

Содержание

Содержание.....	1
1. Описание предметной области	2
1.1. Цель.....	2
1.2. Внешние данные	2
1.3. Основные сценарии использования	2
2. Концептуальная модель	3
2.1. Диаграмма «Сущность-связь»	3
2.2. Описание сущностей и связей	3
3. Инфологическая модель	5
3.1. Диаграмма «Таблица-связь»	5
3.2. Словарь данных.....	6
4. Дatalogическая модель	11
4.1. Используемая СУБД и диалект <i>SQL</i>	11
4.2. Хранимые процедуры и триггеры	11
4.3. Описание механизмов обеспечения целостности данных	14
4.4. <i>DDL</i> -скрипты.....	15
5. Клиентское приложения	18
5.1. Архитектура	18
5.2. Сценарии использования.....	18
5.3. Организация доступа к данным.....	18
5.4. Интерфейс с пользователем	18
5.5. Отчёты.....	19
6. Заключение	26
6.1. Объёмные характеристики разработки.....	26
6.2. Авторский вклад и комментарии по выполнению проекта	26
6.3. Репозиторий.....	26
7. Источники	27

1. Описание предметной области

В наши дни огромное количество людей использует автомобили. На данный момент топовые производители топлива – производители нефти и газа. В своем проекте мы хотим создать продукт, с помощью которого можно будет осуществлять контроль над поставками некоторых видов топлива.

1.1. Цель

Цель: создание приложения (Telegram-бот) для просмотра и учета информации о сетях АЗС, определенных точках и поставках. На основе данных выдавать определенные статистики.

В перспективе приложение должно уметь отслеживать цепочку поставок от места добычи газа/нефти до распределения топлива от перерабатывающих компаний по определенным сетям АЗС и отдельным точкам. Безусловно, для этого потребуются внести изменения в имеющуюся структуру проекта, начиная с небольших изменений в ER-диаграмме и заканчивая самим клиентом.

1.2. Внешние данные

В роли внешних данных выступали отчетности нефтегазовых компаний и следующий сайт: <https://energybase.ru>. Также некоторые данные (такие как данные по таблице заказов) приходилось генерировать самостоятельно. Это связано с неимением доступа к подобного рода информации.

1.3. Основные сценарии использования

Важной частью является сбор и хранение данных о поставках/продажах. Это необходимо для возможности ведения учета об операциях компаний. При этом в нашем продукте сбор данных обязательно должен проходить через создателей данного проекта. Вывод информации реализуется с помощью чат-бота. Основной сценарий работы приложения происходит от лица внешнего пользователя. Он просматривает собранную в базе данных (БД) информацию через реализованные функции. Первым делом он запускает чат-бот, а далее совершает действия через интуитивно понятный интерфейс взаимодействия (кнопки).

2. Концептуальная модель

2.1. Диаграмма «Сущность-связь»

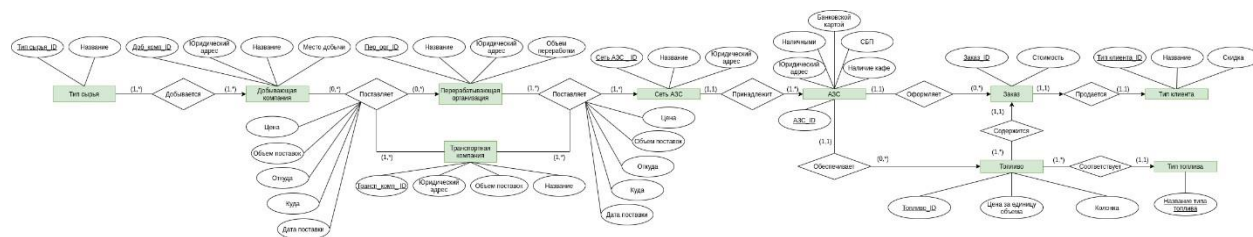


Рисунок 2.1. ER-диаграмма в нотации Чена

Диаграмму в лучшем качестве можно посмотреть по ссылке:

[https://github.com/ArtIrina/TDB_project/blob/main/ERD-ERD.drawio%20\(3\).png](https://github.com/ArtIrina/TDB_project/blob/main/ERD-ERD.drawio%20(3).png)

2.2. Описание сущностей и связей

ER-диаграмма создавалась на основе выделенных нами сущностей.

Пусть существует сеть АЗС. Для нормального функционирования она должна наладить поставки продаваемого на ней товара (топлива). Исходя из этого, понимаем, что существует связь с поставщиками. Получив в свои хранилища товар, они должны найти, куда его сбыть. Так появляется некоторый клиент в данной области.

Разобравшись чуть детальнее, мы выделили 10 сущностей. У девяти из них имеется уникальный идентификатор – name_ID. Уникальность десятой (тип топлива) формируется через единственный атрибут “Название типа топлива”.

Перечислим сущности и их атрибуты:

- Тип сырья – то, какие типы сырья могут встречаться в нашей БД. Атрибуты: Типы_сырья_ID, Название.
- Добывающая компания – сущность, описывающая добывающие топливо организации. Атрибуты: Доб_комп_ID, Юридический адрес, Название, Место добычи.
- Перерабатывающая компания – сущность, описывающая перерабатывающие организации. Атрибуты: Пер_орг_ID, Название, Юридический адрес, Объем переработки (млн. тонн в год).
- Транспортная компания – сущность, описывающая организацию, специализирующуюся на транспортировке топлива или сырья от одной организации

другой.

Атрибуты: Трансп_комп_ID, Юридический адрес, Объем поставок, Название.

