**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)

Кафедра «Математическая кибернетика и информационные технологии»

# Отчет по учебной практике (ознакомительной)

Выполнил: студент группы БВТ2304 Никулина И.Д.

Руководитель: _____

Москва
2024

# Оглавление

# Введение.

Целью учебной практики является реализация полученных теоретических и практических знаний, умений и навыков в ходе прохождения курса, в следствии чего была установлена следующая цель работы: “Разработка платформы парсинга данных о вакансиях с платформы hh.ru”.

В ходе постановки цели работы, были поставлены следующие задачи:

1. Изучить существующие платформы для парсинга (beautiful soup, selenium или API платформ).
2. Сформулировать функциональное требования к системе.
3. Спроектировать базу данных.
4. Написать инструкцию пользователя.
5. Провести тестирование системы.

При постановке задач, были выдвинуты следующие требования к функционалу:

1. Возможность формирования запроса для парсинга данных по таким полям как название должности, навыки, формат работы и т.п.
2. Загрузка в базу данных информации по результатам парсинга данных и вывод аналитики по параметрам.

За время прохождения учебной практики были достигнуты все поставленные цели и задачи. Было разработано приложение парсинга сайта hh.ru при помощи API сервиса. Для интерфейса был использован телеграм бот, а для хранения данных база данных mysql

# 1. Архитектура приложения.

Приложение состоит из одной программы которая содержит в себе 4 файла

Проект написан на языке python и содержит такие файлы как main.py, bot.py, db.py, vacancies.py

Разберем каждый из файлов

bot.py представляет из себя файл с описанием логики работы телеграм бота

Бот был сделан при помощи библиотеки aiogram для python

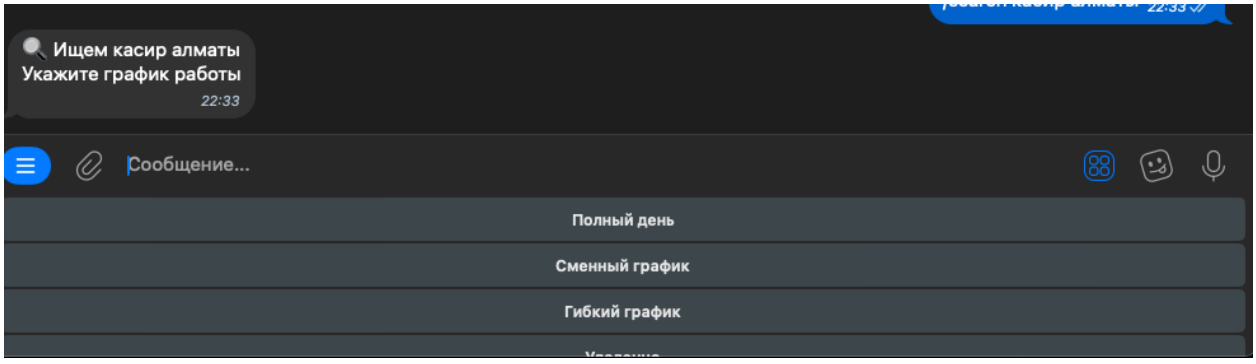
Пользуясь документацией был написан код. Суть заключается в том, что приложение реагирует на события, в данном случае использование команды пользователем. В зависимости от команды пользователя бот выполняет ту или иную задачу. Бот имеет три команды: /start, /search, /stats

Команда start (используется при первом запуске бота), получает имя пользователя, чтобы написать в ответ «Привет, ИМЯ» и отправляет краткую инструкцию пользования ботом


Бот предлагает пользователю найти вакансии используя команду search

При использовании команды, бот отделяет запрос пользователя от команды и обращается к файлу parser.py (его работа будет описана позже)

