САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра Системного Программирования
Группа 20.Б11-мм

Винцукевич Михаил Михайлович

Разработка инфраструктуры библиотеки MIRF

Отчет по учебной практике

Научный руководитель:
доцент кафедры СП, к.т.н. Литвинов Ю.В.

Санкт-Петербург
2022

Содержание

Введение

Большинство проектов используют библиотеки, и чтобы библиотека была действительно полезной и лёгкой в использовании, для неё важно обеспечить:

1. Сопровождение,
2. Непрерывную интеграцию (CI),
3. Документацию,
4. Рефакторинг.

MIRF [1] — это проект, продвигаемый сотрудниками и студентами кафедры системного программирования СПбГУ, платформа с открытым исходным кодом для разработки медицинских приложений, в которых используются различные типы медицинских изображений. Исходный репозиторий [1] включает в себя библиотеку MIRF 2.0, микросервисы и демо–проекты.

Задачей практики было улучшение инфраструктуры проекта MIRF, а именно настройка непрерывной интеграции, улучшение документации, рефакторинг, и внедрение её в репозиторий, что может помочь при дальнейшем развитии проекта на кафедре в последующие года. Так же в ходе практики была задача объединения всех подпроектов из исходного репозитория в один проект, чтобы из корневой папки можно было работать как со всеми проектами разом, так и с каждым по отдельности. Проектом уже заинтересована одна частная клиника, что делает наличие хорошей инфраструктуры более актуальным.

Для практики была выбрана библиотека MIRF именно из-за того, что ей стали интересоваться, и потому что работа вокруг библиотеки идёт не переставая, хотя ядро системы давно никто не обновлял. Оно до сих пор является работающим прототипом, в то время как с развитием технологий вокруг проекта требуется поднять его на должный уровень качества. Дополнительно есть возможность, что в последующие года люди будут активнее брать задачи, связанные с MIRF и хотелось бы, чтобы им было легче приступить непосредственно к работе. В это можно поверить, учитывая, что в этом году сверх данной практики было взято ещё четыре проекта, связанные с MIRF.

На момент начала практики для проверки изменений в коде на Linux и macOS X использовался сервис Travis CI [3], а для платформы Windows – AppVeyor [4]. В данной работе предлагается настройка GitHub Actions

[5], обновление всех ключевых компонентов библиотеки, рефакторинг кода с использованием современных возможностей языка[1].

Также целью практики было желание научиться создавать и развивать инфраструктуру проектов.

[1]Важно прохождение всех тестов, желательно минимизировать предупреждения компилятора

1 Цели и задачи

Целью работы является разработка инфраструктуры для библиотеки MIRF. Для её выполнения были поставлены следующие задачи:

1. Объединение всех подпроектов из исходного репозитория в один проект,
2. Настройка CI,
3. Обновление версий языков и системы сборки:
    1) Kotlin до версии 1.5.31,
    2) Java до версии 16,
    3) Gradle до версии 7.1;
4. Рефакторинг кода.

2 Обзор репозитория

Главный репозиторий включает в себя:

1. Основную библиотеку MIRF,
2. Микросервисы,
3. Демо–проекты.

2.1 Основная библиотека

Библиотека MIRF представляет собой набор модулей для различных задач. Данные модули делятся на два глобальных пакета:

- Core — содержит модули для успешной работы библиотеки. Здесь содержатся классы для работы с данными для отображения данных, используемых в нуждах библиотеки. Диаграма классов показана на Рис 1[2].

