МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА—Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Цифровые технологии обработки данных»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине «Программные средства манипулирования данными»

Тема курсового проекта (работы) «Разработка базы данных для гостиничного бизнеса»

Студент группы **Утц В.В., БСБО-04-20** _____
(*подпись студента*)

Студент группы **Ковалёв З.А., БСБО-04-20** _____
(*подпись студента*)

Студент группы **Чекунков А.В., БСБО-07-20** _____
(*подпись студента*)

Руководитель курсового проекта (работы) **Котилевец И.Д.** _____
(*подпись руководителя*)

Курсовой проект (работа) представлен(а) к защите «_____»_____20__г.

Допущен(а) к защите «_____»_____20__г.

Москва 2022 г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА—Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Цифровые технологии обработки данных»

Утверждаю
Заведующий кафедрой: _____

_____
(*подпись*) *(Ф.И.О.)*
«_____»_____20__г.

ЗАДАНИЕ
на выполнение курсового проекта (работы) по дисциплине
«Программные средства манипулирования данными»

Тема курсового проекта (работы) «Разработка базы данных для гостиничного бизнеса»
Студент Утц Владислав Владимирович Группа БСБО-04-20
Студент Ковалёв Захар Алексеевич Группа БСБО-04-20
Студент Чекунков Александр Владимирович Группа БСБО-07-20
Тема «Разработка базы данных для гостиничного бизнеса»
Исходные данные: нет
Перечень вопросов, подлежащих обработке, и обязательного графического материала:
-Титульный лист
-Содержание
-Введение
- Глава 1. Анализ предметной области
- Глава 2. Моделирование предметной области
- Глава 3. Разработка программного обеспечения
- Заключение
- Список использованной литературы

Срок предоставление к защите курсового проекта (работы) до «_____»_____20__г.
Задание на курсовую работу выдал _____ _____
(*подпись руководителя*) (*Ф.И.О. руководителя*)
«_____»_____20__г.
Задание на курсовую работу получил _____ _____
(*подпись обучающегося*) (*Ф.И.О. обучающегося*)

Содержание

1. Анализ предметной области

1.1.Введение

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения.

На сегодняшний день база данных является неотъемлемой частью любого приложения, поэтому требования к их грамотному проектированию и реализации становятся все выше.

Целью данной курсовой работы является проектирование и реализация базы данных в указанной предметной области. В качестве СУБД для реализации базы данных была использована СУБД реляционного типа – PostgreSQL.

В рамках курсовой работы были поставлены следующие задачи:

1. Анализ предметной области: сфера гостиничного бизнеса
2. Создание ER-диаграммы
3. Автоматизация функционала базы данных
4. Создание графического интерфейса пользователя

1.2. Общая характеристика

Целью сервиса, реализуемого для отеля, является автоматизация процесса заселения гостей. Функции сервиса организованы следующим образом: отель предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортабельностью и ценой. Клиентами отеля являются различные лица, о которых администратор собирает определенную информацию (фамилия, имя, отчество, паспортные данные). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При поселении фиксируется дата въезда, в системе,

номер отеля отмечается как занятый, при выезде гостя из отеля для каждого места запоминается дата освобождения.

1.3. Организационная структура

Сервис реализует бронирование номера клиентом и оформление гостя в номер администратором.

У администраторов и гостей должны быть разные права на доступ к базе данных, последствием чего является установка логина и пароля для администрации в отеле.

При работе с сервисом клиент имеет следующие возможности:

- Просмотр наличия свободных номеров в гостинице и их категория;
- Выбор нужного номера из свободных;
- Бронирование;
- Знание стоимости каждого номера в отдельности.

При работе с сервисом администратор должен принять и зарегистрировать новых клиентов в свободные номера, той категории, которая была выбрана.

1.4. Функциональная характеристика

Бронирование номера:

Входные данные:

- Дата заселения
- Дата выезда
- Категория
- Вид из окна
- ФИО клиента
- Паспортные данные клиента
- Номер телефона клиента

Выходные данные:

- Свободные номера

Алгоритм обработки:

После ввода желаемых дат проживания, проверяется наличие свободных номеров, клиенту отображается их список. После выбора ввода своих данных, гость выбирает желаемый номер.

Заселение:

Входные данные:

- Дата заселения
- Дата выезда (предполагаемая)
- Забронированный номер
- Паспортные данные клиента
- Доп. услуги

Выходные данные:

- Номер
- Даты проживания

Алгоритм обработки:

Администратор вносит паспортные данные клиента в сервис, они сохраняются в базе данных, после оплаты, номер транзакции также добавляется в БД. При оплате гостю выдается чек, в котором указано номер квитанции, имя, фамилия гостя, код номера, количество дней в номере, сумма и администратор, оформляющий регистрацию.

Выезд:

Входные данные:

- Дата выезда
- Освобождающийся номер

Выходные данные:

- Номер
- Данные об оплате

Алгоритм обработки:

Администратор указывает в системе, что номер освободился, в базе данных меняется его состояние.

2. Модель предметной области

2.1. Диаграмма модели предметной области

Для построения диаграммы модели предметной области типа «Сущность-Связь» было использовано ПО «ER-Master». Сама диаграмма представлена на Рис. 1

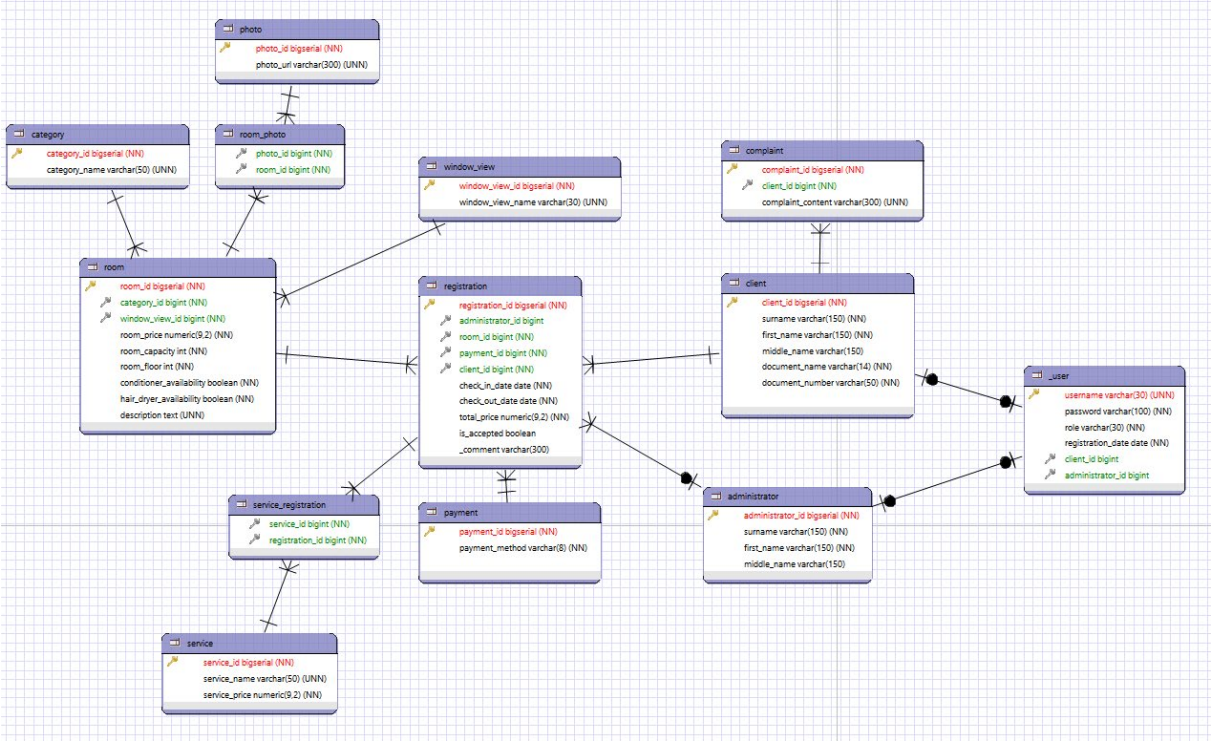

Рис. 1 – Диаграмма модели предметной области типа «Сущность - Связь».

2.2. Спецификация модели предметной области

Спецификация модели предметной области включает следующие: спецификация сущностей, спецификация атрибутов, спецификация связей.

2.2.1. Спецификация сущностей

Спецификация сущностей представлена в таблице 1.

Название сущности	**Назначение сущности**
Room	Информация о номере (категория, стоимость за ночь, вместимость, вид из окна,

	наличие фена, кондиционера, описание)
Category	Информация о категории (эконом, комфорт, люкс и т. д.)
Client	Информация о клиенте (паспортные данные, ФИО)
Administrator	ФИО
Payment	Способ оплаты
Complaint	Содержит информацию о жалобе от клиента
Registration	Содержит даты заселения/выезда, итоговую стоимость,
Service	Содержит название и стоимость дополнительных услуг
Photo	Содержит URL фотографий номеров
User	Содержит логин, пароль (зашифрованный) и роль пользователей приложения
Window view	Содержит названия видов из окна

Таблица 1. – Спецификация сущностей.

2.2.2. Спецификация атрибутов

Спецификация атрибутов представлена в таблице 2.

Сущность «Room»				
Имя атрибута	**Назначение атрибута**	**Тип атрибута**	**Формат атрибута**	**Допустимые значения**
room_id	Первичный ключ	bigint	число	

category_id	Содержит категорию номера	bigint	число	
room_floor	Содержит информацию на каком этаж находится номер	int	число	
conditioner_availability	Содержит информацию о наличии кондиционера	boolean		
hair_dryer_availability	Содержит информацию о наличии фена	boolean		
window_view_id	Содержит вид из окна	bigint	число	
room_capacity	Содержит информацию о вместимости	int	число	
description	Содержит описание номера	text	символы	латиница, кириллица, дефис, пробел
room_price	Содержит информацию о стоимость за ночь	numeric(9,2)	число	
Сущность «Category»				
Имя атрибута	**Назначение атрибута**	**Тип атрибута**	**Формат атрибута**	**Допустимые значения**
category_id	Первичный ключ	bigint	число	

category_name	Содержит название категории	varchar (50)	символы	латиница, кириллица, дефис, пробел
Сущность «Registration»				
Имя атрибута	**Назначение атрибута**	**Тип атрибута**	**Формат атрибута**	**Допустимые значения**
registration_id	Первичный ключ	bigint	число	
room_id	Содержит id номера	bigint	число	
administrator_id	Содержит id администратора	bigint	число	Положительные числа, null
client_id	Содержит id клиента	bigint		
payment_id	Содержит id способа оплаты	bigint	число	
check_in_date	Содержит дату заселения	date	дата	
check_out_date	Содержит дату выселения	date	дата	
total_price	Содержит итоговую стоимость проживания	numeric(9,2)	число	
is_accepted	Содержит информацию об	boolean		true, false, null

	обработке заявки администратором			
_comment	Содержит комментария администратора после обработки заявки	varchar(300)	символы	латиница, кириллица, цифры, пробел, знаки препинания
Сущность «Client»				
Имя атрибута	**Назначение атрибута**	**Тип атрибута**	**Формат атрибута**	**Допустимые значения**
client_id	Первичный ключ	bigint	число	
surname	Содержит фамилию клиента	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
first_name	Содержит имя клиента	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
middle_name	Содержит отчество клиента	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
document_name	Содержит название	varchar (14)	символы	кириллица

	документа клиента			
document_number	Содержит номер документа клиента	varchar (50)	символы	латиница, кириллица, цифры, пробел, дефис
Сущность «Administrator»				
Имя атрибута	**Назначение атрибута**	**Тип атрибута**	**Формат атрибута**	**Допустимые значения**
administrator_id	Первичный ключ	bigint	число	
surname	Содержит фамилию администратора	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
first_name	Содержит имя администратора	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
middle_name	Содержит отчество администратора	varchar (150)	символы	латиница, кириллица, цифры, пробел, дефис
Сущность «Service»				
service_id	Первичный ключ	bigint	число	

service_name	Содержит название услуги	varchar (50)	символы	латиница, кириллица, дефис, пробел
service_price	Содержит информацию о стоимости услуги	numeric(9,2)	число	
Сущность «Complaint»				
complaint_id	Первичный ключ	bigint	число	
complaint_content	Содержит текст жалобы	varchar(300)	число	латиница, кириллица, цифры, пробел, знаки препинания
client_id	Содержит id гостя	bigint	число	
Сущность «Payment»				
payment_id	Первичный ключ	bigint	число	
payment_method	Содержит название способа оплаты	varchar (8)	символы	латиница, кириллица
Сущность «Photo»				
photo_id	Первичный ключ	bigint	число	
photo_url	Содержит URL фотографии	varchar (300)	символы	
Сущность «Window view»				
window_view_id	Первичный ключ	bigint	число	

window_view_name	Содержит название вида из окна	varchar (30)	символы	латиница, кириллица
Сущность «User»				
username	Первичный ключ	varchar(30)	число	латиница, цифры
password	Содержит пароль (хэш) пользователя	varchar(100)	символы	
role	Содержит роль пользователя в виде (ROLE_CLIENT)	varchar(30)	символы	
registration_date	Содержит дату регистрации пользователя	date	дата	
client_id	Содержит id клиента (если пользователь клиент)	bigint	число	Положительные числа, null
administrator_id	Содержит id администратора (если пользователь администраторё)	bigint	число	Положительные числа, null

Таблица 2. – Спецификация атрибутов.

2.2.3. Спецификация связей

Спецификация связей представлена в таблице 3.

Участвующие сущности	Кардинальность связи	Классы принадлежности
НОМЕР-ИМЕЕТ-КАТЕГОРИЯ	М:1	ОБ:Н/О
РЕГИСТРАЦИЯ - ВКЛЮЧАЕТ - НОМЕР	М:1	ОБ:ОБ
КЛИЕНТ-ОФОРМЛЯЕТ-РЕГИСТРАЦИЯ	1:М	ОБ:ОБ
КЛИЕНТ-ОСТАВЛЯЕТ-ЖАЛОБА	1:М	Н/О:ОБ
РЕГИСТРАЦИЯ-ВКЛЮЧАЕТ-УСЛУГИ	1:М	Н/О:Н/О

Таблица 3. – Спецификация связей.

2.3. Нормализация схем сущностей

Спецификация атрибутов отношений совпадает со спецификацией атрибутов сущностей модели предметной области.

2.4. Получение реляционных отношений из модели предметной области.

