Курс: “Теория баз данных”
Тема проекта: “Houses booking database”

Авторы проекта: Александр Андреевич Ширнин, Шибанин Георгий Вячеславович, Мячин Данил Александрович.

Декабрь 2020

Содержание

Концептуальное проектирование

Описание предметной области

В качестве предметной области мы рассматриваем аренду различной собственности, подходящей для жилья, а также её сдачу в аренду. Собственность может быть разной, например это может быть: дом, коттедж, квартира, апартаменты, даже лодка, в которой можно жить. Арендодатели могут создавать различные объявления: в объявление его создатель может написать краткое описание жилья, указать различные параметры, координаты.

Данная область вызывает интерес, так как люди постоянно что-то арендуют для разных целей: иногда для постоянного жилья, иногда для проживания во время поездок, соответственно, со временем работы такого сервиса можно собирать исторические данные, после чего либо анализировать их, либо собирать некие датасеты для продажи.

Так как этот учебный проект, то для его реализации некоторые данные будут синтезированы или же взяты с публичных датасетов airbnb. Кроме того, на данном этапе мы не обладаем нужными ресурсами для построения глобальной системы, поэтому ограничимся небольшим количеством туристических городов. Начальный функционал мы будем реализовывать с помощью чат-бота в telegram, так как это относительно удобный интерфейс для пользователя, а для создания ботов существует удобные фреймворки.

Построение инфологической модели

Мы выделили основные сущности, необходимые для построения данного сервиса.

Каждая сущность описывает абстрактные уникальные объекты, то есть каждому объекту заведём определённый id. Между сущностями могут быть связи, причём некоторые связи сделаем специально для оптимизации памяти, то есть: сущность “объявление” имеет атрибут “id города”, где он является внешним ключом, поэтому мы в каждой строке храним не название города, а определенную цифру, по которой можем найти название города в отдельной таблице. Теперь опишем каждую сущность:

1. Сущность “Арендодатель” описывает людей, которые сдают в аренду что-то и создают объявления. Сущность включает в себя атрибуты: id, имя арендодателя, количество объявлений, которые он создал, а также дату его регистрации, которая может показывать надёжность пользователя.
2. Сущность “Клиент” описывает пользователей, которые смотрят объявления и, скорее, желают арендовать что-то. Для них мы создадим атрибуты: имя, а также количество отзывов, которые написал человек.
3. Далее нам потребуется сущность “объявление” , которая содержит важную информацию о собственности, сдаваемой в аренду. Ориентируясь на лидеров в нашей предметной области, мы создали следующие атрибуты:
    a. “тип жилья” - некоторые люди могут сдавать дом или гараж, лодку, у клиента должно быть представление об этом).
    b. “Текст описания” - описание, которое составляют арендодатели, где могут описать преимущества своего предложения, указать некие требования или особенности.
    c. “Максимально/Минимально ночей” - это два разных атрибута, они показывают временные ограничичения на аренду собственности.
    d. “Актуальность” - это бинарный признак, который показывает, действует ли объявление сейчас. Для чего он нужен? Как мы написали в начале, одна из целей такого сервиса - это сбор исторических данных, которые могут быть ценным в аналитике, поэтому мы не удаляем объявления, а скрываем их для пользователей.
    e. “Площадь” - площадь жилья.
    f. “Рейтинг” - средняя оценка жилья пользователями.
    g. “Количество отзывов под объявлением” - показывает количество отзывов, которые написали клиенты.
    h. “Залог” - сумма, которую требуют арендодатели перед проживанием, может быть нулевой. Иногда требуется для ремонта или страховки, если недобросовестные жители нанесли ущерб.
    i. “Цена за ночь” - стоимость проживания за ночёвку.

j. “Координаты” - просто координаты жилья. Могут быть полезным для аналитики, а также для нахождения локации.

4. Создадим также сущность “город”, которая будет иметь следующие атрибуты:
    a. “Название” - название города.
    b. “Количество музеев” - для некоторых клиентов важный признак для выбора места проживания.
    c. “Количество аэропортов” - аналогично.
    d. “Количество вокзалов” - также относится к транспорту.
    e. “Средняя зарплата” - в нашем случае показывает среднюю зарплату в евро (у нас европейские города), может быть полезным признакам для понимания расходов на туризм.
    f. “Название валюты” - тоже полезный признак, чтобы человек понимал, в какую валюту ему конвертировать средства. Хоть сейчас пока у нас используется “евро”, в будущем при расширении сервиса атрибут сильно пригодится.
5. Перейдём к описанию сущности “район”. Если город маленький, то его может и не существовать, но если он есть, то это более подробное описание ближайшей местности:
    a. “Название” - название района.
    b. “Количество больниц” - отражает комфортность района для лечения.
    c. “Наличие метро” - это бинарный признак, который отражает удобство передвижение людям без автомобиля.
    d. “Количество круглосуточных магазинов” - у некоторых клиентов может быть сложный график работы, для них будет важна эта информация.
6. Теперь нам нужна сущность “комментарии”. Она будет содержать нужные атрибуты, чтобы люди могли писать отзыв о своём проживании где-то.
    a. “Текст” - сам текст комментария.
    b. “Количество лайков на отзыве”, “Количество дизлайков на отзыве” обычно показывают, сколько человек согласны или не согласны с автором отзыва.
    c. “Оценка пользователя” - то, как оценил проживание клиент.
    d. “Редакция” - бинарный признак, который показывает, менял ли отзыв человек.

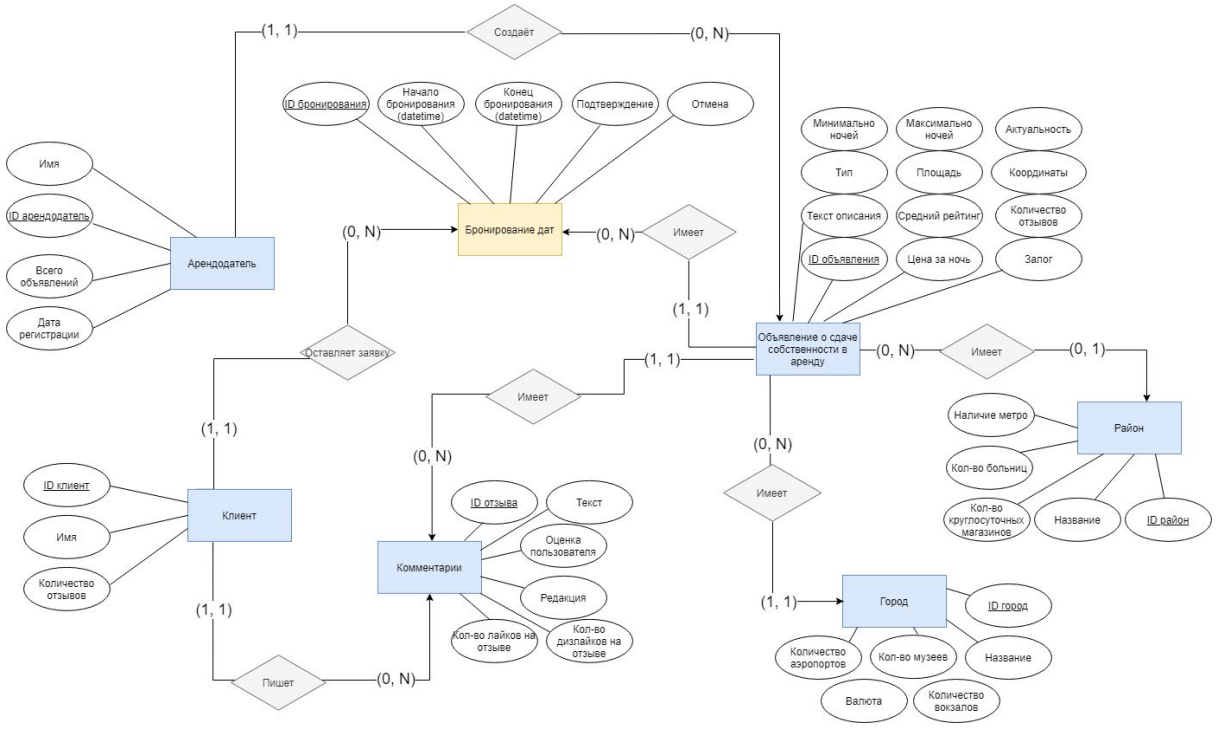
7. “Бронирование дат” - это особенная сущность, которая зависит от клиентов. С помощью неё мы будем хранить заявки на бронирование жилья. Для этого потребуются следующие атрибуты:
    a. “Начало брони”, “Конец брони” - интервал дат, которые хочет забронировать клиент.
    b. “Подтверждение” - бинарный признак, будет показывать, подтвердил ли арендодатель бронь.
    c. “Отмена” - бинарный признак, чтобы мы могли реализовать возможность отказа клиента от брони.

