Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“Национальный исследовательский университет ИТМО”

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РЕШЕНИЕ ТИПОВЫХ ПЕДАГОГИЧЕСКИХ ЗАДАЧ ПРИ ОРГАНИЗАЦИИ ПРАКТИЧЕСКОГО КУРСА С ПОМОЩЬЮ СИСТЕМЫ FLAXO

Автор Цибин Андрей Игоревич
(Фамилия, Имя, Отчество)

Направление подготовки (специальность) 09.04.04,
(код, наименование)
Нейротехнологии и программная инженерия

Квалификация магистр
(бакалавр, магистр, инженер)*

Руководитель ВКР Ефимчик Евгений Александрович., к.т.н.
(Фамилия, И., О., ученое звание, степень)

Санкт-Петербург, 2020 г.

Обучающийся _Цибин Андрей Игоревич_____
(ФИО полностью)

Группа_P42212_____Факультет/институт/кластер_ПИиКТ_____

Направленность (профиль), специализация _Нейротехнологии и программная инженерия_
_____

Дата защиты “_18_” _июня_2020_г.

Секретарь ГЭК _Бутько Е.Ф._____
(ФИО)

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

УТВЕРЖДАЮ

Руководитель ОП

_Лисицына Л.С._____
(Фамилия, И.О.)

«_04_» «_февраля_____» 2020 г.

З А Д А Н И Е
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся_ Цибин Андрей Игоревич_____
(ФИО полностью)

Группа___P42212_____**Факультет/институт/кластер**_____ПИиКТ_____

Квалификация _____магистр_____
(магистр, инженер, бакалавр)**

Направление подготовки _09.04.04 Программная инженерия_____
(код, название направления подготовки)

Направленность (профиль) образовательной программы

Нейротехнологии и программная_инженерия_____

Специализация Разработчики и исследователи программного обеспечения_____
нейротехнологий_____

Тема ВКР_ Решение типовых педагогических задач при организации практического курса с помощью системы Flaxo_____

Руководитель_ Ефимчик Евгений Александрович, Университет ИТМО, доцент, к.т.н._____
(ФИО полностью, место работы, должность, ученая степень, ученое звание)

2 Срок сдачи студентом законченной работы до «__01__» «__июня_____» 2020___г.

3 Техническое задание и исходные данные к работе Требуется провести_исследование__ процесса эксплуатации системы Flaxo для организации практическов курсов по_____ программированию и оптимизировать процесс решения типовых педагогических задач и___ процессы обслуживания._____

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

_1. Анализ педагогического процесса и процесса технической поддержки._____

_2. Выделение неэффективных участков функционирования процессов._____

_3. Оптимизация отдельных неэффективных участков._____

_____

_____

_____

5 Перечень графического материала (с указанием обязательного материала)

_____

_____

_____

_____

_____

_____

_____

6 Исходные материалы и пособия

_____

_____

_____

_____

_____

_____

7 Дата выдачи задания «_04_» «_февраля_____» 2020__г.

Руководитель ВКР_Ефимчик Евгений Александрович_____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

АННОТАЦИЯ

ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Цибин Андрей Игоревчи
(ФИО)

Наименование темы ВКР: Решение типовых педагогических задач при организации практического курса с помощью системы Flaxo

Наименование организации, где выполнена ВКР Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования Исследование основных бизнес-процессов функционирования оригинальной системы Flaxo и их оптимизация

2 Задачи, решаемые в ВКР Анализ педагогического процесса и процесса технической поддержки, выделение неэффективных участков функционирования процессов, а также оптимизация отдельных неэффективных участков.

3 Число источников, использованных при составлении обзора

4 Полное число источников, использованных в работе 22

5 В том числе источников по годам

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
5			9	5	3

6 Использование информационных ресурсов Internet да, 2
(Да, нет, число ссылок в списке литературы)

7 Использование современных пакетов компьютерных программ и технологий (Указать, какие именно, и в каком разделе работы)

Пакеты компьютерных программ и технологий	**Раздел работы**
Draw.io	2
PostgreSQL, Kotlin, Docker, GitHub Actions, GitPlag	3
Python, MOSS, JPlag	4

8 Краткая характеристика полученных результатов В ходе выполнения настоящей работы были исследованы и оптимизированы основные бизнес-процессы оригинальной системы Flaxo, а именно процессы решения типовых педагогических задач и процесс технической поддержки. В частности, был разработан инструмент графовой визуализации результатов анализа плагиата, переработан механизм анализа плагиата, настроены процесса автоматизированной сборки, публикации и развертывания системы, а также добавлена пользовательская

документация.

9 Полученные гранты, при выполнении работы _____
(Название гранта)

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы да
(Да, нет)

а) 1 Цибин А.И., Ефимчик Е.А. Графовая визуализация плагиата в практических курсах по программированию // Сборник тезисов докладов конгресса молодых ученых. Электронное издание. - [2020, https://kmu.itmo.ru/file/download/application/13166]. - Режим доступа: ссылка на страницу с тезисом, своб.
(Библиографическое описание публикаций)

2

3

б) 1 Цибин А.И., Ефимчик Е.А. Графовая визуализация плагиата практических курсов по программированию : IX Конгресса молодых ученых, 2020.
(Библиографическое описание выступлений на конференциях)

2

3

Ministry of Science and Higher Education of the Russian Federation

ITMO University

SUMMARY
OF A GRADUATION THESIS

Student Tsibin Andrey Igorevich
(full name)

Title of the thesis Typical pedagogical problems solving for practical courses hosting using Flaxo system

Name of organization ITMO University

DESCRIPTION OF THE GRADUATION THESIS

1 Research objective Research and optimization of the main Flaxo system business processes.

2 Research tasks Analysis of the pedagogical and technical support processes, highlighting and optimization of non-efficient parts of the processes.

