Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет «Высшая школа экономики»
Факультет компьютерных наук

О Т Ч Е Т
по дисциплине
«Теория Баз Данных»

Выполнили студенты гр. 188

Квиндт Ева Сергеевна, Лежанкина Александра Игоревна
(ФИО)

_____

Проверил:

(должность, ФИО руководителя практики)

18.12.2020
(дата)

2020 год

С о д е р ж а н и е

1. Описание предметной области

В качестве предметной области мы выбрали куплю-продажу-доставку цветов в магазине. Особенность заключается в устройстве “конструктора”. Пользователь самостоятельно набирает букет, выбирая цветки из списка, готовых букетов в магазине нет.

Данные содержат подробную информацию о цветах, поставщиках, заказах и клиентах. Данные были собраны вручную на основе самых популярных реальных поставщиков цветов в Россию, а также самых продаваемых цветов. С помощью них можно обнаружить самые дорогие цветы или самого активного поставщика.

Упаковали мы этот функционал в чат-бота Вконтакте. Его интерфейс очень удобен. Он позволяет, например, при помощи кнопок собрать заказ, или же через представление “Карусель” продемонстрировать цветок в разном цвете с изображениями и возможностью добавить понравившийся в корзину созданного нами сообщества.

1.1. Цель

Цель нашего проекта -- это оптимизация процесса покупки букетов и отслеживания своего заказа.

1.2. Внешние данные

Мы не использовали готовых баз данных. Однако, вручную искали информацию о популярных поставщиках, цветах. Также в Интернете были найдены подходящие изображения цветов.

1.3. Основные сценарии использования

Среди сценариев можно выделить:

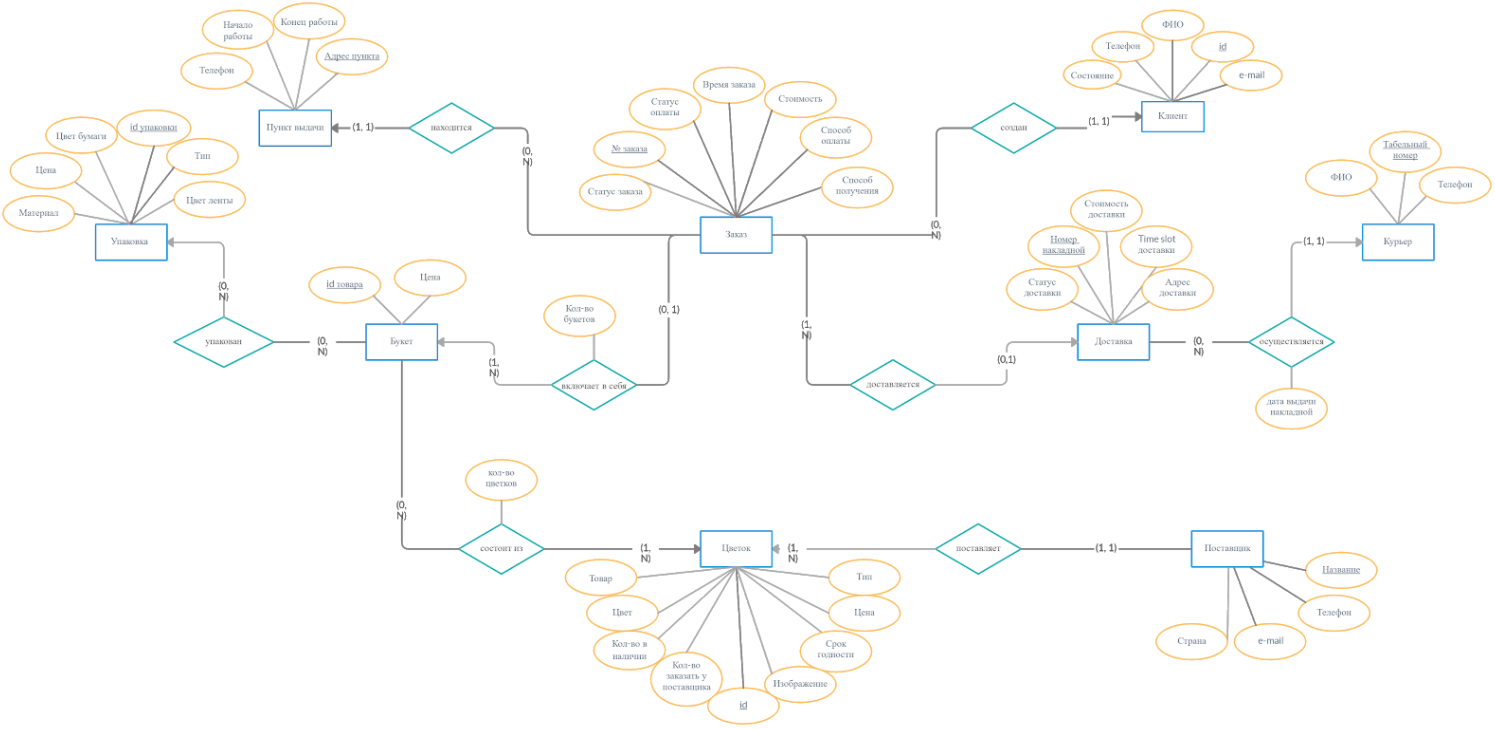
- Со стороны покупателя: Сборка букета, оформление заказа и доставки, просмотр истории своих заказов.
- Со стороны продавца: Просмотр статистики по самым продаваемым цветам, востребованным поставщикам, проверка корректности заказов, просмотр статистики по самому популярному времени доставки.