Теперь опишем связи, необходимые для работы сервиса.

1. Связь “создаёт” [один ко многим] - позволяет арендодателям создавать несколько объявлений, с помощью неё к каждому объявлению относится только один арендодатель. Если же владелец собственности сменился, то прежний арендодатель может деактивировать старые объявления.
2. Связь “имеет” между “объявлением” и “бронированием дат” показывает, сколько запросов на бронь имеет жильё, описанное в объявлении. У объявления может быть много запросов, однако в каждой заявке клиент указывает только одно объявление, на которое он подаёт запрос.
3. Связь “имеет” между “объявлением” и “городом” [один ко многим] - показывает, в каком городе находится собственность из объявления. Как мы написали выше, это сделано для оптимизации.
4. Связь “имеет” между “объявлением” и “районом” [один ко многим], аналогично предыдущему пункту.
5. Связь “имеет” между “объявлением” и “комментариями” - связывает комментарии с нужными объявлениями. Каждое объявление может иметь 0 или несколько комментариев.
6. Связь “пишет” между “комментариями” и “клиентом” - означает, кто клиент, может, написать несколько комментариев о собственности, где он жил.
7. Связь “оставляет заявку” между “клиентом” и “бронированием дат” обозначает, что клиент может отправить заявку на проживание в каком-то месте, описанном в объявлении. Далее эту заявку либо подтвердит, либо не подтвердит арендодатель. В заявке важно

указать объявление, а id арендодателя уже восстанавливает через объявление.

Так выглядит **ER - диаграмма.** По ссылке можно посмотреть в лучшем качестве:



Ссылка на модель:
https://github.com/25icecreamflavors/db_project/blob/master/models/er_model_booking.svg

Реляционная модель

Выполним переход к реляционной модели. Укажем внутренние ключи для обеспечения уникальности объектов, хранимых в таблицах. Установим внешние ключи для каждой связи. Для связей [один ко многим] установим внешний ключ в таблицу с множественной связью. Вначале для каждой сущности построим отдельную таблицу с типами данных, связями, кратким описанием, что означает, каждый атрибут.

Получим следующие таблицы:

landlord

Key	Name	Data type	References	Description
PK	id_owner	INT		Unique id number of a landlord
	name	VARCHAR(50)		Name of a landlord
	announcement_number	INT		Number of posts created
	date_registration	DATE		

client

Key	Name	Data type	References	Description
PK	id_client	INT		Unique id number of a client
	name	VARCHAR(50)		Name of a client

city

Key	Name	Data type	References	Description
PK	id_city	INT		unique number
	name	VARCHAR(50)		City name
	currency	VARCHAR(50)		Currency that’s used in this city
	number_of_airports	INT		Number of airports located in

				the city
	number_of_museums	INT		Number of museums located in the city
	number_of_railway_stations	INT		Number of train stations located in the city
	average_salary	FLOAT		Average salary in euro in the city

suburb

Key	Name	Data type	References	Description
PK	id_suburb	INT		Unique id for suburb
	availability_of_metro	BIT		Shows is there a metro
	number_of_hospitals	INT		Number of hospitals in suburb
	`number_of_conv_stores`	INT		Number of stores that work all day

announcement

Key	Name	Data type	References	Description
PK	id_post			
FK	id_owner		landlord(id_owner)	owner of house
FK	id_city		city(id_city)	city where house is located
FK	id_suburb		suburb(id_suburb)	suburb where house is located
	text_description	VARCHAR(8000)		Text about rented house
	number_of_review	INT		Number of comments about this house
	average_rating	FLOAT		Rating of the house

	min_nights	INT		Minimum number of nights are available to book
	max_nights	INT		Maximum number of nights are available to book
	pledge	FLOAT		Money that client pays before living and get back after leaving if all is ok
	cost_per_night	FLOAT		Price that client pays for one night
	type_house	VARCHAR(50)		Type of the accommodation
	square	FLOAT		Area of the house
	latitude	FLOAT		Coordinate
	longitude	FLOAT		Coordinate
	relevance	BIT		Shows if an announcement is activated now

comments

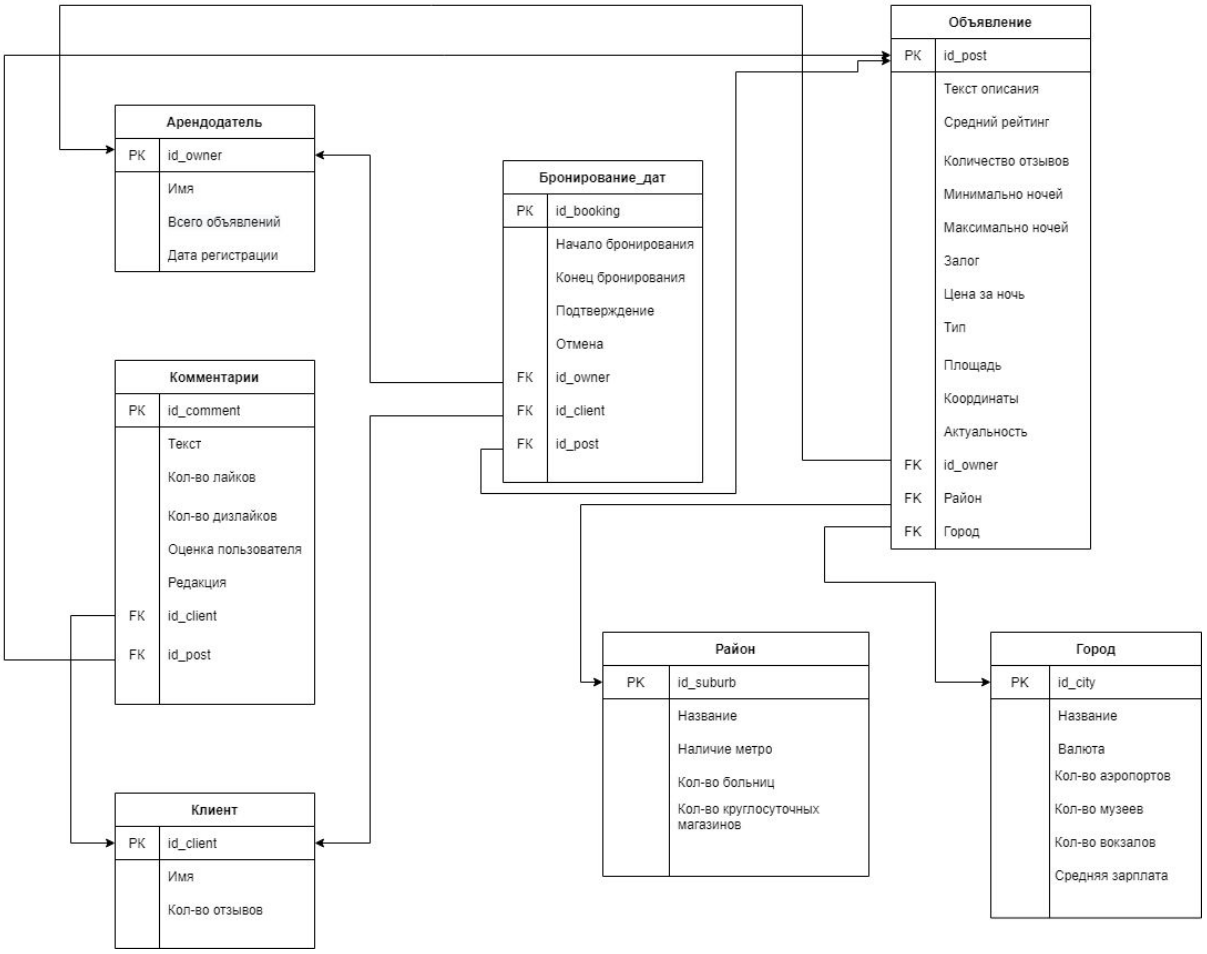
Key	Name	Data type	References	Description
PK	id_comment			
	text	VARCHAR(1000)		Text of the comment
	number_of_likes	INT		Likes on the comment by other users
	number_of_dislikes	INT		Dislikes on the comment by other users
	grade_of_client	INT		How client rated the house
	edition	BIT		If the comment was edited
FK	id_client		client(id_client)	Client that wrote the comment
FK	id_post		announcement(id_	Post where client

			post)	leave the comment

booking_date

Key	Name	Data type	References	Description
PK	id_booking	INT		id of booking request
	start_booking	DATETIME		Date of desired booking beginning
	end_booking	DATETIME		Date of desired booking end
	confirmation	BIT		If the landlord accepted request
	cancelling	BIT		If the client cancelled booking
FK	id_client	INT	client(id_client)	Client who sent the request
FK	id_owner	INT	landlord(id_owner)	Person who posted the announcement and can confirm the request
FK	id_announcement	INT	announcement(id_post)	Announcement of the estate that client wants to book

Теперь перейдём от них к одной общей диаграмме. Так выглядит диаграмма, по ссылке ниже можно увидеть её в лучшем качестве:



https://github.com/25icecreamflavors/db_project/blob/master/models/tr_model_house.svg

Механизмы обеспечения целостности данных

Хранимые данные имеют определённый тип, но пользователи при вводе могут нарушать специально или случайно некие правила. Для корректной работы базы данных мы будем это ограничивать ещё на этапе вводе этих данных, то есть на стороне приложения. Если данные некорректные, то просто не будем отправлять их в базу данных, а попросим пользователя ввести ещё раз.

