МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Кафедра компьютерных образовательных технологий

КУРСОВАЯ РАБОТА

Тема: Разработка системы оценки стоимости аренды квартир

Работу выполнил Ян Цзяфэн **группы** P34212
(фамилия, имя, отчество) (номер группы)

Руководитель Штенников Дмитрий Геннадьевич
(фамилия, имя, отчество)

Работа защищена "_____" _____ 2022 г.

с оценкой _____

Подписи членов комиссии: _____

_____

САНКТ-ПЕТЕРБУРГ, 2012

СОДЕРЖАНИЕ

ТЗ по ГОСТ 19

1 ВВЕДЕНИЕ

1.1 Наименование программы

«Система оценки стоимости аренды квартир в Санкт-Петербурге».

Данная система предназначена для научной оценки стоимости аренды квартир в Санкт-Петербурге с помощью машинного обучения.

1.2 Область применения

Областью применения программного продукта является аренда квартир в Санкт-Петербурге.

1.3 Объект, в котором используют программу

Арендодатель и арендатор квартир.

2 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1 Документы, на основании которых ведется проектирование

Учебный план университета ИТМО для кафедры компьютерных

образовательных технологий

2.2 Организация, утвердившая документ, и дата утверждения

НИУ ИТМО

01.09.2021

2.3 Шифр темы

Наименование темы разработки – «Разработка системы оценки стоимости аренды квартир в Санкт-Петербурге».

Условное обозначение темы разработки (шифр темы) – «СОС-АК-СПБ».

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Программа будет использоваться двумя группами пользователей: арендодатель и арендатор квартир.

Для арендодателя квартиры программа предоставляет возможность прогноза разумной стоимости аренды своей квартиры и поста информации об аренде своей квартиры.

Для арендатора квартиры программа позволяет просматривать текущую информацию об аренде из сторонних сайтов и сравнивать их цены с ценой, прогнозируемыми системой.

4 ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1 Требования к функциональным характеристикам

Данные о пользователях и аренде квартиры хранятся в базе данных. СУБД управляет классификацией базы данных и доступом к данным – даёт гостю права на чтение, пользователю – на чтение и запись, а администратору - права на редактирование и управление.

4.1.1 Требования к составу выполняемых функций

Данная программа должна выполнять следующие функции:

- Регистрация и логин пользователей
- Отображение информация об аренде из сторонних сайтов
- Предоставление интерфейса для ввода данных о характеристиках квартиры
- Предоставление интерфейса для поста информации об аренде своей квартиры
- Управление историей прогноза стоимости
- Предоставление API для анализа данных и прогноза стоимости

4.2 Требования к надежности

- Обеспечение защиты личных данных пользователей от несанкционированного доступа.
- Безотказной работы системы при условии исправности сети.

4.3 Условия эксплуатации

Программу необходимо запускать при температуре и влажности окружающей среды, при которых компьютер и сеть могут нормально работать. Программа не требует проведения каких-либо видов обслуживания. И к арендатору специальные требования не предъявляются.

4.4 Требования к составу и параметрам технических средств

Компьютер пользователя, включающий в себя:

- процессор x86 с тактовой частотой, не менее 1 ГГц;
- оперативную память объемом, не менее 1 Гб;
- видеокарту, монитор, мышь, клавиатура.

4.5 Требования к информационной и программной совместимости

Модель прогноза обновится один раз в месяце.

Программы между пользователем обмениваются с СУБД сообщениями при протоколе HTTP.

4.6 Требования к маркировке и упаковке

Требования не предъявляются.

4.7 Требования к транспортированию и хранению

Требования не предъявляются.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

- Инструкция программы
- Описание функций каждого API
- Описание прав пользователей
- Техническое задание

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Ориентировочная экономическая эффективность не рассчитываются. Предполагаемая годовая потребность продукта – 5000 раз в год. Зарубежных и отечественных аналогов – Airbnb, Авито.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

- Техническое задание
- Технический проект
- Дизайн интерфейса взаимодействия пользователей
- Выбор и тестирование модели прогноза
- Тестирование программы

8 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1 Виды испытаний

- Тестирование прав доступа неавторизованных пользователей
- Тестирование точности и отказоустойчивости модели прогноза
- Тестирование работоспособности программы при использовании многими пользователями.

8.2 Общие требования к приемке работы

Неавторизованные пользователи не могут просмотреть личные данные арендодателя. Точность и отказоустойчивость модели прогноза выше, чем 80%. Программа может нормально работать, когда 500 пользователей используют программу одновременно.

ПРИЛОЖЕНИЕ 1

Разработка ТЗ по ГОСТ 19.

http://rugost.com/index.php?option=com_content&view=article&id=105:19-1-3&catid=25&Itemid=62

http://rugost.com/index.php?option=com_content&view=article&id=106:19-4-8&catid=25&Itemid=62

ТЗ по ГОСТ 34

1 ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: Система оценки стоимости аренды квартир в Санкт-Петербурге.

Краткое наименование системы: СОСаренды.

1.2 Наименования организации-заказчика и организаций-участников работ

Заказчиком системы является Штенников Дмитрий Геннадьевич.

Разработчиком системы является студент группы Р34212 Ян Цзяфэн.

1.3 Перечень документов, на основании которых создается система

Учёный план о курсовой работе 4 курса.

1.4 Плановые сроки начала и окончания работы по созданию системы

Плановый срок начала работ по созданию системы - сентябрь 2021.

Плановый срок окончания работ по созданию системы - январь 2022.

1.5 Источники и порядок финансирования работ

Финансирование отсутствует.

1.6 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передается в виде функционирующего комплекса на базе средств вычислительной техники Заказчика и Исполнителя в сроки, установленные Госконтрактом. Приемка системы осуществляется комиссией в составе уполномоченных представителей Заказчика и Исполнителя.

Порядок предъявления системы, ее испытаний и окончательной приемки определен в п.6 настоящего ЧТЗ. Совместно с предъявлением системы производится сдача разработанного Исполнителем комплекта документации согласно п.8 настоящего ЧТЗ.

1.7 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

ГОСТ 34.201-89, ГОСТ 34.602-89, ГОСТ 34.601-90, ГОСТ 34.603-92, ГОСТ 7.32-2001.

1.8 Определения, обозначения и сокращения

ИС – Информационная система.

БД – База данных.

Программа – Система оценки стоимости аренды квартир в Санкт-Петербурге.

СУБД – система управления базами данных.

2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

2.1 Назначение системы

Программа будет использоваться двумя группами пользователей: арендодатель и арендатор квартир. Для арендодателя квартиры программа предоставляет возможность прогноза разумной стоимости аренды своей квартиры и поста информации об аренде

своей квартиры. Для арендатора квартиры программа позволяет просматривать текущую информацию об аренде из сторонних сайтов и сравнивать их цены с ценой, прогнозируемыми системой.

2.2 Цели создания системы

Основными целями создания программы являются:

- Повышение эффективности аренды.
- Создание максимально прозрачной платформы аренды.
- Рационализация цены на аренду.

3 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

3.1 Краткие сведения об объекте автоматизации

Объектом автоматизации является процесс агрегация информации аренды.

3.2 Сведения об условиях эксплуатации приложения

Программу необходимо запускать при температуре и влажности окружающей среды, при которых компьютер и сеть могут нормально работать. Программа не требует проведения каких-либо видов обслуживания. И к арендатору специальные требования не предъявляются.

4 ТРЕБОВАНИЯ К СИСТЕМЕ

4.1 Требования к системе в целом

4.1.1 Требования к структуре и функционированию системы

Система должна представлять собой систему, включающую в себя модуля:

- Модуль оценки
- Модуль предоставления информации

4.1.1.1 Перечень подсистем, их назначение и основные характеристики

- Подсистем оценки:

  Данная подсистема предназначена для оценки стоимости аренды квартиры с помощью моделей Машиного Обучения и предоставляет интерфейс для ввода данных о характеристиках квартиры.
- Подсистем предоставления информации:

  Данная подсистема предназначена для предоставления информации об аренде квартир из сторонних сайтов.

4.1.1.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Система должна быть подключена к Интернету. Модуля системы должны иметь доступ к БД системы и могут сохранить и читать данные.

4.1.1.3 Требования к характеристикам взаимосвязей создаваемой системы со смежными системами

Требования не предъявляются.

4.1.1.4 Требования к режимам функционирования системы

Требования не предъявляются.

4.1.1.5 Требования по диагностированию системы

Требования не предъявляются.

4.1.1.6 Перспективы развития, модернизации системы

Улучшения качества оценки системы.

4.1.2 Требования к численности и квалификации персонала системы

Требования не предъявляются.

4.1.3 Требования к надежности

- Обеспечение защиты личных данных пользователей от несанкционированного доступа.
- Безотказной работы системы при условии исправности сети.

4.1.4 Требования к безопасности

Требования не предъявляются.

4.1.5 Требования к эргономике и технической эстетике

Интерфейс системы должен быть понятным и удобным.

4.1.6 Требования к транспортабельности для подвижных АС

Требования не предъявляются.

4.1.7 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Требования не предъявляются.

4.1.8 Требования к защите информации от несанкционированного доступа

Обеспечение защиты личных данных пользователей от несанкционированного доступа.

4.1.9 Требования по сохранности информации при авариях

Требования не предъявляются.

4.1.10 Требования к защите от влияния внешних воздействий

Требования не предъявляются.

4.1.11 Требования к патентной чистоте

Требования не предъявляются.

4.1.12 Требования по стандартизации и унификации

Требования не предъявляются.

4.2 Требования к функциям (задачам), выполняемым системой

- Регистрация и логин пользователей
- Отображение информация об аренде из сторонних сайтов
- Предоставление интерфейса для ввода данных о характеристиках квартиры
- Предоставление интерфейса для поста информации об аренде своей квартиры
- Управление историей прогноза стоимости
- Предоставление API для анализа данных и прогноза стоимости

4.3 Требования к видам обеспечения

Компьютер пользователя, включающий в себя:

- процессор x86 с тактовой частотой, не менее 1 ГГц;
- оперативную память объемом, не менее 1 Гб;
- видеокарту, монитор, мышь, клавиатура.

5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ (РАЗВИТИЮ) СИСТЕМЫ

Таблица 1.1 – Календарный план работ по созданию системы

Наименование этапа создания системы	Сроки выполнения работ
Определение темы	01.09.2021
Проектирование	30.09.2021
Сбор данных	15.10.2021
Реализация	10.01.2021
Тестирование системы	18.01.2022
Оформление отчёта и защита	20.01.2022

6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

6.1 Виды, состав, объем и методы испытаний системы

Виды, состав, объем, и методы испытаний подсистемы должны быть изложены в программе и методике испытаний системы, разрабатываемой в составе рабочей документации.

6.2 Общие требования к приемке работ по стадиям

Установить контроль и приемку результатов работ на каждой стадии создания системы в соответствии с разделом 5.

6.3 Статус приемочной комиссии

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ

Требования не предъявляются.

8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

Проектная документация должна быть разработана в соответствии с ГОСТ 34.201-89 и ГОСТ ЕСПД. Отчетные материалы должны включать в себя текстовые материалы (в электронном виде в формате PDF) и графические материалы.

9 ИСТОЧНИКИ РАЗРАБОТКИ

ПРИЛОЖЕНИЕ А

Диаграмма BPMN

В рисунке 3.1 описана диаграмма BPMN этой системы.

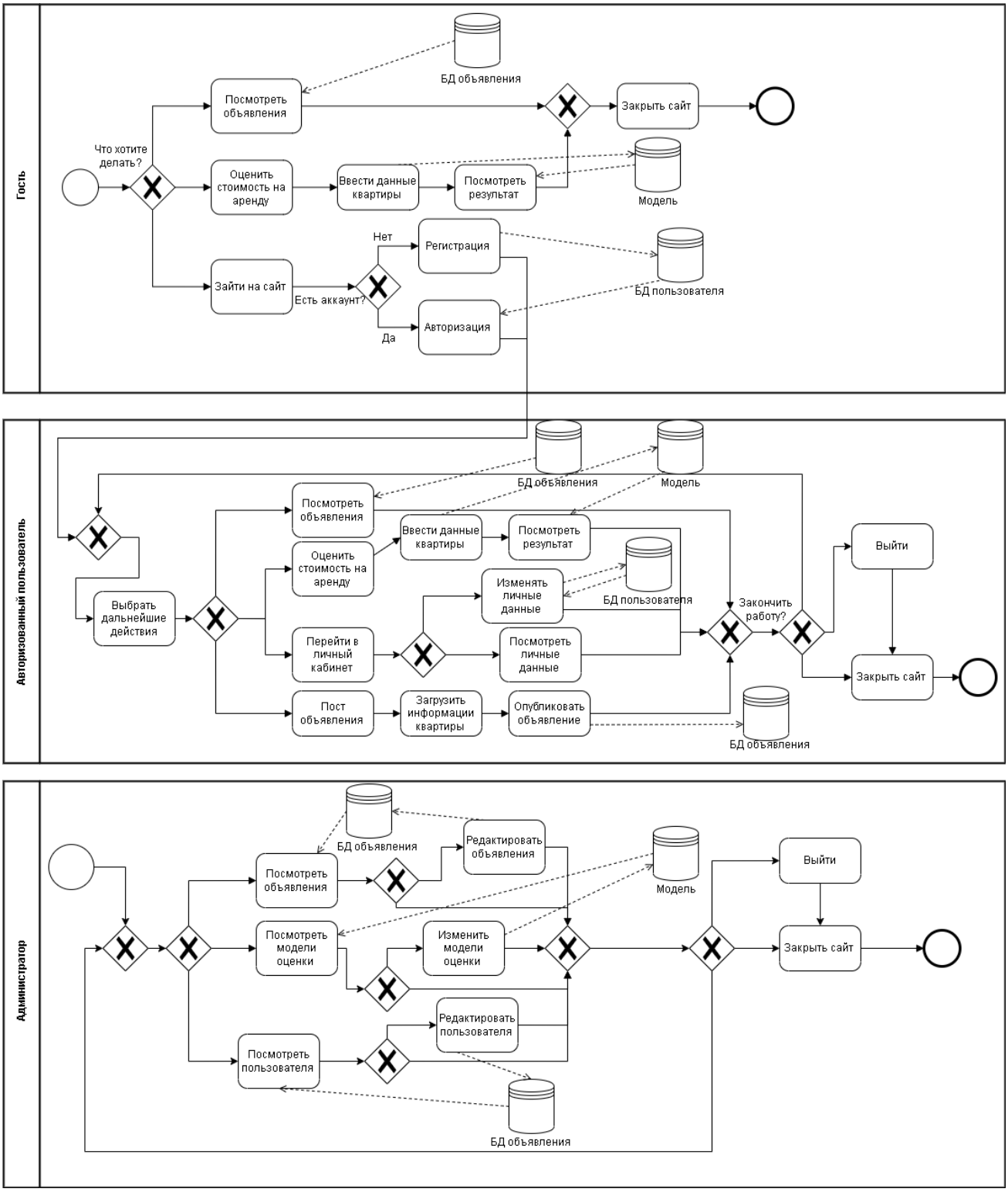
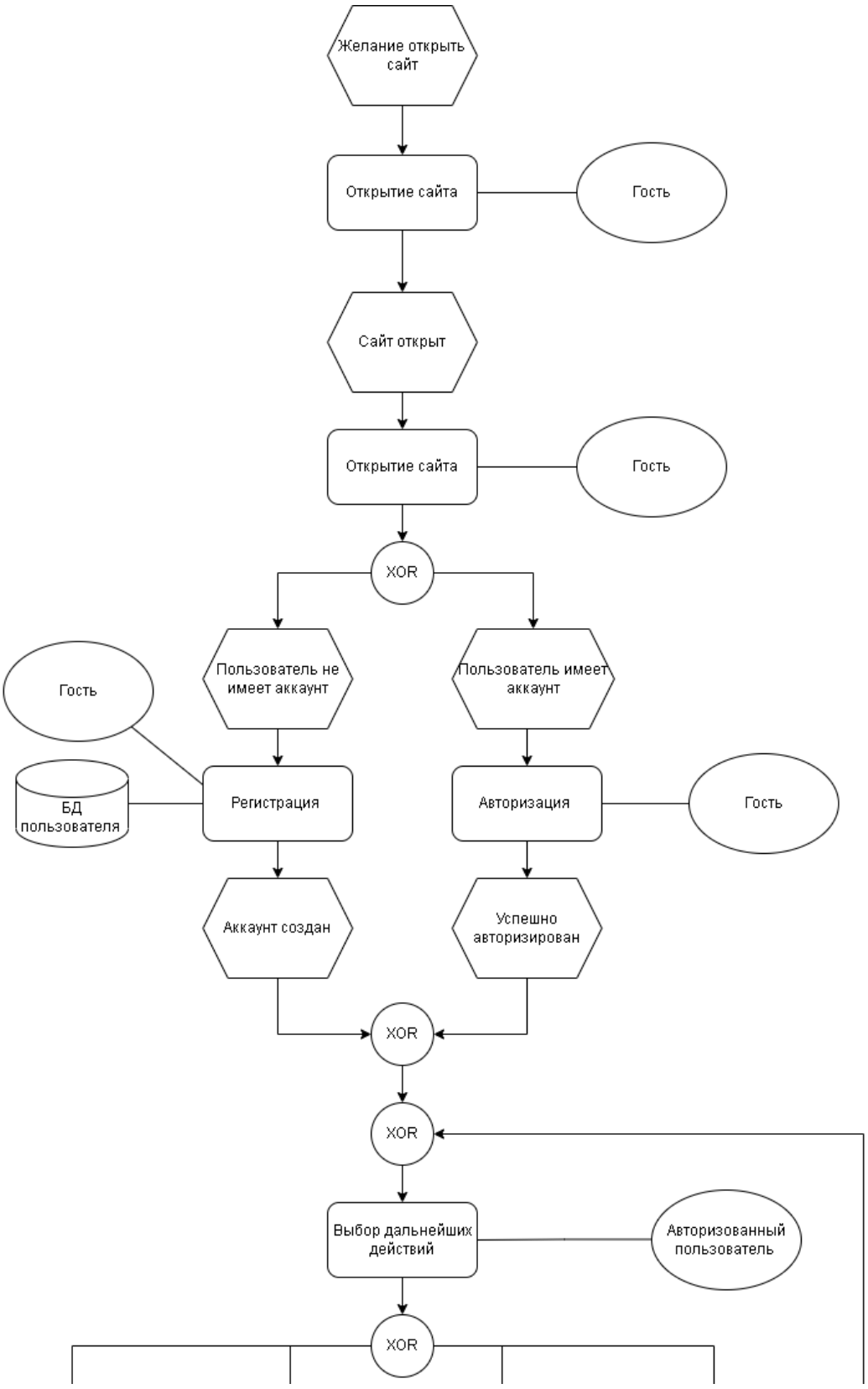


рис. 3.1 Диаграмма BPMN

Диаграмма EPC

В рисунке 4.1 описана диаграмма BPMN этой системы.