- Сеть АЗС – компания, владеющая сразу несколькими АЗС. Например, Лукойл.
Атрибуты: Сеть_АЗС_ID, Название, Юридический адрес.
- Точка АЗС (далее – АЗС) – сущность, описывающая каждую отдельную автозаправочную станцию.
Атрибуты: АЗС_ID, Юридический адрес, Возможно ли оплата наличными, Банковской картой, СБП, Наличие кафе.
- Заказ – сущность, описывающая заказ, проведенный на АЗС.
Атрибуты: Заказ_ID, Стоимость.
- Топливо – сущность, описывающая проданное топливо, регламентированное в заказе на АЗС.
Атрибуты: Топливо_ID, Цена за единицу объема, Колонка.
- Тип клиента – сущность, описывающая, какой тип клиента совершал покупку на АЗС. Атрибуты: Тип_клиента_ID, Название, Скидка.
- Тип топлива – сущность-справочник. Содержит названия каждого доступного вида топлива.
Атрибут: Название типа топлива.

Наши сущности объединены друг с другом связями “один к одному”, “один ко многим”, “многие ко многим”. Имеются тернарные связи, связанные с транспортной компанией. Ниже покажем, как связаны сущности между собой:

- “Добывается” – сырье добывается многими добывающими компаниями, и компания может добывать несколько типов сырья. Связь “многие ко многим”.
- “Поставляет” – тернарная связь. Представлена в двух типах. Первый соединяет добывающую компанию и перерабатывающую компании, между которыми исполняет свои услуги транспортная компания. Второй тип описывается связь между перерабатывающей организацией и сетью АЗС. Как раз ее можно будет расширить для улучшения проекта. Тернарная связь имеет свои атрибуты: цену перевозки, объем поставки, начало пути конкретной перевозки, конец пути конкретной перевозки, а также дату, в которую осуществляется поставка.

Рисунок 3.2. TR-диаграмма в нотации Чена

3.2. Словарь данных

По составленному словарю данных очень удобно составлять TR-диаграмму. Мы определяем какие атрибуты сущностей будут иметь статус внешнего ключа, то же самое для первичного ключа, а также на этом этапе мы определяем типы данных.

Сущность	Ключ	Атрибут	Тип	Связь
Типы сырья FeedstockType	PK	Типы сырья_ID FeedstockType_ID	int	
		Название title	varchar(200)	
Типы сырья_Доб комп FeedstockType_MiningOrg	PK, FK1	Доб_комп_ID mining_ID	int	Добывающие компании (Доб_комп_ID) MiningOrg (mining_ID)

	PK, FK2	Тип сырья_ID FeedstockType_ID	int	Типы сырья (Тип сырья_ID) FeedstockType (FeedstockType_ID)
Добывающие компании MiningOrg	PK	Доб_комп_ID mining_ID	int	
		Название title	varchar(200)	
		Юридический адрес address	varchar(200)	
		Место добычи location	varchar(200)	
Доб_Тран_Пер Transit_Refiner	PK, FK1	Доб_комп_ID mining_ID	int	Добывающие компании (Доб_комп_ID) MiningOrg (mining_ID)

	PK, FK2	Трансп_комп_ID <i>trans_ID</i>	int	Транспортные компании (Трансп_ комп_ID) <i>TransporterOrg</i> (<i>trans_ID</i>)
	PK, FK3	Пер_орг_ID <i>refiner_ID</i>	int	Перерабатывающие организации (Пер_орг_ID) <i>Refiner</i> (<i>refiner_ID</i>)
		Объем поставки <i>shipment_vol</i>	float	
		Цена поставки <i>shipment_cost</i>	money	
		Откуда <i>_from_</i>	varchar(200)	
		Куда <i>_to_</i>	varchar(200)	
		Дата поставки <i>shipment_date</i>	date	
Перерабатывающие организации <i>Refiner</i>	PK	Пер_орг_ID <i>refiner_ID</i>	int	
		Название <i>title</i>	varchar(200)	
		Юридический адрес <i>address</i>	varchar(200)	
		Объем переработки за период времени <i>processing_volume</i>	float	
Транспортные компании <i>TransporterOrg</i>	PK	Трансп_комп_ID <i>trans_ID</i>	int	
		Название	varchar(200)	

		title		
		Юридический адрес address	varchar(200)	
		Объем поставок supply_vol	float	
Пер_Тран_АЗС Transit_Net	PK, FK1	Пер_орг_ID refiner_ID	int	Перерабатывающие организации (Пер_орг_ID) Refiner (refiner_ID)

	PK, FK2	Тран_ID trans_ID	int	Транспортные компании (Трансп_комп_ID) TransporterOrg (trans_ID)
	PK, FK3	Сеть_АЗС_ID net_ID	int	Сети АЗС (Сеть АЗС_ID) GAS_Station_Net (net_ID)
		Объем поставки shipment_vol	float	
		Цена поставки shipment_cost	money	
		Откуда _from_	varchar(200)	
		Куда _to_	varchar(200)	
		Дата поставки shipment_date	date	
Сети АЗС GAS_Station_Net	PK	Сеть АЗС_ID net_ID	int	

		Название brand	varchar(200)	
		Юридический адрес address	varchar(200)	
A3C GAS_Station	PK	A3C_ID station_ID	int	
	FK	Сеть A3C_ID net_ID	int	Сети A3C (Сеть A3C_ID) GAS_Station_Net (net_ID)

		Наличие кафе has_cafe	bool	
		Юридический адрес address	varchar(200)	
		Наличными cash	bool	
		Банковской картой by_card	bool	
		СБП fps	bool	
Заказы Orders	PK	Заказ_ID order_ID	int	
	FK1	A3C_ID station_ID	int	A3C (A3C_ID) GAS_Station (station_ID)

	FK2	Тип клиента_ID customer_type	int	Типы клиентов (Тип клиента_ID) Customer (customer_type)
	FK3	Топливо_ID fuel_ID	int	Топливо (Топливо_ID) Fuel (fuel_ID)
		Стоимость total_sum	money	
Топливо Fuel	PK	Топливо_ID fuel_ID	int	
	FK1	A3C_ID station_ID	int	A3C (A3C_ID) GAS_Station (station_ID)
	FK2	Название типа топлива type	varchar(200)	Типы топлива (Название типа топлива) FuelType (type)
		Цена за единицу объема price	money	
		Колонка pump_num	int	
Типы топлива FuelType	PK	Название типа топлива type	varchar(200)	
Типы клиентов Customer	PK	Тип клиента_ID customer_type	int	
		Название title	varchar(200)	

		Скидка	float	
		sale		

Зеленым цветом выделены соответствующие названия на английском. Так они будут называться в DDL-скрипте, представленном далее.

