

## Отчет о командном проекте “AeroBot” по курсу “Теория баз данных”

Подготовили:

Мугтасимов Даниил, Журавлев Виталий,  
Тороп Виктория, Пугачев Александр

1. Содержание.
2. Техническое задание.
3. Концептуальное проектирование:
  - a. Краткое описание предметной области.
  - b. Описание процесса построения инфологической модели с обоснованием выделения сущностей и связей.
  - c. ER-диаграмма с комментариями.
4. Проектирование реляционной модели:
  - a. Описание процесса перехода к реляционной модели с акцентом на представлении связей «многие ко многим» и наследования.
  - b. Диаграмма реляционной модели (можно совместить со схемой БД, но обязательно должны быть отражены первичные и внешние ключи, ограничения и т.д.).
  - c. Диаграмма схемы БД (должны быть отражены типы данных и особенности ссылочной целостности). Желательно адекватно прорисовать диаграмму для облегчения визуального восприятия.
  - d. Дополнительные механизмы обеспечения целостности данных.
5. Развертывание БД в выбранной СУБД
  - a. DDL-скрипт создания схемы БД.
  - b. Примеры DML-операторов вставки тестовых данных.
  - c. Контроль работы ограничений целостности с примерами.
6. Разработка клиентского приложения
  - a. Бизнес-логика тестового приложения.
  - b. Техническое описание тестового приложения.
  - c. Примеры ввода и редактирования данных в БД (включая экранные формы, при наличии).
  - d. Запросы (включая SQL-операторы) и отчёты (включая примеры).  
Результаты функционального тестирования.

## 7. Заключение.

### 1. Содержание

В рамках проекта по курсу “Теория баз данных” наша команда создала реляционную базу данных и написала Телеграм-бота для получения из базы информации об авиарейсах, авиакомпаниях и аэропортах. Вся нужная информация была взята из соревнований на Kaggle и сайта GitHub:

<https://www.kaggle.com/usdot/flight-delays>

<https://github.com/quankiquanki/skytrax-reviews-dataset/tree/master/data>

Первый датасет представляет из себя информацию о совершившихся авиаперелетах всех американских авиалиний за 2015 год. Второй датасет - отзывы людей на авиакомпаниях.

### 2. Техническое задание

Требовалось придумать общую концепцию будущей базы данных, такую что построенная база данных может быть применима и полезна в каком-то определенном виде деятельности. Сначала находится подходящий датасет, на основе которого и будет создаваться база данных (БД) SQL. Для улучшения качества создания базы данных на основе логического представления о структуре будущей БД предварительно создается ER-диаграмма, которая впоследствии переводится в более приближенную к структуре БД TR-диаграмму. Чтобы пользователю было удобно использовать созданную базу данных, требовалось создать клиент (в нашем случае Телеграм-бота), который на основе определенных запросов (SQL-запросов), отправляемых к базе данных, выводит необходимую для пользователя информацию.

### 3. Концептуальное проектирование:

#### а. Краткое описание предметной области

Данная база данных содержит всю нужную информацию по аэропортам, авиакомпаниям, полетам, и предполагается, что ее использование облегчит жизнь людям, часто пользующимся услугами американских авиалиний.

По аэропортам имеется информация о городе и штате, в котором он находится, а также полное название аэропортов и их коды. Поэтому с помощью БД можно осуществить поиск аэропорта по любой из имеющихся у него позиций.

Кроме того, можно будет найти информацию о рейсе, включающую в себя номер рейса, дату, время, аэропорты, города вылета и прилета, а также предсказание по задержке данного рейса, любым удобным способом: по номеру

рейса, по названию авиакомпании или по пунктам вылета-назначения, причем в последнем случае выведется еще и карта с отмеченными городами вылета и назначения.

Информация об авиакомпаниях специально организована в более удобной для пользователя форме - в виде различных рейтингов, которые делятся на несколько категорий: рейтинг компаний, построенный на основе задержек принадлежащих ей рейсов, рейтинг по рекомендациям либо по оценкам пользователей (на выбор), для построения которого можно выбрать нужный класс (бизнес или эконом), либо нужный временной промежуток. Также, для любой из авиакомпаний можно получить отзыв вместе со всей сопутствующей информацией от пользователя (оценкой, рекомендацией, классом (бизнес или эконом)).

#### **b. Описание процесса построения инфологической модели с обоснованием выделения сущностей и связей**

Для создания БД было выделено шесть основных сущностей, которые лежат в основе всей схемы базы данных: 1) аэропорт с базовыми атрибутами, такими как уникальный код (ID), полное название, город, штат (так как будут рассматриваться только американские авиалинии) и координаты расположения, 2) конкретный самолет, который идентифицируется уникальным номером, 3) авиакомпанию, к которой самолет принадлежит с ее ID, названием и страной привязки, и 4) рейс, у которого имеется номер и прогноз задержки, предсказанный с помощью машинного обучения на основе предыдущих задержек данного рейса. Также, для того чтобы была возможность узнать качество предоставляемых услуг авиакомпанией, мы включили о них 5) отзывы, для которых будет храниться номер, дата, содержание, класс, оценка и рекомендация, а также 6) автор и информация о нем: ник и его страна происхождения.

Между указанными сущностями существуют связи. Так, у авторов есть отзыв, который может быть оставлен, как на авиакомпанию, так и на конкретный рейс, поэтому автор и отзыв соединены между собой прямой связью, а от отзыва идет разветвление и соединение с двумя сущностями: рейс и авиакомпания. У каждой авиакомпании есть какое-то количество самолетов, летающих от их имени, и кроме того каждый самолет привязан к определенному рейсу, поэтому все эти три сущности последовательно связаны между собой. В соответствии с рейсом каждый самолет вылетает из какого-то аэропорта и прилетает в какой-то другой аэропорт, поэтому рейс и аэропорт связаны между собой двумя связями: вылет, прилет, каждая из которых имеет по атрибуту: фактическое время и время по расписанию (или плановое), для того, чтобы можно было отслеживать задержки по каждому рейсу и делать предсказания, о которых было упомянуто выше.

### с. ER-диаграмма с комментариями

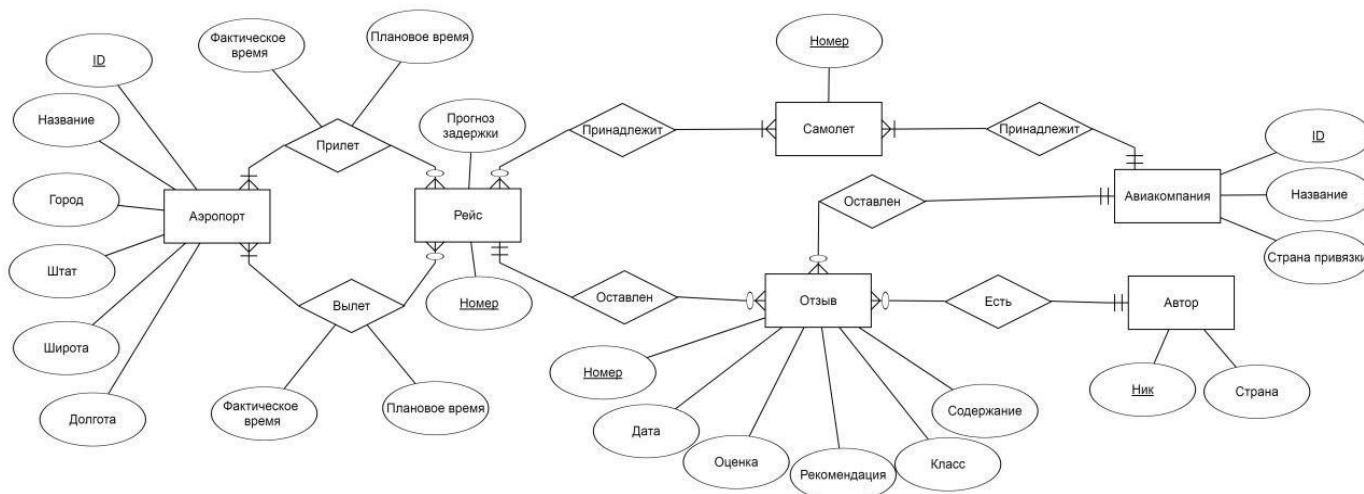
Теперь рассмотрим по очереди все связи между сущностями, а именно значения модальности и множественности.

Связь самолет – рейс:

Связь «многие ко многим», так как самолет может летать на нескольких рейсах, и на рейсе может быть несколько самолетов, модальность – 1 и 0 соответственно, так как в какой-то момент времени может быть такое, что к самолету не будет приписан рейс.

Связь самолет – авиакомпания:

Связь «один ко многим», так как одной авиакомпании обычно принадлежит несколько самолетов, при этом самолет принадлежит только одной авиакомпании, модальность – 1 и 1, так как самолет принадлежит одной



авиакомпания и у каждой авиакомпании есть хотя бы один самолет.

