МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 Программное обеспечение информационной

безопасности мобильных систем

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных риэлторского агенства с реализацией технологии хранения мультимедийных данных»

Выполнил студент Барабанов Дмитрий Сергеевич

(Ф.И.О.)

Руководитель проекта преподаватель-стажер Комарова Е.И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: преподаватель-стажер Комарова Е.И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: преподаватель-стажер Комарова Е.И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2023

**Содержание**

[Введение 5](#_Toc134394596)

[1 Разбор аналогов и постановка задачи 6](#_Toc134394597)

[2 Разработка модели базы данных 8](#_Toc134394598)

[3 Разработка необходимых объектов 9](#_Toc134394599)

[3.1 Таблицы базы данных 9](#_Toc134394600)

[3.2 Процедуры базы данных 10](#_Toc134394601)

­

Введение

Целью данной работы являлась разработка реляционной базы данных для сайта риэлторского агентства. Эта база данных проектируется для обеспечения компании и клиентов информацией о доступных предложениях, реализовывающихся данным агентством. Так же было необходимо разработать соответствующее приложение, для демонстрации её работы.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана PostgreSQL, в связи с ее высокой производительностью и надежностью.

Так же было необходимо разработать приложение для демонстрации работы базы данных и взаимодействия с ней. Приложение было написано на языке JavaScript при помощи фреймворка ReactJS. Для взаимодействия с базой данных PostgreSQL используюется фреймворк ExpressJS.

В основной части будут затронуты все аспекты разработки проекта и обоснованы некоторые технические приёмы, к которым приходилось прибегнуть, с целью реализации работы приложения с базой данных.

Для демонстрации пользователям актуальных предложений в полной мере, в данном проекте была использована технология хранения мультимедийных типов данных в БД.

Основные требования к приложению:

* Реализация ролей администратора и рядового пользователя.
* Поиск недвижимости по типу, размеру жилой площади и количеству комнат.
* Подборка актуальных предложений для пользователя
* Взаимодействие с базой данных при помощи хранимых процедур.

В пояснительной записке вы сможете найти краткую информацию о похожих продуктах, архитектуре, реализации проекта, руководстве пользователя

1. Разбор аналогов и постановка задачи

Недвижимость играет огромную роль в жизни человека. Одной из главных ее функций является обеспечение комфортного жилья и объединение людей в социальных группах. Значимость недвижимости заключается в ее способности обогащать жизнь человека и общества в целом.

Изобретение интернета и развитие информационных технологий позволило перейти на следующий уровень инженерных разработок – риэлторских сайтов, которые стали играть важную роль в продвижении недвижимости и популяризации агентств. В начале 2000-х гг. риэлторские сайты затмили важность агентств. А сегодня многие компании используют базы данных и специализированные программы для удобного хранения информации и взаимодействия с клиентами.

Одним из основных трендом риэлторских приложений остается курируемый подбор объектов недвижимости. Это так называемые рекомендации, создающиеся с помощью использования сложных алгоритмов. В подборе объектов недвижимости участвуют профессиональные агенты – эксперты в области недвижимости, а также клиенты, которые уже работали с данным агентством.

При разработке своего приложения с обратил внимание на самые успешные сайты риэлторских агентств. Рассмотрим их ниже.

На сегодняшний день самой крупной риэлторской компанией в РБ является Realt BY. Интерфейс сайта можно увидеть на рисунке 1.1 ниже.



Рисунок 1.1 – Домашняя страница realt.by

На домашней странице нас привествует хорошо проработанный поиск, в котором можно выставить все необходимые параметры для начального поиска объекта. Также в шапке сайта имеются ссылки на все доступные разделы сайта.

Следующим рассматриваемым ресурсом является – Куфар Недвижимость. Сам Куфар – торговая площадка для размещения всех возможных товаров, как новых так и старых. Однако, Куфар Недвижимость вынесена как отдельный тематический ресурс. Домашнюю страницу ресурса можно увидеть на рисунке 1.2 ниже.

