Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка Telegram-бота для добавления, поиска жилья и нахождения соседей

Студент ИУ7-62Б _____ И.С.Климов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ О.В.Кузнецова
(Подпись, дата) (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И.В. Рудаков
(И.О.Фамилия)
« _____ » _____ 2022 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Базы данных

Студент группы ИУ7-62Б

Климов Илья Сергеевич
(Фамилия, имя, отчество)

Тема курсовой работы Разработка Telegram-бота для добавления, поиска жилья и нахождения соседей

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

учебная

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения работы: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание разработать Telegram-бота, позволяющего регистрироваться пользователям, добавлять и искать объявления об аренде квартир, поиске соседей и продаже бытовых товаров. Предоставить возможность подписки на определенных арендодателей и на появление объявлений с выставленными фильтрами. Разработать функционал, позволяющий пользователю выставлять оценки арендодателям, отмечать понравившиеся квартиры и получать уведомления о новых пользователях, поставивших аналогичную отметку.

Оформление курсовой работы:

2.1. Расчетно-пояснительная записка на 25-30 листах формата А4.
Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.
2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.) На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 2022 г.

Руководитель курсовой работы _____ О.В. Кузнецова
(Подпись, дата) (И.О.Фамилия)

Студент _____ И.С. Климов
(Подпись, дата) (И.О.Фамилия)

Содержание

Введение

Выбор места проживания является одним из самых главных вопросов в жизни человека. Рано или поздно он встанет перед каждым. Особенно это важно для студентов: зачастую ради лучшего образования приходится перебираться в другие города и искать новое жилье, что может быть довольно проблематично. Общежитие является далеко не лучшим вариантом, в своем большинстве студенческие общежития при вузах не отличаются комфортом, нередко здесь можно столкнуться с некачественным ремонтом и плохими условиями. При этом возможное наличие шумных соседей не позволит полноценно отдохнуть и готовиться к учебе. К тому же количество мест здесь ограничено, и существует высокая вероятность остаться без крыши над головой [1]. По состоянию на 2020 год около 20% нуждающихся остались без места [2]. Поэтому оптимальной альтернативой является съем квартиры.

Существует большое количество сервисов с возможностью поиска аренды квартиры. Однако цена может быть довольно высокой, студент попросту не сможет оплатить подходящую квартиру. Решением этой проблемы может стать поиск соседа (сожителя). Помимо экономии денег, это отличный повод для нахождения новых людей и товарищей в незнакомом городе.

Целью данной работы является реализация базы данных, используемой в Telegram-боте, который позволяет студентам находить жилье и соседей. Для достижения данной цели необходимо решить поставленные **задачи**:

1) определить функциональные требования к разрабатываемому программному продукту;
2) определить ролевую модель;
3) провести анализ моделей данных и выбрать наиболее подходящую;
4) спроектировать базу данных, описать ее сущности и связи;
5) заполнить базу данных данными
6) реализовать сервис, позволяющий решить поставленную задачу и обеспечивающий доступ к базе данных;

7) провести сравнительный анализ …

1 Аналитическая часть

В данном разделе ставится постановка задачи, определяются тип сервиса, ролевая модель, модель данных и представляются Use-Case диаграммы и ER-модель.

1.1 Постановка задачи

Для поиска соседа по квартире существует не так много способов: от оставления комментариев в различных тематических группах до опроса знакомых. Все это не является удобным способом достижения желаемого результата. Сервис, включающий в себя соответствующий функционал, существенно облегчит поиски.

1.1.1 Тип сервиса

Сервис можно представить в виде десктопного приложения и веб-приложения. В таком случае приходится тратить ресурсы на создание, например, интерфейса. В данной же работе целесообразно взять технологию, позволяющую сфокусироваться на поставленной цели. Таковой является Telegram-бот, который предоставляет удобный API для взаимодействия.

1.1.2 Ролевая модель

Сервис предполагает многопользовательское использование, при этом четко выделяются несколько типов пользователей и в соответствии от этого доступность определенного функционала.

1. Гость (неавторизированный пользователь) – просмотр всех типов объявлений без выставления фильтров, регистрация для взаимодействия в

системе, авторизация. На рисунке 1.1 представлена Use-Case диаграмма для гостя.