Далее будет предложено указать график работы

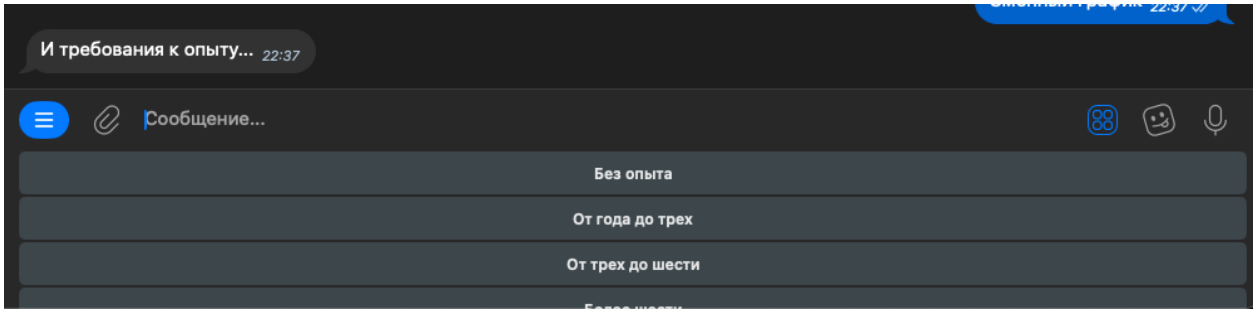


Указание графика работы производится посредством кнопок на клавиатуре, при помощи указания reply_markup

Суть в том, что мы создаем кнопки и отправляем их пользователю, чтобы было удобнее выбирать из чего то

```
exp = []

exp.append([KeyboardButton(text='Без опыта')])
exp.append([KeyboardButton(text='От года до трех')])
exp.append([KeyboardButton(text='От трех до шести')])
exp.append([KeyboardButton(text='Более шести')])

await message.answer(f'И требования к опыту...', reply_markup=ReplyKeyboardMarkup(keyboard=exp,
await state.set_state(Form.exp)
```

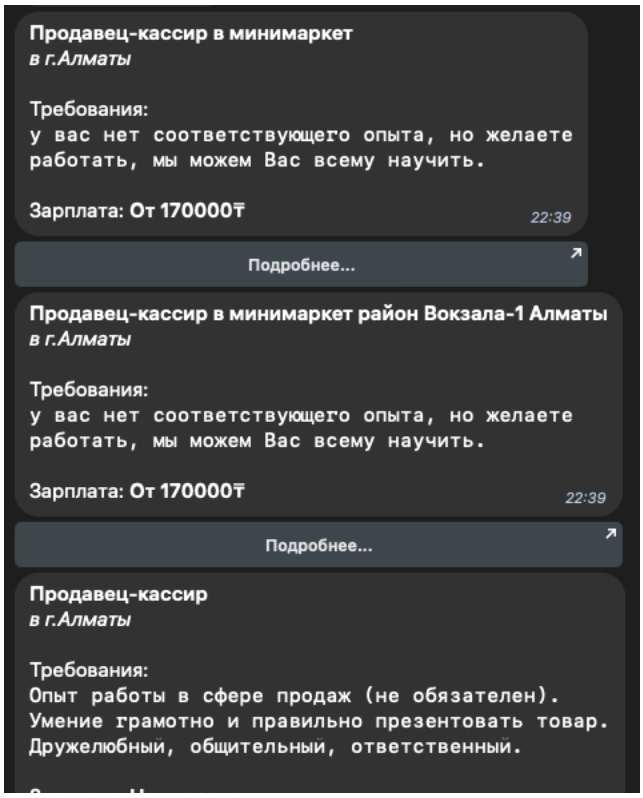
Далее пользователю будет предложено требования к опыту




Кнопки реализованы аналогично

Создание анкеты было сделано через FSMContext (конечный автомат в aiogram), когда пользователь обращается к боту, ему предлагается заполнить «анкету», полученные данные записываются, и предлагаются следующие данные для заполнения. И так до момента пока анкета не будет заполнена

После заполнения анкеты, бот отправляет пользователю параметры которые он выбрал и производит поиск по этим параметрам, отправляя каждую вакансию в отдельном сообщении




В сообщении указано названии вакансии, требования, зарплата и реализована inline ссылка на вакансию, которая перебросит пользователя на страницу с вакансией.

