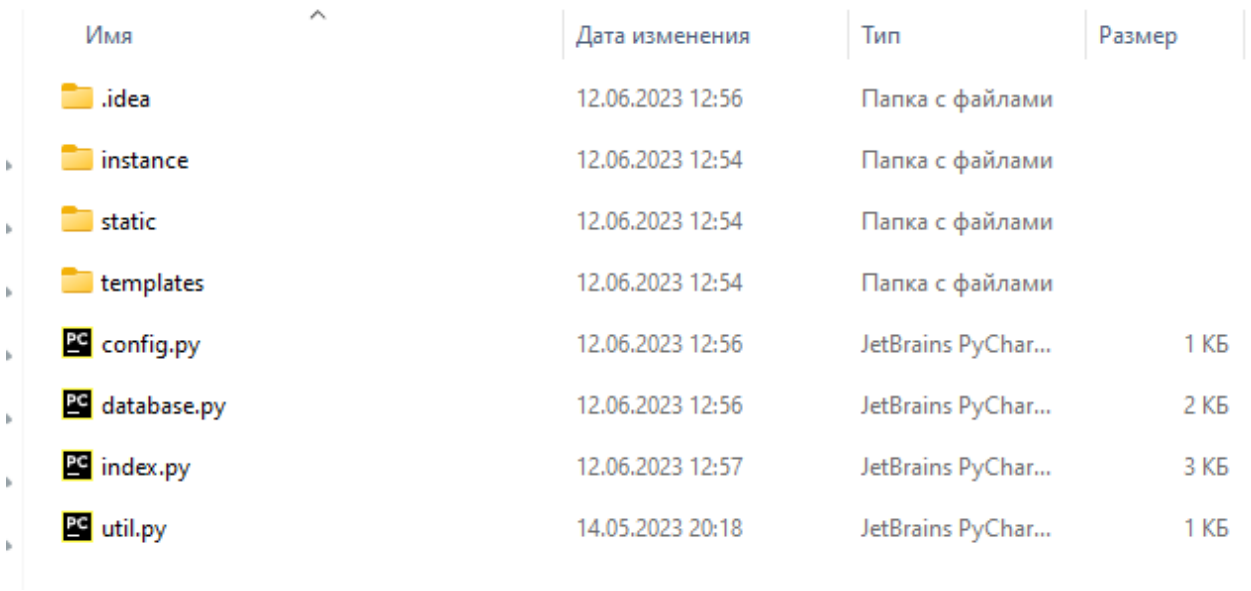


Рисунок 2.11 – Структура директории проекта

Ниже приводится краткое описание файлов и папок, представленных на рисунке.

.idea – папка, содержащая конфигурационные файлы самой среды разработки, не относится непосредственно к работе приложения

instance – папка содержащая файл базы данных SQLite

static – содержит медиа файлы, а также таблицы стилей для оформления веб страниц.

templates – это папка, содержащая html шаблоны веб страниц, такие как страница с каталогом недвижимости или страница авторизации.

index.py – основной python-файл проекта. Содержит все функции маршрутизации, а также точку старта программы.

database.py – файл, содержащий ORM представление классов базы данных.

config.py – файл конфигурационные настройки приложения.

2.9 Проектирование интерфейса

При разработке пользовательского интерфейса будет использоваться библиотека Bootstrap5.

Bootstrap5 – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типо-

графики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Ниже представлен пример компонентов bootstrap – кнопки (рис. 2.11).

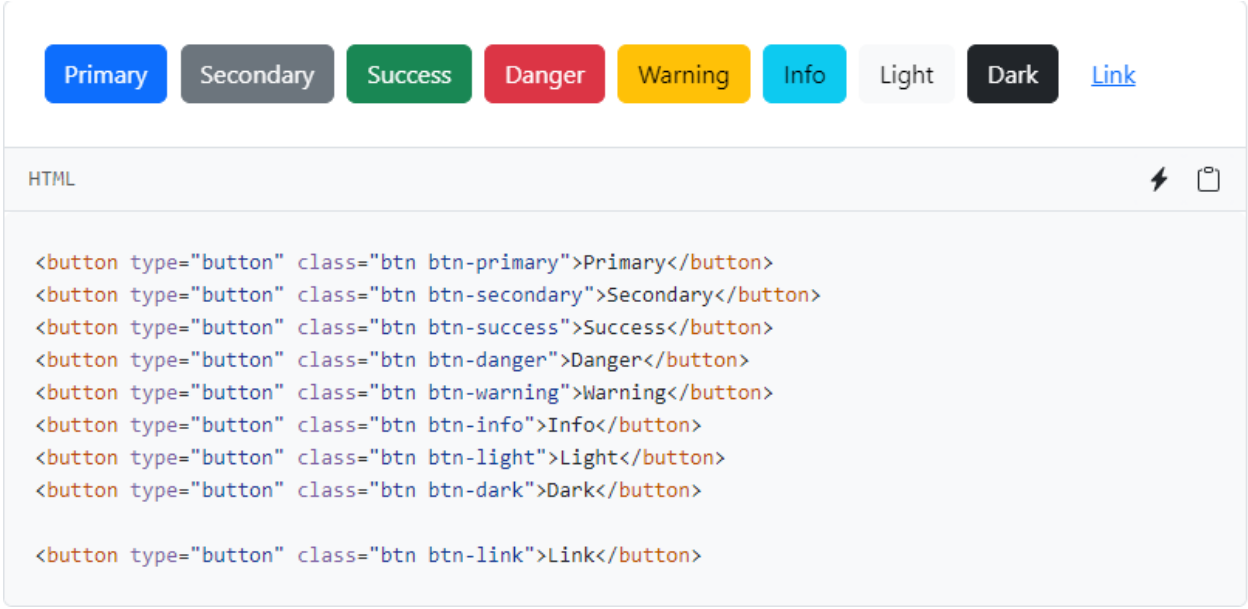


Рисунок 2.12 – Примеры оформления кнопок Bootstrap

Помимо кнопок, Bootstrap также предоставляет и другие компоненты: к примеру, списки (рис. 2.12) или заголовки текста (рис. 2.13).