2. Концептуальная модель

2.1. Диаграмма «Сущность-связь»

ER-диаграмма в нотации Чена

(построена в приложении app.creately.com/diagram/)



2.2. Описание сущностей и связей

Имеем 9 сущностей: поставщик, цветок, букет, упаковка, заказ, пункт выдачи, доставка, курьер, клиент. У каждой от 2 до 8 атрибутов, уникальный идентификатор. Уникальность подразумевается самостоятельно для каждой сущности. Сущности взаимодействуют друг с другом при помощи 8 связей. Общий принцип взаимодействия заключается в подобных закономерностях: букет состоит из цветов, которые поставляет поставщик, и имеет упаковку; заказ доставляется курьером клиенту в пункт выдачи; заказом является букет.

Подробнее о сущностях и связях в разделе 3.2.

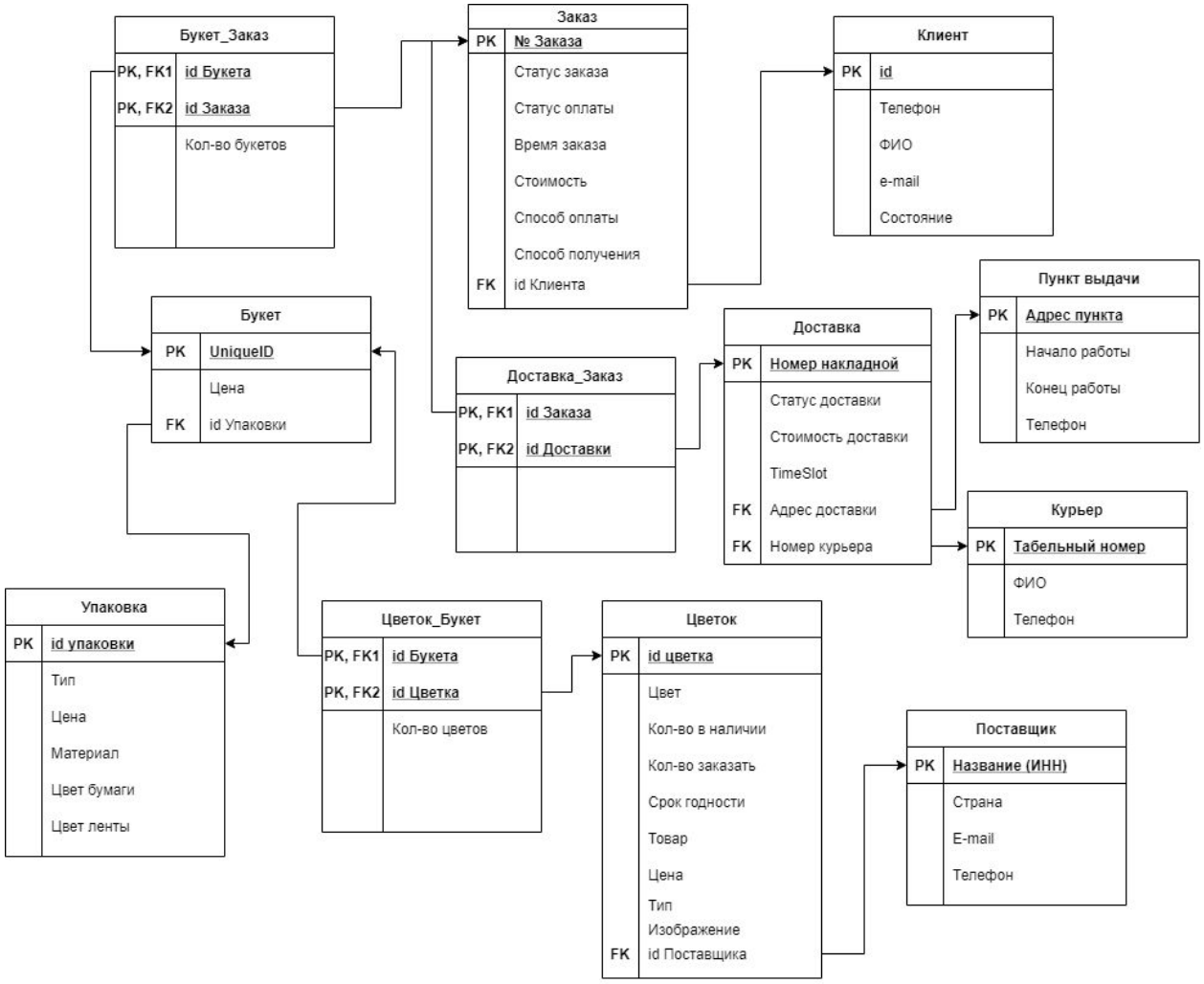
3. Инфологическая модель

При построении реляционной модели были установлены ограничения на связи, выбраны внешние ключи, а также задано ограничение множественности. Были добавлены таблицы пар ключей для реализации связей вида [многие-ко-многим].

