Задание на тему:

«Сервис обработки курса валют»

Дата постановки: **30.12.2020**

Дата выполнения: **08.01.2021**

Работу выполнил

Петрухин Алексей Дмитриевич

+7 (985) 073-53-25

[alexnord95@gmail.com](mailto:alexnord95@gmail.com)

GitHub: <https://github.com/AlexN0rth/AlexGif>

Содержание

[1. Постановка задания 3](#_Toc61026790)

[2. Проработка задания 4](#_Toc61026791)

[3. Архитектура сервиса 6](#_Toc61026792)

[3.1 Диаграмма архитектуры 6](#_Toc61026793)

[3.2 Описание потоков сервиса 7](#_Toc61026794)

[4. JavaDoc 8](#_Toc61026795)

[5. Инструкция по запуску 9](#_Toc61026796)

[5.1 Запуск сервиса с использованием JVM Intellij IDEA 11](#_Toc61026797)

[5.2 Запуск сервиса с использованием Docker 15](#_Toc61026798)

[6. Контактная информация 22](#_Toc61026799)

# 1. Постановка задания

Создать сервис, который обращается к сервису курсов валют, и отдает gif в ответ:

если курс по отношению к рублю за сегодня стал выше вчерашнего, то отдаем рандомную отсюда https://giphy.com/search/rich

если ниже - отсюда https://giphy.com/search/broke

Ссылки

REST API курсов валют - https://docs.openexchangerates.org/

REST API гифок - https://developers.giphy.com/docs/api#quick-start-guide

**Must Have**

Сервис на Spring Boot 2 + Java / Kotlin

Запросы приходят на HTTP endpoint, туда передается код валюты

Для взаимодействия с внешними сервисами используется Feign

Все параметры (валюта по отношению к которой смотрится курс, адреса внешних сервисов и т.д.) вынесены в настройки

На сервис написаны тесты (для мока внешних сервисов можно использовать @mockbean или WireMock)

Для сборки должен использоваться Gradle

Результатом выполнения должен быть репо на GitHub с инструкцией по запуску

**Nice to Have**

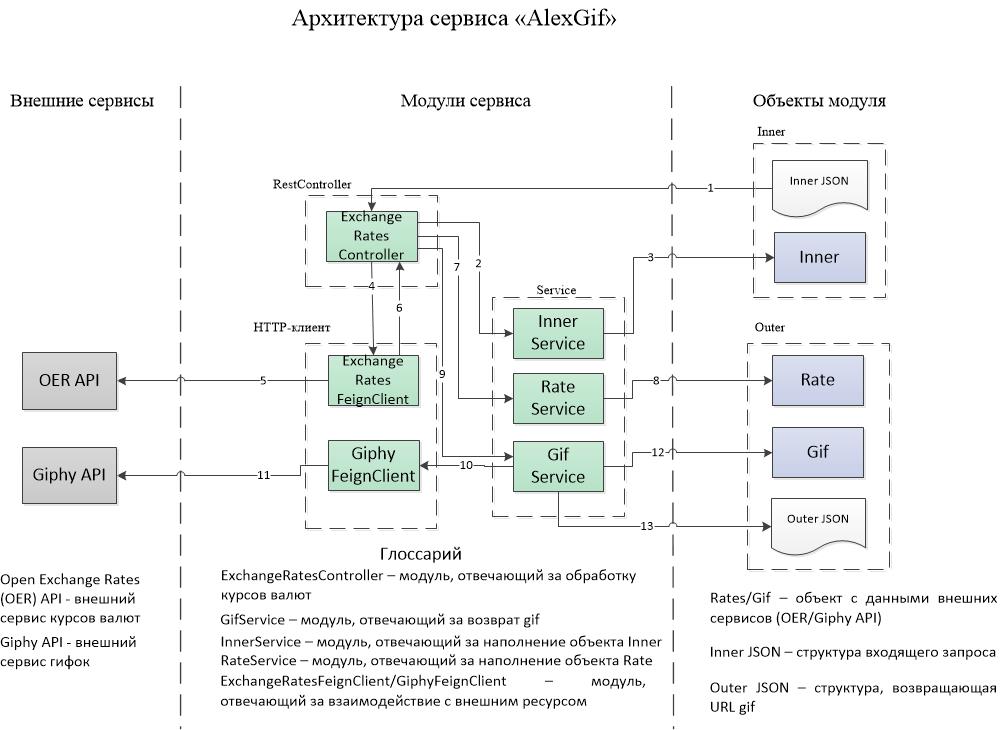
Сборка и запуск Docker контейнера с этим сервисом

# 2. Проработка задания

1. Применяемый стек технологий:
   1. Java/Kotlin;
   2. Spring Boot 2;
   3. Spring Feign;
   4. Spring Boot Test;
   5. Gradle;
   6. GitHub;
   7. Docker.
2. Инструментарий для реализации:
   1. ОС Windows Home/Pro;
   2. IntelliJ IDEA 2020.1 Ultimate Edition (среда разработки);
   3. Java 8;
   4. JavaDoc (подготовка документации);
   5. MS Visio 2010 (подготовка архитектуры);
   6. MS Word (подготовка инструкции);
   7. Postman Version 6.7.4+ (эмуляция работы с сервисом);
   8. Docker desktop 3.0.4.
3. Особые условия:
   1. В постановке задачи рубль, как валюта, не содержал описания принадлежности к стране, поэтому реализация будет выполнена для стран: Россия (RUB), Беларусь (BYN);
   2. В поставке задачи не был указан сценарий при равенстве курсов. В реализации учтено как сценарий «рост курса»;
   3. При обращении к API Open Exchange Rates был выявлен дефект в документации (запрос https://openexchangerates.org/api/historical/ :date.json содержит лишний символ «**:**», исключено из запроса);
   4. Наименование валюты, по отношению к которой вычисляется курс, имена, адреса, идентификаторы приложений внешних сервисов и пр. вынесены в настройки Spring Boot;
   5. Подготовка архитектуры сервиса с описанием информационных потоков;
   6. Подготовка документации по проекту с использованием JavaDoc;
   7. Подготовка инструкции по запуску сервиса;
   8. Размещение проекта на GitHub;
   9. Сборка и запуск Docker контейнера с этим сервисом (задание со \*).
4. Источники данных:
   1. REST API курсов валют - https://docs.openexchangerates.org/;
   2. REST API гифок - <https://developers.giphy.com/docs/api#quick-start-guide>;
5. Открытые вопросы:
   1. Применение Docker compose для монтирования локальных файлов к контейнеру (Особенности настройки для ОС Windows).