Рисунок 2.13 – Пример оформления списка Bootstrap

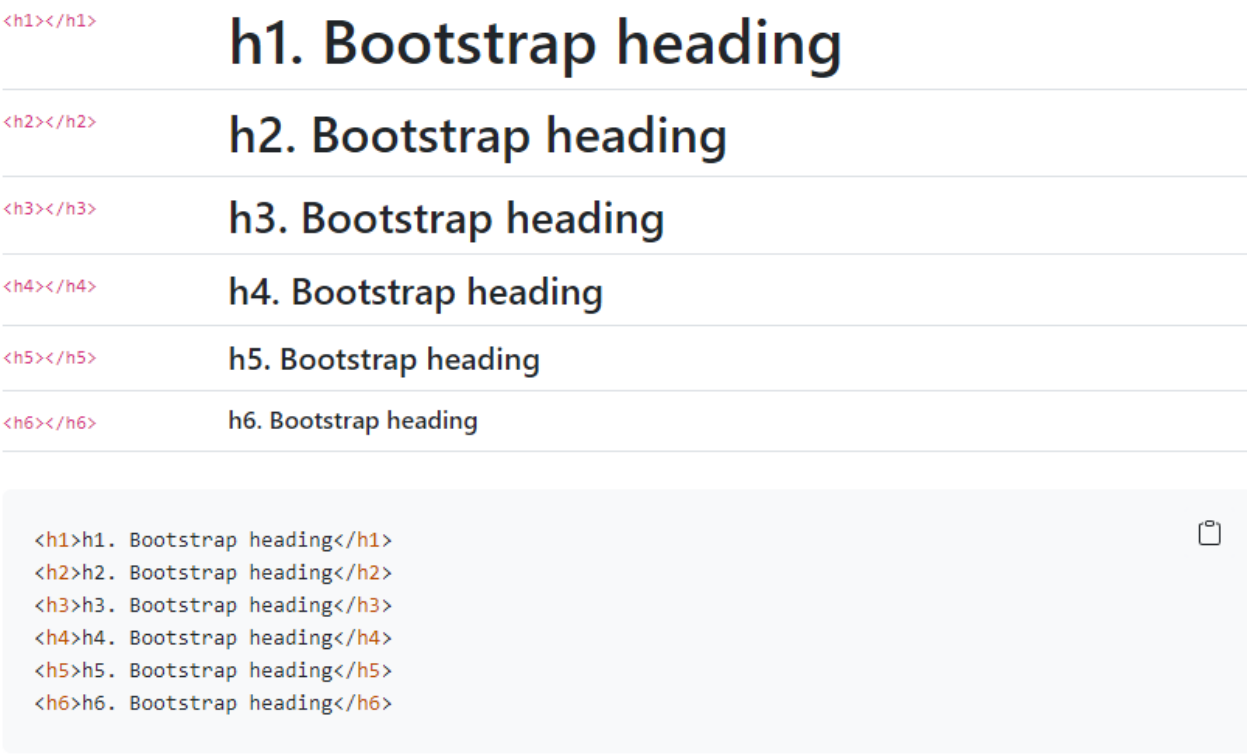


Рисунок 2.14 – Пример оформления заголовков Bootstrap

Также в библиотеке присутствуют и модальные окна, реализованные с помощью нативного JavaScript.

Каждый компонент предоставляемый библиотекой Bootstrap можно кастомизировать и настроить под себя.

Отдельно необходимо отметить систему flex верстки. Основным структурным компонентом такой верстки являются строки и столбцы.

Как они работают:

– столбцы основаны на архитектуре flexbox сетки. Flexbox означает, что у нас есть опции для изменения отдельных столбцов и изменения групп столбцов на уровне строки. Выбирая эти опции, определяется поведение того как столбцы увеличиваются, уменьшаются или иным образом изменяются;

– при создании макетов сетки весь контент размещается в столбцах. Иерархия сетки Bootstrap идет от контейнера к строке и столбцу к контенту. В редких случаях можно комбинировать контент и столбец, но необходимо помнить, что это может привести к непредвиденным последствиям;

– Bootstrap включает предопределенные классы для создания быстрых и отзывчивых макетов. С шестью точками останова и дюжиной столбцов на каждом уровне сетки у нас есть десятки классов, уже созданных для того, чтобы создавать желаемые макеты.

Как видно на рисунке 2.13, для «присваивания» какого-либо стиля необходимо у каждого тега объявить атрибут class. В данном атрибуте в последовательном порядке указываются все наследуемые стили, которые предоставляются библиотекой Bootstrap5.

К примеру добавление стиля «btn» оформит любой компонент как кнопку, она будет имеет тень, эффект нажатия (рис. 2.14), и границы.

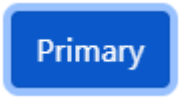


Рисунок 2.15 – Эффект нажатия

В официальной документации Bootstrap представлено огромное множество компонентов, с помощью которых можно оформить практически любую страницу.

Помимо этого, каждый компонент можно изменить по-своему, поскольку css файлы хранятся локально, и автор может изменить их по своему желанию.

Такой подход очень гибок, поскольку не всегда стандартные решения подходят в рамках проекта.

3 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Безопасность

Для проверки защиты от угроз использовался инструмент OWASP ZAP.

OWASP ZAP (Zed Attack Proxy) - это бесплатный и открытый инструмент для тестирования на проникновение и безопасности веб-приложений. OWASP ZAP разрабатывается сообществом Open Web Application Security Project (OWASP) и предоставляет разработчикам и тестировщикам возможности для обнаружения уязвимостей в веб-приложениях и проведения атак с целью оценки их безопасности.

Основные возможности OWASP ZAP включают:

– пассивное и активное сканирование: OWASP ZAP может сканировать веб-приложение и автоматически обнаруживать уязвимости, такие как межсайтовый скриптинг (XSS), SQL-инъекции, нарушения управления сеансами и другие;

– проксирование и перехват трафика: Запуская OWASP ZAP в качестве прокси-сервера, можно перехватывать и анализировать трафик между клиентом и сервером. Это позволяет обнаруживать и исправлять уязвимости, связанные с передачей данных;

– инструменты для взлома: OWASP ZAP предоставляет набор инструментов для проведения активных атак, таких как инъекции параметров, фаззинг, перебор словарей и другие, чтобы проверить стойкость веб-приложения к различным видам атак;

– отчеты и анализ: OWASP ZAP может генерировать детальные отчеты о найденных уязвимостях и предоставлять различные аналитические данные, которые помогут лучше понять и оценить уровень безопасности приложения.

Краткое содержание отчета представлено на рисунках 3.1 – 3.4.

Об этом отчете

Параметры отчета

Контексты

Контексты не были выбраны, поэтому по умолчанию были включены все контексты.

Места

Были включены следующие сайты:

- http://127.0.0.1:5000

(Если сайты не были выбраны, по умолчанию включались все сайты.)

Включенный сайт также должен находиться в одном из включенных контекстов, чтобы его данные были включены в отчет.

Уровни риска

Включено : Высокий , Средний , Низкий , Информационный

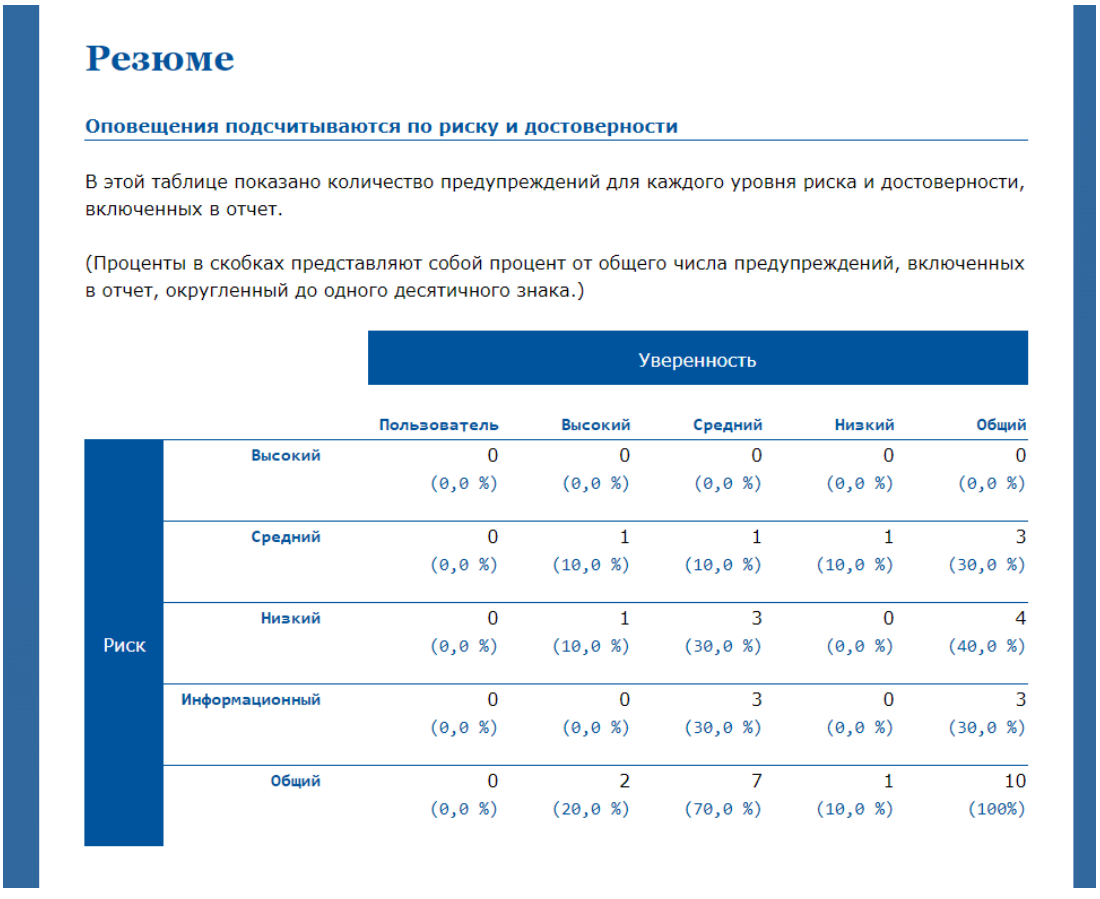
Исключено : Нет

Уровни достоверности

Включено : Пользователь Просмотр , Высокий , Средний , Низкий

Исключены : Пользователь Просмотр , Высокий , Средний , Низкий , Ложно-положительное

Рисунок 3.1 – Об отчете



Резюме

Оповещения подсчитываются по риску и достоверности

В этой таблице показано количество предупреждений для каждого уровня риска и достоверности, включенных в отчет.