Связь авиакомпания – отзыв:

Связь «один ко многим», так как на одну авиакомпанию может быть несколько отзывов, при этом один отзыв не может быть на несколько авиакомпаний сразу, модальность 1 и 0 соответственно, так как на авиакомпанию может и не быть отзыва, при этом если есть отзыв, то он точно написан на одну конкретную авиакомпанию.

Связь отзыв – автор:

Связь «один ко многим», так как у отзыва может быть только один автор, но при этом у одного автора может быть несколько отзывов, модальность – 0 и 1 соответственно, так как в какой-то момент времени может быть автор, который еще не оставил свой отзыв.

Связь отзыв – рейс:

Связь «один ко многим», так как на один рейс может быть несколько отзывов пользователей, при этом конкретный отзыв может быть только на один конкретный рейс, модальность – 0 и 1 соответственно, так как в какой-то момент времени может быть рейс, на который еще не был оставлен ни один отзыв.

Связи аэропорт – рейс (аналогичные):

Связь «многие ко многим», так как каждый рейс включает в себя несколько аэропортов, а каждый аэропорт обслуживает множество рейсов, модальность – 1 и 0 соответственно, так как в какой-то момент времени к аэропорту еще могут не быть приписаны рейсы.

#### **4. Проектирование реляционной модели:**

##### **а. Описание процесса перехода к реляционной модели с акцентом на представлении связей «многие ко многим» и наследования.**

На основе составленной ER-диаграммы строится более наглядная с точки зрения представления будущей БД TR-диаграмма (реляционная модель). Каждая сущность вместе с ее атрибутами записывается в виде таблицы, так что в заголовок записывается название сущности, а в тело таблицы ее атрибуты, из которых пометкой «primary key (PK)» выделяются уникальные идентификаторы сущности. Между собой все таблицы соединяются на основе обозначенных связей в ER-диаграмме.

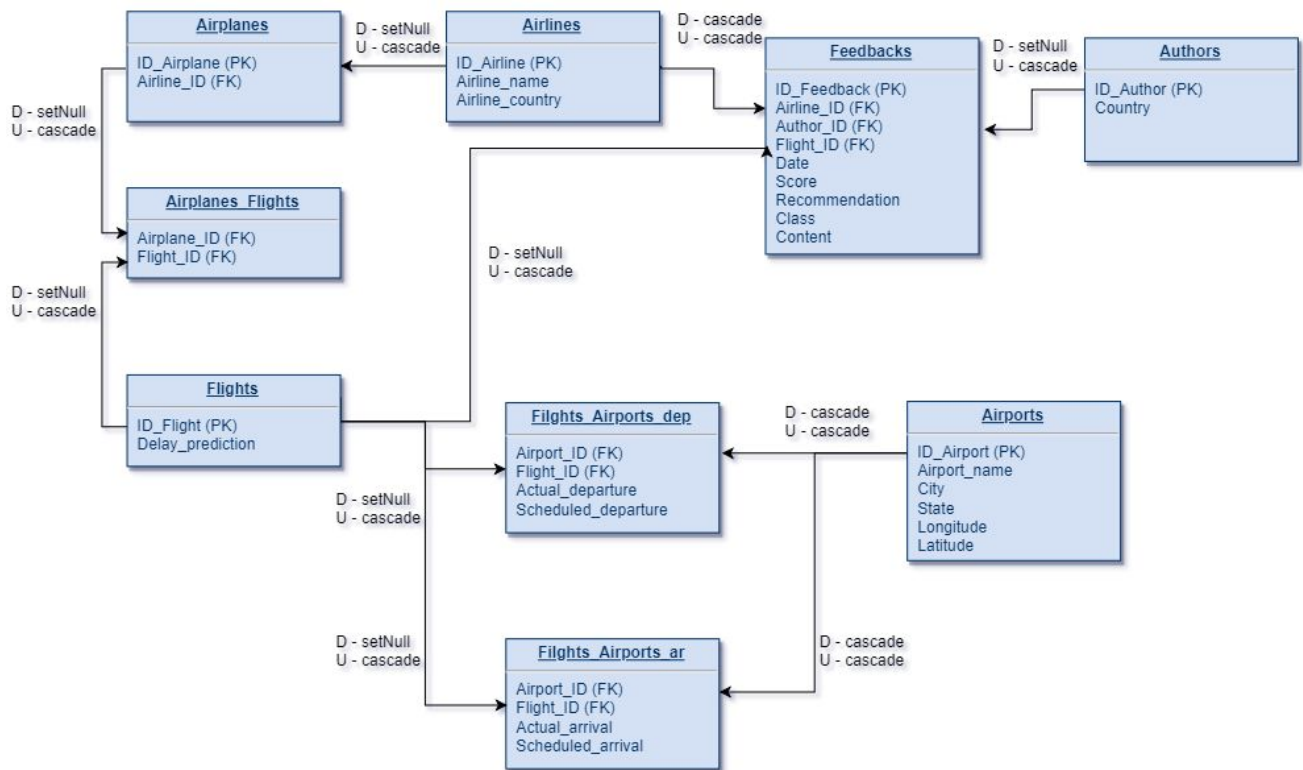
Сущности, имеющие между собой связь «один ко многим» отображаются в TR-диаграмме в виде стрелки, ведущей от сущности, имеющей множественность «один», к сущности множественности «многие», причем в таблицу последней сущности дублируется уникальный идентификатор первой сущности, но уже с пометкой «foreign key (FK)», показывающий, что это дублированный идентификатор из другой таблицы.

Сущности, имеющие связь «многие ко многим» соединяются немного иначе. Между такими двумя сущностями появляется еще одна таблица, называемая таблицей связи, заголовком которой становится соединение заголовков связываемых таблиц. В нее помещаются дублированные уникальные идентификаторы обеих сущностей с пометкой «foreign key (FK)», и обе таблицы сущностей стрелками ссылаются на таблицу связи.

##### **б. с. Диаграмма реляционной модели. Диаграмма схемы БД**

Для каждой связи в реляционной модели указываются ограничения на обновление и удаление данных, которые могут быть трех типов: «cascade» (при удалении/изменении данных во всех таблицах, содержащих эту информацию,

данные тоже удаляются или изменяются), “setNull” (оставляет все связанные данные, установив Null на удаленной позиции), “restrict” (запрещает производить действия с данными). В нашей БД все изменение данных каскадного типа. Удаление данных того же типа присутствует в связи аэропорт-рейс, так как при удалении аэропорта из базы данных, рейсы совершающиеся в уже отсутствующий аэропорт не будут иметь значения, поэтому информация о них тоже удалится. То же со связью авиакомпания-отзыв, не нужны отзывы на уже отсутствующую авиакомпанию. Для всех остальных связей поставлено ограничение на удаление данных типа каскад.



### с. Дополнительные механизмы обеспечения целостности данных.

При построении БД ставятся некоторые важные ограничения на типы и значения данных переменных. Например, в нашем случае ставится ограничение на значения “оценок” и “рекомендаций” - целое число. Также, допускается, что в базе данных фактического времени вылета/прилета может и не быть, в связи со случаями отмены рейсов. Введя данные ограничения, можно быть уверенным, что база данных будет работать правильно и без ошибок. Помимо этого существует триггер, обеспечивающий контроль на изменение данных об отзывах. Так, например, имя автора, оставившего отзыв не может быть изменено на другое имя автора, информации о котором у нас не имеется.

## 5. Развертывание БД в выбранной СУБД

а. DDL-скрипт создания схемы БД.

```
1. CREATE TABLE IF NOT EXISTS Airports (  
2.     ID_Airport text PRIMARY KEY,  
3.     Airport_name text NOT NULL UNIQUE,  
4.     City text NOT NULL,  
5.     State text NOT NULL,  
6.     Longitude real NOT NULL,  
7.     Latitude real NOT NULL  
8. );  
9.  
10. CREATE TABLE IF NOT EXISTS Flights (  
11.     ID_Flight int PRIMARY KEY,  
12.     Delay_prediction real NOT NULL  
13. );  
14.  
15. CREATE TABLE IF NOT EXISTS Flights_Airports_dep (  
16.     ID_Airport text,  
17.     ID_Flight text,  
18.     Actual_departure text,  
19.     Scheduled_departure text NOT NULL,  
20.     PRIMARY KEY (ID_Airport, ID_Flight, Actual_departure, Scheduled_departure),  
21.     FOREIGN KEY (ID_Airport) REFERENCES Airports (ID_Airport)  
22.     ON DELETE CASCADE ON UPDATE CASCADE,  
23.     FOREIGN KEY (ID_Flight) REFERENCES Flights (ID_Flight)  
24.     ON DELETE SET NULL ON UPDATE CASCADE  
25. );  
26.  
27. CREATE TABLE IF NOT EXISTS Flights_Airports_ar (  
28.     ID_Airport text,  
29.     ID_Flight text,  
30.     Actual_arrival text,  
31.     Scheduled_arrival text NOT NULL,  
32.     PRIMARY KEY (ID_Airport, ID_Flight, Actual_arrival, Scheduled_arrival),  
33.     FOREIGN KEY (ID_Airport) REFERENCES Airports (ID_Airport)
```

```

34.     ON DELETE CASCADE ON UPDATE CASCADE,
35.     FOREIGN KEY (ID_Flight) REFERENCES Flights (ID_Flight)
36.     ON DELETE SET NULL ON UPDATE CASCADE
37. );
38.
39. CREATE TABLE IF NOT EXISTS Airlines (
40.     ID_Airline text PRIMARY KEY,
41.     Airline_name text NOT NULL UNIQUE,
42.     Airline_country text NOT NULL
43. );
44.
45. CREATE TABLE IF NOT EXISTS Airplanes (
46.     ID_Airplane text PRIMARY KEY,
47.     ID_Airline text,
48.     FOREIGN KEY (ID_Airline) REFERENCES Airlines (ID_Airline)
49.     ON DELETE SET NULL ON UPDATE CASCADE
50. );
51.
52. CREATE TABLE IF NOT EXISTS Airplanes_Flights (
53.     ID_Airplane text,
54.     ID_Flight text,
55.     PRIMARY KEY (ID_Airplane, ID_Flight),
56.     FOREIGN KEY (ID_Airplane) REFERENCES Airplanes (ID_Airplane)
57.     ON DELETE SET NULL ON UPDATE CASCADE,
58.     FOREIGN KEY (ID_Flight) REFERENCES Flights (ID_Flight)
59.     ON DELETE SET NULL ON UPDATE CASCADE,
60.     UNIQUE(ID_Airplane, ID_Flight)
61. );
62.
63. CREATE TABLE IF NOT EXISTS Authors (
64.     ID_Author text PRIMARY KEY,
65.     Country text
66. );