НОМЕР М ОБ ИМЕЕТ КАТЕГОРИЯ 1 Н/О	4	НОМЕР {**ID**, ID КАТЕГОРИИ, ВМЕСТИМОСТЬ, СТОИМОСТЬ (НОЧЬ), ВИД ИЗ ОКНА, ЭТАЖ, КОНДИЦИОНЕР, ФЕН} ПЕРВ. КЛЮЧ: **ID** ВНЕШН. КЛЮЧ: **ID КАТЕГОРИИ** КАТЕГОРИЯ {***ID***, НАИМЕНОВАНИЕ} ПЕРВ. КЛЮЧ: **ID**

РЕГИСТРАЦИЯ М ОБ ВКЛЮЧАЕТ НОМЕР 1 ОБ	4	РЕГИСТРАЦИЯ {**ID**, ID НОМЕРА, ДАТА ЗАСЕЛЕНИЯ, ДАТА ВЫЕЗДА, СТОИМОСТЬ, КОД ГОСТЯ)} ПЕРВ. КЛЮЧ: **ID** ВНЕШН. КЛЮЧ: **ID НОМЕРА** НОМЕР {**ID**, ID КАТЕГОРИИ, ВМЕСТИМОСТЬ, СТОИМОСТЬ (НОЧЬ), ВИД ИЗ ОКНА, ЭТАЖ, КОНДИЦИОНЕР, ФЕН} ПЕРВ. КЛЮЧ: **ID**
КЛИЕНТ М ОБ ОФОРМЛЯЕТ РЕГИСТРАЦИЯ М ОБ	6	КЛИЕНТ {**КОД Г ОСТЯ**, ID РЕГИСТРАЦИИ, ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, НАИМЕНОВАНИЕ ДОКУМЕНТА, НОМЕР ДОКУМЕНТА} ПЕРВ. КЛЮЧ: **КОД ГОСТЯ** ВНЕШ. КЛЮЧ: ID РЕГИСТРАЦИИ ОФОРМЛЯЕТ {КОД ГОСТЯ, ID РЕГИСТРАЦИИ} ПЕРВ. КЛЮЧ: **КОД ГОСТЯ, ID РЕГИСТРАЦИИ** ВНЕШ. КЛЮЧ: КОД ГОСТЯ, ID РЕГИСТРАЦИИ РЕГИСТРАЦИЯ {**ID**, ID НОМЕРА, ДАТА ЗАСЕЛЕНИЯ, ДАТА ВЫЕЗДА, СТОИМОСТЬ, КОД ГОСТЯ)} ПЕРВ. КЛЮЧ: **ID**

		ВНЕШ. КЛЮЧ: **КОД ГОСТЯ**
КЛИЕНТ 1 Н/О ОСТАВЛЯЕТ ЖАЛОБА М ОБ	4	КЛИЕНТ {**КОД ГОСТЯ**, ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, НАИМЕНОВАНИЕ ДОКУМЕНТА, НОМЕР ДОКУМЕНТА} ПЕРВ. КЛЮЧ: **КОД ГОСТЯ** ЖАЛОБА {**ID**, КОД ГОСТЯ, СОДЕРЖАНИЕ} ПЕРВ. КЛЮЧ: **ID** ВНЕШ. КЛЮЧ: **КОД ГОСТЯ**
РЕГИСТРАЦИЯ 1 Н/О ВКЛЮЧАЕТ УСЛУГИ М Н/О	4	РЕГИСТРАЦИЯ {**ID**, ID НОМЕРА, ДАТА ЗАСЕЛЕНИЯ, ДАТА ВЫЕЗДА, СТОИМОСТЬ, ID БРОНИ, КОД ГОСТЯ)} ПЕРВ. КЛЮЧ: **ID** УСЛУГИ {**ID**, НАИМЕНОВАНИЕ, СТОИМОСТЬ} ПЕРВ. КЛЮЧ: **ID**

Таблица 4. Получения реляционных отношений.

2.5. Нормализация реляционных отношений.

1) **Отношение:** НОМЕР

НОМЕР {**ID**, ID КАТЕГОРИИ, ВМЕСТИМОСТЬ, СТОИМОСТЬ (НОЧЬ), ВИД ИЗ ОКНА, ЭТАЖ, КОНДИЦИОНЕР, ФЕН

ПЕРВ. КЛЮЧ: **ID**

ВНЕШН. КЛЮЧ: ID КАТЕГОРИИ

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

2) **Отношение:** КАТЕГОРИЯ

КАТЕГОРИЯ {**ID**, НАИМЕНОВАНИЕ}

ПЕРВ. КЛЮЧ: **ID**

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

3) **Отношение:** КЛИЕНТ

КЛИЕНТ {**КОД ГОСТЯ**, ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, НАИМЕНОВАНИЕ ДОКУМЕНТА, НОМЕР ДОКУМЕНТА}

ПЕРВ. КЛЮЧ: **КОД ГОСТЯ**

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

4) **Отношение:** АДМИНИСТРАТОР

АДМИНИСТРАТОР {**ID**, ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО}

ПЕРВ. КЛЮЧ: **ID**

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

5) **Отношение:** РЕГИСТРАЦИЯ

ОТЧЕТ {**ID**, ID НОМЕРА, КОД ГОСТЯ, ДАТА ЗАСЕЛЕНИЯ, ДАТА ВЫЕЗДА, СТОИМОСТЬ}

ПЕРВ. КЛЮЧ: **ID**

ВНЕШН. КЛЮЧ: ID НОМЕРА, КОД ГОСТЯ

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

6) **Отношение:** УСЛУГИ

УСЛУГИ {**ID**, НАИМЕНОВАНИЕ, СТОИМОСТЬ}

ПЕРВ. КЛЮЧ: **ID**

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

7) **Отношение:** ЖАЛОБА

ЖАЛОБА {**ID**, КОД ГОСТЯ, СОДЕРЖАНИЕ}

ПЕРВ. КЛЮЧ: **ID**

ВНЕШ. КЛЮЧ: КОД ГОСТЯ

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

8) **Отношение:** ОПЛАТА

ОПЛАТА {**ID**, КОД ГОСТЯ, ИТОГОВАЯ СУММА, СПОСОБ}

ПЕРВ. КЛЮЧ: **ID**

ВНЕШ. КЛЮЧИ: КОД ГОСТЯ

Находится в 1 НФ, так как каждый его элемент имеет атомарное значение

Находится во 2 НФ, так как первичный ключ не составной

Находится в 3 НФ, так как транзитивные замыкания отсутствуют

2.6. Спецификация реляционной модели данных

Спецификация реляционной модели данных совпадает со спецификацией модели «Сущность – связь».

3. Разработка программного обеспечения

Для разработки ПО были выбраны следующие средства:

- СУБД – PostgreSQL
- Backend – Kotlin + Spring Boot
- Frontend – Android приложение (Kotlin)

В качестве шаблона разработки используется микросервисная архитектура, общение сервисов обеспечено по REST.

3.1. Конфигурация базы данных

После анализа предметной области и создания Er диаграммы перейдём к созданию базы данных и реализации заявленного в требованиях к курсовой работе функционала. Элементарные запросы на добавление данных и выборки из таблиц реализованы на стороне backend'а приложения, подробное описание которого будет дано ниже в главе 3. Здесь приведён основной функционал, реализованный с помощью PostgreSQL.

Для создания таблиц воспользуемся функцией ERD tool программы PGADMIN, с помощью которой сгенерируем DDL скрипт, на основе имеющейся ER диаграммы.

Данный скрипт проводит первичный сброс таблиц, если они ранее существовали в базе данных. Далее с помощью оператора CREATE TABLE создаются таблицы в полном соответствии с ограничениями и содержащие все атрибуты, представленными на этапе проектирования в Главе 1. Кроме того создаются комментарии к каждой таблице для удобства в дальнейшей эксплуатации базы данных.

Полный код приведён в листинге ниже.

```
/* Drop Tables */
DROP TABLE IF EXISTS service_registration;
DROP TABLE IF EXISTS registration;
DROP TABLE IF EXISTS _user;
DROP TABLE IF EXISTS administrator;
DROP TABLE IF EXISTS room_photo;
DROP TABLE IF EXISTS room;
DROP TABLE IF EXISTS category;
DROP TABLE IF EXISTS complaint;
DROP TABLE IF EXISTS client;
DROP TABLE IF EXISTS payment;
DROP TABLE IF EXISTS photo;
```

```
DROP TABLE IF EXISTS service;
DROP TABLE IF EXISTS window_view;

/* Create Tables */

-- Таблица администраторов отеля
CREATE TABLE administrator
(
    -- Уникальный идентификатор администратора
    administrator_id bigserial NOT NULL,
    -- Фамилия администратора
    surname varchar(150) NOT NULL,
    -- Имя администратора
    first_name varchar(150) NOT NULL,
    -- Отчество администратора (необязательно)
    middle_name varchar(150),
    PRIMARY KEY (administrator_id)
) WITHOUT OIDS;


-- Таблица категорий номеров отеля
CREATE TABLE category
(
    -- Уникальный идентификатор категории номеров
    category_id bigserial NOT NULL,
    -- Название категории номеров
    category_name varchar(50) NOT NULL UNIQUE,
    PRIMARY KEY (category_id)
) WITHOUT OIDS;


-- Таблица гостей отеля
CREATE TABLE client
(
    -- Уникальный идентификатор гостя
    client_id bigserial NOT NULL,
    -- Фамилия гостя
    surname varchar(150) NOT NULL,
    -- Имя гостя
    first_name varchar(150) NOT NULL,
    -- Отчество гостя (при наличии)
    middle_name varchar(150),
    -- Наименование документа (ПАСПОРТ/ЗАГРАНПАСПОРТ)
    document_name varchar(14) NOT NULL,
    -- Нномер документа
    document_number varchar(50) NOT NULL,
    PRIMARY KEY (client_id)
) WITHOUT OIDS;


-- Таблица жалоб
CREATE TABLE complaint
(
    -- Уникальный идентификатор жалобы
    complaint_id bigserial NOT NULL,
    -- Уникальный идентификатор гостя
    client_id bigint NOT NULL,
    -- Текст жалобы
    complaint_content varchar(300) NOT NULL UNIQUE,
    PRIMARY KEY (complaint_id)
) WITHOUT OIDS;
```

```sql
-- Таблица оплат
CREATE TABLE payment
(
    -- Уникальный идентификатор оплаты
    payment_id bigserial NOT NULL,
    -- Способ оплаты (НАЛИЧНЫЕ/КАРТА)
    payment_method varchar(8) NOT NULL,
    PRIMARY KEY (payment_id)
) WITHOUT OIDS;


-- Таблица фотографий для номеров
CREATE TABLE photo
(
    -- Уникальный идентификатор фотографии
    --
    photo_id bigserial NOT NULL,
    -- URL фотографии
    --
    photo_url varchar(300) NOT NULL UNIQUE,
    PRIMARY KEY (photo_id)
) WITHOUT OIDS;


-- Таблица регистраций гостей
CREATE TABLE registration
(
    -- Уникальный идентификатор регистрации
    registration_id bigserial NOT NULL,
    -- Уникальный идентификатор администратора
    administrator_id bigint,
    -- Уникальный идентификатор номера
    room_id bigint NOT NULL,
    -- Уникальный идентификатор оплаты
    payment_id bigint NOT NULL,
    -- Уникальный идентификатор гостя
    client_id bigint NOT NULL,
    -- Дата заселения
    check_in_date date NOT NULL,
    -- Дата выезда
    check_out_date date NOT NULL,
    -- Итоговая стоимость проживания
    total_price numeric(9,2) NOT NULL,
    -- Поле, отображающее обработона ли заявка
    is_accepted boolean,
    -- Комментарий к заявке
    _comment varchar(300),
    PRIMARY KEY (registration_id)
) WITHOUT OIDS;


-- Таблица номеров отеля
CREATE TABLE room
(
    -- Уникальный идентификатор номера
    room_id bigserial NOT NULL,
    -- Уникальный идентификатор категории номеров
    category_id bigint NOT NULL,
    -- Уникальный идентификатор вида из окна
    window_view_id bigint NOT NULL,
    -- Стоимость номера за одну ночь в рублях
    room_price numeric(9,2) NOT NULL,
    -- Вместимость номера (кол-во человек)
```

```
    room_capacity int NOT NULL,
    -- Этаж, на котором располагается номер
    room_floor int NOT NULL,
    -- Наличие кондиционера (есть/нет)
    conditioner_availability boolean NOT NULL,
    -- Наличие фена (есть/нет)
    hair_dryer_availability boolean NOT NULL,
    -- Поле "описание комнаты"
    description text NOT NULL UNIQUE,
    PRIMARY KEY (room_id)
) WITHOUT OIDS;


CREATE TABLE room_photo
(
    -- Уникальный идентификатор фотографии
    --
    photo_id bigint NOT NULL,
    -- Уникальный идентификатор номера
    room_id bigint NOT NULL
) WITHOUT OIDS;


-- Таблица услуг, предоставляемых отелем
CREATE TABLE service
(
    -- Уникальный идентификатор услуги
    service_id bigserial NOT NULL,
    -- Название услуги
    service_name varchar(50) NOT NULL UNIQUE,
    -- Стоимость услуги
    service_price numeric(9,2) NOT NULL,
    PRIMARY KEY (service_id)
) WITHOUT OIDS;


CREATE TABLE service_registration
(
    -- Уникальный идентификатор услуги
    service_id bigint NOT NULL,
    -- Уникальный идентификатор регистрации
    registration_id bigint NOT NULL
) WITHOUT OIDS;


-- Таблица наименований видов из окон номеров
CREATE TABLE window_view
(
    -- Уникальный идентификатор вида из окна
    window_view_id bigserial NOT NULL,
    -- Название вида из окна
    window_view_name varchar(30) NOT NULL UNIQUE,
    PRIMARY KEY (window_view_id)
) WITHOUT OIDS;


CREATE TABLE _user
(
    -- Никнейм пользователя
    username varchar(30) NOT NULL UNIQUE,
    -- Пароль пользователя (хранится в зашифрованном виде)
    password varchar(100) NOT NULL,
    -- Роль пользователя (USER/ADMIN)
```

```
    role varchar(30) NOT NULL,
    -- Дата регистрации пользователя
    registration_date date NOT NULL,
    -- Уникальный идентификатор гостя
    client_id bigint,
    -- Уникальный идентификатор администратора
    administrator_id bigint,
    PRIMARY KEY (username)
) WITHOUT OIDS;



/* Create Foreign Keys */

ALTER TABLE registration
    ADD FOREIGN KEY (administrator_id)
    REFERENCES administrator (administrator_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE _user
    ADD FOREIGN KEY (administrator_id)
    REFERENCES administrator (administrator_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE room
    ADD FOREIGN KEY (category_id)
    REFERENCES category (category_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE complaint
    ADD FOREIGN KEY (client_id)
    REFERENCES client (client_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE registration
    ADD FOREIGN KEY (client_id)
    REFERENCES client (client_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE _user
    ADD FOREIGN KEY (client_id)
    REFERENCES client (client_id)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;


ALTER TABLE registration
```

```
   ADD FOREIGN KEY (payment_id)
   REFERENCES payment (payment_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE room_photo
   ADD FOREIGN KEY (photo_id)
   REFERENCES photo (photo_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE service_registration
   ADD FOREIGN KEY (registration_id)
   REFERENCES registration (registration_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE registration
   ADD FOREIGN KEY (room_id)
   REFERENCES room (room_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE room_photo
   ADD FOREIGN KEY (room_id)
   REFERENCES room (room_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE service_registration
   ADD FOREIGN KEY (service_id)
   REFERENCES service (service_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;


ALTER TABLE room
   ADD FOREIGN KEY (window_view_id)
   REFERENCES window_view (window_view_id)
   ON UPDATE RESTRICT
   ON DELETE RESTRICT
;

/* Comments */

COMMENT ON TABLE administrator IS 'Таблица администраторов отеля';
COMMENT ON COLUMN administrator.administrator_id IS 'Уникальный
идентификатор администратора';
COMMENT ON COLUMN administrator.surname IS 'Фамилия администратора';
COMMENT ON COLUMN administrator.first_name IS 'Имя администратора';
COMMENT ON COLUMN administrator.middle_name IS 'Отчество администратора
(необязательно)';
```

```
COMMENT ON TABLE category IS 'Таблица категорий номеров отеля';
COMMENT ON COLUMN category.category_id IS 'Уникальный идентификатор
категории номеров';
COMMENT ON COLUMN category.category_name IS 'Название категории номеров';
COMMENT ON TABLE client IS 'Таблица гостей отеля';
COMMENT ON COLUMN client.client_id IS 'Уникальный идентификатор гостя';
COMMENT ON COLUMN client.surname IS 'Фамилия гостя';
COMMENT ON COLUMN client.first_name IS 'Имя гостя';
COMMENT ON COLUMN client.middle_name IS 'Отчество гостя (при наличии)';
COMMENT ON COLUMN client.document_name IS 'Наименование документа
(ПАСПОРТ/ЗАГРАНПАСПОРТ)';
COMMENT ON COLUMN client.document_number IS 'Нномер документа';
COMMENT ON TABLE complaint IS 'Таблица жалоб';
COMMENT ON COLUMN complaint.complaint_id IS 'Уникальный идентификатор
жалобы';
COMMENT ON COLUMN complaint.client_id IS 'Уникальный идентификатор гостя';
COMMENT ON COLUMN complaint.complaint_content IS 'Текст жалобы';
COMMENT ON TABLE payment IS 'Таблица оплат';
COMMENT ON COLUMN payment.payment_id IS 'Уникальный идентификатор оплаты';
COMMENT ON COLUMN payment.payment_method IS 'Способ оплаты
(НАЛИЧНЫЕ/КАРТА)';
COMMENT ON TABLE photo IS 'Таблица фотографий для номеров';
COMMENT ON COLUMN photo.photo_id IS 'Уникальный идентификатор фотографии';
COMMENT ON COLUMN photo.photo_url IS 'URL фотографии';
COMMENT ON TABLE registration IS 'Таблица регистраций гостей';
COMMENT ON COLUMN registration.registration_id IS 'Уникальный идентификатор
регистрации';
COMMENT ON COLUMN registration.administrator_id IS 'Уникальный
идентификатор администратора';
COMMENT ON COLUMN registration.room_id IS 'Уникальный идентификатор
номера';
COMMENT ON COLUMN registration.payment_id IS 'Уникальный идентификатор
оплаты';
COMMENT ON COLUMN registration.client_id IS 'Уникальный идентификатор
гостя';
COMMENT ON COLUMN registration.check_in_date IS 'Дата заселения ';
COMMENT ON COLUMN registration.check_out_date IS 'Дата выезда';
COMMENT ON COLUMN registration.total_price IS 'Итоговая стоимость
проживания';
COMMENT ON COLUMN registration.is_accepted IS 'Поле, отображающее
обработона ли заявка';
COMMENT ON COLUMN registration._comment IS 'Комментарий к заявке';
COMMENT ON TABLE room IS 'Таблица номеров отеля';
COMMENT ON COLUMN room.room_id IS 'Уникальный идентификатор номера';
COMMENT ON COLUMN room.category_id IS 'Уникальный идентификатор категории
номеров';
COMMENT ON COLUMN room.window_view_id IS 'Уникальный идентификатор вида из
окна';
COMMENT ON COLUMN room.room_price IS 'Стоимость номера за одну ночь в
рублях';
COMMENT ON COLUMN room.room_capacity IS 'Вместимость номера (кол-во
человек)';
COMMENT ON COLUMN room.room_floor IS 'Этаж, на котором располагается
номер';
COMMENT ON COLUMN room.conditioner_availability IS 'Наличие кондиционера
(есть/нет)';
COMMENT ON COLUMN room.hair_dryer_availability IS 'Наличие фена
(есть/нет)';
COMMENT ON COLUMN room.description IS 'Поле "описание комнаты"';
COMMENT ON COLUMN room_photo.photo_id IS 'Уникальный идентификатор
фотографии';
COMMENT ON COLUMN room_photo.room_id IS 'Уникальный идентификатор номера';
COMMENT ON TABLE service IS 'Таблица услуг, предоставляемых отелем';
```

```
COMMENT ON COLUMN service.service_id IS 'Уникальный идентификатор услуги';
COMMENT ON COLUMN service.service_name IS 'Название услуги';
COMMENT ON COLUMN service.service_price IS 'Стоимость услуги';
COMMENT ON COLUMN service_registration.service_id IS 'Уникальный
идентификатор услуги';
COMMENT ON COLUMN service_registration.registration_id IS 'Уникальный
идентификатор регистрации';
COMMENT ON TABLE window_view IS 'Таблица наименований видов из окон
номеров';
COMMENT ON COLUMN window_view.window_view_id IS 'Уникальный идентификатор
вида из окна';
COMMENT ON COLUMN window_view.window_view_name IS 'Название вида из окна';
COMMENT ON COLUMN _user.username IS 'Никнейм пользователя';
COMMENT ON COLUMN _user.password IS 'Пароль пользователя (хранится в
зашифрованном виде)';
COMMENT ON COLUMN _user.role IS 'Роль пользователя (USER/ADMIN)';
COMMENT ON COLUMN _user.registration_date IS 'Дата регистрации
пользователя';
COMMENT ON COLUMN _user.client_id IS 'Уникальный идентификатор гостя';
COMMENT ON COLUMN _user.administrator_id IS 'Уникальный идентификатор
администратора';
```