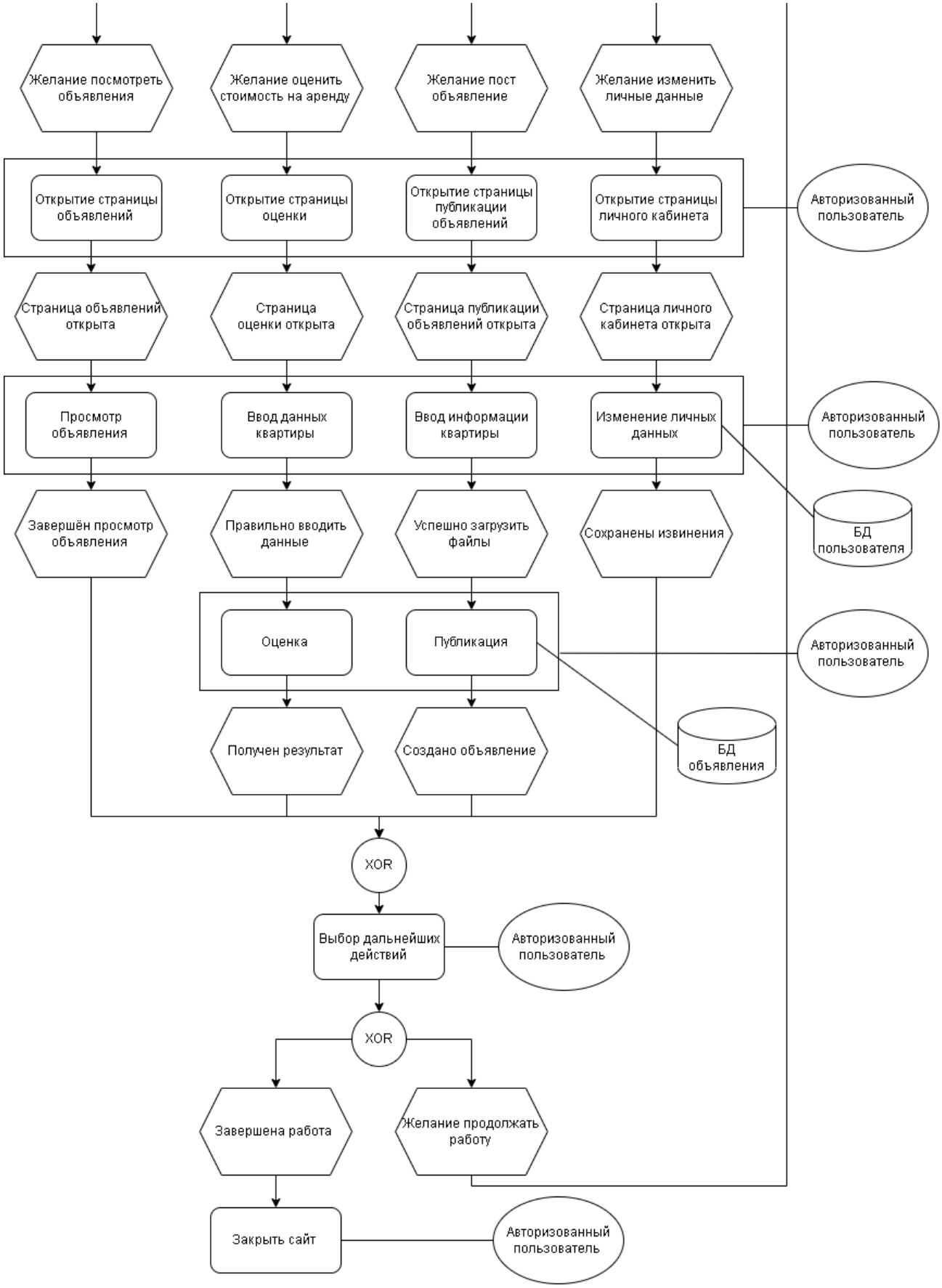


рис. 4.1 Диаграмма EPC

Конструирование

ВВЕДЕНИЕ

Проект представляет собой система оценки стоимости аренды квартир в Санкт-Петербурге, основанный на языке Python и фреймворке Django. Он использует базу данных Mysql для хранения информации, такой как пользователи и объявления, и базу данных Redis для хранения информации о сеансах. Это веб-приложение реализует функции регистрации, входа в систему, изменения паролей, изменения личной информации пользователя, публикации объявления, оценка стоимости, агрегации объявления со сторонних сайтов (https://spb.cian.ru/), рекомендации объявления и пейджинга.

Django - это фреймворк для веб-приложений с открытым исходным кодом, написанный на Python. Используя Django, вы можете легко заполнить большую часть контента, необходимого для формального веб-сайта, с помощью очень небольшого количества кода и в дальнейшем разработать полнофункциональную веб-службу. Сам Django основан на модели MVC, а именно Model + View + Controller режим разработки. Режим MVC упрощает последующую модификацию и расширение программы и дает возможность повторно использовать определенную часть программы.

MySQL - самая популярная система управления реляционными базами данных. С точки зрения веб-приложений MySQL - одна из лучших прикладных программ СУБД (система управления реляционными базами данных).

Redis является полностью открытым исходным кодом, соответствует протоколу BSD и представляет собой высокопроизводительную базу данных «ключ-значение». Redis поддерживает сохранение данных.Данные в памяти могут быть сохранены на диске и могут быть загружены снова для использования при перезапуске. Redis не только поддерживает простые данные типа «ключ-значение», но также предоставляет хранилище для списков, наборов, zset, хешей и других структур данных.

В части моделирования системы веб-приложений используется унифицированный язык моделирования UML. UML состоит из нескольких частей, таких как представления, диаграммы, элементы модели и общие механизмы. Он может объяснять, визуализировать и документировать продукты объектно-ориентированной системы разработки.

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Разработка информационной системы начинается с анализа функциональных требований. В этом процессе устанавливается модель прецедентов использования, которая описывает динамическое поведение системы моделирования с точки зрения взаимодействия между внешними агентами и системой и описывает отношения между пользователями, требованиями и функциональными единицами системы. Диаграмма прецедентов использования состоит из субъектов, прецедентов использования и отношений между ними. Ниже приведена диаграмма прецедентов использования для разработки этой системы оценки стоимости аренды квартир.

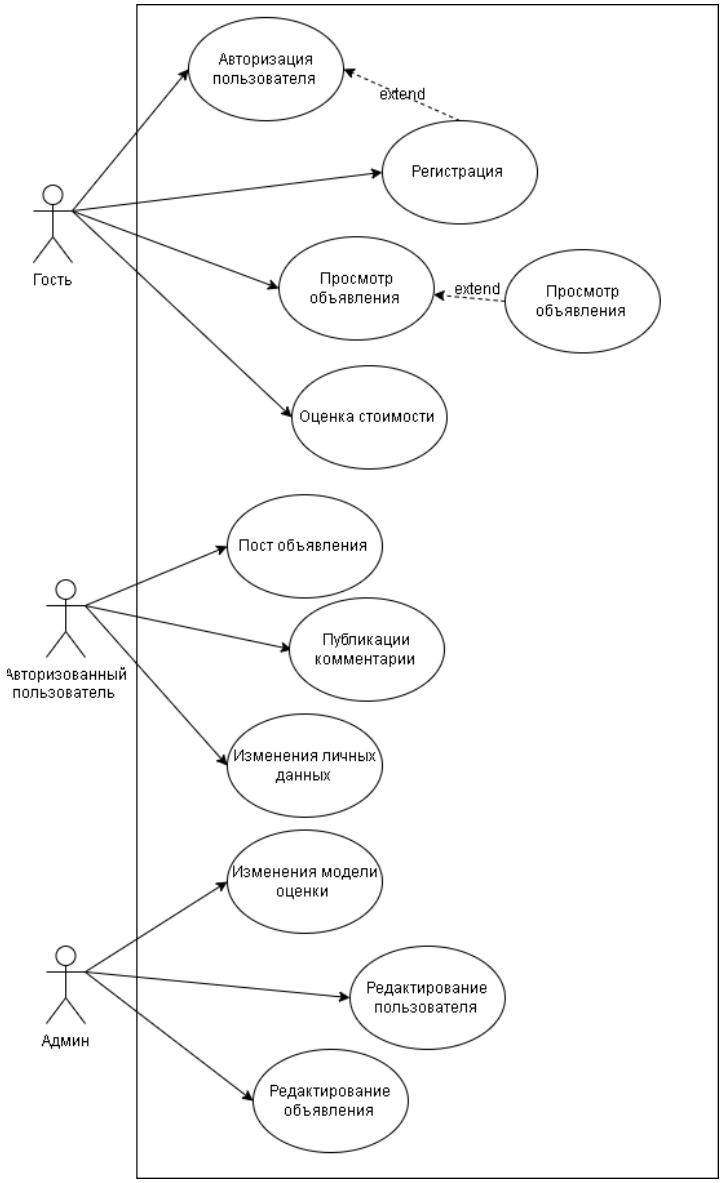


Рисунок 1.1 – Диаграмма прецедентов использования

Участники этой модели прецедентов использования делятся на гостей, авторизованных пользователей и администраторов.

- Гости — это неавторизованные пользователи, которые имеют право регистрироваться и входить в систему, просматривать объявления и комментарии, а также оценить стоимость квартиры.

- Авторизованные пользователи имеют право редактировать личную информацию, публиковать объявления и комментарии.

- Администратор является авторизованным пользователем и имеет право создавать и редактировать пользователей, редактировать категории объявления и редактировать модели оценки.

Следует отметить, что вариант использования Регистрации расширяет вариант использования Авторизации. Основная цель варианта использования Редактирования

информации пользователя и объявления - помочь администратору просматривать информацию о пользователях и объявлений. Администратор обычно изменяет информацию о категории объявлений.

Далее в таблицах 1.1-1.5 будут описаны данные прецеденты более детально.

Прецедент использования	**Регистрация пользователя**	**Авторизация пользователя**
Краткое описание	Этот вариант использования позволяет гостю зарегистрировать новую учетную запись в системе.	Этот вариант использования позволяет гостю войти в свою учетную запись в системе.
Действующие лица	Гость	Гость
Предусловия	Действующее лицо Гость желает зарегистрировать новую учетную запись.	Действующее лицо Гость желает войти в свою учетную запись.
Основной поток	Прецедент использования начинается с того, что Гость решает зарегистрировать новую учетную запись и заполняет соответствующую регистрационную форму, обязательно указав имя учетной записи и пароль, а также код подтверждения, после чего отправляет заявку на регистрацию. Прецедент использования завершается.	Прецедент использования начинается с того, что Гость решает войти в свою учетную запись и заполнить соответствующую форму входа, убедившись, что учетная запись и пароль верны, а затем отправляет заявку на вход. Прецедент использования заканчивается.
Альтернативные потоки	Действующее лицо Гость неверно заполнило регистрационную форму или учетная запись с указанным именем уже существует. Заявка на регистрацию не создается, действующее лицо Гость получает сообщение об ошибке и должно либо повторно заполнить форму, продолжая прецедент использования, либо отменить регистрацию, завершая прецедент использования.	Действующее лицо Гость неверно заполнило свою учетную запись или пароль. Заявка на вход не создается, действующее лицо Гость получает сообщение об ошибке и должно либо повторно заполнить форму, продолжая прецедент использования, либо отменить вход, завершая прецедент использования.
Постусловия	Если прецедент использования завершился успешно, в системе регистрируется новая учетная запись, в противном случае состояние системы остается неизменным	Если прецедент использования завершился успешно, Гость войдет в систему как учетная запись, в противном случае состояние системы остается неизменным

Таблица 1.1 Описание прецедента Регистрация и Авторизация.

Прецедент использования	Просмотр объявления и комментариев	Публикации комментариев
Краткое описание	Этот вариант использования позволяет Актёру просматривать объявления и комментарии в системе.	Этот вариант использования позволяет Пользователю оставить свои комментарии в системе.
Действующие лица	Гость, Пользователь	Пользователь
Предусловия	Актер желает читать объявления или комментарии.	Действующее лицо Пользователь желает оставить свои комментарии.
Основной поток	Прецедент использования начинается с того, что Актер нажимает на одно из названий на странице. Система перенаправляет пользователя на страницу данной объявления. Прецедент использования завершается.	Прецедент использования начинается с того, что Пользователь решает оставить свои комментарии и заполнить свои комментарии в области комментариев, а затем отправляет заявку. Прецедент использования заканчивается.
Постусловия	Если прецедент использования завершился успешно, Актер перенаправлен на страницу объявления.	Если прецедент использования завершился успешно, обновится страница и отобразят новые комментарии

Таблица 1.2 Описание прецедента Просмотра объявления и комментариев и Публикации комментариев.

Прецедент использования	**Публикации объявления**	**Изменения личной информации пользователя**
Краткое описание	Этот вариант использования позволяет Пользователю публиковать объявления в системе.	Этот вариант использования позволяет Пользователю редактировать личную информацию в системе.
Действующие лица	Пользователь	Пользователь
Предусловия	Действующее лицо Пользователь желает публиковать объявления.	Действующее лицо Пользователь желает редактировать личную информацию.
Основной поток	Прецедент использования начинается с того, что Пользователь решает публиковать объявления и заполняет соответствующую форму, обязательно указав заголовок и фото квартиры, а также описания, после чего отправляет заявку. Прецедент использования завершается.	Прецедент использования начинается с того, что Пользователь решает редактировать личную информацию и заполнить соответствующую форму, а затем отправляет заявку. Прецедент использования заканчивается.
Альтернативные потоки	Действующее лицо Пользователь неверно заполнило форму или объявление уже существует. Заявка на публикацию объявления не создается, действующее лицо Пользователь получает сообщение об ошибке и должно либо повторно заполнить форму, продолжая прецедент использования, либо отменить публикацию, завершая прецедент использования.	Действующее лицо Пользователь неверно заполнило форму. Заявка на редактирование не создается, Пользователь получает сообщение об ошибке и должно либо повторно заполнить форму, продолжая прецедент использования, либо отменить редактирование, завершая прецедент использования.
Постусловия	Если прецедент использования завершился успешно, в системе публикуется новое объявление, в противном случае состояние системы остается неизменным	Если прецедент использования завершился успешно, обновится страница и отобразит новая информация, в противном случае состояние системы остается неизменным

Таблица 1.3 Описание прецедента Публикации объявления и Изменения личной информации пользователя.

Прецедент использования	Редактирование информации пользователя	Редактирование информации объявления	Редактирование категорий объявления
Краткое описание	Этот вариант использования позволяет Администратору редактировать информацию пользователя в системе.	Этот вариант использования позволяет Администратору редактировать информацию объявления в системе.	Этот вариант использования позволяет Администратору редактировать категории объявления в системе.
Действующие лица	Администратор	Администратор	Администратор
Предусловия	Действующее лицо Администратор желает редактировать информацию пользователя.	Действующее лицо Администратор желает редактировать информацию объявления.	Действующее лицо Администратор желает редактировать категории объявления.
Основной поток	Прецедент использования начинается с того, что Администратор решает редактировать информацию пользователя и входит в раздел пользователи на странице администратора, вносит изменения в поля данных пользователя, после чего нажимает кнопку сохранить. Прецедент использования завершается.	Прецедент использования начинается с того, что Администратор решает редактировать информацию объявления и входит в раздел пользователи на странице администратора, вносит изменения в поля данных объявления, после чего нажимает кнопку сохранить. Прецедент использования завершается.	Прецедент использования начинается с того, что Администратор решает редактировать категории объявления и входит в раздел пользователи на странице администратора, вносит изменения в поля данных категорий объявления, после чего нажимает кнопку сохранить. Прецедент использования завершается.
Постусловия	Информация пользователя изменена.	Информация данного объявления изменена.	Информация категорий объявления изменена.