```



```

66. );
67.
68. CREATE TABLE IF NOT EXISTS Feedbacks (
69.     ID_Feedback integer PRIMARY KEY AUTOINCREMENT,
70.     ID_Airline text,
71.     ID_Author text,
72.     ID_Flight text,
73.     Date text NOT NULL,
74.     Score integer,
75.     Recommendation integer,
76.     Class text,
77.     Content text,
78.     FOREIGN KEY (ID_Airline) REFERENCES Airlines (ID_Airline)
79.     ON DELETE CASCADE ON UPDATE CASCADE,
80.     FOREIGN KEY (ID_Author) REFERENCES Authors (ID_Author)
81.     ON DELETE SET NULL ON UPDATE CASCADE,
82.     FOREIGN KEY (ID_Flight) REFERENCES Flights (ID_Flight)
83.     ON DELETE SET NULL ON UPDATE CASCADE
84.     UNIQUE(ID_Airline, ID_Author, ID_Flight, Date)
85. );
86.
87. CREATE TRIGGER IF NOT EXISTS validate_feedback_rec_before_insert BEFORE UPDATE ON Feedbacks
88. BEGIN
89.     SELECT CASE
90.     WHEN ((SELECT Feedbacks.ID_Author FROM Feedbacks WHERE Feedbacks.ID_Author = NEW.ID_Author ) ISNULL)
91.     THEN RAISE(ABORT, 'This is an User Define Error Message - This Author_id does not exist.')
92.     END;
93. END;

```

#### b. Примеры DML-операторов вставки тестовых данных.

```

1. INSERT INTO Authors (
2.     ID_Author,
3.     Country
4. )
5. VALUES
6.     ('Michael Jackson',
7.     'United States'
8. )
9.
10. INSERT INTO Airports (
11.     ID_Airport,
12.     Airport_name,
13.     City,
14.     State,
15.     Longitude,
16.     Latitude
17. )
18. VALUES
19.     ('ABE',
20.     'Lehigh Valley International Airport',
21.     'Allentown',
22.     'PA',
23.     40.65236,
24.     -75.4404
25. )

```

#### с. Контроль работы ограничений целостности с примерами.

- i. *Повторное добавление одинаковой информации (ограничения первичного ключа).*

Поскольку для удобства заполнения таблицы Feedbacks мы использовали функцию AUTOINCREMENT (см. DDL скрипт), то появляется необходимость отслеживать “случайные” повторные добавления отзывов. Это возможно, потому что функция AUTOINCREMENT присвоит повторному отзыву новое уникальное значение. Для этого случае дополнительно отслеживается уникальность оставшихся столбцов. Попробуем дважды внести один и тот же отзыв:

```
1 INSERT INTO Feedbacks (  
2     ID_Airline,  
3     ID_Author,  
4     ID_Flight,  
5     Date,  
6     Score,  
7     Recommendation,  
8     Class,  
9     Content  
10 )  
11 VALUES  
12     ('AA',  
13     'Katy Montanez',  
14     'A5312',  
15     '04-15-2015',  
16     '4',  
17     '2',  
18     'economy',  
19     'So good! Thx!'  
20 )  
21
```

Статус

[02:06:26] Ошибка при выполнении SQL запроса к базе данных 'pythonsqlite': UNIQUE constraint failed: Feedbacks.ID\_Airline, Feedbacks.ID\_Author, Feedbacks.ID\_Flight, Feedbacks.Date

- ii. *Значения “NULL” в недопустимых для этого полях.*

Предполагается, что отзывы клиентов собираются в режиме “онлайн”, а значит всегда можно отследить текущее время при котором пользователь оставляет отзыв. В приведенном ниже примере демонстрируется добавление данных, содержащих пропуск в значении “Дата”, на что БД реагирует соответствующее.

```
Запрос
1 INSERT INTO Feedbacks (
2   ID_Feedback,
3   ID_Airline,
4   ID_Author,
5   ID_Flight,
6   Date,
7   Score,
8   Recommendation,
9   Class,
10  Content
11 )
12 VALUES
13  ( '23858',
14    'AA',
15    'Katy Montanez',
16    'AS312',
17    Null,
18    5,
19    1,
20    'Economy',
21    'So good! Thx!'
22 )
23
```

```
Статус
[02:19:17] Ошибка при выполнении SQL запроса к базе данных 'pythonsqlite': NOT NULL constraint failed: Feedbacks.Date
```

iii. *Добавление нового внешнего ключа, не являющегося первичным ключом в соответствующей таблице.*

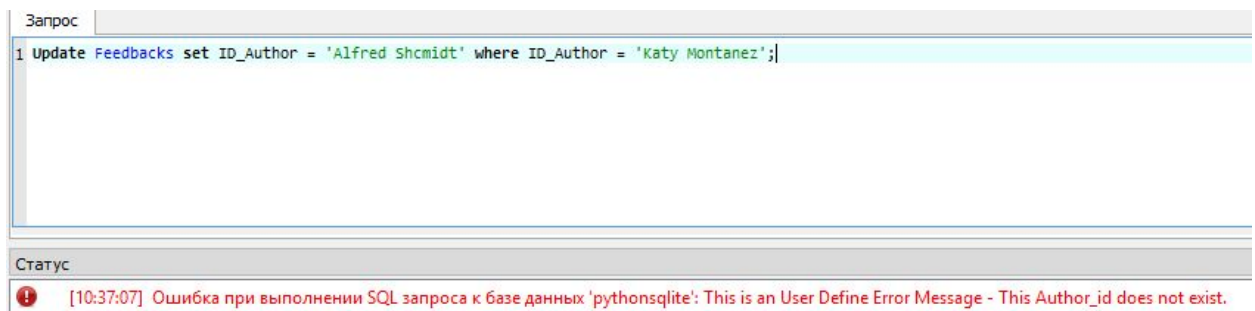
Допустим, пользователь хочет добавить отзыв о рейсе и вводит несуществующее название авиакомпании. Поскольку название авиакомпании является внешним ключом таблицы Airlines, то необходимо, чтобы такая запись присутствовала в ней. Так как пользователь ошибся, такой записи быть не должно, на что БД реагирует соответствующим образом:

```
1 INSERT INTO Feedbacks (
2   ID_Feedback,
3   ID_Airline,
4   ID_Author,
5   ID_Flight,
6   Date,
7   Score,
8   Recommendation,
9   Class,
10  Content
11 )
12 VALUES
13  ( '23858',
14    'A1',
15    'Katy Montanez',
16    'AS312',
17    '01-01-2016',
18    5,
19    1,
20    'Economy',
21    'So good! Thx!'
22 )
23
```

```
Статус
[02:41:44] Ошибка при выполнении SQL запроса к базе данных 'pythonsqlite': FOREIGN KEY constraint failed
```

#### IV. *Контроль вводимых данных при изменении таблицы.*

Допустим, необходимо изменить имя автора, оставившего отзыв. В таком случае специальный триггер (см ddl скрипт 87 строка) будет заранее проверять наличие информации об авторе в таблице Authors. Если же мы хотим изменить информацию на “несуществующее” имя, то БД выдаст ошибку 'This is an User Define Error Message - This Author\_id does not exist'.



