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 Дубровин А.П.

Руководитель: _____

Москва
2024

Оглавление

Введение……...3

1. Архитектура приложения……..…….4

2. CareerBackend…….....5

3. CareerFrontend…..……..……...6

4. Тестирование……9

5. Docker……..10

6. Развертывание на сервере……...…11

Заключение…….12

Список литературы……13

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 создан сайт. Все сервисы были упакованы в Docker контейнеры.

Для ознакомления с кодом проекта и проверки результатов работы, вы можете перейти по следующим ссылкам:

https://github.com/Pe4en1eInMTUCI/CareerFinder - Репозиторий, где можно ознакомится с кодом, и инструкцией по запуску.

http:// http://109.120.135.64:8080/ - Уже развернутый проект.

1. Архитектура приложения.

Приложение состоит из таких сервисов как CareerBackend, CareerFrontend, база данных MySQL. Каждый сервис упакован в контейнер Docker.

CareerBackend – backend сервис. Написан на flask, работает с API https://api.hh.ru и базой данных MySQL

CareerBackend – frontend сервис. Написан без использования фреймворков, запускается через Flask.

MySQL база данных используется для хранения данных

Приложение запускается посредством docker-compose up --build

2. CareerBackend

Разработка проекта была начата с написания backend сервиса. Backend сервис создан на Flask. Были прописаны следующие роуты: /getVacancies, /getStats, /getAll

Сервис состоит из трех модулей: main.py – основной, parser.py – для работы с api, DataWorker.py – для работы с базой данных

Роут /getVacancies обращается к модулю parser методу getVacancies

Данный метод обращается к API hh.ru и ищет вакансии по запросам пользователя, учитывая фильтры указанные пользователем, такие как график и требования к опыту. Метод получает первые 100 вакансий по запросу и форматирует данные (например, красивое отображение зарплат со знаком валюты) оставляя только необходимые. Метод возвращает 100 вакансий по запросу пользователя только с необходимыми требованиями (Зарплата, требования, город, ссылка на вакансию)

При каждом вызове метода, полученные данные в сохраняются в базу данных

При первом вызове обращается к методу DataWorker.createTable() который создает таблицу requests в базе данных, позже в нее будут записываться данные о количестве полученных вакансий

Роут /getMost обращается к модулю DataWorker методу getMost

Данный метод возвращает запрос с самым большим количеством вакансий

Роут /getAllt обращается к модулю DataWorker методу getAll

Данный метод возвращает информацию о всех запросах с количеством найденных для них вакансий

Модуль parser.py обращается к api при помощи библиотеки requests

Модуль DataWorker.py обращается к БД MySQL при помощи библиотеки mysql-connector-python