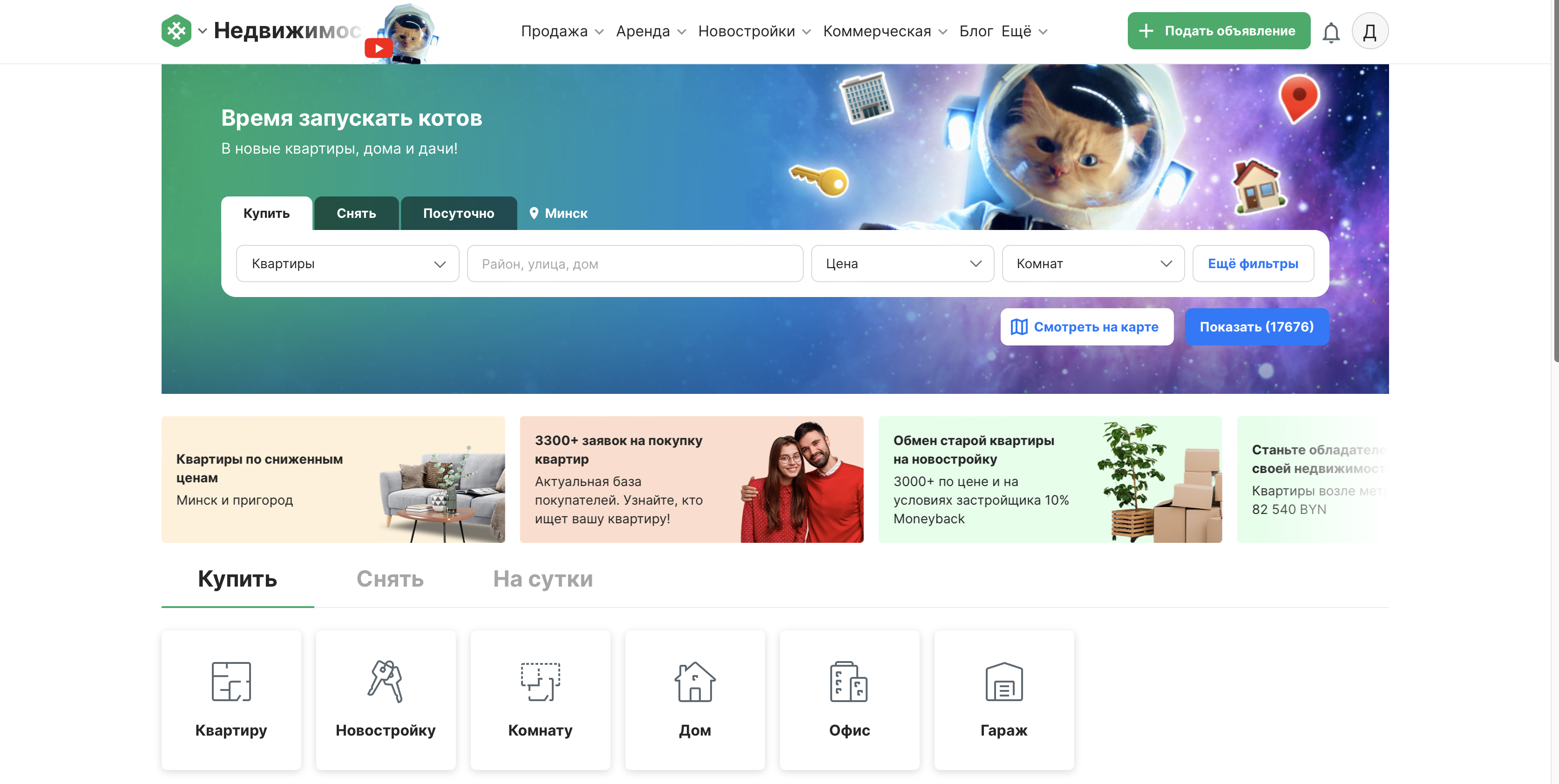


Рисунок 1.2 – Домашняя страница Куфар Недвижимости

На домашней странице также как и на сайте Realt.by нас встречает поиск с достаточным количеством параметров поиска для начального взаимодействия с ресурсом. Можно сходу выбрать тип недвижимости который вы ищете, будь то частный дом, квартира или офис. В отличии от ресурса Realt.by на домашней странице пользователя привлекает кнопка подачи объявления, что дает пользователю возможность не только искать недвижимость, но и реализовывать собственную.