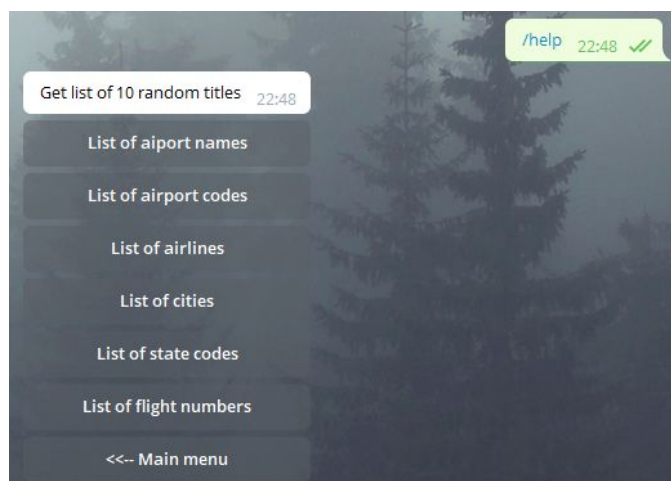
## 6. Разработка клиентского приложения

### а. Бизнес-логика тестового приложения.

Тестовое клиентское приложение представляет из себя Телеграм-бота, запущенного на локальном сервере и обращающегося к локальной SQL базе данных.

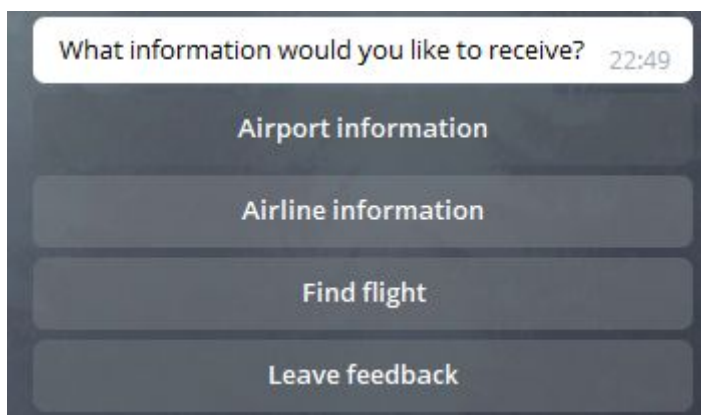
При открытии бота пользователю предоставляется выбор - начать использование бота написав или нажав /start или получить помощь написав или нажав /help.

При нажатии на /help пользователь получает списки имеющихся в нашей базе данных объектов: названий аэропортов, кодов аэропортов, названий авиалиний, городов и т. д:

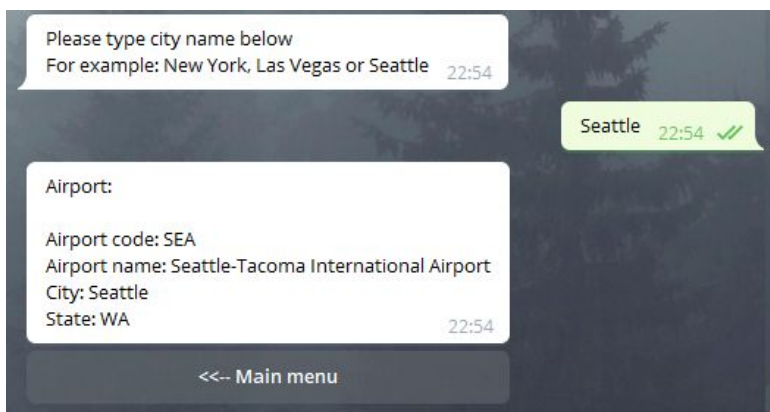


Меню /start и /help могут быть вызваны в любой момент использования бота. Также всегда доступны кнопки возвращения на 1 шаг назад и в главное меню.

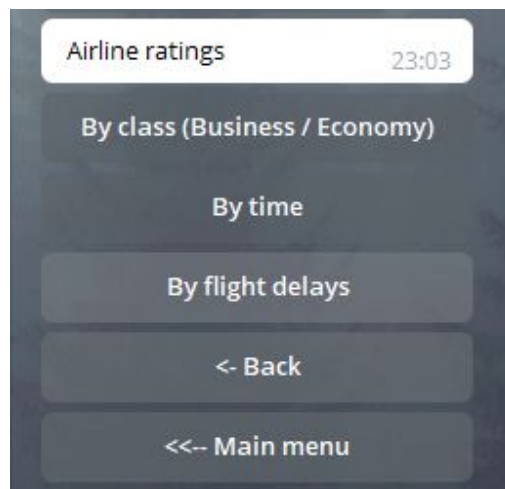
При нажатии на /start пользователю предлагаются варианты: получить информацию об аэропорте, авиалинии, найти рейс или написать отзыв на рейс:



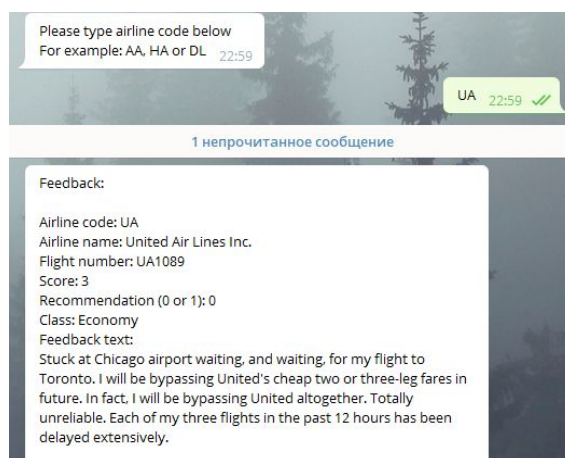
При нажатии кнопки получения информации об аэропорте пользователю предложат получить информацию об аэропорте по названию, по коду, вывести список аэропортов конкретного города или штата. После выбора того или иного варианта, пользователю предложат ввести соответствующее название, например так выглядит вывод информации о всех аэропортах Сиэтла:



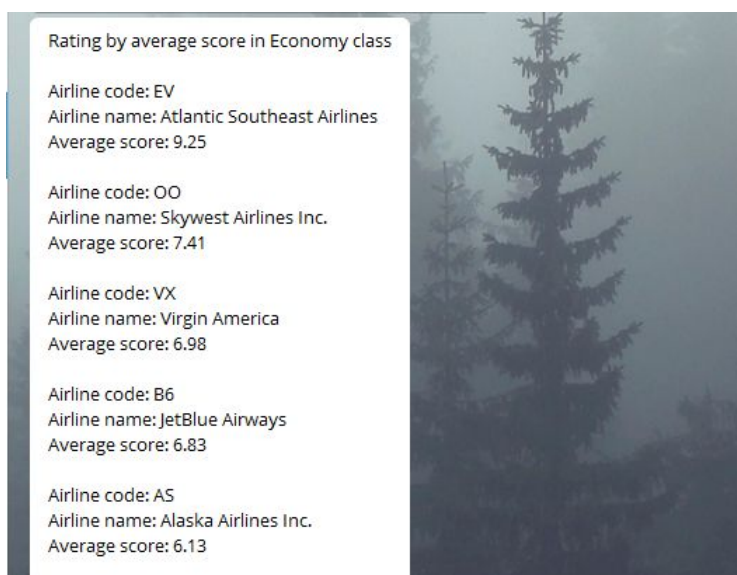
При нажатии на кнопку получения информации об авиалинии пользователю предлагается узнать рейтинг авиалиний в зависимости от класса (бизнес или эконом), времени (за последнюю неделю, месяц или 2 месяца) или задержек рейсов. Также предлагается узнать отзывы о той или иной авиакомпании. Рейтинги базируются на оценках пользователей (от 0 до 10) и от рекомендаций пользователей (0 или 1).



Например, так выглядит отзыв о United Airlines:

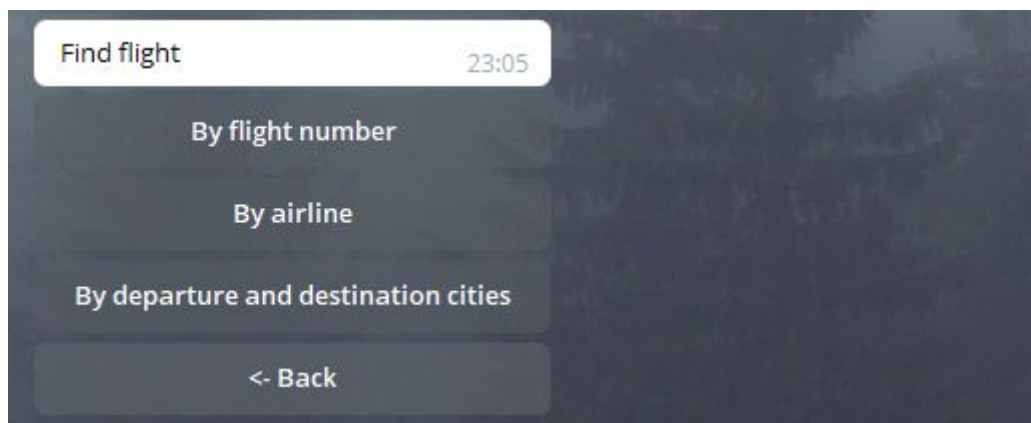


Так выглядит рейтинг авиалиний, основанный на оценках пользователей, летающих эконом-классом:

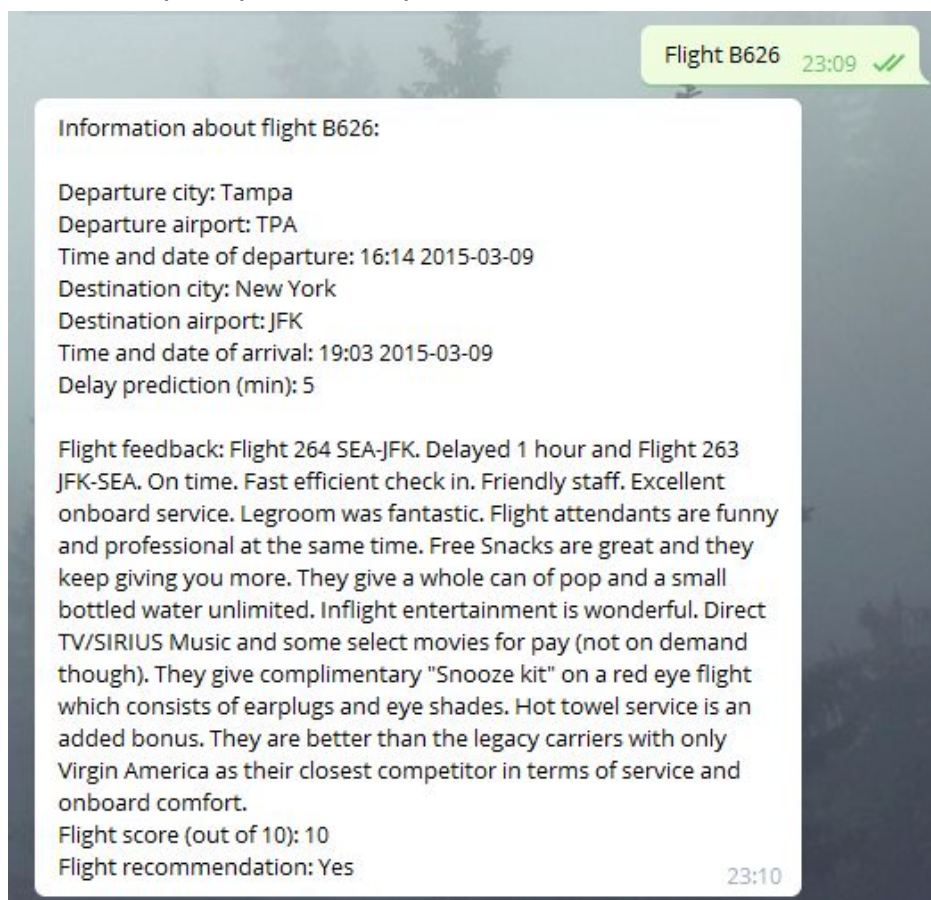




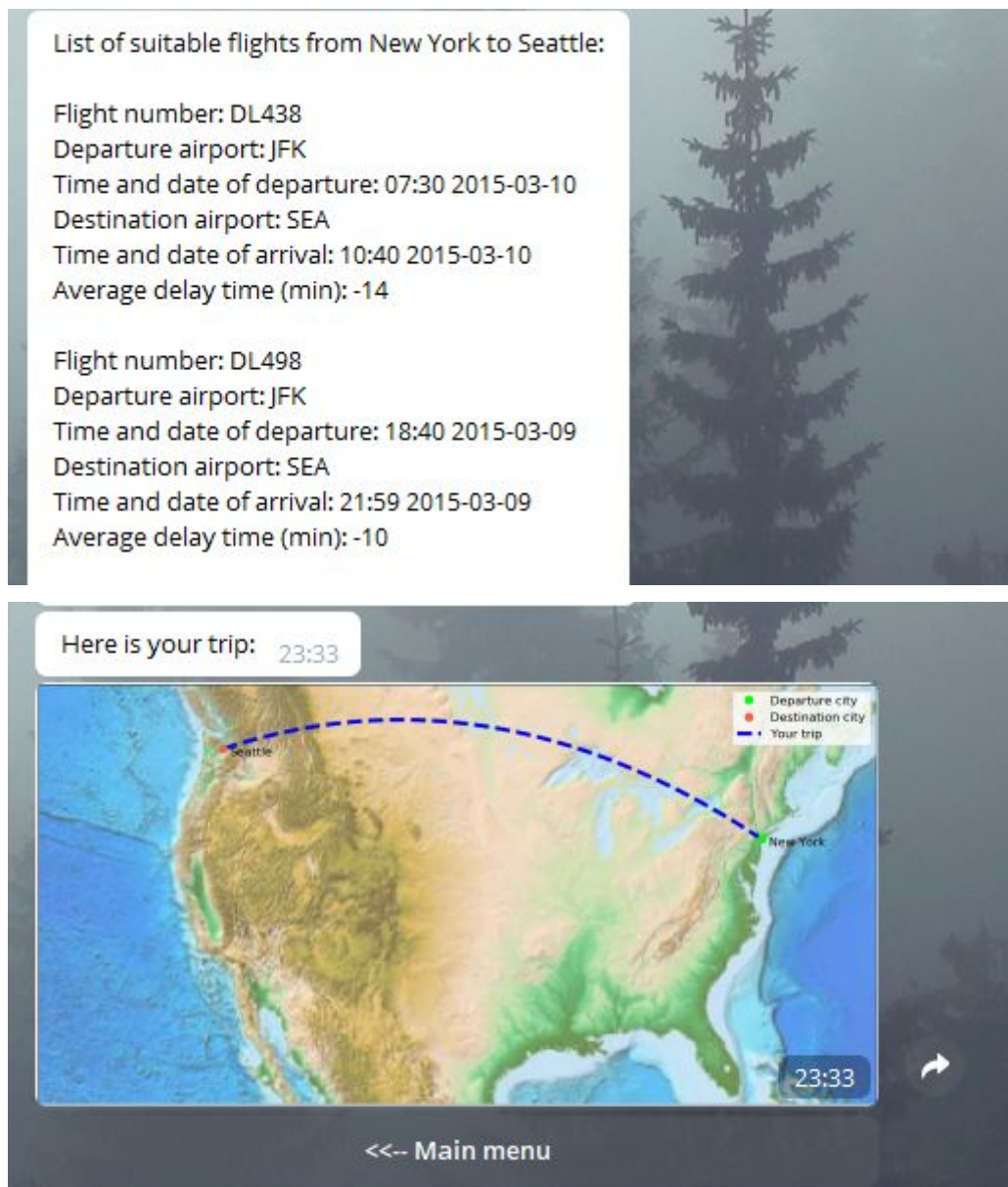
Выбрав третий пункт главного меню - поиск рейса пользователю предлагается выбрать способ поиска рейса - по номеру рейса, по авиакомпании или по пунктам отправления и назначения:



После нажатия на поиск рейса по номеру пользователю предлагается ввести номер рейса. После этого пользователю будет выведена информация об этом рейсе - время, аэропорт, город отправления / прибытия, среднее время задержки рейса, прогноз задержки рейса в минутах, отзыв об этом рейсе, оценка и рекомендация. Пример вывода приведен ниже:



При нажатии на поиск рейса по авиакомпании будет выведены все рейсы для определенной авиакомпании. При нажатии на поиск рейса по пунктам отправления и назначения будет выведен список подходящих рейсов, а также карта-маршрут рейса:



При нажатии на кнопку оставить отзыв в главном меню пользователю будет предложено последовательно ввести его имя, страну, номер рейса, оценку, рекомендацию и содержание отзыва для его дальнейшей записи в базу данных.

## **b. Техническое описание тестового приложения.**

Для написания Телеграм-бота использовался следующий инструментарий:

Язык программирования:

Python 3.6.3

Библиотеки:



sqlite3 3.14.2  
pyTelegramBotAPI 3.6.6  
telebot 0.0.3  
numpy 1.15.2  
matplotlib 2.1.0  
basemap 1.2.0

Изначально для создания бота требуется получить уникальный ключ - токен, с помощью которого осуществляется работа с ботом. С помощью библиотек pyTelegramBotAPI и telebot происходит полное управление и взаимодействие с ботом. С помощью библиотеки sqlite3 происходит подключение к базе данных и отправление к ней запросов.

Каждая кнопка представляет из себя кнопку типа Inline с уникальным callback идентификатором. Callback идентификатор служит для навигации пользователя по меню. При нажатии на определенную кнопку становится актуальным ее callback-идентификатор и впоследствии обрабатывается именно он. Например, при нажатии на кнопку "Airport name" в главном меню сработает callback "airport\_name", который в свою очередь вызовет сообщение с просьбой пользователя ввести название аэропорта.

Любой вводимый пользователем текст по умолчанию обрабатывается единственным обработчиком - самым первым. Но каждому обработчику можно представить функцию, таким образом он будет срабатывать только в тех случаях, когда функция выполняет истинное значение. Например, если пользователь вводит название аэропорта и это название есть в списке всех аэропортов, обработчик отреагирует на введенный текст, в противном случае - нет. Аналогично можно проверять подходит ли введенный пользователем текст под определенный шаблон или нет. Примечательно, что функции не чувствительны к регистру, то есть можно вводить текст как строчными, так и прописными буквами.