(Проценты в скобках представляют собой процент от общего числа предупреждений, включенных в отчет, округленный до одного десятичного знака.)

		Уверенность				
		Пользователь	Высокий	Средний	Низкий	Общий
Риск	Высокий	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)
	Средний	0 (0,0 %)	1 (10,0 %)	1 (10,0 %)	1 (10,0 %)	3 (30,0 %)
	Низкий	0 (0,0 %)	1 (10,0 %)	3 (30,0 %)	0 (0,0 %)	4 (40,0 %)
	Информационный	0 (0,0 %)	0 (0,0 %)	3 (30,0 %)	0 (0,0 %)	3 (30,0 %)
	Общий	0 (0,0 %)	2 (20,0 %)	7 (70,0 %)	1 (10,0 %)	10 (100%)

Рисунок 3.2 – Резюме отчета

Тип оповещения	Риск	Считать
Заголовок Content Security Policy (CSP) не задан	Средний	11 (110,0 %)
Отсутствует заголовок (Header) для защиты от кликджекинга	Средний	4 (40,0 %)
Cookie без атрибута SameSite	Низкий	2 (20,0 %)
Сервер передает информацию о версии через поле заголовка HTTP-ответа «Сервер»	Низкий	16 (160,0 %)
Заголовок X-Content-Type-Options отсутствует	Низкий	9 (90,0 %)
Раскрытие ошибок приложений	Низкий	1 (10,0 %)
Современное веб-приложение	Информационный	5 (50,0 %)
Пользовательский агент Фаззер	Информационный	24 (240,0 %)
Раскрытие информации - подозрительные комментарии	Информационный	2 (20,0 %)
Общий		10

Рисунок 3.3 – Оповещения о найденных уязвимостях

Все полученные оповещения не представляют из себя реальные уязвимости и являются следствием локального размещения веб-приложения.

Отдельно необходимо отметить оповещение «Раскрытие информации – подозрительные комментарии». Данное оповещение возникает поскольку в отладочном режиме приложение отображает все ошибки сервера пользователю (рис. 3.4), подразумевая что пользователем выступает разработчик тестирующий приложение.

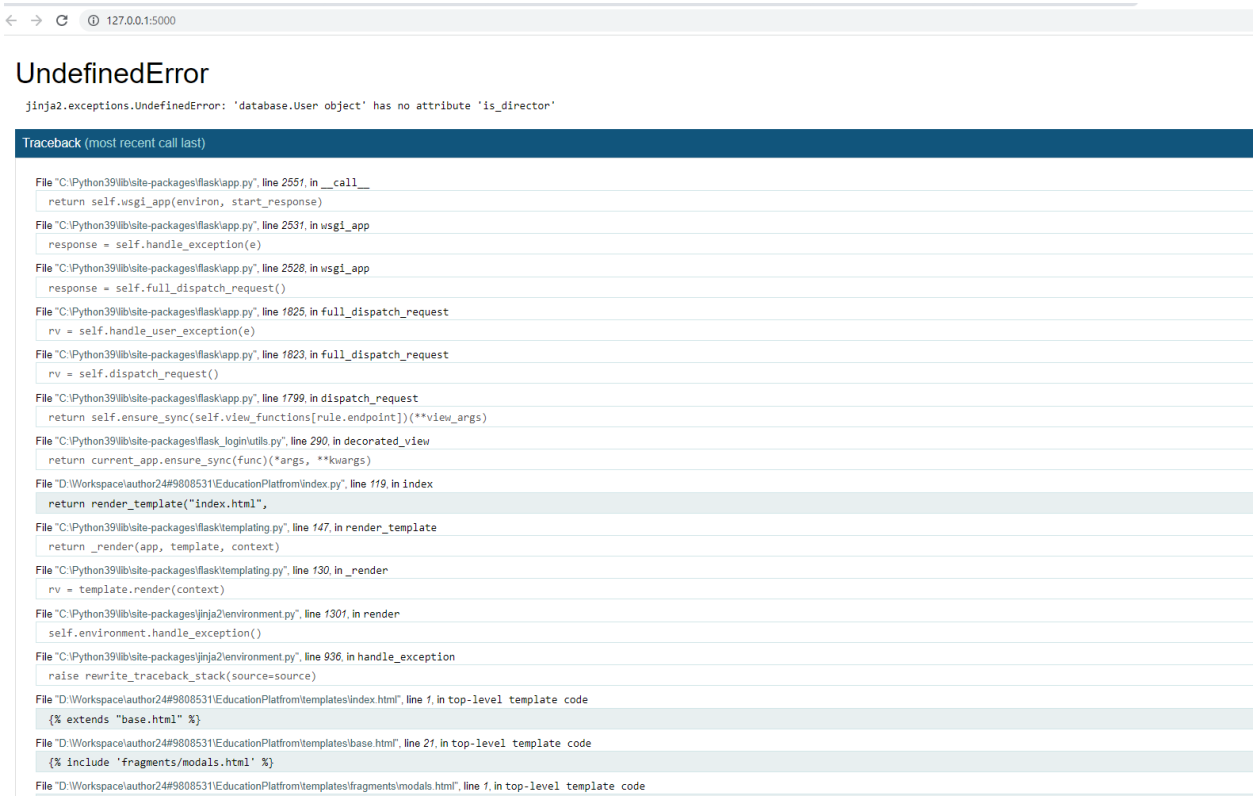


Рисунок 3.4 – Отображение информации об ошибке на стороне сервера

В результате проведенного исследования можно сделать вывод о том, что разработанное веб-приложение достаточно защищено от различных угроз в том числе от угроз из списка OWASP 10 [7].

3.2 Нагрузочное тестирование

Для проведения нагрузочного тестирования серверной части использовался специализированный инструментарий, позволяющий генерировать симулированные пользовательские запросы на сервер и проверять ответы сервера на корректность и время ответа.

В рамках тестирования приложения были протестированы следующие процессы:

– тестирование процесса регистрации\авторизации;

– тестирование процесса регистрации заявки на покупку недвижимости.

Для каждого из процессов были написаны специальные автоматические скрипты на языке python, которые имитировали пользователя и посылали различные запросы на веб-сервер.

Ниже представлены графики зависимости времени ответа сервера от количества запросов.