4. Даталогическая модель

4.1. Используемая СУБД и диалект *SQL*

Использовались СУБД SQLite Studio 3.3.3 и диалект SQLite.

4.2. Хранимые процедуры и триггеры

Процедуры:

- output_stations - выводить список точек АЗС (адрес, к какой сети АЗС принадлежит точка, есть ли кафе, возможен ли безналичный расчет, есть ли система быстрых платежей);
- output_minings - выводить название и адрес добывающей компании;
- output_refiners - выводить название и адрес перерабатывающей компании;
- output_feedstocks_oil - выводить месторождение нефти, локацию, добывающую компанию;
- output_feedstocks_gas - выводить месторождение нефти, локацию, добывающую компанию;
- output_stations - выводить список сетей АЗС (название сети, адрес, число точек);
- output_nets - выводить список сетей АЗС (название сети, адрес, число точек);
- output_transfer_nets - выводить список поставок от переработчика к сети АЗС;
- output_transfer_refiner - выводить список поставок от доб. орг. к сети переработчику;
- output_min_prices - выводить минимальную цену по каждому типу топлива для всех точек АЗС;

- `output_max_prices` - выводить максимальную цену по каждому типу топлива для всех точек АЗС;
- `output_avg_prices` - выводить среднюю цену по каждому типу топлива для всех точек АЗС.

Соответствующий код:

```

def output_stations(self):
    """Выводить список точек АЗС (адрес, к какой сети АЗС принадлежит точка,
    есть ли кафе, возможен ли безналичный расчет, есть ли система быстрых платежей)"""
    result = self.cursor.execute(
        "SELECT GS.address, GSN.brand, GS.has_cafe, GS.by_card, GS.fps FROM GAS_Station AS GS \
        JOIN GAS_Station_Net AS GSN ON (GS.net_id = GSN.net_id)"
    )
    return result.fetchall()

def output_minings(self):
    """Выводить название и адрес добывающей компании"""
    result = self.cursor.execute("SELECT title, address FROM MiningOrg")
    return result.fetchall()

def output_refiners(self):
    """Выводить название и адрес перерабатывающей компании"""
    result = self.cursor.execute("SELECT title, address FROM Refiner")
    return result.fetchall()

def output_feedstocks_oil(self):
    """Выводить месторождение нефти, локацию, добывающую компанию"""
    result = self.cursor.execute("SELECT MO.location, MO.title FROM MiningOrg AS MO \
        JOIN FeedstockType_MiningOrg AS FSMO ON (MO.mining_ID = FSMO.mining_ID) \
        JOIN FeedstockType AS FS ON (FS.FeedstockType_ID = FSMO.FeedstockType_ID) \
        WHERE FS.title = \"Нефть\"")
    return result.fetchall()

def output_feedstocks_gas(self):
    """Выводить месторождение нефти, локацию, добывающую компанию"""
    result = self.cursor.execute("SELECT MO.location, MO.title FROM MiningOrg AS MO \
        JOIN FeedstockType_MiningOrg AS FSMO ON (MO.mining_ID = FSMO.mining_ID) \
        JOIN FeedstockType AS FS ON (FS.FeedstockType_ID = FSMO.FeedstockType_ID) \
        WHERE FS.title = \"Газ\"")
    return result.fetchall()

def output_stations(self):
    """Выводить список сетей АЗС (название сети, адрес, число точек)"""
    result = self.cursor.execute(
        "SELECT GS.address, GSN.brand, GS.has_cafe, GS.by_card, GS.fps FROM GAS_Station AS GS \
        JOIN GAS_Station_Net AS GSN ON (GS.net_ID = GSN.net_ID)"
    )
    return result.fetchall()

def output_nets(self):
    """Выводить список сетей АЗС (название сети, адрес, число точек)"""
    result = self.cursor.execute(
        "SELECT GSN.brand, GSN.address, count(GS.station_ID) FROM GAS_Station_Net AS GSN \
        JOIN GAS_Station AS GS ON (GS.net_ID = GSN.net_ID) \
        GROUP BY (GSN.net_ID)"
    )
    return result.fetchall()

```

```

def output_transfer_nets(self):
    """Выводить список поставок от переработчика к сети АЗС"""
    result = self.cursor.execute(
        "SELECT R.title, TO_.title, GSN.brand, TN.shipment_vol, TN.shipment_cost FROM Transit_Net AS TN \
        JOIN Refiner AS R ON (TN.refiner_ID = R.refiner_ID) \
        JOIN TransporterOrg AS TO_ ON (TN.trans_ID = TO_.trans_ID) \
        JOIN GAS_Station_Net AS GSN ON (TN.net_ID = GSN.net_ID)"
    )
    return result.fetchall()

def output_transfer_refiner(self):
    """Выводить список поставок от доб. орг. к сети переработчику"""
    result = self.cursor.execute(
        "SELECT MO.title, TO_.title, R.title, TR.shipment_vol, TR.shipment_cost FROM Transit_Refiner AS TR \
        JOIN MiningOrg AS MO ON (TR.mining_ID = MO.mining_ID) \
        JOIN Refiner AS R ON (TR.refiner_ID = R.refiner_ID) \
        JOIN TransporterOrg AS TO_ ON (TR.trans_ID = TO_.trans_ID)"
    )
    return result.fetchall()

def output_min_price(self):
    result = self.cursor.execute("SELECT DISTINCT F.type, min(F.price) AS price FROM Fuel AS F \
    JOIN GAS_Station AS GS ON (GS.station_ID = F.station_ID) \
    GROUP BY F.type ORDER BY min(F.price) DESC")

    return result.fetchall()

def output_max_price(self):
    result = self.cursor.execute("SELECT DISTINCT F.type, max(F.price) AS price FROM Fuel AS F \
    JOIN GAS_Station AS GS ON (GS.station_ID = F.station_ID) \
    GROUP BY F.type ORDER BY max(F.price) DESC")

    return result.fetchall()

def output_avg_price(self):
    result = self.cursor.execute("SELECT DISTINCT F.type, avg(F.price) AS price FROM Fuel AS F \
    JOIN GAS_Station AS GS ON (GS.station_ID = F.station_ID) \
    GROUP BY F.type ORDER BY avg(F.price) DESC")

    return result.fetchall()

