Основы Grid и Cloud вычислений

Практика

Петров Михаил 433 группа
Романычев Леонид 433 группа

Задача

Автоматический агрегатор данных с Инстаграмм (используя Scrapy)

видео файлы + тексты из этого поста

с архитектурой сильно проще но похожей на https://github.com/HronoSF/DSS_2020/tree/master/vk-crawler

+ Поиском по скаченным видео файлам в интерфейсе пользователя (пример тега #underwater имени канала underwaterstuffs)

Стек технологий

1. Scrapy (парсинг)
2. Scrapyrt (парсинг в реальном времени)
3. Flask (бекенд)
4. React js (фронтенд)
5. MongoDB (база данных)

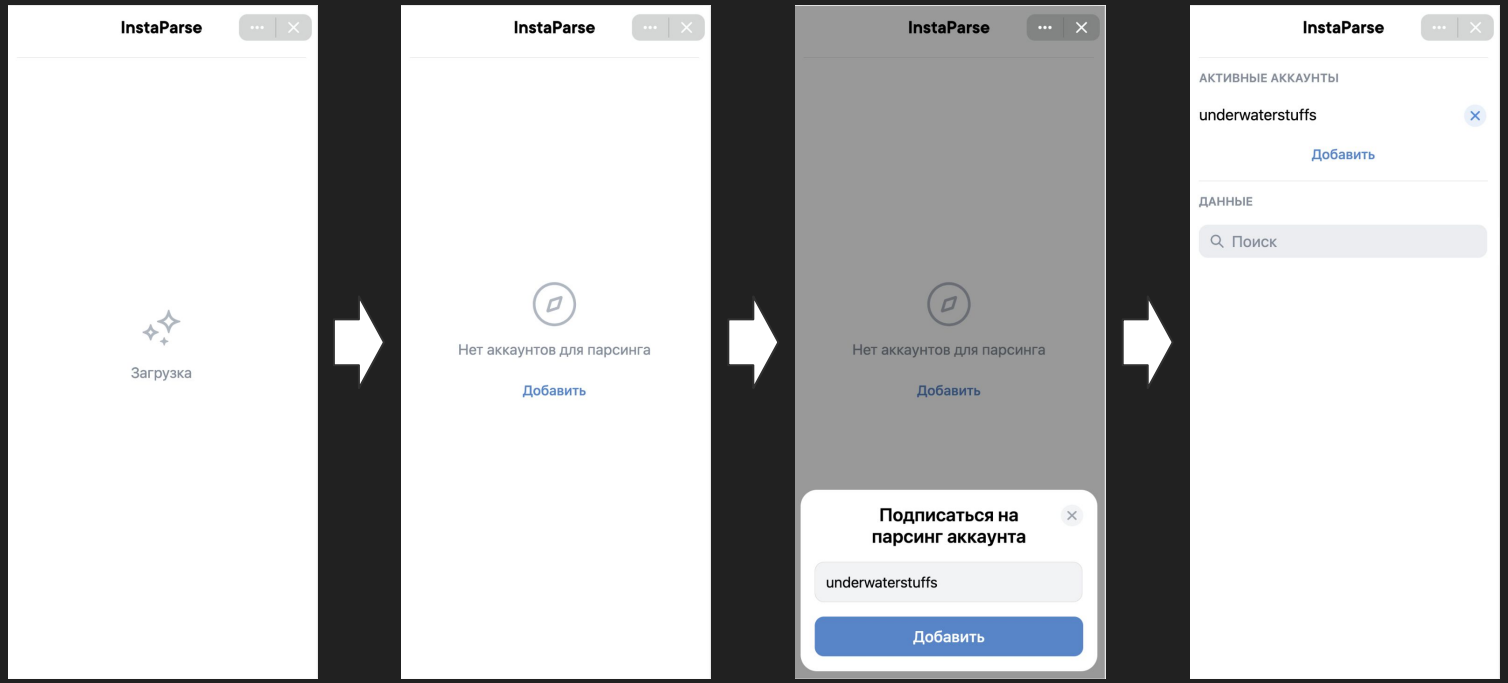
Также использовано:

- платформа VK MINI APPS (статика)
- сервер Digital Ocean (процессы)
- ScraperApi (Proxy API for Web Scraping)