3 Number of sources listed in the review section _____

4 Total number of sources used in the thesis 22

5 Sources by years:

Russian			Foreign		
In the last 5 years	5 to 10 years	More than 10 years	In the last 5 years	5 to 10 years	More than 10 years
5			9	5	3

6 Use of online (internet) resources yes, 2
(Yes/No, number of items in the list of references)

7 Use of modern computer software suites and technologies (List which ones were used and for which section of the thesis)

Software suites and technologies	Thesis section
Draw.io	2
PostgreSQL, Kotlin, Docker, GitHub Actions, GitPlag	3
Python, MOSS, JPlag	4

8 Short summary of results/conclusions Flaxo system pedagogical and technical support processes were researched and optimized as the result of original thesis. In specific, graph-based plagiarism analysis results visualization tool was created, continuous integration, delivery and deployment processes were configured for the system and user documentation was created.

9 Grants received while working on the thesis _____
(Grant name)

10 Have you produced any publications or conference reports on the topic of the thesis? _yes_____
(Yes, no)

a) 1 _Цибин А.И., Ефимчик Е.А. Графовая визуализация плагиата в практических курсах по программированию // Сборник тезисов докладов конгресса молодых ученых. Электронное__ издание. - [2020, https://kmu.itmo.ru/file/download/application/13166]. - Режим доступа: ссылка на страницу с тезисом, своб. _____
(Bibliographical description of a publication)

2_____

3_____

b) 1_ Цибин А.И., Ефимчик Е.А. Графовая визуализация плагиата практических курсов по программированию : IX Конгресса молодых ученых, 2020. _____
(Bibliographical description of a conference report)

2_____

3_____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Активным развитием образовательных платформ в информационных технологиях давно уже никого не удивишь. Образование становится массовым и доступным. В таких условиях автоматизация связанных процессов играет ключевую роль.

В настоящее время активно создаются новые и развиваются существующие образовательные системы, всесторонним образом оптимизирующие работу преподавателей и обучающихся в различных областях науки и техники.

Так, в контексте автоматизации практических образовательных курсов по дисциплине «Программирование», в рамках выполнения бакалаврской выпускной квалификационной работы «Разработка информационной системы создания, сдачи и проверки заданий по дисциплине Программирование» на кафедре компьютерных образовательных технологий в 2018 году была создана образовательная платформа Flaxo, позволяющая решать широкий спектр задач автоматизации процессов организации заданий и проверки работ, связанных с написанием программного кода.

Настоящая диссертационная работа является естественным наследником оригинальной выпускной квалификационной работы и ее основной целью является исследование практического применения системы Flaxo и оптимизация всех бизнес-процессов.

Основным бизнес-процессом эксплуатации платформы является, безусловно, построенный педагогический процесс. Настоящее исследование включает его анализ, выделение неэффективных элементов и поиск путей по их улучшению.

Кроме того, одним из бизнес-процессов, не являющихся частью педагогического, является процесс технической поддержки самой системы Flaxo. В настоящем исследовании приведены анализ существующего процесса технической поддержки, а также оптимизация отдельных частей этого процесса.

Таким образом, в настоящей диссертационной работе приведены анализ практического использования системы Flaxo, выделение неэффективные

элементов бизнес-процессов платформы, описаны подходы по их улучшению, а также детализированы процессы проектирования, разработки и апробации обновленных элементов.

1 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ НАПРАВЛЕНИЙ РАЗВИТИЯ

1.1 Исходная система

В рамках выполнения оригинальной выпускной квалификационной работы была разработана и апробирована образовательная платформа Flaxo [20; 21; 22] (Рисунок 1), краткое описание которой приведено ниже в настоящем разделе.