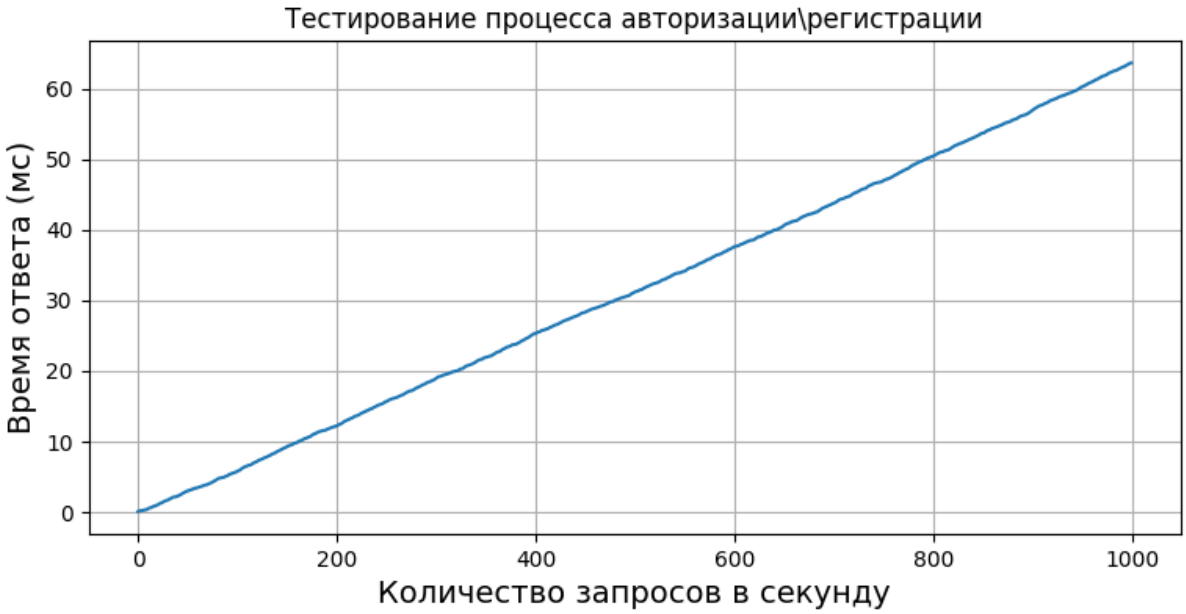


Рисунок 3.5 – Регистрация\Авторизация

Рисунок 3.6 – Регистрация заявки на покупку недвижимости

Исходя из данных, представленных на графиках (рис. 3.5 и рис. 3.6), можно сделать вывод, что серверная часть приложения успешно прошла нагрузочное тестирование и готова к эксплуатации.

3.3 Руководство пользователя

После перехода по адресу веб-приложения пользователь попадает на главную страницу (рис. 3.7).

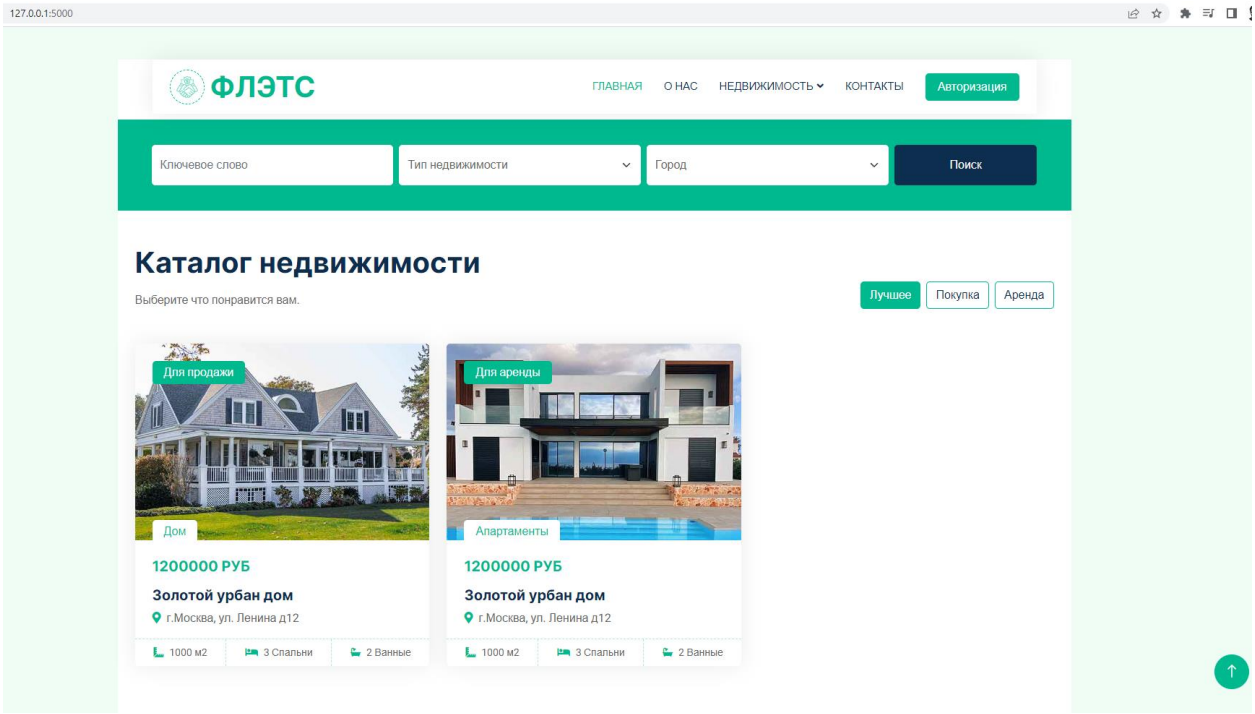


Рисунок 3.7 – Главная страница

В верхней части приложения расположено навигационное меню (рис. 3.8).

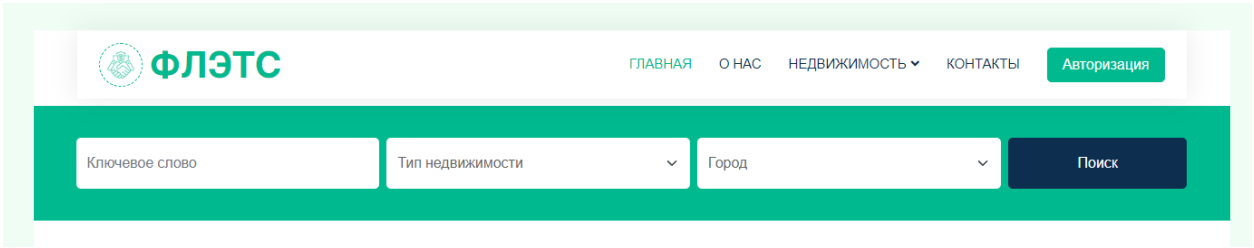


Рисунок 3.8 – Навигационное меню

При нажатии на кнопку «Авторизация» пользователь попадет на страницу авторизации (рис. 3.9).

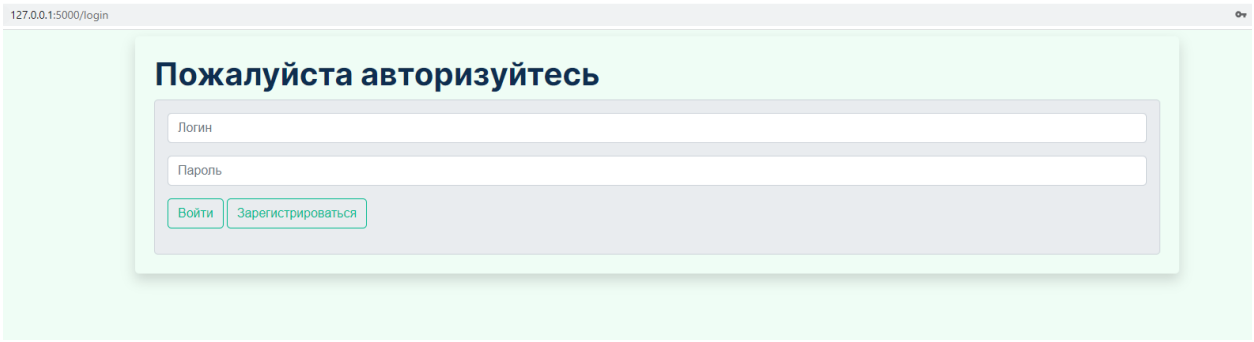


Рисунок 3.9 – Страница авторизации

Для дальнейшего использования приложения пользователю необходимо авторизоваться.