Рассмотрев самые популярные тематические ресурсы в РБ, можно делать сформулировать общий функционал необходимый для работы ресурса.

Первым делом нужно реализовать корректную работу базы данных с файлами изображений, грамотно настроить зависимости и реализовать поиск по квадратуре, адресу и классу квартиры. Пользователь должен иметь возможность оставить заявку как на консультацию по определенной квартире, так и без указания конкретного объекта. Также пользователю должны предоставляться актуальные объекты в предназначенном для этого разделе.

1. Проектирование базы данных

Первым этапом курсового проекта будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуру базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

При выборе модели базы данных я решил использовать реляционную модель. Реляционная модель данных — это способ описания и хранения данных, который используется в базах данных. Она описывает данные в виде таблиц (также называемых реляциями), в которых строки соответствуют объектам, а столбцы - характеристикам этих объектов. Также реляционная модель данных определяет способы взаимосвязи между таблицами с помощью ключей (первичные и внешние).

Одним из главных преимуществ реляционной модели данных является ее функциональность и возможность использования стандартных языков запросов (SQL) для работы с данными. Это позволяет легко управлять базой данных и производить различные операции над ней, такие как вставка, обновление и удаление.

Диаграмма базы данных, спроектированной в ходе разработки представлена в приложении А.

При разработке приложение получило рабочее название «Приложение по поиску недвижимости», соответственно схема базы данных называется «Realtool». База данных, основанная на реляционной модели данных. В ее структуру входят следующие таблицы: USERS, REALTORS, RESIDENTIAL\_COMPLEXES, APARTMENTS, USER\_REQUESTS. Таблицы хранят в себе данные необходимые для описания и представления объектов недвижимости пользователям. А также информацию о риэлторах, которые их реализовывают. Ниже мы разберем каждую из них более подробно.

Разрабатывая модель базы данных, было взято во внимание соблюдений трёх первых нормальных форм, также были учтены вопросы производительности базы данных в рабочем процессе приложения или сайта, использующего её. Ещё одним важным вопрос является безопасность базы данных и разграничение доступа различных пользователей к данным – для этого были созданы различные пользователи базы данных, а доступ к таблицам был организован через функции и процедуры.

1. Разработка необходимых объектов

При разработке приложения для курсового проекта была использована база данных Postgres.

* 1. Таблицы базы данных

Для реализации базы данных риэлторского агентства было разработано 5 таблиц.

В структуру схемы базы данных курсового проекта входят следующие таблицы: USERS, REALTORS, RESIDENTIAL\_COMPLEXES, APARTMENTS, USER\_REQUESTS. Ниже мы разберем каждую из них более подробно.

Таблица USERS представляет собой перечень всех зарегистрированных пользователей, которые могут иметьразличные роли. Имеющиеся столбцы: ID (идентификатор пользователя, типа integer (\*), первичный ключ) , EMAIL (представляет email пользователя, тип TEXT), PASSWORD (служит для установки пароля пользователю, тип TEXT), ROLE (служит для задании роли пользователю, тип TEXT) и NAME (имя пользователя для авторизации, тип TEXT).

Таблица REALTORS представляет собой ссылки на пользователей, являющихся администраторами. Содержит в себе следующие столбцы: ID (служит для идентификации риэлтора, тип INTEGER) и USER\_ID (ссылается на пользователя которому выдана роль риэлтора, тип INTEGER).

Таблица RESIDENTIAL\_COMPLEXES служит для хранения всех жилых комплексов, которые представляет данная риэлторская компания. Имеет такие столбцы как: ID (идентификатор ЖК, тип INTEGER), NAME (название ЖК, тип TEXT), ADRESS (адрес расположения ЖК, тип TEXT) и REALTOR\_ID (идентификатор риэлтора, который закреплён за данным ЖК, тип INTEGER).

Таблица APARTMENTS является основной таблицей в БД, и выполняет функции хранения информации непосредственно о реализуемых компанией объектах недвижимости. Состоит из следующих столбцов: ID (идентификатор объекта, тип INTEGER), RESIDENTIONAL\_COMPLEX\_ID (идентификатор ЖК для привязки, тип INTEGER), NUMBER (номер квартиры в доме, тип INTEGER), FLOOR (этаж , тип INTEGER), ROOMS (количество комнат, тип INTEGER), PRICE (цена квартиры, тип INTEGER), DESCRIPTION (текстовое описание объекта, тип TEXT), PHOTO\_KEY (пути для мультимедиа, тип TEXT), REALTOR\_ID (идентификатор риэлтора, ответсвенного за объект, тип INTEGER), CREATED\_AT (дата добавления объекта в БД, тип TIMESTAMP).