# 3. Архитектура сервиса

# 3.1 Диаграмма архитектуры



# 3.2 Описание потоков сервиса

Архитектура поделена на 3 зоны:

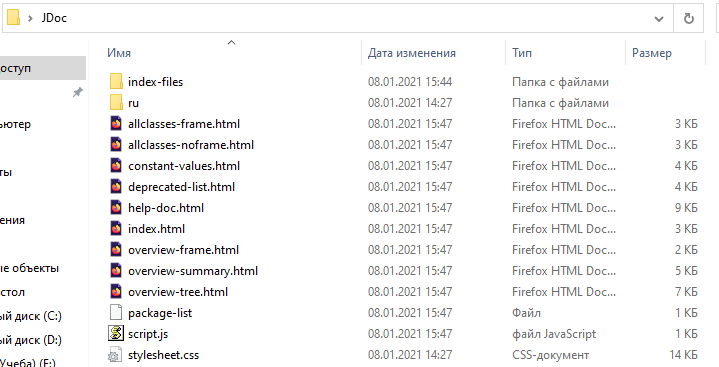
1. Внешние сервисы (источники данных);
2. Модули сервиса (набор элементов, описывающий бизнес-логику сервиса);
3. Объекты модуля (объекты, хранящие данные внешних сервисов, а также JSON).

Потоки сервисов представлены на диаграмме в виде нумерованных стрелок. Направление стрелки задаёт характер взаимодействия элементов архитектуры.

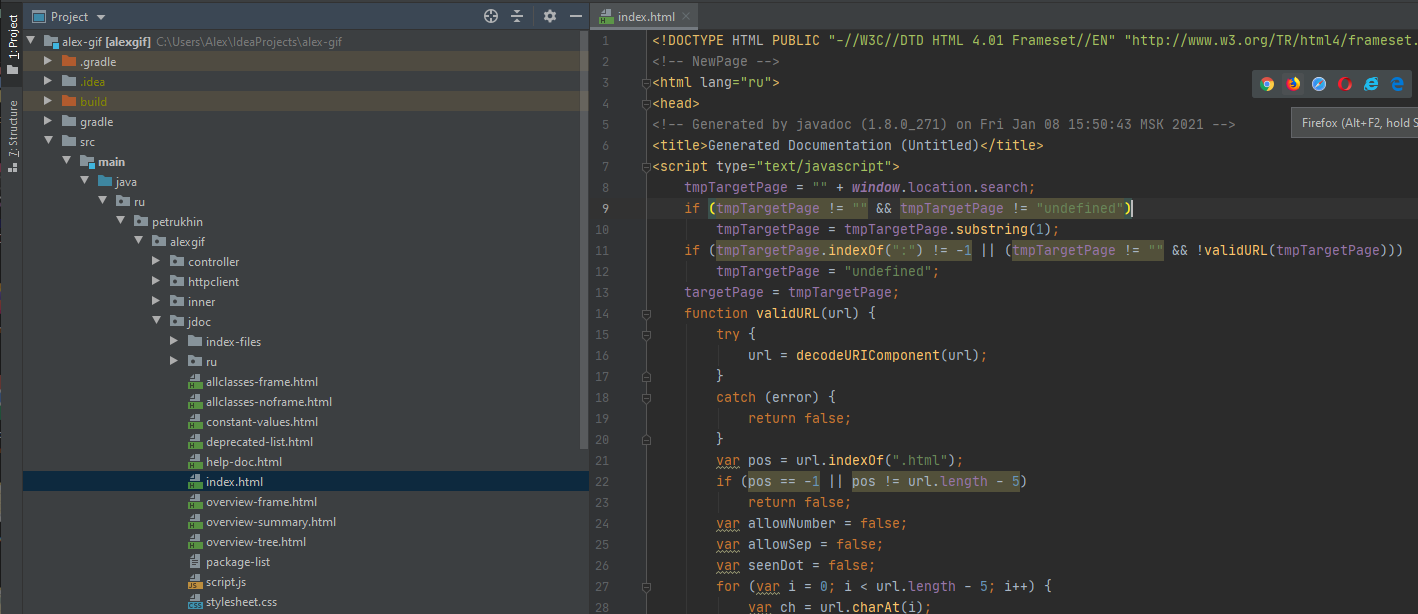
1. На вход ExchangeRatesController подаётся Inner JSON;
2. ExchangeRatesController обращается к InnerService;
3. InnerService создаёт объект Inner с данными из Inner JSON;
4. ExchangeRatesController, обращается к ExchangeRatesFeignClient с передачей наименования валюты;
5. ExchangeRatesFeignClient обращается к внешнему сервису Open Exchange Rates (OER) API для получения курса валюты;
6. Возврат ответа от ExchangeRatesFeignClient к ExchangeRatesController;
7. ExchangeRatesController обращается к RateService;
8. RateService создает объект Rates с данными по курсу валюты;
9. Объект Rates передается GifService для обработки (проверки условия);
10. GifService обращается к GiphyFeignClient с тегом по условию;
11. GiphyFeignClient обращается к внешнему сервису Giphy API для получения gif URL;
12. GifService создает объект Gif c gif URL;
13. GifService формирует Outer JSON.

# 4. JavaDoc

В процессе реализации решения описана документация JavaDoc, сформированная в среде разработки IntelliJ IDEA. Для доступа к документации необходимо загрузить файл JDoc.zip из репозитория GitHub, разархивировать и запустить файл index.html



Альтернативный способ работы с документацией является запуск из IntelliJ IDEA по пути: src/main/java/ru/petrukhin/alexgif/jdoc/index.html с вызовом браузера.