3.1. Диаграмма «Таблица-связь»

Tr-диаграмма

(построена в приложении draw.io)



3.2. Словарь сущностей

В процессе создания модели было выделено 9 сущностей. В скобках атрибутов указан тип данных SQL, а также обозначение “id”, если атрибут является идентификатором, то есть PK:

- Сущность “Клиент” описывает самого клиента, включает в себя атрибуты: id (id; varchar(255)), телефон (varchar(20)), email (varchar(255)), ФИО (varchar(255)),

состояние (varchar(20); из возможных значений, определяющих взаимоотношение клиента с приложением, нужно, чтобы зафиксировать, на каком этапе остановился пользователь, чтобы приложение корректно обрабатывало его запросы спустя время, а также для работы с несколькими пользователями сразу);

- Сущность “Поставщик” содержит информацию о поставщике, включает в себя атрибуты: название (id; int), телефон (varchar(20)), e-mail (varchar(255)), страна (varchar(255));
- Сущность “Цветок” содержит информацию и цветке, включает атрибуты: тип (varchar(255)), цвет (varchar(255)), цена (int), срок годности (date), изображение (varchar(255)), id (id; int), количество в наличии (int), количество для заказа у поставщика (int), товар (varchar(255); является ссылкой на товар в магазине, нужен для эффективной работы приложения);
- Сущность “Букет” содержит информацию о букете, включает атрибуты: id товара (id; int), цена (int);
- Сущность “Упаковка” содержит информацию об упаковке, включает атрибуты: цвет бумаги (varchar(30)), цвет ленты (varchar(30)), тип (varchar(30); из возможных значений: “лента”, “бумага”, “бумага-лента”), материал (varchar(30)), цена (int), id упаковки (id; int);
- Сущность “Заказ” содержит информацию о заказе, включает атрибуты: статус заказа (varchar(20); из значений “готов”, “в процессе”), номер заказа (id; int), статус оплаты (varchar(20); из значений “оплачено” и “не оплачено”), время заказа (date), стоимость (int), способ оплаты (varchar(30)), способ получения (varchar(20); из значений “доставка” и “самовывоз”);
- Сущность “Доставка” содержит информацию о доставке заказа, включает атрибуты: статус доставки (varchar(20)), номер накладной (id; varchar(255)), стоимость доставки (int), time slot доставки (datetime), адрес доставки (varchar(255));
- Сущность “Пункт выдачи” содержит информацию о пункте выдаче заказа, включает атрибуты: начало работы (time), конец работы (time), адрес пункта (id; varchar(255)), телефон (varchar(20));
- Сущность “Курьер” содержит информацию о курьере, включает в себя атрибуты: ФИО (varchar(255)), телефон (varchar(20)), табельный номер (id; int).

Еще в базе данных представлены два вида связей: один-ко-многим, многие-ко-многим:

- Связь “состоит из” [многие-ко-многим] между сущностями “Букет” и “Цветок”, означает, из каких цветов состоит букет;
- Связь “поставляет” [один-ко-многим] между сущностями “Поставщик” и “Цветок”, означает, какие цветы поставляет поставщик;
- Связь “упакован” [многие-ко-многим] между сущностями “Букет” и “Упаковка”, означает, какая упаковка используется для букета;
- Связь “включает в себя” [один-ко-многим] между сущностями “Заказ” и “Букет”, означает, какие/ой букет, включает в себя заказ;
- Связь “находится” [один-ко-многим] между сущностями “Заказ” и “Пункт выдачи”, означает, какой пункт выдачи закреплен за заказом;
- Связь “создан” [один-ко-многим] между сущностями “Заказ” и “Клиент”, означает, кем создан заказ;
- Связь “доставляется” [один-ко-многим] между сущностями “Заказ” и “Доставка”, означает, какие параметры доставки закреплены за заказом;
- Связь “осуществляется” [один-ко-многим] между сущностями “Доставка” и “Курьер”, означает, кем осуществляется доставка.

4. Даталогическая модель

4.1. Используемая СУБД и диалект *SQL*

В качестве СУБД использован клиент Azure Data Studio, диалект *Transact-SQL* [1]. Такой выбор обусловлен тем, что при работе с различными таблицами на семинарах использовался Azure Data Studio, то есть опыт работы в этом клиенте уже был, что облегчило дальнейший процесс разработки клиентского приложения, а также он сам по себе достаточно удобен и понятен.

4.2. Хранимые процедуры и триггеры

Процедуры:

```
USE FLOWERSHOP;
    GO
    CREATE PROCEDURE ClientInfo AS
    BEGIN
         SELECT name AS Name, client_phone AS Phone, email AS Email
         FROM client
    END;
```

```
USE FLOWERSHOP;
    GO
    CREATE PROCEDURE PackingInfo AS
    BEGIN
        SELECT color_paper AS Color, material AS Material, type AS Type
        FROM packaging
    END;

USE FLOWERSHOP;
    GO
    CREATE PROCEDURE ProviderInfo AS
    BEGIN
        SELECT country AS Country, email AS Email, phone AS Phone
        FROM provider
    END;

EXEC ClientInfo // That command runs procedure

DROP PROCEDURE PackingInfo // That command drops procedure
```

Триггеры:

```
// Inserting data

USE FLOWERSHOP

GO

CREATE TRIGGER Packing_INSERT

ON packaging

AFTER INSERT

AS

INSERT INTO History (Price, Operation)

SELECT price, 'Добавлен товар ' + id + '  , который имеет ' + color + ' цвет'

FROM INSERTED


// Deleting data

USE FLOWERSHOP

GO

CREATE TRIGGER Packing_DELETE

ON packaging

AFTER DELETE

AS
```

```
INSERT INTO History (Price, Operation)
SELECT price, 'Удален товар ' + id + '  , который имел ' + color + ' цвет'
FROM DELETED


// Updating data
USE FLOWERSHOP
GO
CREATE TRIGGER Packing_UPDATE
ON packaging
AFTER UPDATE
AS
INSERT INTO History (Price, Operation)
SELECT price, 'Обновлен товар ' + id + '  , который имеет ' + color + ' цвет'
FROM INSERTED
```

4.3. Описание механизмов обеспечения целостности данных

Механизмы взаимодействия с БД реализованы так, что мы знаем, какую функцию хочет выполнить конкретный пользователь и предоставляем ему доступ только к необходимым таблицам.