Для авторизации необходимо ввести учетные данные и нажать на кнопку «Войти», в случае если пользователь использует приложение впервые, то он может зарегистрировать новую учетную запись, нажав на кнопку «Зарегистрироваться» после чего откроется новая форма регистрации (рис. 3.10).

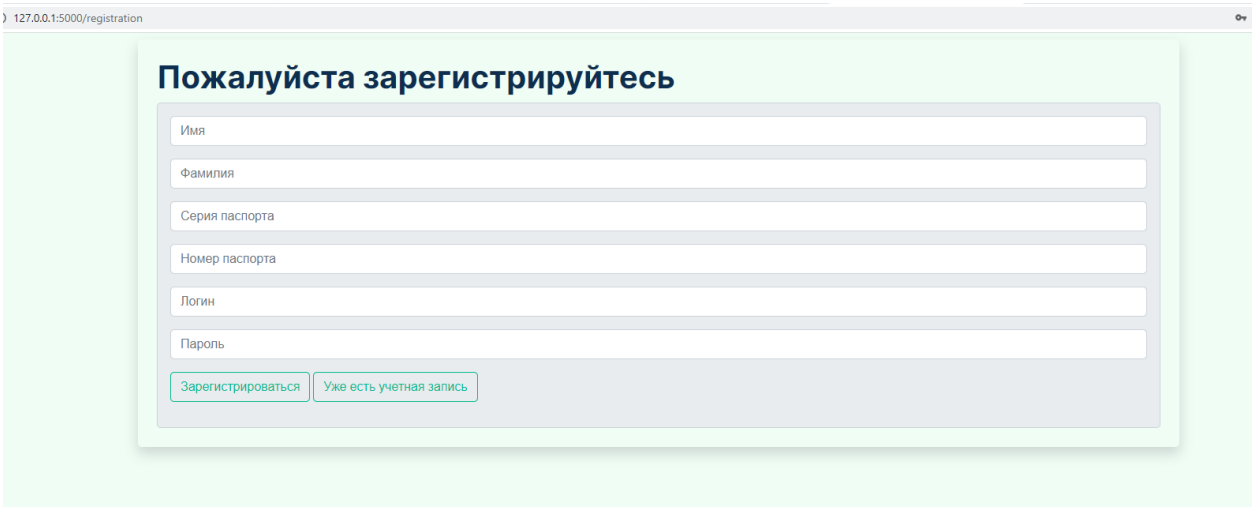


Рисунок 3.10 – Страница регистрации

После регистрации и авторизации под учетной записью пользователь попадает обратно на главную страницу.

В основной части страницы расположен каталог недвижимости. Доступны поиск и фильтрация.

К примеру, если отметить что интересуют только объявления о продаже, то часть объявлений будет исключена (рис. 3.12).

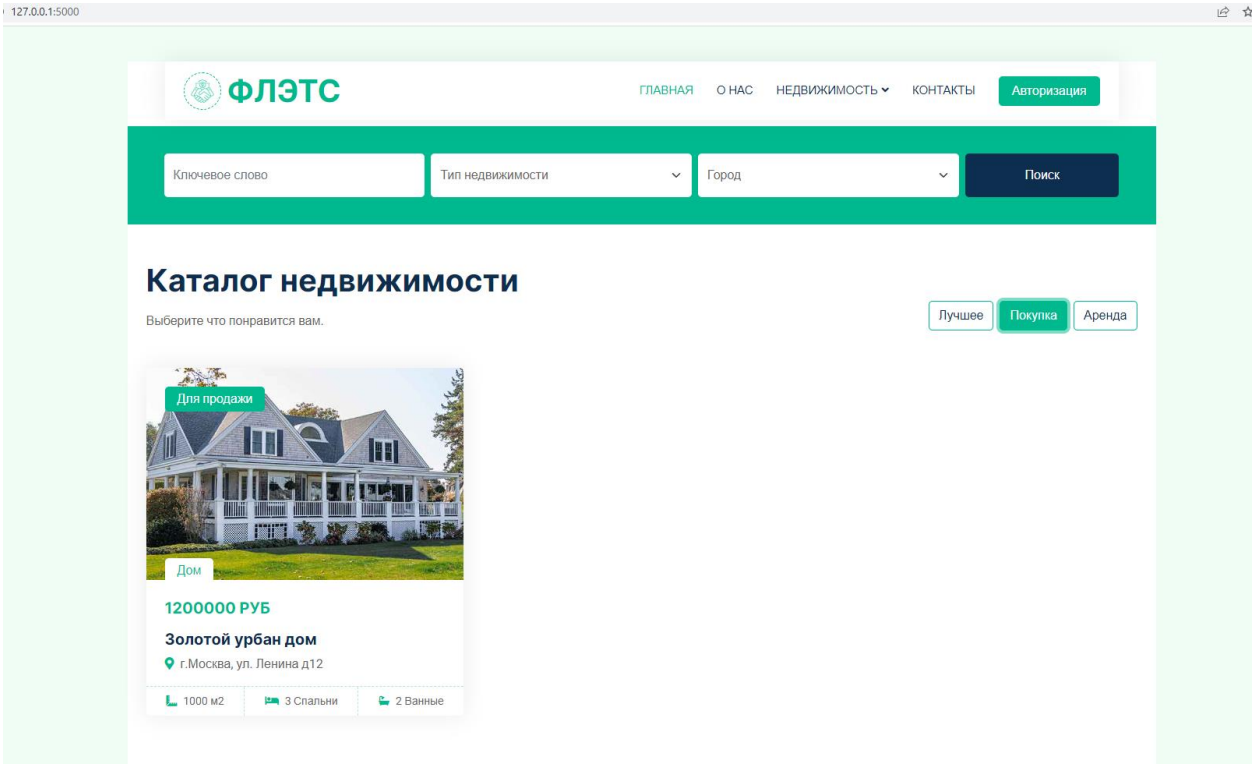


Рисунок 3.11 – Фильтрация каталога

Доступна также форма обратной связи для составления заявок на аренду и покупок или же для решения других возникших вопросов (рис. 3.13).

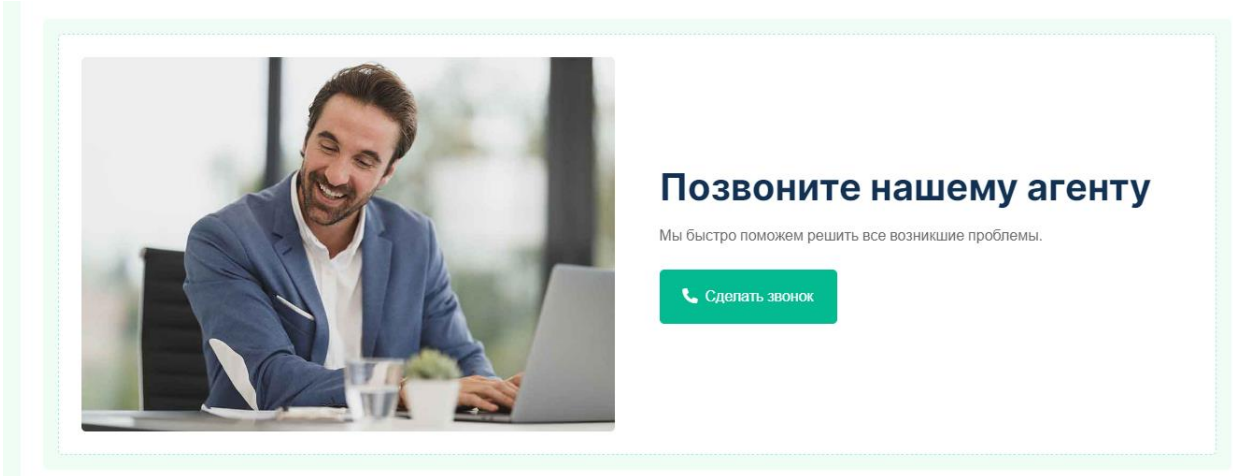


Рисунок 3.12 – Форма обратной связи

4 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ

4.1 Безопасность

В данном разделе рассматривается вопрос обеспечения безопасности работника на рабочем месте при разработке веб-приложения для аренды и продажи недвижимости. Он включает в себя анализ условий труда, организацию рабочего места и организацию графического интерфейса, которые направлены на создание безопасной и комфортной рабочей среды для сотрудников.