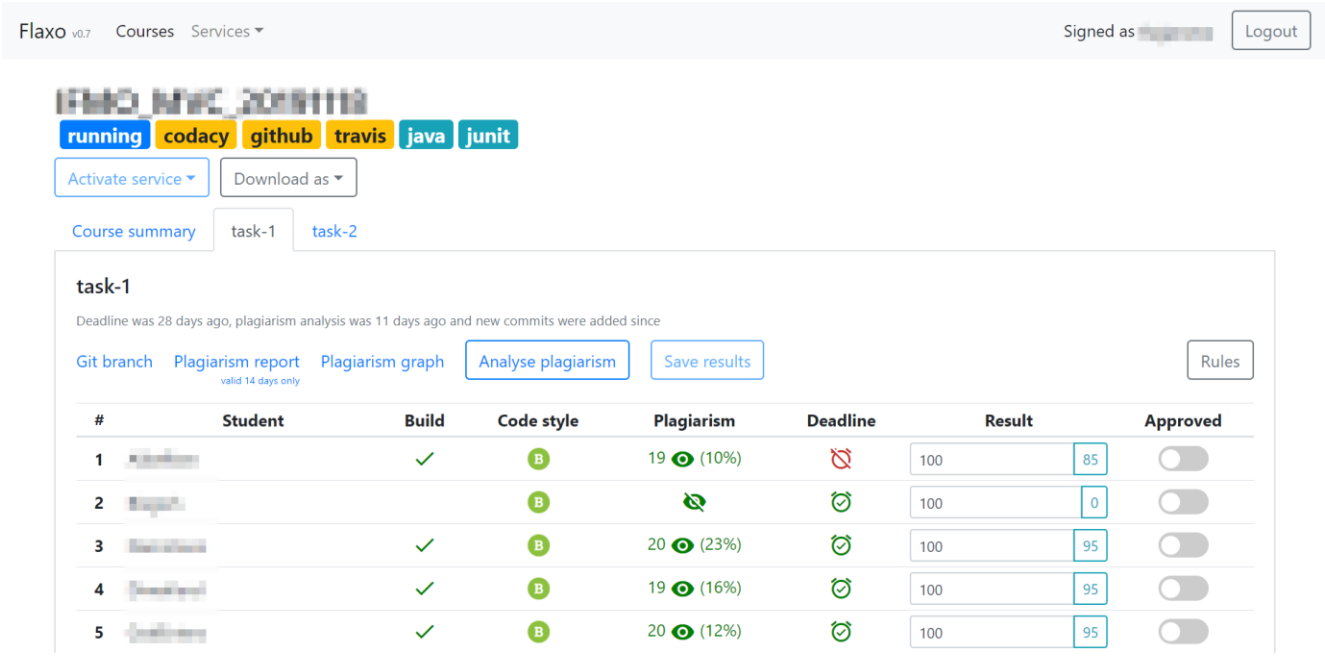

Рисунок 1 – Скриншот основной страницы системы Flaxo

Система (Рисунок 2) может использоваться преподавателями практических курсов по программированию с целью оптимизации своих трудозатрат на организацию и проведение обучающих курсов, а также проверку и оценку решений обучающихся. В терминах платформы каждый курс представляет из себя репозиторий исходного кода в системе контроля версий GitHub, задания – наборы автоматизированных тестов, написанных преподавателем, а решения – pull request-ы обучающихся к этому репозиторию. При этом каждое новое или обновленное решение подвергается ряду автоматических проверок, включающих:

– проверку исходного кода обучающегося соответствующими тестами, проводимую сервисом непрерывной интеграции Travis;
– проверку качества кода, осуществляемую системой статического анализа кода Codacy.

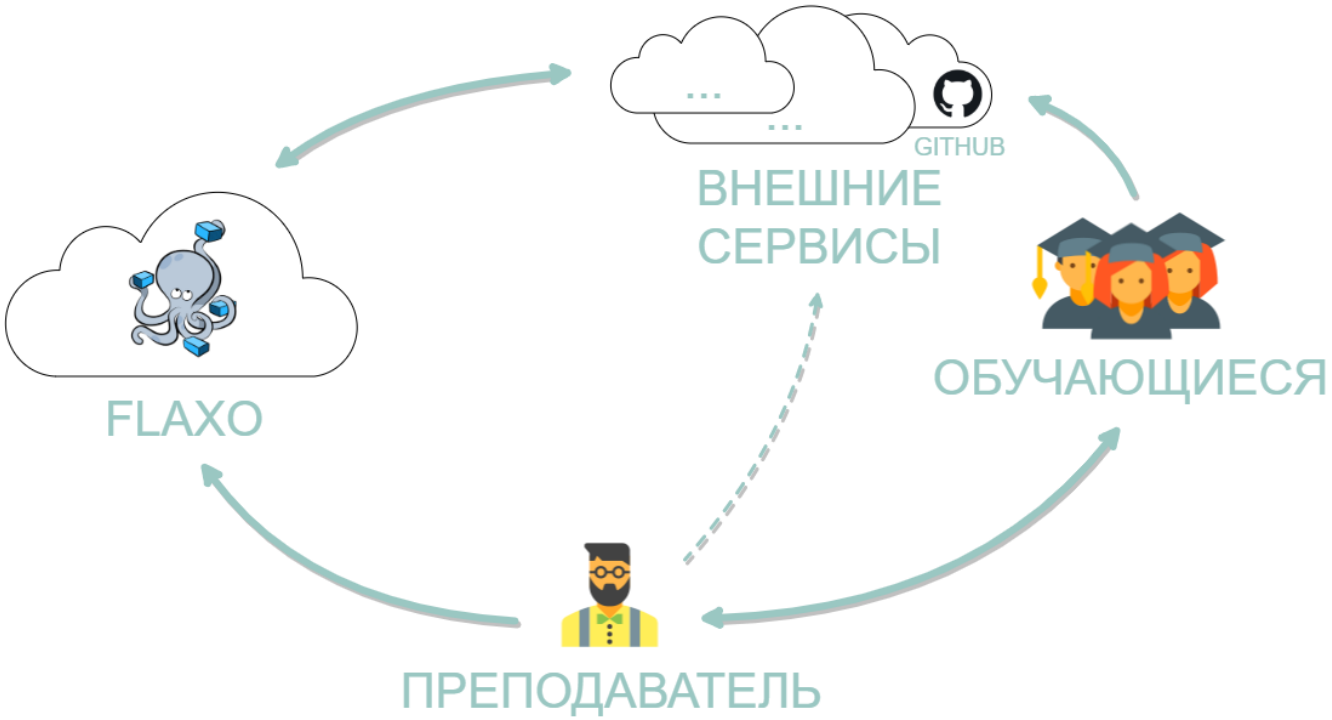

Рисунок 2 – Схема функционирования системы Flaxo

Отдельной возможностью, доступной преподавателю, является автоматизированный анализ плагиата имеющихся решений обучающихся, который производится по требованию системой анализа плагиата исходного кода Moss [1; 15] и результаты которого отображаются рядом с результатами остальных автоматизированных проверок.

