МФТИ

Finassassin

Telegram-bot финансовый ассистент

Финансовые технологии и аналитика

Выполнил: Бережной Александр Александрович

ОБЛАСТЬ ПРИМЕНЕНИЯ БОТА

Бот сможет помочь пользователям на лету получать данные с биржи и центрального банка, пользоваться yandex GPT. Целевая аудитория - все интересующиеся финансами и инвестициями: начинающие инвесторы, сотрудники финансовых организаций, сэлеры маркетплэйсов, предприниматели и другие.

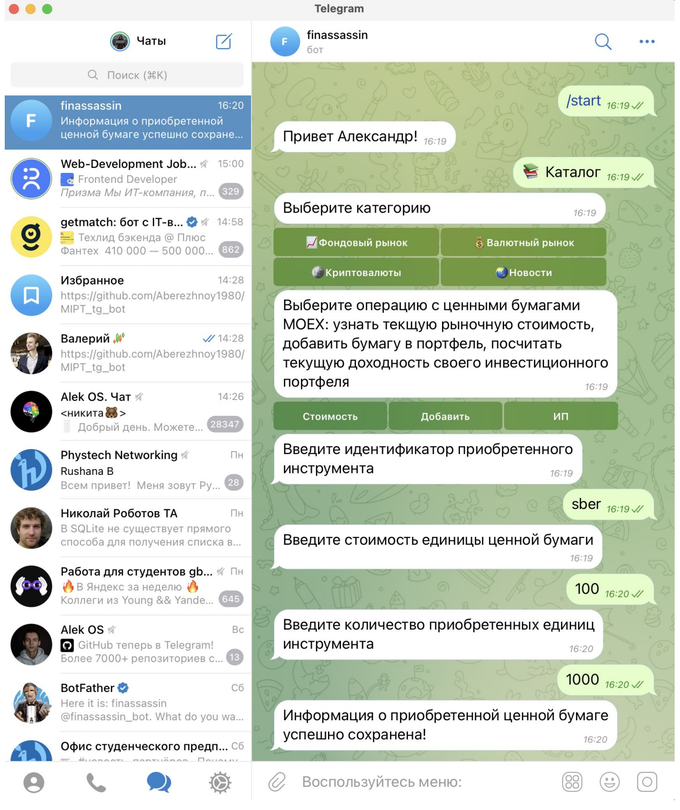
ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Функция	Описание	Данные
Организация постоянного хранения регистрационных данных пользователей	При запуске серверной части бота инициируется создание базы данных и таблицы для хранения метаданных пользователя	• telegram_id • Имя пользователя (из учетной записи) • email
Регистрация нового пользователя	Новому пользователю, для дальнейшего использования бота, предлагается зарегистрироваться. Зарегистрированного пользователя бот приветствует по имени	• email (для рассылки) • Имя и telegram_id берутся из учетной записи • Другие параметры профиля пользователь сможет настраивать самостоятельно
Получение стоимости ценной бумаги по тикеру	У пользователя запрашивается тикер, формируется запрос по API к Московской бирже, проверяется ответ, информация отдается пользователю. Ошибки ввода обрабатываются	• Тикер эмитента • Стоимость и валюта # - sber - 289,43 RUB
Анализ инвестиционного портфеля	Пользователю предлагается сформировать свой инвестиционный портфель для мониторинга его прибыльности. Он может добавлять неограниченное количество ценных бумаг и рассчитывать отклонения номинальной стоимости портфеля от текущей. Все данные хранятся в базе данных	• Номаинальная стоимость (агрегируется) • Текущая стоимость (агрегируется) • Значение абсолютного и относительного отклонение # - Номинальная стоимость портфеля **200,000.00 RUB** - Текущая стоимость портфеля **250,000.00 RUB** - Прибыль **50,000.00 RUB**

ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Функция	Описание	Данные
Получение котировок валют и сырья	У пользователя запрашивается валюта, формируется запрос по API к Центральному Банку РФ, проверяется ответ, информация отдается пользователю. Ошибки ввода обрабатываются	● Тикер валюты ● Стоимость в рублях # - USD - 85.37 RUR/USD
Конвертер валют на дату	Обычный конвертер валют на указанную пользователем дату. Обрабатываются неторговые дни и ошибки ввода.	● Тикер валюты источника ● Сумма ● Тикер валюты назначения ● Результат # - USD -> 20 -> RUR -> 21.12.1997 - 600 RUR
База знаний	В качестве базы знаний, для повышения финансовой грамотности, бот подключен по API к Yandex GPT, что в свою очередь не ограничивает пользователя задавать вопросы на любые темы	● Текст # - Сколько сантиметров в одном дюйме? - 2,54
Сбор и суммаризация новостей	По запросу пользователь получает краткую выдержку из новостной ленты	текст

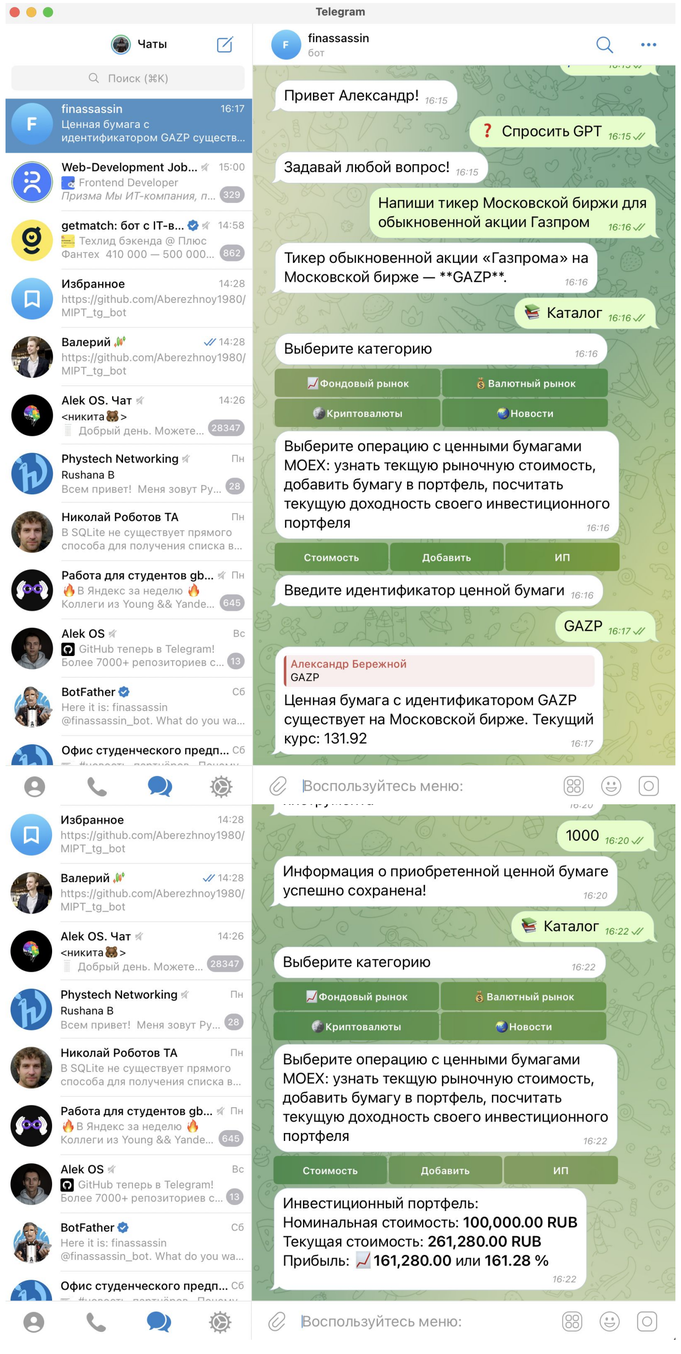
СЦЕНАРИИ



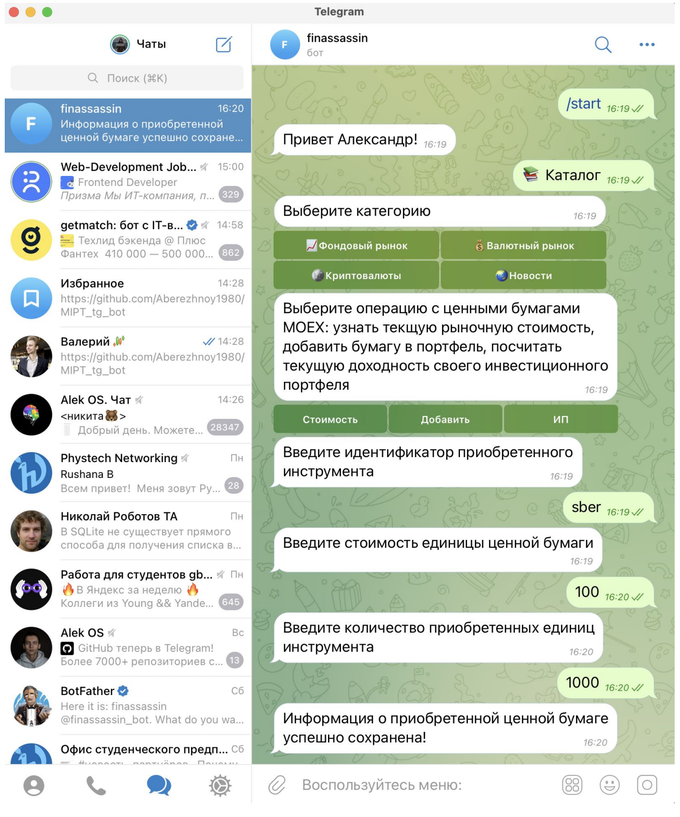
С помощью кнопок меню или команды /checkStock пользователь проверяет стоимость интересующей его бумаги, попутно уточнив у GPT наименование тикера при необходимости:

С помощью кнопок меню или команды /addStock пользователь добавляет выбранные тикер, количество и стоимость бумаги в базу данных:

С помощью кнопок меню или команды /checkPortfolioSummary пользователь получает расчеты по своему инвестиционному портфелю:



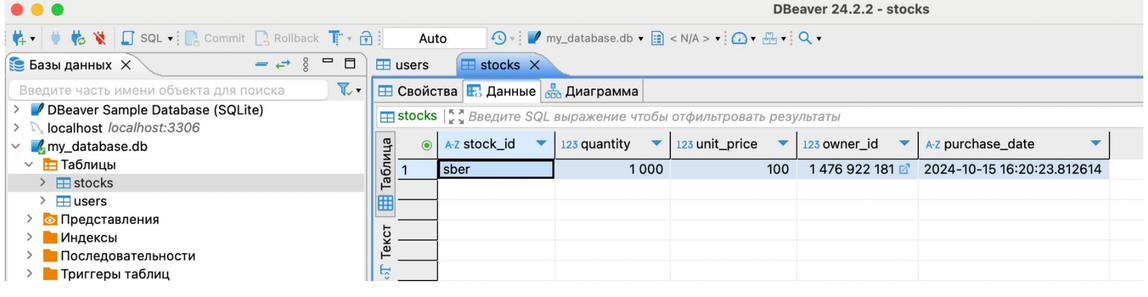
СЦЕНАРИИ

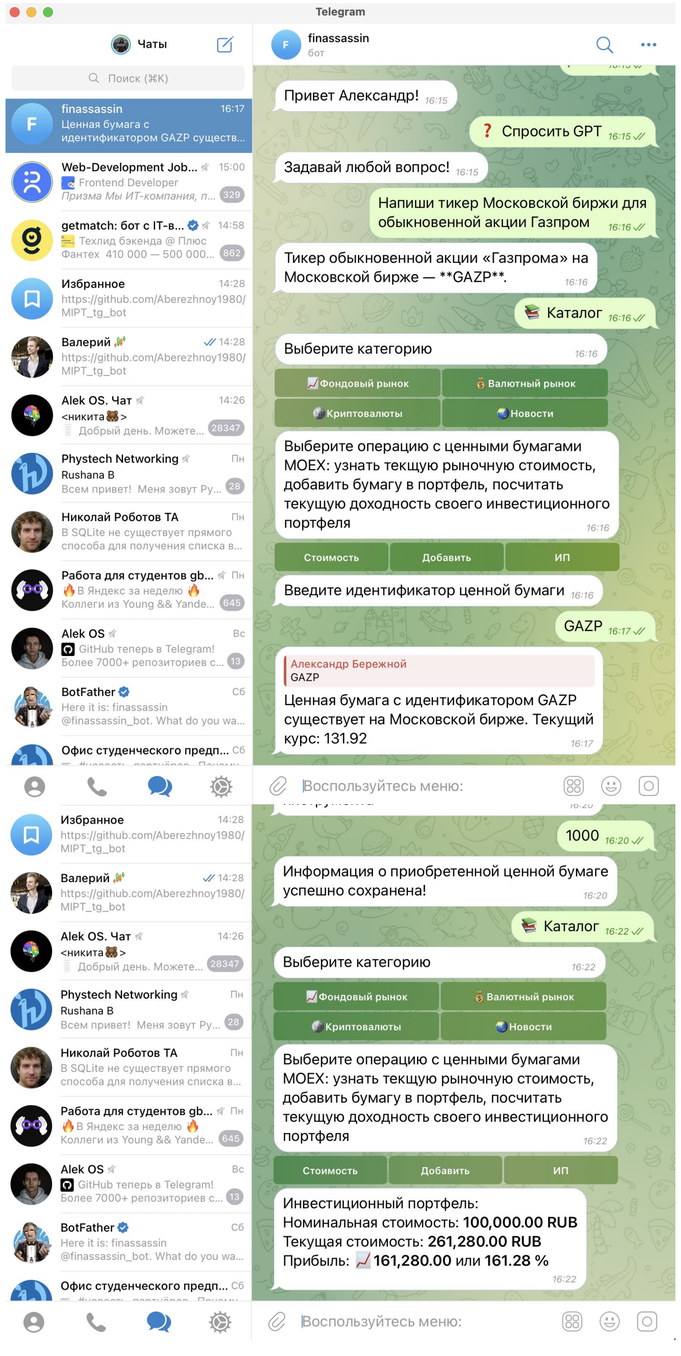


С помощью кнопок меню или команды /checkStock пользователь проверяет стоимость интересующей его бумаги, попутно уточнив у GPT наименование тикера при необходимости:

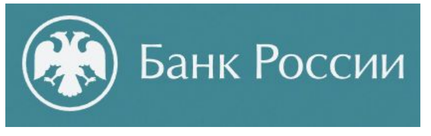
С помощью кнопок меню или команды /addStock пользователь добавляет выбранные тикер, количество и стоимость ценной бумаги в базу данных:

С помощью кнопок меню или команды /checkPortfolioSummary пользователь получает расчеты по своему инвестиционному портфелю:





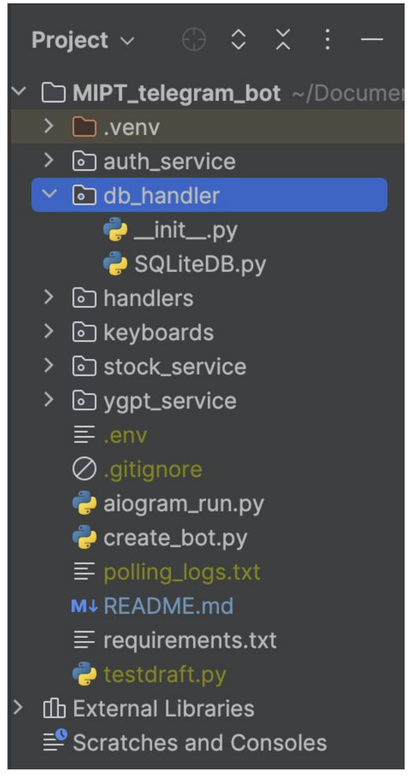
ИНТЕГРАЦИИ С ВНЕШНИМИ СЕРВИСАМИ



ИНТЕГРАЦИЯ С БД (SQLite) - АВТОМАТИЗАЦИЯ

```python
import sqlite3


9 usages  Aleksandr Berezhnoy
class SQLiteDB:

    Aleksandr Berezhnoy
    def __init__(self, db_name):
        self.db_name = db_name

    Aleksandr Berezhnoy
    def __enter__(self):
        self.conn = sqlite3.connect(self.db_name)
        self.cursor = self.conn.cursor()
        return self

    Aleksandr Berezhnoy
    def __exit__(self, exc_type, exc_val, exc_tb):
        self.conn.commit()
        self.conn.close()

    2 usages  Aleksandr Berezhnoy
    def create_table(self, table_name, columns):
        columns_str = ', '.join(columns)
        self.cursor.execute(f"CREATE TABLE IF NOT EXISTS {table_name} ({columns_str})")

    2 usages  Aleksandr Berezhnoy
    def insert_data(self, table_name, *data):
        placeholders = ', '.join(['?'] * len(data))
        self.cursor.execute( __sql: f"INSERT INTO {table_name} VALUES ({placeholders})", data)

    2 usages  Aleksandr Berezhnoy
    def select_data(self, table_name, columns=None, condition=None):
        if columns:
            columns_str = ', '.join(columns)
        else:
            columns_str = '*'

        if condition:
            self.cursor.execute(f"SELECT {columns_str} FROM {table_name} WHERE {condition}")
        else:
            self.cursor.execute(f"SELECT {columns_str} FROM {table_name}")

        return self.cursor.fetchall()
```

В проекте автоматизировано взаимодействие с SQLite с помощью модуля **db_handler** содержащего класс **SQLiteDB**