Рабочее место и организация.

Рабочее место веб-разработчика должно быть комфортным и подходящим для продолжительной работы за компьютером. Это включает в себя удобный стол и стул с правильной высотой, чтобы обеспечить правильную осанку и предотвратить нагрузку на спину и шею.

Рекомендуемая высота стола для большинства веб-разработчиков составляет около (70-75) см. Это позволяет сохранить правильное положение рук и предотвращает напряжение в плечах и спине.

Идеальная высота стула должна быть настроена таким образом, чтобы ваши ноги создавали прямой угол в коленях, а стопы полностью опиралась на пол. Рекомендуемая высота стула составляет примерно (40-50) см.

Стул должен иметь хорошую поддержку спины, особенно в области поясницы. Регулируемые спинки и подлокотники могут обеспечить дополнительный комфорт и поддержку.

Рабочая поверхность должна быть достаточно просторной для размещения компьютера, клавиатуры, мыши и других необходимых инструментов.

Рекомендуемый минимальный размер рабочей поверхности для веб-разработчика составляет примерно 120 см в ширину и (60-80) см в глубину. Однако, если вам требуется работать с несколькими мониторами или иметь дополнительное пространство для расстановки бумажных документов или других

инструментов, то размер рабочей поверхности может быть увеличен соответственно.

Важно также учесть возможность настройки и организации рабочего пространства в соответствии с личными предпочтениями. Регулируемые столы или рабочие столы с возможностью изменения высоты и угла наклона могут быть предпочтительными, так как они позволяют настроить рабочее пространство под свои индивидуальные требования.

Наконец, помимо размера рабочей поверхности, также важно обеспечить достаточное пространство для свободного перемещения ног, чтобы предотвратить скручивание и неудобство во время работы.

Важно иметь хорошее освещение рабочего места, чтобы избежать напряжения глаз и усталости. Оптимально использовать естественное освещение и добавить дополнительные источники света, если это необходимо. Кроме того, экраны компьютеров должны быть настроены на оптимальную яркость и контрастность, чтобы предотвратить усталость глаз и повысить четкость изображения.

Компьютерное оборудование и программное обеспечение.

Веб-разработчикам необходимы быстрые и надежные компьютеры с достаточным объемом оперативной памяти и процессорами высокой производительности. Это позволяет им выполнять сложные задачи разработки, работать с большими объемами данных и запускать веб-приложения без задержек и зависаний.

Важно иметь доступ к качественному программному обеспечению и инструментам разработки. Веб-разработчики часто используют интегрированные среды разработки (IDE) или текстовые редакторы для написания кода, браузеры для проверки совместимости и отладки веб-приложений, а также другие специализированные программы и библиотеки для работы с базами данных, графикой и т. д.

Эргономика и здоровье.

Организация рабочего места веб-разработчика должна соответствовать принципам эргономики. Это включает правильное расположение клавиатуры и мыши, использование эргономичных аксессуаров, таких как подставка для запястья и вертикальная мышь, а также настройку высоты монитора для оптимального обзора. Правильная эргономика помогает предотвратить перенапряжение мышц и суставов, снижает риск развития повреждений опорно-двигательной системы и повышает общий комфорт во время работы.

Веб-разработчики должны делать регулярные перерывы и проводить упражнения, чтобы предотвратить статическое положение тела и снизить риск развития мускульных и суставных проблем. Это может включать растяжку, глазные упражнения, короткую физическую активность и другие методы релаксации и разминки.

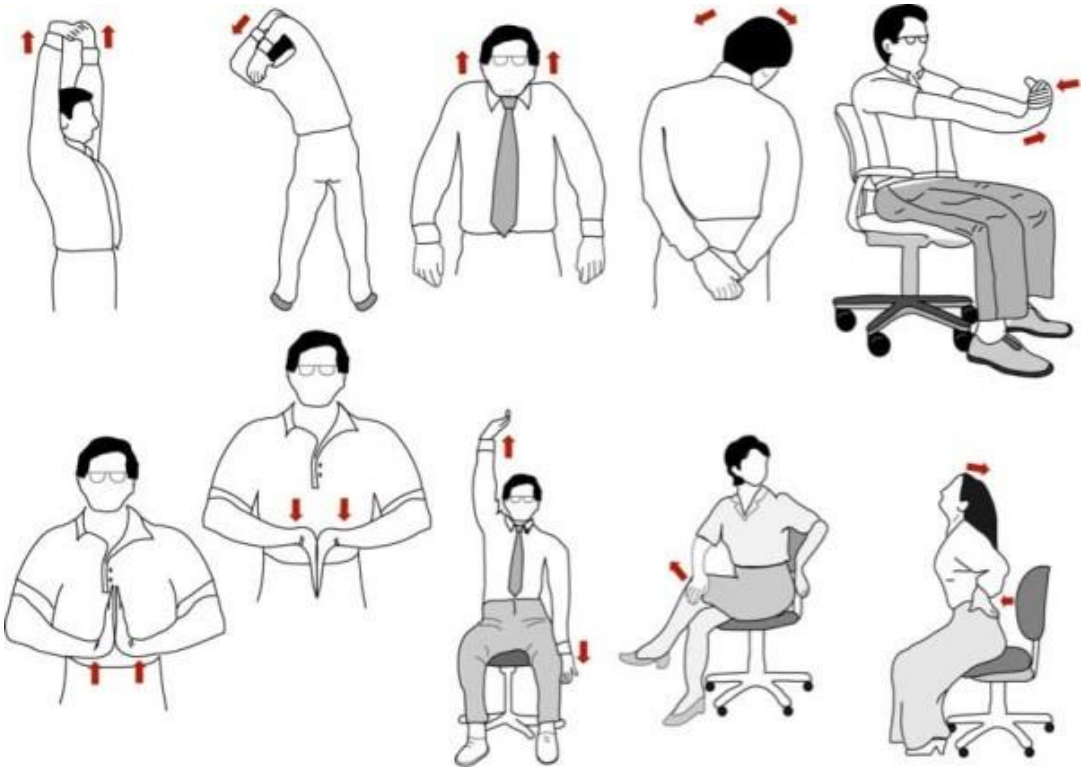


Рисунок 4.1 – Разминка

Рабочее время и гибкость.

Гибкость в рабочем графике позволяет веб-разработчикам оптимально использовать свое время и работать в наиболее продуктивные для себя часы.

Это может включать гибкие начало и окончание рабочего дня, возможность удаленной работы или разгрузку рабочего времени, которая соответствует их индивидуальным потребностям и предпочтениям.

Важно создать баланс между работой и личной жизнью, чтобы веб-разработчики могли отдыхать, восстанавливаться и наслаждаться своими внерабочими интересами. Поддержка гармонии между работой и жизнью помогает предотвращать переутомление, повышает уровень удовлетворенности и способствует долгосрочной продуктивности и эффективности.

Коммуникация и сотрудничество.

Веб-разработчики часто работают в командах и взаимодействуют с другими коллегами и клиентами. Поэтому важно иметь доступ к современным коммуникационным инструментам, таким как электронная почта, мессенджеры, видеоконференции и совместное использование документов. Это позволяет эффективно общаться, обмениваться информацией, задавать вопросы и решать проблемы в режиме реального времени.