Также в каждой из реализованных таблиц указан строгий тип полей. За счет этого обеспечение целостности данных достигается при помощи средств СУБД, которые не позволяют вставить данные неправильного типа.

Сущностная целостность достигается благодаря наличию первичного ключа PK у каждой сущности. СУБД не позволит добавить еще запись с уже существующим первичным ключом.

Ссылочная целостность достигается благодаря наличию внешних ключей FK, связывающих таблицы.

4.4. *DDL*-скрипты

```
CREATE DATABASE FLOWERSHOP;

CREATE TABLE client(
    email VARCHAR(255),
    id VARCHAR(255) NOT NULL,
    client_phone VARCHAR(20),
```

```
    name VARCHAR(255),
    state VARCHAR(20) NOT NULL,
    CONSTRAINT PK_Client PRIMARY KEY (id)
);

CREATE TABLE order(
    order_id INT NOT NULL,
    order_status VARCHAR(20) NOT NULL,
    payment_status VARCHAR(20) NOT NULL,
    date DATE NOT NULL,
    order_price INT NOT NULL,
    payment_method VARCHAR(30) NOT NULL,
    production_method VARCHAR(20) NOT NULL,
    client_id VARCHAR(255) NOT NULL,
    CONSTRAINT PK_Order PRIMARY KEY (order_id),
    CONSTRAINT FK_Client FOREIGN KEY (client_id) REFERENCES client(id)
);

CREATE TABLE point(
    address VARCHAR(255) NOT NULL,
    work_time_begin TIME NOT NULL,
    work_time_end TIME NOT NULL,
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT PK_Point PRIMARY KEY (address)
);

CREATE TABLE courier(
    courier_id INT NOT NULL,
    name VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT PK_Courier PRIMARY KEY (courier_id)
);

CREATE TABLE delivery(
    invoice_number VARCHAR(255) NOT NULL,
    status VARCHAR(20) NOT NULL,
    price INT NOT NULL,
    time_slot DATETIME NOT NULL,
    address VARCHAR(255) NOT NULL,
    order_id INT NOT NULL,
    courier_id INT NOT NULL,
```

```
    CONSTRAINT PK_Delivery PRIMARY KEY (invoice_number),
    CONSTRAINT FK_Adress FOREIGN KEY (address) REFERENCES point(address),
    CONSTRAINT FK_Courier FOREIGN KEY (courier_id) REFERENCES courier(courier_id),
    CONSTRAINT FK_Order FOREIGN KEY (order_id) REFERENCES order(order_id)
);

CREATE TABLE provider(
    tin INT NOT NULL,
    country VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT PK_Provider PRIMARY KEY (tin)
);

CREATE TABLE flower(
    id INT NOT NULL,
    in_stock INT NOT NULL,
    order_number INT NOT NULL,
    shelf_life DATE NOT NULL,
    price INT NOT NULL,
    type VARCHAR(255) NOT NULL,
    image VARCHAR(255) NOT NULL,
    provider_id INT NOT NULL,
    color VARCHAR(255),
    product VARCHAR(255),
    CONSTRAINT PK_Flower PRIMARY KEY (id),
    CONSTRAINT FK_Provider FOREIGN KEY (provider_id) REFERENCES provider(tin)
);

CREATE TABLE bouquet(
    id INT NOT NULL,
    price INT NOT NULL,
    packaging_id INT,
    CONSTRAINT PK_Bouquet PRIMARY KEY (id),
    CONSTRAINT FK_Order FOREIGN KEY (packaging_id) REFERENCES packaging(id)
);

CREATE TABLE packaging(
    id INT NOT NULL,
    color_paper VARCHAR(30),
    color_tape VARCHAR(30),
```

```
    price INT NOT NULL,
    material VARCHAR(30),
    type VARCHAR(30) NOT NULL,
    CONSTRAINT PK_Packaging PRIMARY KEY (id)
);

CREATE TABLE flower_bouquet(
    id_flower INT NOT NULL,
    id_bouquet INT NOT NULL,
    num_flowers INT NOT NULL,
    FOREIGN KEY (id_flower) REFERENCES flower(id),
    FOREIGN KEY (id_bouquet) REFERENCES bouquet(id),
    CONSTRAINT PK_flower_bouquet PRIMARY KEY (id_flower, id_bouquet)
);

CREATE TABLE order_bouquet(
    id_order INT NOT NULL,
    id_bouquet INT NOT NULL,
    num_bouquets INT NOT NULL,
    FOREIGN KEY (id_order) REFERENCES order(order_id),
    FOREIGN KEY (id_bouquet) REFERENCES bouquet(id),
    CONSTRAINT PK_order_bouquet PRIMARY KEY (id_order , id_bouquet)
);

CREATE TABLE order_delivery(
    id_order INT NOT NULL,
    id_delivery VARCHAR(255) NOT NULL,
    FOREIGN KEY (id_order) REFERENCES order(order_id),
    FOREIGN KEY (id_delivery ) REFERENCES delivery(invoice_number),
    CONSTRAINT PK_order_delivery PRIMARY KEY (id_order , id_delivery )
);
```

5. Клиентское приложения

Суть приложения заключается в чат-боте Вконтакте. Сначала мы создали сообщество “Flower”, являющееся аналогом странички магазина в Интернете. Загрузили туда товары, которые планируем продавать, то есть цветы. Добавили фотографии. Любой человек может зайти в это сообщество и написать туда, но отвечать ему сразу будет чат-бот. Он предложит набор команд, которые может выполнить.