Листинг 1. – Создание таблиц и связей между ними

Далее перейдём к наполнению таблиц тестовыми данными. Добавление данных реализовано с помощью оператора INSERT и VALUES для формирования набора констант.

```
--ЗАПОЛНЕНИЕ ТАБЛИЦ
INSERT INTO photo(photo_url)
values ('https://ex-terior.ru/wp-content/uploads/2018/05/otel-ritc-
karlton2-1024x623.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/otel-balchug-
kempinski-moskva1-e1525789527246.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/gostinica-
nacional1-1024x681.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/crowne-plaza-
moscow-world-trade-centre4-e1535452613493.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/palmira-biznes-
klub2-e1547325068855.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/crowne-plaza-
moscow-world-trade-centre6-e1535452604929.jpg'),
       ('https://ex-terior.ru/wp-content/uploads/2018/05/otel-katerina-
siti1-e1526037932223.jpg'),

('https://www.pogostite.ru/images/1280/960/0/admin/images/82/redisson_slavj
anskaja_otel_radisson_slavyanskaya_hotel_13.jpg');

insert into category(category_name)
values ('Люкс'),
       ('Полулюкс'),
       ('Комфорт');

insert into window_view (window_view_name)
values ('Кремль'),
       ('река Москва');

insert into room(category_id, window_view_id, room_price, room_capacity,
room_floor, conditioner_availability,
                 hair_dryer_availability, description)
```

```
values (1, 1, 10000.00, 3, 15, true, true, 'Наш лучший номер с видом на
Кремль'),
       (2, 2, 7500.00, 2, 9, false, true, 'Замечательный номер с видом на
реку Москва'),
       (3, 2, 5000.00, 1, 4, false, true, 'Отличный номер с видом на реку
Москва');

insert into payment (payment_method)
values ('НАЛИЧНЫЕ'),
       ('КАРТА');

insert into service (service_name, service_price)
values ('Завтрак', 1000.00),
       ('Ужин', 1500.00),
       ('Уборка номера', 700.00);

insert into room_photo (photo_id, room_id)
values (1, 1),
       (2, 1),
       (3, 1),
       (4, 2),
       (5, 2),
       (6, 2),
       (7, 3),
       (8, 3);

insert into administrator (surname, first_name, middle_name)
values ('Петров', 'Петр', 'Петрович'),
       ('Иванов', 'Иван', 'Иванович');

insert into _user (username, password, role, registration_date, client_id,
administrator_id)
values ('koval', 'qwertyQWERTY12345$', 'ROLE_SUPER_ADMIN', '2022-12-12',
null, null),
       ('petrov', 'qwertyQWERTY12345$', 'ROLE_ADMINISTRATOR', '2022-12-12',
null, 1),
       ('ivanov', 'qwertyQWERTY12345$', 'ROLE_ADMINISTRATOR', '2022-12-12',
null, 2);
```

Листинг 2. – Заполнение таблиц тестовыми данными

Хранение пароля в незашифрованном виде является прямой угрозой безопасности базы данных, поэтому добавим триггер на хэширование пароля.

Для этого создадим так называемую триггерную процедуру hash_password, тип возвращаемого значения которой – триггер, который срабатывает после операции insert в таблицу users. Для непосредственного хэширования паролей применяется расширение pgcrypto. Кроме того, при срабатывании функции хэширования генерируется так называемая “соль” - строка данных, которая передаётся хеш-функции вместе с входным массивом данных для вычисления хэша. Используется для усложнения

определения прообраза хэш-функции методом перебора по словарю возможных входных значений, включая атаки с использованием радужных таблиц. Для фильтрации данных, которые необходимо захэшировать используется переменная NEW, которая содержит новую строку базы данных для команд INSERT/DELETE в триггерах уровня строки. Логика триггерной процедуры следующая – после операции insert для строки таблицы _user, где содержится новый пароль выполняется команда update, которая вместо незашифрованного значения пароля устанавливает результат работы хэш функции.

Листинг создания триггерной процедуры и непосредственно триггера приведён ниже.

```
--ФУНКЦИЯ ХЭШИРОВАНИЯ ПАРОЛЯ
CREATE FUNCTION hash_password() RETURNS trigger AS
$hash_pass$ -- триггер
BEGIN
    CREATE EXTENSION IF NOT EXISTS pgcrypto;
    update _user set password = crypt(new.password, gen_salt('bf', 8))
WHERE (username = new.username);
    RETURN new;
END
$hash_pass$ LANGUAGE plpgsql;

CREATE EXTENSION IF NOT EXISTS pgcrypto;

--ТРИГГЕР НА ВСТАВКУ В _user
CREATE TRIGGER hash_pass
    after INSERT
    on _user
    for each ROW
EXECUTE PROCEDURE hash_password();
```

Листинг 3. – Триггер на хэширование пароля в таблице _user

При выборке с большой селективностью для повышения производительности необходимо создавать индексы. Приведём листинг создания индекса на photo_id.

Для добавления тестовых данных была реализована функция make_random_photos, которая с помощью цикла генерирует 500 записей в таблицу photos с помощью конкатенации строки с URL фотографии и номера итерации цикла. Таким образом в таблице будет содержаться пять сотен ссылок на фотографии, а для каждой страницы номера необходимо

будет выбрать лишь небольшую часть из них, следовательно, селективность выборки будет высокой, а значит созданный индекс уместен.

```
--ФУНКЦИЯ ДОБАВЛЕНИЯ 500 РАНДОМНЫХ ФОТО
CREATE OR REPLACE FUNCTION make_random_photos()
    RETURNS int AS
$$
DECLARE
    url          VARCHAR;
    photos_count integer;
BEGIN
    photos_count := 0;
    url := 'https://photo';

    FOR i IN 1..500
        LOOP
            INSERT INTO photo (photo_url)
            VALUES (url || photos_count || '.ru');
            photos_count := photos_count + 1;
        END LOOP;
    RETURN photos_count;
END;
$$ LANGUAGE plpgsql;

SELECT make_random_photos();

--ИНДЕКС НА photo.photo_id
create index photo_f on photo (photo_id);
```

Листинг 4. – Создание индекса

В соответствии с логикой работы системы необходимо реализовать процедуру по бесплатному добавлению всех возможных дополнительных услуг в номер категории люкс. Такая процедура должна обладать свойствами атомарности, изоляции и целостности. Соответственно данная процедура была реализована с применение транзакции.

Логика процедуры следующая: с помощью переменной типа record и циклической конструкции производится перебор и добавление всех возможных дополнительных услуг в таблицу service_registration, которая отвечает за хранение информации о приобретенных услугах. В случае если созданный до начала транзакции счетчик, который инкрементируется на каждом шаге цикла имеет такое же значение, как и счетчик общего количества дополнительных услуг транзакция фиксируется, иначе происходит откат данной транзакции.

Полный код процедуры приведен в следующем листинге.

```
--ПРОЦЕДУРА С ТРАНЗАКЦИЕЙ НА БЕСПЛАТНОЕ ДОБАВЛЕНИЕ ВСЕХ ДОП. УСЛУГ, ЕСЛИ
НОМЕР ЛЮКС
CREATE PROCEDURE add_services_to_lux_transaction(reg_id bigint)
    LANGUAGE plpgsql
AS
$$
declare
    c       record;
    counter integer;
BEGIN
    counter := 0;
    SAVEPOINT my_savepoint;
    FOR c IN SELECT service_id FROM service
        LOOP
            INSERT INTO service_registration(service_id, registration_id)
VALUES (c.service_id, reg_id);
            counter := counter + 1;
        END LOOP;
    if counter = (select count(service_name) from service) then
        COMMIT;
    else
        rollback to savepoint my_savepoint;
    end if;
END;
$$;
```

Листинг 5. – Создание транзакции

В качестве дополнительного функционала реализована функция для автоматической выгрузки отчёта о работе того или иного администратора в формате csv. Данная функция принимает на вход следующие аргументы: id администратора, по которому необходимо заполнить отчёт, начальное и конечное время периода за который строится отчёт и путь в системе, куда необходимо сохранить csv файл. Далее выполняются SQL запросы, с помощью которых создаётся выборка интересующих данных, а именно – общее количество сданных номеров, количество сданных люксовых номеров и общая выручка, которую получила гостиница от данного администратора за указанный период. Для расчётов применяются агрегатные функции COUNT и SUM, для соединения данных из разных таблиц оператор JOIN. Полный листинг функции приведён ниже.

```
--ФУНКЦИЯ СОХРАНЕНИЯ СТАТИСТИКИ В CSV
CREATE OR REPLACE FUNCTION save_admin_statistic_to_csv(id bigint,
                                                       start_period date,
                                                       end_period date,
                                                       filepath text)
RETURNS void
AS
$$
DECLARE
```

```
    all_checkins_count INTEGER; -- Общее кол-во проведённых заселений
    lux_count          INTEGER; -- Кол-во номеров категории люкс, которые
продал сотрудник
    total_revenue      NUMERIC(9, 2); -- Тотальная выручка
    statement          TEXT;
BEGIN
    all_checkins_count := (SELECT COUNT(*)
                           FROM registration
                                    inner join administrator
                                               on
registration.administrator_id = administrator.administrator_id
                           where ((administrator.administrator_id = id) and
                                  (registration.check_in_date >
start_period) and
                                  (registration.check_in_date <
end_period)));
    lux_count := (SELECT COUNT(*)
                  FROM registration
                           JOIN room on registration.room_id = room.room_id
                           join administrator on
registration.administrator_id = administrator.administrator_id
                  WHERE (room.category_id = 1)
                    and (administrator.administrator_id = id)
                    and (registration.check_in_date > start_period)
                    and (registration.check_in_date < end_period));
    total_revenue := (SELECT SUM(total_price)
                      FROM registration
                               inner join administrator
                                          on registration.administrator_id
= administrator.administrator_id
                      where ((administrator.administrator_id = id) and
                             (registration.check_in_date > start_period)
and
                             (registration.check_in_date < end_period)));
    if total_revenue is null then
        total_revenue = 0.00;
    end if;

    statement := format('copy (SELECT %s AS all_checkins_count, %s AS
lux_count, %s AS total_revenue)
to ''%s/admin_statistics.csv'' with csv
header;
',
                        all_checkins_count, lux_count, total_revenue,
filepath);
    EXECUTE statement;
END;
$$
    LANGUAGE plpgsql;
```

Листинг 6. – Выгрузка отчёта

Для автоматизации операции увеличения цены за проживание, если имуществу гостиницы нанесён какой-то урон реализована хранимая процедура bigger_price, которая принимает на вход данные о id регистрации и сумме причиненного ущерба, затем с помощью оператора update устанавливает новое значения поля total_price, прибавляя к уже

хранящемуся там значению сумму причинного ущерба, для исключения ошибок, связанных с несовпадением типов данных используется явное приведение типов.