Веб-разработчики часто сотрудничают с другими разработчиками, дизайнерами и проектными менеджерами. Создание условий для эффективного сотрудничества, обмена идеями и обратной связи способствует повышению качества работы и результативности команды. Это может включать проведение совещаний, планирование проектов, распределение задач и обеспечение прозрачности в рабочем процессе[1].

Обучение и развитие навыков.

Веб-разработка - динамичная область, требующая постоянного обновления знаний и навыков. Компания должна поддерживать профессиональное развитие веб-разработчиков, предоставляя доступ к обучающим ресурсам, курсам и тренингам. Это позволяет разработчикам совершенствовать свои навыки, изучать новые технологии и быть в курсе последних тенденций и инноваций в веб-разработке.

Обеспечение оптимальных условий труда для веб-разработчиков не только способствует их комфорту и здоровью, но и повышает производительность, качество работы и уровень удовлетворенности. Компании, которые уделяют внимание этим аспектам, создают благоприятную рабочую среду, способствующую росту и развитию веб-разработчиков.

4.2 Экологичность

Подраздел "Экологичность" в работе включает в себя описание влияния деятельности проекта на окружающую среду и анализ организации обращения с отходами в организации. В данном случае рассматривается обращение с различными видами отходов, включая оргтехнику, люминесцентные лампы, офисную мебель, макулатуру и другие.

Обращение с оргтехникой.

Обращение с оргтехникой является важным аспектом обеспечения экологичности деятельности организации, занимающейся разработкой веб-приложения для аренды и продажи недвижимости. Для достижения оптимальных результатов в данной сфере, следует уделить внимание следующим подробностям:

Идентификация устаревшей или вышедшей из строя оргтехники:

– разработка системы отслеживания и инвентаризации оргтехники, включая регистрацию устройств и их характеристик;

– регулярное полное обновление информации об оргтехнике и отметка устройств, которые требуют замены или утилизации, для оперативного контроля.

Ремонт и вторичное использование:

– при возможности, организация ремонтных работ для восстановления функциональности вышедшей из строя оргтехники;

– выделение специального места для хранения оргтехники, ожидающей ремонта или возможного вторичного использования, с ясной маркировкой и системой учета.

Утилизация устаревшей оргтехники:

– в соответствии с современным законодательством, организация сбора и утилизации оргтехники, которая не может быть восстановлена или использована повторно;

– установление партнерства с специализированными компаниями или сервисами, занимающимися утилизацией электронной техники, для обеспечения безопасного и экологически ответственного обращения с отходами.

Закупка экологически безопасной оргтехники:

– при выборе новой оргтехники, предпочтение отдается моделям, которые соответствуют современным экологическим стандартам;

– оценка сертификации продуктов и их соответствия энергоэффективности и экологическим требованиям для сокращения негативного влияния на окружающую среду.

Обучение сотрудников:

– проведение обучающих программ и сессий для сотрудников, направленных на правильное обращение с оргтехникой и повышение осведомленности об экологических проблемах;

– популяризация практик энергосбережения, использования эффективных режимов работы оргтехники и рационального использования ресурсов;

– путем детального внедрения и регулярного контроля этих мер можно обеспечить эффективное обращение с оргтехникой, соблюсти требования современного законодательства и принести пользу окружающей среде.

Обращение с люминесцентными лампами.

Люминесцентные лампы содержат определенное количество ртути и других вредных веществ, поэтому их правильное обращение является важным аспектом экологической ответственности. В рамках организации, занимающейся разработкой веб-приложения для аренды и продажи недвижимости, следует учесть следующие моменты при обращении с люминесцентными лампами:

– установка специальных контейнеров для сбора использованных люминесцентных ламп в рабочих помещениях;

– обеспечение наличия контейнеров в удобных и видимых местах, где сотрудники смогут безопасно сдать использованные лампы;

– маркировка контейнеров яркими этикетками, указывающими на необходимость сбора и переработки люминесцентных ламп;

– проведение обучающих семинаров или вебинаров для сотрудников, посвященные правильному обращению с использованными люминесцентными лампами;

– установка партнерства с организациями или службами, специализирующимися на переработке и утилизации люминесцентных ламп.

– введение учета количества использованных и собранных люминесцентных ламп;

– создание отчетов о деятельности по обращению с лампами, включая информацию о количестве сданных ламп и их утилизации.

Соответствие законодательству:

– ознакомление с местными нормативными актами и законодательством, регулирующим обращение с ртутьсодержащими отходами и люминесцентными лампами;

– процедуры и практики, связанные с обращением с люминесцентными лампами, соответствуют требованиям законодательства.

Мониторинг и улучшение:

– регулярный мониторинг эффективности системы обращения с люминесцентными лампами и оценку соответствия современным экологическим стандартам;

– идентификация областей, требующих улучшений, и разработка плана действий для совершенствования обращения с люминесцентными лампами.

Обращение с офисной мебелью.

Планирование расстановки мебели должно основываться на принципах эргономики, чтобы обеспечить комфортные и безопасные рабочие места.

Размещение мебели таким образом, чтобы использовать пространство офиса максимально эффективно. Во избежание перегруженности или пустот, оптимизация расстановки столов, стульев, шкафов и других элементов мебели.

При выборе офисной мебели следует отдавать предпочтение качественным материалам и прочным конструкциям, которые выдержат интенсивное использование на протяжении длительного времени.

Разработка системы сбора и классификации использованной офисной мебели, которая больше не требуется. Создайте специальные места или контейнеры для сбора мебели перед ее дальнейшей обработкой.

Исследование возможности переработки материалов мебели. Некоторые компоненты мебели, такие как дерево или металл, могут быть переработаны или использованы для создания новых изделий.

Изучение местных нормативных актов и законодательство, которые регулируют обращение с офисной мебелью и отходами. Убедитесь, что ваша организация соответствует всем требованиям и принимает необходимые меры для соблюдения законодательства.

При обращении с мебелью, следует учитывать правила утилизации, переработки и выброса, чтобы предотвратить негативное воздействие на окружающую среду и соблюдать требования по утилизации определенных материалов.

Эффективное обращение с офисной мебелью позволит организации не только соблюсти требования законодательства и минимизировать негативное воздействие на окружающую среду, но и создать экологически ответственный образ деятельности и стимулировать устойчивое развитие[2].

Обращение с макулатурой и другими видами отходов.

Определение различных видов отходов, которые генерируются в офисе, включая макулатуру, пластик, стекло, металлы и другие материалы. Разделение

контейнеров для сбора отходов в соответствии с их типами для облегчения последующей обработки и переработки.

Размещение контейнеров для сбора отходов в удобных и доступных местах офиса, чтобы стимулировать сотрудников к правильной классификации и утилизации отходов. Обеспечение отметок или ярлыков на контейнерах, указывающие виды отходов, которые следует туда складывать.

Следует провести обучение сотрудников по правильной классификации и обращению с отходами и объяснить, какие отходы относятся к каким категориям и как правильно складывать их в соответствующие контейнеры. Распространение информационных материалов или памятки, чтобы сотрудники всегда могли обратиться к ним при необходимости.

Утилизация и переработка отходов:

– установка партнерства с организациями, занимающимися утилизацией и переработкой отходов. Сотрудничество с сертифицированными компаниями или утилизационными центрами, которые могут обеспечить безопасное и экологически ответственное обращение с отходами;

– урганизация регулярного сбора и вывоз отходов для последующей утилизации или переработки. Убедитесь, что сбор отходов осуществляется вовремя и в соответствии с местными требованиями и нормативами.

Сокращение использования отходов: внедрение практики сокращения использования отходов в офисе. Привлечение сотрудников к экономии ресурсов, уменьшению использования бумаги, пластика и других материалов, а также к рециклингу и повторному использованию предметов, где это возможно.

Соответствие законодательству: изучение местных нормативных актов и законодательства, регулирующие обращение с отходами. Организация соблюдает все требования по классификации, утилизации и переработке отходов.