С помощью бота можно удобно информировать потенциального клиента о товарах, оформлять заказы. Бот поможет пользователю отслеживать статус своего заказа и

историю. Такая технология очень удобна и активно применяется в реальных бизнес-процессах Вконтакте, особенно после того, как там появился определенный тип сообществ -- магазины.

Чат-бот позволяет обновлять (UPDATE) информацию в БД, Вставлять (INSERT) новую и Удалять (DELETE) старую. Это особенно активно используется при формировании заказа -- пользователь добавляет цветы в букет в приложении, при этом добавляя их в БД, а если пользователь отменяет заказ, то он также удаляет всю информацию о набранных за этот заказ цветах и букетах из БД. Добавлять также можно информацию о самом клиенте. UPDATE применяется, например, при изменении состояния пользователя, то есть при переходе к новому запросу.

INSERT:

```
INSERT INTO EA_client VALUES (NULL, NULL, \'{name}\', {user_id}, \'start\') //значения name и user_id берутся из полученных данных о пользователе
```

DELETE:

```
DELETE FROM EA_order WHERE order_id = {orderid} //orderid берется как параметр из функции
```

UPDATE:

```
UPDATE EA_client SET state = \'{state}\' WHERE id = {user_id} //значения state и user_id берутся из полученных данных о пользователе
```

5.1. Архитектура

В основе приложения язык Python 3 и библиотека vk_api для создания UI. База данных Azure Data Studio лежит на учебном сервере. Для идентификации своих таблиц в названия были добавлены префиксы EA. Связь между vk_api и Azure Data Studio осуществлялась через провайдер ODBC.

5.2. Сценарии использования

Приложение предназначено для покупателей и продавцов магазина цветов. Оно предоставляет интуитивно понятный интерфейс, позволяющий осуществить заказ, узнать статус и историю заказов, просмотреть прайс-лист цветов, а также увидеть изображения конкретного цветка.

Приложение можно усовершенствовать и доработать, внедрив процесс оформления доставки и упаковки букета, но на данном этапе задача стояла только в реализации 5 различных запросов к БД.

5.3. Организация доступа к данным

Провайдером для нашего проекта стал ODBC, а точнее ODBC Driver 17 for SQL Server -- Windows. Мы решили установить его, во-первых, по рекомендации преподавателей, а во-вторых, так как было найдено достаточное количество понятной документации, с помощью которой не составило труда разобраться, как с этим работать.

Язык программирования использовался Python 3. Основными библиотеками стали:

- vk_api, в частности vk_api.bot_longpoll, vk_api.keyboard, vk_api.utils -- для непосредственной связи кода с интерфейсом Вконтакте, а именно чат-бот
- pyodbc -- для подключения к учебному серверу через

```
connection_string = "Driver={ODBC Driver 17 for SQL Server};"\
"Server=tcp:2020hsedbmstest.database.windows.net;"\
"Database=AdventureWorksLT;"\
"uid=stud20;pwd=!Student2020"
```

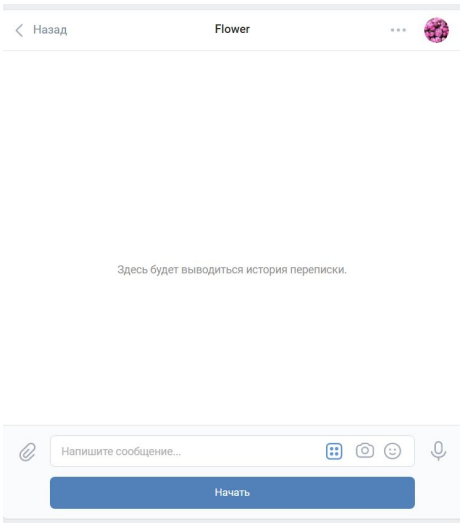
- transliterate -- для транскрипта имен пользователей Вконтакте при занесении их ФИО в БД, так как Azure Data Studio не умеет работать с кириллицей
- json -- для представления “Карусели” в одном из запросов

Авторизация пользователя осуществлялась автоматически средствами vk_api. При получении сообщения от человека мы также получали его id -- номер страницы, который и становился его идентификатором в базе клиентов.