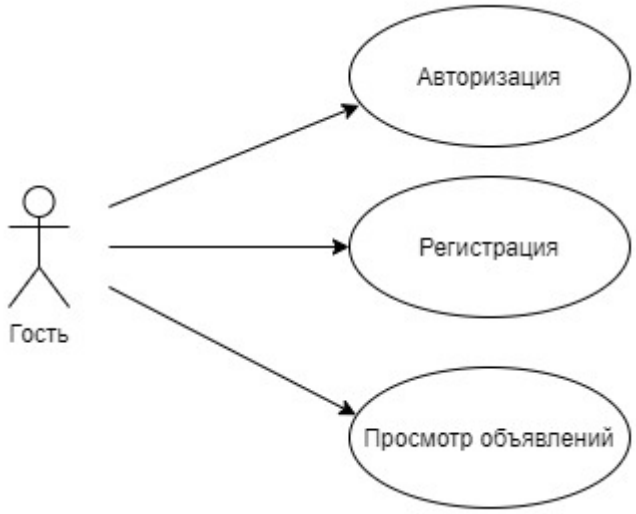

Рисунок 1.1 – Use-Case диаграмма для гостя

2. Арендатор (авторизированный пользователь) – просмотр всех типов объявлений как с фильтрами, так и без; добавление объявлений о поиске соседа и продаже бытовых товаров; подписка на арендодателей и появление квартир; оценка арендодателей, отметка понравившихся квартир, получение уведомлений по подпискам. На рисунке 1.2 представлена Use-Case диаграмма для арендатора.

Рисунок 1.2 – Use-Case диаграмма для разрабатываемой системы

3. Арендодатель (авторизированный пользователь) – просмотр всех типов объявлений как с фильтрами, так и без; добавление объявлений о сдаче квартир в аренду квартир с фотографиями. На рисунке 1.3 представлена Use-Case диаграмма для арендодателя.

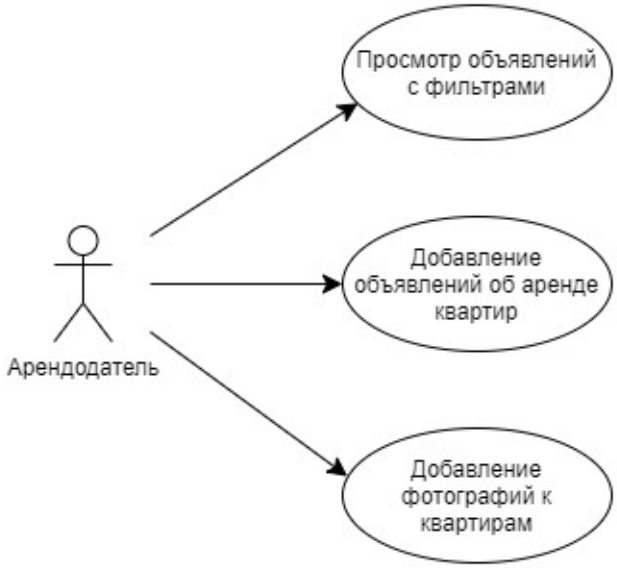

Рисунок 1.3 – Use-Case диаграмма для арендодателя

4. Администратор – добавление удалений пользователей и объявлений всех типов, получение информации о них, ее изменение. На рисунке 1.4 представлена Use-Case диаграмма для администратора.