Чтобы избежать ситуаций, когда базу данных изменяют одновременно несколько человек, мы будем это контролировать опять же на уровне

бэкэнда со стороны клиентского приложения, и изменять данные будем по очереди.

Также мы сделали триггер, который увеличивает “количество объявлений” у арендодателя, когда он создаёт новое, чтобы их там было корректное количество. Мы не стали создавать отдельные триггеры для логирования регистрации владельца собственности, дата регистрации уже хранится в сущности “арендодатель”.

Развёртывание базы данных

Ниже приведём DDL код для создания базы данных:

Будем использовать диалект T-SQL и SQL Azure database. Обратите внимание, что в некоторых таблицах мы генерируем id с помощью `identity(1,1)`, он будет увеличиваться на 1 с добавлением каждой новой строки, а в некоторых таблицах мы его будем прописывать сами.

```
CREATE DATABASE HOUSES;

CREATE TABLE landlord (
    id_owner INT PRIMARY KEY,
    name VARCHAR(50),
    announcement_number INT,
    date_registration DATE
);

CREATE TABLE suburb (
    id_suburb INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(20),
    availability_of_metro BIT,
    number_of_hospitals INT,
    number_of_conv_stores INT
);

CREATE TABLE city (
    id_city INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(30),
    currency VARCHAR(50),
    number_of_airports INT,
    number_of_museums INT,
    number_of_railway_stations INT,
    average_salary FLOAT
);

CREATE TABLE client (
    id_client INT PRIMARY KEY,
    name VARCHAR(50)
```

```
);

CREATE TABLE announcement (
    id_post INT IDENTITY(1,1) PRIMARY KEY,
    text_description VARCHAR(8000),
    average_rating FLOAT,
    number_of_review INT,
    min_nights INT,
    max_nights INT,
    pledge FLOAT,
    cost_per_night FLOAT,
    type_house VARCHAR(50),
    square FLOAT,
    latitude FLOAT,
    longitude FLOAT,
    revelance BIT,
    id_owner INT,
    id_suburb INT,
    id_city INT,
    FOREIGN KEY (id_owner) REFERENCES landlord(id_owner),
    FOREIGN KEY (id_suburb) REFERENCES suburb(id_suburb),
    FOREIGN KEY (id_city) REFERENCES city(id_city)
);

CREATE TABLE comments (
    id_comment INT IDENTITY(1,1) PRIMARY KEY,
    text VARCHAR(1000),
    number_of_likes INT,
    number_of_dislikes INT,
    grade_of_client INT,
    edition BIT,
    id_client INT,
    id_post INT,
    FOREIGN KEY (id_client) REFERENCES client(id_client),
    FOREIGN KEY (id_post) REFERENCES announcement(id_post)
);

CREATE TABLE booking_date (
    id_booking INT IDENTITY(1,1) PRIMARY KEY,
    start_booking DATETIME,
    end_booking DATETIME,
    confirmation BIT,
    canceling BIT,
    id_owner INT,
```

```
    id_client INT,
    id_post INT,
    FOREIGN KEY (id_owner) REFERENCES landlord(id_owner),
    FOREIGN KEY (id_client) REFERENCES client(id_client),
    FOREIGN KEY (id_post) REFERENCES announcement(id_post)
);
```

Также мы создавали некоторые процедуры для удобства использования базы данных: например, процедуру добавления данных, а также процедуры для вывода некоторой статистики по объявлениям или городам. Код приведём ниже, а также код для триггера, о котором писали в предыдущем разделе:

Процедура для добавления информации о районе:

```
GO
CREATE PROCEDURE AddSuburb
    @name NVARCHAR(20),
    @metro BIT,
    @hospital INT,
    @shops INT
AS
INSERT INTO suburb
VALUES(@name, @metro, @hospital, @shops);
```

Процедура для добавления информации о городе:

```
GO
CREATE PROCEDURE AddCity
    @name VARCHAR(30),
    @currency  VARCHAR(50),
    @airports INT,
    @museums INT,
    @train INT,
    @salary INT
AS
INSERT INTO city
VALUES(@name, @currency, @airports, @museums, @train,
@salary);
```

Процедура для получения средней цены собственности, сданной в аренду в каждом городе:

```
GO
CREATE PROCEDURE Houses.RentAveragePrices As
    SELECT AVG(cost_per_night) AS [Average price], id_city AS
city
    FROM Houses.announcement
    GROUP BY id_city
```

Процедура для получения id всех объявлений, по заданному городу:

```
GO
CREATE PROCEDURE get_posts_bycity
@city NVARCHAR(50) AS
    SELECT announcement.id_post, city.name
    FROM announcement LEFT JOIN city ON announcement.id_city =
city.id_city
    WHERE city.name = @city;
```

Триггер для увеличения количества объявлений у пользователя после добавления нового объявления в таблицу:

```
GO
CREATE TRIGGER post_addition ON announcement
    AFTER INSERT
    AS
      UPDATE landlord
SET landlord.announcement_number =
landlord.announcement_number + 1
      FROM inserted
      LEFT JOIN landlord
      ON inserted.id_owner = landlord.id_owner
      WHERE landlord.id_owner =  inserted.id_owner;
```

Приведём примеры DML-операторов для вставки данных в таблицу:

```
INSERT INTO city VALUES (‘Barcelona’, ‘euro’, 3, 55, 2, 2400);
```

`INSERT INTO client VALUES (‘12’, ‘Mario Fibonaci’);` - заметим, что тут мы сами указываем его id, потому что в данном случаем мы работаем с телеграммом и будем для удобства брать номер пользователя из него.

```
INSERT INTO suburb VALUES (‘Plaka’, 1, 3, 56);
```

Примеры отчётов

1. Выведем среднюю цену за ночь, средний залог, среднюю площадь из объявлений, где цена за ночь больше 100 евро, а также названия городов, по которым сгруппировали это. Таким образом получим некую статистику по “элитной собственности”:

```
SELECT AVG(cost_per_night) AS [Average price], AVG(pledge) AS [Average pledge],
AVG(square) AS Square, Houses.city.name AS City
FROM Houses.announcement
LEFT JOIN Houses.city ON Houses.announcement.id_city = Houses.city.id_city
WHERE cost_per_night > 100
GROUP BY Houses.city.name
```

	Average price	Average pledge	Square	City
1	163,07142857142858	42,142857142857146	30,785714285714285	Amsterdam
2	120	41	37	Athens
3	140	18,333333333333332	49	Barcelona

2. Выведем города, начинающиеся на “a”, и среднюю зарплату по ним в евро, где есть хотя бы один аэропорт, а также больше 10 музеев, отсортируем их по названиям:

```
SELECT name AS City, average_salary AS [Average Salary]
FROM Houses.city
WHERE number_of_airports >= 1 AND number_of_museums > 10
AND LOWER(name) LIKE 'a%'
ORDER BY name
```

	City	Average Salary
1	Amsterdam	3400
2	Athens	2300

3. Выведем id объявлений, цену за ночь, залог, площадь и город объявлений, где цена за ночь больше 30 евро, площадь больше 35, а дата регистрации владельца раньше 1 октября 2020 года, а также в городе есть хотя бы один аэропорт:

```
SELECT id_post AS [Post id], cost_per_night AS Price,
pledge AS Pledge, square AS Square, Houses.city.name AS City
FROM Houses.announcement
LEFT JOIN Houses.city ON Houses.announcement.id_city =
Houses.city.id_city
LEFT JOIN Houses.landlord ON Houses.announcement.id_owner =
Houses.landlord.id_owner
WHERE cost_per_night > 30 AND square > 35
AND Houses.landlord.date_registration < '2020-10-01' AND
Houses.city.number_of_airports > 0
ORDER BY id_post
```

	Post id	Price	Pledge	Square	City
1	9	219	44	36	Amsterdam
2	10	160	46	37	Amsterdam
3	19	96	42	38	Amsterdam
4	27	131	17	47	Barcelona
5	33	60	22	48	Athens
6	34	36	23	38	Athens
7	35	33	25	48	Athens
8	36	36	41	48	Athens
9	37	38	34	45	Athens
10	41	120	41	37	Athens