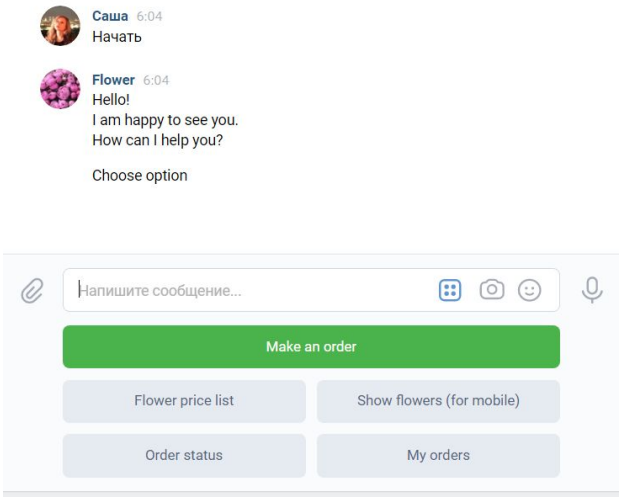
Доступ к данным осуществляется за счет хранения состояния каждого пользователя из базы. Под состоянием подразумевается его нахождение “внутри” приложения. То есть, чат-бот ждет от него определенную команду, а до тех пор, пока она не задана, соответствующее состояние приписывается пользователю. Тогда в любое время, когда он вернется, то сможет начать с того же места. Такая философия реализации чаще всего используется при проектирование реальных чат-ботов для действующих бизнес-процессов.

5.4. Интерфейс с пользователем

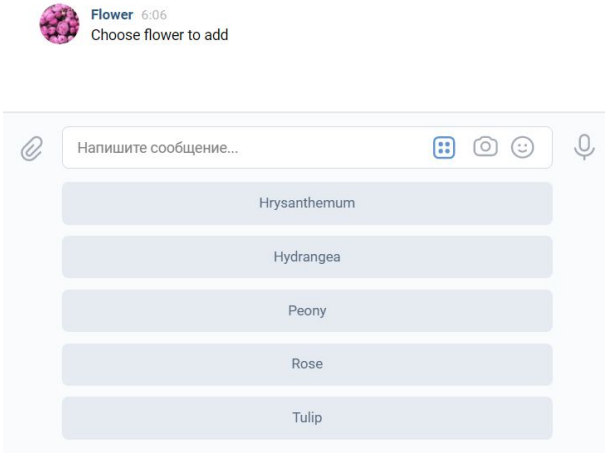
Сначала пользователь видит такой приветственный интерфейс:



При нажатии кнопки “Начать” открывается возможность выбрать дальнейшие функции в виде кнопок, где каждая кнопка какой-то запрос:



В основном взаимодействие происходит благодаря такой клавиатуре:



Но в мобильной версии ещё доступна Карусель, где при нажатии на картинку она увеличивается, а при нажатии на кнопку “Buy”, пользователь переходит в раздел соответствующего товара в Сообществе магазина для покупки:

Стоит отметить, что язык приложения английский не случайно. Во-первых, это сделано для интернационального использования, а во-вторых Azure Data Studio, где размещались БД, не может идентифицировать кириллицу, поэтому заполнялась БД цветами на английском языке, и чтобы не портить общую картину, взаимодействие с пользователем тоже было переделано на английский.

5.5. Отчёты

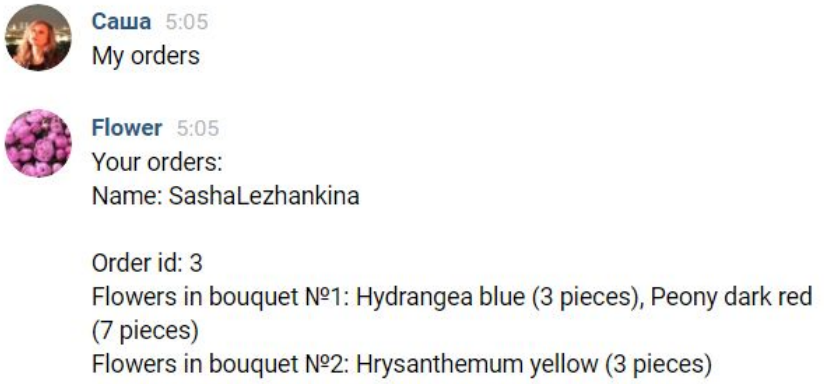
1) Демонстрация всех заказов клиента (id клиента определяется автоматически средствами vk api):

```
SELECT c.name, o.order_id, fl.type, fl.color, fb.id_bouquet, fb.num_flowers FROM
EA_client c
JOIN EA_order o ON c.id = o.client_id
JOIN EA_order_bouquet ob ON o.order_id = ob.id_order
JOIN EA_flower_bouquet fb ON ob.id_bouquet = fb.id_bouquet
JOIN EA_flower fl ON fb.id_flower = fl.id
WHERE c.id = 74269505 //для примера id Александры Лежанкиной
ORDER BY o.order_id
```

Результат:

	name	order_id	type	color	id_bouquet	num_flowers
1	SashaLezhankina	3	Hydrangea	blue	4	3
2	SashaLezhankina	3	Peony	dark red	4	7
3	SashaLezhankina	3	Hrysanthemum	yellow	5	3

Результат в интерфейсе чат-бота:



2) Продемонстрировать прайс-лист цветов в магазине:

```
SELECT type, color, price FROM EA_flower
```

Результат:

	type	color	price
1	Hrysanthemum	pink	43
2	Hrysanthemum	white	55
3	Hrysanthemum	yellow	43
4	Hydrangea	blue	23
5	Hydrangea	white	25
6	Hydrangea	dark blue	75
7	Hydrangea	pink	80
8	Peony	pink	53
9	Peony	white	27
1…	Peony	dark red	43
1…	Peony	light pi…	40
1…	Rose	red	13
1…	Rose	pink	10
1…	Rose	lilac	16
1…	Rose	white	14
1…	Tulip	red	21
1…	Tulip	white	20
1…	Tulip	pink	23
1…	Tulip	yellow	20