На текущий момент система предоставляет пользовательский интерфейс только для преподавателя, исключая необходимость дополнительных действий по отношению к системе со стороны обучающегося. Таким образом процесс взаимодействия с практическим курсом для обучающегося выглядит очень похожим на процесс промышленной разработки.

В любой момент времени система отображает преподавателю текущую успеваемость обучающихся в разрезе решений отдельных заданий практического курса. Успеваемость визуализируется различными средствами, одним из них которых является табличное представление, включающее следующую информацию для каждого из обучающихся:

- индикатор прохождения решением обучающегося набора тестов, предоставленный сервисом непрерывной интеграции;
- маркер качества кода, рассчитанный сервисом статического анализа качества кода;
- обобщенный результат анализа плагиата решения обучающего, включающий количество найденных *потенциальных* случаев плагиата и процент совпадающего кода наиболее яркого из них;
- индикатор удовлетворения решения заданному дедлайну задания;
- табло рекомендуемых баллов за решение обучающегося, рассчитанных по определенной фиксированной формуле, которая учитывает результат каждой из автоматизированных проверок.

Для каждого из решений обучающихся преподаватель имеет возможность установки баллов самостоятельно или автоматически, используя кнопку применения рекомендуемой оценки.

Помимо прочего, преподаватель может просматривать сводную таблицу баллов обучающихся всех заданий практического курса, также экспортировать ее в одном из доступных файловых форматов.

На текущий момент разработанная в рамках оригинальной выпускной квалификационной работы система Flaxo активно применяется для проведения практических курсов по программированию в рамках учебных программ Университета ИТМО, а также за его пределами.

За время практического применения оригинальной системы был накоплен и продолжает накапливаться широкий опыт использования платформы, который позволяет более глубоко исследовать построенные в рамках системы бизнес-

процессы, а также выдвигать актуальные требования по улучшению отдельных аспектов функционирования платформы.

1.2 Актуальность среди аналогов

Несмотря на то что обзор аналогов не является неотъемлемой частью настоящего исследования, является важным утверждение об актуальности развития оригинальной платформы. В связи этим далее в настоящем разделе приведен обзор аналогов оригинальной системы Flaxo.

На текущий момент количество платформ, поддерживающих возможность организации практических курсов по программированию с функциями автоматизированных проверок довольно широкий. При этом список систем, по спектру решаемых задач автоматизации проверок решений сравнимых с Flaxo, значительно меньше. Основными платформами-аналогами являются следующие:

- Coursera,
- Edx,
- Stepik,
- JetBrains Education.

Далее в настоящем разделе приводятся описание каждого из сервисов и анализ представленных возможностей по автоматизации проверок решений практических курсов по программированию.

Образовательная платформа Coursera предоставляет базовые возможности по автоматизации проверки решений, при этом для создания нестандартных механизмов проверок, таких как, например, анализ качества кода, анализ плагиата, существует возможность написания специальных алгоритмов оценивания (ориг. custom grader, https://github.com/coursera/programming-assignments-demo/tree/master/custom-graders), которые запускаются в Docker-контейнерах и предоставляют практически неограниченные возможности по проверке решений. Несмотря на то, что платформа предоставляет набор инструментов, упрощающих

написание собственных контейнеров, его создание и отладка полностью лежит на преподавателе.

Платформы Edx и Stepik предоставляют базовые возможности по созданию автоматических проверок решений практических курсов по программированию, а для создания неспецифичных проверок предоставляется возможность подключения собственных внешних алгоритмов оценивания (ориг. external grader, https://www.jetbrains.com/help/education/educator-start-guide.html?section=Java), которые представляют из себя отдельные веб-сервисы, оценивающие решения обучающихся, которые сервис получает, пользуясь REST API соответствующей платформы. При этом задача разработки веб-сервиса, а также его хостинга стоит перед отдельным преподавателем.

Система JetBrains Education выступает и как самостоятельная образовательная система, позволяющая организовывать практические курсы по программированию с автоматизированными проверками, и как удаленный интерфейс решения заданий по программированию для таких систем, как Stepik и Coursera. Система позволяет решать задания по программированию прямо в Intellij Idea, Android Studio или PyCharm – интегрированных средах разработки от компании JetBrains, которые распространяются, в том числе, бесплатно. JetBrains Education предоставляют возможности создания автоматизированных проверок заданий по программированию путем написания автоматизированных тестов, а также выделяет нарушения качества кода прямо в редакторе, а в некоторых случаях даже позволяет ему исправить их автоматически. При этом, однако, при подсчете результирующего балла за решение качество кода не учитывается.

Из всех рассмотренных систем только JetBrains Education предоставляет простой способ создания автоматизированных тестов для решений, остальные же оставляют решение задачи автоматизации механизма запуска тестов за преподавателем. Помимо прочего, все системы за исключением указанной поддерживают проверки исключительно одного файла с исходным кодом, однако, в большинстве практических задач в области программирования речь идет о намного большем множестве исходных файлов. С другой стороны, JetBrains

Education в отличие от остальных платформ в качестве шкалы оценки решений использует бинарную. Кроме того, платформа не предоставляет возможностей по автоматизации проверок никаким образом, кроме как автоматизированными тестами.