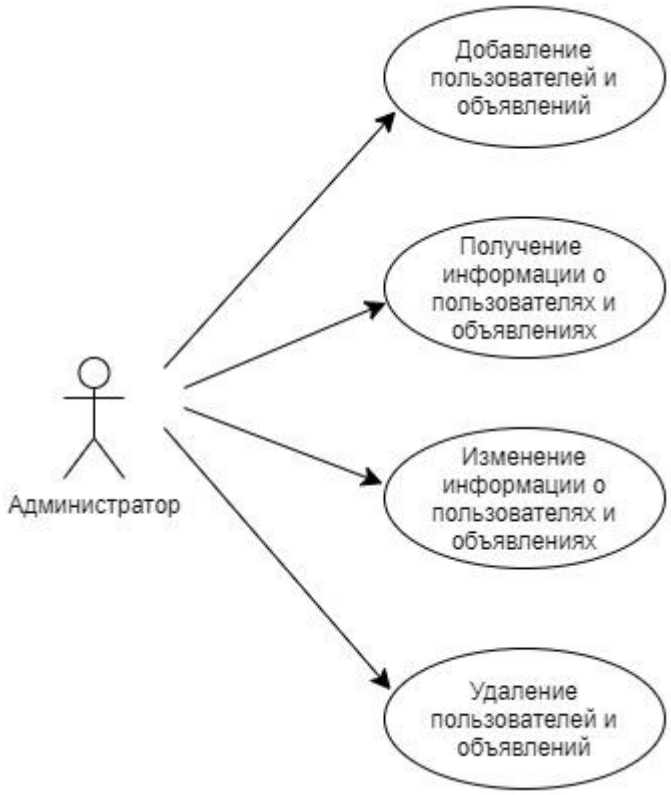

Рисунок 1.4 – Use-Case диаграмма для администратора

1.1.3 Формализация данных

База данных будет содержать информацию про пользователей (арендаторов и арендодателей), объявления (сдаче квартир в аренду, поиске соседа и продаже бытовых товаров), отношения между ними (подписки. На рисунке 1.5 представлена соответствующая ER-модель.