Полный листинг данной процедуры приведён ниже.

```
--ПРОЦЕДУРА С ТРАНЗАКЦИЕЙ НА БЕСПЛАТНОЕ ДОБАВЛЕНИЕ ВСЕХ ДОП. УСЛУГ, ЕСЛИ
НОМЕР ЛЮКС
CREATE PROCEDURE add_services_to_lux_transaction(reg_id bigint)
    LANGUAGE plpgsql
AS
$$
declare
    c       record;
    counter integer;
BEGIN
    counter := 0;
    FOR c IN SELECT service_id FROM service
        LOOP
            INSERT INTO service_registration(service_id, registration_id)
VALUES (c.service_id, reg_id);
            counter := counter + 1;
        END LOOP;
    if counter = (select count(service_name) from service) then
        COMMIT;
    else
        rollback;
    end if;
END;
$$;
```

Листинг 7. – Хранимая процедура bigger_price

Согласно бизнес-логике администратору необходимо понимать, какой сервис пользуется наибольшим спросом в конкретный период времени. Для более удобной работы с этими данными было создано представление top_service, которое подсчитывает сколько раз была заказана та или иная услуга и с помощью сортировки этих данных получает идентификатор и количество заказов услуги, которая пользуется наибольшим спросом.

Полный листинг данного представления приведён ниже.

```
--ПРЕДСТАВЛЕНИЕ "Топ 1 сервис"
CREATE view top_service AS
SELECT service_id, count(registration_id)
FROM service_registration
GROUP BY service_id
ORDER BY count(registration_id) DESC
limit 1;
```

Листинг 8. – Представление top_service

3.2. Графический интерфейс

Для реализации графического интерфейса необходимо разработать мобильное приложение для операционной системы Android на языке программирования Kotlin.

3.2.1. Авторизация и регистрация

При запуске приложения пользователя встречает экран авторизации, на котором необходимо ввести логин и пароль. Так же на данном экране присутствует кнопка, позволяющая перейти к экрану с регистрацией. На экране создания нового аккаунта находятся несколько EditText: имя, фамилия, отчество (при наличии), номер документа, логин и пароль; RadioButton: выбор типа документа (паспорт или заграничный паспорт). Экраны авторизации и регистрации изображены на Рисунке 2.

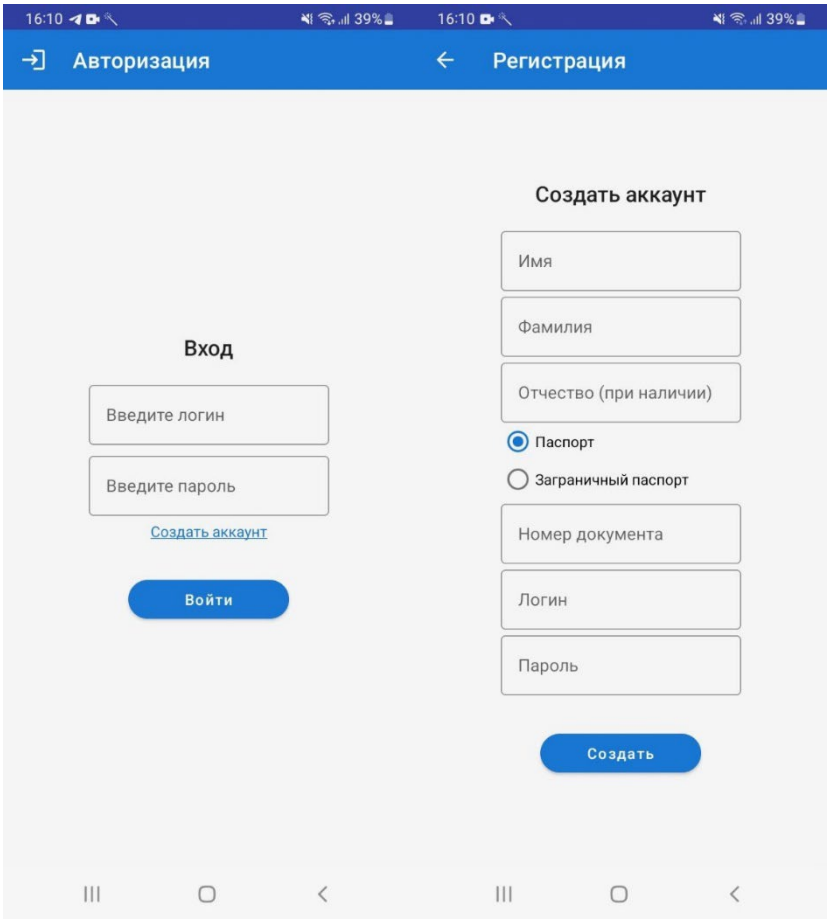

Рисунок 2. – Экраны авторизации и регистрации

3.2.2. Защита от SQL инъекций

Помимо вышеперечисленного функционала мобильное приложение берёт на себя часть функционала по защите от SQL инъекций. Помимо того, что на стороне СУБД реализовано разграничение полномочий с помощью ролей и политик, а основной функционал представлен на хранимых процедурах реализуется фильтрация данных на основе “черного списка”, таким образом пользовательские данные, содержащие специальные символы, которые используются в сценариях атаки типа SQL инъекция, никак не попадут в базу данных, тем самым исключая возможность такого вектора атаки.

Так же на данных экранах были реализованы проверки на корректность ввода данных в EditText: длина введенных данных и ввод запрещенных слов. Экраны авторизации и регистрации с сообщениями об ошибках изображены на Рисунке 3.

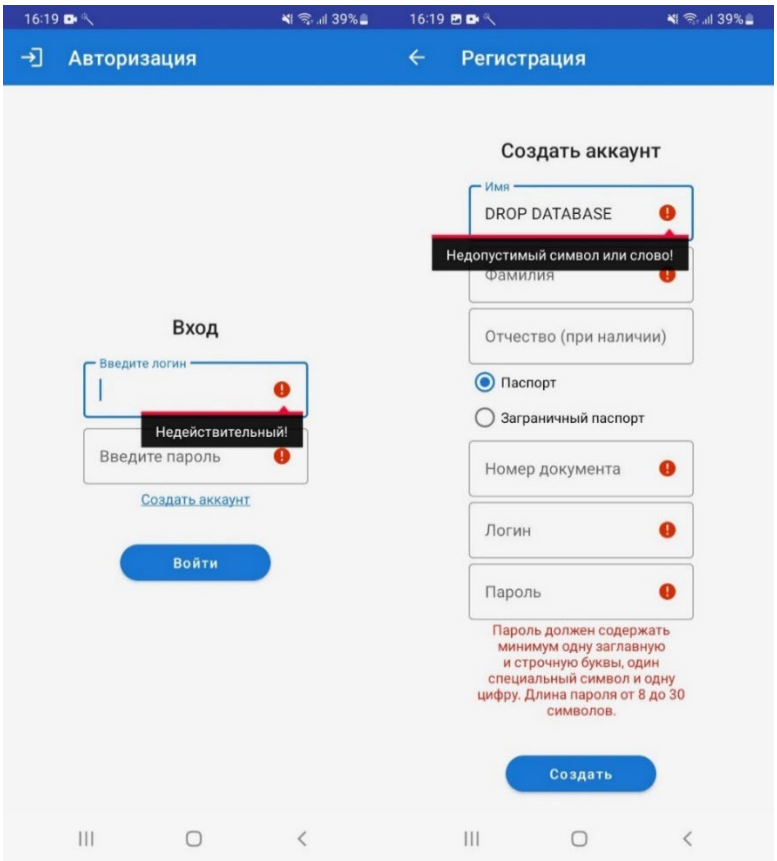

Рисунок 3. - Экраны авторизации и регистрации с сообщениями об ошибках

3.2.3. Основное меню

После того, как пользователь авторизовался открывается основное меню, на котором для каждой из ролей доступны свои действия. Основное меню для клиента, администратора и супер-администратора изображено на Рисунке 4.

Так же у каждой из ролей есть меню «Профиль» - Рисунок 5, где любой пользователь имеет возможность просмотреть свои данные: имя, фамилия, отчество (при наличии), роль, логин, номер документа и дату создания аккаунта.

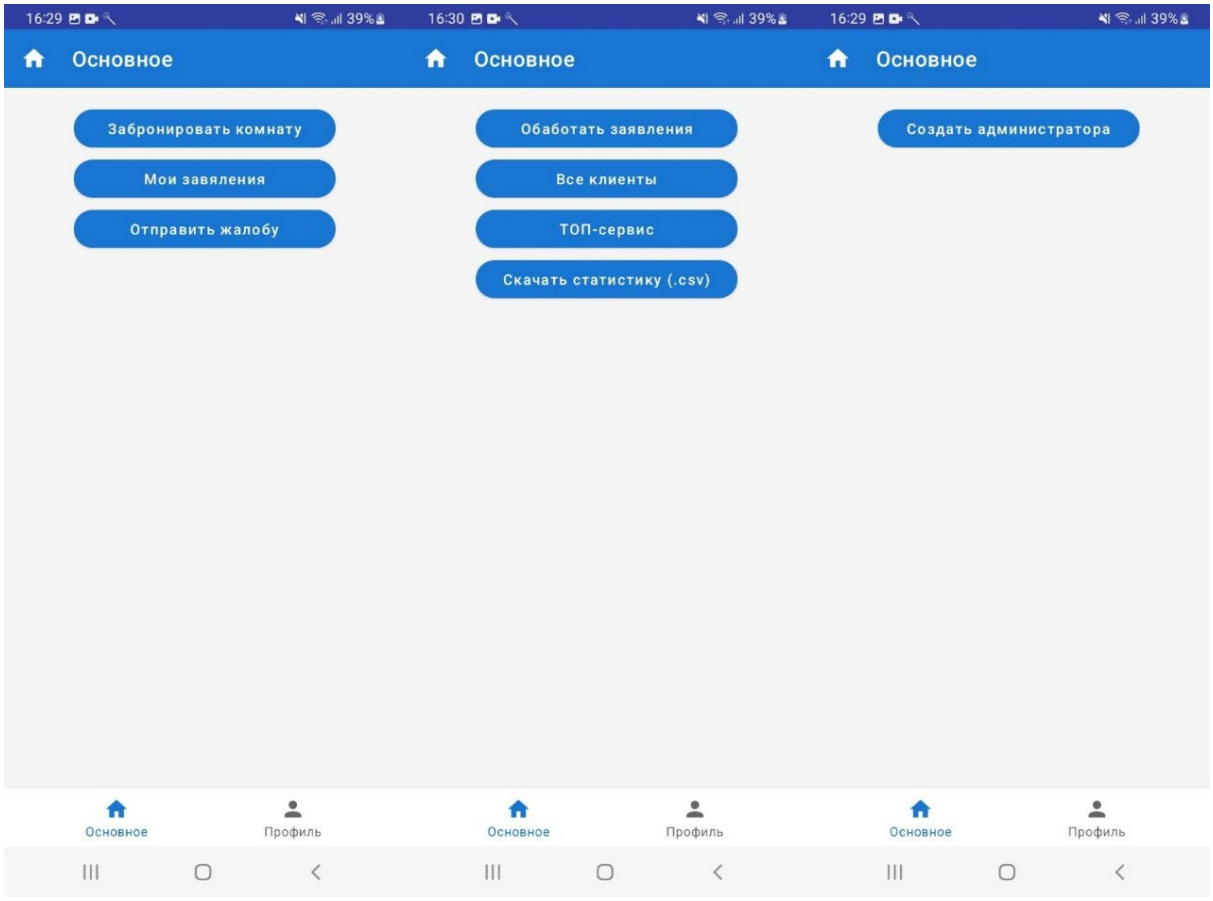

Рисунок 4. - Основное меню для клиента, администратора и супер-администратора

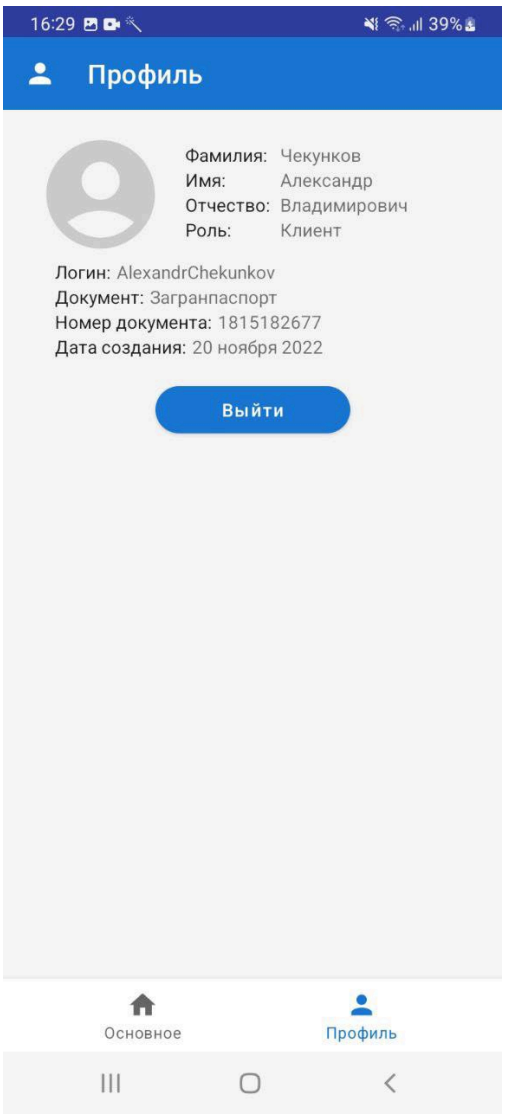

Рисунок 5. - Меню с информацией о пользователе

3.2.4. Функционал супер-администратора

В случае, если пользователь авторизовался как супер-администратор, в главном меню отображается доступное действие «Создать нового администратора». Данная функция позволяет заполнить данные о новом администраторе и добавить его в базу данных. Проверка корректности ввода данных реализована аналогично авторизации и регистрации. На Рисунке 6 изображен экран с добавлением нового администратора.

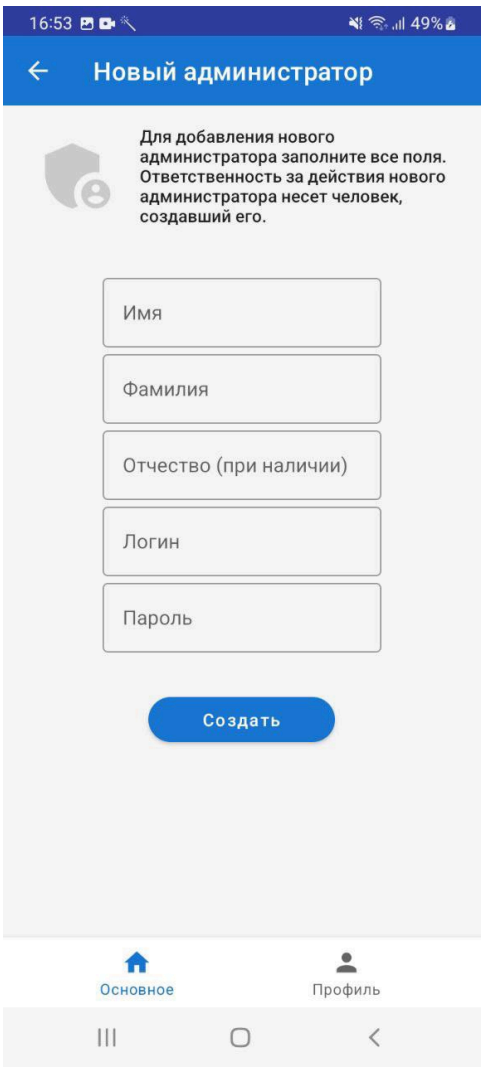

Рисунок 6. - Экран добавления нового администратора

3.2.5. Функционал клиента

В случае, если пользователь авторизовался как клиент, в главном меню отображаются следующие доступные действия: «Забронировать комнату», «Мои заявления» и «Отправить жалобу».