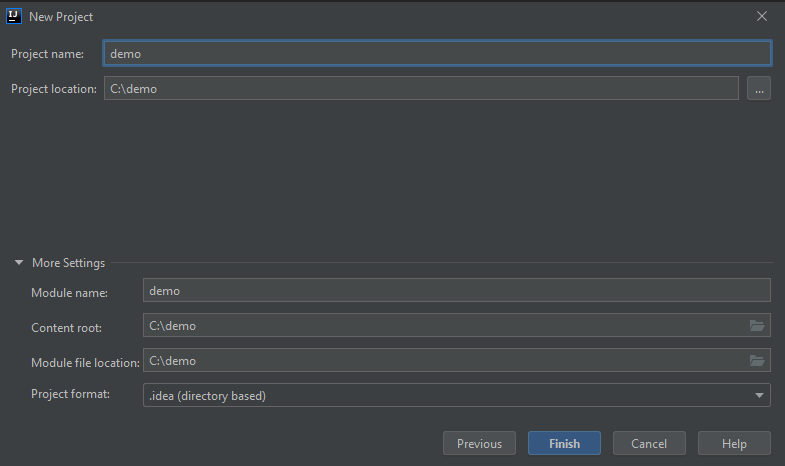
# 5. Инструкция по запуску

В процессе реализации решения подготовлены 2 варианта запуска сервиса:

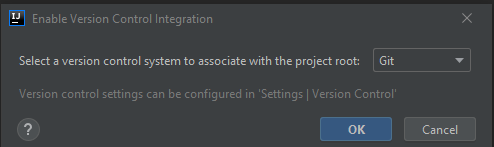
1. JVM Intellij IDEA;
2. Docker.

Перед запуском сервиса необходимо выполнить подготовительные работы:

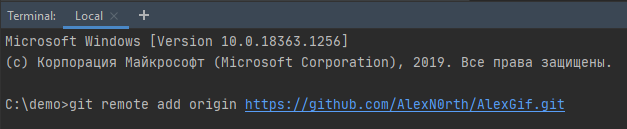
1. Подготовить в среде разработки новый пустой проект;

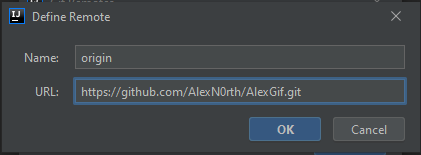


1. Включить поддержку системы контроля версий;



1. Связать локальный репозиторий с удаленным на GitHub с помощью команды или средствами Intellij IDEA по ссылке <https://github.com/AlexN0rth/AlexGif.git>;

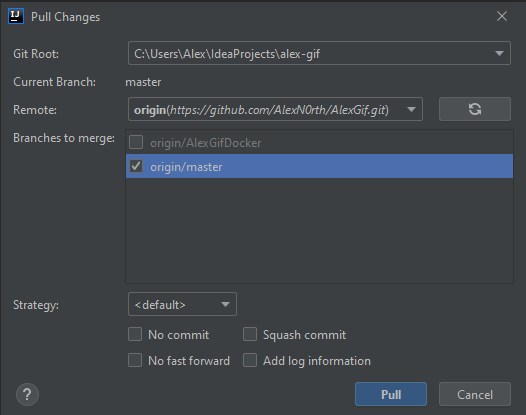




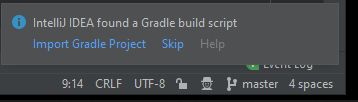
1. Далее по тексту описание действий в соответствии с вариантами запуска;

# 5.1 Запуск сервиса с использованием JVM Intellij IDEA

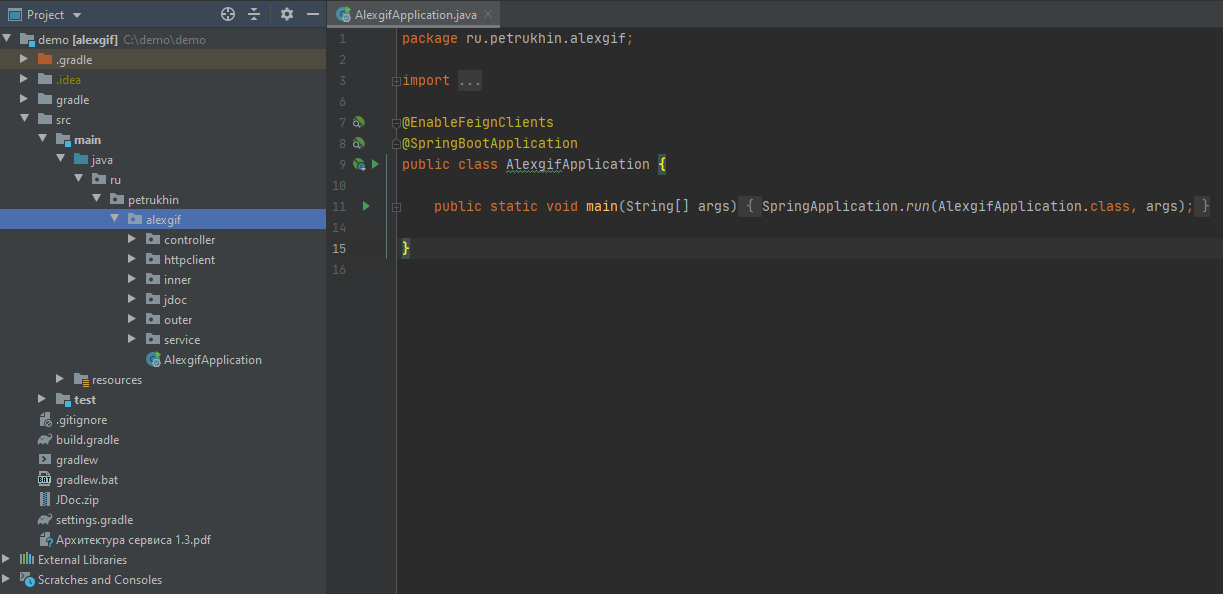
1. Выгрузить ветку *master* из репозитория проекта с GitHub;