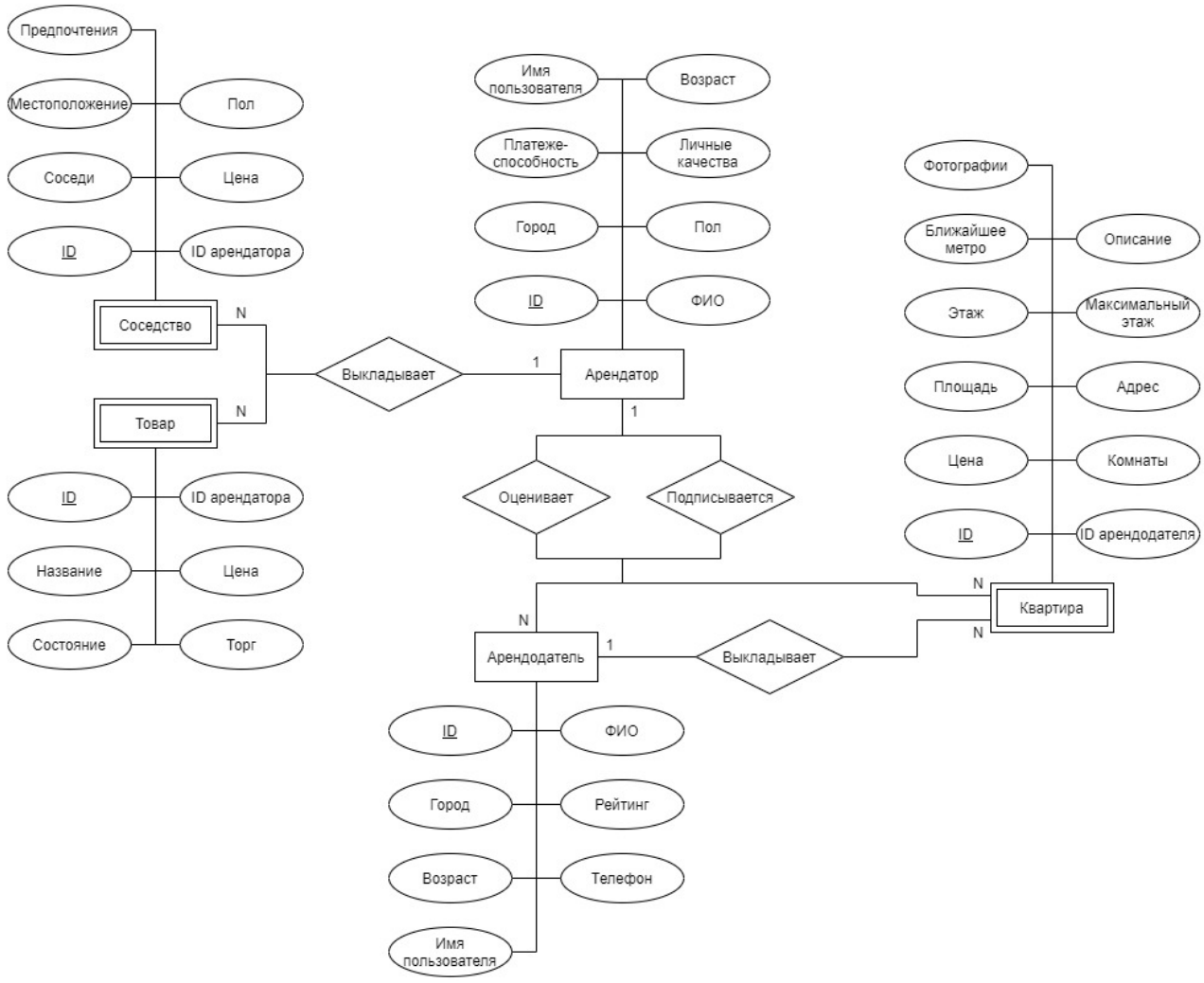

Рисунок 1.5 – ER-модель разрабатываемой БД в нотации Чена

1.2 Модель данных

Модель данных – это совокупность структур данных и операций их обработки [3]. Выделяют три основные типа данных моделей.

1. Дореляционная модель – появилась одной из первых. Она предшествуют реляционной и в свою очередь делится на иерархическую и сетевую модели:
   - иерархическая модель – совокупность элементов, расположенных в порядке их подчинения от общего к частному и образующих по структуре дерево;
   - сетевая модель – модель, при которой каждый элемент может быть связан с любым другим элементом.
2. Реляционная модель – модель, в которой объекты и связи представляются в виде таблиц. Эта модель характеризуется простотой структуры данных и удобным табличным представлением. На данный момент является наиболее популярной моделью.
3. Постреляционная модель – представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Она допускает многозначные поля и поддерживает ассоциированные многозначные поля [4].

Реляционная модель является наиболее оптимальным выбором в данной работе. Исходя из составленной модели данных, очевидно, что табличное представление соответствует лучше всего, при этом видна необходимость целостности данных.

1.3 Существующие аналоги

В качестве аналогов рассмотрены два веб-сервиса, похожие по функционалу на разрабатываемый продукт: Quickl (https://thequickl.ru/) и Живем! (http://sosed.zhivem.ru/). Был проведен сравнительный анализ данных продуктов с разрабатываемым (ОбщагиНет) по следующим критериям:

- возможность поиска и добавления объявлений о сдаче квартир в аренду;
- возможность поиска и добавления объявлений о поиске соседа;

– возможность поиска и добавления объявлений о продаже бытовых товаров;
– подписка на арендодателей и появление объявлений о сдаче квартир в аренду по выставленным фильтрам;
– оценка арендодателей и квартир;
– получений уведомлений по подпискам.

Для удобства проведем краткий сравнительный анализ при помощи таблицы (таблица 1.1).

Таблиц 1.1 – Сравнительная таблица существующих аналогов с разрабатываемым сервисом

Критерий	**Quickl**	**Живем!**	***ОбщагиНет***
Сдача квартир в аренду	+	+	+
Поиск соседа	+	+	+
Продажа бытовых товаров	–	–	+
Подписка на арендодателей и квартиры	–	–	+
Оценка арендодателей и квартир	–	–	+
Получение уведомлений по подпискам	–	–	+

Как видно, из таблицы разрабатываемый продукт имеет ряд преимуществ, что делает обоснованной его разработку.

Вывод

В результате была поставлена задача на данную работу (определены тип сервиса, ролевая модель), выбрана реляционная модель данных, представлены Use-Case диаграмма и ER-модель. Также были проанализированы существующие аналоги и выявлены преимущества разрабатываемого сервиса по сравнению с другими.

2 Конструкторская часть

В данном разделе представлена диаграмма базы данных, описаны таблицы, приведены схемы триггеров.

2.1 Проектирование базы данных

В соответствии с ER-моделью, представленной на рисунке 1.5, база данных должна хранить следующие таблицы:

- таблица арендаторов (tenant);
- таблица арендодателей (landlord);
- таблица квартир (flat);
- таблица фотографий квартир (flat_photo);
- таблица объявлений о соседстве (neighborhood);
- таблица товаров (goods);
- таблица подписок на арендодателей (subscription_landlord);
- таблица понравившихся квартир (likes_flat);
- таблица подписок на квартиры (subscription_flat);
- таблица станций метро, указанной в подписке на квартиру (subscription_metro).