Функция «Отправить жалобу» позволяет клиенту заполнить сообщение с жалобой и отправить её. Проверка корректности ввода данных в сообщение реализована аналогично авторизации и регистрации. Экран отправки жалобы изображен на Рисунке 7.

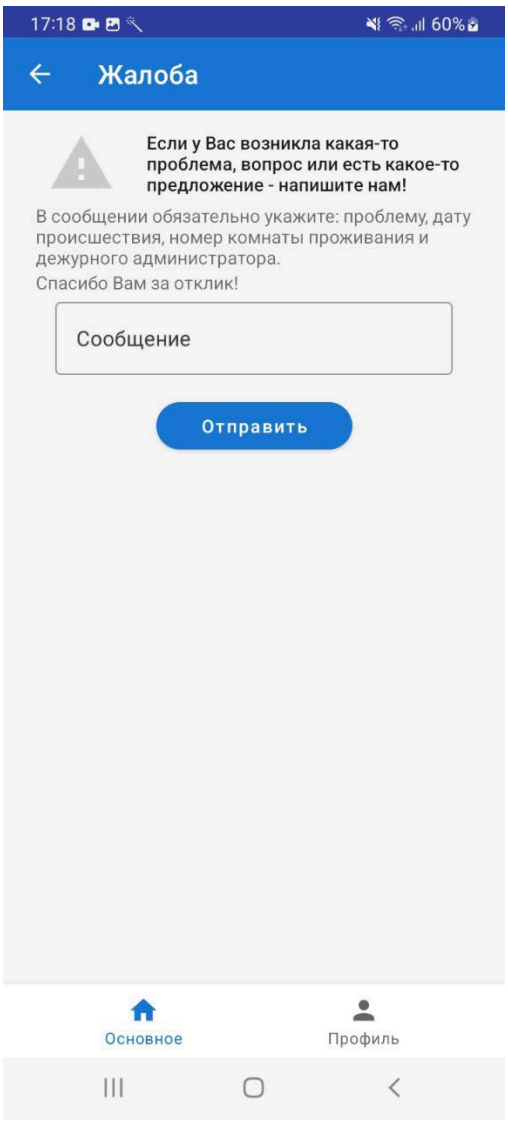

Рисунок 7. - Экран отправки жалобы клиентом

Функция «Забронировать комнату» позволяет клиенту оставить заявку на бронирование комнаты. При нажатии на кнопку действия открывается экран со списком всех доступных комнат. Каждый элемент списка хранит в себе статус комнаты (люкс, полулюкс и т.д.), уникальный номер комнаты, цену за ночь и вместимость.

При выборе комнаты, клиенту отображается новый экран со всей информацией о комнате (фотографии, статус, уникальный номер, описание, цена за ночь, вместимость, этаж, вид из окна, наличие кондиционера, наличие фена для волос), возможность выбрать способ оплаты (наличные при заселении или банковская карта), возможность добавить

дополнительные услуги (уборка, завтрак и т.д.) и возможность выбрать даты заселения/выселения.

После того, как клиент нажмет кнопку «Забронировать комнату» откроется экран с информацией о том, что заявка была принята и ожидает обработки администратором. На данном экране отображается статус заявки, номер забронированной комнаты, даты заселения/выселения, способ оплаты, итоговая стоимость и комментарий.

Экраны со списком всех комнат, информацией о выбранной комнате и информацией о заявке изображены на Рисунке 8.

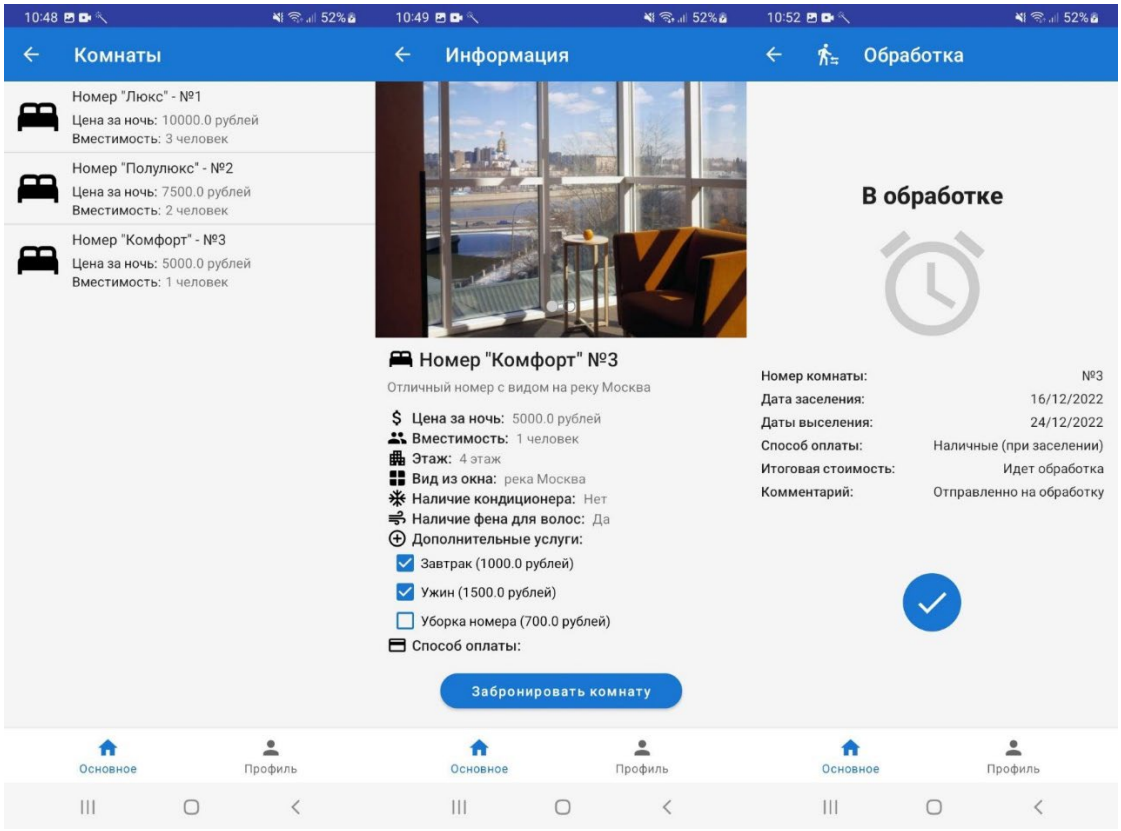

Рисунок 8. – Экраны со списком всех комнат, информацией о выбранной комнате и информацией о заявке

Функция «Мои заявления» позволяет клиенту просматривать список всех заявлений, которые он отправлял при бронировании комнаты. Каждый элемент списка хранит в себе статус комнаты и её уникальный номер, статус заявления и даты заселения – Рисунок 9.

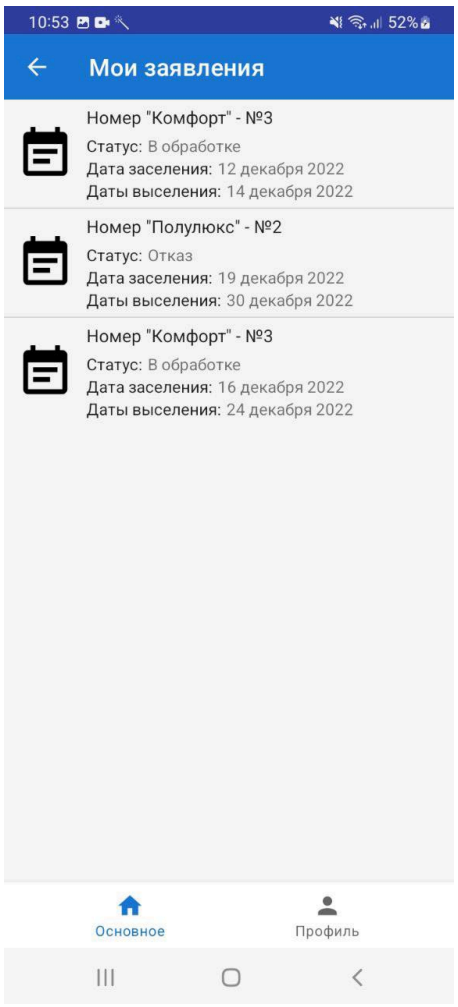

Рисунок 9. – Экран со списком всех заявлений клиента

Клиент имеет возможность просматривать всю информацию о выбранном заявлении. Для этого реализован экран, который хранит в себе такую же информацию, как экран заявление после заполнения данных при бронировании комнаты. Необходимо отметить, что статус заявление может быть следующим: «в обработке» (администратор еще не обработал заявление), «одобрено» (администратор одобрил заявление) и «отказ» (администратор отказал в заявлении и добавил комментарий). Экраны со статусом «в обработке», «одобрено» и «отказ» изображены на Рисунке 10.

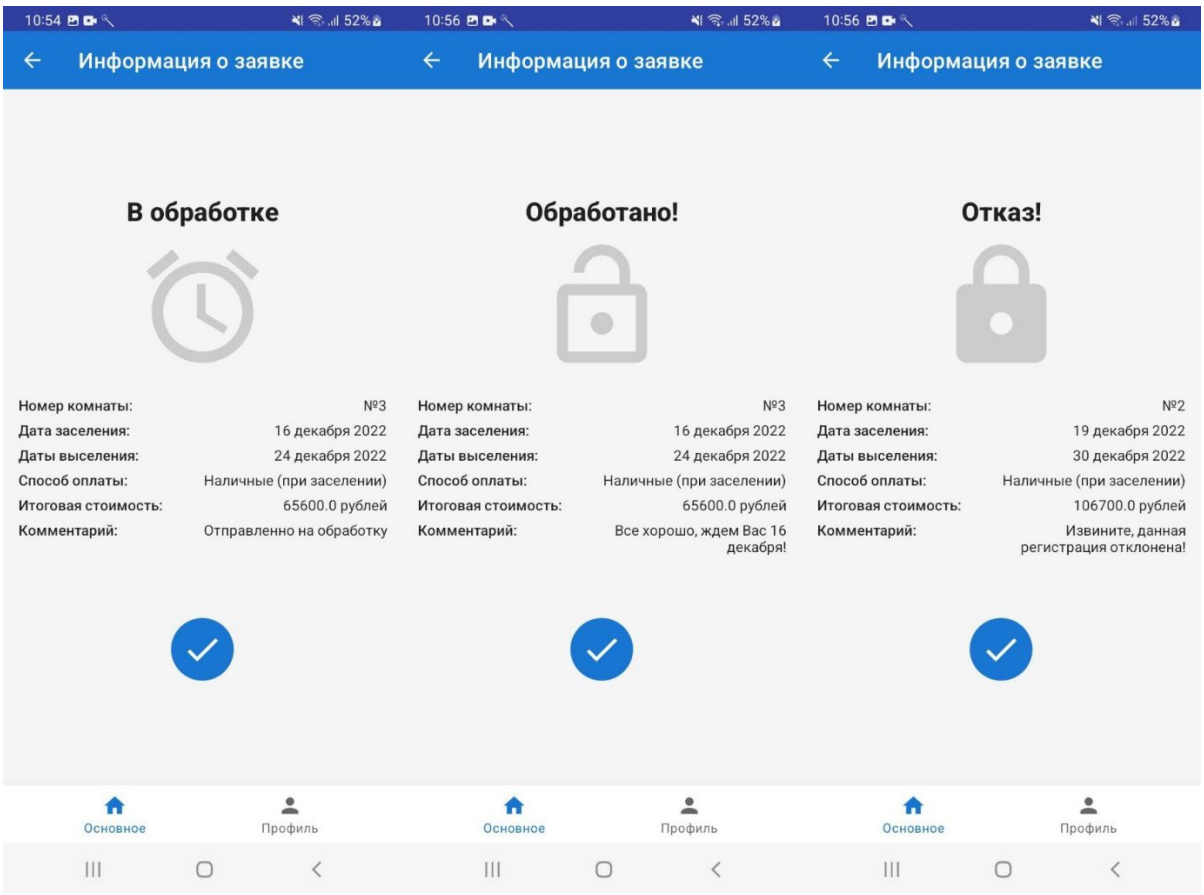

Рисунок 10. – Экраны со статусом «в обработке», «одобрено» и «отказ»

3.2.6. Функционал администратора

В случае, если пользователь авторизовался как администратор, в главном меню отображаются следующие доступные действия: «Все клиенты», «Топ-сервис», «Обработать заявления» и «Скачать статистику (.csv)».

Функция «Все клиенты» позволяет администратору открыть список со всеми существующими клиентами в системе. Каждый элемент списка хранит в себе информацию о каждом клиенте, а именно фамилию, имя, отчество, номер документа и роль. Экран со списком всех клиентов изображен на Рисунке 11.

Функция «Топ-сервис» даёт возможность администратору посмотреть на самый популярный сервис (завтрак, уборка т.д.), который заказывают клиенты. На экране Топ-сервиса изображены следующие данные – название, цена и количество заказов – Рисунок 12.

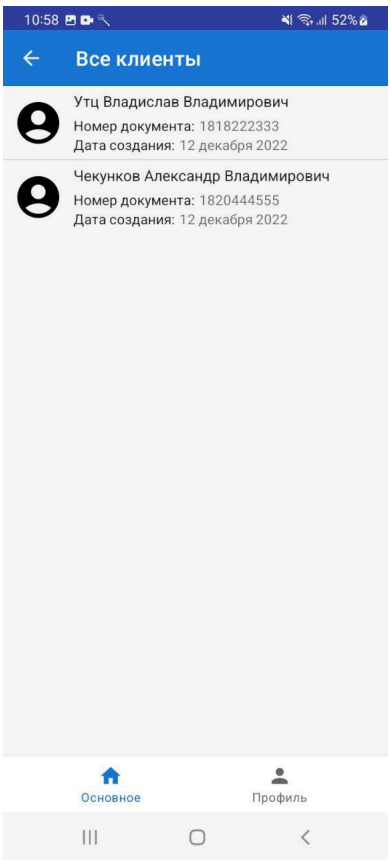

Рисунок 11. – Экран со списком всех клиентов в системе

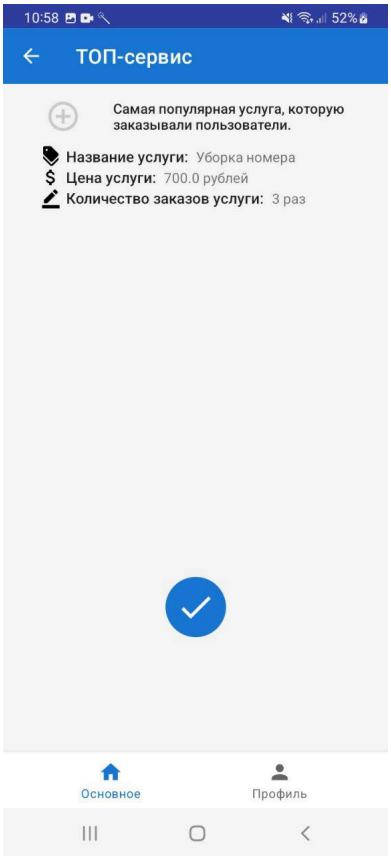

Рисунок 12. – Экран Топ-сервиса

Функция «Скачать статистику (.csv)» позволяет администратору скачать статистику по определенным датам. Для реализации данной функции используется WebView и Google Documents, которая позволяет открыть для просмотра файл-csv. Экран с отображением статистических данных изображен на Рисунке 13.