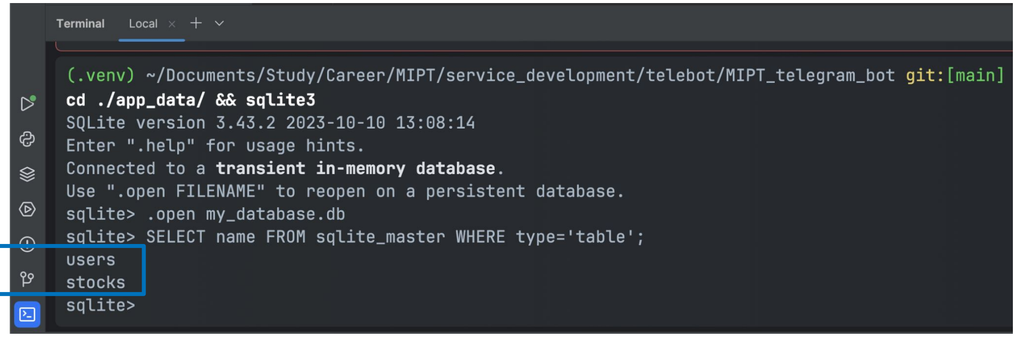


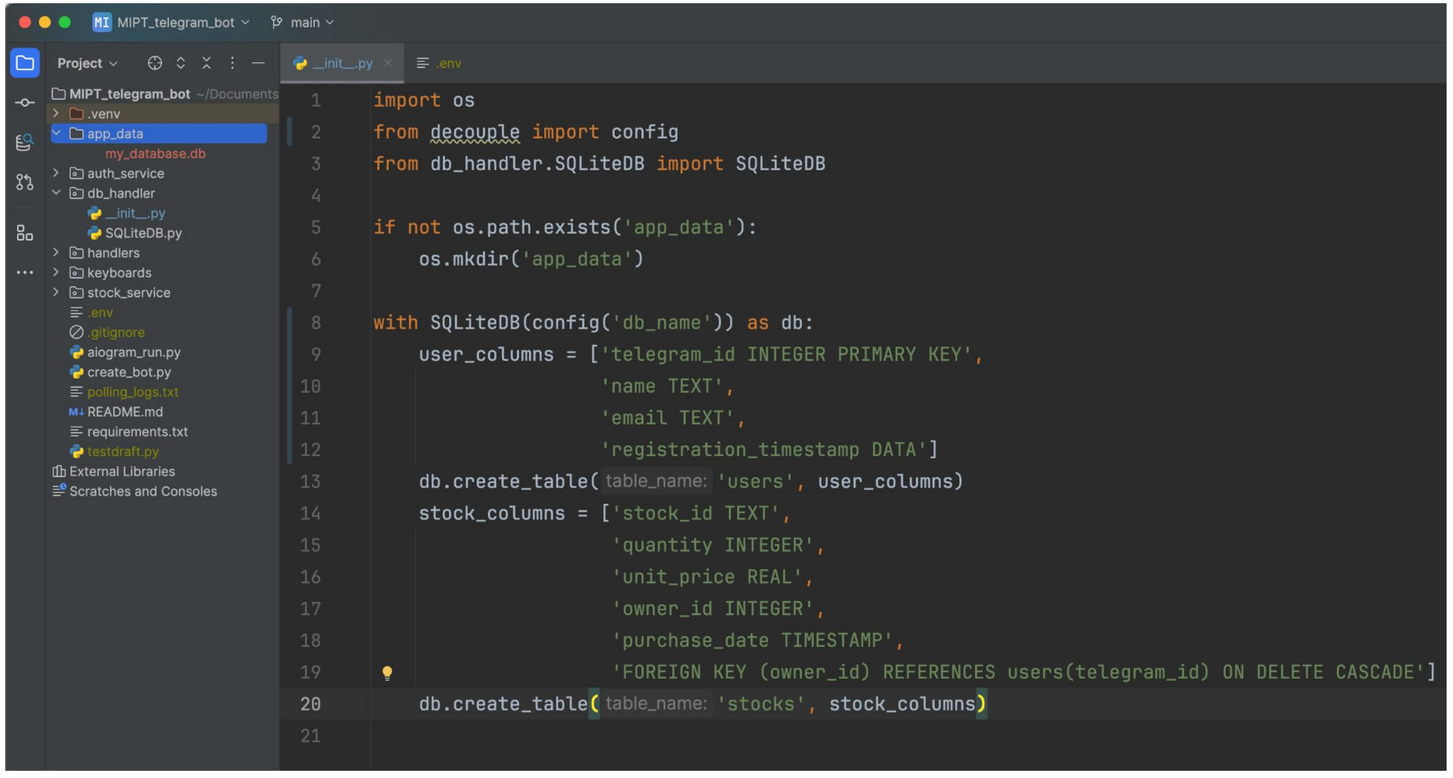
Сервисный класс **SQLiteDB** переопределяет методы __enter__ и __exit__, что позволяет использовать оператор **with** при работе с классом для корректного управления соединением с БД, а также содержит базовые методы для CRUD операций

ИНТЕГРАЦИЯ С БД (SQLite) - ИНИЦИАЛИЗАЦИЯ