Помимо прочего, из представленных систем ни одна не предоставляет функциональности по использованию инструментов анализа плагиата исходного кода, перекладывая трудоемкость конфигурации подобного процесса на преподавателя.

Таким образом, можно заключить, что использование и развитие оригинальной системы Flaxo является актуальным и позволяет преподавателю эффективно проводить практические курсы по программированию, сводя к минимуму трудозатраты преподавателя на конфигурацию окружения.

1.3 Основные бизнес-процессы

В рамках исследования функционирования системы Flaxo необходимо рассмотреть два основных косвенно связанных бизнес-процесса: педагогический процесс, а также процесс технической поддержки платформы.

1.3.1 Педагогический процесс

Педагогический процесс подразумевает под собой все этапы, происходящие между созданием практического курса и до момента его окончания, наблюдаемые основными участниками бизнес-процесса, а именно преподавателем и обучающимися.

Важно понимать, что педагогический процесс описанный ниже в настоящем разделе упрощен для представления в виде набора последовательных этапов. На практике этапы могут перекрываться, хотя общих порядок в большинстве случаев именно такой.

Процесс начинается с инициализации практического курса. На основании выбранных преподавателем параметров в системе формируется новый практический курс, для которого автоматически создается связанный репозиторий в системе контроля версий. В качестве конфигурируемых параметров при создании курса преподаватель имеет возможность выбрать название, описание, число заданий и используемые технологии, такие как язык программирования решений, язык программирования тестов и тестовый фреймворк. На основании выбранных технологий система автоматически может создать некоторое стандартное окружение, если подобная конфигурация предусмотрена.

Второй этап заключается в описании заданий практического курса и формализации требований к решениям обучающихся в виде тестов, которые будут автоматически запускаться для каждого из решений обучающихся. На этом этапе преподаватель взаимодействует непосредственно с системой контроля версий, в которой и сохраняются все описания и тесты.

На следующем этапе после того как преподаватель запускает курс, обучающиеся начинают работать с репозиторием, выполнять задания и публиковать решения в виде пулл реквестов. Обучающиеся взаимодействуют непосредственно с системой контроля версий, не отвлекаясь на взаимодействие с системой-посредником Flaxo.

После того, как обучающиеся публикуют свои решения в виде пулл реквестов к соответствующему репозиторию практического курса, автоматически срабатывают настроенные сервисы непрерывной интеграции. Спустя некоторый промежуток времени сервисы автоматически оповещают систему Flaxo о результатах прохождения каждого решения каждой из проверок. В результате платформа автоматически обновляет статистику об успеваемости обучающихся и преподаватель в любой момент времени имеет наиболее актуальную картину успеваемости каждого из обучающихся.

В какой-то момент времени, например, после наступления дедлайна задания или перед выставлением окончательных баллов обучающихся, преподаватель запускает анализ плагиата для каждого из заданий практического курса. Спустя

несколько минут после запуска анализа плагиата, преподаватель получает доступ к непосредственным результатам анализа плагиата и простейшей визуализации найденных заимствований.

На основании полученных данных преподаватель исследует решения, в которых были найдены заимствования, и пытается определить, какие из зафиксированных случаев не являются ложными срабатывания и требует применения мер в отношении недобросовестных обучающихся.

После исследования результатов анализа плагиата и применения необходимых мер воздействия на недобросовестных обучающихся, если такие имеются, чаще всего следует этап окончания курса, когда преподаватель перестает оценивать новые решения, а система перестает их регистрировать. В этот момент баллы, выставленные преподавателем, могут быть пересмотрены с учетом исследования результатов анализа плагиата и окончательно оглашены обучающимся преподавателем самостоятельно.

Каждый из описанных этапов педагогического процесса задействует или преподавателя, или обучающегося. Система Flaxo призвана минимизировать трудозатраты обеих ролей на поддержание этого процесса путем автоматизации максимально возможного числа шагов.

1.3.2 Процесс технической поддержки

Другим крупным рассматриваемым процессом является процесс технической поддержки системы Flaxo. В настоящем исследовании под технической поддержкой подразумевается набор характеристик, которыми должен обладать программный продукт для эффективно использования, обновления и резервного копирования данных.

Будучи свободно распространяемым решением с открытым исходным кодом, платформа может быть развернута в любом подходящем окружении. Тем не менее, процесс развертывания включает ряд шагов, которые должны быть проделаны администратором самостоятельно.

В частности, развертывание включает в себя использование операционной системы Linux и установку ряда зависимостей: языков программирования, инструментов сборки и других. Кроме того, одним из требований является наличие локально развернутой базы данных PostgreSQL, с необходимой конфигурацией резервного копирования. При первом развертывании или последующих обновлениях системы администратору необходимо вручную обновлять зависимости, собирать и развертывать приложение.

Описанный процесс технической поддержки системы организован максимально примитивно, тем не менее он позволяет выполнять основные манипуляции по использованию системы, как программного продукта.