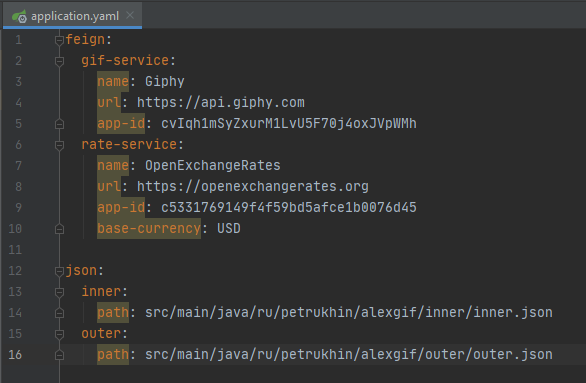
1. Настроить Gradle build script для сборки проекта;



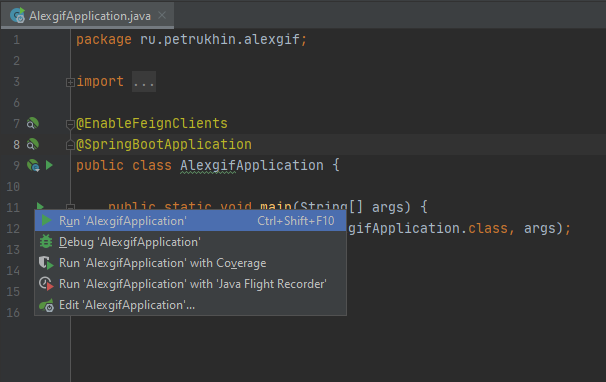
1. Проверить наполнение папок проекта, при необходимости перезагрузить Intellij IDEA;

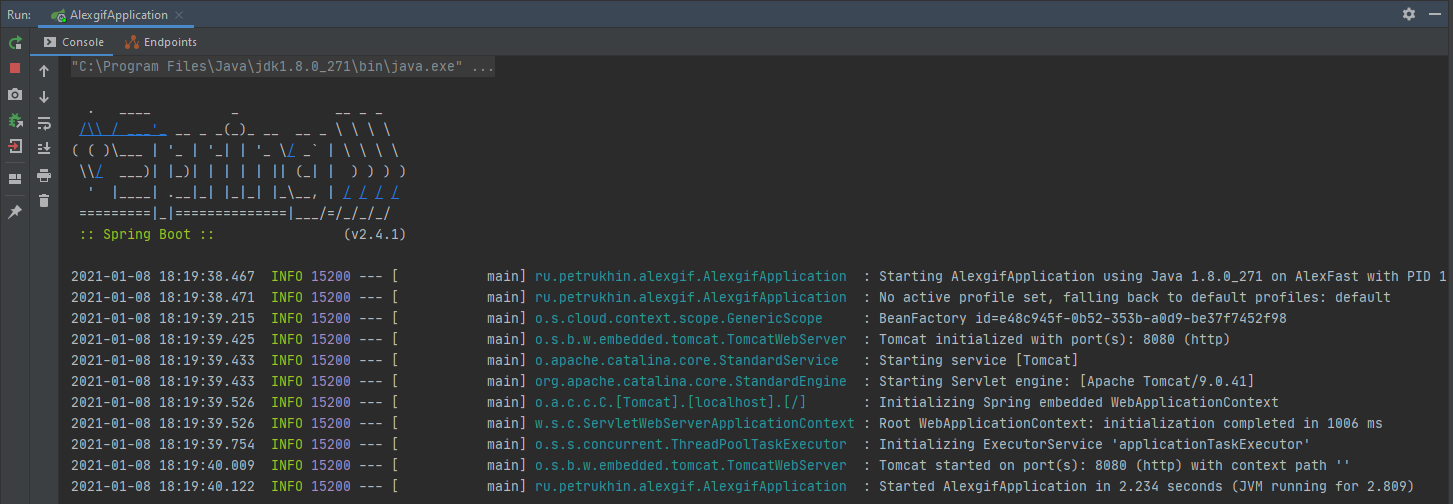


1. Файл настроек приложения application.yaml;

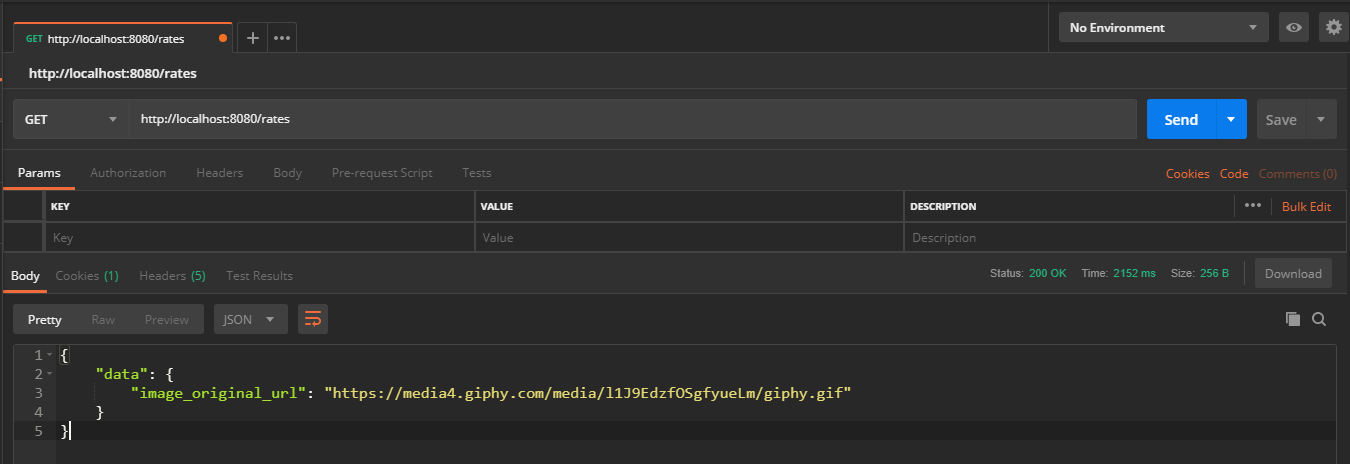


1. Запуск сервиса;