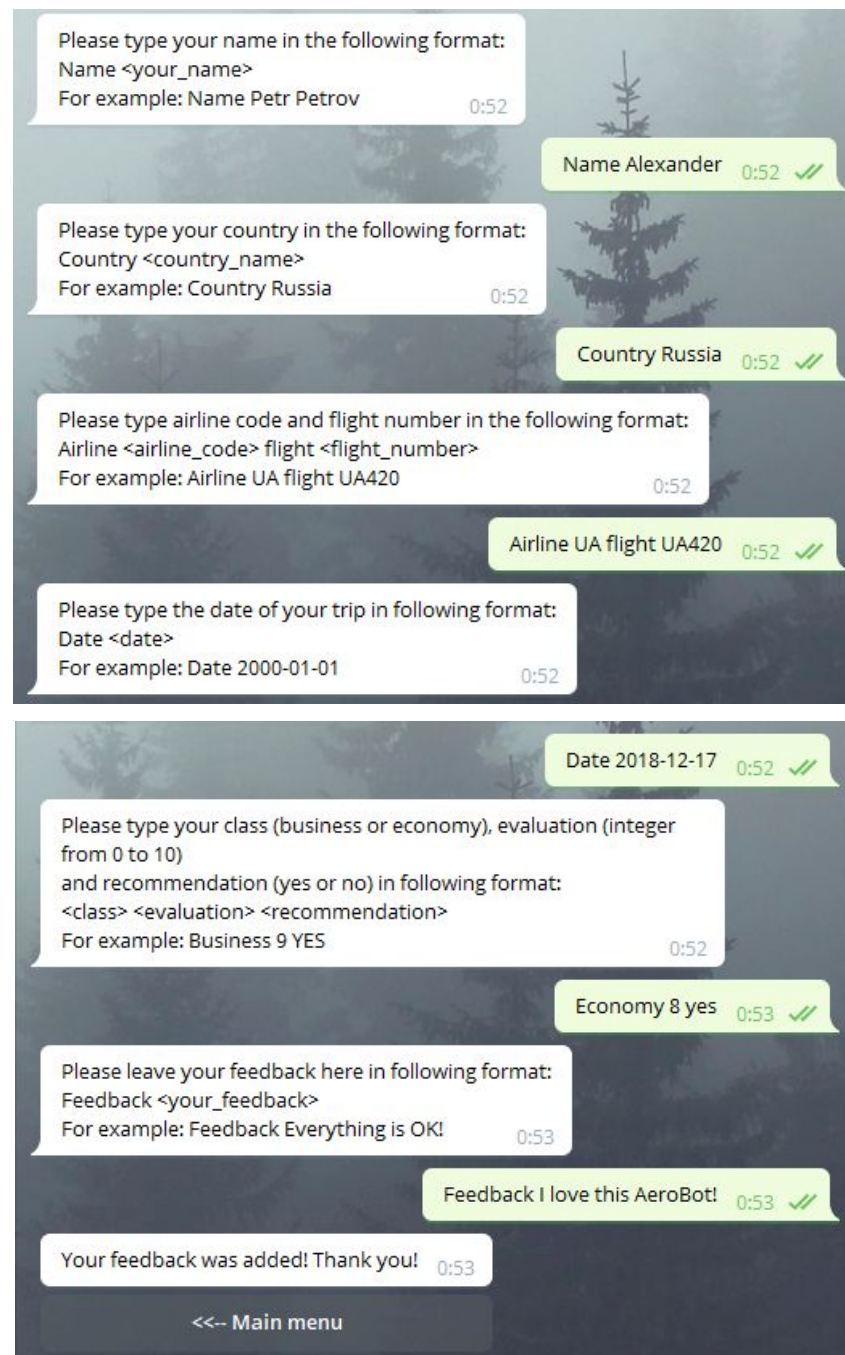
Исходя из нажатых пользователем кнопок и введенного пользователем текста, формируется SQL-запрос в виде строки, который впоследствии подается на вход методу execute библиотеки sqlite3. Возвращаемый список из элементов базы данных парсится соответствующими функциями и затем выводится пользователю в виде сообщения от бота.

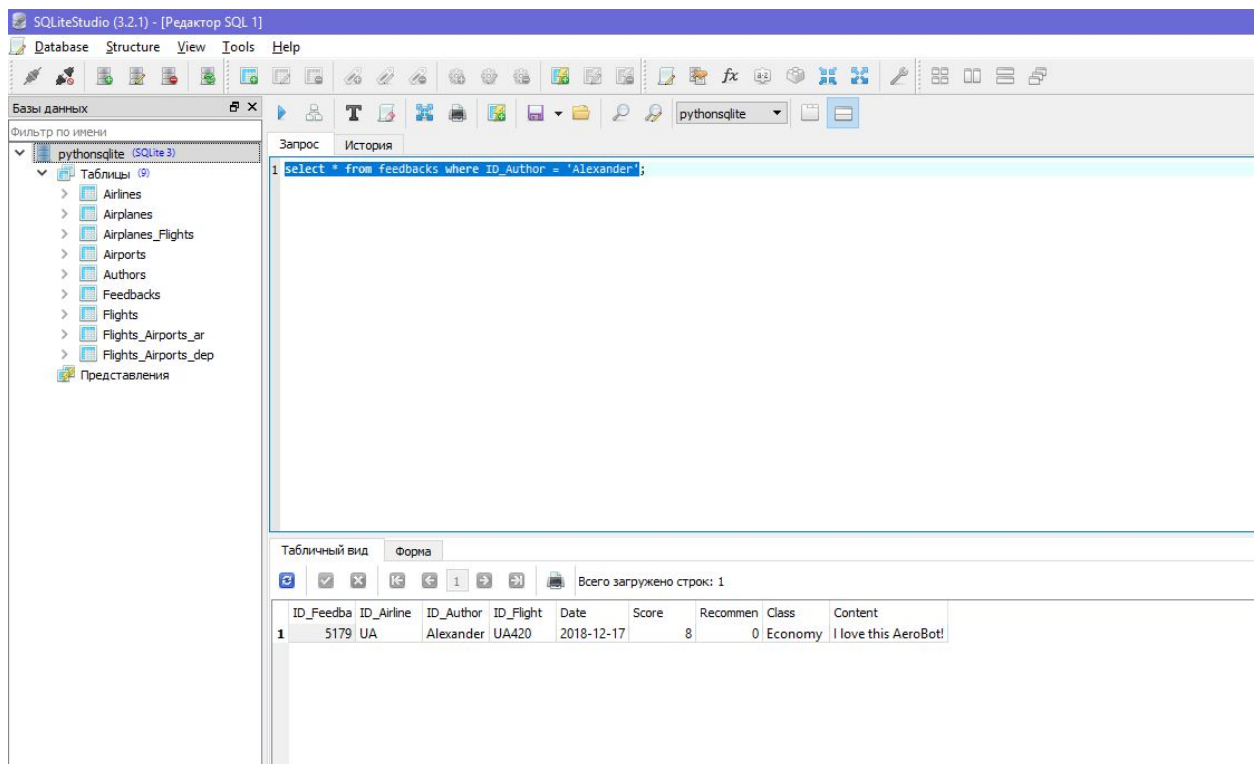
Отдельно рассмотрим создание и отрисовку карты с маршрутом. После выполнения запроса с указанными городами (условно) А и Б, выполняется запрос, который возвращает координаты данных городов на земном шаре. Далее с помощью библиотеки basemap рисуется карта Америки, затем с помощью библиотеки matplotlib отрисовываются на карте города А и Б. Далее, положив, что линия, соединяющая два города представляет из себя параболу, найдя третью точку на определенном расстоянии от центра отрезка, соединяющего города А и Б, найдем коэффициенты параболы. В конечном итоге, с помощью библиотеки matplotlib нанесем часть параболы на карту.

Также, отдельно опишем процедуру получения предсказаний времени полета для каждого рейса. Для каждого уникального рейса был получен набора разностей между фактическим временем прилета и временем прилета по расписанию. Для каждого уникального рейса была обучена линейная регрессия на  $n$  задержках, соответствующих  $n$  полетам. Далее предсказывалось значение времени задержки  $n+1$ -го полета.

**с. Примеры ввода и редактирования данных в БД (включая экранные формы, при наличии).**

Приведем пример ввода отзыва в базу данных:





Как можно видеть, отзыв добавляется в нашу базу данных.