```

4.3. Описание механизмов обеспечения целостности данных

Для минимизации аномалий при работе с реляционной моделью применяли приемы нормализации.

Создали уникальный идентификатор для каждой таблицы, прописали условия на связи: как действовать, если удаляем запись в таблице, как действовать, если обновляем данные таблицы. Для атрибутов, которые мы считаем важными, поставили “NOT NULL”. То есть хотим, чтобы некоторые данные вводились в обязательном порядке. В полях с типом varchar указываем достаточный, на наш взгляд, размер 200.

Мы предоставляем доступ к БД без регистрации. Соответственно, ни к чему добавлять всевозможные проверки на корректную работу пользователя. Если пользователь не использует интерфейс взаимодействия – кнопки в Telergram-боте, значит он не работает, не обращается к нашему приложению. Соответственно, бот не реагирует.

Проход по меню приложения мы ограничиваем, и выдаем сообщение, если пользователь решает пойти дальше разработанного меню. Если на первом меню нажать “назад”, то он скажет, что мы уже на первой странице, и раньше ничего нет. Аналогично для последнего. Только предупреждение уже о том, что дальше ничего нет.

4.4. DDL-скрипты

```
1 create table MiningOrg (  
2   mining_ID int NOT NULL primary key,  
3   title varchar(200) NOT NULL,  
4   address varchar(200),  
5   location varchar(200)  
6 );  
7  
8 create table FeedstockType (  
9   FeedstockType_ID int NOT NULL primary key,  
10  title varchar(200) NOT NULL  
11 );  
12  
13 create table FeedstockType_MiningOrg (  
14  mining_ID int references MiningOrg ON DELETE SET NULL ON UPDATE CASCADE,  
15  FeedstockType_ID int references FeedstockType ON DELETE SET NULL ON UPDATE CASCADE,  
16  primary key(mining_ID, FeedstockType_ID)  
17 );  
18  
19 create table TransporterOrg (  
20  trans_ID int NOT NULL primary key,  
21  title varchar(200) NOT NULL,  
22  address varchar(200),  
23  supply_vol float  
24 );  
25  
26 create table Refiner (  
27  refiner_ID int NOT NULL primary key,  
28  title varchar(200) NOT NULL,  
29  address varchar(200),  
30  processing_volume float  
31 );  
32  
33 create table Transit_Refiner (  
34  mining_ID int references MiningOrg ON DELETE SET NULL ON UPDATE CASCADE,  
35  trans_ID int references TransporterOrg ON DELETE SET NULL ON UPDATE CASCADE,  
36  refiner_ID int references Refiner ON DELETE SET NULL ON UPDATE CASCADE,  
37  shipment_date date NOT NULL,  
38  shipment_vol float NOT NULL,  
39  shipment_cost money NOT NULL,  
40  _from_ varchar(200) NOT NULL,  
41  _to_ varchar(200) NOT NULL,  
42  primary key(mining_ID, trans_ID, refiner_ID)  
43 );  
44  
45 create table GAS_Station_Net (  
46  net_ID int NOT NULL primary key,  
47  brand varchar(200) NOT NULL,  
48  address varchar(200)  
49 );  
50
```



```

51 create table Transit_Net (
52     refiner_ID int references Refiner ON DELETE SET NULL ON UPDATE CASCADE,
53     trans_ID int references TransporterOrg ON DELETE SET NULL ON UPDATE CASCADE,
54     net_ID int references GAS_Station_Net ON DELETE SET NULL ON UPDATE CASCADE,
55     shipment_date date NOT NULL,
56     shipment_vol float NOT NULL,
57     shipment_cost money NOT NULL,
58     _from_ varchar(200) NOT NULL,
59     _to_ varchar(200) NOT NULL,
60     primary key(refiner_ID, trans_ID, net_ID)
61 );
62
63
64 create table GAS_Station (
65     station_ID int NOT NULL primary key,
66     net_ID int references GAS_Station_Net ON DELETE CASCADE ON UPDATE CASCADE,
67     address varchar(200),
68     has_cafe bool,
69     by_card bool,
70     cash bool,
71     fps bool
72 );
73
74 create table FuelType (
75     type varchar(200) NOT NULL primary key
76 );
77
78 create table Fuel (
79     fuel_ID int NOT NULL primary key,
80     station_ID int references GAS_Station ON DELETE SET NULL ON UPDATE CASCADE,
81     type varchar(200) references FuelType ON DELETE CASCADE ON UPDATE CASCADE,
82     price money,
83     pump_num int
84 );
85
86 create table Customer (
87     customer_type varchar(200) NOT NULL primary key,
88     sale float
89 );
90
91 create table Orders (
92     order_ID int NOT NULL primary key,
93     station_ID int references GAS_Station ON DELETE SET NULL ON UPDATE CASCADE,
94     customer_type varchar(200) references Customer ON DELETE SET NULL ON UPDATE CASCADE,
95     fuel_ID int references Fuel ON DELETE SET NULL ON UPDATE CASCADE,
96     total_sum money NOT NULL
97 );
98
99
100