4. Выведем количество объявлений по районам, где цена за ночь находится в интервале (25, 80) евро, площадь жилья от 30 кв.метров, в городе присутствует хотя бы 1 аэропорт, а дата регистрации арендодателя раньше 20 сентября 2020 года. Выведем также город и отсортируем по убыванию количества таких объявлений.

```
SELECT COUNT(id_post) AS [Number of announcements],
Houses.suburb.name AS [District name], Houses.city.name AS
[City]
FROM Houses.announcement
LEFT JOIN Houses.city ON Houses.announcement.id_city =
Houses.city.id_city
LEFT JOIN Houses.suburb ON Houses.suburb.id_suburb =
Houses.announcement.id_suburb
LEFT JOIN Houses.landlord ON Houses.landlord.id_owner =
Houses.announcement.id_owner
WHERE cost_per_night BETWEEN 25 AND 80
AND square > 30 AND Houses.city.number_of_airports > 0
AND Houses.landlord.date_registration < '2020-09-20'
GROUP BY Houses.suburb.name, Houses.city.name
ORDER BY [Number of announcements] DESC
```

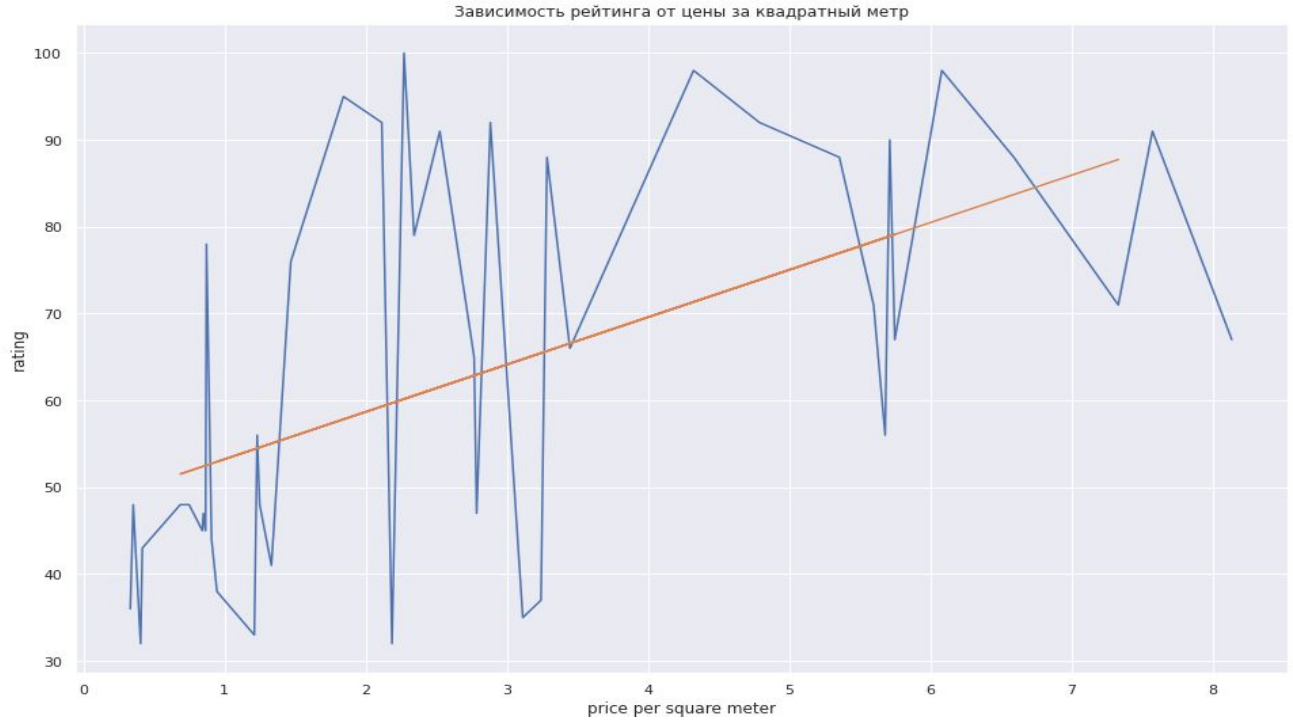
	Number of announcements	District name	City
1	4	Kolonaki	Athens
2	3	Plaka	Athens
3	1	Eixample	Barcelona
4	1	Amsterdam West	Amsterdam

5. Теперь выведем ещё один отчёт. Получим количество объявлений по городам с одинаковым типом жилья, в районах которых находится метро, как минимум, 2 больницы, а также больше 2 круглосуточных магазинов.

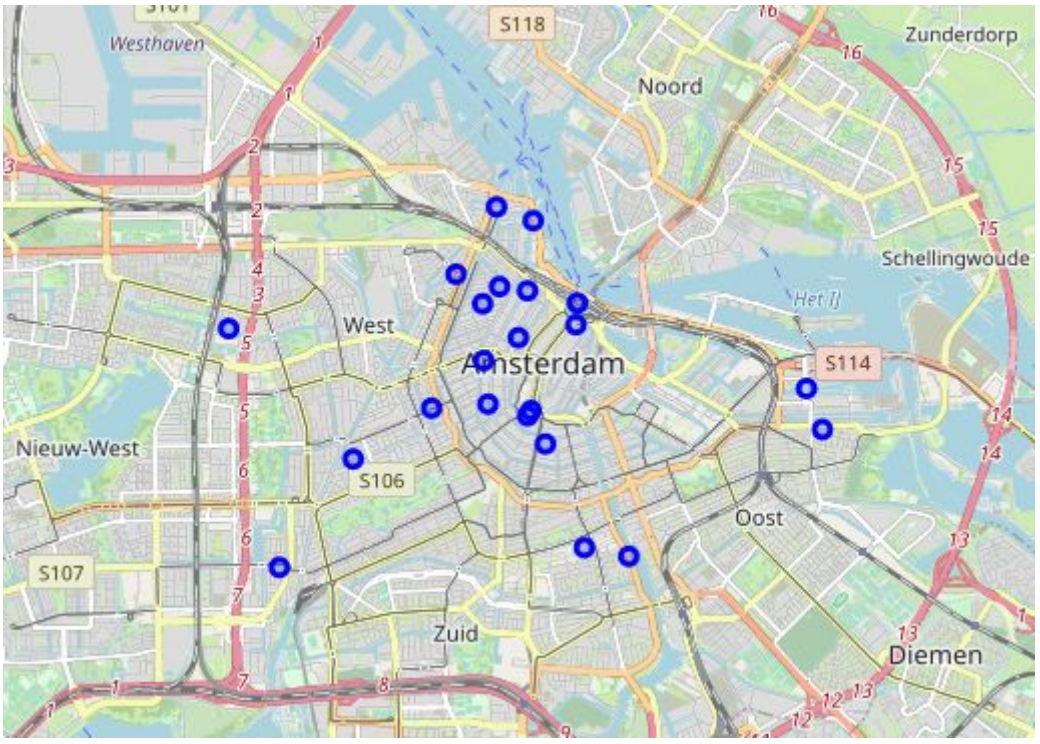
```
SELECT COUNT(id_post) AS [Number of announcements],
Houses.announcement.type_house, Houses.city.name AS [City]
FROM Houses.announcement
LEFT JOIN Houses.city ON Houses.announcement.id_city =
Houses.city.id_city
LEFT JOIN Houses.suburb ON Houses.suburb.id_suburb =
Houses.announcement.id_suburb
WHERE Houses.suburb.availability_of_metro = 1 AND
Houses.suburb.number_of_hospitals >= 2 AND
Houses.suburb.number_of_conv_stores > 2
GROUP BY Houses.announcement.type_house, Houses.city.name
```

	Number of announcements	type_house	City
1	1	2 bedroom appts.	Amsterdam
2	4	Entire apartment	Amsterdam
3	1	Entire guest suite	Amsterdam
4	1	Entire townhouse	Amsterdam
5	1	Flat	Amsterdam
6	2	Private room in apartment	Amsterdam
7	1	Private room in bed and brea…	Amsterdam
8	1	Private room in guesthouse	Amsterdam
9	4	Entire apartment	Athens
10	1	Private room in apartment	Athens
11	3	Entire apartment	Barcelona
12	7	Private room in apartment	Barcelona