#### d. Запросы (включая SQL-операторы) и отчёты (включая примеры). Результаты функционального тестирования.

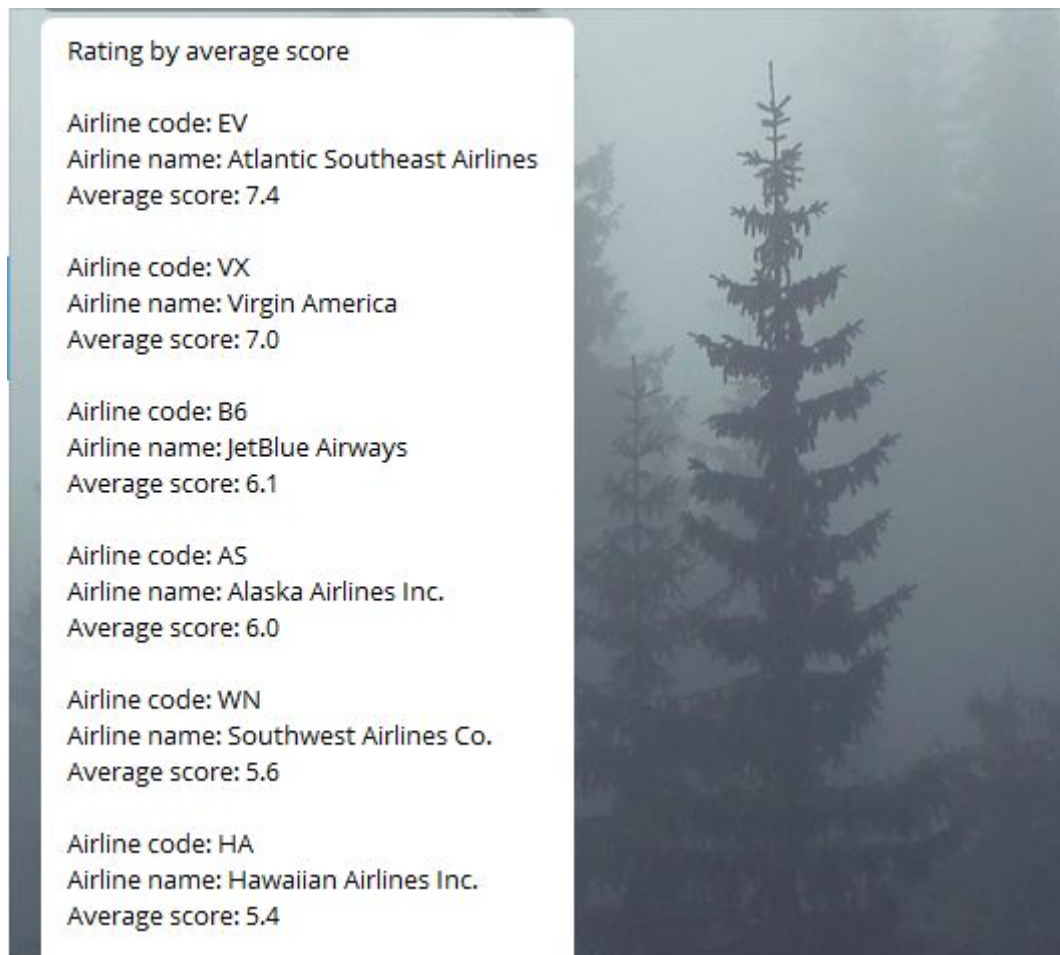
Выберем 5 наиболее сложных с нашей точки зрения SQL-запросов и продемонстрируем как они работают:

1. Рейтинг авиалиний на основе оценок пользователей за последний месяц:

Запрос:

```
SELECT a.ID_Airline, Airline_name, ROUND(AVG(Cast(Score as Float)), 1) As
Rating
FROM Airlines AS a INNER JOIN Feedbacks AS f
ON a.ID_Airline = f.ID_Airline WHERE date(f.Date, '+1 month') > (SELECT
MIN(Date) FROM Feedbacks) GROUP BY a.ID_Airline ORDER BY Rating
DESC
```

Вывод:



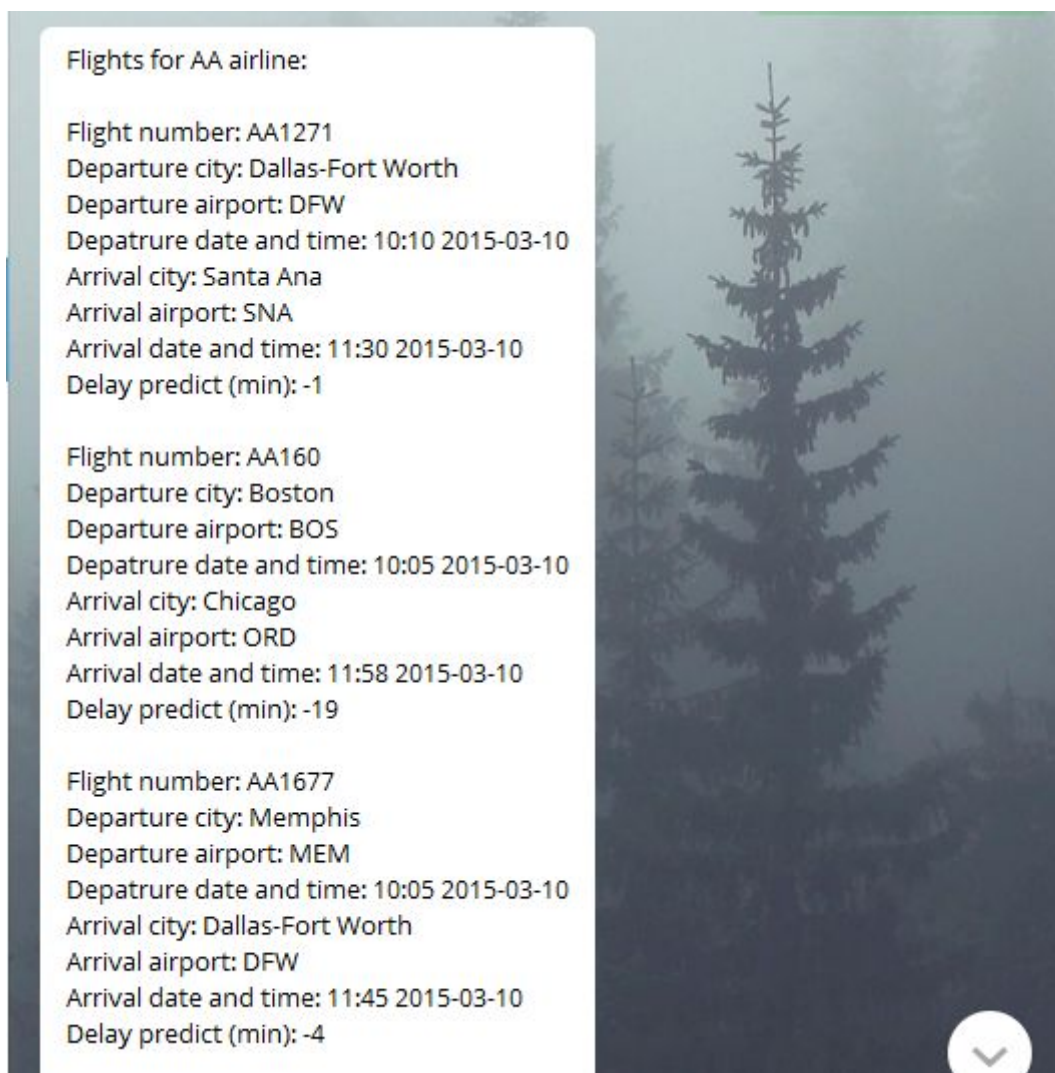
## 2. Информация о рейсах авиакомпании с кодом AA:

Запрос:

```
SELECT DISTINCT t.ID_flight, t.Airport_dep, t.Airport_ar,  
ROUND(Delay_Prediction) AS Delay_Prediction,  
a1.City AS City1, t.Scheduled_departure, a2.City AS City2,  
t.Scheduled_arrival  
FROM (  
SELECT dep.ID_flight, dep.ID_Airport AS Airport_dep, dep.Actual_departure,  
dep.Scheduled_departure,  
ar.ID_Airport AS Airport_ar, ar.Actual_arrival, ar.Scheduled_arrival  
FROM Flights_Airports_dep AS dep  
INNER JOIN Flights_Airports_ar AS ar ON dep.rowid = ar.rowid) AS t  
INNER JOIN Flights AS f ON f.ID_Flight = t.ID_Flight  
INNER JOIN Airports AS a1 ON a1.ID_Airport = t.Airport_dep  
INNER JOIN Airports AS a2 ON a2.ID_Airport = t.Airport_ar  
WHERE f.ID_flight LIKE 'AA%'
```

```
GROUP BY f.ID_flight
HAVING Delay_prediction <= 0
ORDER BY t.Scheduled_departure DESC
LIMIT 5
```

Вывод:



### 3. Информация о рейсе B626:

Запрос:

```
SELECT DISTINCT t.*, ROUND(Delay_Prediction) AS Delay_Prediction, a1.City
AS City1, a2.City AS City2,
ID_Author, Date, Score, Recommendation, Class, Content
FROM (
    SELECT dep.ID_flight, dep.ID_Airport AS Airport_dep,
    dep.Scheduled_departure,
```