На рисунке 2.1 представлена диаграмма разрабатываемой базы данных.

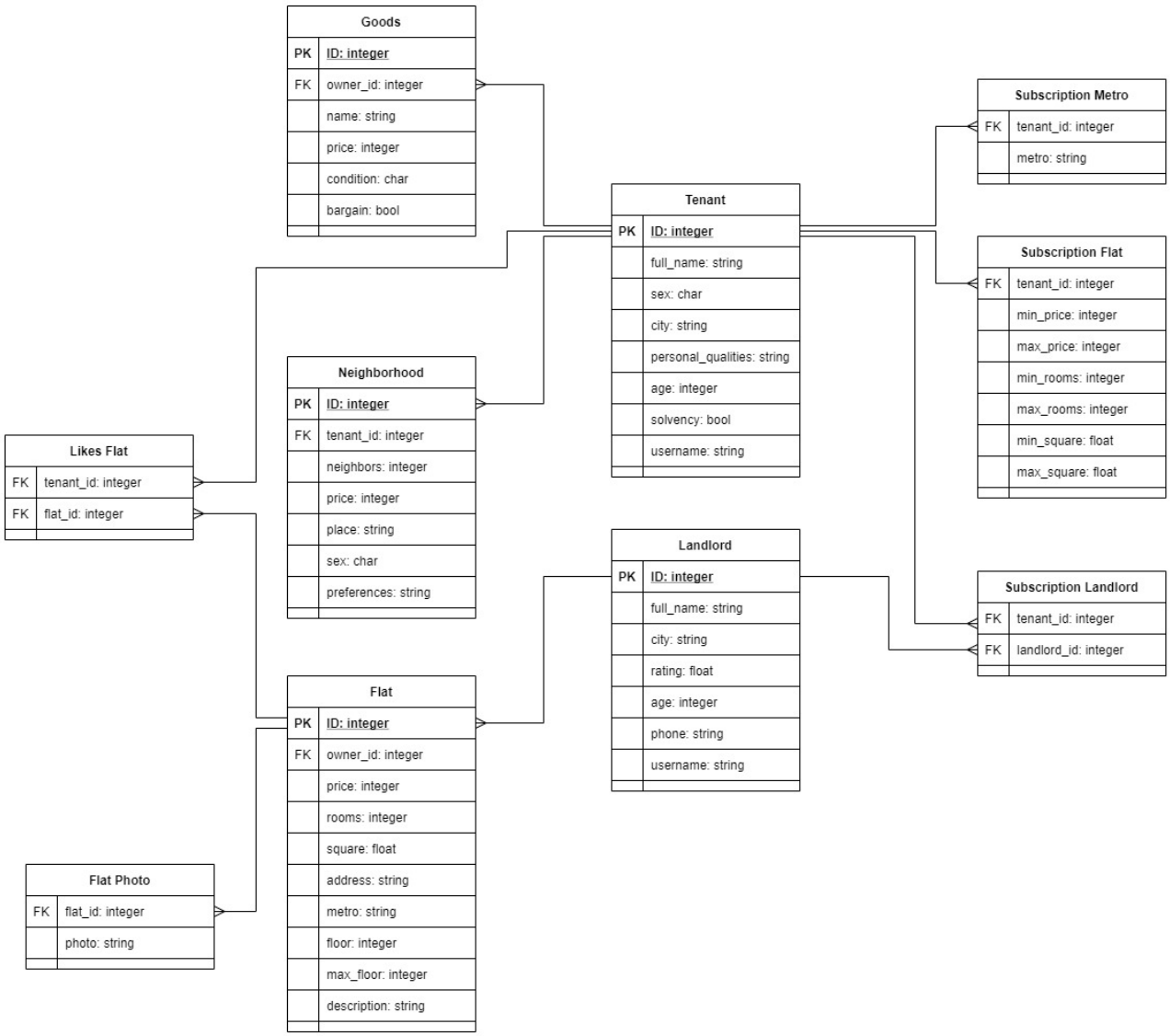

Рисунок 2.1 – Диаграмма базы данных

Таблица *tenant* хранит информацию об арендаторе и содержит следующие поля:

– ID – уникальный идентификатор арендатора, первичный ключ, integer;
– full_name – полное имя, string;
– sex – пол, принимает значения **M** или **F**, char;
– city – город, string;
– personal_qualities – персональные качества, string;
– age – возраст, integer;
– solvency – платежеспособность, bool;
– username – имя пользователя, string.