Таблица CLIENT\_REQUESTS служит для хранения клиентских заявок. Состоит из следующих столбцов: ID (номер заявки, тип INTEGER), NAME (имя пользователя, тип TEXT), EMAIL (email пользователя, тип TEXT), PHONE (номер телефона пользователя для обратной связи, тип TEXT), APARTMENT\_ID (идентификатор объекта по поводу которого оставляется заявка, тип INTEGER), MESSAGE (тело сообщения пользователя, тип TEXT).

* 1. Процедуры базы данных

Для управления данными через приложение пользователи и администраторы использует хранимые процедуры. Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Написанные в ходе разработки курсового проекта процедуры можно разбить на несколько категорий:

* выборка данных из таблиц,
* добавление данных в таблицы,
* заполнение таблиц 100 000 строк,
* удаление данных из таблиц,
* изменение данных в таблицах,
* экспорт и импорт таблицы APARTAMENTS в JSON.
  + 1. Выборка данных из таблиц

Для вывода данных из таблиц были написаны следующие процедуры: GET\_RESIDENTIAL\_COMPLEXES,

GET\_APARTMENTS\_BY\_RESIDENTIAL\_COMPLEX\_NAME.

Добавление данных в таблицы

Для управления базой данных были созданы процедуры добавления новых строк в основные таблицы: ADD\_USER, ADD\_REALTOR, ADD\_RESIDENTIAL\_COMPLEX, ADD\_APARTMENT, ADD\_CLIENT\_REQUEST.

* + 1. Выборка данных из таблиц

Для получения данных из таблиц были написаны следующие процедуры: GET\_APARTMENTS, GET\_APARTMENTS\_BY\_RESIDENTIAL\_COMPLEX\_NAME, GOOD\_PRICE, GET\_APARTMENT\_BY\_ID и GET\_RESIDENTIAL\_COMPLEXES. Основной процедурой является GET\_APARTMENTS. Данная процедура позволяет приложению получить данные о всех апартаментах содержащихся в базе данных. GET\_APARTMENTS представлена на листинге 3.1 ниже.

CREATE OR REPLACE PROCEDURE get\_apartments()

LANGUAGE SQL

AS $$

SELECT \* FROM apartments ORDER BY price DESC;

$$;

Листинг 3.1 – процедура GET\_APARTMENTS

* + 1. Заполнение таблиц 100 000 строк

Для заполнения таблиц было разработана процедура INSERT\_100000\_LINES, заполняющая таблицу CLIENTS\_REQUESTS 100000 записями. Процедура представлена в листинге 3.2.

CREATE OR REPLACE PROCEDURE insert\_100000\_lines()

LANGUAGE plpgsql

AS $$

DECLARE

i INTEGER := 1;

BEGIN

WHILE i <= 100000 LOOP

INSERT INTO client\_requests (id, name, email, phone, apartment\_id, message)

VALUES (i, 'Name ' || i, 'email' || i || '@example.com', 'Phone ' || i, i, 'Message ' || i);

i := i + 1;

END LOOP;

END;

$$;

Листинг 3.2 – процедура INSERT\_100000\_LINES

* + 1. Добавление данных в таблицы

Для управления базой данных были созданы процедуры добавления новых строк в основные таблицы: ADD\_USER, ADD\_REALTOR, ADD\_RESIDENTIAL\_COMPLEX, ADD\_APARTMENTS, ADD\_CLIENT\_REQUEST.

* + 1. Удаление данных из таблиц

Для управления базой данных были созданы процедуры удаления: DELETE\_CLIENT\_REQUEST\_BY\_ID, DELETE\_APARTMENT\_BY\_ID, DELETE\_RESIDENTIAL\_COMPLEX\_BY\_NAME. Они были разработаны для удаления данных из таблиц соответственно: CLIENT\_REQUESTS, APARTMENTS, RESIDENTIAL\_COMPLEXES.