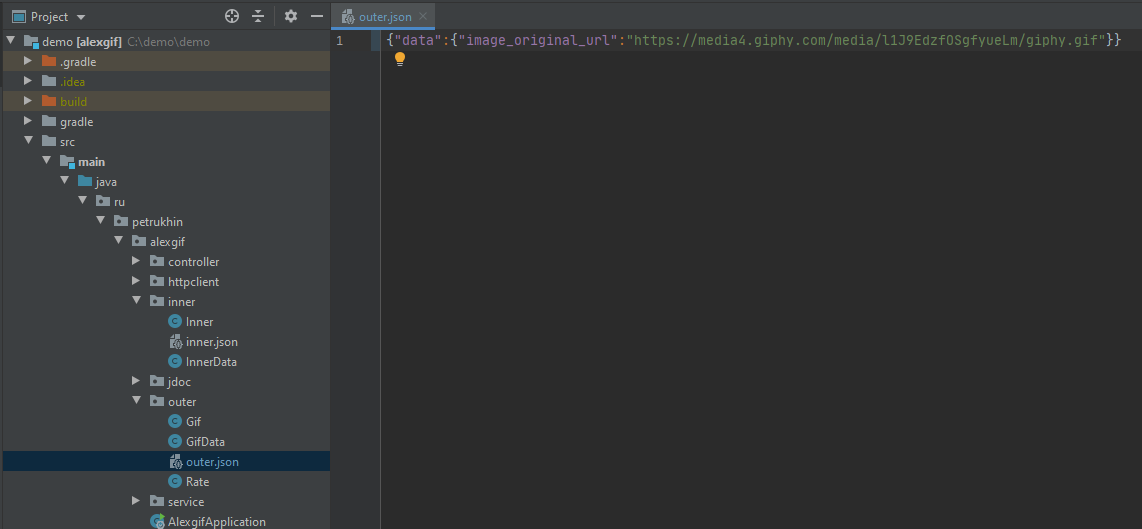




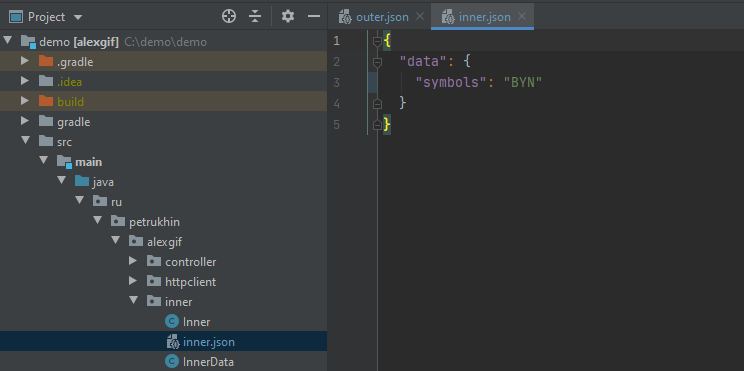
1. Запустить приложение Postman и сформировать запрос *GET* по адресу <http://localhost:8080/rates>;



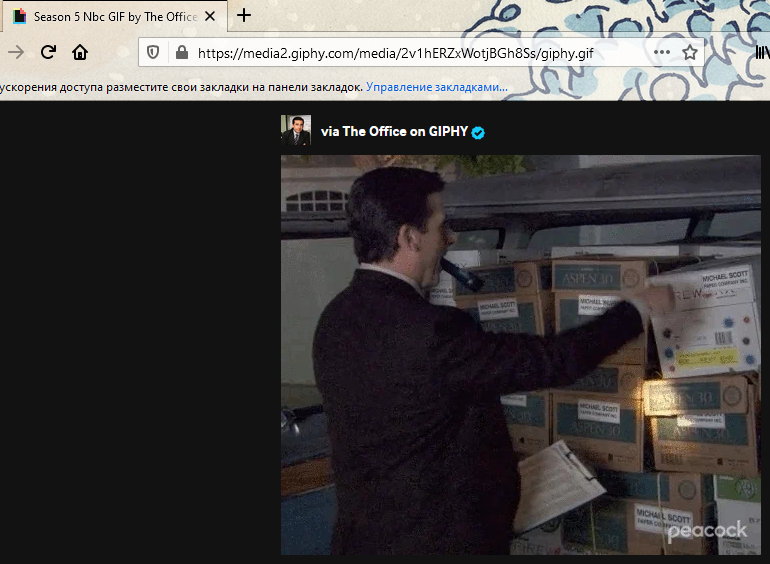
1. Проверить идентичность контента файла outer.json и отображения в Postman;



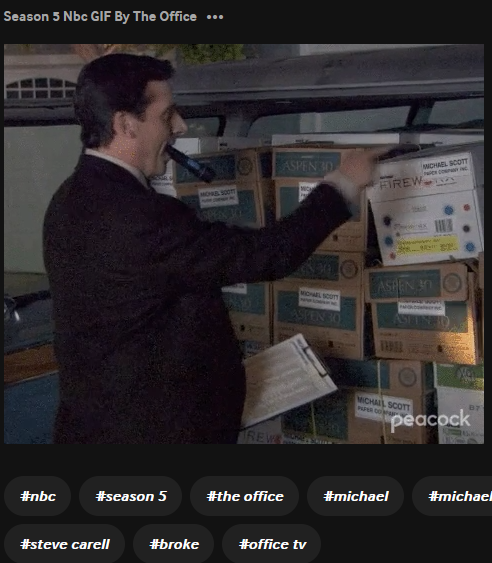
1. При необходимости изменить входящий JSON, указать значение BYN для ключа symbols;



1. Выполнить пункты 5-7, оценка выполнения;
2. При клике по ссылке URL Gif запускается браузер с отображением анимации Gif;