Inline - это кнопка под сообщением, к которой привязано какое то действие, в данном случае открытие ссылки

Принцип работы похож на кнопки в клавиатуре, с тем отличием что у inline обязательно должно быть какое-то действие, в то время как кнопки клавиатуры просто быстрый способ отправить уже заготовленное сообщение

```
btns = []

btns.append([InlineKeyboardButton(text="Подробнее...", url=vac['url'])])



await message.answer(html.bold(vac['name']) +
                     f'\n{html.italic('в г.' + vac['area'])}\n\nТребования: \n' +
                     html.code(vac['req']) +
                     f'\n\nЗарплата: {html.bold(vac['sal'])}', reply_markup=InlineKeyboardMarkup(inline_keyboard=btns))
```

У бота есть команда /stats, которая отобразит статистику бота, количество пользователей и самый популярный запрос

Бот получает вакансии при помощи файла vacancies.py, разберем его подробнее

Файл содержит только одну функцию get(), которая делает запрос к API hh.ru и получает ответ из 5 первых вакансий, больше просто грузило бы проект и было бы неудобно просматривать такое большое кол-во вакансий.

Запрос производится по таким параметрам как название, график, опыт

Когда мы получаем ответ от API, мы получаем не всегда нужные данные, поэтому на базе ответа необходимо построить свой собственный список

Строим его при помощи цикла for перебирая каждую вакансию в ответе и оставляя лишь необходимые данные (название, город, зп, требования, ссылка) и добавляем вакансию в список, в последствии функция возвращает этот список

Так же функция добавляет запись о том сколько было найдено вакансий по конкретному запросу в базу данных используя файл db.py

Файл db.py подключается к базе данных MySQL и выполняет необходимые действия

Например, есть функция createUser(), которая добавляет запись о новом пользователе бота

А функция createRecord() делает запись о кол-ве вакансий по определенному запросу

Позже эти данные можно получить из базы данных используя getUserCount() и getPopular() (используется в боте при использовании команды /stats)

В базе данных есть две таблицы: users и stats

В одной хранятся id пользователей, которые когда либо запускали бота

Во второй записи о все запросах

Так же есть файл main.py

Это главный файл программы из которого запускается бот, создаются таблицы в базе данных, если они еще не созданы

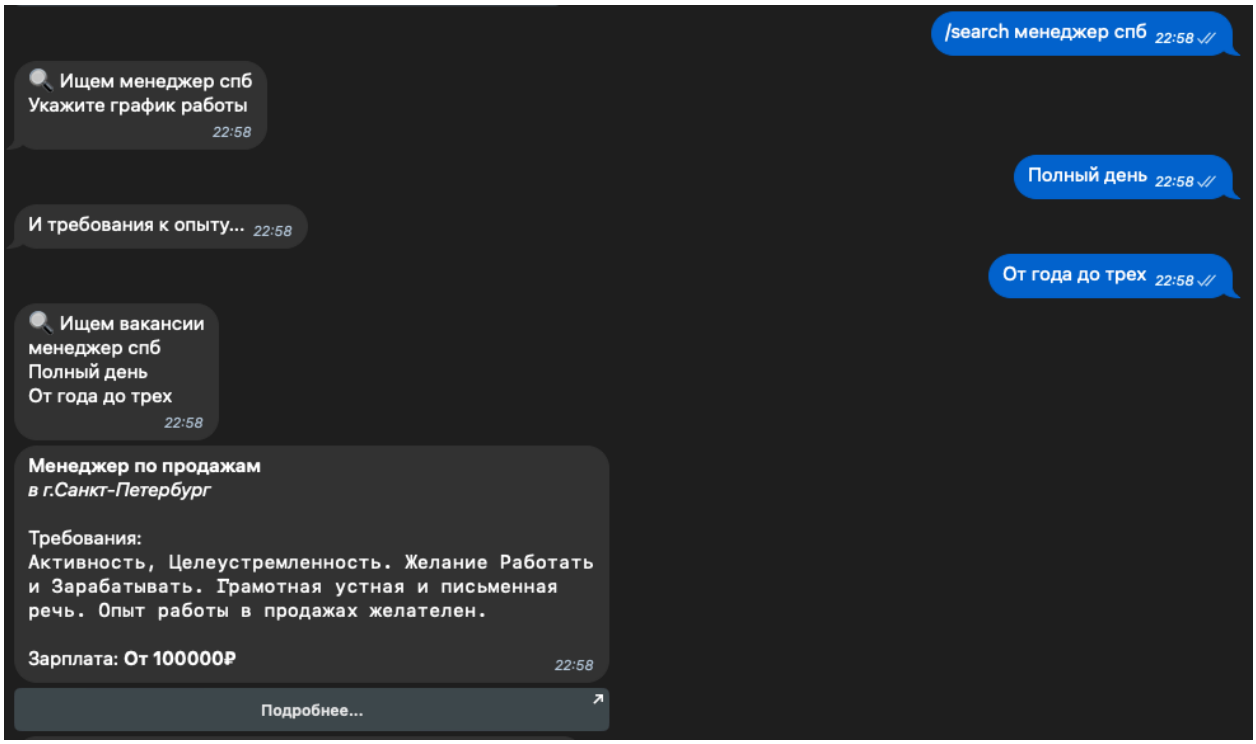
```
import bot
import asyncio
import logging
import sys
import db

if __name__ == "__main__":
    db.createUsers()
    db.createStats()
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    asyncio.run(bot.main())
```