Таблица 1.4 Описание прецедента Редактирования информации пользователя, объявления и категорий.

Прецедент использования	Оценка стоимости	Изменения модели оценки
Краткое описание	Этот вариант использования позволяет Актёру оценить стоимость квартира в системе.	Этот вариант использования позволяет Администратору редактировать параметры модели оценки в системе.
Действующие лица	Гость, Пользователь	Администратор
Предусловия	Действующее лицо Актёр желает оценить стоимость квартира.	Действующее лицо Администратор желает редактировать параметры модели оценки.
Основной поток	Прецедент использования начинается с того, что Актёр решает оценить стоимость квартира и заполняет соответствующую форму, обязательно указав количество комнат, площадь квартиры и метро около квартиры, после чего отправляет заявку. Прецедент использования завершается.	Прецедент использования начинается с того, что Администратор решает редактировать параметры модели оценки и заполнить соответствующую форму, после чего нажимает кнопку сохранить. Прецедент использования заканчивается.
Постусловия	Если прецедент использования завершился успешно, в системе появится результат.	Параметры модели оценки изменены.

Таблица 1.5 Описание прецедента Оценки стоимости и Изменения модели оценки пользователя.

ДИАГРАММЫ СОСТОЯНИЙ МОДЕЛИ

Диаграммы состояний модели в основном используются для описания различных состояний объекта, процесса перехода между состояниями, а также различных событий и условий, запускающих переходы между состояниями. Веб-страница изображается в виде прямоугольника с закругленными углами, внутри которого указывается название. Переходы на диаграмме деятельности снабжаются текстовыми метками следующего формата: eventName [guardCondition] / actions, где eventName – имя события, которое может вызвать переход; guardCondition – сторожевое условие, соблюдение которого требуется для запуска перехода; actions – список действий, которые выполняются при запуске перехода.

Диаграмма состояний, показанная на рисунке ниже, описывает возможные действия Гости в этой системе. Его конечным состоянием является авторизация пользователя. Эта диаграмма состояний в основном рассматривает состояние регистрации и состояние авторизации, оба из которых должны проверять достоверность данных.

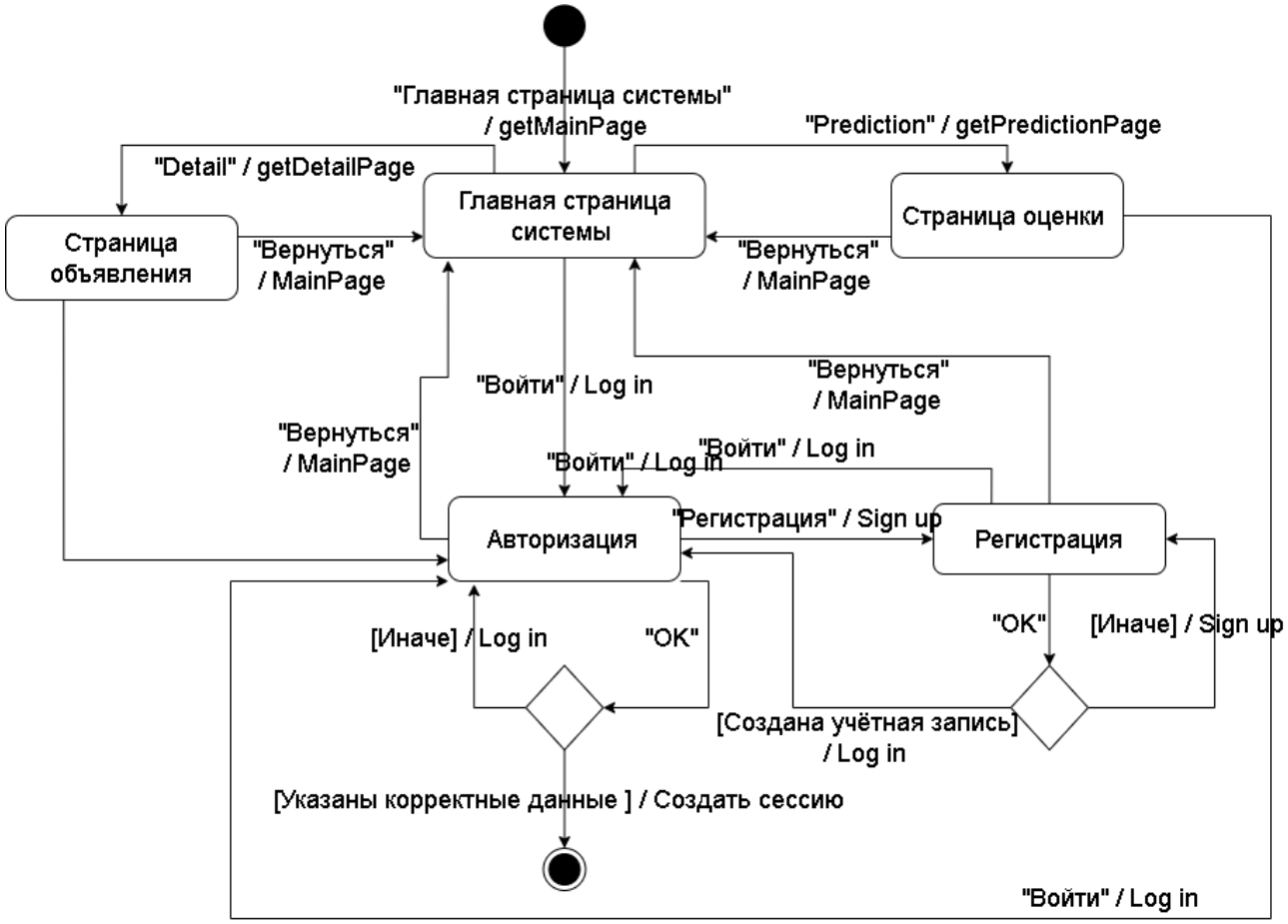


Рисунок 2.1 – Диаграмма состояний модели. (Гость)

На рисунке 2.2 изображена диаграмма состояний авторизованных пользователей, показывающая переходы различных страниц, и конечным состоянием является выход из системы.

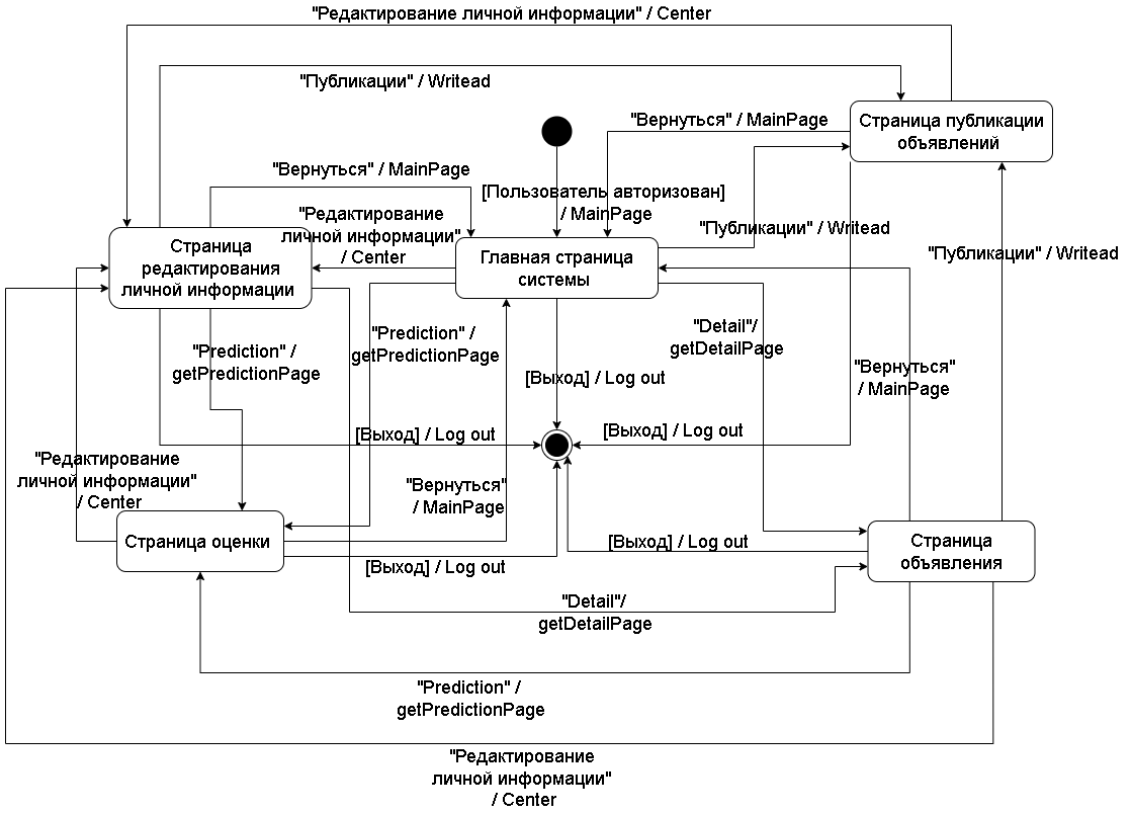


Рисунок 2.2 – Диаграмма состояний модели. (Пользователь)

ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ СУЩНОСТЕЙ

Django предоставляет унифицированный API вызовов для различных баз данных, и в этом проекте используется база данных MySQL. Модель Django использует собственный ORM для преобразования кода Python в операторы SQL. Операторы SQL отправляются на сервер базы данных через pymysql, где операторы SQL выполняются в базе данных и возвращаются результаты.

На следующем рисунке представлена даталогическая модель сущностей, которые необходимо создать в этой системе. Вы можете понять, какие атрибуты должны иметь созданные сущности(Пользователь, Объявление, Комментарии и Категории объявлений), а также связи между ними.

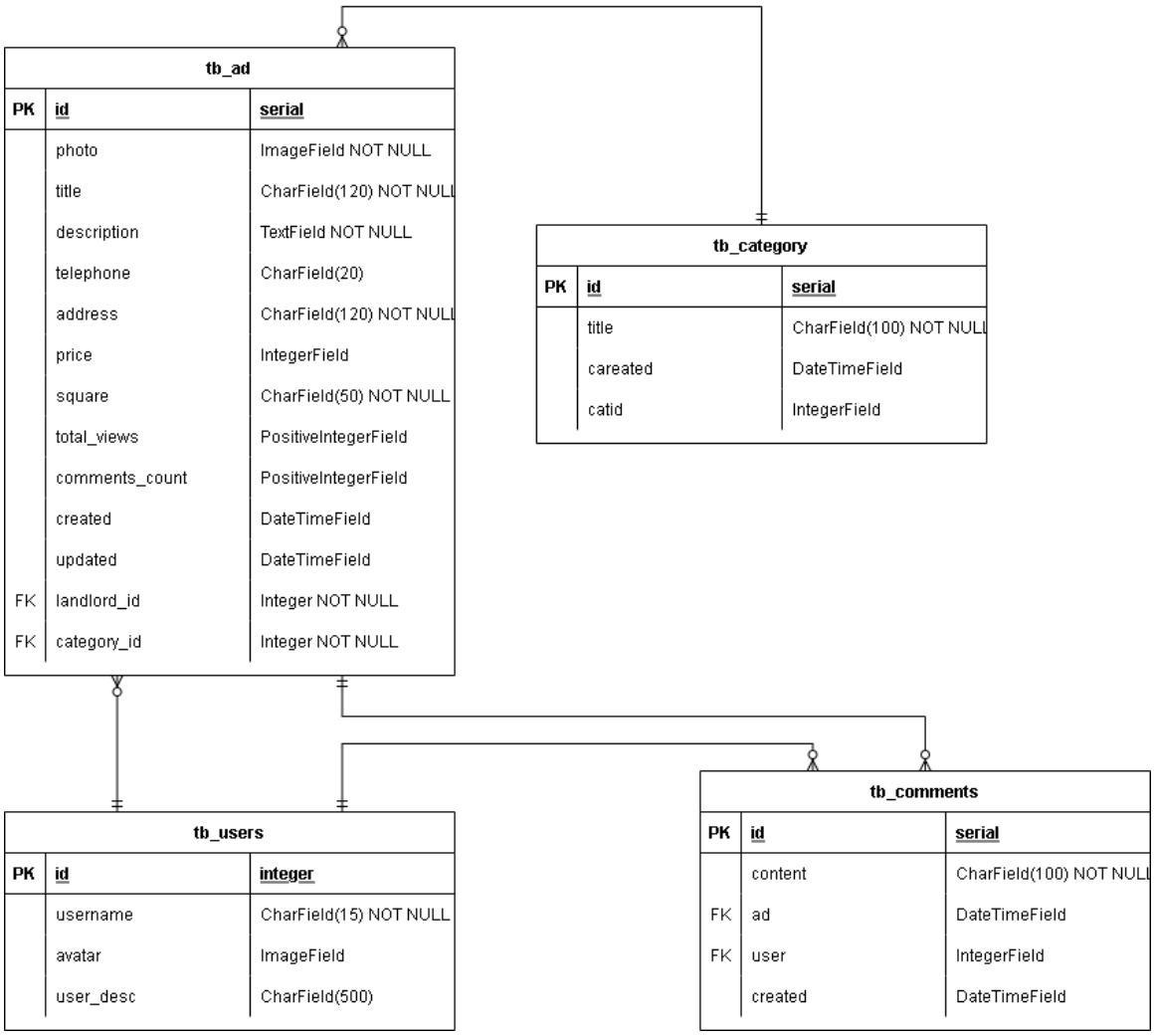


Рисунок 3.1 – даталогическая модель сущностей

```python
from django.db import models
from django.contrib.auth.models import AbstractUser
# Create your models here.

class User(AbstractUser):
    #account
    username=models.CharField(max_length=15,unique=True, blank=False)
    #Avatar
    avatar=models.ImageField(upload_to='avatar/%Y%M%D', blank=True)
    #Introduction
    user_desc=models.CharField(max_length=500, blank=True)
    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['email']
    class Meta:
        db_table='tb_users'
        verbose_name='User_Management' #admin background display
        verbose_name_plural=verbose_name #admin background display
    def __str__(self):
        return self.username
```

Рисунок 3.2 – Код для создания модели User

```python
from django.db import models
from django.utils import timezone
# Create your models here.

class AdsCategory(models.Model):
    title = models.CharField(max_length=100, blank=True)
    catid=models.IntegerField(blank=False)
    created = models.DateTimeField(default=timezone.now)
    def __str__(self):
        return self.title
    #admin site display, easy to debug and view objects
    class Meta:
        db_table = 'tb_category' #modify table name
        verbose_name = 'Category management' #admin site display
        verbose_name_plural = verbose_name
```

Рисунок 3.3 – Код для создания модели Category

```python
from users.models import User
from django.utils import timezone
class Ad(models.Model):
    #on_delete: When the data in the user table is deleted, the article information is also deleted synchrono
    landlord = models.ForeignKey(User,on_delete=models.CASCADE)
    adid = models.IntegerField(null=True, blank=True)
    telephone = models.CharField(null=True, blank=True,max_length=20)
    #author = models.CharField(max_length=50, blank=True)
    address=models.CharField(max_length=120,blank=False)
    metro=models.CharField(max_length=120,blank=False)
    square=models.CharField(max_length=50,blank=False)
    title = models.CharField(max_length=120,blank=False)
    price=models.IntegerField(blank=False)
    category = models.ForeignKey(AdsCategory,null=True,blank=True,on_delete=models.CASCADE,related_name='ad')
    photo = models.ImageField(upload_to='imgcian/', blank=True)
    #tags = models.CharField(max_length=50,blank=True)
    #summary = models.CharField(max_length=600,null=False,blank=False)
    description = models.TextField()
    total_views = models.PositiveIntegerField(default=0)
    comments_count = models.PositiveIntegerField(default=0)
    created = models.DateTimeField(default=timezone.now)
    updated = models.DateTimeField(auto_now=True)
    #Modify the table name and configuration information.
    class Meta:
        db_table = 'tb_ad'
        ordering = ('-created',)
        verbose_name = 'Ad management'
        verbose_name_plural = verbose_name
        unique_together = (("address", "title", "category", "photo"),)
    def __str__(self):
        return self.title
```

Рисунок 3.4 – Код для создания модели Ad

```python
class Comment(models.Model):
    content = models.TextField()
    ad = models.ForeignKey(Ad,on_delete=models.SET_NULL,null=True)
    user = models.ForeignKey('users.User',on_delete=models.SET_NULL, null=True)
    created = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.ad.title
    class Meta:
        db_table = 'tb_comment'
        verbose_name = 'Comment management'
        verbose_name_plural = verbose_name
```

Рисунок 3.5 – Код для создания модели Comment

```python
class RPESModel(models.Model):
    r1 = models.FloatField()
    m1 = models.FloatField()
    m2 = models.FloatField()
    m3 = models.FloatField()
    m4 = models.FloatField()
    m5 = models.FloatField()
    m6 = models.FloatField()
    m7 = models.FloatField()
    m8 = models.FloatField()
    m9 = models.FloatField()
    m10 = models.FloatField()
    m11 = models.FloatField()
    m12 = models.FloatField()
    m13 = models.FloatField()
    m14 = models.FloatField()
    m15 = models.FloatField()
    s1 = models.FloatField()
    c1 = models.FloatField()
    def __str__(self):
        return str(self.r1)+' * room + '+str(self.m1)+' * metro1 + '+str(self.m2)+' * metro2 + '+str(self.m3)+' * metro3 + '+str(self.m4)+\
               ' * metro4 + '+str(self.m5)+' * metro5 + '+str(self.m6)+' * metro6 + '+str(self.m7)+' * metro7 + '+str(self.m8)+' * metro8 + '+\
               str(self.m9)+' * metro9 + '+str(self.m10)+' * metro10 + '+str(self.m11)+' * metro11 + '+str(self.m12)+' * metro12 + '+\
               str(self.m13)+' * metro13 + '+str(self.m14)+' * metro14 + '+str(self.m15)+' * metro15 + '+str(self.s1)+' * area + ' + str(self.c1)
    class Meta:
        db_table = 'tb_tempModel'
        verbose_name = 'Model management'
        verbose_name_plural = verbose_name
```

Рисунок 3.6 – Код для создания модели RPESModel

На рисунках 3.2, 3.3, 3.4, 3.5 и 3.6 показаны коды, используемые для создания моделей Пользователь, Категория, Объявления, Комментарий и Моделя оценки соответственно. Стоит отметить, что в связанных моделях, таких как Объявление и Пользователь, если пользователь удален, Объявление, опубликованные этим пользователем, также будут удалены.