```


Примеры инsertов (DML код):

```
191 INSERT INTO Transit_Net VALUES (4, 6, 2, '2021-05-15', 20, 2.0, 'Россия, 410022, Красноярский край,  
Большеулуйский район, промзона НПЗ ОАО "АНПЗ ВНК.', 'ул. Старый Скит, 1/2, Дивногорск, Россия');  
192 INSERT INTO Transit_Net VALUES (5, 5, 4, '2019-11-05', 15, 0.9, '681007, Хабаровский край, г.  
Комсомольск-на-Амуре, ул. Ленинградская, д. 115', 'Комсомольск-на-Амуре, Хабаровский край, Россия,  
681017');  
193 INSERT INTO Transit_Net VALUES (1, 3, 4, '2019-09-25', 50, 3.0, '446007, Самарская обл., г.  
Новокуйбышевск, ул. Осипенко, д. 12, стр. 1', 'Самарская область, Безенчукский район, поселок  
городского типа Осинки, Роснефть АЗС №101');  
194 INSERT INTO Transit_Net VALUES (11, 8, 4, '2021-06-18', 22, 2.1, 'Россия, 446009, Самарская обл., г.  
Сызрань, ул. Астраханская, д. 1', 'Россия, Самарская область, городской округ Кинель, посёлок  
городского типа Алексеевка (справа)');  
195 INSERT INTO Transit_Net VALUES (12, 7, 5, '2020-07-23', 15, 1.0, '443004, Самарская обл., г. Самара,  
ул. Грозненская, д. 25', 'Россия, Самарская область, Красноярский район, Р-241, 65-й километр,  
241');  
196  
197 INSERT INTO GAS_Station VALUES (1, 1, 'Россия, Красноармейский район, станица Полтавская,  
Центральная улица', 1, 1, 1, 1);  
198 INSERT INTO GAS_Station VALUES (2, 1, 'Куйбышевское ш., 53, Рязань, Россия', 0, 1, 0, 0);  
199 INSERT INTO GAS_Station VALUES (3, 1, 'Кемеровская обл., автотрасса Ленинск-Кузнецкий-Прокопьевск-  
Новокузнецк, 5 км (справа)', 1, 0, 1, 1);  
200 INSERT INTO GAS_Station VALUES (4, 2, 'ул. Старый Скит, 1/2, Дивногорск, Россия', 1, 1, 1, 0);  
201 INSERT INTO GAS_Station VALUES (5, 4, 'Комсомольск-на-Амуре, Хабаровский край, Россия, 681017', 1,  
0, 1, 0);  
202 INSERT INTO GAS_Station VALUES (6, 4, 'Самарская область, Безенчукский район, поселок городского  
типа Осинки, Роснефть АЗС №101', 1, 1, 1, 0);  
203 INSERT INTO GAS_Station VALUES (7, 4, 'Россия, Самарская область, городской округ Кинель, посёлок  
городского типа Алексеевка (справа)', 1, 1, 1, 0);  
204 INSERT INTO GAS_Station VALUES (8, 5, 'Россия, Самарская область, Красноярский район, Р-241, 65-й  
километр, 241', 1, 0, 1, 0);  
205  
206 INSERT INTO FuelType VALUES ('АИ-100');  
207 INSERT INTO FuelType VALUES ('АИ-80');  
208 INSERT INTO FuelType VALUES ('АИ-92');  
209 INSERT INTO FuelType VALUES ('АИ-95');  
210 INSERT INTO FuelType VALUES ('АИ-97');  
211 INSERT INTO FuelType VALUES ('АИ-98');  
212 INSERT INTO FuelType VALUES ('Газ');  
213 INSERT INTO FuelType VALUES ('ДТ');  
214  
215 INSERT INTO Fuel VALUES (1, 1, 'АИ-100', 54.2, 1);  
216 INSERT INTO Fuel VALUES (2, 1, 'АИ-80', 44.0, 2);  
217 INSERT INTO Fuel VALUES (3, 1, 'АИ-92', 46.8, 3);  
218 INSERT INTO Fuel VALUES (4, 1, 'АИ-95', 47.2, 4);  
219 INSERT INTO Fuel VALUES (5, 1, 'АИ-97', 49.0, 5);  
220 INSERT INTO Fuel VALUES (6, 1, 'АИ-98', 49.5, 6);  
221 INSERT INTO Fuel VALUES (7, 1, 'Газ', 27.5, 7);  
222 INSERT INTO Fuel VALUES (8, 1, 'ДТ', 52.7, 8);  
223 INSERT INTO Fuel VALUES (9, 2, 'АИ-92', 46.8, 1);
```

5. Клиентское приложения

5.1. Архитектура

Наше клиентское приложение – это чат-бот в мессенджере Telegram, реализованный на языке программирования Python.

Библиотеки, использованные в работе: `sqlite3`, `aiogram`.

Реализован класс `FuelControlDB`, через который обеспечивается доступ к базе данных. Проводимые им методы (функции) описаны в пункте 4.2.

5.2. Сценарии использования

Для кого полезно наше приложение? Приложение может принести пользу тем, кто интересуется нефтегазовой логистикой, кому нужна информация по АЗС. Это могут быть сотрудники аналитического или логистического отдела сети АЗС, менеджеры нефтегазовых компаний, акционеры, сотрудники дочерних компаний, налоговая служба, или простые автомобилисты.

Данный бот позволяет своим пользователям получать информацию из базы данных по запросам, используя функции, перечисленные в предыдущем пункте. Вывод информации осуществляется очень быстро. Это важный критерий, который ценят пользователи.

Стандартный сценарий использования выглядит так:

- открываем чат с ботом в Telegram;
- нажимаем «старт» или же вводим команду `/start`;
- выбираем меню с интересующим нас запросом (кнопки `/назад`, `/далее`);
- нажимаем на запрос и просматриваем появившийся вывод;
- либо завершаем пользование, либо повторяем пункты 3, 4.