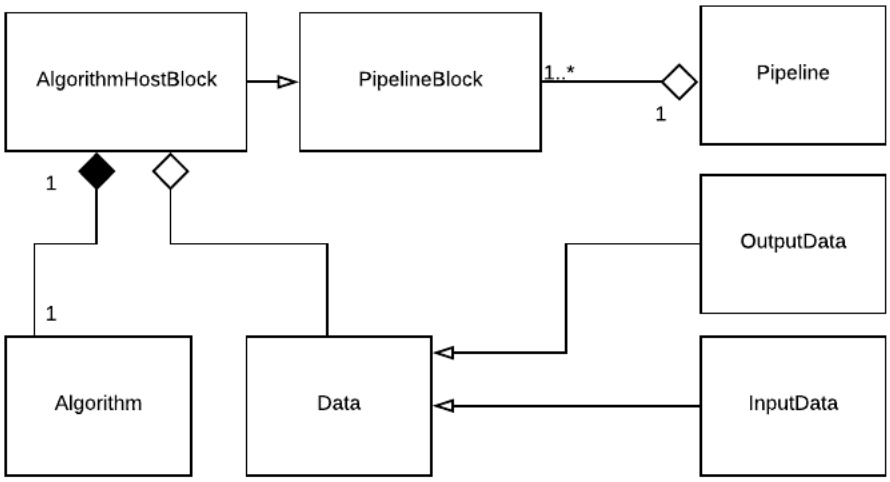

Рис. 1: Диаграмма классов пакета Core

- Features — содержит набор модулей, необходимых для облегчения разработки нестандартных сценариев работы. Пользовательские модули должны расширять этот пакет.

[2]Рисунок взят из работы [2]

2.2 Микросервисы

Проект включает в себя четыре микросервиса:

- block — занимается обработкой данных, сериализацией результата и загрузкой его в микросервис repository.

  При развёртывании запускается столько экземпляров этого микросервиса, сколько блоков участвует в вычислении. При необходимости они могут реплицироваться для параллельной обработки.

- orchestrator — управляет всей системой. Получает задания от пользовательского приложения, отправляет их сервису block, следит за текущим состоянием системы и балансирует нагрузку.

- repository — нужен для хранения данных.

- utils — сервис, в котором хранятся полезные инструменты, необходимые остальным сервисам.

2.3 Демо-проекты

Проект включает в себя три демо проекта:

- MirfBrainTumor — демонстрирует использование библиотеки MIRF путем создания конвейера для анализа изображений МРТ мозга и последовательного создания отчетов на основе полученных данных.

- MirfSkinCancer — использует MIRF для классификации изображения кожи из памяти телефона и определения того, может ли на изображении присутствовать случай рака кожи.

  Основная цель этой демонстрации — показать, что MIRF можно использовать не только на компьютере, но и на устройстве Android.

- MultipleSclerosis — демонстрирует работу библиотеки, проверяет медицинский снимок на наличие рассеянного склероза.

  Содержит в себе два подпроекта: MirfMs — клиентский, и MirfMsService — проект, где происходят вычисления.

3 Ход работы

3.1 Объединение всех подпроектов в один проект

Изначально было восемь проектов: четыре микросервиса, три демо–проекта и основная библиотека. Все они существовали независимо друг от друга. Задача была в объединении всех проектов в один, чтобы была возможность одной командой из консоли работать как сразу со всеми проектами, так и с каждым по отдельности.

Было принято решение не трогать демо проект MirfSkinCancer, так как работа с этим проектом была задачей другой практики.

После изучения устройства системы сборки Gradle, все проекты были интегрированы в один. Это позволяет сделать метод include, где перечисляются все подпроекты. После объединения проектов, Gradle позволяет работать через консольные команды как сразу со всеми, так и с каждым по отдельности. Для объединения подпроектов MIRF был использован следующий код:

```
include('demos:mirfBrainTumor', 'demos:MultipleSclerosis:mirf-ms-service',
'demos:MultipleSclerosis:mirfMs', 'microservises:utils',
'microservises:block', 'microservises:orchestrator',
'microservises:repository')
```

После интеграции была получена конфигурация проекта, показанная на Рис 2. На рисунке можно увидеть корневой проект MIRF, который содержит семь подпроектов, которые разбиты на две группы:

1. Группа demos:
    1) mirfBrainTumor;
    2) Подгруппа MultipleSclerosis:
        2.1) mirf-ms-service,
        2.2) mirfMs;
2. Группа microservices:
    1) block,
    2) orchestrator,
    3) repository,
    4) utils.

Так как микросервис utils создаёт файлы, необходимые для микросервиса block, то была добавлена зависимость проекта block от проекта utils, что позволяет при работе с проектом block проверять, был ли собран проект utils и в случае, если не был, собрать его. Это позволяет спокойно работать с проектом build, не собирая перед этим вручную проект utils.