1.4 Направления развития

На основании рассмотренных основных бизнес-процессов системы Flaxo становится возможным исследовать возможные пути по оптимизации ее работы. Далее в настоящем разделе рассматриваются существующие особенности и недостатки функционирования платформы и выдвигаются гипотезы о подходах к их улучшению.

Рассматриваемые особенности и выдвигаемые гипотезы группируются в так называемые направления развития. Выделение основных направлений развития платформы позволяет рассматривать процесс улучшения системы в виде большого набора некоторых локальных независимых улучшений, которые проще исследовать и с которыми можно более эффективно работать.

На основании рассмотренных бизнес-процессов и с учетом накопленного практического опыта в настоящей работе можно выделить широкий ряд направлений развития. Тем не менее основными направлениями развития системы являются следующие:

- добавление новых функций,
- персонализации настроек существующих возможностей,

- развитие визуализации представления информации,
- оптимизация процессов развертывания системы,
- добавления пользовательской документации.

Отдельным направлением развития платформы Flaxo являются архитектурные преобразования, позволяющие открыть широкие возможности по модуляризации системы, которые в свою очередь упростят добавление и изменение функционала платформы в целом, а также позволят увеличить тестовое покрытие отдельных модулей.

1.4.1 Расширение возможностей

Говоря про первое направление развития, стоит обозначить, какие педагогические задачи в системе еще не решаются, однако имеют значительный эффект на пользовательский опыт преподавателя. Список основных нерешенных задач представлен ниже:

- Связь между реальные имена обучающихся и их никнеймами на GitHub. В текущей версии преподавателю необходимо вручную сопоставлять имена, чтобы проставить баллы обучающихся в систему своей организации. Кроме того, отсутствует дополнительная информация об обучающихся, как например, номер учебной группы или название университета.
- Отслеживание обучающимися свой успеваемости. В текущей версии обучающийся узнает количество полученных баллов за решение только после момента выставления баллов преподавателем во внутреннюю систему организации.
- Перезапуск курса для новой группы обучающихся. В оригинальной системе при перезапуске курса преподавателю необходимо вручную переносить задания из одного репозитория в другой.

- Возможность создания нестандартного курса. Вероятна ситуация, при которой преподавателю необходимо использовать систему для практического курса на одном из языков, которые не поддерживаются в качестве языков, для которых существует возможность автоматической генерации курсов.
- Отслеживание прогресса долго идущих процессов, таких, как например, анализ плагиата. Подобные процессы могут длиться от нескольких секунд до нескольких минут в зависимости от размеров групп обучающихся и самих заданий. В течение всего этого времени у преподавателя нет возможности отслеживать прогресс того или иного процесса.
- Пометка решения, как верифицированного или как требующего доработки. В связи с этим преподавателю необходимо вести эту статистику самостоятельно, например, в системе контроля версий GitHub.
- Добавление дополнительных проверок к заданиям, помимо представленных по умолчанию.
- Обработка часто совершаемых обучающимися ошибок, связанных с оформлением своих решений. Например, обучающиеся часто могут оформить pull request не к той ветке или добавить в индекс системы контроля версий лишние файлы. Преподаватель должен отслеживать такие случаи вручную и указывать обучающимся на их ошибки.

В качестве решения проблем, связанных с сопоставлением именем обучающихся их профилям на GitHub, а также отслеживанием ими собственной успеваемости, может являться введение в системе дополнительной роли и создание отдельного интерфейса для *обучающихся.* В основные задачи данного интерфейса входит отображение статистики авторизованного обучающегося и возможность указания его личной информации.

Проблему перезапуска образовательного курса можно полностью автоматизировать и при необходимости дублировать репозитории, использую команду fork, которая позволяет создать новый репозиторий на основе уже существующего.

В оригинальной системе присутствует возможность генерации репозитория курса на основе выбранных технологий: языка программирования решений, языка программирования тестов и тестового фреймворка. При этом, возможности генерации курса без указания этих параметров не существует. В качестве решения этой проблемы необходимо сделать необязательными настройки языка программирования тестов и тестового фреймворка, а в качестве возможных языков программирования решений добавить большое количество самых распространенных языков программирования.

Сложность работы с долгими процессами можно значительно снизить, если добавить в систему возможность отслеживать выполнение задач с отображение прогресс-бара и логов запущенной операции.

В качестве решения проблемы с невозможностью отметки решений, как успешно прошедших верификацию, а также как требующих некоторых изменений, можно применить подход, использованный для похожей классификации pull request-ов на GitHub, а именно механизм одобрения изменений. Система Flaxo может использовать этот механизм в качестве фундамента и синхронизировать те или иные изменения статуса решения в двустороннем порядке между двумя сервисами.

В настоящей системе существует несколько автоматизированных проверок, которые устанавливаются для каждого из курсов по умолчанию. При этом, у преподавателя может существовать потребность использования отличных наборов проверок для разных курсов и добавления собственных проверок аналогично тому, как это осуществляется у некоторых из описанных систем-аналогов. Чтобы добавить подобное поведение в систему, необходимо унифицировать все существующие проверки к одному интерфейсу, а также предоставить возможность подключения дополнительных на уровне исходного кода платформы. При таком

подходе, задачи инициализации, оценки и визуализации могут быть решены одномоментно.

Обработку типичных ошибок обучающихся можно автоматизировать путем добавления механизма обратной связи с обучающими, который будет использовать комментарии и механизм approve\disapprove в pull request-ах в качестве нотификации пользователя о совершенной ошибке с указанием возможных путей решения проблемы. Кроме того, обратная связь может включать в себя изменение различных параметров pull request-а.