Результат в интерфейсе чат-бота:

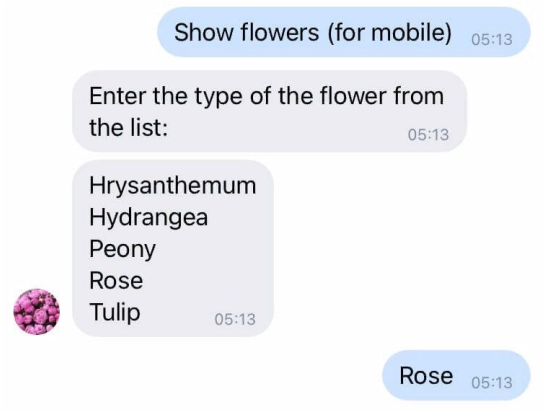
3) Продемонстрировать изображение и стоимость одного цветка в разном цвете, а также ссылку на покупку:

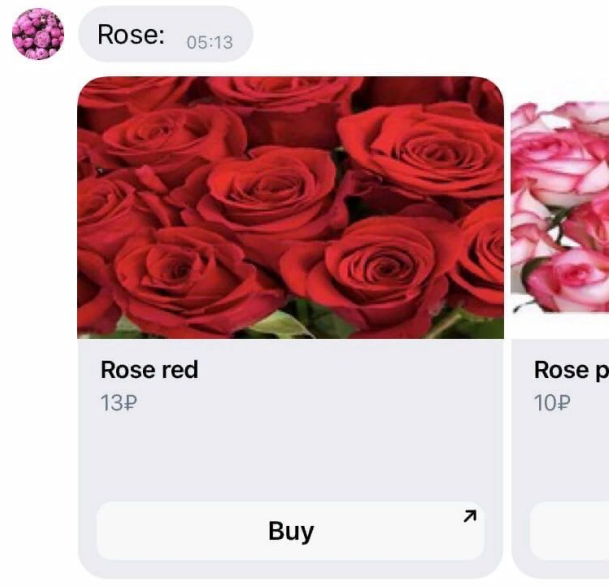
```
SELECT color, price, image, product FROM EA_flower
WHERE type = 'Rose' // такой тип в качестве примера
```

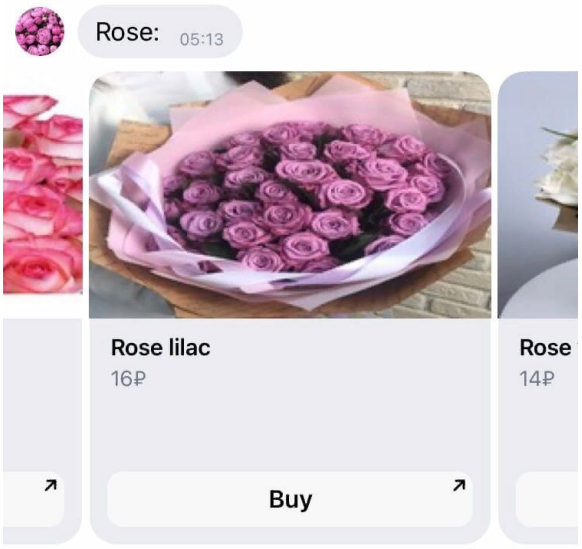
Результат (по понятным причинам в Azure нельзя красиво вывести изображение, поэтому остаются только ссылки):

	color	price	image	product
1	red	13	-201084201_457239080	https://vk.com/market-201084…
2	pink	10	-201084201_457239081	https://vk.com/market-201084…
3	lilac	16	-201084201_457239078	https://vk.com/market-201084…
4	white	14	-201084201_457239079	https://vk.com/market-201084…

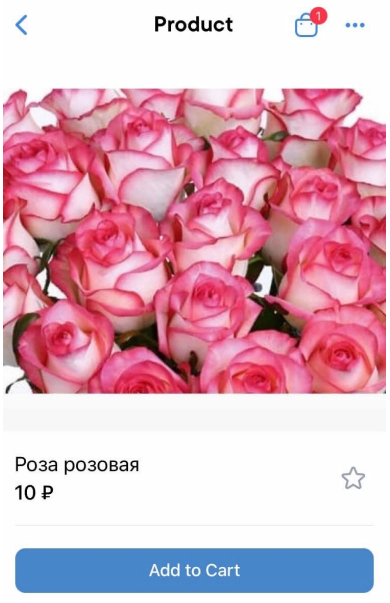
Результат в интерфейсе чат-бота (мобильная версия):







Интерфейс при нажатии “Buy” возле розовых роз:



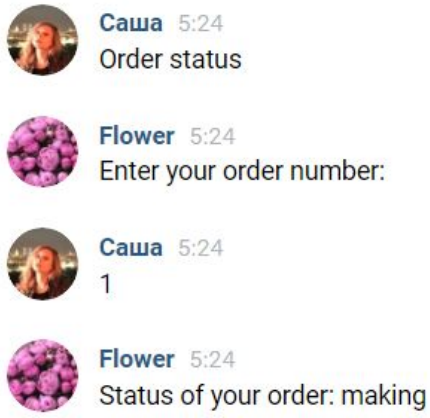
4) Проверить статус конкретного заказа по его номеру:

```
SELECT order_status FROM EA_order
JOIN EA_client ON EA_client.id = EA_order.client_id
WHERE client_id = 74269505 AND order_id = 1 // id взят в качестве примера, номер заказа тоже
```

Результат:

	order_status
1	making

Результат в интерфейсе чат-бота:



5) Сформировать заказ, позволяющий неоднократно выбирать цветы и букеты, а также отменять выбор.