Все приложение запускается через команду python3 main.py

## 2. Тестирование

Тестирование приложения было проведено посредством поиска различных вакансий

Например, попробуем найти вакансии менеджера в Санкт-Петербурге на полный день и с опытом от года до трех лет




Указываем название, занятость и опыт, нажимаем кнопку поиска

По итогу получаем данные, все выглядит хорошо, делаем вывод, что все работает как надо

# 3. Docker

Тестирование проведено успешно, поэтому упаковываем приложение в docker

Предварительно необходимо создать файл requirements.txt в котором хранится информация о необходимых для работы приложениях библиотеках

requirements.txt созданы при помощи команды pip freeze > requirements.txt

Так же был создан файл docker-compose.yml и Dockerfile в корне проекта

Были указаны сервисы backend, frontend, mysql

При упаковке сервисов названия хостов для подключений, например к базе данных или фронта к бэкенд части необходимо поменять localhost на названия сервиса из docker-compose

В конечном итоге, вызвав команду docker-compose up –build в корневой директории проект успешно собирается и работает

Все приложение загружено на GitHub, поэтому развернуть его можно где угодно с использованием лишь пары команд

Ссылка на GitHub: https://github.com/nikulinaxo/practice-bvt2304/

## 4. Развертывание на сервере

Для того чтобы развернуть приложение, я арендовала vps сервер на операционной системе ubuntu

После подключения к серверу, необходимо установить git, docker и docker-compose командами

sudo apt install git

sudo apt install docker

sudo apt install docker-compose

После чего нужно скачать репозиторий с GitHub

git clone https://github.com/nikulinaxo/practice-bvt2304.git

Переходим в скаченный репозиторий командой cd practice-bvt2304

Запускаем проект посредством

docker-compose up --build

Открываем диалог с ботом и проверяем работу
Бот работает постоянно, ссылка на бота https://t.me/vacancies_practice_bot

## Заключение.

Подводя итог учебной практики, можно заявить, что благодаря пройденному заданию мы не только смогли создать рабочее приложение, но и закрепили и получили следующие знания и навыки:

1. Знания о работе различных библиотек, и API.
2. Знания о создании Docker-контейнера для развертывания приложения.
3. Практический опыт в написании кода на Python с применением различных методов и обработки ошибок.
4. Написания бота на aiogram используя асинхронное програмирование
5. Знание и практических опыт создания SQL баз данных и таблиц,

# Список использованных источников.


1. https://api.hh.ru/openapi/redoc
2. https://docs.docker.com/engine/api/sdk/
3. https://dev.mysql.com/doc/
4. https://docs.aiogram.dev/en/latest/