Дипломный проект VKinder

консультация

Владимир Хомутов

Владимир Хомутов

python разработчик РТК ИТ

План занятия

1. Разбор условий задания
2. Архитектура проекта
3. Права для доступа к VK API
4. Инструменты
5. WEB-API
6. Модули
7. Логика frontend
8. Логика backend
9. Логика БД
10. Типовые ошибки
11. Разбор вопросов

Условия

Все слышали про известное приложение для знакомств - Tinder. Приложение предоставляет простой интерфейс для выбора понравившегося человека. Сейчас в Google Play более 100 миллионов установок.

Используя данные из VK, нужно сделать сервис намного лучше, чем Tinder, а именно: чат-бота "VKinder". Бот должен искать людей, подходящих под условия, на основании информации о пользователе из VK:

* Возраст, * пол, * город, * семейное положение.

У тех людей, которые подошли по требованиям пользователю, получать топ-3 популярных фотографии профиля и отправлять их пользователю в чат вместе со ссылкой на найденного человека. Популярность определяется по количеству лайков и комментариев.

Входные данные

Имя пользователя или его id в ВК, для которого мы ищем пару.

если информации недостаточно нужно дополнительно спросить её у пользователя.

Условия

Разработать (однопользовательский) ВК-бот для поиска людей:

- Бот должен получать информацию из профиля пользователя(Возраст, пол, город, семейное положение, если информации недостаточно нужно дополнительно спросить её у пользователя). Входные данные: Имя пользователя или его id в ВК, для которого мы ищем пару

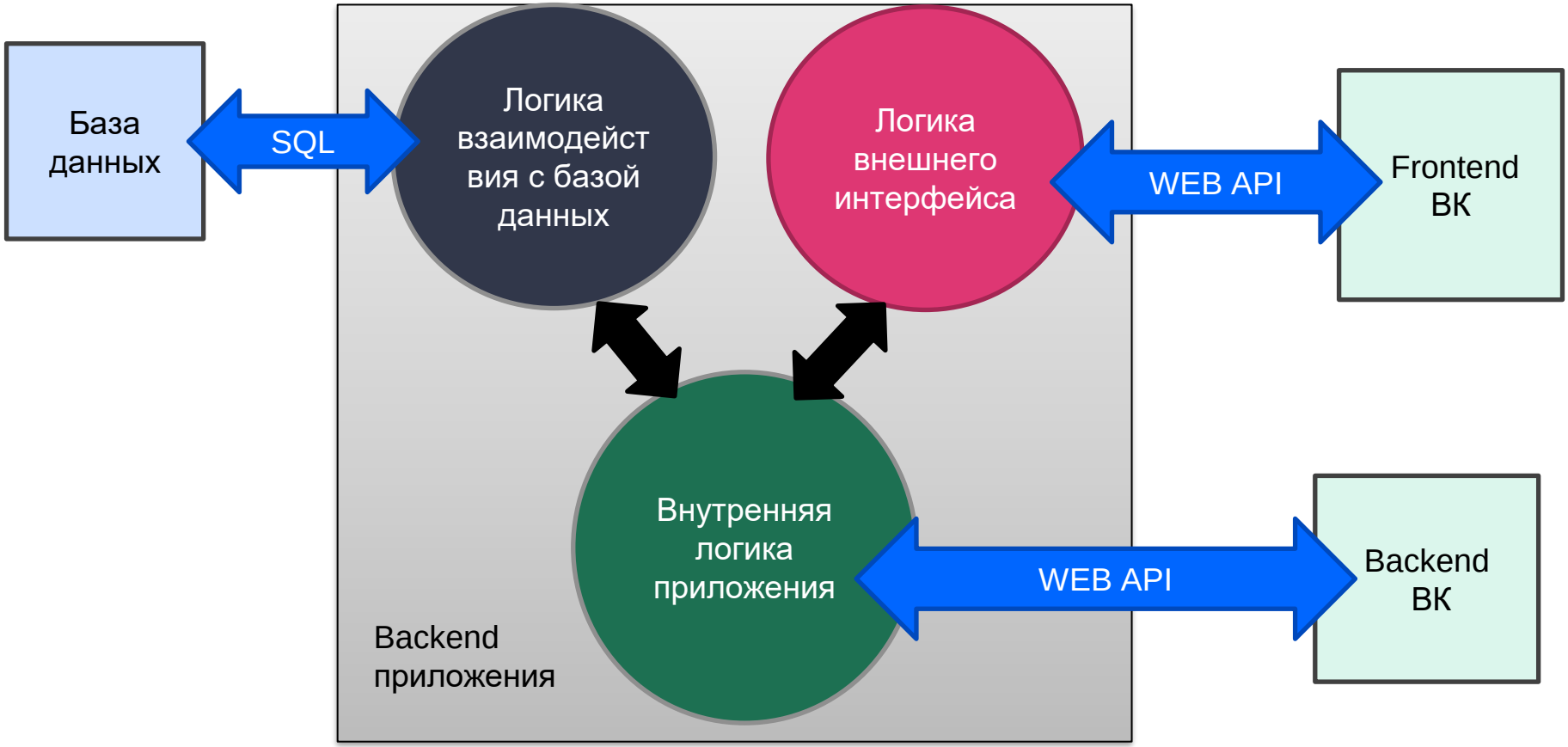
- Бот должен выполнять поиск анкет на основе полученных данных

- Бот должен выдавать пользователю по очереди найденные анкеты (ссылка на профиль, топ-3 популярных фотографии, Популярность определяется по количеству лайков и комментариев.)