КОД ПРОГРАММЫ

На рисунке 4.1 показан код setting.py. Он указывает, что аутентифицированный пользователь системы использует определенную нами модель пользователя вместо пользовательской модели аутентификации по умолчанию системы. Также изменен путь входа в систему и путь сохранения и доступа к изображениям.

```
#Replace the User of the system to use the User defined by ourselves
#The configuration information is "Sub-application name. Model type"
AUTH_USER_MODEL='users.User'

#modify the default redirect link when user is not logged in
LOGIN_URL='/login/'

#setting for the uploaded pictures,save them to the directory 'media'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')

#setting for the unified route for image access.
MEDIA_URL = '/meida/'
```

Рисунок 4.1 - Фрагмент кода программы setting.py.

На рисунке 4.2 показан код /user/urls.py. Этот код связывает функцию view с путем ссылки.

```
#Perform view routing of users sub-applications
from django.urls import path
from users.views import RegisterView, ImageCodeView, LoginView, LogoutView, ForgetPasswordView, UserCenterView, WriteAdView
    #, ImageCodeView, LoginView, LogoutView, ForgetPasswordView, UserCenterView
    # , WriteBlogView
urlpatterns = [
    #The first parameter of path: routing
    #The second parameter of path: view function name
    path('register/',RegisterView.as_view(), name='register'),

    #Image verification code routing
    path('imagecode/', ImageCodeView.as_view(), name='imagecode'),
    #
    path('login/',LoginView.as_view(),name='login'),

    path('logout/', LogoutView.as_view(), name='logout'),

    path('forgetpassword/', ForgetPasswordView.as_view(), name='forgetpassword'),

    path('center/', UserCenterView.as_view(), name='center'),

    path('writead/',WriteAdView.as_view(), name='writead'),

]
```

Рисунок 4.2 - Фрагмент кода программы /user/urls.py.

На рисунке 4.3 показан код /home/urls.py. Этот код связывает функцию view с путем ссылки.

```
from django.urls import path
from home.views import IndexView, DetailView, PredictView
urlpatterns = [
    path('',IndexView.as_view(),name='index'),

    path('detail/',DetailView.as_view(),name='detail'),

    path('predict/', PredictView.as_view(), name='predict')
]
```

Рисунок 4.3 - Фрагмент кода программы /home/urls.py.

```
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

from django.http import HttpResponse
#1.import system logging
#import logging
# #2.create(get) logger
#logger = logging.getLogger('django')
#
#def log(request):
#     # 3.use logger to record info
#    logger.info('info')
#    return HttpResponse('test')

urlpatterns = [
    path('admin/', admin.site.urls),
    #The parameters of include: urlconf_module, app_name
    #urlconf_module: Content of sub-application
    #app_name: Name of sub-application
    path('',include(('users.urls', 'users'), namespace='users')),
    #path('', log),

    path('',include(('home.urls','home'),namespace='home')),
]

#The route of image access.
from django.conf import settings
from django.conf.urls.static import static
urlpatterns += static(settings.MEDIA_URL,document_root = settings.MEDIA_ROOT)
```

Рисунок 4.4 - Фрагмент кода программы /RPESystem/urls.py.

На рис. 4.5–4.7 показан код IndexView, DetailView, в /home/view.py. Когда пользователь открывает главную страницу веб-сайта, соответствующая информация, которая должна отображаться на главней странице, например информация о категории объявлений и информация о пейджинге, получается посредством запроса GET. Если эта категория не существует или эта страница не существует, будет выведена соответствующая информация об ошибке.

```python
class IndexView(View):
    def get(self,request):
        categories=AdsCategory.objects.all()
        cat_id=request.GET.get('cat_id',1)
        try:
            category=AdsCategory.objects.get(id=cat_id)
        except AdsCategory.DoesNotExist:
            return HttpResponseNotFound('No such category!')
        # 4. Get paging parameters
        page_num = request.GET.get('page_num', 1)
        page_size = request.GET.get('page_size', 6)
        ads = Ad.objects.filter(category=category)
        # 6. Create Paginator
        paginator = Paginator(ads, page_size)
        # 7. Paging
        try:
            page_ads = paginator.page(page_num)
        except EmptyPage:
            return HttpResponseNotFound('Empty Page!')
        # total page
        total_page = paginator.num_pages
        # 8. Translate data to templates
        context = {
            'categories': categories,
            'category': category,
            'ads': page_ads,
            'page_size': page_size,
            'total_page': total_page,
            'page_num': page_num
        }
        return render(request, 'index.html',context=context)
```

Рисунок 4.5 - Фрагмент кода IndexView программы /home/view.py.

```python
class DetailView(View):
    def get(self, request):
        # 1. Receive ad id
        id = request.GET.get('id')
        # 4. Get paging parameters
        page_size = request.GET.get('page_size', 6)
        # print(page_size)
        page_num = request.GET.get('page_num', 1)
        # print(page_num)
        # 3. Query category data
        categories = AdsCategory.objects.all()
        # 2. Query ad data by id
        try:
            ad = Ad.objects.get(id=id)
        except Ad.DoesNotExist:
            return render(request, '404.html')
        else:
            ad.total_views += 1
            ad.save()

        # Query the top 10 article data
        hot_ads = Ad.objects.order_by('-total_views')[:9]

        # 5. Query comment data by the info of paging
        comments = Comment.objects.filter(ad=ad).order_by('-created')

        total_count = comments.count()
        # 6. Create Paginator
        paginator = Paginator(comments, page_size)
        # 7. Paging
        try:
            page_comments = paginator.page(page_num)
        except EmptyPage:
            return HttpResponseNotFound('Empty Page!')
        # print(paginator.page(3))
        total_page = paginator.num_pages

        # 8. Translate data to templates
        context = {
            'categories': categories,
            'category': ad.category,
            'ad': ad,
            'hot_ads': hot_ads,
            'total_count': total_count,
            'comments': page_comments,
            'page_size': page_size,
            'total_page': total_page,
            'page_num': page_num
        }
        return render(request, 'detail.html', context=context)
    def post(self, request):
```

```python
def post(self,request):
    #1. Receive user info
    user = request.user
    # 2. Check user is logged in or not
    if user and user.is_authenticated:
        # 3. Logged in users can post form data
        #     3.1 Receive comment data
        id = request.POST.get('id')
        content = request.POST.get('content')
        #     3.2 Check the article is exist or not
        try:
            ad = Ad.objects.get(id=id)
        except Ad.DoesNotExist:
            return HttpResponseNotFound('No such ad!')
        #     3.3 Save comment data
        Comment.objects.create(
            content=content,
            ad=ad,
            user=user
        )
        #     3.4 Modify the number of comments on the article
        ad.comments_count += 1
        ad.save()

        path = reverse('home:detail')+'?id={}'.format(ad.id)
        return redirect(path)
    else:
        # 4. Not logged in users will jump to the login page
        return redirect(reverse('users:login'))
```

Рисунок 4.6 - Фрагмент кода DetailView программы /home/view.py.

```
class PredictView(View):
    def get(self, request):
        return render(request, 'predict.html')
    def post(self,request):
        room = int(request.POST.get('room'))
        metro = request.POST.get('metro')
        square = float(request.POST.get('square'))
        modelob = RPESModel.objects.get(id=1)
        result = modelob.r1 * room
        if metro in ['м.Обухово', 'м.Озерки', 'м.Лесная', 'м.Площадь Мужества', 'м.Ладожская', 'м.Пионерск
                     'м.Академическая', 'м.Московские ворота', 'м.Парк Победы', 'м.Бухарестская', 'м.Моско
                     'м.Обходной канал','м.Звенигородская','м.Черная река','м.Международный','м.Площадь Ал
                     'м.Площадь Восстания','м.Лиговский проспект','м.Гостиный двор','м.Беговая','м.Маяковс
                     'м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект','м.Адмиралтейская','м.П
            result += modelob.m1
        if metro in ['м.Лесная','м.Площадь Мужества','м.Ладожская','м.Пионерская','м.Елизаровская','м.Конк
            'м.Парк Победы','м.Бухарестская','м.Московская','м.Электросила','м.Старая деревня','м.Зенит','
            'м.Международный','м.Площадь Александра Невского','м.Политехнический','м.Приморская','м.Садова
            'м.Беговая','м.Маяковская','м.Выборгская','м.Владимирская','м.Петроградская','м.Сенная площадь
            'м.Площадь Ленина','м.Горьковская','м.Чернышевская','м.Крестовский остров']:
            result += modelob.m2
        if metro in ['м.Бухарестская','м.Московская''м.Электросила','м.Старая деревня','м.Зенит','м.Василе
                     'м.Площадь Александра Невского','м.Политехнический','м.Приморская','м.Садовая','м.Нов
                     'м.Маяковская','м.Выборгская','м.Владимирская','м.Петроградская','м.Сенная площадь','
                     'м.Площадь Ленина','м.Горьковская','м.Чернышевская','м.Крестовский остров']:
            result += modelob.m3
        if metro in ['м.Московская','м.Электросила','м.Старая деревня','м.Зенит','м.Василеостровская','м.Ф
                     'м.Площадь Александра Невского','м.Политехнический','м.Приморская','м.Садовая','м.Нов
                     'м.Маяковская','м.Выборгская','м.Владимирская','м.Петроградская','м.Сенная площадь','
                     'м.Площадь Ленина','м.Горьковская','м.Чернышевская','м.Крестовский остров']:
            result += modelob.m4
        if metro in ['м.Фрунзенская','м.Обходной канал','м.Звенигородская','м.Черная река','м.Международны
                     'м.Новочеркасская','м.Площадь Восстания','м.Лиговский проспект','м.Гостиный двор','м.
                     'м.Сенная площадь','м.Балтийская','м.Спорт','м.Достоевская','м.Чкаловская','м.Невский
            result += modelob.m5
        if metro in ['м.Звенигородская','м.Черная река','м.Международный','м.Площадь Александра Невского',

    if metro in ['м.Звенигородская','м.Черная река','м.Международный','м.Площадь Александра Невского','м.Политехн
                 'м.Площадь Восстания','м.Лиговский проспект','м.Гостиный двор','м.Беговая','м.Маяковская','м.Выб
                 'м.Балтийская','м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект','м.Адмиралтейская',
        result += modelob.m6
    if metro in ['м.Черная река','м.Международный','м.Площадь Александра Невского','м.Политехнический','м.Приморс
                 'м.Площадь Восстания','м.Лиговский проспект','м.Гостиный двор','м.Беговая','м.Маяковская','м.Выб
                 'м.Сенная площадь','м.Балтийская','м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект',
                 'м.Чернышевская','м.Крестовский остров']:
        result += modelob.m7
    if metro in ['м.Маяковская','м.Выборгская','м.Владимирская','м.Петроградская','м.Сенная площадь','м.Балтийска
                 'м.Невский проспект','м.Адмиралтейская','м.Площадь Ленина','м.Горьковская','м.Чернышевская','м.Р
        result += modelob.m8
    if metro in ['м.Выборгская','м.Владимирская','м.Петроградская','м.Сенная площадь','м.Балтийская','м.Спорт','м
                 'м.Адмиралтейская','м.Площадь Ленина','м.Горьковская','м.Чернышевская','м.Крестовский остров']:
        result += modelob.m9
    if metro in ['м.Сенная площадь','м.Балтийская','м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект',
                 'м.Чернышевская','м.Крестовский остров']:
        result += modelob.m10
    if metro in ['м.Балтийская','м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект','м.Адмиралтейская',
        result += modelob.m11
    if metro in ['м.Спорт','м.Достоевская','м.Чкаловская','м.Невский проспект','м.Адмиралтейская','м.Площадь Лени
        result += modelob.m12
    if metro in ['м.Достоевская','м.Чкаловская','м.Невский проспект','м.Адмиралтейская','м.Площадь Ленина','м.Гор
        result += modelob.m13
    if metro in ['м.Чернышевская','м.Крестовский остров']:
        result += modelob.m14
    if metro in ['м.Крестовский остров']:
        result += modelob.m15
    result += modelob.s1 * square + modelob.c1
```

```
context = {
    'result': result,
    'room': room,
    'metro': metro,
    'square': square
}
return render(request, 'predict.html', context=context)
```

Рисунок 4.7 - Фрагмент кода PredictView программы /home/view.py.

Остальной код можно будет просмотреть на странице github - https://github.com/YanTszyafen/RPESystem

ВИЗУАЛЬНЫЙ РЕЗУЛЬТАТ РАБОТЫ

Вся работа над фронтендом сайта реализована с использованием html + CSS + Javascript + DTL, где DTL - язык шаблонов Django, позволяющий интегрировать код python в html.

На рисунке 5.1 показана главная страница, независимо от того, авторизован пользователь или нет.

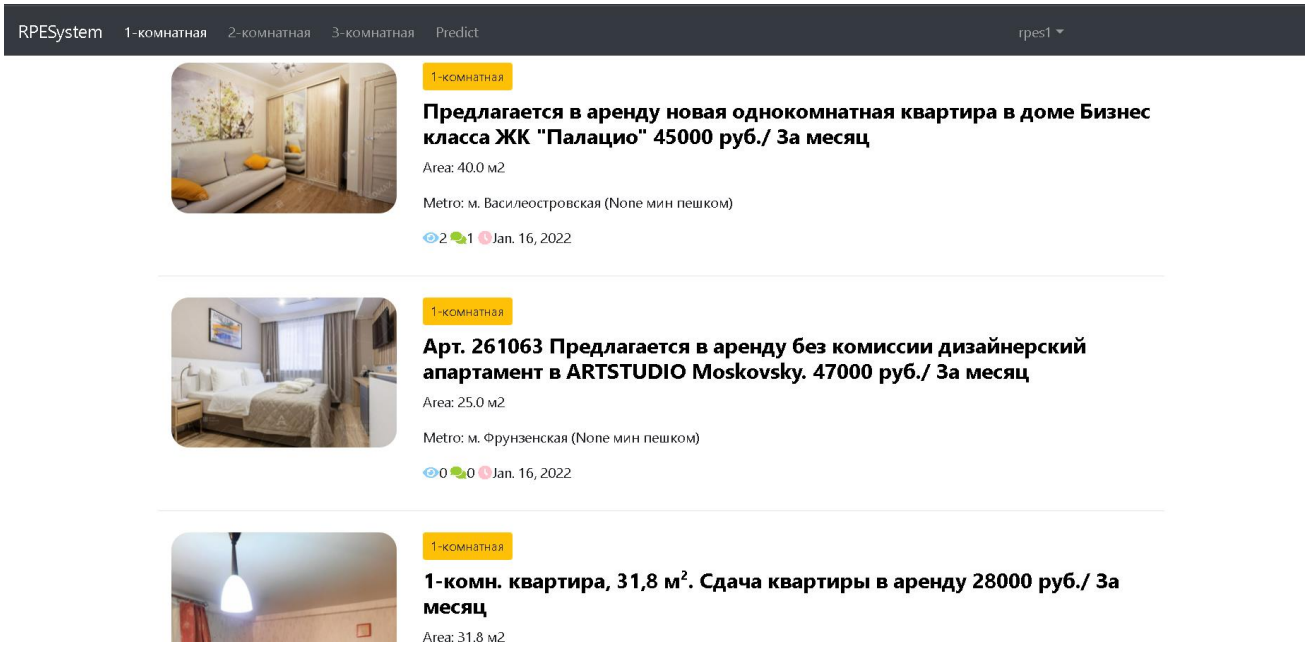
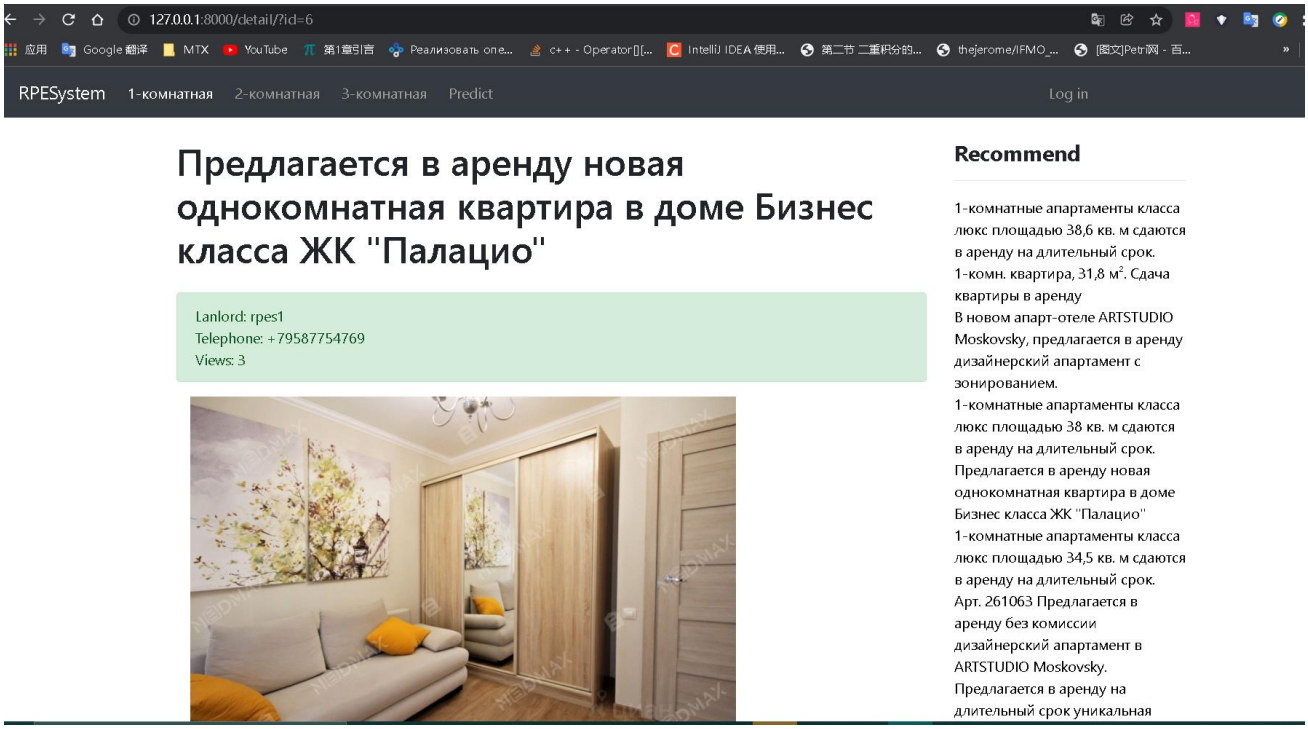