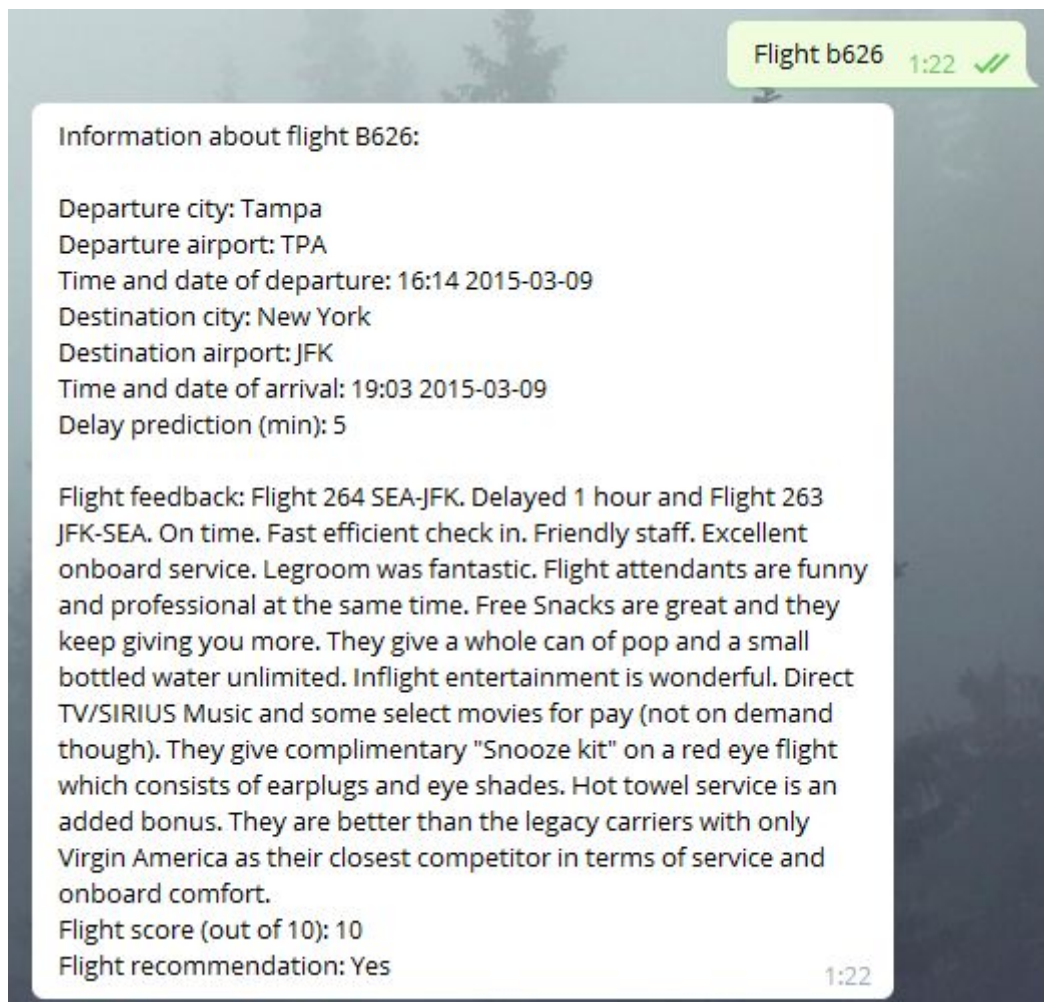


```

ar.ID_Airport AS Airport_ar, ar.Scheduled_arrival
FROM Flights_Airports_dep AS dep
INNER JOIN Flights_Airports_ar AS ar ON dep.rowid = ar.rowid) AS t
INNER JOIN Flights AS f ON f.ID_Flight = t.ID_Flight
INNER JOIN Airports AS a1 ON a1.ID_Airport = t.Airport_dep
INNER JOIN Airports AS a2 ON a2.ID_Airport = t.Airport_ar
LEFT JOIN Feedbacks AS r ON f.ID_Flight = r.ID_Flight
WHERE t.ID_flight = 'B626'
GROUP BY t.Scheduled_departure
ORDER BY t.Scheduled_departure DESC
LIMIT 1

```

Вывод:



#### 4. Список рейсов, летающих из Денвера в Нью Йорк:

Запрос:

```

SELECT DISTINCT t.*, ROUND(Delay_Prediction) AS Delay_Prediction, a1.City
AS City1, a1.Longitude AS long1, a1.Latitude AS lat1, a2.City AS City2, a2.Longitude
AS long2, a2.Latitude AS lat2
FROM (
    SELECT dep.ID_flight, dep.ID_Airport AS Airport_dep, dep.Actual_departure,
    dep.Scheduled_departure,
        ar.ID_Airport AS Airport_ar, ar.Actual_arrival, ar.Scheduled_arrival
    FROM Flights_Airports_dep AS dep
    INNER JOIN Flights_Airports_ar AS ar ON dep.rowid = ar.rowid) AS t
INNER JOIN Flights AS f ON f.ID_Flight = t.ID_Flight
INNER JOIN Airports AS a1 ON a1.ID_Airport = t.Airport_dep
INNER JOIN Airports AS a2 ON a2.ID_Airport = t.Airport_ar
WHERE a1.City = 'Denver' AND a2.City = 'New York'
ORDER BY t.Scheduled_departure DESC
LIMIT 5

```

Вывод:



## 5. Рейтинг авиакомпаний по задержкам рейсов:

Запрос:



```

SELECT DISTINCT q.Airline_code, Airline_name, q.Avg_delay, q.Avg_prediction
FROM (
    SELECT f.ID_Flight, substr(f.ID_Flight, 1, 2) AS Airline_code,
        ROUND(AVG(JulianDay(ar.Actual_arrival) -
JulianDay(ar.Scheduled_arrival)) * 24 * 60) AS Avg_delay,
        ROUND(AVG(Delay_prediction)) AS Avg_prediction
    FROM Flights AS f
    INNER JOIN Flights_Airports_ar AS ar ON f.ID_Flight = ar.ID_Flight
    WHERE time(ar.Actual_arrival) != '00:00:00' AND (time(ar.Actual_arrival) -
time(ar.Scheduled_arrival)) BETWEEN -12 AND 12
    GROUP BY substr(f.ID_Flight, 1, 2)
    HAVING Avg_delay <= 10) AS q
INNER JOIN Airplanes_Flights AS af ON q.ID_Flight = af.ID_Flight
INNER JOIN Airplanes AS p ON af.ID_Airplane = p.ID_Airplane
INNER JOIN Airlines AS l ON p.ID_Airline = l.ID_Airline
ORDER BY q.Avg_delay

```

Вывод:

Rating by flight delays:

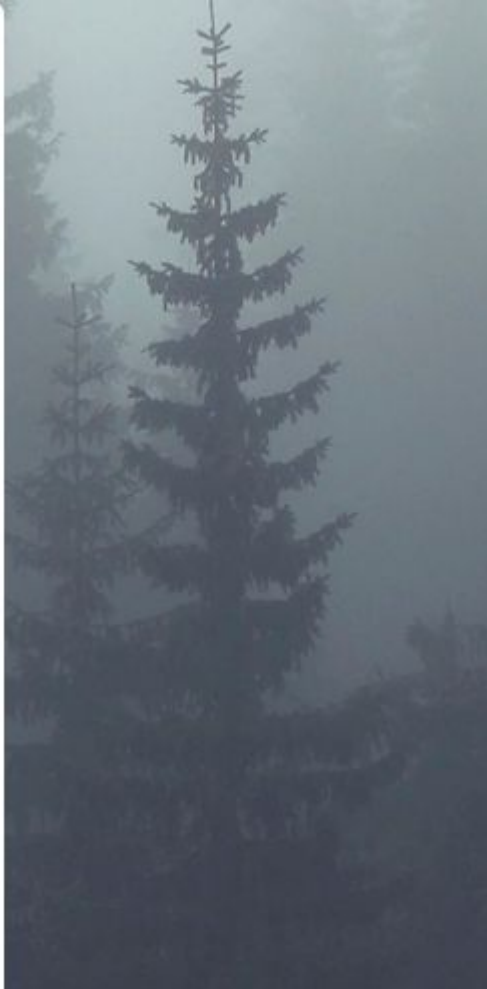
Airline code: AS  
Airline name: Alaska Airlines Inc.  
Average delay time (min): -1  
Average delay prediction (min): -1

Airline code: DL  
Airline name: Delta Air Lines Inc.  
Average delay time (min): 1  
Average delay prediction (min): -2

Airline code: WN  
Airline name: Southwest Airlines Co.  
Average delay time (min): 3  
Average delay prediction (min): 5

Airline code: HA  
Airline name: Hawaiian Airlines Inc.  
Average delay time (min): 4  
Average delay prediction (min): 0

Airline code: VX  
Airline name: Virgin America  
Average delay time (min): 4  
Average delay prediction (min): 5



## 7. Заключение.

В результате была создана база данных со всей информацией об аэропортах, авиакомпаниях и рейсах, которую можно находить удобным способом - с помощью Телеграм-бота. Он пригодится людям, которые часто совершают перелеты американскими авиалиниями. Его интерфейс интуитивно понятен, а внутри зашиты все необходимые и полезные запросы, обращающиеся к базе данных и выдающие нужную информацию. Телеграм-бот полностью соответствует заявленному функционалу.