# 2025-09-08

## Статические и динамические веб-данные

Статические веб-данные обычно* не меняются со временем и не зависят от клиента. Например, видео на видеохостинге: загруженное единожды видео не меняет содержимое видеопотока «на лету», оно обычно загружается 1 раз в определенный момент времени и остается неизменным вне зависимости от того, сколько раз или с каких устройств вы с ним взаимодействуете.

*но видео может быть удалено, некоторые хостинги (например, YouTube) позволяет части своих клиентов менять видео без его перезагрузки на платформу, в видео могут быть ошибки, которые исправлены не в видеопотоке (а, например, в посте в социальных сетях) и т.п.

Но есть и неплохие примеры — скажем, библиотека Мошкова; загруженный файл, например, «Преступления и наказания» не меняется многие годы (Lib.ru/Классика: Достоевский Федор Михайлович. Преступление и наказание).

Динамические веб-данные либо меняются в зависимости от клиента («черные списки» или пейволл — доступ к полной версии только после оплаты, могут быть и другие виды ограничений — например, региональные, по подсетям и т.п.), либо без предупреждения регулярно меняются со временем. Например, описание под видео с видеохостинга: после загрузки оно может быть изменено без уведомлений или даже следов изменения сколько угодно раз; можно менять даже название видео и обложку.

Разумеется, динамических данных сейчас гораздо больше. Поэтому важно при сборе и анализе данных учитывать временной атрибут: когда данные были получены. Отсюда же идут понятия версионности данных (сама история изменений), множества состояний (возможность получить любую из предыдущих версий в полном объеме) и среза данных (единовременный сбор и фиксация текущего состояния веб-ресурса и размещенных на нем данных без сравнения с предыдущими версиями напрямую). Без учета фактора времени корректный анализ динамических веб-данных чаще всего невозможен.

Множество срезов данных позволяет со временем сформировать свои наборы данных с возможностью уже сравнения и версионирования. Но сама процедура среза (или снятия дампа, снятия слепка) это не подразумевает, это просто следствие накопления данных у сборщика.

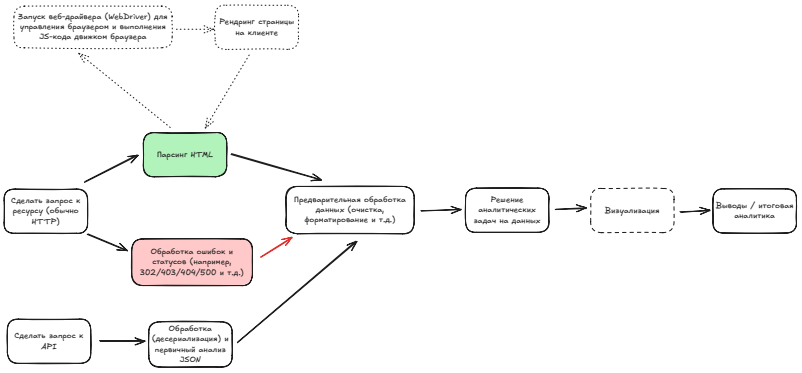
Посмотреть на то, как работает такой механизм, можно через Wayback Machine: https://archive.org/web/

> ⁝☰ **Example**

> **Как можно использовать Wayback Machine**
> При помощи хроники срезов пользователи оценивали кадровую ситуацию в публичной компании на основании того, как со временем менялась страница с персоналом (сотрудник + его должность).
> Анализ динамики веб-данных позволил отследить изменения в должностях, а также найти возможные даты начала и конца периода работы в компании.
> Через сопоставление с другими источниками (социальные сети) удалось также оценить тональность высказываний бывших сотрудников и в целом сделать ряд предположений об условиях труда в компании (позднее частично подтвердившихся).

**В настоящее время** в анализе динамических веб-данных появился дополнительный риск, связанный с некачественными генеративными (синтетическими) веб-данными: тексты, изображения, аудио и видео, полученные в результате работы генеративных нейронных сетей (т.н. AI Slop)

## Сбор, хранение и обработка веб-данных




## Практика 1

Веб-данные необходимо хранить с сохранением связей между ними. Для этого активно применяются документоориентированные и графовые базы данных (TypeDB). GraphQL — язык запросов к графовым базам данных.

Одно из крупнейших хранилищ, основанное на графовой структуре: https://commoncrawl.org/

- воспользоваться Common Crawl - Get Started; CommonCrawl with Python - Get All Pages from a Domain - JC Chouinard для освоения доступа к базе Common Crawl
- можно визуально поизучать Common Crawl - Overview
- можно посмотреть наработки CmonCrawl · PyPI и GitHub - michaelharms/comcrawl: A python utility for downloading Common Crawl data
- после этого собрать консольное приложение, которое осуществляет поиск по Common Crawl и выводит перечень связанные с запросом страниц

- поискать там упоминания г. Перми, Пермского Политеха, кафедры ИТАС; МГУ им. Ломоносова, МФТИ им. Баумана; Бориса Пастернака в контексте г. Перми
- представить результаты в виде текстового вывода

### Исходные данные и инструменты

- **Платформа:** Common Crawl — открытый архив веб-данных.
- **Источник данных:** Индекс CDX и WARC-файлы, размещенные на Amazon S3.
- **Язык программирования:** Python 3.x.
- **Ключевые библиотеки:** `cdx-toolkit`, `warcio`, `requests` / `httpx`, `beautifulsoup4`, `pandas`, `tqdm`, `argparse`.

1. Приложение должно принимать аргументы командной строки:
   - `keywords` (позиционные аргументы): одно или несколько ключевых слов для поиска.
   - `--domain` (опция): опциональный фильтр для ограничения поиска определенным доменом (например, `pstu.ru`).
   - `--limit` (опция): опциональное ограничение на количество возвращаемых результатов (по умолчанию 10).
   - `--show-text` (флаг): при указании этого флага приложение должно загрузить и отобразить фрагмент текста найденной страницы.
2. Программа должна корректно обрабатывать ошибки сети и отсутствие результатов.
3. Результаты должны быть представлены в виде удобочитаемой таблицы с колонками: URL, Дата архивации, Заголовок страницы (если доступен) и, опционально, Фрагмент текста.

Для поиска должен использоваться только индекс CDX. Загрузка тяжелых WARC-файлов должна производиться только при указании флага `--show-text` и только для требуемой части файла (с использованием заголовка `Range`).

Используя разработанное приложение, проведите поиск по следующим темам и проанализируйте результаты:

1. Найдите упоминания г. Перми и Пермского Политеха.
2. Найдите упоминания кафедры ИТАС ПНИПУ в контексте последних новостей.
3. Сравните количество и характер упоминаний МГУ им. Ломоносова и МФТИ.
4. Исследуйте, в каком контексте Борис Пастернак упоминается вместе с г. Пермью.

Реализовать приложение поэтапно:

- Настройка парсера аргументов командной строки.
- Реализация функции поиска по CDX-индексу.
- Реализация функции точечной загрузки и парсинга WARC-записи.

- Организация вывода результатов в виде таблицы.