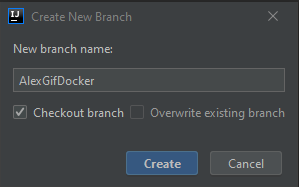
1. Для проверки тега Gif (rich/broke) кликнуть по изображению;



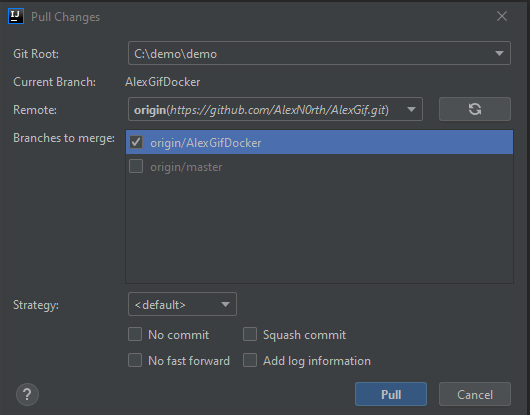
1. Сервис работает в соответствии с настройками.

# 5.2 Запуск сервиса с использованием Docker

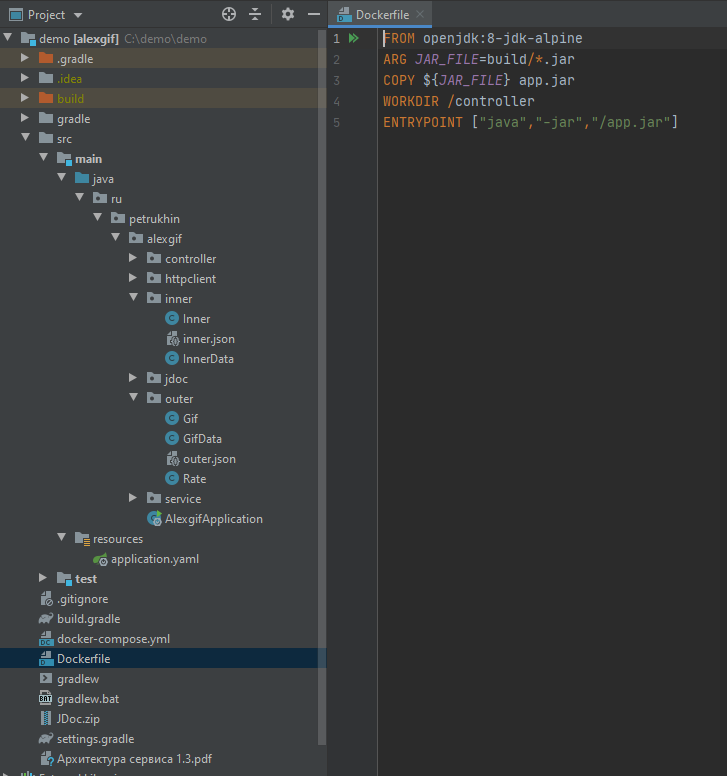
1. Создать новую ветку в локальном репозитории;



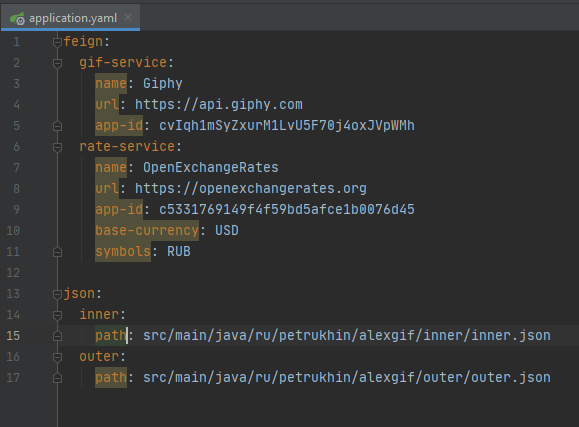
1. Выгрузить ветку *AlexGifDocker* из репозитория проекта с GitHub;



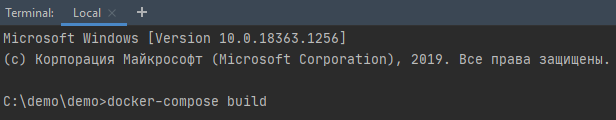
1. Проверить наполнение папок проекта, при необходимости перезагрузить Intellij IDEA. Добавляются файлы Dockerfile, docker-compose.yml и jar архив;



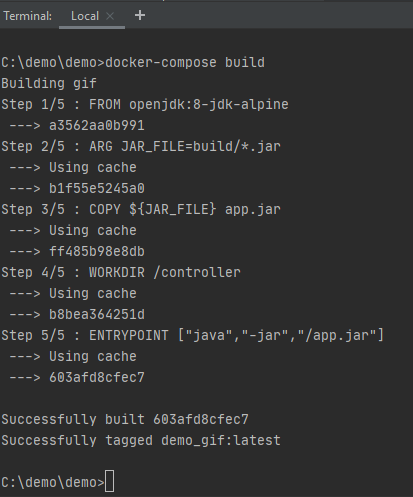
1. Файл настроек приложения application.yaml;