Рисунок 5.1 - Главная страница(Ad)

На рис. 5.2 показана страница статьи, когда пользователь не авторизован, и появляется сообщение с приглашением «войти в систему».



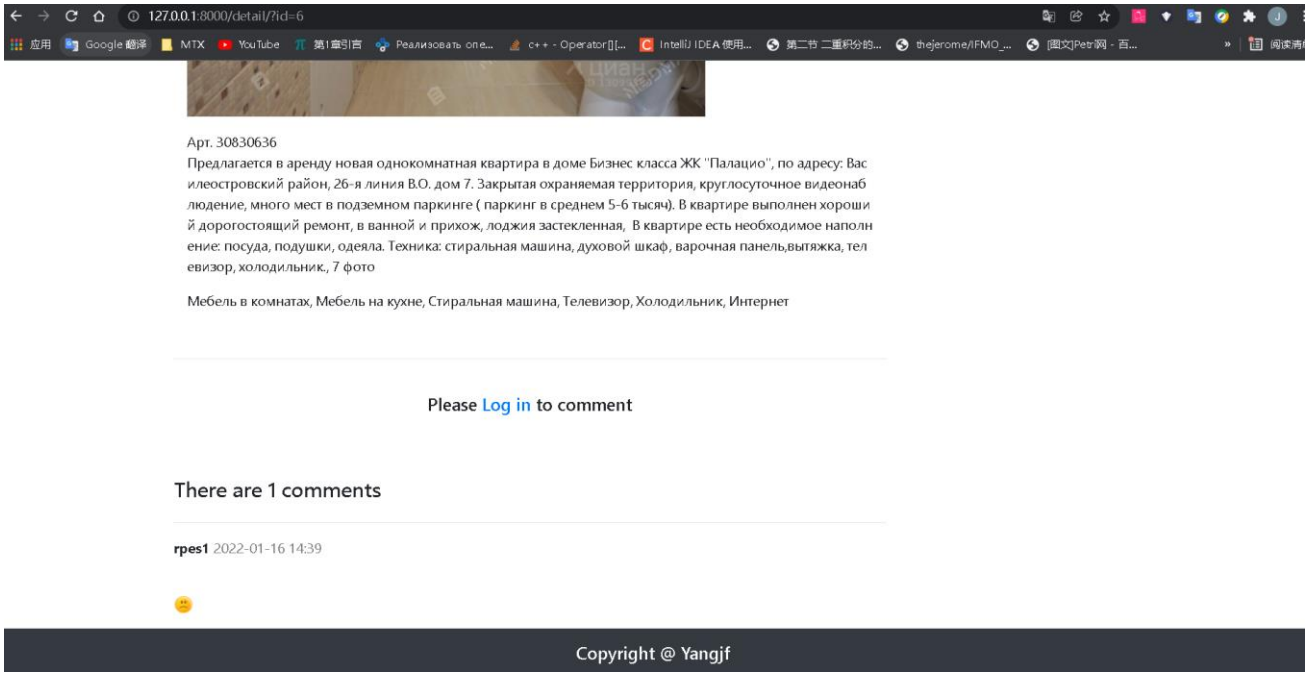


Рисунок 5.2 – Страница Объявления (пользователь не авторизован)

На рисунках 5.3 и 5.4 показаны страницы входа и регистрации соответственно. Код подтверждения изображения на странице регистрации может отображаться как обычно, и щелкните его, чтобы заменить код подтверждения изображения.

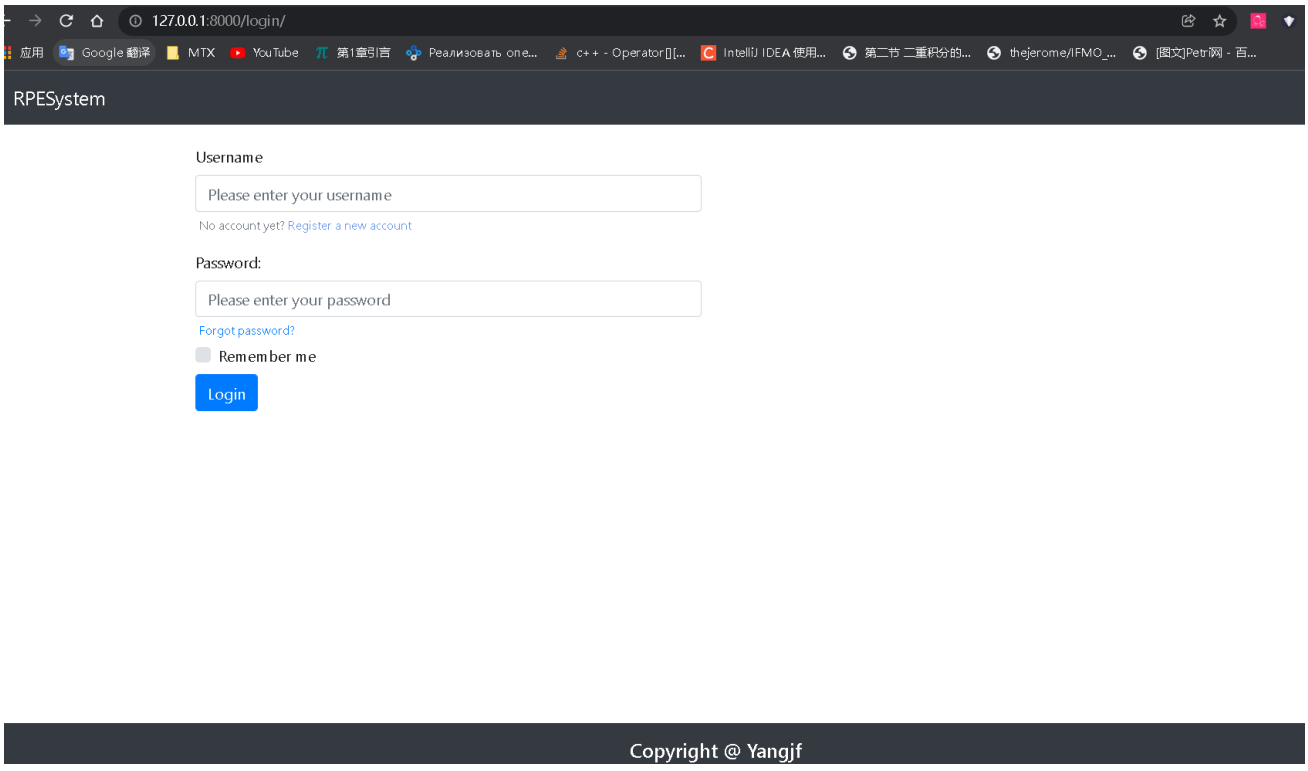


Рисунок 5.3 – Страница Log in

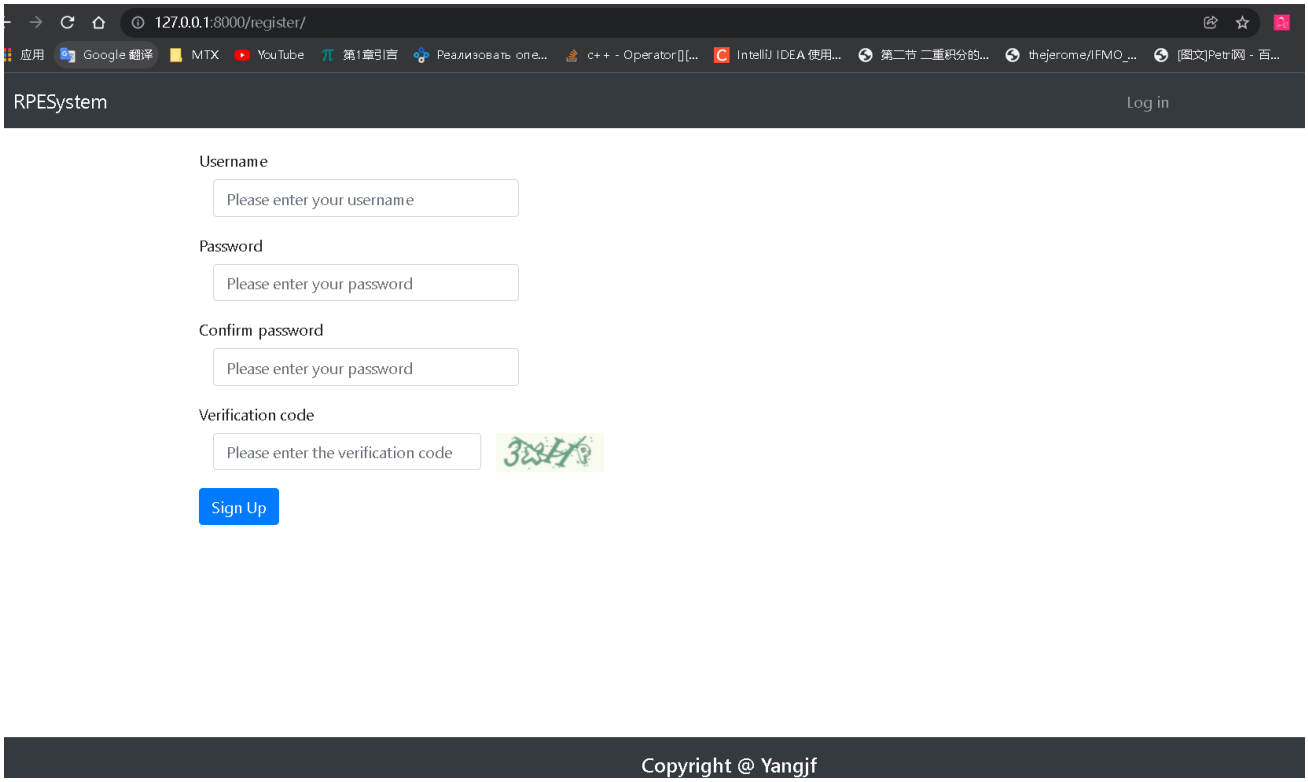


Рисунок 5.4 – Страница регистрации

На рисунке 5.5 показана страница объявления, когда пользователь авторизован, и появляется область для комментариев.

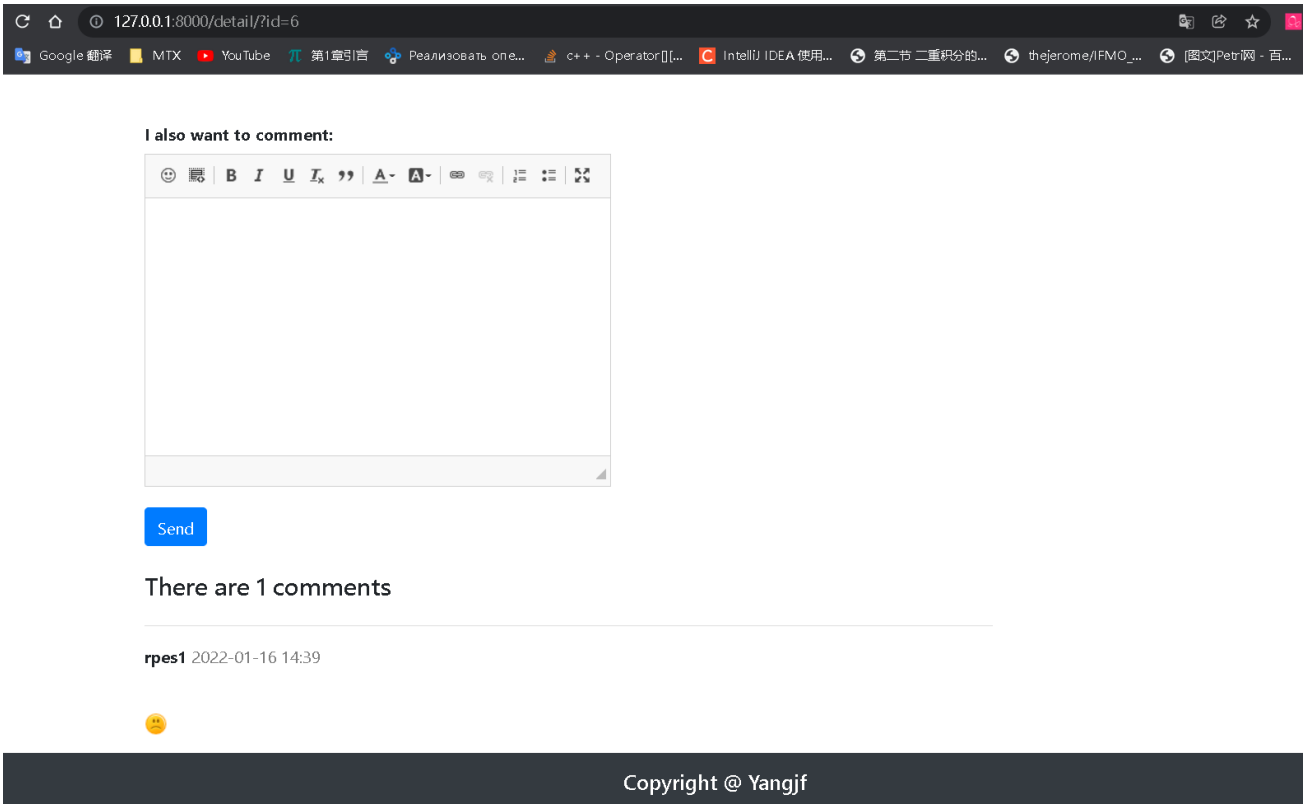


Рисунок 5.5 – Страница объявления(пользователь авторизован)

На рисунках 5.6 и 5.7 соответственно показаны страницы для публикации объявления и изменения личной информации, которые могут появиться только после авторизации пользователя.