1.4.2 Персонализация настроек

Другим направлением развития, которое приводится в настоящей работе является персонализации настроек. В частности, речь идет об изменениях механизмов инициализации заданий практических курсов и вычисления рекомендованного балла за решение, а также о кастомизации процесса анализа плагиата.

Инициализация заданий в оригинальной системе осуществляется в момент создания курса. При таком подходе преподаватель должен знать точное количество заданий будущего курса еще до момента его создания, однако, вполне возможно, что в процессе разработки заданий окажется, что необходимо изменить число заданий. Поэтому, одним из подходящих решений является замена существующего принципа инициализации заданий механизмом добавления, удаления и изменения заданий.

Существование механизма рекомендации баллов за решение показало высокую эффективность в рамках проведения нескольких практических курсов по программированию. Однако механизм вычисления рекомендованного балла не прозрачен, а также строго фиксирован. При этом у преподавателя может существовать потребность в более точной настройке каждого из критериев оценивания при вычислении рекомендуемого балла. Кроме того, дополнительные проверки, речь о которых шла в предыдущем разделе не будут учитываться при

текущем механизме подсчета рекомендуемого балла за решение. В описанных обстоятельствах одним из самых функциональных решений является добавление возможности описания уникальной функции вычисления рекомендуемого балла для каждого из заданий и для курса в целом, в которой преподаватель может оперировать простейшими математическими операторами и нормированными результатами каждой из проверок.

Анализ плагиата является одной из наиболее востребованных функций оригинальной платформы [5; 6; 13; 17; 18], которая, действительно, обладает большим потенциалом и показывает высокий уровень практической применимости. Несмотря на то, что анализ плагиата может, фактически, применяться для любых практических курсов по программированию, довольно часто преподавателю необходимо проделать большое количество работы по исследованию результатов анализа плагиата, прежде чем становится возможным делать какие-либо весомые выводы.

Так, решением может быть улучшение существующих процессов анализа плагиата путем интеграции с новейшими сервисами [3; 4; 8; 11; 12; 14; 16], расширяющими существующие возможности платформы. Одним из новейших открытых сервисов анализа плагиата является GitPlag [19], разработанный в 2019 году в университете ИТМО. Сервис предоставляет более широкие возможности по настройке анализа плагиата, большее число анализаторов плагиата, а также более подробные результаты анализа плагиата.

1.4.3 Визуализация

Одним из самых важных элементов системы является то, каким образом в ней представляется информация. Flaxo решает большое количество задач автоматизации, одной из ключевых является анализ плагиата решений обучающихся. Оригинальная версия платформы включает в себя простейшую визуализацию результатов анализа плагиата, которая отображает количество обнаруженных случаев заимствования, а также значения процента общего кода в

самом ярком из найденных случаев. Кроме того, цветовая индикация позволяет выделить решения, значения процента совпадающего кода которых с одним из других решений, превышает некоторый заданный порог.

Подобное отображение результатов анализа плагиата позволяет выделять наиболее яркие случаи плагиата, однако не дает четкой картины недобросовестных заимствований на уровне всей образовательной группы, а также не позволяет определить группы связанных обучающихся.

Существуют различные подходы к визуализации плагиата, в их числе: табличное представление пар связанных решений, гистограммы распределения процента заимствованного кода, а также графовое представление. На основе существующих решений задачи визуализации результатов анализа плагиата можно прийти к выводу, что одними из самых универсальных в этой области являются инструменты графового отображения заимствований.

В связи с описанными обстоятельствами для улучшения наглядности результатов анализа плагиата, разумным является создание интерактивного инструмента визуализации графа заимствований, который бы позволял преподавателю увидеть целостную картину плагиата в рамках всего практического курса, а также персонализировать отображение графа.

1.4.4 Развертывание

Как было показано ранее, оригинальная система имеет слабо автоматизированный процесс сборки и развертывания системы. Однако, одной из характеристик современного open source продукта являются максимально автоматизированные сборка, публикация и развертывание. Это позволяет клиентам продукта очень легко его запускать или обновлять до последних версий.

С целью оптимизации процесса технической поддержки системы Flaxo ее сборка, публикации и развертывание должны быть оптимизированы и доведены до автоматического режима работа.

Задача сборки в настоящий момент решается в полной степени парой инструментов Gradle и NPM, которые позволяют крайне удобным и прозрачным образом собирать, тестировать и запускать все приложения системы Flaxo. При этом задача развертывания на текущий момент решается мануально и, как было описано ранее, для развертывания системы на локальном сервере необходимо вручную устанавливать множество зависимостей уровня операционной системы.

В качестве решения описанной проблемы возможно использование контейнеризации на основе инфраструктуры Docker, которая позволяет независимым образом описывать конфигурацию каждого из приложений, а затем запускать их единым образом, используя технологию Docker compose. Собранные образы всех приложений могут быть загружены на общедоступный портал Docker Hub, что позволит любому исследователю запустить и использовать всю инфраструктуру платформы Flaxo лишь несколькими командами в консоли своего локального сервера.

Помимо контейнеризации всей инфраструктуры платформы, одним из важных элементов, влияющих на качество развития системы является настройка процесса, т.н. непрерывного развертывания. Под непрерывным развертыванием понимается автоматизация процесса развертывание приложения в тестовое или реальное окружение.