Таблица *landlord* хранит информацию об арендодателе и содержит следующие поля:

- ID – уникальный идентификатор арендодателя, первичный ключ, integer;
- full_name – полное имя, string;
- city – город, string;
- rating – рейтинг, float;
- age – возраст, integer;
- phone – номер телефона, string;
- username – имя пользователя, string.

Таблица *flat* хранит информацию о квартире и содержит следующие поля:

- ID – уникальный идентификатор квартиры, первичный ключ, integer;
- owner_id – id владельца (арендодателя), внешний ключ, integer;
- price – цена, integer;
- rooms – количество комнат, integer;
- square – площадь, float;
- address – адрес, string;
- metro – ближайшее метро, string;
- floor – этаж, integer;
- max_floor – максимальный этаж, integer;
- description – описание, string.

Таблица *flat_photo* хранит пути к фотографиям квартир и содержит следующие поля:

- flat_id – id квартиры, внешний ключ, integer;
- photo – путь к фотографии, string.

Таблица *neighborhood* хранит информацию об объявлениях о поиске соседа и содержит следующие поля:

- ID – уникальный идентификатор объявления, первичный ключ, integer;
- tenant_id – id арендатора, который ищет соседа, внешний ключ, integer;
- neighbors – количество соседей, integer;
- price – предполагаемая цена, integer;

– place – местоположение, string;
– sex – предпочитаемый пол, char;
– preferences – предпочтения, string.

Таблица *goods* хранит информацию о бытовых товарах и содержит следующие поля:

– ID – уникальный идентификатор товара, первичный ключ, integer;
– owner_id – id владельца (арендатора), внешний ключ, integer;
– name – название, string;
– price – цена, integer;
– condition – состояние, принимает значения **E** (отличное), **G** (хорошее), **S** (удовлетворительное), **U** (неудовлетворительное), **T** (ужасное), char;
– bargain – возможен ли торг, bool.

Таблица *goods* хранит информацию о бытовых товарах и содержит следующие поля:

– ID – уникальный идентификатор товара, первичный ключ, integer;
– owner_id – id владельца (арендатора), внешний ключ, integer;
– name – название, string;
– price – цена, integer;
– condition – состояние, принимает значения **E** (отличное), **G** (хорошее), **S** (удовлетворительное), **U** (неудовлетворительное), **T** (ужасное), char;
– bargain – возможен ли торг, bool.

Таблица *tenant* и *landlord* связаны отношением многие-ко-многим (подписка арендатором на арендодателей), таблица *subscription_landlord* хранит данную связь и содержит следующие поля:

– tenant_id – id арендатора, внешний ключ, integer;
– landlord_id – id арендодателя, внешний ключ, integer.

Таблица *tenant* и *flat* связаны отношением многие-ко-многим (арендаторы отмечают понравившуюся квартиру), таблица *likes_flat* хранит данную связь и содержит следующие поля:

– tenant_id – id арендатора, внешний ключ, integer;

– flat_id – id квартиры, внешний ключ, integer.

Таблица *subscription_flat* хранит информацию о параметрах квартиры, которые были указаны при подписке арендатором, и содержит следующие поля:

– tenant_id – id арендатора, внешний ключ, integer;
– min_price – минимальная цена, integer;
– max_price – максимальная цена, integer;
– min_rooms – минимальное количество комнат, integer;
– max_rooms – максимальное количество комнат, integer;
– min_square – минимальная площадь, float;
– max_square – максимальная площадь, float.

Таблица subscription_metro хранит информацию о станциях метро, указанных в подписке на квартиру, и содержит следующие поля:

– tenant_id – id арендатора, внешний ключ, integer;
– metro – станция метро, string.

2.2 Схемы триггеров

При удалении некоторых данных необходимо также удалять связанные с ними строки в других таблицах (внешние ключи). Для этого необходимы триггеры.

На рисунке 2.2 представлена схема триггера, срабатывающего после удаления арендатора.