5.3. Организация доступа к данным

Пользователь имеет доступ к данным, обозначенным в кнопках, только на просмотр. Все изменения в таблицы вносят исключительно создатели проекта, так как основной целью бота является обеспечение `read-only` доступа к хранимым данным. Отмечается, что главной идеей было создать справочник.

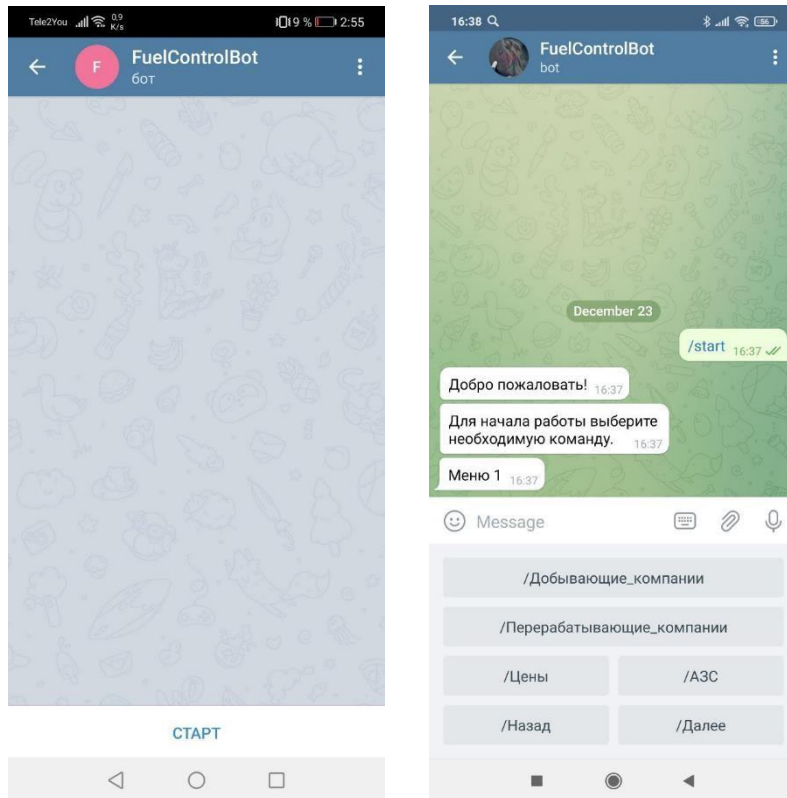
5.4. Интерфейс пользователя

Пользование приложением (чат-ботом) осуществляется через Telegram. Все взаимодействие осуществляется через кнопки (см. пункт 1.3 и 5.2).

5.5. Отчёты

Вот как выглядит работа в нашем приложении.

Начало работы с ботом:

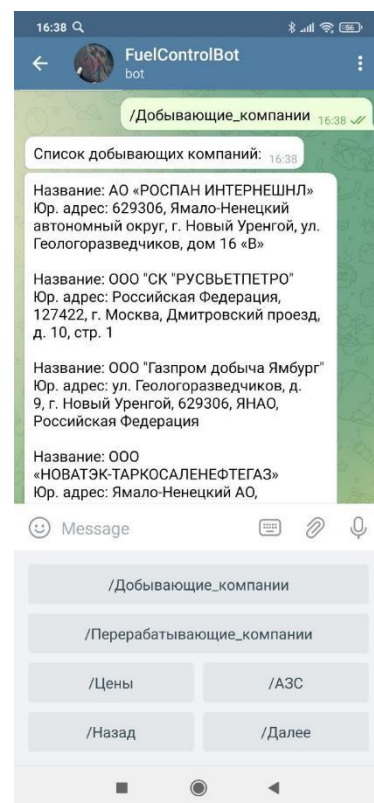


(Нажимаем на старт)

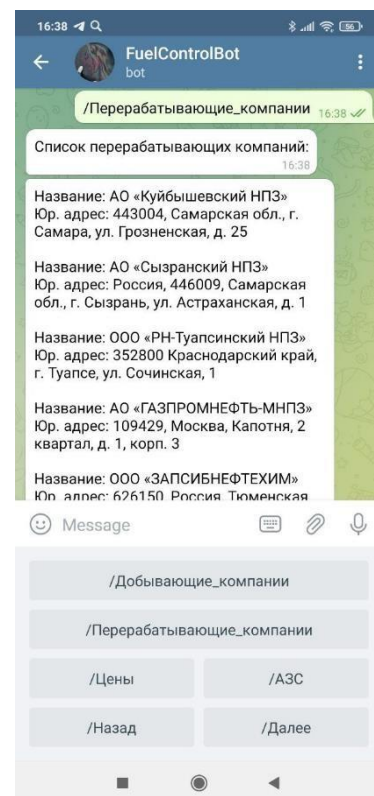
Мы оказались на уровне первого меню, где уже представлена часть команд.

Запросы:

Список добывающих организаций



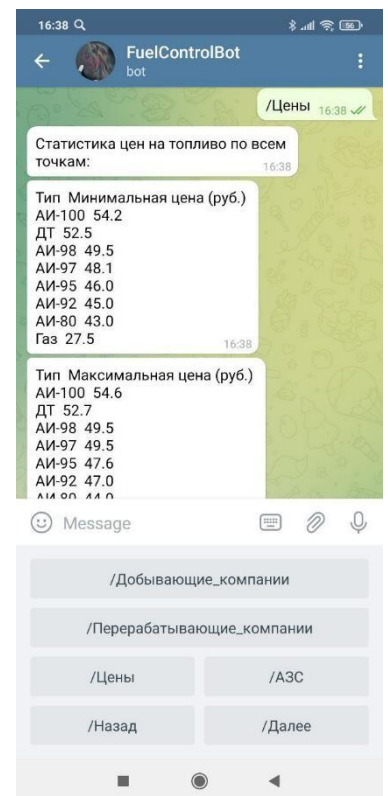
Список перерабатывающих организаций



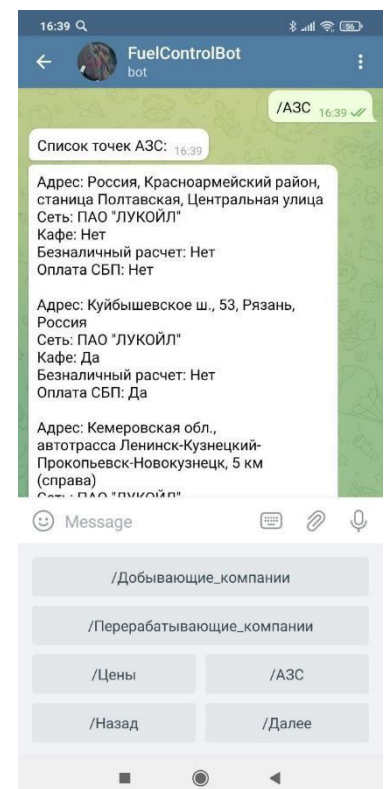
Цены →

Вид статистического запроса.

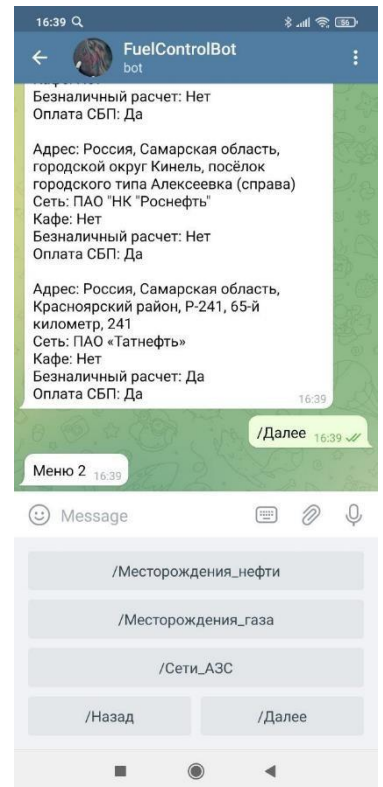
Идет подсчет минимальной / максимальной / средней цены по всем АЗС.



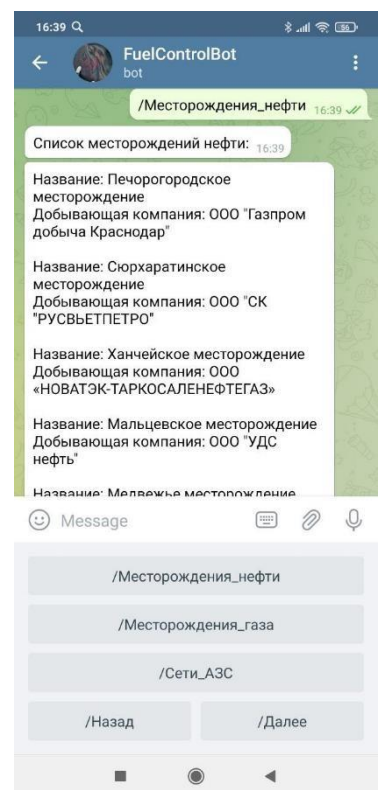
Список АЗС →



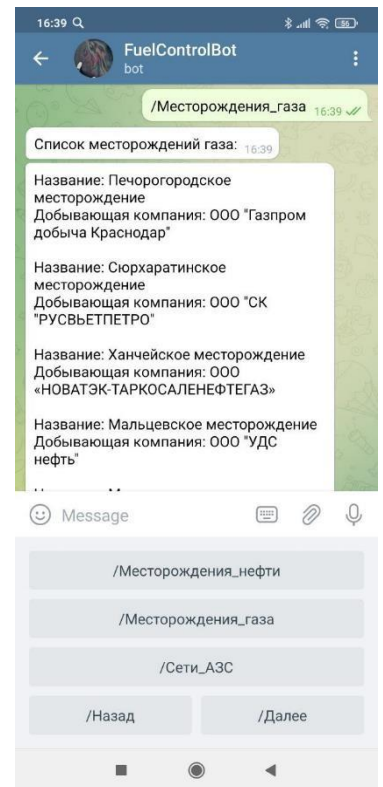
Функции в первом меню закончились. Чтобы посмотреть остальные виды выводимой нами информации, перейдем на меню 2. ➡