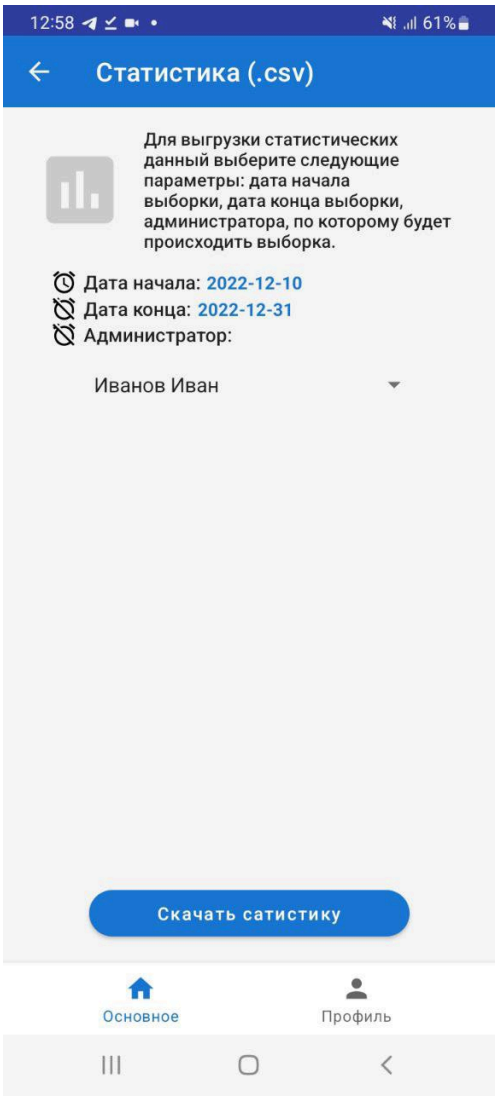

Рисунок 13. – Экран с отображением статистических данных

Функция «Обработать заявления» даёт возможность администратору обработать заявления клиентов. При нажатии на кнопку «Обработать заявления» открывается экран со списком всех заявок. Каждый элемент списка хранит информацию об уникальном номере и статусе комнаты, ФИО клиента, который оставил заявку, статус и финальную стоимость.

Если заявка в списке имеет статус «в обработке», администратор может открыть её и изменить статус на «одобрено» или «отказ», добавив комментарий.

Если заявка в списке имеет статус «отказ», администратор может открыть её и добавить сумму ущерба, которая может образоваться при возникновении каких-либо проблем со стороны клиента (разбитая посуда, сломанная вещь и т.д.).

Экраны со списком всех заявок, изменения статуса и добавления суммы ущерба изображены на Рисунке 14.

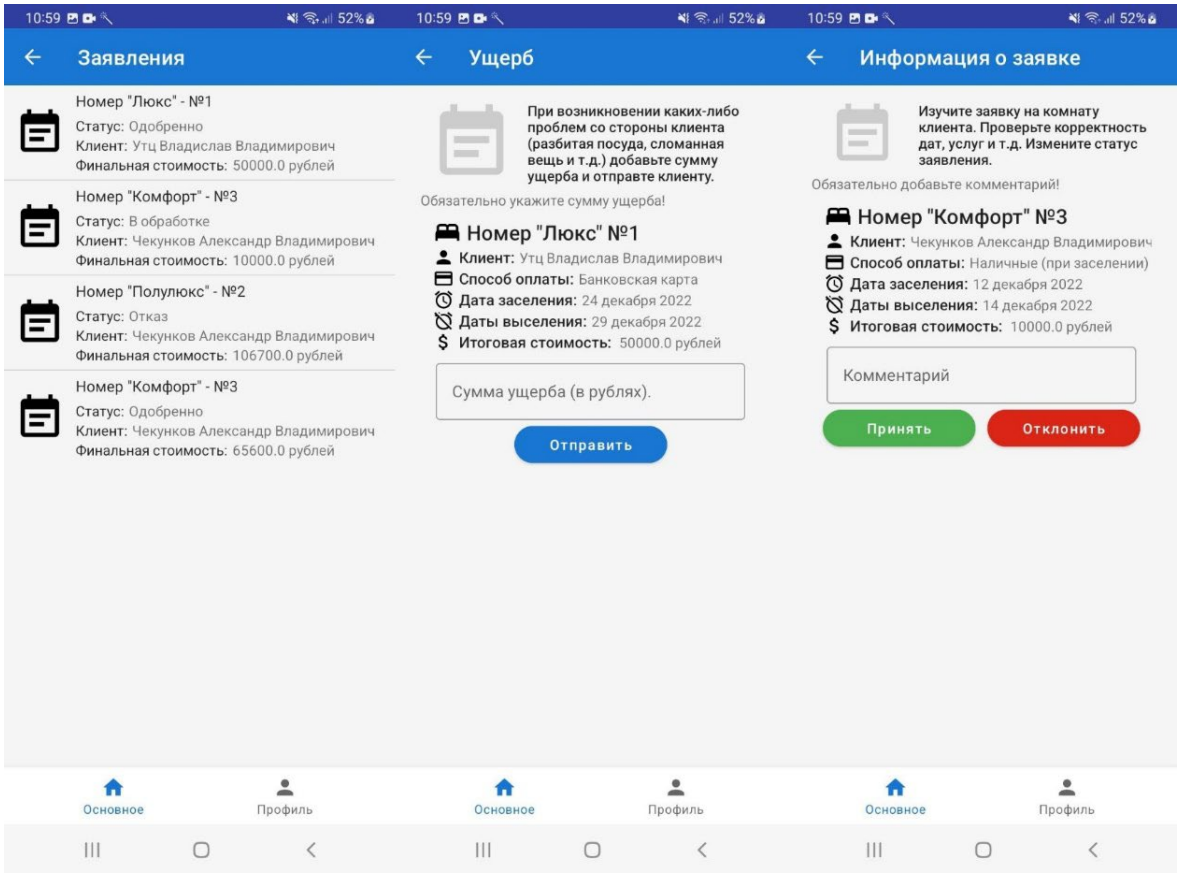

Рисунок 14. - Экраны со списком всех заявок, добавления суммы ущерба и изменения статуса

3.2.7. Дополнительные функции приложения

Дополнительно была реализована темная тема приложения, которая автоматически устанавливается при выборе системной темы на устройстве. Также в приложение был добавлен экран, сообщающий об отсутствии

доступа в интернет. На Рисунке 15 изображены два экрана информирующих пользователя об отсутствии интернета – один светлой темы и один, в качестве примера реализации, тёмной темы.

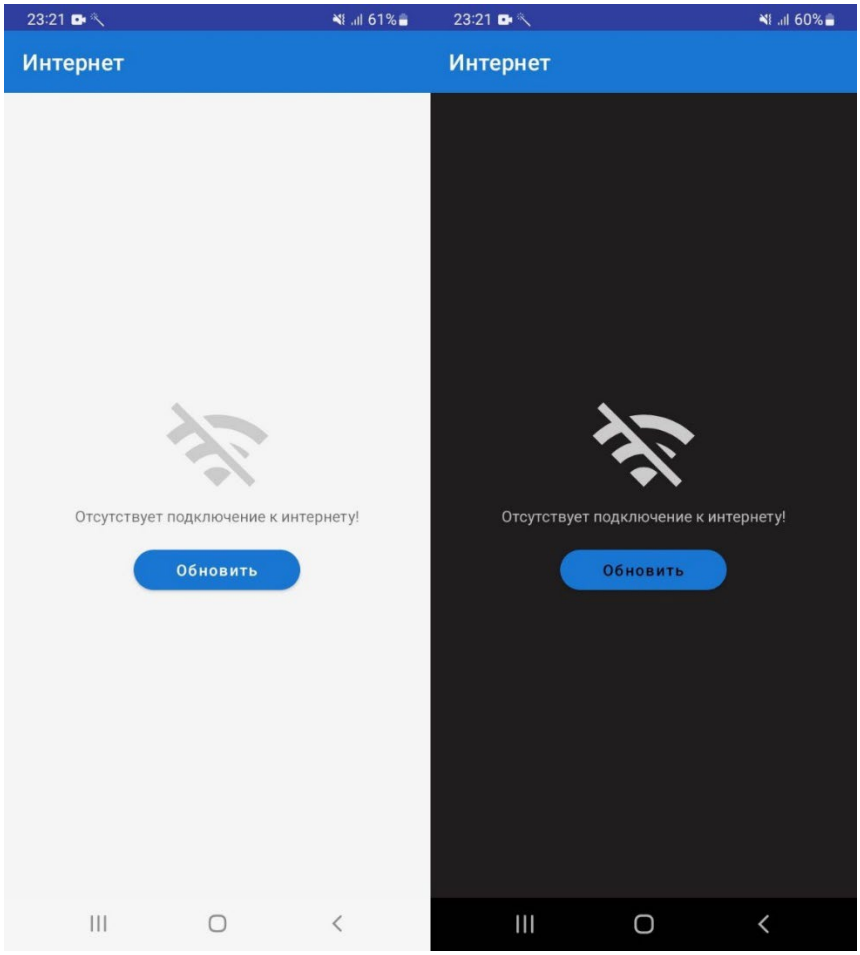

Рисунок 15. – Экран, сообщающий об отсутствии интернета на устройстве (светлая и тёмная тема)

В приложении были использованы такие сторонние библиотеки как Retrofit (для связи приложения с сервером), Gilde (для загрузки и отображения фотографий), CarouselView (для создания карусели фотографий), EmojiRain (для создания анимации падений эмодзи).

3.3. Серверная часть

Backend разрабатывался на языке Kotlin с использованием фреймворка Spring, а именно его модулей таких как: Spring Core, Spring Security, Spring Data и Spring Web.

Для обработки входящих запросов были созданы 4 REST-контроллера: AuthenticationController, AdministratorController, ClientController и SuperAdminController.

3.3.1. AuthenticationController

В контроллере аутентификации находятся методы для обработки 2 эндпоинтов:

- /registration POST
- /login POST

На рисунках 16 и 17 представлены примеры их работы.

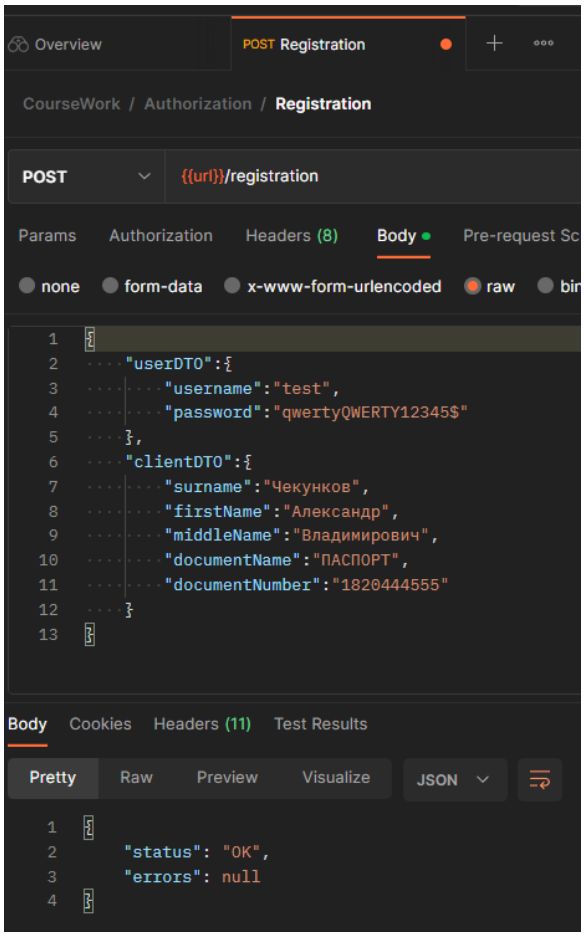

Рисунок 16. – Пример работы /registration

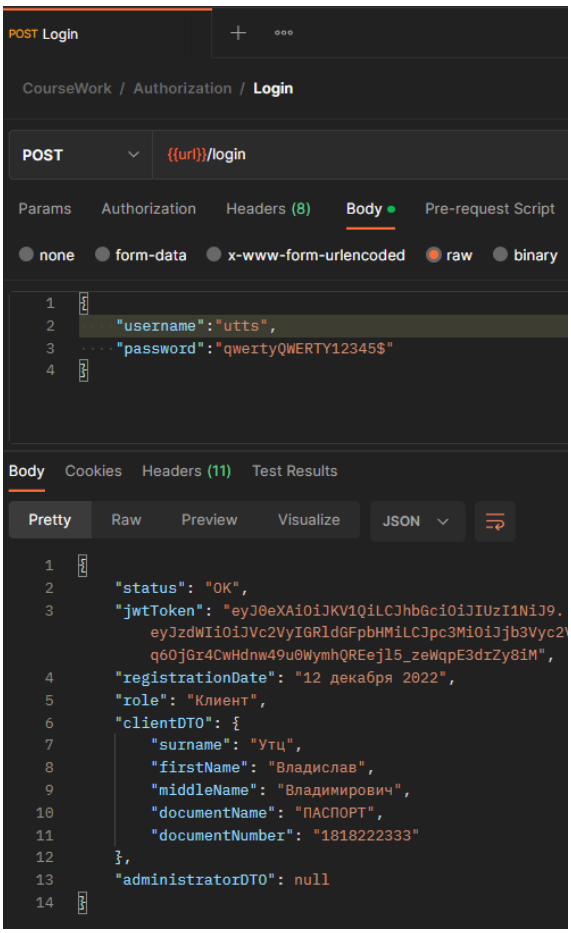

Рисунок 17. – Пример работы /login

В теле запроса авторизации находится JSON с именем пользователя и паролем. В теле ответа /login приходит JSON, в котором указана основная информация о пользователе, а также его Access JWT-токен, который используется в качестве авторизации для остальных запросов. В теле запроса для регистрации.

3.3.2. AdministratorController

В AdministratorController представлен следующий функционал (доступ ко всем эндпоинтам осуществляется с префиксом /administrator):

- /showComplaints GET
- /showClients GET
- /showBids GET
- /handleBid POST
- /addDamage POST
- /topService GET

В теле ответа на запрос GET /showComplaints, пример работы которого представлен на рисунке 18 приходит список всех жалоб с ФИО клиента, написавшего её.