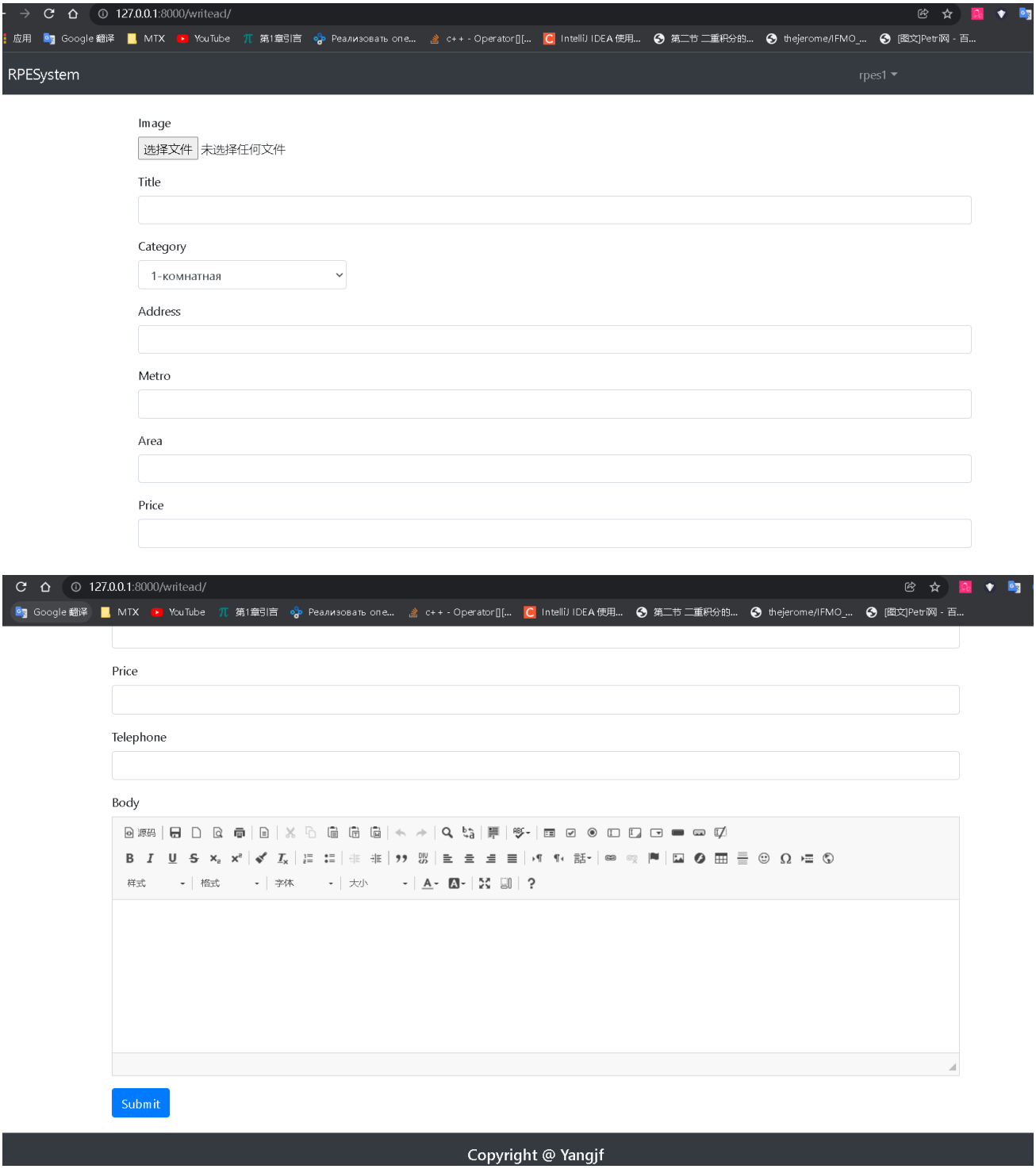


Рисунок 5.6 – Страница публикации объявления (пользователь авторизован)

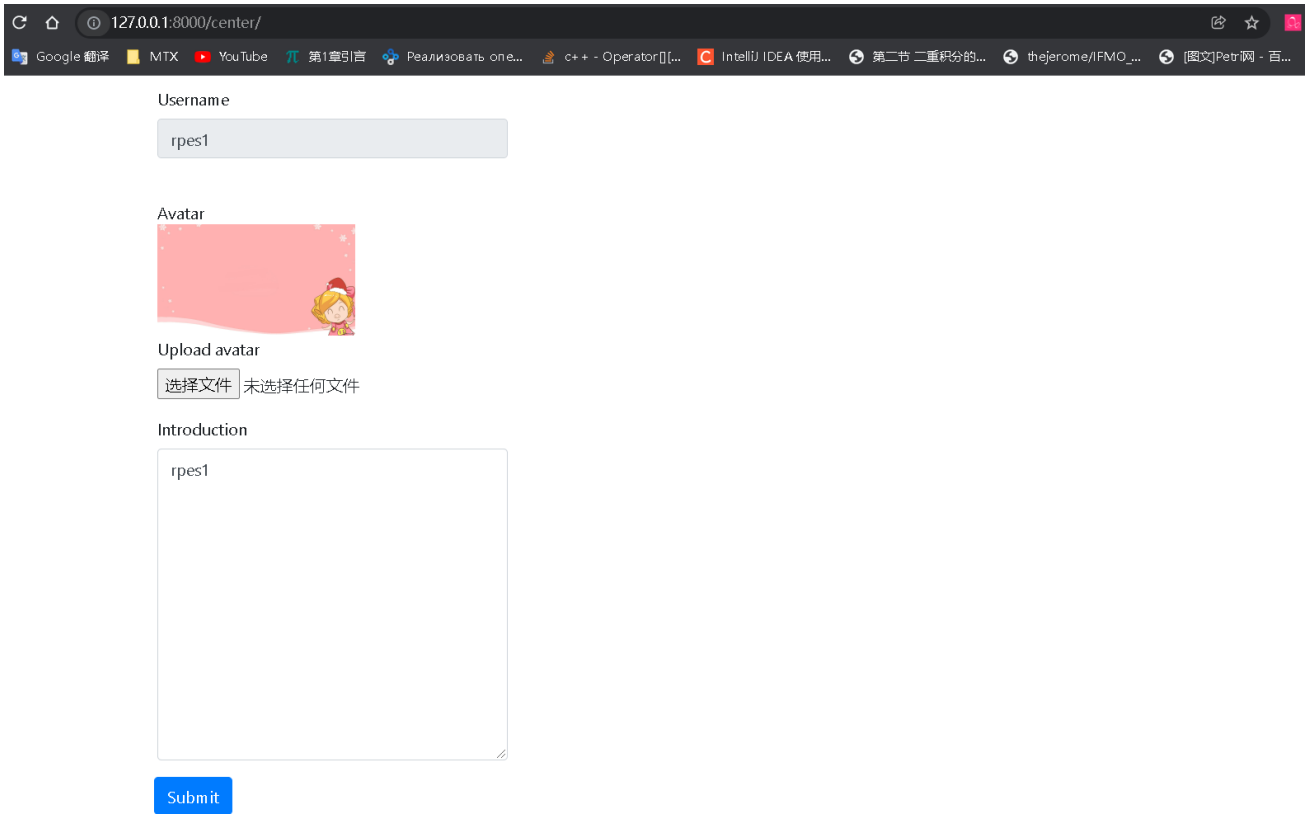


Рисунок 5.7 – Страница редактирования личной информации (пользователь авторизован)

На рисунках с 5.8 по 5.13 показана страница администратора, включая управление пользователей, объявления, категорий объявления и параметров модели оценки.

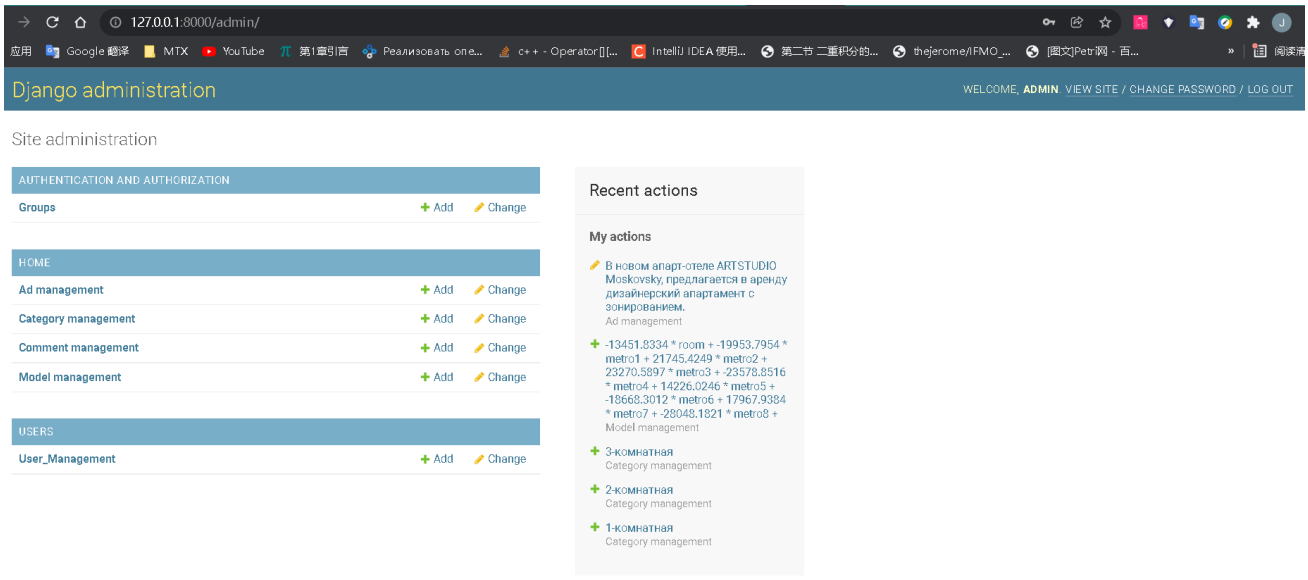


Рисунок 5.8 – Страница administration

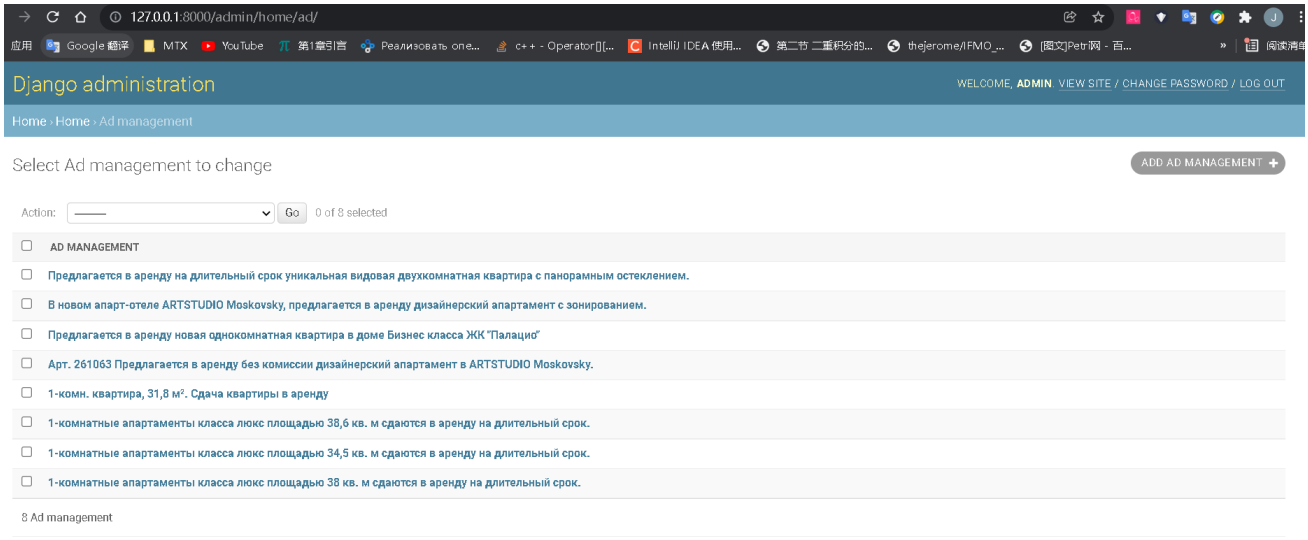


Рисунок 5.9 – Страница administration(Ad-management)

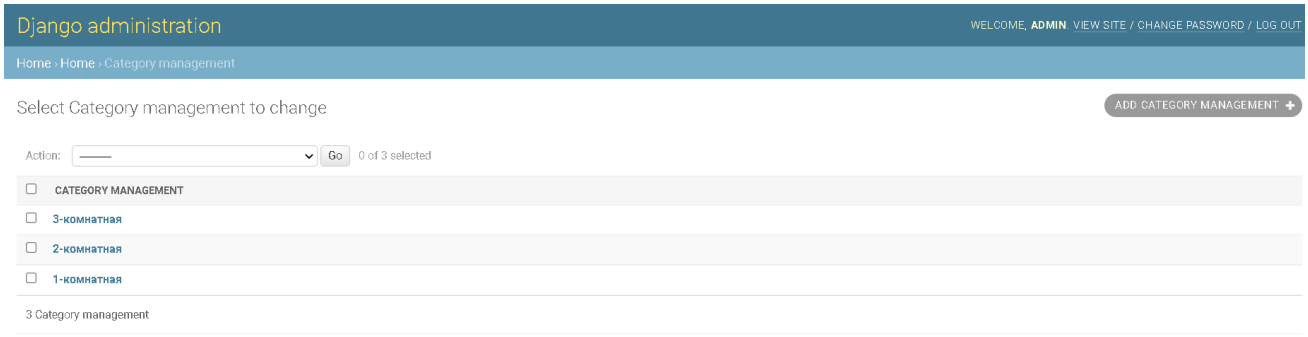


Рисунок 5.10 – Страница administration(Category)

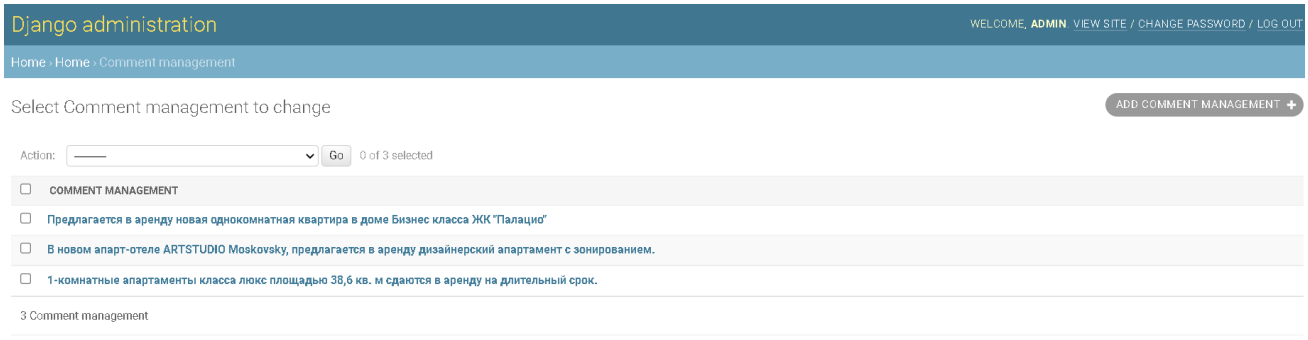


Рисунок 5.11 – Страница administration(Comment)

Рисунок 5.12 – Страница administration(Model)

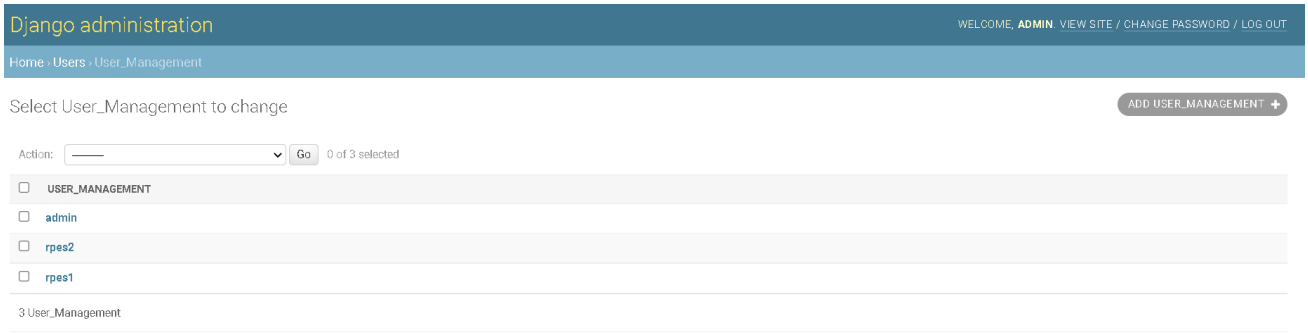


Рисунок 5.13 – Страница administration(User)

На рисунке 5.14 показана страница оценки стоимости квартиры.

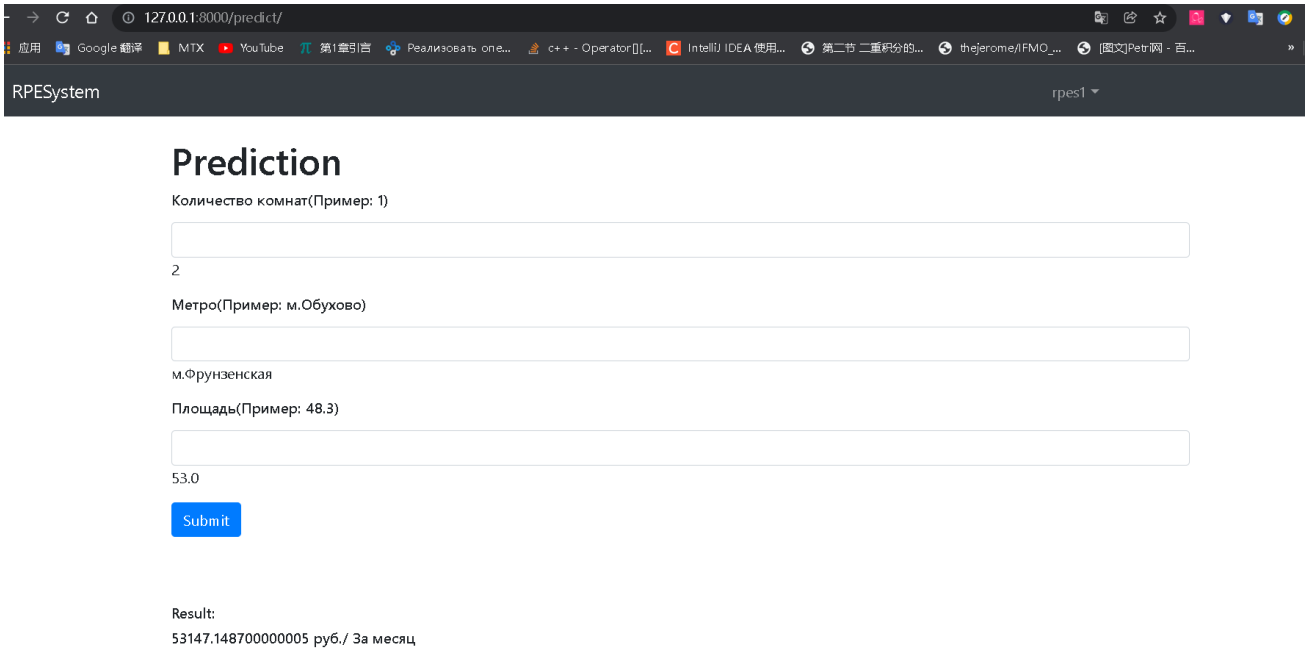


Рисунок 5.13 – Страница оценки стоимости квартиры

Тестирование

Задание

Основное задание для тестирования своего приложения состоит в том, чтобы представить каждый модуль в виде УГП (Управляющий граф программы) и оценить их покрытие данными, ветвями и тестами.