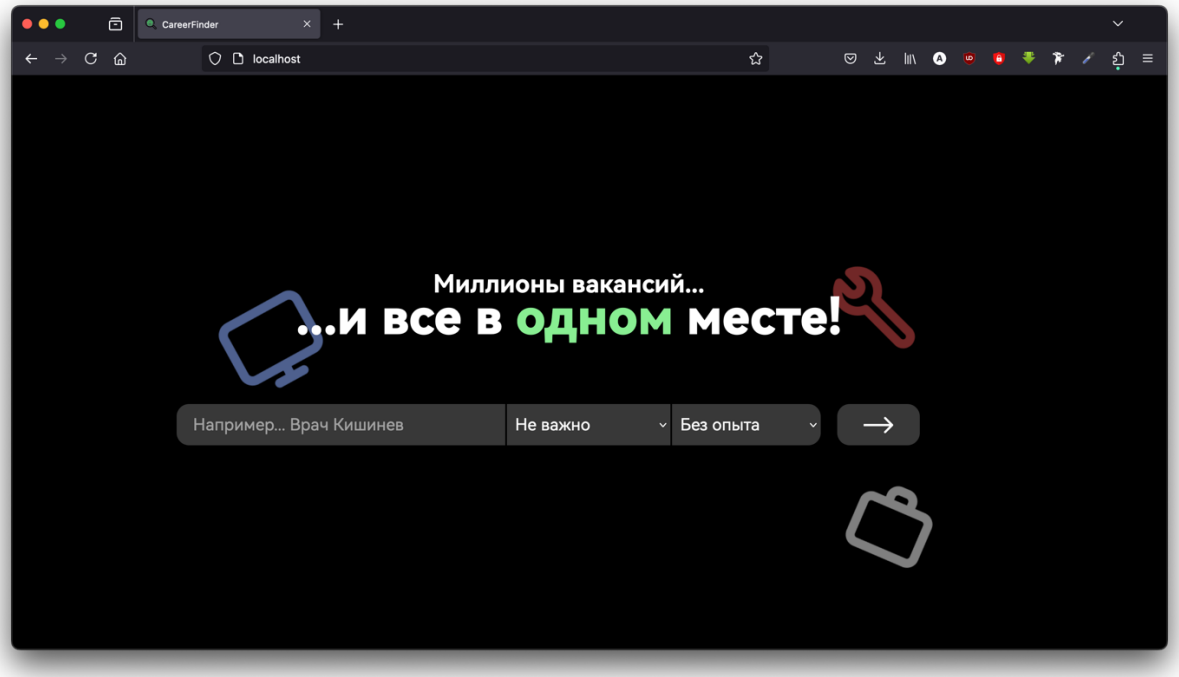
С кодом можно ознакомится на GitHub

3. CareerFrontend

Пользовательский интерфейс реализован через сайт, который написан без использования фреймворков, чистый HTML и CSS для стилей.

Сайт запускается посредством Flask, в котором были прописаны такие роуты как: ‘/’, ‘/search’, ‘/stats’

Роут ‘/’, корневой роут, который возвращает пользователю главную страницу



На главной странице есть поле поиска, в котором можно указать название вакансии или более подробную информацию, например Повар Астана 200000, тогда будет осуществлен поиск по вакансиям Повара в Астане с зарплатой 200000 тенге

В строке поиска можно указать график работы и наличие опыта

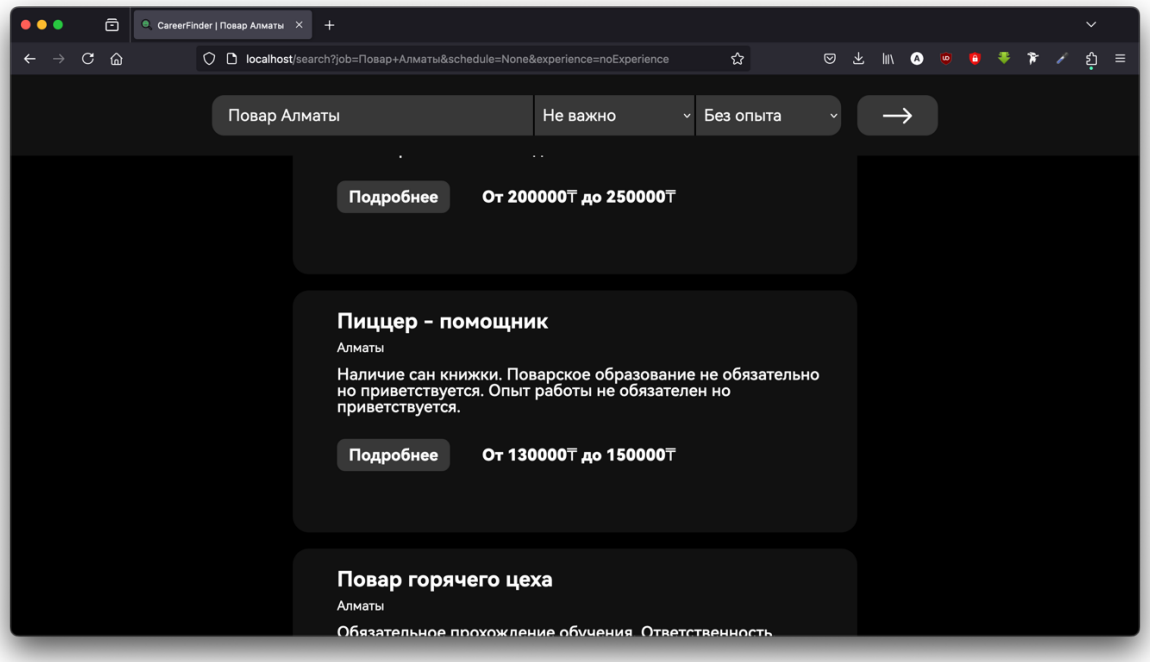
Выпадающие списки реализованы через теги html <select> и <option> для параметров

Кнопка поиска делает GET запрос к backend сервису с query параметрами названием работы, графиком и занятостью и отправляет пользователя на страницу /search