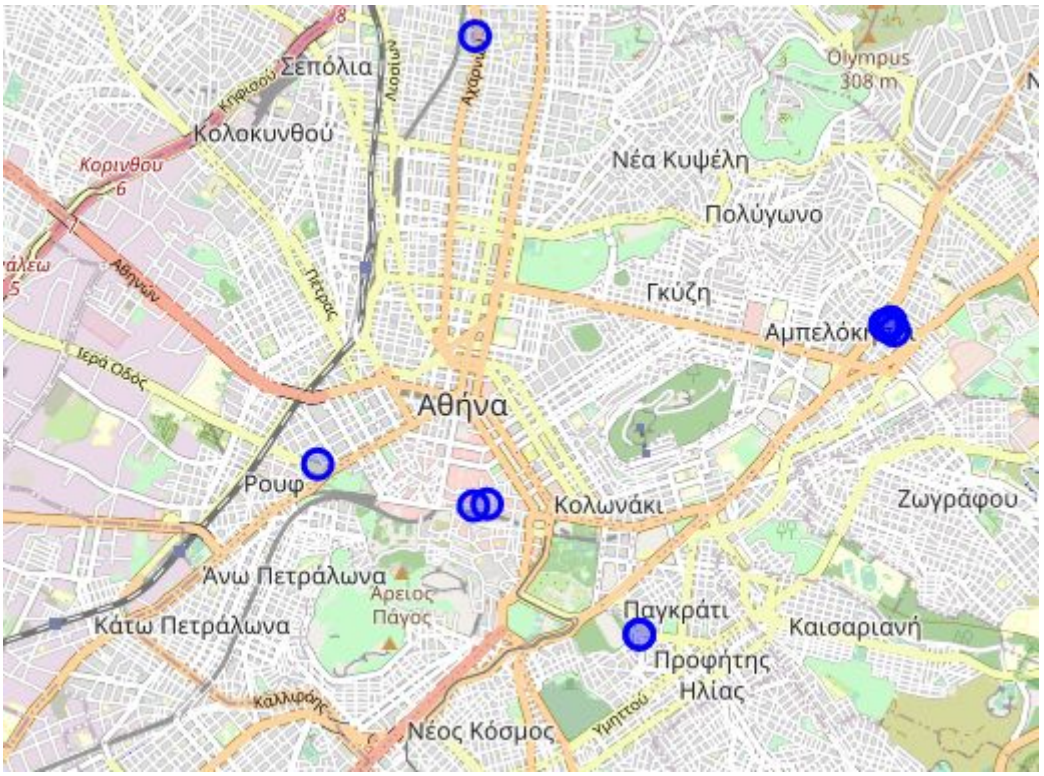
6. Построим график зависимости рейтинга (от 0 до 100) от цены (в евро) аренды за кв. метр. Этот признак создадим сами: разделим площадь на стоимость за ночь. Построим также линейную регрессию, чтобы проверить тренд. Действительно, видим, что несмотря на большое количество выбросов, прослеживается тренд увеличения рейтинга с ростом цены за кв. метр:



Так выглядит расположение собственности в разных городах на карте:


Westhaven
S118
Noord
Zunderdorp
Schellingwoude
Het IJ
West
Amsterdam
S114
Nieuw-West
S106
Oost
S107
Zuid
Diemen


la Creueta del Coll 246 m
la Rovira 262 m
Diagonal Mar
Besòs
Poblenou
Poblenou
Ronda Litoral
Carrer d'Aragó
Barcelona
Port Vell


Σεπόλια
Κολοκυνθού
Κορινθου
Αθηνών
Ιερά Οδός
Πέτρας
Λιοσίων
Αχαρνών
Olympus
308 m
Νέα Κυψέλη
Πολύγωνο
Γκύζη
Αμπελόκι
Αθήνα
Ρουφ
Κολωνάκι
Ζωγράφου
Άνω Πετράλωνα
Άρειος Πάγος
Κάτω Πετράλωνα
Παγκράτι
Καισαριανή
Προφήτης Ηλίας
Καλλιφάς
Νέος Κόσμος
Υμηττού

Разработка приложения

Бизнес логика

В качестве клиентского приложения мы разработали телеграм-бота. Взаимодействие с ним происходит как с обычным ботом (приложу пикрелейтед стартового сообщения)

С помощью бота можно как арендовать, так и сдавать в аренду помещения. Пользователям нужно зарегистрироваться (есть процесс регистрации и для арендодателей, и для арендателей). Эти взаимодействия происходят с таблицами Landlord и Client.

При добавлении нового объявления оно появляется в таблице Announcement.

Использовался провайдер данных ODBC с драйвером 17 for SQL Server. Реализовывалось всё на языке Python3.

Прямой доступ к таблицам закрыт, пользователи только общаются с ботом, передавая ему информацию, а серверная часть бота уже изменяет данные в таблицах.

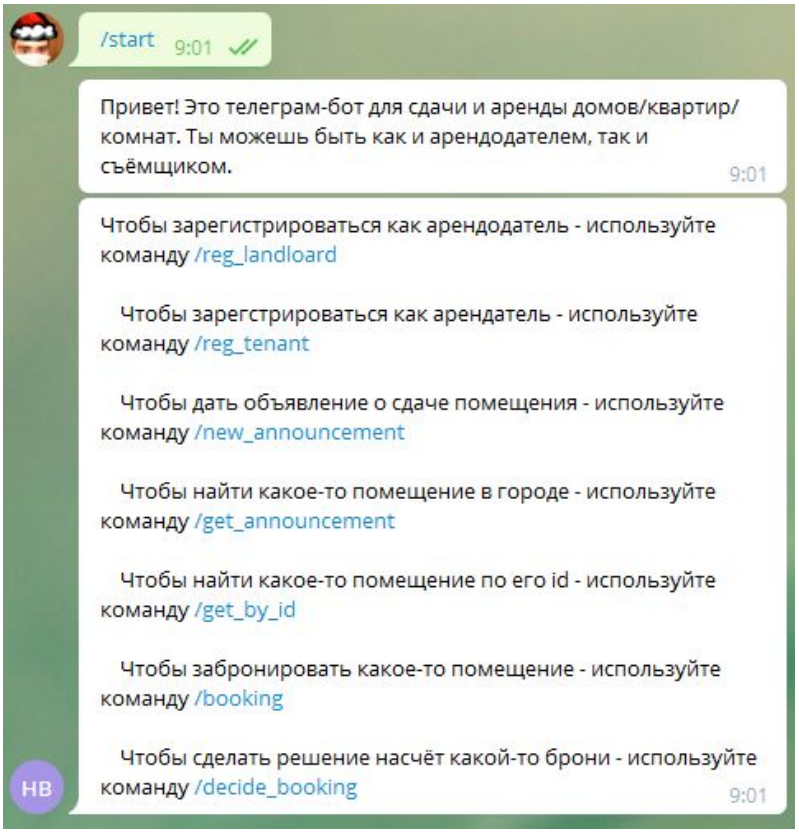
Библиотеки все выбраны стандартные, за исключением ODBC, так как только для неё был готовый пример работы, не требующий анализа информации из интернета.

Код бота находится по адресу:

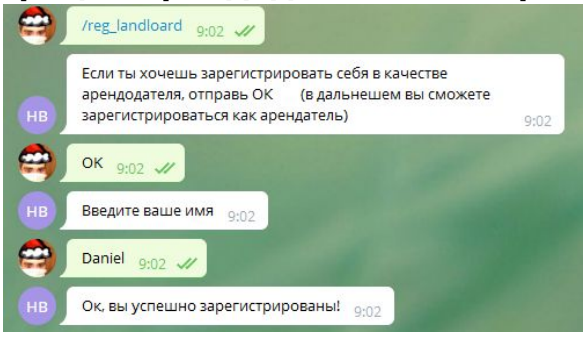
https://github.com/mdan2000/DBMS_project/tree/master/bot

Ввод данных

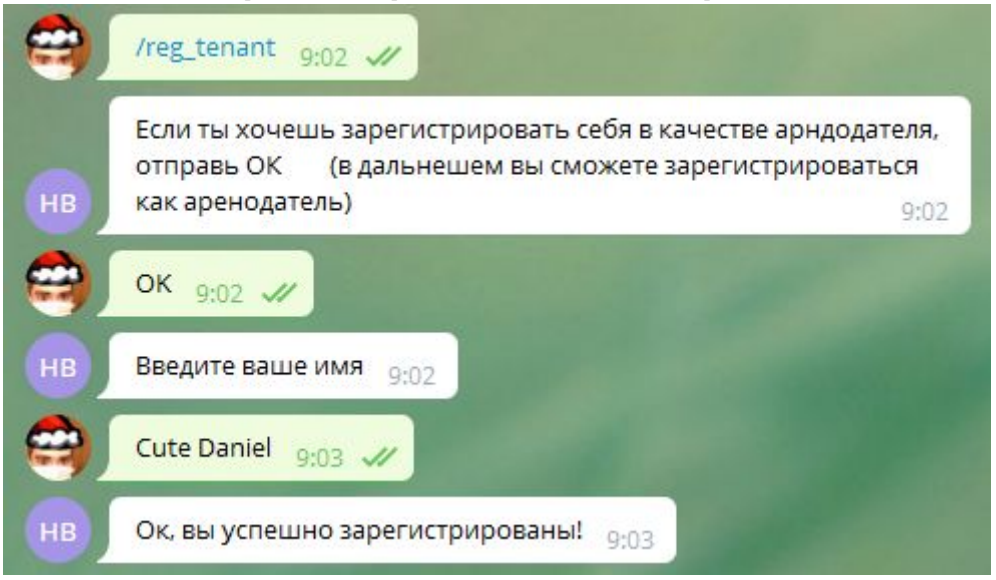
Первое сообщение с ботом



Регистрация арендодателей(авторизация)