При первом обращении к модулю, в корне проекта создается рабочая директория, и с помощью методов сервисного класса SQLiteDB инициализируется база данных и создаются необходимые таблицы

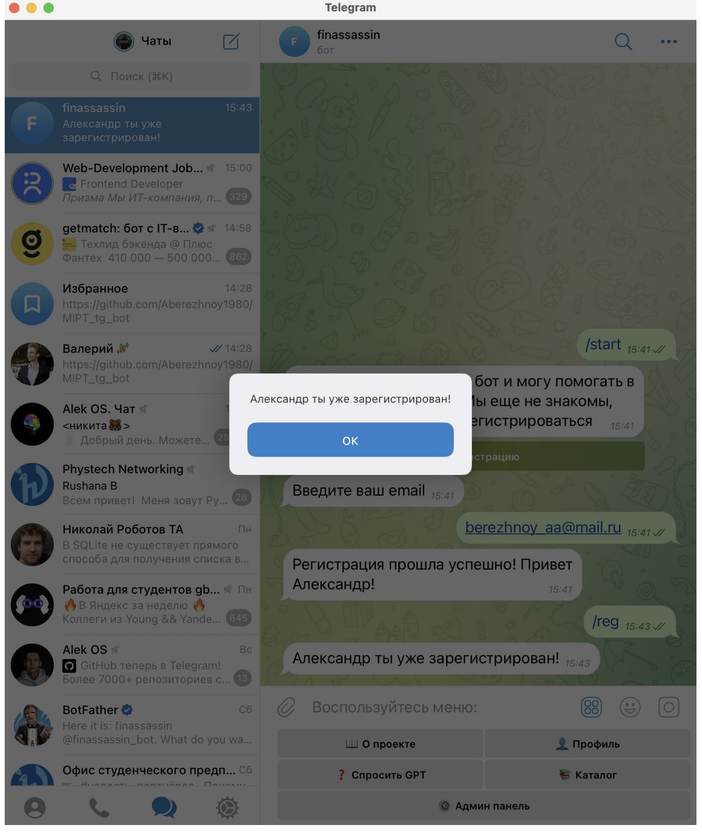
В терминале IDE переходим в автоматически созданную директорию и запускаем клиент SQLite3. Видим что БД создана и содержит две таблицы: users и stocks