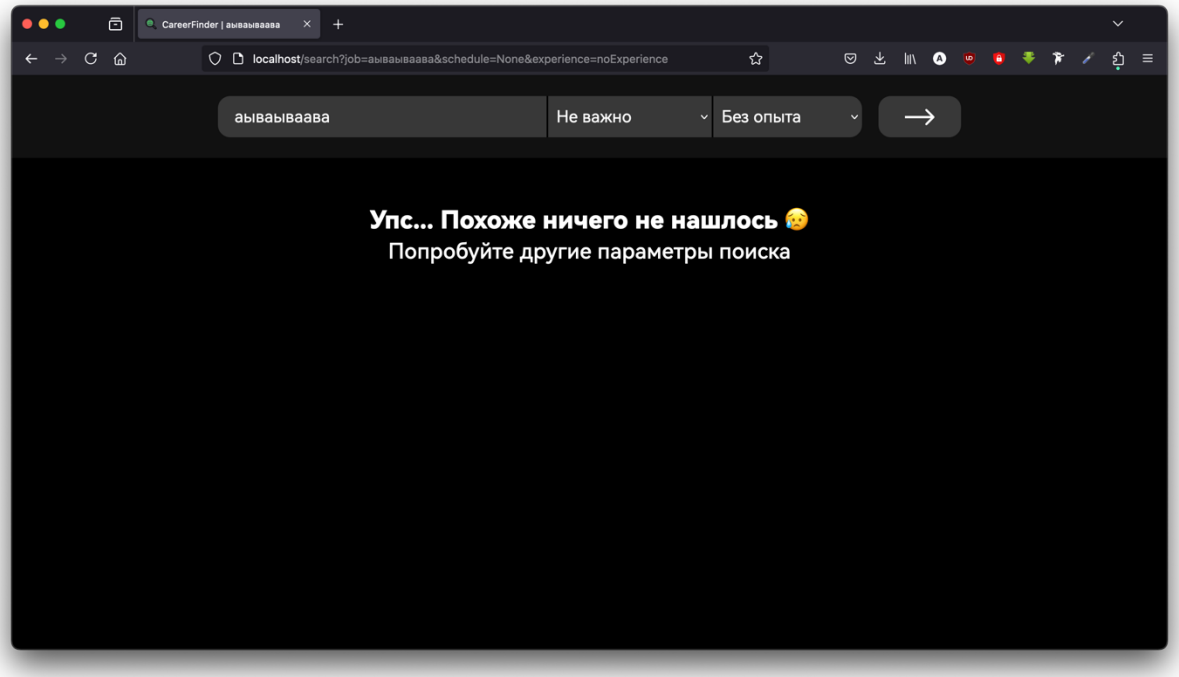
Фронт получает данные от сервера в виде списка, поэтому для отображения вакансий используется jinja2, а точнее конструкция {{ for }}

```
{% for vacancy in vacancies %}

<div class="vacCard">
    <div class="vacInfo">
        <div class="vacDesc">
            <h1 class="vacTitle">{{ vacancy['name'] }}</h1>
            <h1 class="vacArea">{{ vacancy['area'] }}</h1>
            <h1 class="vacDescription">{{ vacancy['requirement'] }}</h1>
            <div class="lowerInfo">
                <div class="openVacBtn">
                    <a href="{{ vacancy['url'] }}" class="linkTo">
                        Подробнее
                    </a>
                </div>
                <h1 class="vacSalary">{{ vacancy['salary'] }}</h1>
            </div>
        </div>
    </div>
</div>

{% endfor %}
```

Список вакансий передается в переменную vacancies, а после для каждой вакансии отображается карточка по заданному шаблону



Если вдруг вакансий не нашлось, пользователь получит сообщение об ошибке



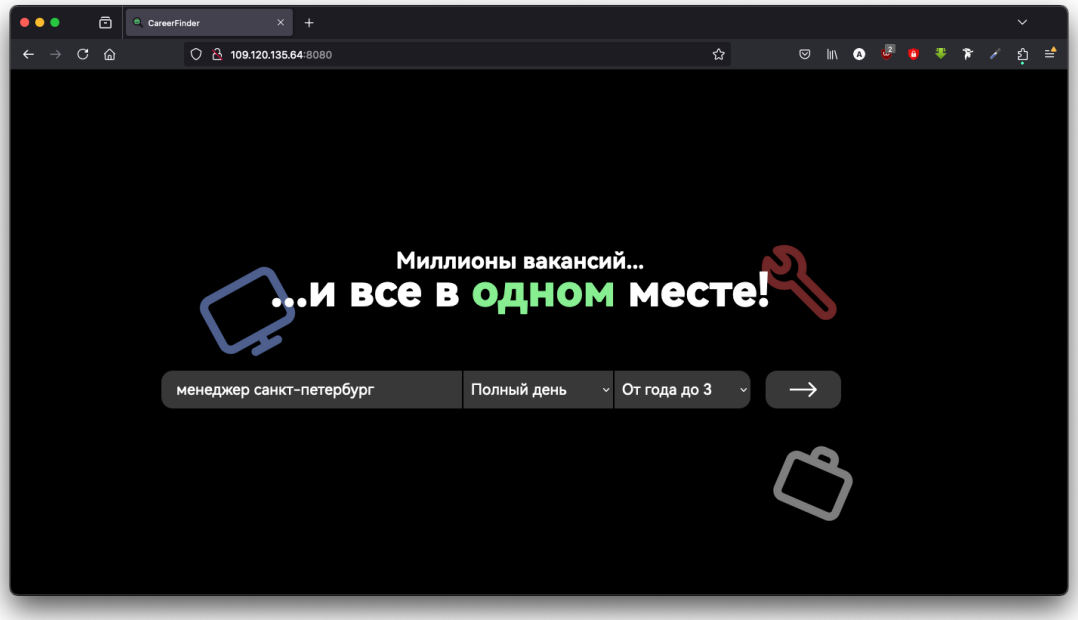
Роут ‘/stats’ возвращает страницу со статистикой где отображены данные по самому популярному запросу и всем остальным