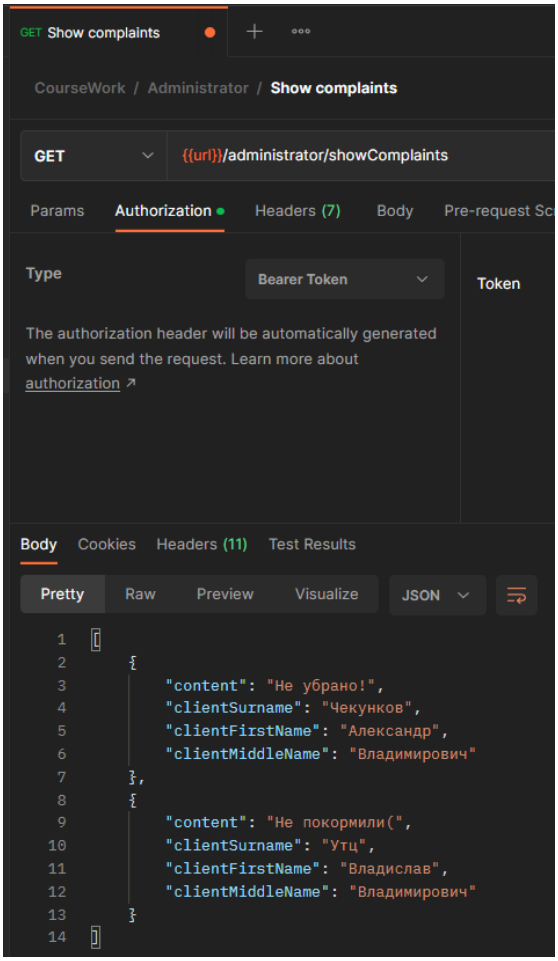

Рисунок 18. – Пример работы /showComplaints

В ответе на GET /showClients приходит информация обо всех клиентах, зарегистрированных в приложении. Пример его работы представлен на рисунке 19.

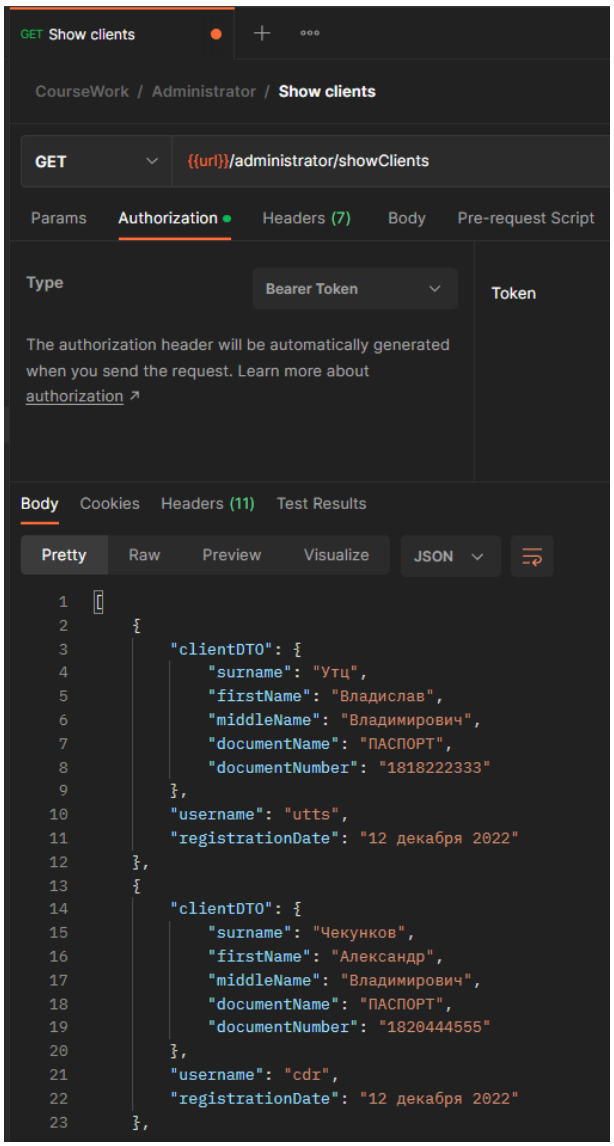

Рисунок 19. – Пример работы /showClients.

В ответ на запрос GET /showBids приходит список всех заявок на регистрацию от клиентов с информацией о категории номера, датах проживания, итоговой стоимости, дополнительных услугах, включенных в регистрацию и ФИО клиента, оставившего заявку. Пример работы данного запроса представлен на рисунке 20.

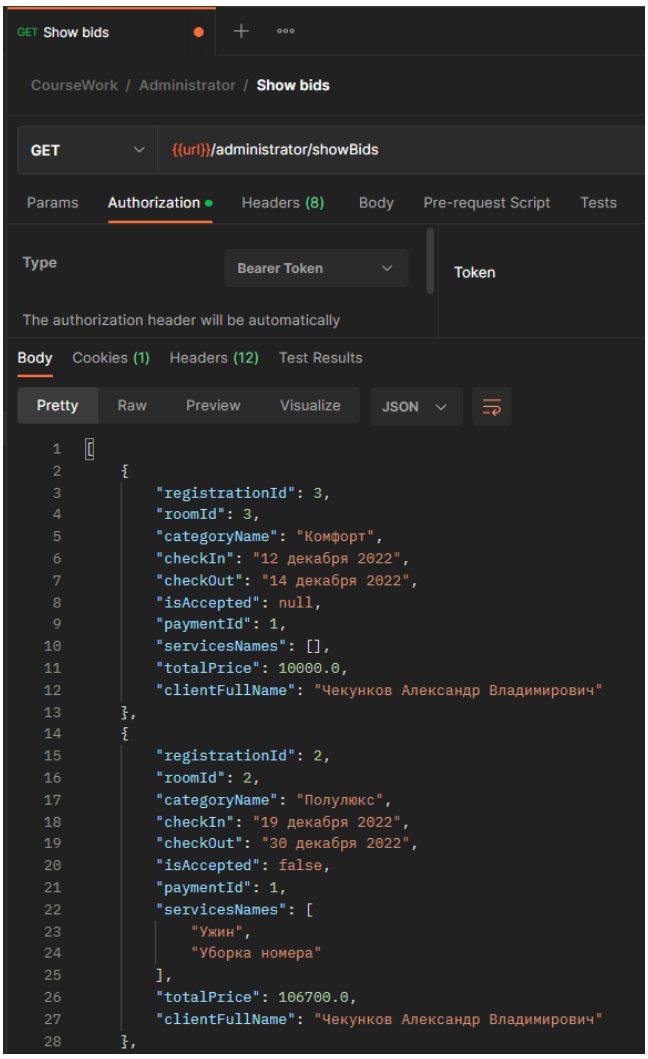

Рисунок 20. – Пример работы /showBids

В теле запроса POST /handleBid отправляется JSON с id регистрации, которую требуется обработать, именем пользователя администратора, статус регистрации принята/отклонена и комментарием. Пример работы представлен на рисунке 21.

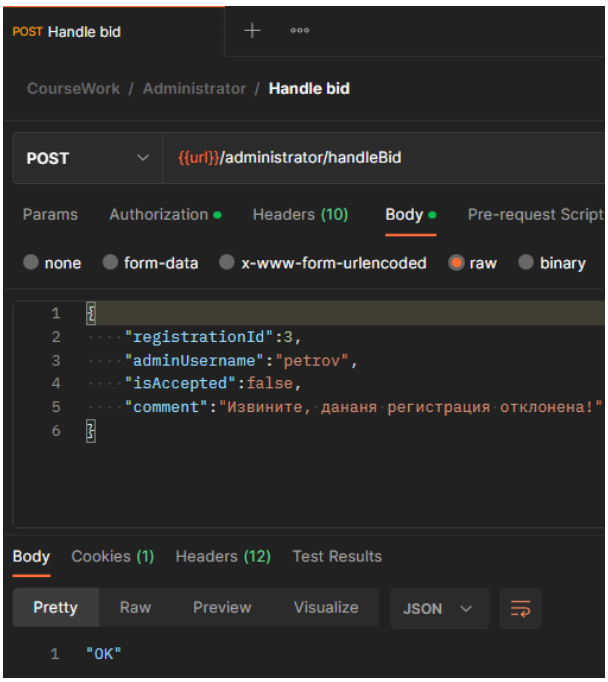

Рисунок 21. – Пример работы /handleBid

В теле запроса POST /addDamage отправляется JSON с id регистрации и сумой ущерба, после успешной отправки, к итоговой стоимости регистрации добавляется указанная сумма с помощью созданной процедуры bigger_price в PostgreSQL. Работа данного запроса представлена на рисунке 22.

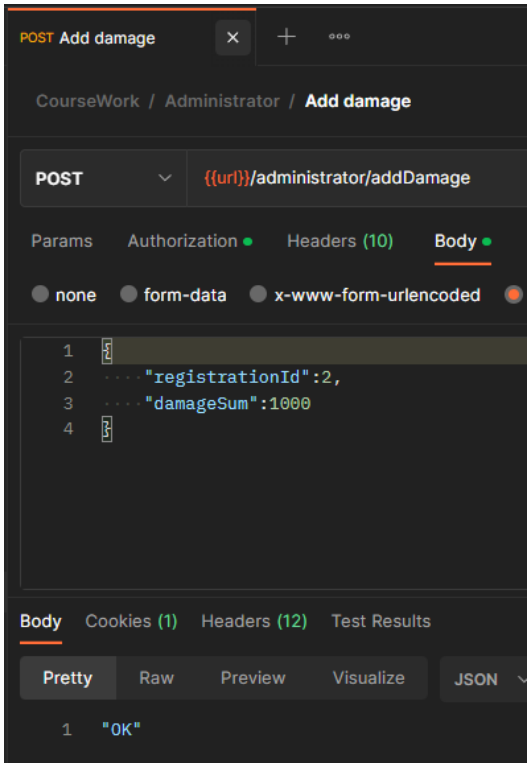

Рисунок 22. – Пример работы /addDamage

В теле ответа GET /topService приходит информация о самой часто покупаемой дополнительной услуге в отеле, с помощью view top_service созданного в PostgreSQL. Пример работы представлен на рисунке 23.

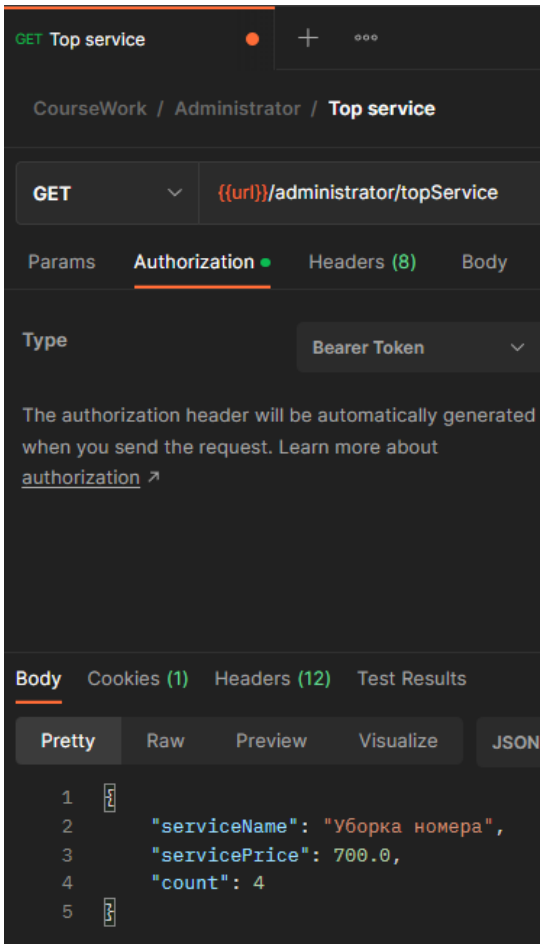

Рисунок 23. – Пример работы /topService

3.3.3. ClientController

В ClientController обеспечен следующий функционал (доступ ко всем эндпоинтам осуществляется с префиксом /client):

- /createComplaint POST
- /showRoomInfo/{id} GET
- /showAllRooms GET
- /sendRoomRegistration POST
- /{username}/getBids GET

В теле запроса POST /createComplaint отправляется «content» - содержание жалобы. Пример работы данного запроса представлен на рисунке 24.

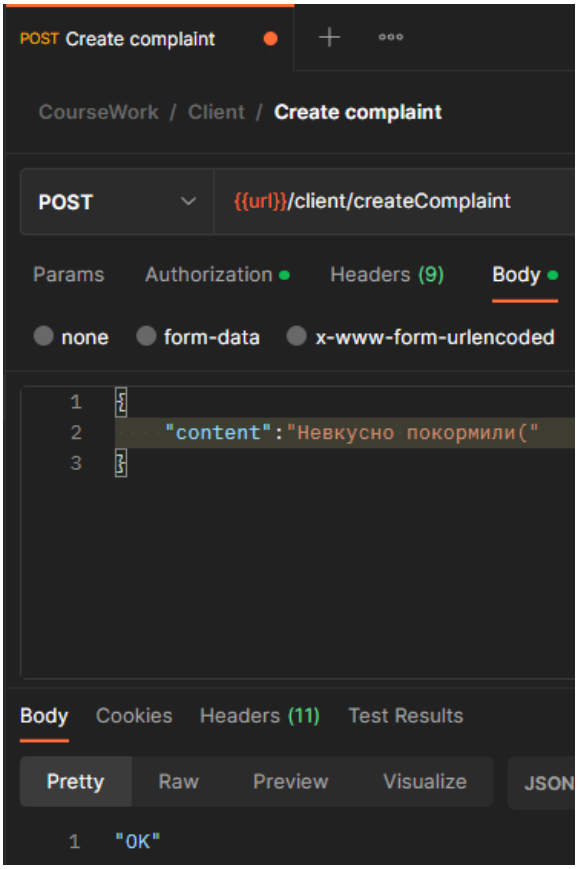

Рисунок 24. – Пример работы /createComplaint

В ответе GET /showAllRooms, пример работы которого представлен на рисунке 25, приходит список комнат с основной информацией о них.

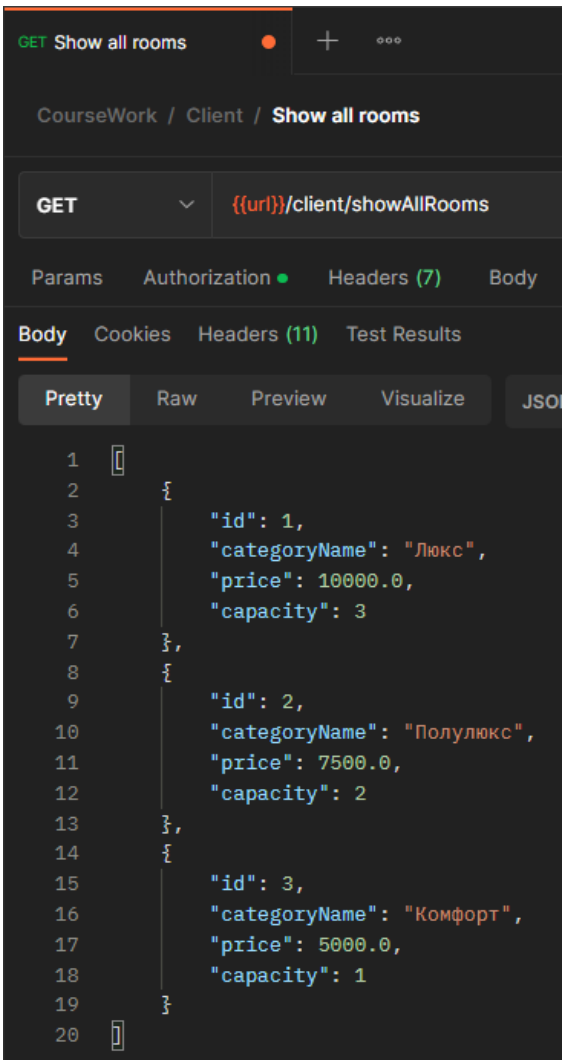

Рисунок 25. – Пример работы /showAllRooms

В запросе GET /showRoomInfo вместо {id} подставляется id комнаты, по которой нужно получить подробную информацию. В теле ответа приходит категория, вид из окна, цена за ночь, вместимость, этаж, доступность кондиционера/фена для волос, фотографии и все доступные дополнительные услуги. Пример работы данного запроса представлен на рисунке 26.

