Задание

Напишите бота, который принимает сообщения от пользователя. Если в нем содержится файл, то бот должен сохранить его в S3 хранилище.

Основные требования:

- при первом сообщении бот должен ответить сообщением, которое содержит строку **[greeting]**
- при втором и последующих сообщениях, если в сообщении нет файла, бот должен ответить строкой, которая содержит **[document is required]**
- бот должен реагировать только на файлы (в tg api поле называется document). А картинки, отправленные как картинки, бот не должен воспринимать, как файлы
- если боту отправили файл, он должен сразу отправить сообщение, которое содержит строку **[document].** Затем загрузить файл в S3 и отправить сообщение, которое содержит **[document has been saved]**
- файл нужно загрузить в **корень** bucket
- файл нужно назвать так, как назвал его Telegram, то есть загрузить его в S3 с тем названием, которое пришло из tg api в поле **file_name.**

ВАЖНО!

Если сообщения не будут содержать указанные названия, тесты будут падать.

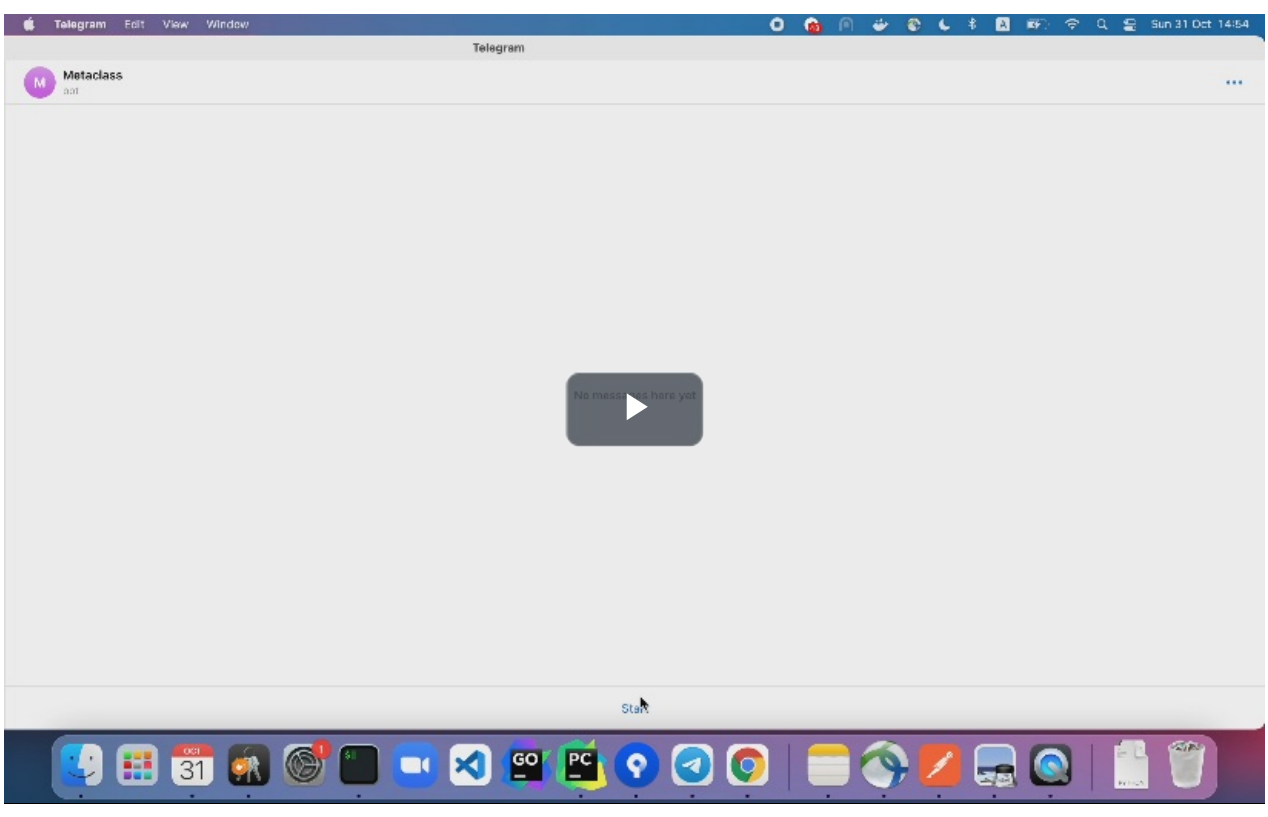
Корректные сообщения:

- **[greeting]**
- **[greeting] Приветствие**
- **[greeting] Любой текст, который вы пожелали дописать**

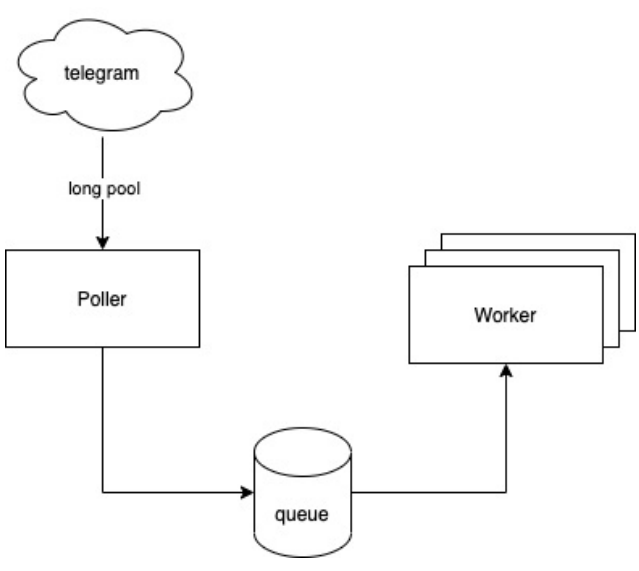
Некорректные сообщения:

- **greeting**
- **Приветствие**

Как должен работать бот:



Архитектура



Бот состоит из трех основных частей:

1. poller (поллер)
2. queue (очередь)
3. worker (воркеры)

Poller. Внутри должна работать фоновая корутина, которая получает данные из Телеграма методом **getUpdates.** Когда новые данные появились, poller должен положить их в очередь.

Важно: параметр timeout в этом методе нужно указать 60.

Требования:

- **poller должен быть один.** Если параллельно кто-то будет вызывать getUpdates, новые события от бота могут быть потеряны
- **poller не должен выполнять бизнес-логику.** Он должен только получить событие из Телеграма, положить объект в очередь и забыть.

Worker. Должен запустить внутри *несколько корутин,* которые будут получать сообщения из queue и выполнять бизнес-логику. Количество корутин определяется в параметрах воркера (класс WorkerConfig поле concurrent_workers).

Такая архитектура позволяет масштабировать производительность, потому что мы можем гибко добавить количество воркеров в случае большой нагрузки.

Подсказка. Запущенные корутины лучше обернуть в **bot.utils:log_exceptions.** Если у вас будут исключения в фоновых корутинах, то вы о них не узнаете. Просто будет ощущение, что ничего не работает.

Запуск и тестирование

Работа для выполнения задания будет проводиться в папке bot. Нужные файлы:

```
base.py
poller.py
utils.py
worker.py
```

Для запуска бота нужно запустить файл:

```
python3 run_bot.py
```

Настоятельно рекомендую сначала реализовать бота и протестировать его руками, а потом запускать автотесты. Для тестирования бота нужно запустить:

```
pytest tests/bot -vv -s
```

Рекомендуемый ход выполнения

1. Реализовать poller. Результатом этапа будет заполненная очередь объектами UpdateObj
2. Реализовать worker без бизнес-логики. То есть он должен вычитывать сообщение из очереди и на новые события отправлять, например, hello обратно в чат
3. Реализовывать бизнес-логику

Дополнительные требования

Являются необязательными, автотестами не покрываются.

- При запуске бота нужно проверить корректность Telegram-токена (метод getMe). Если токен некорректный, завершить запуск бота и вывести ошибку
- При запуске бота проверять bucket, в которых нужно загружать файлы из Telegram. Если такого bucket не существует, завершить запуск бота и вывести ошибку
- После отправки пользователем файла нужно спросить пользователя, желает ли он переименовать файл. Если да, спросить новое имя файла, а после загрузить файл с новым именем в s3
- Добавить команду для получения списка загруженных файлов
- При загрузке файла в хранилище S3 отправлять прогресс загрузки сообщений пользователю в Telegram
- Сделать поддержку бота в VK. Описание bot api — https://vk.com/dev/bots_docs