Файл home/view.py

На рис. 1 показан функция get в классе IndexView в /home/view.py.

```
10  class IndexView(View):
11      def get(self,request):
12          categories=AdsCategory.objects.all()
13          cat_id=request.GET.get('cat_id',1)
14          try:
15              category=AdsCategory.objects.get(id=cat_id)
16          except AdsCategory.DoesNotExist:
17              return HttpResponseNotFound('No such category!')
18          # 4. Get paging parameters
19          page_num = request.GET.get('page_num', 1)
20          page_size = request.GET.get('page_size', 6)
21          ads = Ad.objects.filter(category=category)
22          # 6. Create Paginator
23          paginator = Paginator(ads, page_size)
24          # 7. Paging
25          try:
26              page_ads = paginator.page(page_num)
27          except EmptyPage:
28              return HttpResponseNotFound('Empty Page!')
29          # total page
30          total_page = paginator.num_pages
31          # 8. Translate data to templates
32          context = {
33              'categories': categories,
34              'category': category,
35              'ads': page_ads,
36              'page_size': page_size,
37              'total_page': total_page,
38              'page_num': page_num
39          }
40          return render(request, 'index.html',context=context)
41
```

Рисунок 6.1 - Функция get (class IndexView) в файле /home/view.py.

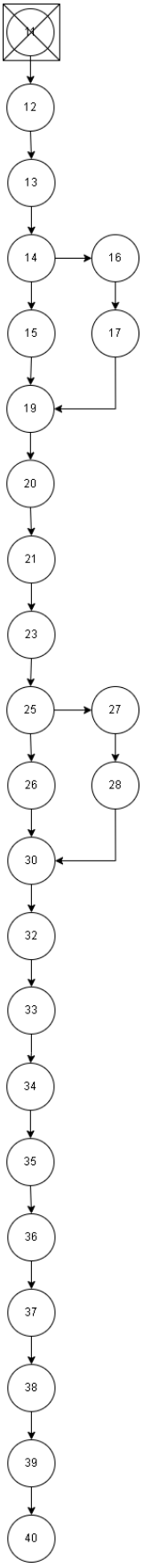


Рисунок 6.2 - УПГ функции get (class IndexView) в файле /home/view.py.

Покрытие данными — 1. Покрытие ветвями — 4/4. Покрытие тестами — 3/3.

```python
class DetailView(View):
    def get(self, request):
        # 1. Receive ad id
        id = request.GET.get('id')
        # 4. Get paging parameters
        page_size = request.GET.get('page_size', 6)
        # print(page_size)
        page_num = request.GET.get('page_num', 1)
        # print(page_num)
        # 3. Query category data
        categories = AdsCategory.objects.all()
        # 2. Query ad data by id
        try:
            ad = Ad.objects.get(id=id)
        except Ad.DoesNotExist:
            return render(request, '404.html')
        else:
            ad.total_views += 1
            ad.save()

        # Query the top 10 article data
        hot_ads = Ad.objects.order_by('-total_views')[:9]

        # 5. Query comment data by the info of paging
        comments = Comment.objects.filter(ad=ad).order_by('-created')

        total_count = comments.count()
        # 6. Create Paginator
        paginator = Paginator(comments, page_size)
        # 7. Paging
        try:
            page_comments = paginator.page(page_num)
        except EmptyPage:
            return HttpResponseNotFound('Empty Page!')

        # print(paginator.page(1))
        # print(paginator.page(2))
        # print(paginator.page(3))
        total_page = paginator.num_pages

        # 8. Translate data to templates
        context = {
            'categories': categories,
            'category': ad.category,
            'ad': ad,
            'hot_ads': hot_ads,
            'total_count': total_count,
            'comments': page_comments,
            'page_size': page_size,
            'total_page': total_page,
            'page_num': page_num
        }
        return render(request, 'detail.html', context=context)
```

Рисунок 6.3 - Функция get (class DetailView) в файле /home/view.py.

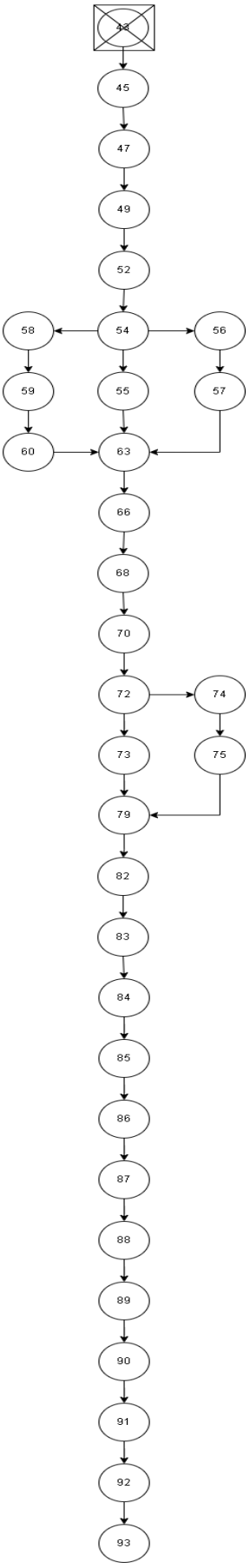


Рисунок 6.4 - УПГ функции get (class DetailView) в файле /home/view.py.

Покрытие данными — 1. Покрытие ветвями — 6/6. Покрытие тестами — 3/3.

```
94      def post(self,request):
95          #1. Receive user info
96          user = request.user
97          # 2. Check user is logged in or not
98          if user and user.is_authenticated:
99              # 3. Logged in users can post form data
100             #     3.1 Receive comment data
101             id = request.POST.get('id')
102             content = request.POST.get('content')
103             #     3.2 Check the article is exist or not
104             try:
105                 ad = Ad.objects.get(id=id)
106             except Ad.DoesNotExist:
107                 return HttpResponseNotFound('No such ad!')
108             #     3.3 Save comment data
109             Comment.objects.create(
110                 content=content,
111                 ad=ad,
112                 user=user
113             )
114             #     3.4 Modify the number of comments on the article
115             ad.comments_count += 1
116             ad.save()
117
118             path = reverse('home:detail')+'?id={}'.format(ad.id)
119             return redirect(path)
120         else:
121             # 4. Not logged in users will jump to the login page
122             return redirect(reverse('users:login'))
```

Рисунок 6.5 - Функция post (class DetailView) в файле /home/view.py.

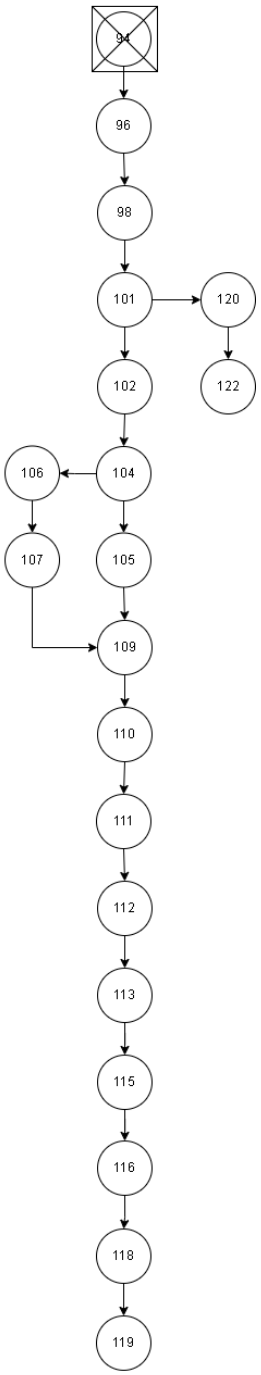


Рисунок 6.6 - УПГ функции post (class DetailView) в файле /home/view.py.

Покрытие данными — 1. Покрытие ветвями — 3/3. Покрытие тестами — 3/3.

Файл user/view.py

```python
class RegisterView(View):
    def get(self, request):
        return render(request, 'register.html')
    def post(self, request):
        #1. Receive data
        username = request.POST.get('username')
        password = request.POST.get('password')
        password2 = request.POST.get('password2')
        image_code = request.POST.get('imgage_code')
        redis_conn = get_redis_connection('default')
        redis_image_code = redis_conn.get('code')
        # print(image_code)
        # 2. Verify data
        #     2.1 Determine whether the parameters are complete
        if not all([username,password,password2,image_code]):
            return HttpResponseBadRequest('Missing required parameters!')
        #     2.2 Verify that the username format is correct
        if not re.match(r'^[0-9A-Za-z]{4,15}$',username):
            return HttpResponseBadRequest('Please enter 4-15 characters, the usern
        #     2.3 Verify that the password format is correct
        if not re.match(r'^[0-9A-Za-z]{8,20}$',password):
            return HttpResponseBadRequest('Please enter 8-20 characters, the passw
        #     2.4 Whether the password and confirm password are the same
        if password != password2:
            return HttpResponseBadRequest('The two passwords are inconsistent!')
        #     2.5
        # Determine whether the image verification code exists
        if redis_image_code is None:
            return HttpResponseBadRequest('Image verification code has expired!')
        #If the image verification code has not expired, we can delete the image v
        try:
            redis_conn.delete('code')
        except Exception as e:
            logger.error(e)
        #
        #Compare image verification code, pay attention to the problem of capitali
        if redis_image_code.decode().lower() != image_code.lower():
            return HttpResponseBadRequest('Image verification code error!')
        # 3. Save registration information
        #create_user can use systematic methods to encrypt the password
        try:
            user = User.objects.create_user(username=username, password=password)
        except DatabaseError as e:
            logger.error(e)
            return HttpResponseBadRequest('Registration failed!')

        from django.contrib.auth import login
        login(request,user)

        # 4. Return the response and jump to the specified page
        # return HttpResponse('Registration is successful!')
        #redirect
        #reverse: The route corresponding to the view can be obtained through the
        response = redirect(reverse('home:index'))
        #setting for info of cookie to show the info of user in homepage
        response.set_cookie('is_login', True)
        response.set_cookie('username',user.username,max_age=7*24*3600)
        return response
```

Рисунок 6.7 – Функции get и post (class RegisterView) в файле /user/view.py.

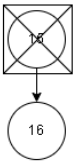


Рисунок 6.8 - УПГ функции get (class RegisterView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 1/1. Покрытие тестами — 1/1.

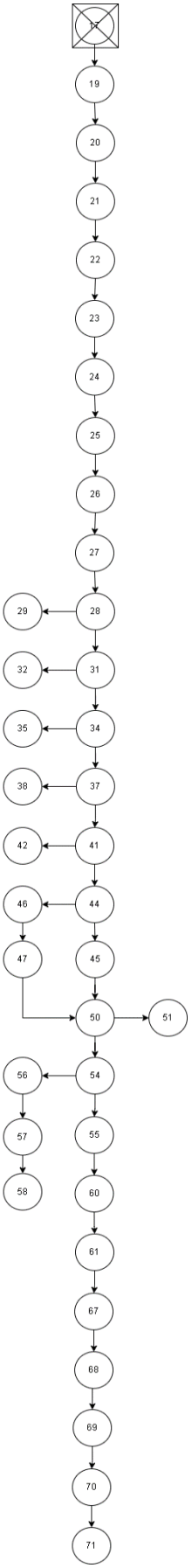


Рисунок 6.9 - УПГ функции post (class RegisterView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 10/10. Покрытие тестами — 8/8.

```python
class LoginView(View):

    def get(self,request):
        return render(request,'login.html')


    def post(self,request):
        username = request.POST.get('username')
        password = request.POST.get('password')
        remember = request.POST.get('remember')
        # 2. Verification of parameters
        #     2.1 Verification of username
        if not re.match(r'^[0-9A-Za-z]{4,15}$', username):
            return HttpResponseBadRequest('Username does not conform to the rules!')
        #     2.2 Verification of password
        if not re.match(r'^[0-9A-Za-z]{8,20}$',password):
            return HttpResponseBadRequest('Password does not conform to the rules!')
        from django.contrib.auth import authenticate
        # The default of authentication method is aimed to judge the username for the
        user = authenticate(username=username,password=password)
        if user is None:
            return HttpResponseBadRequest('Wrong username or password!')
        from django.contrib.auth import login
        login(request, user)
        next_page = request.GET.get('next')
        if next_page:
            response=redirect(next_page)
        else:
            response = redirect(reverse('home:index'))
        if remember != 'on':#did not remember the info of user
            #after the browser is closed
            request.session.set_expiry(0)
            response.set_cookie('is_login', True)
            response.set_cookie('username',user.username,max_age=14*24*3600)
        else:#remembered the info of user
            request.session.set_expiry(None) #default time is 2 weeks
            response.set_cookie('is_login',True, max_age=14*24*3600)
            response.set_cookie('username',user.username,max_age=14*24*3600)
        return response
```

Рисунок 6.10 - Функции get и post (class LoginView) в файле /user/view.py.

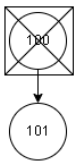


Рисунок 6.11 - УПГ функции get (class LoginView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 1/1. Покрытие тестами — 1/1.

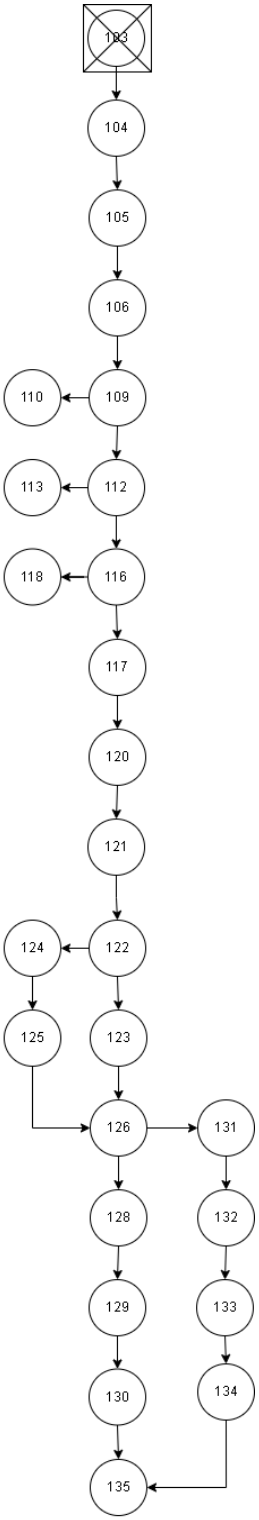


Рисунок 6.12 - УПГ функции post (class LoginView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 7/7. Покрытие тестами — 4/4.

```
139 class LogoutView(View):
140     def get(self,request):
141         # 1.delete session data
142         logout(request)
143         # 2.delete some cookie data
144         response = redirect(reverse('home:index'))
145         response.delete_cookie('is_login')
146         # 3.redirect to the homepage
147         return response
```

Рисунок 6.13 - Функции get (class LogoutView) в файле /user/view.py.