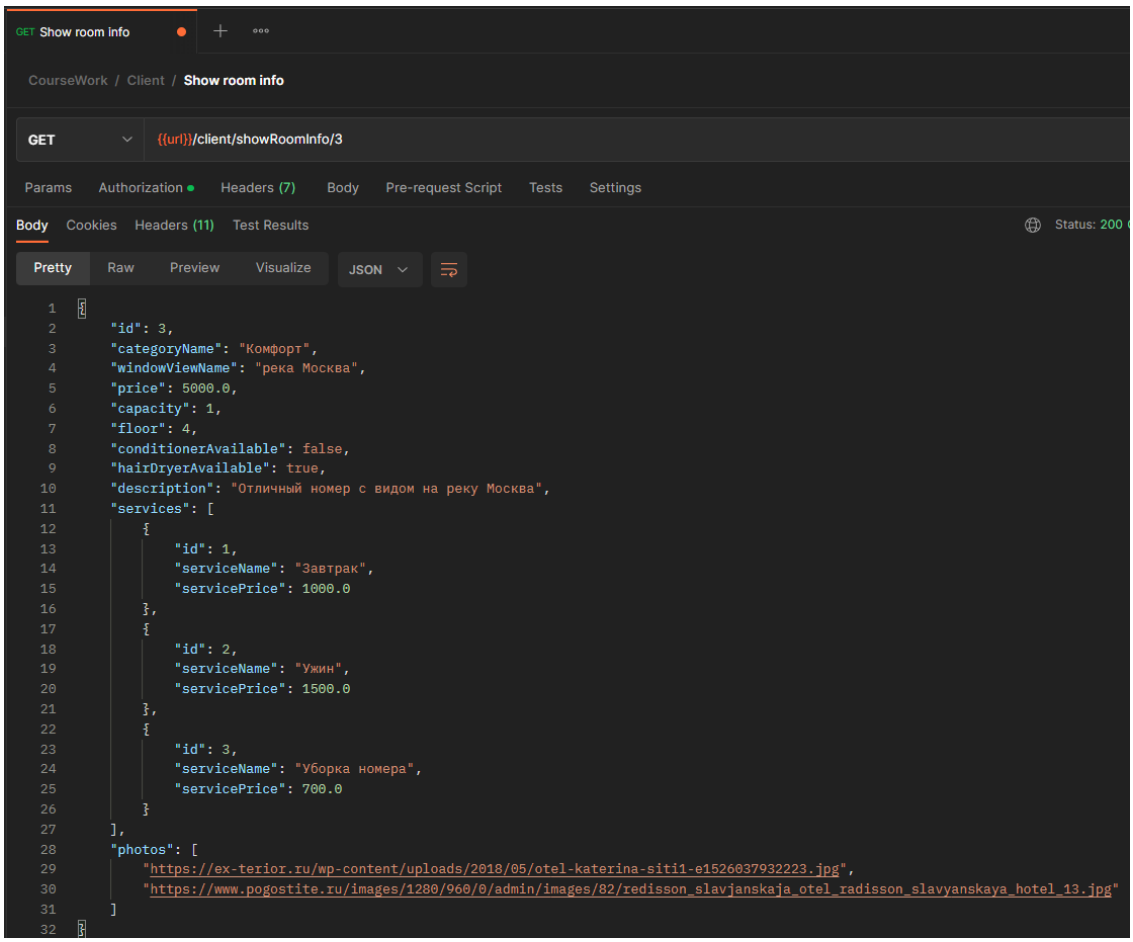

Рисунок 26. – Пример работы /showRoomInfo/{id}

В теле запроса POST /sendRoomRegistration, пример работы которого представлен на рисунке 27, отправляется имя пользователя, оформляющего регистрацию, id комнаты, даты въезда и выезда, массив id дополнительных услуг и id выбранного способа оплаты.

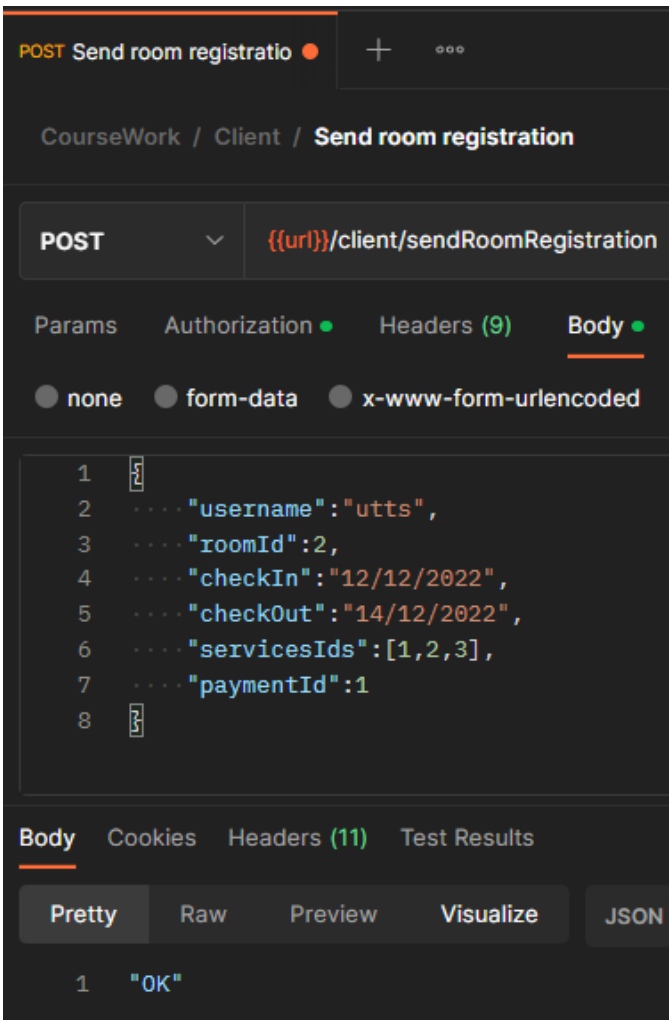

Рисунок 27. – Пример работы /sendRoomRegistration

В запросе GET /{username}/getBids вместо {username} подставляется имя пользователя, заявки на регистрацию которого нужно получить. В теле ответа приходит список всех заявок клиента с информацией о регистрациях – id регистрации, id комнаты, название категории, даты въезда и выезда, статус принята/не принята, комментарий администратора, способ оплаты и названия дополнительных услуг, включенных в регистрацию, а также итоговая стоимость. Пример работы данного запроса представлен на рисунке 28.

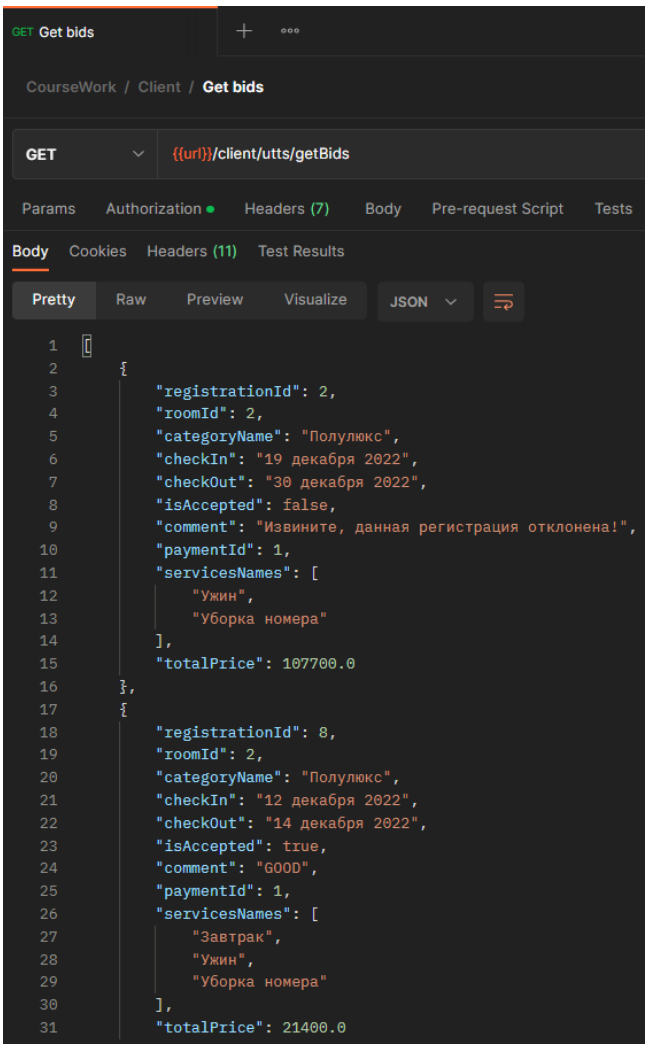

Рисунок 28. – Пример работы /{username}/getBids

3.3.4. SuperAdminController

В SuperAdminControler представлен следующий функционал (доступ ко всем эндпоинтам осуществляется с префиксом /superAdmin):

- /createAdministrator POST
- /showStatistic POST

В теле запроса POST /createAdministrator, пример работы которого представлен на рисунке 29, отправляется два объекта: administratorDTO с ФИО и userDTO с именем пользователя и паролем будущего администратора.

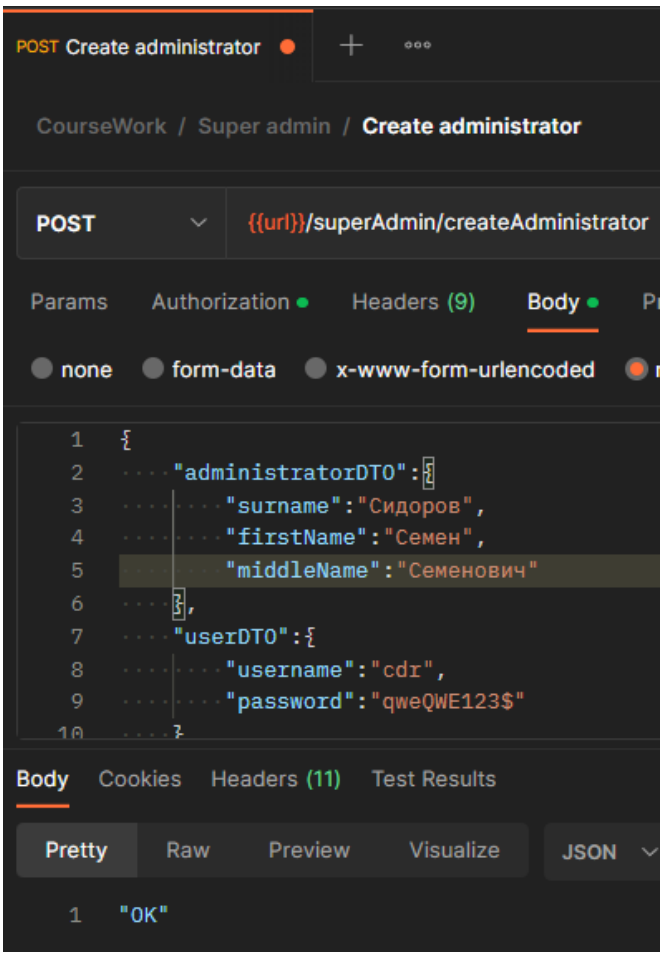

Рисунок 29. – Пример работы /createAdministrator

Запрос POST /showStatistic принимает в теле запроса id администратора и период времени, за который нужно сформировать статистику. После этого, сервер вызывает функцию PostgreSQL save_admin_statistic_to_csv(id bigint, start_period date, end_period date, filepath text), которая сохраняет csv файл на сервер. После этого на сервере формируется объект со статистическими данными и в ответе отправляется статистика администратора за выбранный период времени, в которой содержится количество всех обработанных заявок, количество обработанных заявок номеров люкс и выручка администратора за выбранный период времени. Пример работы данного запроса представлен на рисунке 30.

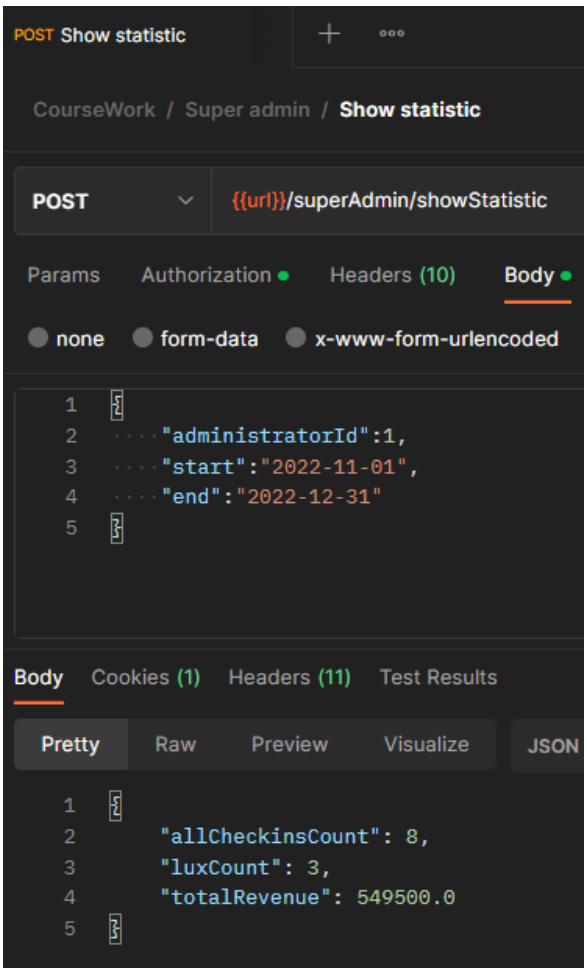

Рисунок 30. – Пример работы /showStatistic

Заключение

В результате выполнения курсовой работы была спроектирована и реализована база данных, отвечающая всем требованиям безопасности и функциональности, кроме того, был разработан графический интерфейс в виде мобильного приложения и серверная часть для частичной автоматизации функционала и обеспечения удобства работы с базой данных.

Репозиторий с полной конфигурацией БД, графическим интерфейсом и серверной частью:

- https://github.com/coder-chekunkov/BookingRoom-Coursework

Список литературы

1. Моргунов Е.П. «PostgreSQL. Основы языка SQL» 2018.

2. ООО “Постгрес Профессиональный” «Документация к Postgres Pro Enterprise 14.5.2» 2022: [Электронный ресурс] URL: https://postgrespro.ru/docs/enterprise/14/ (дата обращения: 18.12.2022).

3. Стоунз PostgreSQL. Основы / Стоунз, Мэттью Ричард; Нейл. - М.: СПб: Символ-Плюс, 2002. - 302 с.

4. Ригс, Саймон Администрирование PostgreSQL 9. Книга рецептов Саймон Ригс, Ханну Кросинг. - М.: ДМК Пресс, 2015. - 291 с.

5. Аллан Бьюли “Изучаем SQL” – М.: Символ-Плюс, 2016 – 179 с.

6. Rod Johnson, Juergen Hoeller, Keith Donald. «Документация к Spring framework» 2022: [Электронный ресурс] URL: https://docs.spring.io/springframework/docs/current/reference/html/ (дата обращения: 18.12.22).

7. Черников В. В. «Must-have документация для мобильного разработчика.» 2017: [Электронный ресурс] URL: https://habr.com/ru/company/microsoft/blog/343660/ (дата обращения: 18.12.2022).