# Анализ журнала логирования

## Что нужно сделать?

Нужно написать cli-приложение, которое будет анализировать логи django-приложения и формировать отчеты. Отчет выводится в консоль. Интерфейс приложения:

* можно передать пути к логам, файлов может быть несколько
* можно указать аргумент **--report** c названием отчета который нужно сформировать

##### Пример формирование отчёта:

```

python3 main.py logs/app1.log logs/app2.log logs/app3.log --report handlers  
```

##### Примеры логов можно скачать по [этой](https://drive.google.com/drive/folders/1zjkgC0iFkvikW1AtGSupT-0KIFdy7GuY?usp=sharing) ссылке.

## Как формируется отчет на основе нескольких файлов?

Каждый файл обрабатывается отдельно, а после обработки данные объединяются в один отчёт. Например, нужно посчитать общее количество запросов: в файле **app1.log** — 3 млн запросов, а в файле **app2.log** — 2 млн запросов, итого в отчете будет 5 млн запросов.

## Какие отчеты нужны?

Нужно реализовать только один отчет, но нужно заложить в архитектуру возможность работы с несколькими отчетами.

## Отчёт о состоянии ручек API по каждому уровню логирования:

* название отчета: **handlers**
* алгоритм формирования:
  + считаем количество запросов к ручкам, это записи django.requests:
    - по каждой ручке
    - по каждому уровню логирования
    - группируем запросы по ручкам
  + при выводе сортируем ручки в алфавитном порядке
  + последней строчкой выводим общее количество запросов

##### Пример формирование отчёта:

```

python3 main.py logs/app1.log logs/app2.log logs/app3.log --report handlers  
```

##### Пример вывода:

```

Total requests: 1000

HANDLER DEBUG INFO WARNING ERROR CRITICAL

/admin/dashboard/ 20 72 19 14 18

/api/v1/auth/login/ 23 78 14 15 18

/api/v1/orders/ 26 77 12 19 22

/api/v1/payments/ 26 69 14 18 15

/api/v1/products/ 23 70 11 18 18

/api/v1/shipping/ 60 128 26 32 25

178 494 96 116 116  
```

## Какие функциональные требования?

#### Основные:

* можно сформировать отчёт **handlers** на основе нескольких файлов
* пути к файлам передается как аргумент
* название отчета передается как аргумент

#### Дополнительные:

* приложение проверяет что указанные файлы существуют
* приложения проверяет что переданное имя отчета верное

## Какие не функциональные требования?

#### Основные:

* для всего кроме тестов, можно использовать только стандартную библиотеку, например, для создания интерфейса командной строки нельзя использовать typer, а для красивого вывода отчета в консоль pandas.
* файлы с логами могут быть размером в несколько гигабайт и более
* код покрыт тестами написанных на pytest
* код содержит аннотации типов
* код соответствует общепринятым стандартам стиля

#### Дополнительные:

* обработка нескольких файлов происходит параллельно, предполагаем, что приложение будет запускаться на машине с несколькими ядрами
* в архитектуру приложения заложена возможность удобного добавления новых отчётов. Удобное добавление означает, что для того чтобы добавить новый отчёт, не нужно переписывать уже существующий код или вносить в него много изменений, а нужно лишь дописать логику формирования нового отчёта

## Какие основные критерии того, что задание выполнено?

* приложение работает и удовлетворяет основным требованиям, которые описаны выше

## Как сдавать задание?

* присылайте ссылку на git репозиторий
* присылайте примеры запуска приложения, например, можно сделать скриншот формирования отчета, скриншот можно положить прямо в репозиторий
* перед отправкой ссылки на репозиторий проверьте, пожалуйста, что репозиторий публичный и его можно посмотреть

## FAQ

* Можно ли присылать задание, если не выполнены дополнительные требования?
  + Да, можно. Главное, чтобы были выполнены все основные требования. Дополнительные требований будут плюсом. Мы понимаем, например, что junior разработчик может не иметь опыта написания кода, который будет задействовать несколько процессов.
* Можно ли использовать нейросети?
  + Рекомендуем не использовать. Постоянно сталкиваемся со случаями, когда кандидаты увлекаются нейросетями, чтобы сделать тестовое, а потом заваливают техническое интервью, потому что не понимают, почему нейросеть написала тот или иной код.
* Будет ли приниматься задание без тестов?
  + Нет, приниматься не будет. Наличие тестов входит в основные требования.
* Код покрыт тестами - это какой процент покрытия?
  + Можно ориентироваться на 80% покрытия по [pytest-cov](https://pypi.org/project/pytest-cov/). Будет больше — отлично! Будет меньше, тоже ок, главное протестировать критическую функциональность, например, тестировать что отчёт красиво выводится по колонкам не нужно.
* Можно ли использовать какие-то дополнительные библиотеки к pytest?
  + Да, всё что помогает вам тестировать код можно использовать.
* Можно ли пользоваться линтерами или форматтерами кода?
  + Да, можно использовать любой линтер или форматтер.
* Нужно ли учитывать что уровни логирования могут поменяться?
  + Нет, не нужно. Берём 5 стандартных уровней: info, debug, warning, error и critical.
* Нужно ли писать комментарии в коде?
  + Если вы считаете что они нужны, то пишите.
* Нужно ли писать README.md?
  + Писать не обязательно, но в README.md можно описать как можно добавить новый отчёт, где какие методы или классы нужно для этого написать или поменять