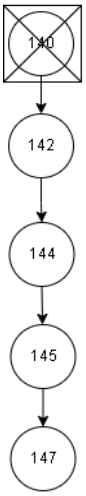


Рисунок 6.14 - УПГ функции get (class LogoutView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 1/1. Покрытие тестами — 1/1.

```python
    def get(self,request):
        return render(request,'forget_password.html')

    def post(self,request):
        username = request.POST.get('username')
        password = request.POST.get('password')
        password2 = request.POST.get('password2')
        image_code = request.POST.get('imgage_code')
        redis_conn = get_redis_connection('default')
        redis_image_code = redis_conn.get('code')
        # 2. Verify data
        #     2.1 Determine whether the parameters are complete
        if not all([username,password,password2,image_code]):
            return HttpResponseBadRequest('Missing required parameters!')
        #     2.2 Verify that the username format is correct
        if not re.match(r'^[0-9A-Za-z]{4,15}$',username):
            return HttpResponseBadRequest('Please enter 4-15 characters, the user
        #     2.3 Verify that the password format is correct
        if not re.match(r'^[0-9A-Za-z]{8,20}$',password):
            return HttpResponseBadRequest('Please enter 8-20 characters, the pass
        #     2.4 Whether the password and confirm password are the same
        if password != password2:
            return HttpResponseBadRequest('The two passwords are inconsistent!')
        #     2.5 Verify that the verification code is correct
        # Determine whether the image verification code exists
        if redis_image_code is None:
            return HttpResponseBadRequest('Image verification code has expired!')
        #If the image verification code has not expired, we can delete the image
        try:
            redis_conn.delete('code')
        except Exception as e:
            logger.error(e)

        #Compare image verification code, pay attention to the problem of capitali
        if redis_image_code.decode().lower() != image_code.lower():
            return HttpResponseBadRequest('Image verification code error!')
        # 3. Query user information based on username
        try:
            user = User.objects.get(username = username)
        except User.DoesNotExist:
            # 5. else create new user
            try:
                User.objects.create_user(username=username,password=password)
            except Exception:
                return HttpResponseBadRequest('The modification failed, please try
        else:
            # 4. If exists this username, reset the password
            user.set_password(password)
            user.save()
        # 6. Redirect to login page
        response = redirect(reverse('users:login'))
        # 7. Return response
        return response
```

Рисунок 6.15 - Функции get и post (class ForgetPasswordView) в файле /user/view.py.

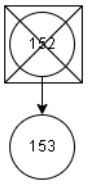


Рисунок 6.16 - УПГ функции get (class ForgetPasswordView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 1/1. Покрытие тестами — 1/1.

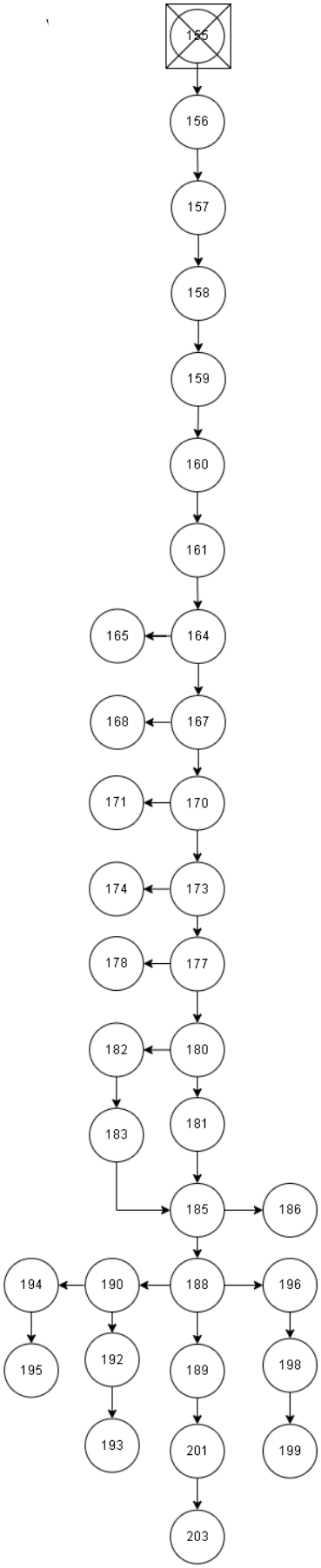


Рисунок 6.17 - УПГ функции post (class ForgetPasswordView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 15/15. Покрытие тестами — 8/8.

```python
class UserCenterView(LoginRequiredMixin, View):

    def get(self,request):
        # get the information of user
        user = request.user
        context = {
            'username': user.username,
            #'mobile': user.mobile,
            'avatar': user.avatar.url if user.avatar else None,
            'user_desc': user.user_desc
        }
        return render(request,'center.html',context=context)
```

```python
    def post(self, request):
        user = request.user
        #1. Receive parameters
        username = request.POST.get('username',user.username)
        user_desc = request.POST.get('desc',user.user_desc)
        avatar = request.FILES.get('avatar')
        # 2. Save parameters
        try:
            user.username=username
            user.user_desc=user_desc
            if avatar:
                user.avatar=avatar
            user.save()
        except Exception as e:
            logger.error(e)
            return HttpResponseBadRequest('The modification failed, please try again later!')
        # 3. Refresh the current page (redirect operation)
        response = redirect(reverse('users:center'))
        # 4. Return response
        return response
```

Рисунок 6.18 - Функции get и post (class UserCenterView) в файле /user/view.py.

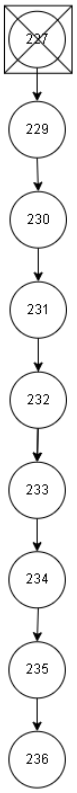


Рисунок 6.19 - УПГ функции get (class UserCenterView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 1/1. Покрытие тестами — 1/1.

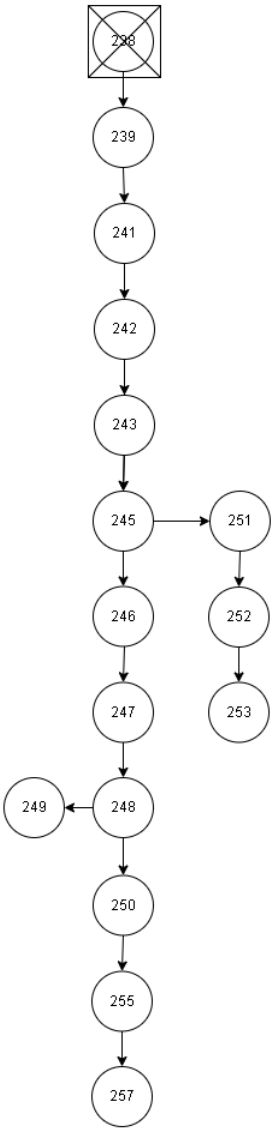


Рисунок 6.19 - УПГ функции post (class UserCenterView) в файле /user/view.py.

Покрытие данными — 1. Покрытие ветвями — 3/3. Покрытие тестами — 2/2.

Вывод

Проверила все модули в приложении и покрытие данными, ветвями и тестами составило 100%.

Модели COCOMO II

Описание модели COCOMO II

COnstructive COst MOdel (конструктивная модель стоимости) – это алгоритмическая модель оценки стоимости разработки программного обеспечения (ПО), разработанная Барри Боэмом (Barry Boehm). Модель использует простую формулу регрессии с параметрами, определенными из данных, собранных по ряду проектов.

Формула для оценивания трудоемкости в чел * мес имеет вид:

$$PM = EAF \times A \times SIZE^{E}$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$, где

1. B = 0.91, A = 2.94 – для предварительной оценки

2. SFj – фактор масштаба (Scale Factors)

3. $SIZE$ – объем программного продукта в тысячах исходных строк

4. EMj – множители трудоемкости (Effort Multiplier). Для предварительной оценки n = 7

5. EAF (Effort Adjustment Factor) – произведение выбранных множителей трудоемкости:

$$EAF = \prod_{k=1}^{n} EM_k$$

Таблица 1. Описание уровней значимости факторов масштаба

SF_j	Описание	Уровень значимости фактора					
		Очень низкий	Низкий	Средний	Высокий	Очень высокий	Критический
PREC	Прецедентность, наличие опыта аналогичных разработок	опыт в продукте и платформе отсутствует	продукт и платформа немного знакомы	некоторый опыт в продукте и платформе присутствует	продукт и платформа в основном известны	продукт и платформа в большой степени знакомы	продукт и платформ а полностью знакомы
FLEX	Гибкость процесса разработки	процесс строго предопределен	допускаются некоторые компромиссы	значительная жесткость процесса	относительная жесткость процесса	незначительная жесткость процесса	определены только общие цели
RESL	Архитектура и разрешение рисков	риски известны/ проанализированы на 20%	__"__ На 40%	__"__ На 60%	__"__ На 75%	__"__ На 90%	__"__ На 100%
TEAM	Сработанность команды	формальные взаимодействия	тяжелое взаимодействие до некоторой степени	чаще всего коллективная работа	в основном коллективная работа	высокая степень взаимодействия	полное доверие, взаимозаменяемость и взаимопомощь
PMAT	Зрелость процессов	CMM Уровень 1 ниже среднего	CMM Уровень 1 выше среднего	CMM Уровень 2	CMM Уровень 3	CMM Уровень 4	CMM Уровень 5

CMM (Capability Maturity Model) — пятиуровневая модель зрелости возможностей компании-разработчика ПО, предложенная SEI (Software Engineering Institute, США).

Таблица 2. Значение фактора масштаба

SF_j	Оценка уровня фактора					
	Очень низкий	Низкий	Средний	Высокий	Очень высокий	Критический
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Таблица 3. Значения множителей трудоемкости

№	Множитель трудоемкости, EM_j	Описание	Оценка уровня множителя трудоемкости						
			Супер низкий	Очень низкий	Низкий	Нормальный	Высокий	Очень высокий	Супер высокий
1	PERS	квалификация персонала	2.12	1.62	1.26	1.00	0.83	0.63	0.50
2	PREX	опыт персонала	1.59	1.33	1.22	1.00	0.87	0.74	0.62
3	RCPX	сложность и надежность продукта	0.49	0.60	0.83	1.00	1.33	1.91	2.72
4	RUSE	разработка для повторного использования	n/a	n/a	0.95	1.00	1.07	1.15	1.24
5	PDIF	сложность платформы разработки	n/a	n/a	0.87	1.00	1.29	1.81	2.61
6	FCIL	оборудование	1.43	1.30	1.10	1.00	0.87	0.73	0.62
7	CSED	требуемое выполнение графика работ	n/a	1.43	1.14	1.00	1.00	n/a	n/a

Примечание: n/a – соответствующий уровень не оценивается

Расчет значения

COCOMO II - Constructive Cost Model

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	4000						
Reused		0	0				
Modified							

Software Scale Drivers

Precedentedness	Nominal	Architecture / Risk Resolution	Low	Process Maturity	Low
Development Flexibility	Nominal	Team Cohesion	Nominal		

Software Cost Drivers

Product		Personnel		Platform	
Required Software Reliability	Low	Analyst Capability	Low	Time Constraint	Nominal
Data Base Size	Low	Programmer Capability	High	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	High	Platform Volatility	Nominal
Developed for Reusability	Low	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	Low	Platform Experience	Nominal	Use of Software Tools	Nominal
		Language and Toolset Experience	Nominal	Multisite Development	High
				Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars): 1000

Calculate

Результат

Results

Software Development (Elaboration and Construction)

Effort = 9.1 Person-months
Schedule = 7.5 Months
Cost = $9106

Total Equivalent Size = 4000 SLOC
Effort Adjustment Factor (EAF) = 0.65

Staffing Profile

Your project is too small to display a staffing profile due to truncation.

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.5	0.9	0.6	$546
Elaboration	2.2	2.8	0.8	$2186
Construction	6.9	4.7	1.5	$6921
Transition	1.1	0.9	1.2	$1093

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.3	0.7	0.2
Environment/CM	0.1	0.2	0.3	0.1
Requirements	0.2	0.4	0.6	0.0
Design	0.1	0.8	1.1	0.0
Implementation	0.0	0.3	2.4	0.2
Assessment	0.0	0.2	1.7	0.3
Deployment	0.0	0.1	0.2	0.3

Your output file is at http://softwarecost.org/tools/COCOMO/data/COCOMO_January_16_2022_21_33_23_257734.txt

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu.

Effort = 9.1 Person-months

Schedule = 7.5 Months

Cost = $9106

ЗАКЛЮЧЕНИЕ

Реализовано веб-приложение с функцией регистрации, входа в систему, изменения паролей, изменения личной информации пользователя, публикации объявления, публикации комментариев, оценки стоимости квартиры, рекомендации объявлений и пейджинга.

Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Ян Цзяфэн

Кафедра Компьютерных образовательных технологий **группы** P34212

Руководитель Штенников Д.Г., СПб НИУ ИТМО, кафедра КОТ, доцент

(Фамилия, И., О., место работы, должность)

Дисциплина Инжиниринг программных систем

Наименование темы Разработка системы оценки стоимости аренды квартир.

Задание Проектирование и реализация клиент-серверного приложения для сбора и анализа информации. Выполнения данного приложения по 7 этапам: ТЗ по ГОСТ 19, ТЗ по ГОСТ 34, Диаграмма BPMN, Диаграмма EPC, Конструирование, Тестирование, COCOMO II. Оформление пояснительной записки.

Студент _____ Дата «_____»_____2022 г.

Подпись Дата

Руководитель _____ Дата «_____» _____2022 г.

Подпись Дата

Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

АННОТАЦИЯ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Ян Цзяфэн

Кафедра Компьютерных образовательных технологий **группы** P34212

Руководитель Штенников Д.Г., СПб НИУ ИТМО, кафедра КОТ, доцент

(Фамилия, И., О., место работы, должность)

Дисциплина Инжиниринг программных систем

Наименование темы Разработка системы оценки стоимости аренды квартир

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ОТЗЫВ РУКОВОДИТЕЛЯ О ВЫПОЛНЕНИИ

КУРСОВОЙ РАБОТЫ

Студент Ян Цязфэн

(Фамилия, И., О.)

Кафедра Компьютерных образовательных технологий **Группа** P34212

Руководитель Штенников Д.Г., СПб НИУ ИТМО, кафедра КОТ, доцент

(Фамилия, И., О., место работы, должность)

Дисциплина Инжиниринг программных систем

Наименование темы Разработка системы оценки стоимости аренды квартир

ОЦЕНКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)

№ п/п	Показатели	Оценка			
		5	4	3	0
1.	Способность к работе с литературными источниками, справочной литературой, Интернет-ресурсами и т. п.				
2.	Использование иностранных источников				
3.	Способность к анализу и обобщению информационного материала				
4.	Владение базовыми знаниями в профессиональной области				
5.	Владение базовыми знаниями в смежных областях				
6.	Владение навыками решения технических задач				
7.	Способность применять знания на практике				
8.	Уровень и корректность использования в работе методов численного моделирования, инженерных расчетов и статистической обработки данных				
9.	Владение навыками использования современных пакетов				

№ п/п	Показатели	Оценка			
		5	4	3	0
	компьютерных программ и технологий				
10.	Владение навыками оформления отчетных материалов с применением современных пакетов программ				
11.	Качество оформления пояснительной записки (общий уровень грамотности, стиль изложения, качество иллюстраций, корректность цитирования и пр.**)				
12.	Качество оформления презентации				
13.	Владение навыками публичного выступления и межперсональной коммуникации				
14.	Владение навыками планирования и управления временем при выполнении работы				
Итоговая оценка					

* - не оценивается (трудно оценить)

** согласно рекомендациям

Отмеченные достоинства: _____

_____

_____

_____

_____

_____

Отмеченные недостатки: _____

_____

_____

_____

_____

Заключение: _____

_____

_____

_____

_____

Студент _____ Дата «_____» _____2022 г.

Подпись Дата

Руководитель_____ Дата «_____» _____2022 г.

Подпись Дата

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

АННОТАЦИЯ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Ян Цязфэн

(Фамилия, И., О.)

Кафедра Компьютерных образовательных технологий **Группа** P34212

Руководитель Штенников Д.Г., СПб НИУ ИТМО, кафедра КОТ, доцент

(Фамилия, И., О., место работы, должность)

Дисциплина Инжиниринг программных систем

Наименование темы Разработка системы оценки стоимости аренды квартир

ХАРАКТЕРИСТИКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)

1. Цели и задачи работы

☐ Сформулированы при участии студента

☐ Предложены студентом

☐ Определены руководителем

Проектирование и реализация клиент-серверного приложения для сбора и анализа информации.

2. Характер работы ☐ Расчет ☐ Конструирование

☐ Моделирование ☐ Другое, _____

_____

_____

3. Содержание работы Работа состоит из семи разделов; ТЗ по ГОСТ 19, ТЗ по ГОСТ 34, Диаграмма BPMN, Диаграмма EPC, Конструирование, Тестирование, COCOMO II. В каждом разделе приведено подробное описание проделанной работы. _____

_____

4. Выводы

По результатам выполнения курсовой работы удалось спроектировать и реализовать клиент-серверное приложение для сбора и анализа информации с функцией регистрации, входа в систему, изменения паролей, изменения личной информации пользователя, публикации объявления, публикации комментариев, оценки стоимости квартиры, рекомендации объявлений и пейджинга. _____

_____

Студент _____ Дата «_____» _____2022 г.

Подпись Дата

Руководитель_____ Дата «_____» _____2022 г

Подпись Дата

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ГРАФИК ВЫПОЛНЕНИЯ КУРСОЙ РАБОТЫ

Студент Ян Цязфэн

(Фамилия, И., О.)

Кафедра Компьютерных образовательных технологий **Группа** P34212

Руководитель Штенников Д.Г., СПб НИУ ИТМО, кафедра КОТ, доцент

(Фамилия, И., О., место работы, должность)

Дисциплина Инжиниринг программных систем

Наименование темы Разработка системы оценки стоимости аренды квартир

№ п/п	Наименование этапа	Дата завершения		Оценка и подпись руководителя
		Планируемая	Фактическая	
1	Получение и уточнение задания			
2	Диаграмма вариантов использования			
3	Модель анализа			
4	Диаграмма кооперации			
5	Диаграмма классов проектирования			
6	Диаграмма последовательности			
7	Оформление пояснительной записки			

Студент _____ Дата «_____» _____2022 г.

Подпись Дата

Руководитель_____ Дата «_____» _____2022 г.

Подпись Дата