Предметная область - Отслеживание вакансий по запросу на платформе HeadHunter.

­­Программа будет скачивать данные о вакансиях, размещенных на платформе HeadHunter.

**Критерии поискового запроса**

Поиск может осуществляться по критериям:

* Название профессии
* Уровень дохода
* Регион / город места работы
* Тип занятости

**Технические требования**

Запрос выполняется не более 5 минут.

Просмотр первых 10 страниц, отсортированные по дате публикации (сначала новые).

**Ограничения проектирования**

При реализации системы скачивания данных будет использоваться python версии не ниже 3.10.

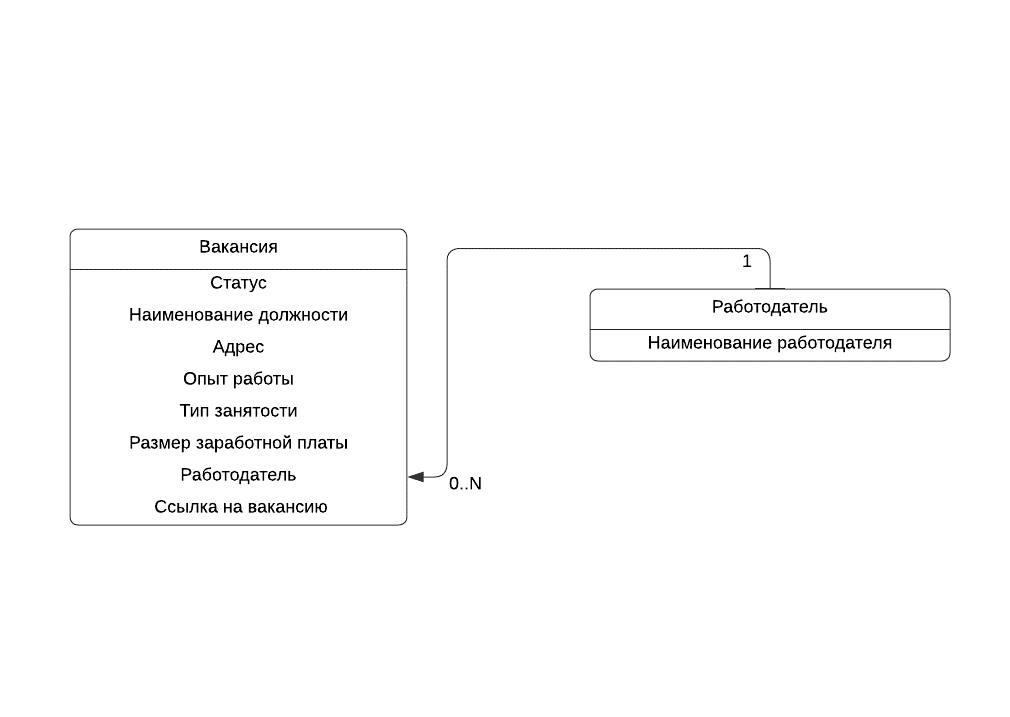
В процессе работы будут необходимы следующие библиотеки:

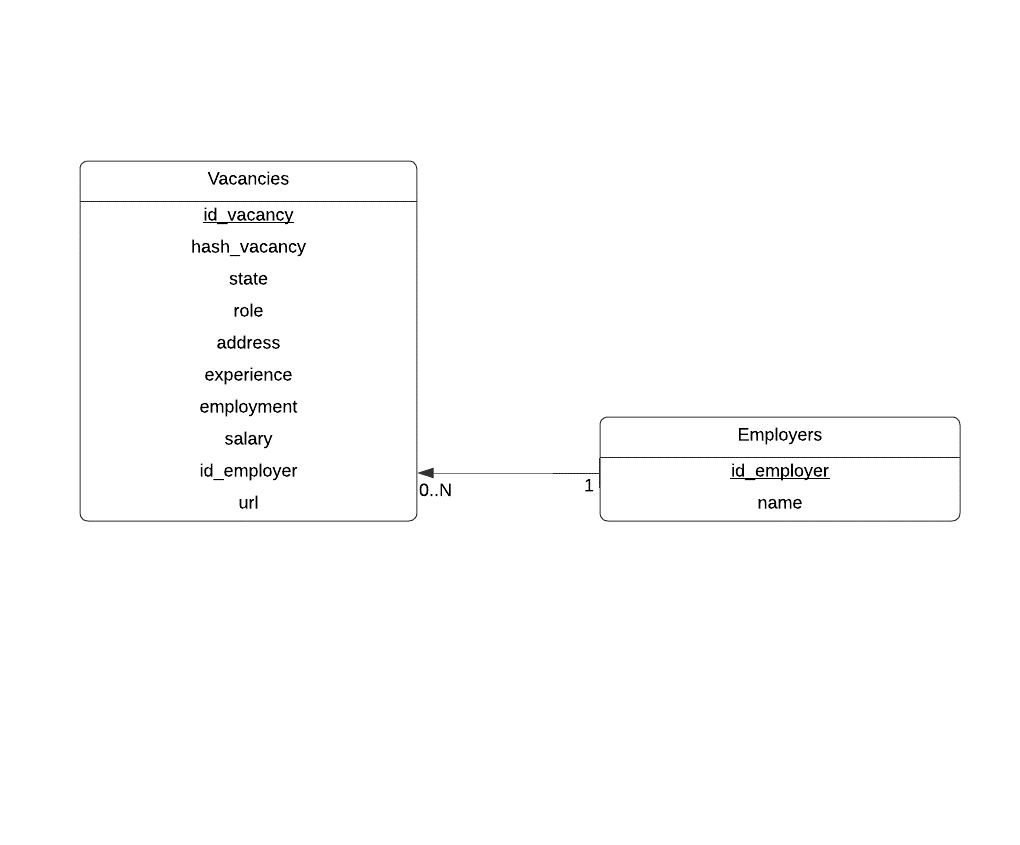
1. requests
2. beautifulSoup
3. flask
4. flask-restful
5. psycopg2
6. hashlib
7. json
8. time

Для хранения собранных данных используется база данных postgeSQL.