Требования к сервису:

1. **Код программы удовлетворяет PEP8.**
2. Получать токен от пользователя с нужными правами.
3. **Программа декомпозирована на функции/классы/модули/пакеты.**
4. Результат программы записывать в БД.
5. **Люди не должны повторяться при повторном поиске.**
6. Не запрещается использовать внешние библиотеки для vk.

Архитектура проекта



Права доступа

1. Токен для frontend (токен сообщества)
2. Токен для backend (токен пользовательского приложения)

Инструменты

1. Visual studio code (https://code.visualstudio.com/)
2. Postgress SQL (https://www.postgresql.org/)
3. PG Admin 4 (https://www.pgadmin.org/)
4. web-браузер (https://www.google.com/intl/ru_ru/chrome/)
5. Аккаунт ВК (https://vk.com/)

WEB-API

https://dev.vk.com/

Отправка сообщений – messages.send
Получение сообщений - messages.sendMessageEventAnswer

Получение информации о пользователе – users.get
Поиск анкет пользователей – users.search
Получение фотографий photos.get

Модули

sqlalchemy + psycopg2
vk-api

Логика для Frontend

```
# импорты
import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType
from vk_api.utils import get_random_id
```

```
# отправка сообщений
vk = vk_api.VkApi(token=comunity_token)
def message_send(user_id, message, attachment=None):
    vk.method('messages.send',
              {'user_id': user_id,
               'message': message,
               'attachment': attachment,
               'random_id': get_random_id()})
```

Логика для Frontend

```
# обработка событий / получение сообщений

vk = vk_api.VkApi(token=comunity_token)
longpoll = VkLongPoll(vk)

# эхо
for event in longpoll.listen():
        if event.type == VkEventType.MESSAGE_NEW and event.to_me:
            message_send(event.user_id, event.text.lower())
```

Логика для Backend

```
# импорты
import vk_api
```

```
# получение данных о пользователе
vkapi = vk_api.VkApi(token=acces_token)

def get_profile_info(user_id):

    info, = vkapi.method('users.get',
                         {'user_id': user_id,
                          'fields': 'city'
                         }
                         )
    return info
```

Логика для БД

```
# импорты
import sqlalchemy as sq
from sqlalchemy.orm import declarative_base
from sqlalchemy import create_engine, MetaData
from sqlalchemy.orm import Session
```

```
# схема БД
metadata = MetaData()
Base = declarative_base()

class Viewed(Base):
    __tablename__ = 'viewed'
    profile_id = sq.Column(sq.Integer, primary_key=True)
    worksheet_id = sq.Column(sq.Integer, primary_key=True)
```

Логика для БД

```
# добавление записи в бд

engine = create_engine(db_url_object)
Base.metadata.create_all(engine)
with Session(engine) as session:
    to_bd = Viewed(profile_id=1, worksheet_id=1)
    session.add(to_bd)
    session.commit()
```

```
# извлечение записей из БД

engine = create_engine(db_url_object)
with Session(engine) as session:
    from_bd = session.query(Viewed).filter(Viewed.profile_id==1).all()
    for item in from_bd:
        print(item.worksheet_id)
```

Типовые ошибки

1. Найти готовую работу на github и выдать за свою
2. Избыточная схема БД
3. Попытка реализации процедуры аутентификации
4. Отсутствие проверок данных полученных от API
5. Получать данные для поиска только из чата
6. Сдавать работу в виде архива
7. Сдавать нерабочий проект
8. Пренебрегать использованием готовых модулей
9. Нерациональная нагрузка на сервера API
10. Нерациональная нагрузка на БД
11. Нерациональное использование ресурсов

Вопросы

Владимир Хомутов

python разработчик РТК ИТ

Условия

Дополнительные требования (не обязательны для получения диплома):

1. В vk максимальная выдача при поиске 1000 человек. Подумать как это ограничение можно обойти.
2. Добавить возможность ставить/убирать лайк, выбранной фотографии.
3. Можно усложнить поиск добавив поиск по интересам. Разбор похожих интересов(группы, книги, музыка, интересы) нужно будет провести с помощью анализа текста.
4. У каждого критерия поиска должны быть свои веса. То есть совпадение по возрасту должны быть важнее общих групп. Интересы по музыке важнее книг. Наличие общих друзей важнее возраста. И так далее.
5. Добавлять человека в избранный список, используя БД.
6. Добавлять человека в черный список чтобы он больше не попадался при поиске, используя БД.
7. К списку фотографий из аватарок добавлять список фотографий, где отмечен пользователь.