Код так же есть на GitHub

4. Тестирование

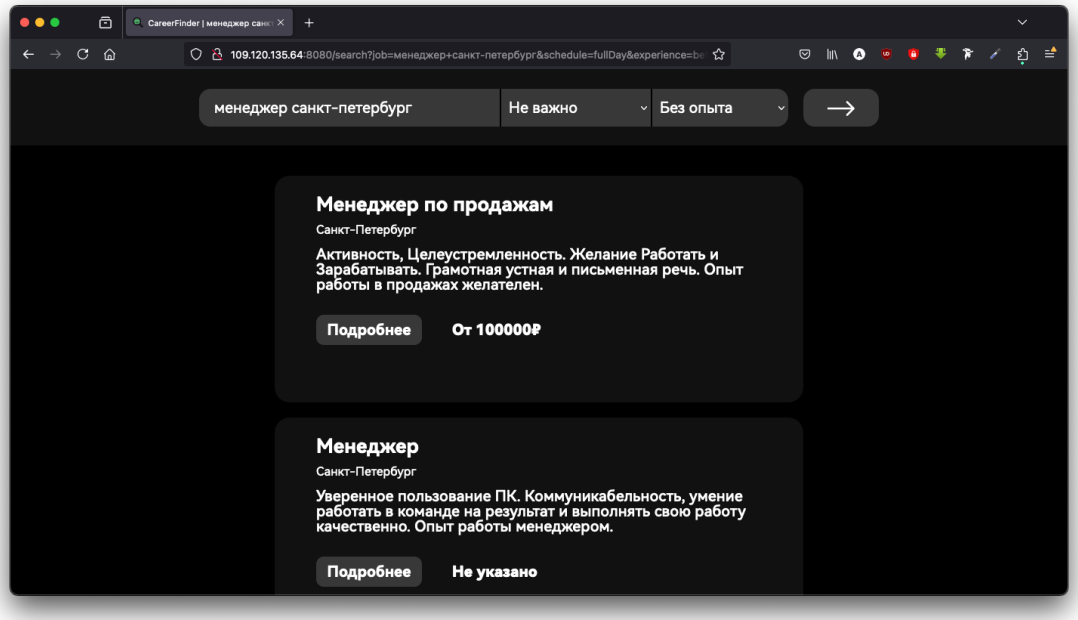
Тестирование приложения было проведено посредством поиска различных вакансий

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



5. Docker

Так как тестирование проведено успешно и приложение работает как должно, я решил упаковать все сервисы в Docker контейнеры.

В папке сервисов Backend и Frontend были созданы Docker файлы и созданы файлы requirements.txt для установки зависимостей проектов

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

В корне был создан файл docker-compose.yml

Были указаны сервисы backend, frontend, mysql, phpMyAdmin

Последний был необязателен, но использовался для удобства при отладке

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

6. Развертывание на сервере

Так как теперь имеется возможность развернуть приложение практически где угодно, я решил развернуть его на своем сервере с ОС ubuntu-server

После подключения к серверу, необходимо установить git, docker и docker-compose если они не установлены 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/Pe4en1eInMTUCI/CareerFinder.git

Переходим в скаченный репозиторий командой cd CareerFinder

Запускаем проект посредством

docker-compose up --build

Переходим в браузере по IP сервера, видим что открывается главная станица приложения. Все работает

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. Получение теоретических знаний о работе HTTP протоколов, методах запросов.
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://flask.palletsprojects.com/en/3.0.x/
4. https://dev.mysql.com/doc/