Будут храниться:

* Наименование вакансии
* Статус заявки (открыта или нет)
* Размер заработной платы
* Адрес работы
* Требуемый опыт работы
* График работы
* Наименование работодателя

ER-модель

 Логическая модель

Описание таблиц:

Vacancies

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Допустимые значения | Прочее |
| id\_vacancy | serial | целое число, большее 1 | Порядковый номер вакансии |
| hash\_vacancy | integer | 1 – 99999999 | Высчитывается как hash-функция от строки, состоящей из всех атрибутов строки (т.е. hash(state + role + …).  Полученное значение берется по модулю 99999999 |
| state | varchar | "open", “close” | Видимость вакансии:  “open” – вакансия открыта, “close” – вакансия закрыта |
| role | varchar | Не менее 1 символа | Наименование вакансии |
| address | json | Json, элементы которого это:  "city": varchar or null,  "building\_address": varchar or null,  "metro\_stations”: varchar[] or null | "city" – город, "street" – улица, "building\_address" – адрес здания, "metro\_stations” – ближайшие станции метро (если метро есть, иначе null) |
| experience | varchar | “Не имеет значения”, “Нет опыта”, “От 1 года до 3 лет”, “От 3 до 6 лет”, “Более 6 лет” |  |
| employment | varchar | “Полная занятость”, “Частичная занятость”, “Проектная работа”, “Стажировка”, “Волонтерство” |  |
| salary | json | В json элементы с ключами:  “from”: varchar or null, “to”: varchar or null, “currency”: varchar or null | “from” – нижняя граница зарплаты, “to” – верхняя граница зарплаты, “currency” – код валюты (например, “RUB”) |
| id\_employer | integer | 1 – 99999999 | Вычисляется как hash-функция от названия организации.  Полученное значение берется по модулю 99999999 |
| url | varchar | Текст ссылки | Ссылка на страницу вакансии |

Employers

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Допустимые значения | Прочее |
| Id\_employer | serial | целое число, большее 1 | Порядковый номер названия работодателя |
| name | varchar or null | Не менее 1 символа |  |

Краулер последовательно читает данные о вакансиях, расположенных на первых 10 страницах, отсортированных по дате публикации (сначала новые). Если количество страниц с результатами меньше 10, то просматриваются все.

Если при получении информации о конкретной вакансии данные какого-либо атрибута некорректны (не соответствуют их описанию в таблицах выше), то вакансия будет пропущена.

Данные об одной и той же вакансии могут быть получены разными запросами, но хранятся один раз. Это условие обеспечивается проверкой полей с хэшами.

Если краулер работает дольше 5 минут, то происходит сохранение последней обрабатываемой вакансии, а затем работа завершается.

Во избежание обработки не валидных ссылок предсмотрена обработка исключения, выбрасываемого модулем request.

Во избежания возможного конфликта в результате одновременной обработки данных двумя и более краулерами, при создании базы данных вводится ограничение на количсаетво подключений: CREATE DATABASE practice5 OWNER postgres CONNECTION LIMIT 1;

Функция: получение списка вакансий

endpoint: GET /api/vacancy/employment={}& area={}&text={}&salary={}&open={}

Поиск будет осуществляться по критериям:

* text - название профессии
* salary – нижняя граница желаемого уровень дохода
* area - регион (вводится наименование города)
* employment - тип занятости (не обязательно; варианты: полная занятость – “full”, частичная занятость – “part”, Проектная работа – “project”, Стажировка – “probation”, Волонтерство – “volunteer”)
* open -открыта ли вакансия (Видимость вакансии:“true” – вакансия открыта, “false” – вакансия закрыта)

Пример: GET /api/vacancy/employment=full&area=Россия&text=стоматолог&salary=155000&open=true

Формат ответа:

{

“id”: 1,

“state”: “open/close”,

“role”: “название\_должности”,

"address": {

"city": "название\_города",

"building\_address": "Вернадского 86",

“metro\_stations”: {“station1”, “station2”, …}

},

"experience": "Например: от 1 года до 3 лет",

"employment": "Например: стажировка",

"salary": {

"from": "10000",

"to”: null,

"currency": “USD”

},

"employer": {

"id": 1455,

"name": "название\_работодателя"

}

}

CREATE TABLE employers (

id\_employer serial PRIMARY KEY,

name varchar(255)

);

CREATE TABLE vacancies(

id\_vacancy serial PRIMARY KEY,

hash\_vacancy integer,

state varchar(255),

role varchar(255),

address json,

experience varchar(255),

employment varchar(255),

salary json,

id\_employer integer,

url varchar(255)

);