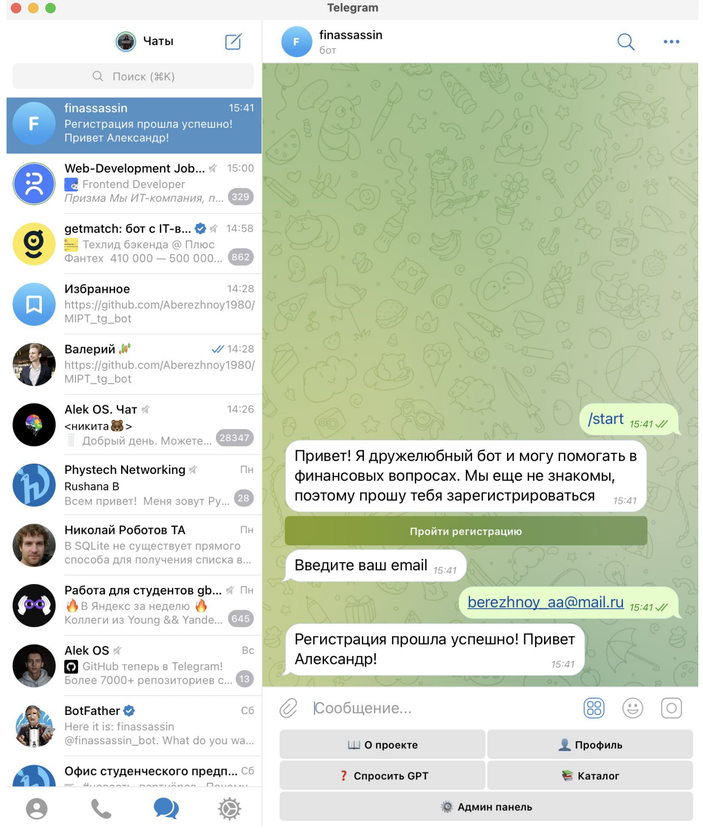
ИНТЕГРАЦИЯ С БД (SQLite) - ДАННЫЕ

```
/Users/alex/Documents/Study/Career/MIPT/service_development/telebot/MIPT_telegram_bot/.venv/bin/python /Users/alex/Documen
/Users/alex/Documents/Study/Career/MIPT/service_development/telebot/MIPT_telegram_bot/.venv/lib/python3.9/site-packages/ur
  warnings.warn(
2024-10-15 15:32:21,619 - aiogram.dispatcher - INFO - Start polling
2024-10-15 15:32:21,708 - aiogram.dispatcher - INFO - Run polling for bot @finassassin_bot id=7603118234 - 'finassassin'
```



Backend бота запущен на сервере - попробуем зарегистрироваться и сохранить метаданные пользователя в БД. Запускаем клиент телеграм:

Повторная попытка регистрации через команду обрабатывается, также, как и обрабатывается попытка повторить регистрацию через inline-кнопку