* 1. Индексы базы данных

Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет искать строки, удовлетворяющие критерию поиска.

На столбце RESIDENTIAL\_COMPLEX\_ID в таблице APARTMENTS был создан индекс для связи данной таблицы с таблицей RESIDENTIAL\_COMPLEXES. Данный индекс позволяет быстро находить и извлекать строки, относящиеся к определённому жилому комплексу. Индекс представлен в листинге 3.2 ниже.

CREATE INDEX idx\_apartments\_residential\_complex\_id ON apartments(residential\_complex\_id);

Листинг 3.2 – индекс IDX\_APARTMENTS\_RESIDENTIAL\_COMPLEX\_ID

1. Описание процедур импорта и экспорта

База данных имеет возможность экспортировать и импортировать данные для таблицы APARTMENTS. Для экспорта таблицы APARTMENTSв формате JSON была разработана процедура APARTMENTS \_TO\_JSON. Процедура APARTMENTS \_TO\_JSON представлена на листинге 4.1 ниже.

Таким образом происходит импорт и экспорт в базе данных.

CREATE OR REPLACE PROCEDURE export\_to\_json(p\_file\_path TEXT)

LANGUAGE plpgsql AS $$

DECLARE v\_json JSON;

BEGIN

SELECT json\_agg(t) INTO v\_json FROM (SELECT \* FROM apartments) t ;

PERFORM pg\_file\_write(p\_file\_path, v\_json::text, 'true');

END;

$$;

Листинг 4.1 – процедура EXPORT\_TO\_JSON

Для импорта данных в таблицу APARMENTS была разработана процедура IMPORT\_FROM\_JSON. Процедура IMPORT\_FROM\_JSON представлена в листинге 4.2 ниже.

create or REPLACE procedure import\_from\_json(OUT usersJson JSON) as $$

BEGIN

SELECT pg\_read\_file('/Users/glumber20033/project-x/Project-X/apartments.json') INTO usersJson;

INSERT into apartments (id,

residential\_complex\_id,

number,

floor,

rooms,

price,

description,

photo\_key,

realtor\_id,

created\_at) select "id","residential\_complex\_id","number","floor","rooms","price","description","photo\_key","realtor\_id","created\_at" from json\_populate\_recordset(null::apartments, usersJson);

raise notice 'Value: %', usersJson;

end;

$$ LANGUAGE plpgsql;

Листинг 4.2 – процедура IMPORT\_FROM\_JSON

1. Тестирование производительности

Производительность базы данных — это способность базы данных обрабатывать запросы и транзакции быстро и эффективно. Высокая производительность базы данных может быть достигнута с помощью эффективного использования индексов, хорошей организации структуры базы данных и оптимизации запросов. Хорошая производительность базы данных важна для обеспечения эффективной работы системы и удовлетворения потребностей пользователей.

Для тестирования производительности была взята процедура PERFOMANCE\_TEST, которая запрашивает нужное количество заявок относящихся к определенному клиенту. Процедура была вызвана со следующими параметрами: для пользователя с заданным именем запрашивалось 500 заявок, пропуская первые 900. Учитывая что в рамках приложения запрос идёт максимум на несколько заявок, подобные условия покажут производительность в крайне радикальных условиях. Процедура запроса показаны в листинге 5.1 ниже.

CREATE OR REPLACE PROCEDURE perfomance\_test()

LANGUAGE plpgsql

AS $$

BEGIN

SELECT \* FROM client\_requests WHERE name = 'Name Edgar' LIMIT 500 OFFSET 900;

END;

$$;

Листинг 5.1 – процедура PERFOMANCE\_TEST

Результат выполнения процедура PERFOMANCE\_TEST представлен на рисунке 5.1 ниже.



Рисунок 5.1 – результат тестирования процедуры PERFOMANCE\_TEST

Исходя из полученных данных можно сделать вывод, что тест производительности успешно пройден.

1. Описание технологии и ее применение в базе данных

В базе данных реализовано хранение мультимедийных данных. В таблице APARTMENTS в столбце PHOTO\_KEY хранится текстовая строка со ссылкой на нужную фотографию. Фрагмент данных таблицы хранящий ссылки на фото можно увидеть на рисунке 6.1.