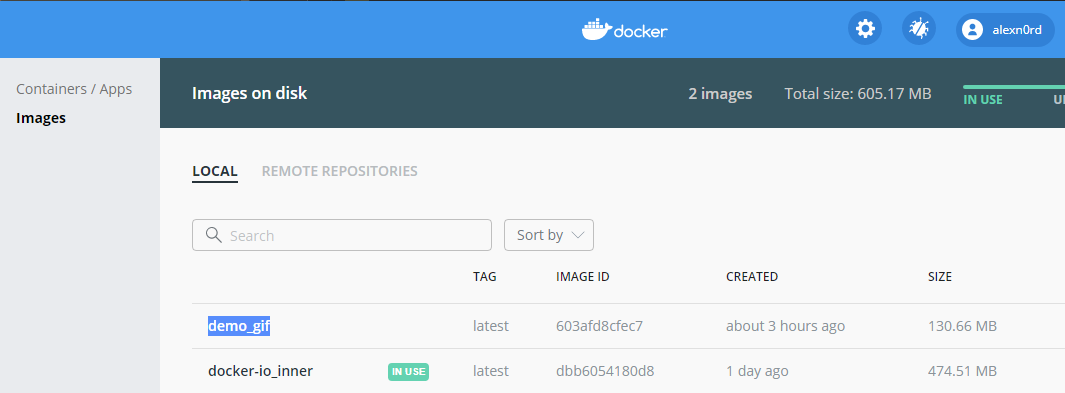


1. Выполнить команду сборки образа Docker контейнера

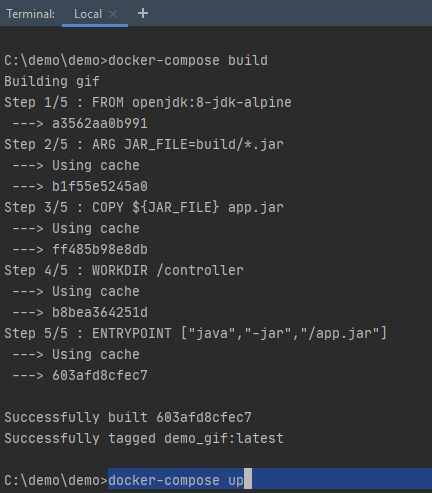


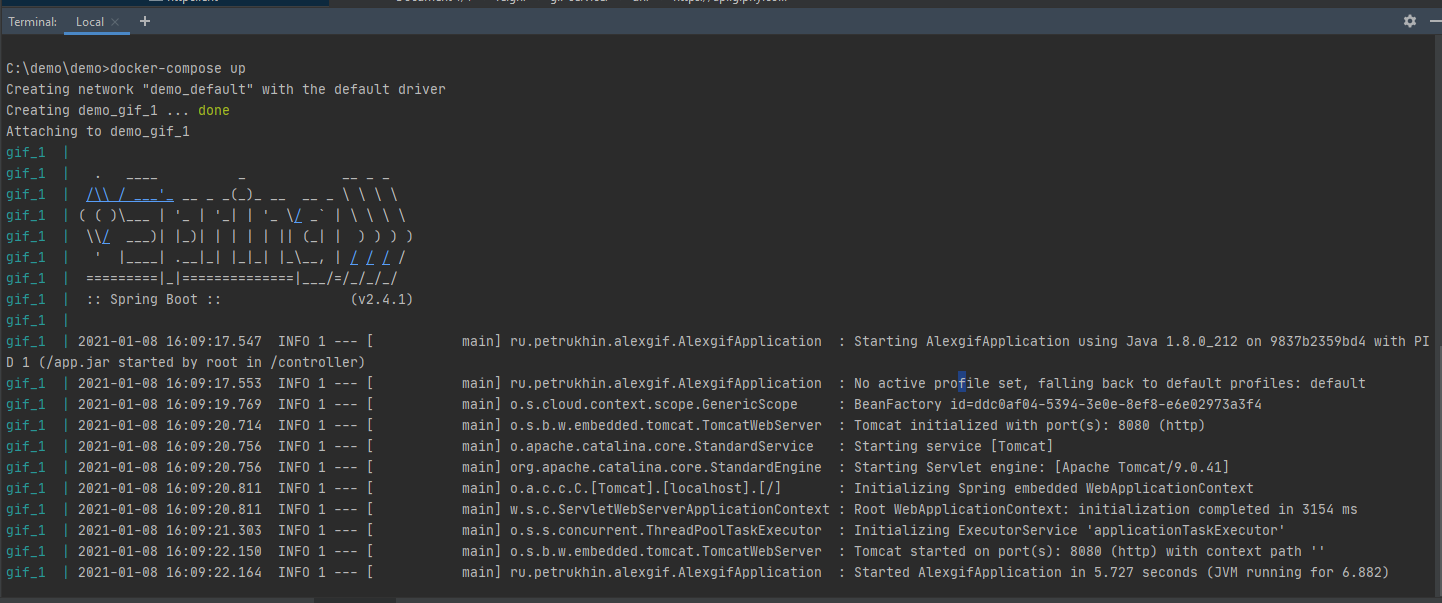
1. Проверить успех выполнения команды, создается образ. Также наличие образа можно проверить в приложении Docker desktop;