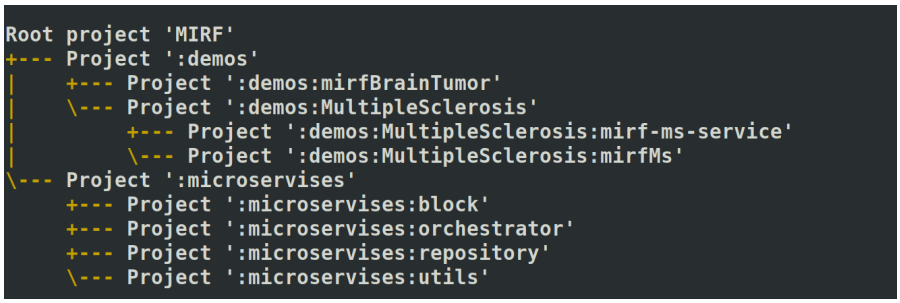

Рис. 2: Конфигурация проектов

3.2 Настройка CI

После слияния всех подпроектов в один, был настроен CI, а именно GitHub Actions:

1. Сборка проектов. При каждой отправке своих изменений в свою ветку работы начинается сборка всех проектов. Проверяется именно сама возможность сборки всех проектов.

2. Тесты проектов. При каждом Pull Request из своей ветки работы в основную запускаються тесты всех проектов. Проверяется прохождение всех тестов.

3.3 Обновление версий языков и системы сборки

Были обновлены:

1. Kotlin до версии 1.5.31,

2. Java до версии 16,

3. Gradle до версии 7.1.

Непосредственно для обновления Kotlin и Java пришлось просто поменять версии в gradle.build файле главного проекта. Единственная сложность была в удалении версий этих компонентов в подпроектах.

Для обновления gradle пришлось во всех проектах заменить версии в специальном gradle-wrapper.properties файле

3.4 Рефакторинг кода

3.4.1 Изменения после обновлений

После обновления всех ключевых компонентов, первым делом была задача исправления всех ошибок, которые появились именно из-за перехода

со старых версий на новые.

После обновления Kotlin нужно было изменить тот код, который был написан на старой версии языка, но не может быть запущен с использованием современных версий. Этот код подвергся изменениям в первую очередь.

Пример кода, который надо было изменить в первую очередь:

```
byteArray[i] = array[i].toByte()
intArray[i] = array[i].toInt()
shortArray[i] = array[i].toShort()
```

Первоначальный код с ошибками

```
byteArray[i] = array[i].toInt().toByte()
intArray[i] = array[i].toInt()
shortArray[i] = array[i].toInt().toShort()
```

Исправленный код

После обновления Gradle так же понадобилось менять код, написанный с использованием старых возможностей языка:

- В старых версиях использовалась конструкция 'compile ...',
- В новых версиях требуют писать 'implementation ...'.

3.4.2 Основные изменения

Первостепенной задачей рефакторинга было уменьшение количества предупреждений у компилятора, чтобы при этом проходили все тесты. За время осенней практики удалось поработать только с основной библиотекой, не беря микросервисы и демо–проекты. Результатом стало уменьшение количества предупреждений с 96 до 10, а так же явное улучшение кода:

1. Удалены многие неиспользуемые переменные, классы, конструкторы и т.п. При этом внимательно просматривалось, что они в будущем точно нигде не используются, а так же прохождение всех тестов.
2. Переработаны строки кода, где это возможно, с современными возможностями языка. Критерием было сокращение и/или улучшенное визуальное представление кода.

4 Заключение

В результате работы над учебной практикой в течение осеннего и весеннего семестров были выполнены следующие задачи:

1. Объеденены все подпроекты в один проект,
2. Настроен CI,
3. Обновлены:
    1) Kotlin до версии 1.5.31,
    2) Java до версии 16,
    3) Gradle до версии 7.1.
4. Провёден рефакторинг кода.

Вся работа велась в репозитории [6], было сделано 3 пулл реквеста в основной репозиторий [1]. Имя пользователя — Mishavint.

Список литературы

[1] Главный репозиторий проекта MIRF

`https://github.com/MathAndMedLab/MIRF2`

(Дата обращения: 19.05.2022)

[2] Мусатян С.А. Библиотека для создания программного обеспечения, использующего медицинские изображения. 2019.

`https://se.math.spbu.ru/thesis/texts/Musatjan_Sabrina_Andranikovna_Bachelor_Thesis_2019_text.pdf`

(Дата обращения: 19.05.2022)

[3] Travis CI

`https://github.com/marketplace/travis-ci`

(Дата обращения: 19.05.2022)

[4] AppVeyor

`https://github.com/marketplace/appveyor`

(Дата обращения: 19.05.2022)

[5] GitHub Actions

`https://github.com/features/actions`

(Дата обращения: 19.05.2022)

[6] Копия репозитория, в котором велась работа

`https://github.com/Mishavint/MIRF2`

(Дата обращения: 19.05.2022)