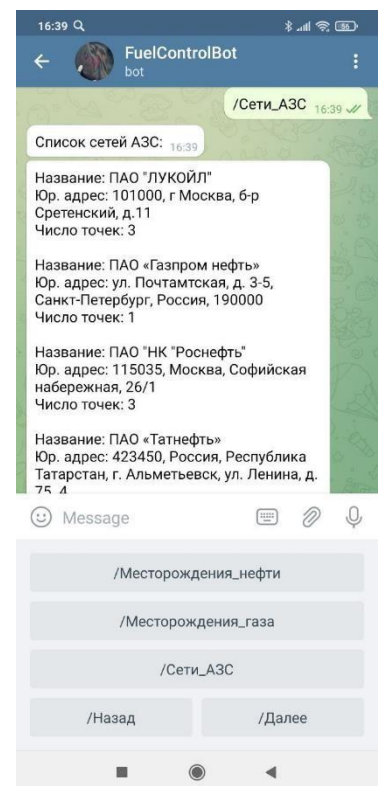
Список месторождений нефти ➡




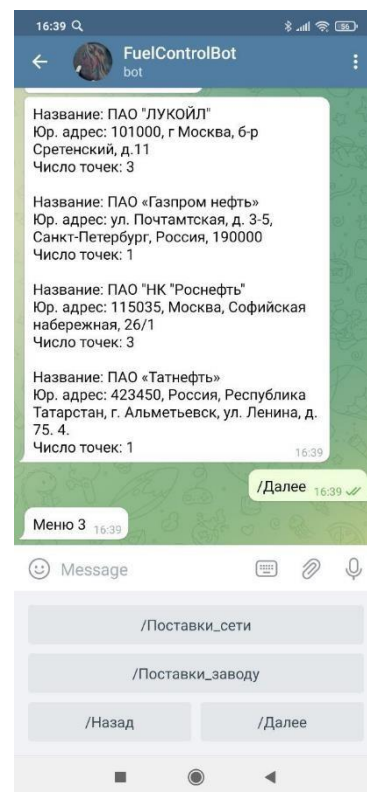
Список месторождений газа ➡




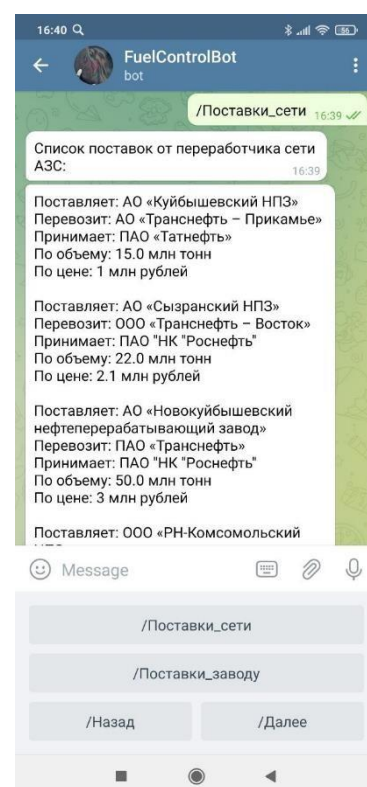
Список сетей АЗС ➡



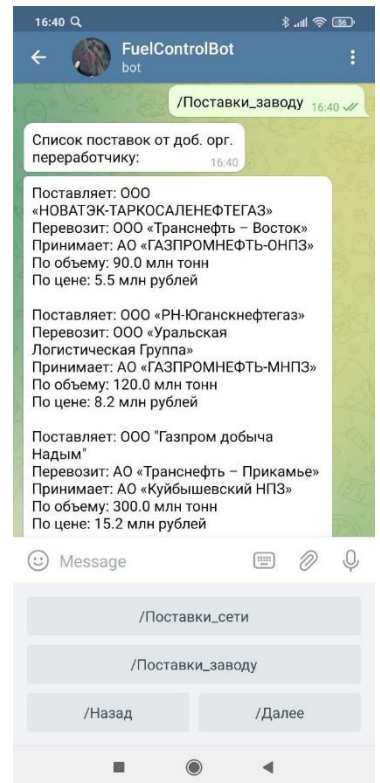
Переход к меню 3. Аналогично переходу к меню 2, рассмотренному выше 



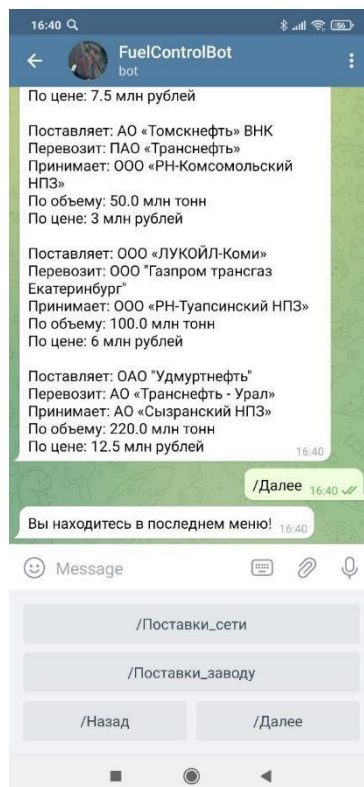
Список поставок от перерабатывающей организации к сети 



Список поставок от добывающей организации к перерабатывающей ➡



Как выглядит защита от неправильных действий клиента



6. Заключение

6.1. Объёмные характеристики разработки

В процессе работы над проектом было выделено 10 сущностей, каждая из которых имеет от 1 до 6 атрибутов. Была создана база данных. Она состоит из 13 отдельных табличек, соединённых различными видами связи.

Бот реализуется с помощью 12 написанных на python файлов. Умеет выводит общую информацию об объектах и статистику по ним.

Количество реализованных функций в приложении: 9.

6.2. Авторский вклад и комментарии по выполнению проекта

При выборе идеи мы не были знакомы с аналогами нашего проекта, считали его уникальным. Однако в процессе работы, столкнулись с наличием чат-ботов по смежным темам. Проект оказался достаточно объёмным, и будь у нас больше времени, возможного на реализацию, были бы реализованы более сложные запросы обращения к БД и дополнительный функционал, например, вывод информации на карте.

За время выполнения данной работы наша команда научилась работать с БД и продумывать детали ее реализации.

6.3. Репозиторий

Ссылка на репозиторий: https://github.com/ArtIrina/TDB_project

7. Источники

1. energybase.ru [Электронный ресурс] / Режим доступа: <https://energybase.ru> свободный. (дата обращения: 18.12.2021)
2. Ternary Relationship in ER Modeling / Trevor H. Jones & Il-Yeol Song. – College of Information Studies Drexel University, Philadelphia, P.A., 19104, 1993. – 30с.
3. Лукойл [Электронный ресурс] / Режим доступа: <https://lukoil.ru/> свободный. (дата обращения: 18.12.2021)
4. Роснефть [Электронный ресурс] / Режим доступа: <https://www.rosneft.ru/> свободный. (дата обращения: 18.12.2021)
5. Газпром нефть [Электронный ресурс] / Режим доступа: <https://www.gazprom-neft.ru/> свободный. (дата обращения: 18.12.2021)
6. TATNEFT [Электронный ресурс] / Режим доступа: <https://www.tatneft.ru/> свободный. (дата обращения: 18.12.2021)
7. Нефть Магистраль [Электронный ресурс] / Режим доступа: <https://neftm.ru/> свободный. (дата обращения: 18.12.2021)