Соответствие современному законодательству.

В результате анализа и разработки плана действий, компания ООО ФЛЭТС сможет обеспечить соответствие современному законодательству в об-

ласти условий труда и обращения с отходами. Это позволит предотвратить нарушения и риски, а также соблюдать требования, установленные законодательством.

4.3 Чрезвычайные ситуации

Данный подраздел посвящен анализу возможных ЧС, которые могут возникнуть в рамках деятельности компании ООО «ФЛЭТС», а также рассмотрению мер, принятых для предупреждения и ликвидации подобных ситуаций. Чрезвычайные ситуации, такие как пожары, аварии электроснабжения или наводнения, могут негативно повлиять на безопасность сотрудников, состояние оборудования и нормальное функционирование организации в целом.

Анализ возможных чрезвычайных ситуаций:

– идентификация потенциальных ЧС, которые могут возникнуть в рабочей среде, включая пожар, аварию электроснабжения, наводнение и другие чрезвычайные ситуации, специфичные для офисного пространства [5];

– оценка вероятности возникновения каждой ЧС и потенциального воздействия на работников, оборудование и деятельность компании.

Меры предупреждения чрезвычайных ситуаций:

– изучение мер предупреждения ЧС, принятых в организации, например, наличие пожарной сигнализации, датчиков дыма и систем автоматического пожаротушения;

– анализ эффективности существующих систем и оценка их соответствия современным нормам и требованиям безопасности.

Меры ликвидации чрезвычайных ситуаций:

– рассмотрение процедур и мероприятий, предусмотренных для ликвидации чрезвычайных ситуаций, включая эвакуацию, вызов спасательных служб и первую помощь;

– оценка наличия необходимого оборудования для ликвидации ЧС, такого как огнетушители, аварийное освещение и аптечки первой помощи, и их соответствие требованиям законодательства.

Обучение персонала:

– изучение программ обучения и тренировок, предоставляемых компанией для обучения персонала действиям в случае возникновения ЧС;

– оценка регулярности и эффективности проведения тренировок, а также вовлеченности сотрудников в подготовку к ЧС.

Планы эвакуации и коммуникации:

– рассмотрение наличия планов эвакуации и механизмов коммуникации в случае ЧС;

– оценка доступности и ясности планов, а также проведение учений для проверки их эффективности.

Сотрудничество с экстренными службами:

– изучение сотрудничества компании с экстренными службами, такими как пожарная охрана, медицинские службы и спасательные службы;

– оценка наличия контактов и процедур для быстрого вызова и координации действий с экстренными службами.

В результате анализа была произведена оценка готовности компании ООО «ФЛЭТС» к предупреждению и ликвидации чрезвычайных ситуаций, включая вероятный пожар в офисе, и оценка соответствия этих мер современному законодательству и требованиям безопасности.

ЗАКЛЮЧЕНИЕ

В заключение данной квалификационной работы были достигнуты поставленные цели, связанные с разработкой веб-приложения для аренды и продажи недвижимости. В ходе исследования предметной области были проведены анализ существующих решений в сфере продажи и аренды недвижимости, что позволило определить основные требования и ожидания пользователей.

На основе собранных требований были сформулированы функциональные требования к разрабатываемому веб-приложению. Проведено проектирование веб-приложения, включая архитектуру и структуру базы данных. Это обеспечило основу для реализации и тестирования приложения.

В результате работы было успешно разработано и протестировано веб-приложение, которое соответствует поставленным требованиям и ожиданиям пользователей. Приложение позволяет пользователям удобно арендовать и продавать недвижимость, предоставляя широкий спектр функциональности и удобный интерфейс.

Данная квалификационная работа имеет практическую значимость, поскольку разработанное веб-приложение может быть использовано для эффективного управления процессами продажи и аренды недвижимости. Результаты работы могут быть использованы как основа для дальнейшего развития и совершенствования веб-приложения, а также для проведения дальнейших исследований в области разработки программного обеспечения для недвижимости.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Арустамов Э.А Безопасность жизнедеятельности: Учебник. М.: " Дашков и К", 2019. 420с.

2 Арустамов Э.А. Природопользование: Учебник. М.: " Дашков и К", 2022. 276с.

3 Ван Россум, Г. "Руководство по языку программирования Python." [Электронный ресурс]. Доступно по: https://python.org. [Дата доступа: 07.06.2023].

4 Веллинг, Л., Томсон, Л. "Разработка веб-приложений с использованием PHP и MySQL." Москва: ДМК Пресс, 2021. 560 с.

5 Гринин А.С., Новиков В.Н., Экологическая безопасность. Защита территорий и населения при чрезвычайных ситуациях: Учеб. пособие. М.: ФАИР- ПРЕС, 2018. 336 с.

6 ДуБуа, П. "MySQL." Санкт-Петербург: Питер, 2020. 432 с.

7 Дукетт, Дж. "HTML и CSS. Разработка и дизайн веб-сайтов." Санкт-Петербург: Питер, 2019. 512 с.

8 Дюрюмик, З., и др. "Жизненный цикл X.509-сертификатов, зашифрованных с использованием алгоритма MD5." [Электронный ресурс]. Доступно по: https://acm.org. [Дата доступа: 07.06.2023].

9 Ламан, Ш., и др. "Архитектура веб-приложений: принципы, протоколы и практика." Москва: ДМК Пресс, 2021. 432 с.

10 Моррисон, М., и др. "Профессиональная разработка веб-сайтов на HTML5 с использованием CSS, JavaScript и мультимедиа." Санкт-Петербург: БХВ-Петербург, 2018. 768 с.

11 Оуэнс, Р. "Форензика SQLite." Москва: ДМК Пресс, 2019. 352 с.

12 Флэнаган, Д. "JavaScript. Подробное руководство." Москва: Вильямс, 2021. 1248 с.

13 Фритчи, Г. "Планы выполнения SQL Server." [Электронный ресурс]. Доступно по: https://red-gate.com. [Дата доступа: 07.06.2023].

14 Чаффер, Дж., Сведберг, К. "Изучаем jQuery." Москва: ДМК Пресс, 2018. 500 с.

15 Эмблер, С., Садаладж, П. "Рефакторинг баз данных: эволюционное проектирование баз данных." Санкт-Петербург: БХВ-Петербург, 2020. 432 с.

ПРИЛОЖЕНИЕ А

```
CREATE TABLE Role (
    ID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL
);
CREATE TABLE User (
    ID INT PRIMARY KEY,
    Login VARCHAR(50) NOT NULL,
    Password VARCHAR(50) NOT NULL,
    Role INT,
    FOREIGN KEY (Role) REFERENCES Role(ID)
);
CREATE TABLE Client (
    ID INT PRIMARY KEY,
    PassportSeries VARCHAR(10) NOT NULL,
    PassportNumber VARCHAR(20) NOT NULL,
    Address VARCHAR(100) NOT NULL,
    UserID INT,
    FOREIGN KEY (UserID) REFERENCES User(ID)
);
CREATE TABLE Property (
    ID INT PRIMARY KEY,
    Area DECIMAL(10,2) NOT NULL,
    RentalPrice DECIMAL(10,2) NOT NULL,
    Location VARCHAR(100) NOT NULL,
    Description VARCHAR(100) NOT NULL
);
CREATE TABLE Contract (
    ID INT PRIMARY KEY,
    ClientID INT,
    PropertyID INT,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    FOREIGN KEY (ClientID) REFERENCES Client(ID),
    FOREIGN KEY (PropertyID) REFERENCES Property(ID)
);
CREATE TABLE PaymentReceipt (
    ID INT PRIMARY KEY,
    ContractID INT,
    Amount DECIMAL(10,2) NOT NULL,
    DateTime DATE NOT NULL,
    FOREIGN KEY (ContractID) REFERENCES Contract(ID)
)
```