Рисунок 2.2 – Схема триггера delete_tenant

На рисунке 2.3 представлена схема триггера, срабатывающего после удаления арендодателя.

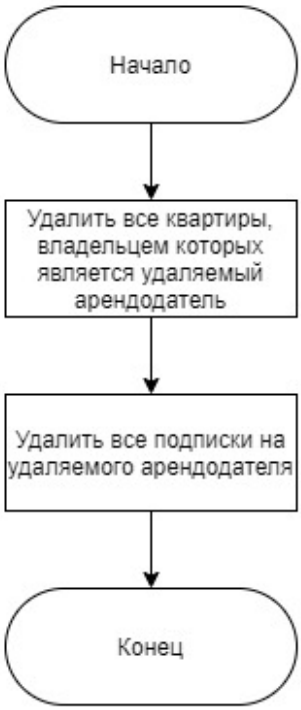

Рисунок 2.3 – Схема триггера delete_landlord

На рисунке 2.4 представлена схема триггера, срабатывающего после удаления квартиры.

Рисунок 2.4 – Схема триггера delete_flat

На рисунке 2.5 представлена схема триггера, срабатывающего после удаления подписки на квартиру.

Рисунок 2.5 – Схема триггера delete_subscription_flat

Также при добавлении новой подписки на квартиру необходимо, что старая удалялась. На рисунке 2.6 представлена схема соответствующего триггера.

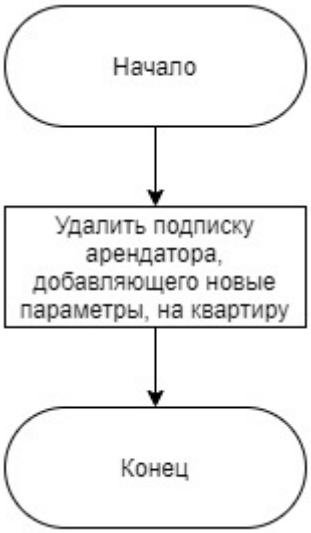

Рисунок 2.6 – Схема триггера insert_subscription_flat

Вывод

В данном разделена была спроектирована база данных и приведены схемы триггеров, срабатывающих при удалении и добавлении данных.

3 Технологическая часть

В данном разделе приведен анализ и выбор инструментов разработки, СУБД,

3.1 Выбор СУБД

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [5]. На основе выбранной реляционной модели данных рассмотрим наиболее популярные соответствующие СУБД.

3.1.1 MySQL

MySQL – это реляционная СУБД с открытым исходным кодом. В настоящее время это одна из наиболее популярных в веб-приложениях, почти все веб-фреймворки поддерживают MySQL уже на уровне базовой конфигурации (без дополнительных модулей) [6]. К преимуществам относятся простота в использовании, масштабируемость, скорость. Однако СУБД имеет определенные ограничения в функционале и недостаточную надежность [7].

3.1.2 PostgreSQL

PostgreSQL — это свободная объектно-реляционная СУБД, которая базируется на языке SQL и поддерживает многочисленные возможности. Она отличается высокой надёжностью и хорошей производительностью. PostgreSQL поддерживает транзакции, обладающие свойствами ACID, репликация реализована встроенными механизмами. Можно создавать свои типы данных

и индексов, а также расширять поведение при помощи языков программирования [8].

3.1.3 Oracle Database

Oracle Database — это объектно-реляционная СУБД, созданная компанией Oracle. Является наиболее популярной в мире. Она поддерживает множество функций, является очень надежной, может применяться практически для любых задач. Особенностью является быстрая работа с большими объемами данных. Однако стоимость пользования данной СУБД является довольно высокой, и работа с ней может требовать достаточного количества ресурсов.

Вывод

На основе анализа рассматриваемых СУБД был сделан выбор в пользу PostgreSQL. Благодаря большому количеству преимуществ, простоте и удобству она является наиболее оптимальным вариантом для разрабатываемого сервиса. Также с данной СУБД имеется опыт работы. При этом для PostgreSQL есть загружаемый процедурный язык PL/pgSQL, который позволяет довольно эффективно и просто писать функции и триггерные процедуры [10].

3.2 Выбор инструментов разработки

В качестве языка программирования был выбран Python [11].

Список использованных источников

1)