Проблемы и решения

1. Instagram довольно защищенная от парсинга система
    a. Использовали прокси для парсинга - ScraperApi
2. Instagram использует обфускацию css классов
    a. Использовали данные через window._sharedData
3. Нет связки Scrapy и Flask
    a. Использовали Scrapyrt
4. Scrapyrt не позволяет передачу параметров через HTTP API URL
    a. Пришлось кастомизировать библиотеку самим используя issue (https://github.com/scrapinghub/scrapyrt/issues/29)

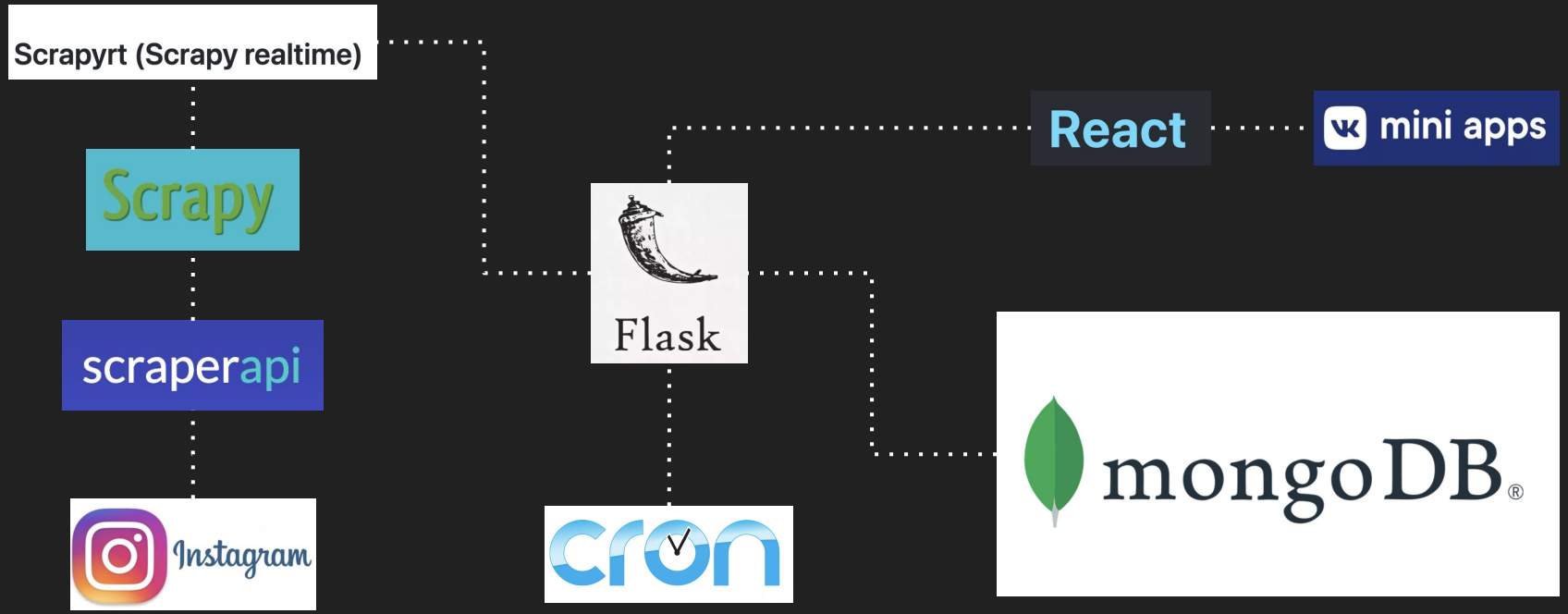
Путь пользователя



Backside

1. Flask
    - /auth - авторизация в приложении
    - /parse/add - добавление аккаунта для парсинга
    - /parse/delete - удаление аккаунта для парсинга
    - /search - поиск по всем данным по аккаунтам (@underwaterstuffs) и хештегам (#top)
    - /scrapy - роут cron задачи для взаимодействия с Scrapyrt
2. Scrapy
    - spider: instagram - парсер видео со всех страниц указанного аккаунта
3. Scrapyrt
    - предоставление доступа к Scrapy в реальном времени
4. MongoDB
    - хранение данных пользователя и спаршенных данных из инстаграмма
5. ScraperApi
    - предоставляет бесплатный тариф для прокси (scraperapi.com)
6. Cron task
    - каждые 10 минут вызывает /scrapy для парсинга данных

Архитектура



Исходный код

https://github.com/mike-petrov/scrapy_instagram