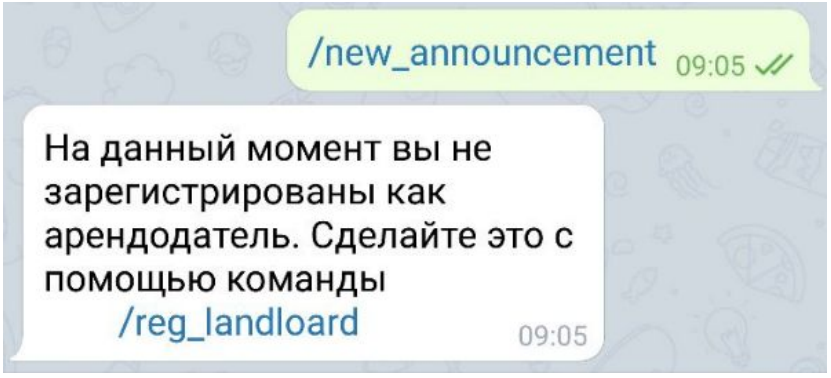
Регистраций арендателей(авторизация)



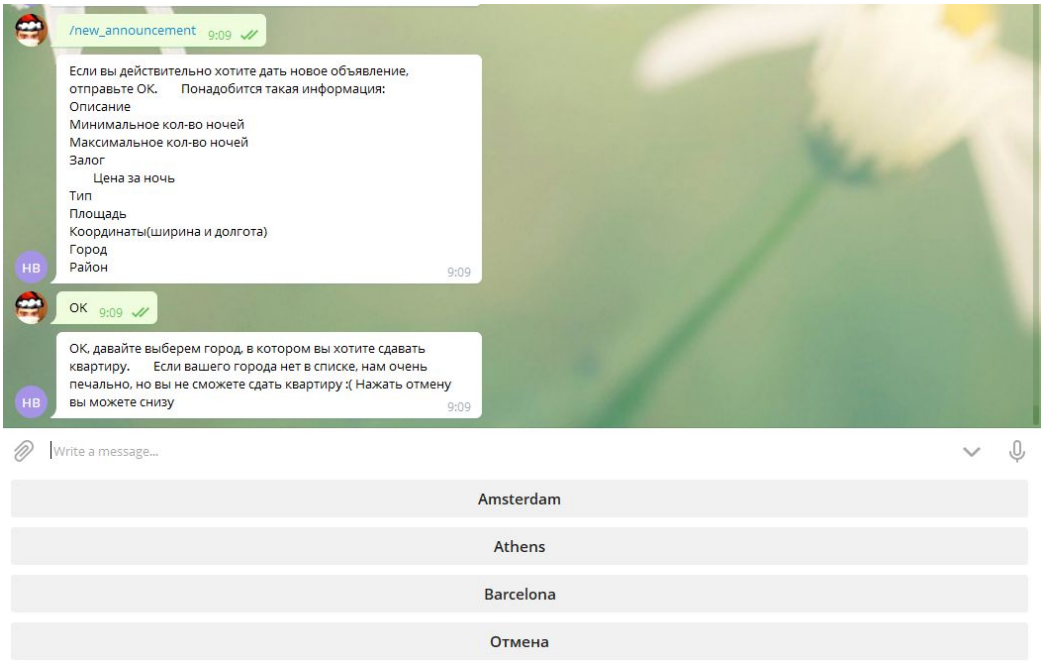
Добавление новых объявлений

И здесь можно упомянуть про то, что все действия, предназначенные для определенной роли, нельзя выполнить, если вы не зарегистрировались в качестве этой роли. Например: нельзя дать объявление, если вы не зарегистрировались в качестве арендодателя или нельзя найти объявление, если вы не зарегистрировались в качестве арендателя.

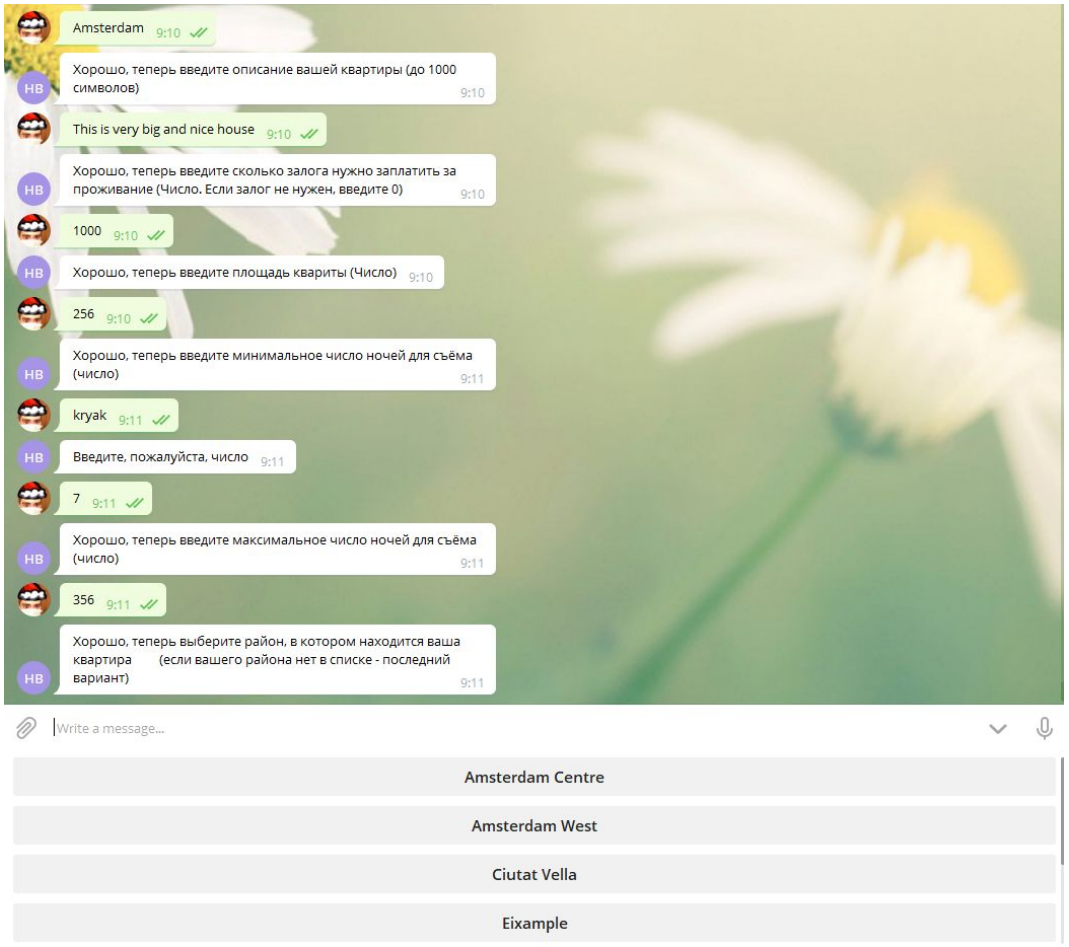
Незарегистрированный пользователь

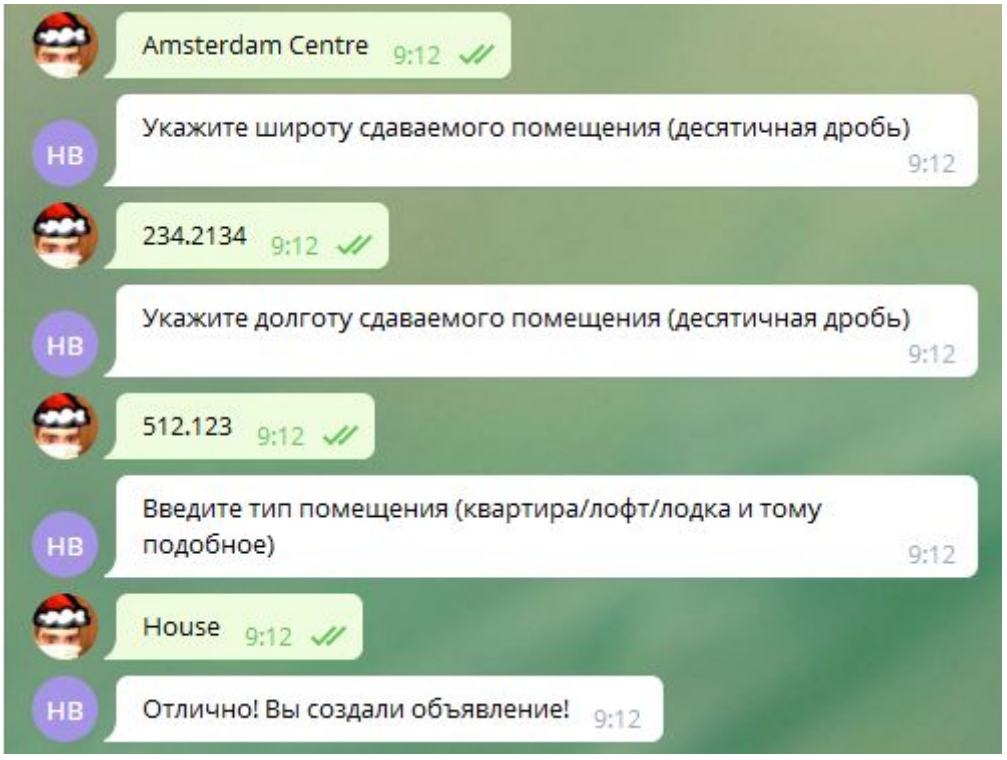


Зарегистрированный пользователь:



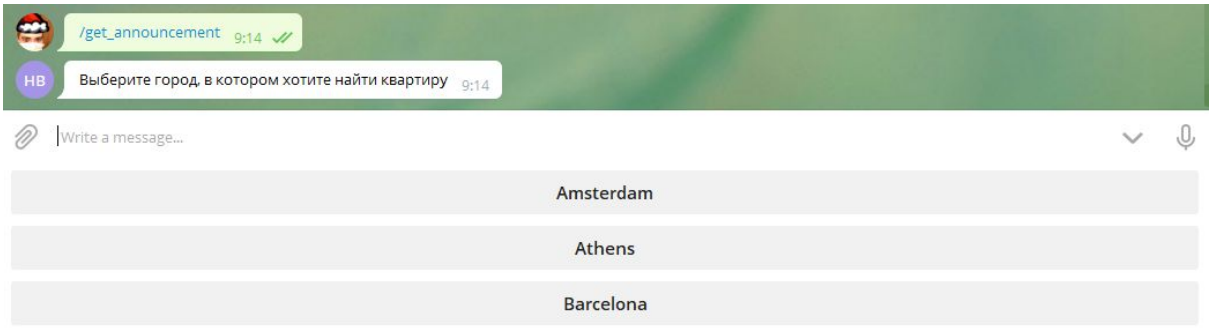
Также нужно заметить, что ввод пользователя проверяется. Если пользователь ввёл некорректные данные, ему об этом скажу и попросят заново их ввести

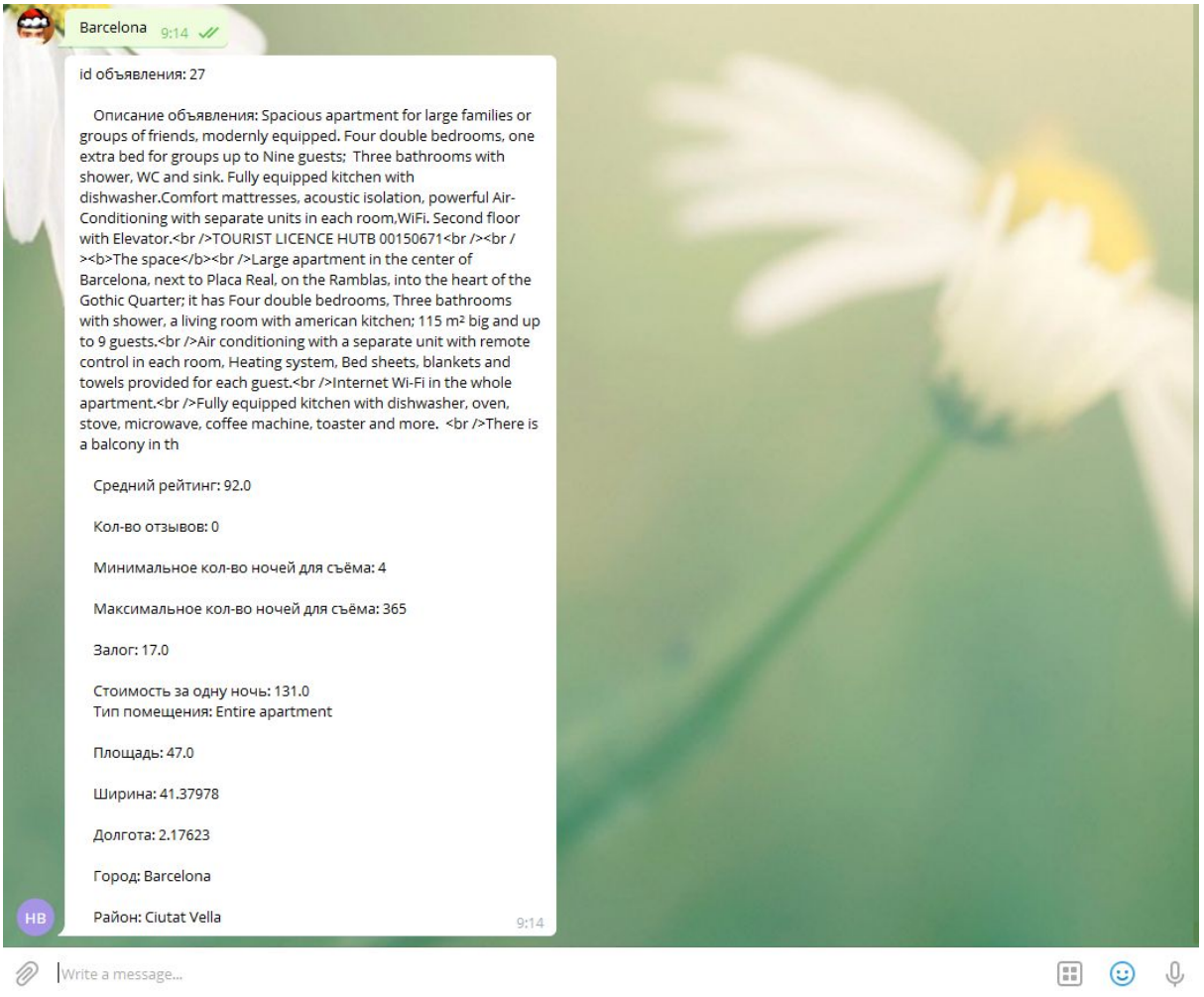




Поиск объявлений в городе

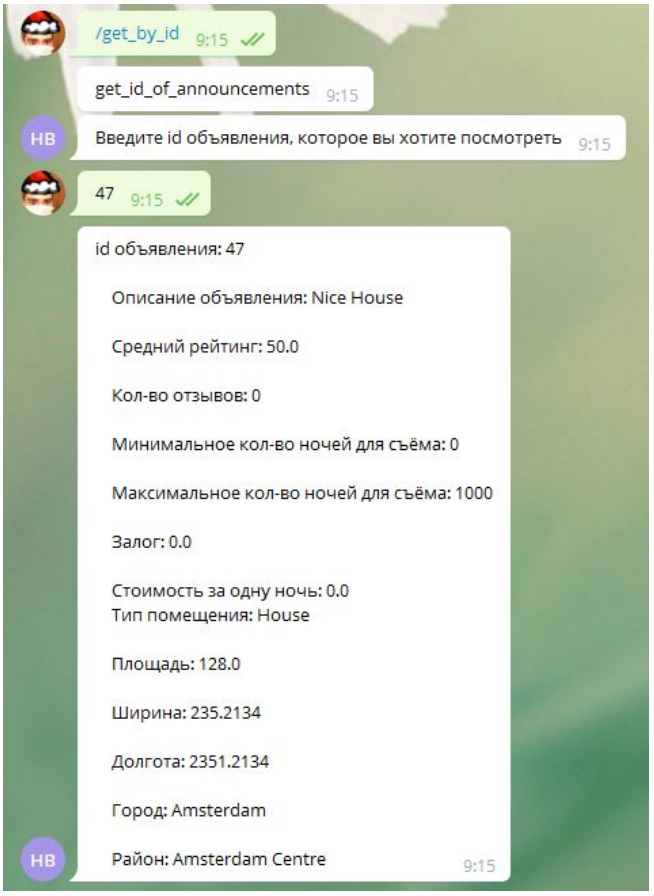
Можно сделать поиск объявлений по конкретному городу, для этого нужно воспользоваться соответствующей командой и выбрать город