Формирование заказа состоит из нескольких состояний клиента: main_menu (‘Make order’), choose_ftype, choose_fcolor, choose_fnum, add_confirm_cancel.

Пользователь решает сделать заказ и нажимает на соответствующую кнопку в боте. Автоматически в БД добавляется новый заказ пользователю со статусом making. Статус пользователя обновляется на ‘choose_ftype’.

```
# Add new order to DataBase for client
order_id = 0
cursor.execute('SELECT max(order_id) FROM EA_order')
for i in cursor:
   order_id = i[0] + 1
bouquet_id = 0
cursor.execute('SELECT max(id) FROM EA_bouquet')
for i in cursor:
   bouquet_id = i[0] + 1
cursor.execute(f'INSERT INTO EA_order VALUES ({order_id}, '
```

```
                    f'\'making\', \'not paid\', \'2020-12-18\', 0, \'unknown\', \'pick
up\', {user_id})')
        cursor.execute(f'INSERT INTO EA_bouquet VALUES ({bouquet_id}, 0, NULL)')
        cursor.execute(f'INSERT INTO EA_order_bouquet VALUES ({order_id}, {bouquet_id},
1)')
```

Затем пользователь выбирает тип цветка, который хочет добавить в букет. Обработка неправильного ввода поддерживается. Статус пользователя при правильном вводе меняется на ‘choose_fcolor’.

```
        cursor.execute(f'INSERT INTO EA_flower_bouquet '
                  f'VALUES ((SELECT max(id) FROM EA_flower WHERE type = \'{ftype}\'), '
                      f'        (SELECT max(id_bouquet) FROM EA_order_bouquet '
                      f'        WHERE id_order = (SELECT max(order_id) FROM EA_order '
                      f'                          WHERE client_id = {user_id})),'
                      f'        0)')
```

Далее функции работают похожим образом, БД обновляется на каждом этапе вплоть до подтверждения, отмены или дополнения заказа.

Вот важные пункты:

- Обработка запросов с ошибочным вводом поддерживается.
- Вставка в БД происходит на каждом состоянии пользователя, чтобы не хранить данные в Python и не возникало коллизий при использовании приложения несколькими клиентами.
- Обработка варианта, когда цветов нет в наличие поддерживается.
- При формировании заказа он сразу (заказ) добавляется в БД.

И наконец пользователь может подтвердить свой заказ (заказу присваивается статус processing), отменить заказ (заказ вычищается из БД, пользователь переходит в главное меню), дополнить последний букет (продолжается добавление цветов), дополнить заказ еще одним букетом (добавляется новый букет в последний заказ и продолжается добавление цветов).

Flower 3:32
Hello!
I am happy to see you.
How can I help you?

Choose option

Ева 3:33
Make an order

Flower 3:33
Choose flower to add

Ева 3:33
Rose

Flower 3:33
Choose flower color:

Ева 3:33
pink

Flower 3:33
Sorry. There are not any pink Rose in stock. Choose another color:

сегодня

Ева 3:33
lilac

Flower 3:33
Enter count of lilac Rose (number from 1 to 92)

Ева 3:33
7

Flower 3:33
Your order is

Order №3
Rose lilac x7 =112₽
Total: 112₽

What do you want to do?

Ева 3:33
Cancel order

Flower 3:33
Okay, your order is canceled.

Choose option

6. Заключение

6.1. Объёмные характеристики разработки

Работа над проектом осуществлялась в течение всего семестра, начиная от постановки задачи и определения бизнес-логики, заканчивая разработкой клиентского приложения на базе чат-бота Вконтакте. Безусловно, приближаясь к дедлайну процесс занимал все больше часов в сутках, так как разработка оказалась самой сложной частью. Смело можно заявить, что средние 3-5 часов в неделю на выполнение домашнего задания по дисциплине перетекли в 8-10 часов в день. По итогу для приложения написано около 500 строк кода.

6.2. Авторский вклад и комментарии по выполнению проекта

Над проектом трудились все более-менее в равной степени, в зависимости от занятости. Стоит отметить, что на полпути наша команда EDA потеряла одного участника Даниила Стрипского, превратившего нас в EA. Вследствие этого самая объёмная часть работы, разработка приложения, легла на плечи всего двух участниц. Хотя выбывший участник на протяжении своего пребывания с нами активного участия не принимал.

Постановка задачи, планирование бизнес-логики, рисование диаграмм ER и TR, написание кода DDL выполнялись в равной степени Евой Квиндт и Александрой Лежанкиной.

В разработке приложения бОльшую часть реализовала Ева Квиндт, а в написании отчета Александра Лежанкина. В тестировании принимали участие обе.

7. Источники

1. Microsoft. SQL Server technical documentation [Электронный ресурс] URL: https://docs.microsoft.com/en-us/sql/sql-server/ (дата обращения: 10.10.2020).
2. Репозиторий MS Teams 2020-DBMS-Teams со всей необходимой информацией [Электронный ресурс] URL: https://teams.microsoft.com/_#/school/conversations/Общий?threadId=19:bb25be41aac44ba29675f23b524862f4@thread.tacv2&ctx=channel (дата обращения: с 01.09.2020 - по 17.12.2020)