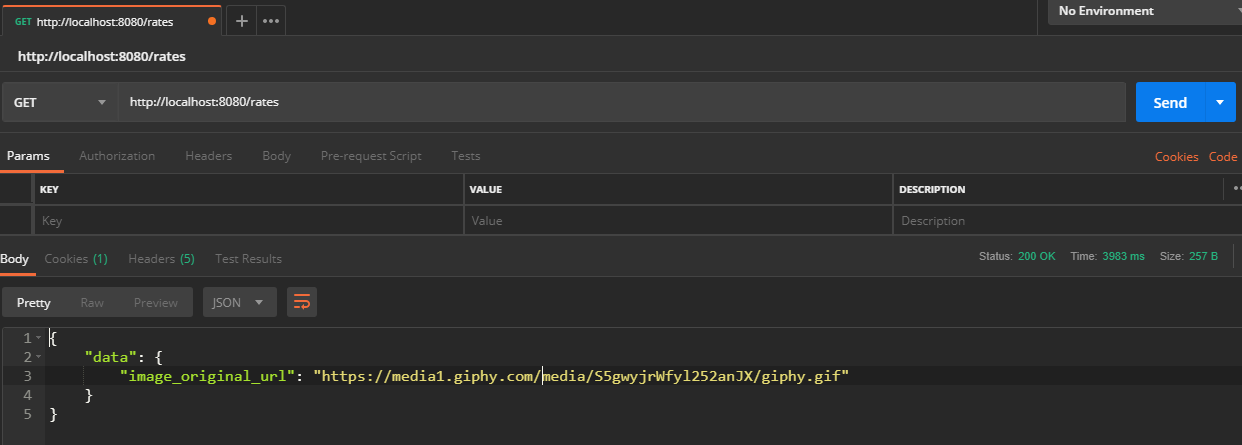


1. Выполнить команду сборки Docker контейнера

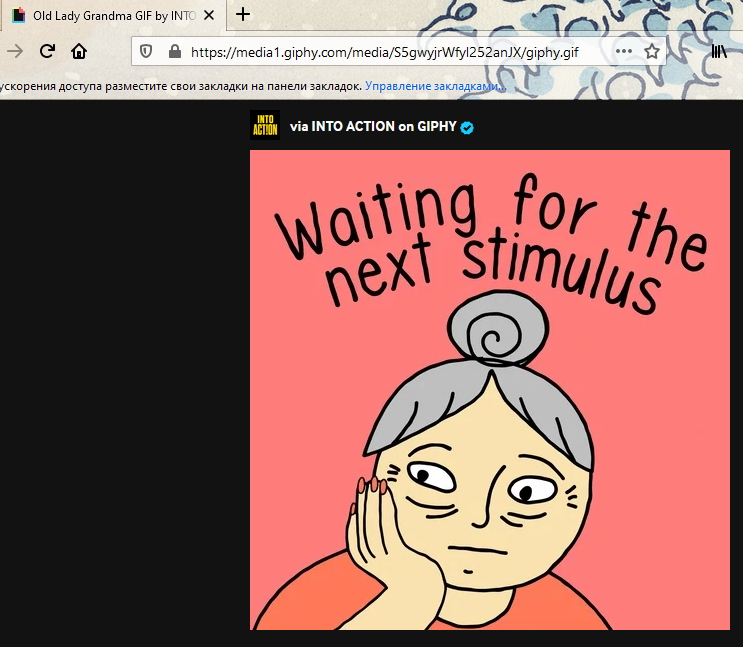




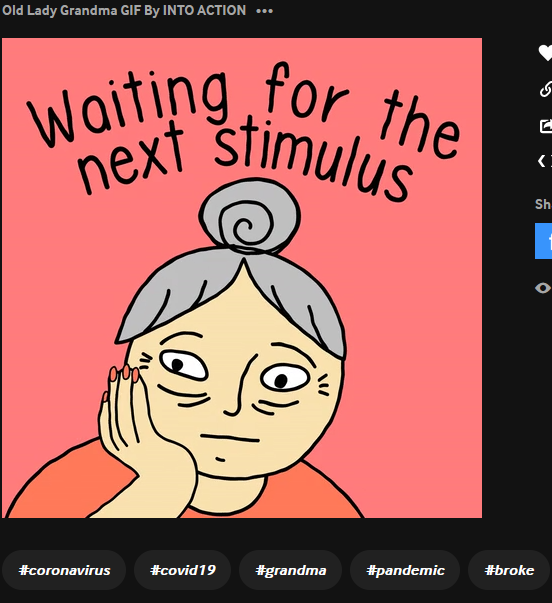
1. Запустить приложение Postman и сформировать запрос *GET* по адресу <http://localhost:8080/rates>;



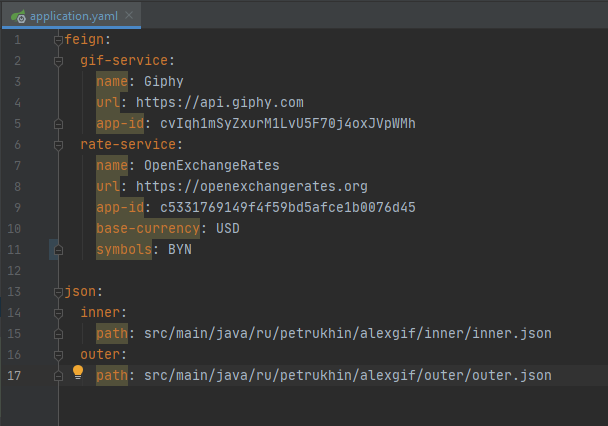
1. При клике по ссылке URL Gif запускается браузер с отображением анимации Gif;



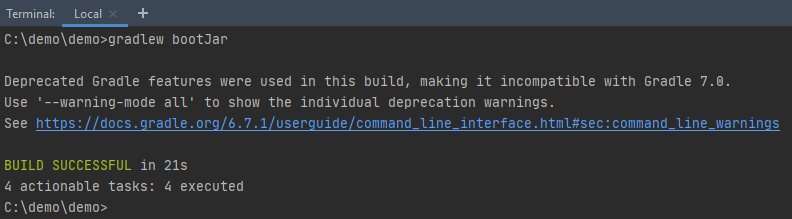
1. Для проверки тега Gif (rich/broke) кликнуть по изображению;



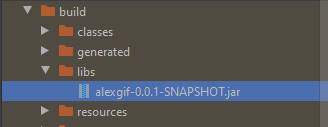
1. При необходимости изменить файл настроек application.yaml, указать значение BYN для ключа symbols

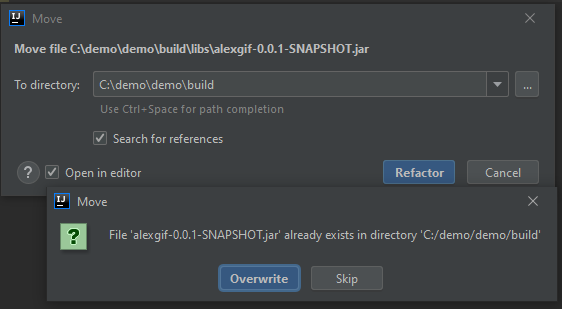


1. Выполнить команду сборки jar;



1. Переместить архив из build\libs в папку build с подтверждением перезаписи;





1. Выполнить пункты 5-10;
2. Сервис работает в соответствии с настройками.

# 6. Контактная информация

ФИО: Петрухин Алексей Дмитриевич

Телефон: +7(985) 073-53-25

Email: [alexnord95@gmail.com](mailto:alexnord95@gmail.com)

Ссылка на репозиторий GitHub: <https://github.com/AlexN0rth/AlexGif>