Поиск конкретного объявления

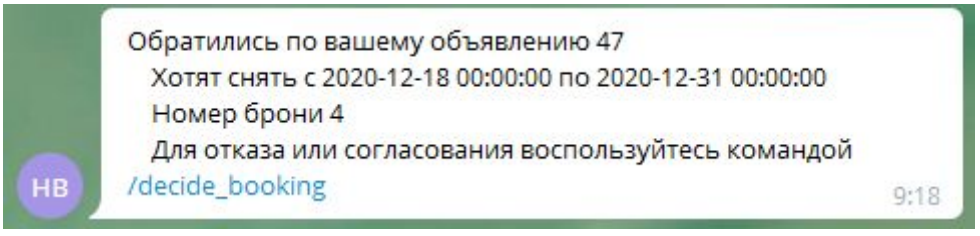
Если мы знаем id нужного объявления, мы можем попросить нам его показать:



Процесс бронирования

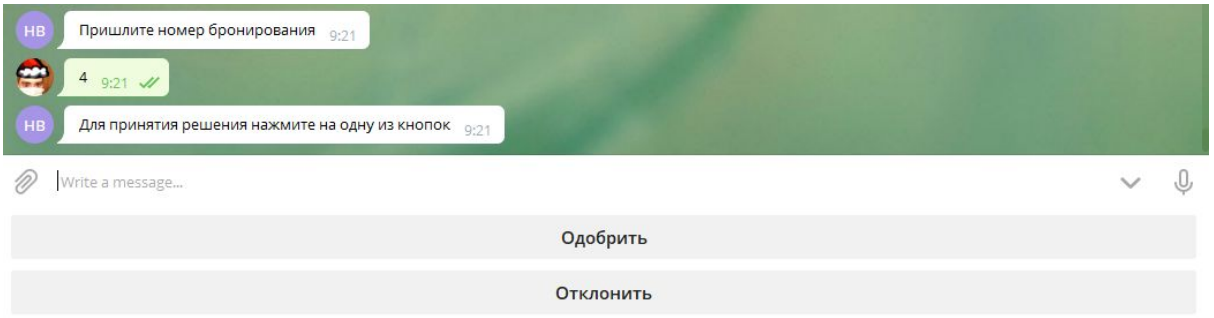
Если попалось нужное объявление, мы можем его забронировать

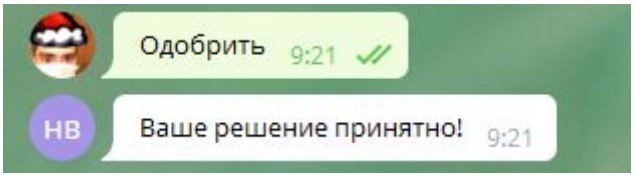
В это же время владельцу объявления приходит сообщение о том, что его помещение хотят снять



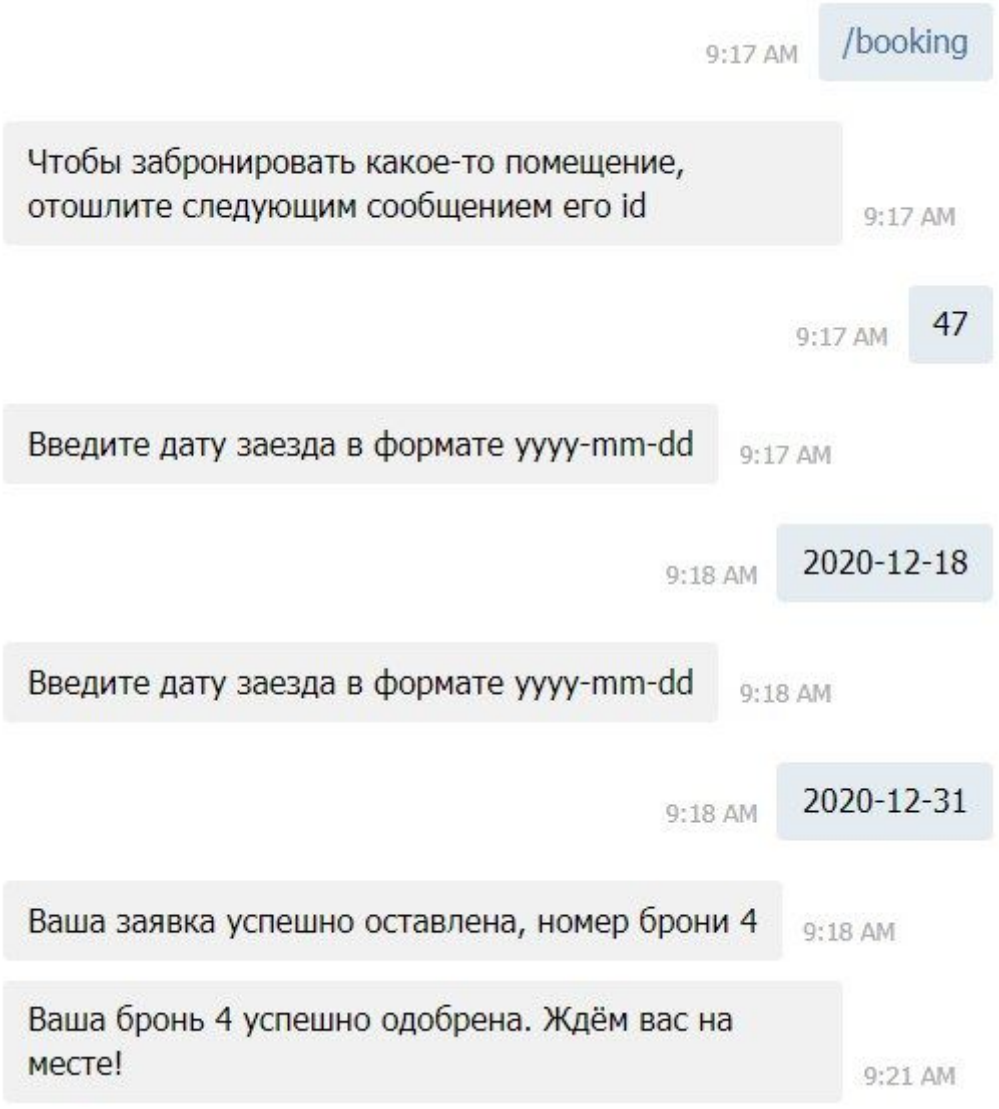
Подтверждение брони

Воспользуемся командой и подтвердим бронь





В это же время, арендатору приходит решение о его брони



Тестирование

Были протестированы следующие функции:

- для первого взаимодействия пользователя с ботом: start_message
- для получения информации о командах: help_message
- для регистрации арендодателя: landlord_reg, landlord_enter_name, landlord_set_name
- для регистрации арендателя: tenant_reg, tenant_enter_name, tenant_set_name
- для добавления нового объявления: new_announcement, begin_announcement, set_new_announcement_city, set_new_announcement_description, set_new_announcement_pledge, set_new_announcement_square, set_new_announcement_min_night, set_new_announcement_max_night, set_new_announment_suburb, set_new_announcement_latitude, set_new_announcement_longitude, set_new_announcement_type, set_new_announcement, delete_new_announcement, exists_new_announcement, set_new_announcement_description, set_new_announcement_pledge, set_new_announcement_square, set_new_announcement_min_night, set_new_announcement_max_night, set_new_announcement_city, set_new_announcement_suburb, set_new_announcement_latitude, set_new_announcement_longitude, set_new_announcement_type, copy_to_announcement_from_temp
- для получения конкретного объявления: get_id_of_announcements, get_announcements_by_id, get_announcement_by_id
- для получения объявления в каком-то городе: get_announcement, send_announcement
- для бронирования помещения: new_booking, set_id_booking, set_start_date, set_end_date
- для решения бронирования: decide_booking, decision_booking, make_decision,
- Общие функции: get_state, in_table_in_row_in_col_set_val, from_table_get_all_values_of_col, exists_row, set_state,

Заключение

Благодаря этому проекту мы получили много новых полезных знаний о SQL и базах данных. Но самое важное, что благодаря этому мы приобрели другое мышление, теперь можем думать о различных проектах именно со стороны БД. Поняли, насколько важно учитывать много нюансов и нестандартных случаев для обеспечения работы даже небольшой надёжной базы данных, стали понимать, как сложно устроены современные БД, где хранится огромное количество данных.

Личный вклад

Александр Андреевич Ширнин: участие в создании концепции БД, разработка диаграмм для БД. Создание словаря сущностей. Редакция кода для создания БД. Поиск данных для начального заполнения БД. Создание процедур и триггеров, а также написание кода для отчётов по БД.

Мячин Данил Александрович: обсуждение и участие в создании концепта БД, создание скрипта для БД, разработка телеграмм бота, тестирование работоспособности бота.

Шибанин Георгий Вячеславович: Обсуждение и участие в создании концепта БД, загрузка и форматирование данных для БД, создание хранимой процедуры, создание графиков отчётов и карт.