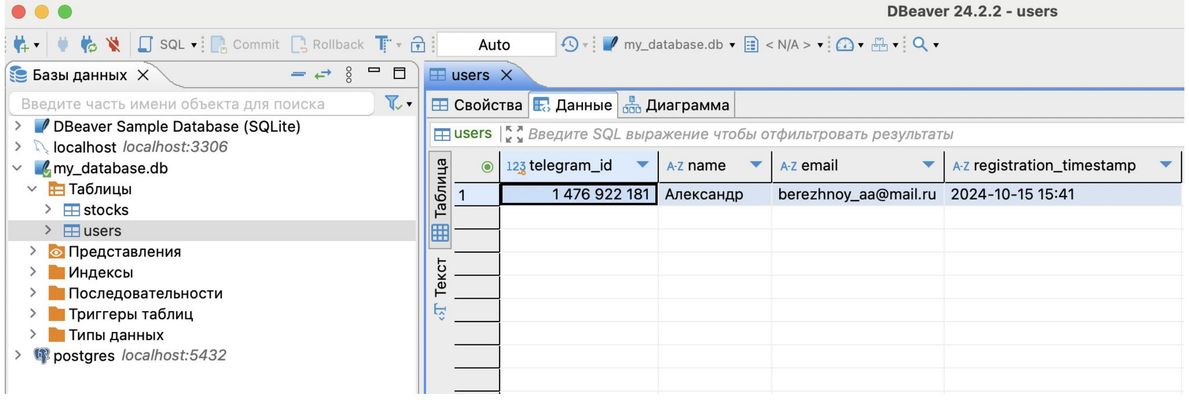
ИНТЕГРАЦИЯ С БД (SQLite) - ДАННЫЕ

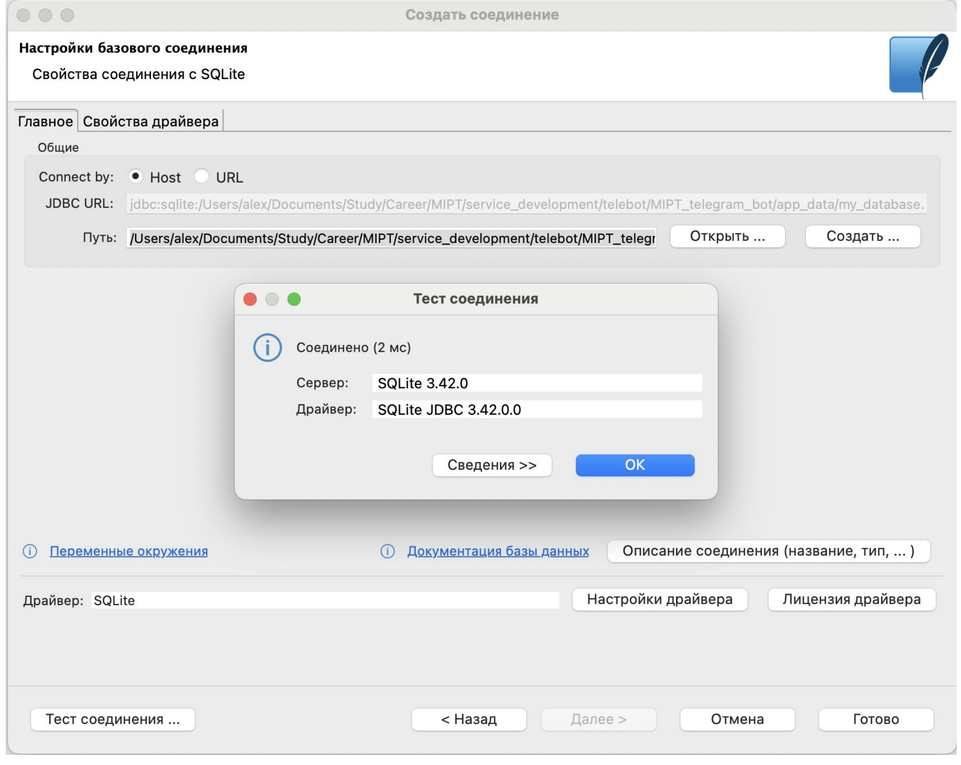
```
(.venv) ~/Documents/Study/Career/MIPT/service_development/telebot/MIPT_telegram_bot git:[main]
cd ./app_data/

(.venv) ~/Documents/Study/Career/MIPT/service_development/telebot/MIPT_telegram_bot/app_data git:[main]
sqlite3
SQLite version 3.43.2 2023-10-10 13:08:14
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open my_database.db
sqlite> SELECT * FROM 'users';
1476922181|Александр|berezhnoy_aa@mail.ru|2024-10-15 15:41
sqlite>
```

✅

Посмотрим на базу данных через терминал:

Подключимся к базе данных через UI-клиент - DBeavar. Проверим соединения с базой данных нажатием кнопки “Тест соединения…”. Проверим структуру и данные БД





МФТИ

Спасибо за внимание!