Конфигурация непрерывного развертывания позволяет замкнуть круг, связывающийся весь процесс разработки, сборки, развертывания, функционирования и обратной связи в непрерывной процесс развития платформы. Подобный уровень автоматизации убирает все инфраструктурные преграды и позволяет максимально автоматизировать процесс разработки всей системы.

1.4.5 Документация

Развитие педагогического и технического бизнес-процессов начинает приносить результат только если конечный пользователь может использовать улучшения. Для этого пользователю необходимо обладать исчерпывающим

понимаем каждого бизнес-процесса. Чтобы добиться подобного состояния, необходимо использовать пользовательскую документацию, к которой в любой момент времени может вернуться исследователь.

Пользовательская документацию должна быть исчерпывающей по каждому из элементов, с которым при использовании системы сталкивается преподаватель. Самая минимальная версия пользовательской системы должна включать следующие разделы:

- краткое описание системы в целом,
- описание компонентов документации,
- описание процессов развертывания системы,
- определение базовых понятий платформы,
- описание доступных проверок,
- примеры базовых пользовательских сценариев.

1.5 Ключевые задачи

На основании приведенных выше основных направлений развития и перечисленных условий их успешности, был сформирован ряд непосредственных задач, которые необходимо решить в рамках настоящей выпускной квалификационной работы.

Приведенные ниже задачи по развитию системы Flaxo направлены на оптимизацию рассмотренных бизнес-процессов и позволяют улучшить возможности практического применения системы Flaxo для практических курсов по программирования для любых организаций. Ключевых задачи, сформированные для решения в рамках настоящей выпускной квалификационной работы, приведены ниже:

- Расширение функций платформы.
- Персонализация возможностей по настройке курсов.
- Поддержка графовой визуализации результатов анализа плагиата.

- Улучшение механизмов развертывания приложений платформы.
- Добавление пользовательской документации, включающей описание как стадии развертывания, так и каждого из элементов функционала платформы.
- Внедрение и проверка обновленных и измененных бизнес-процессов.

2 ПРОЕКТИРОВАНИЕ ИЗМЕНЕНИЙ

После того как ключевые задачи настоящей работы сформированы, следует этап проектирования отдельных элементов по каждому из основных направлений развития системы Flaxo.

Этап проектирования предусматривает рассмотрение современных подходов и технологий, для решения конкретных задач в соответствующих основным направлениям развития областях.

Первым делом будет описан графовый инструмент визуализации плагиата, который призван увеличить общую интерпретируемость результатов анализа плагиата. Вторым пунктом речь пойдет об улучшении непосредственно механизма анализа плагиата. Далее, будут рассмотрены улучшения основной платформы. После этого будет описана модель развертывания системы Flaxo. И в завершении главы будет рассмотрен вопрос о пользовательской документации системы, ее сборке, представлении и наполнении.

2.1 Графовая визуализация плагиата

Оригинальная система Flaxo обладает примитивным способом визуализации результатов анализа плагиата: для каждого из обучающихся указывается количество найденных заимствований, а также максимальное значение процента общего кода среди всех найденных заимствований. При этом, если значение последнего превышает некоторый заданный порог, то в таком случае показатели данного обучающегося выделяются красным цветом, указывая на высокую вероятность недобросовестного заимствования им чужого кода.

Описанный способ визуализации решает свою задачу - он позволяет доступным образом указать преподавателю на работы, которые необходимо проверить в первую очередь, а какие с меньшей вероятностью обладают заимствованиями. Однако, подобная система визуализации не может дать преподавателю полной картины происходящего, поскольку она не указывает на

закономерности, которые определяются более чем двумя работами. Действительно, механизм анализа плагиата предполагает попарное сравнение всех работ обучающихся, поэтому выявление небинарных зависимостей может оказаться непростой задачей. При этом, подобные зависимости, могут выявлять, например, группы недобросовестных обучающихся, которые делятся друг с другом своим исходным кодом. При этом, если похожие заимствования имеются более чем у двух людей, то это значительно уменьшает вероятность ложного срабатывания системы анализа плагиата.

Одним из возможных способов представления данных, позволяющим наблюдать сложные зависимости, является графовая визуализация. На сегодняшний день существует, как минимум один инструмент графовой визуализации результатов анализа плагиата. Как показано в связанной статье, использование графов для отображения результатов анализа плагиата наравне с гистограммами распределения процента заимствованного кода, повышает эффективность применения системы анализа плагиата в целом. Кроме того, использование наложения графовых представлений нескольких заданий в практическом курсе позволяет свести на нет влияние ложных срабатываний алгоритма анализа плагиата. При таком подходе становится возможным выявлять группы обучающихся, которые стабильно выделяются на фоне своих коллег и тем самым, вероятнее всего, действительно недобросовестно используют чужой исходный код.

Использование графов для отображения результатов анализа плагиата даже одного задания в отдельности, с точки зрения наглядности, имеет очевидные преимущества перед табличным представлением.

2.1.1 Граф плагиата

Упомянутый выше существующий инструмент визуализации плагиата делает большой шаг вперед к простоте интерпретации результатов анализа плагиата. Однако, система не позволяет динамически менять параметры,

определяющие структуру графа, что ограничивает преподавателя в возможности наглядного рассмотрения зависимостей в различных перспективах и с отличающимся масштабом. По этой причине в рамках связанной работы был создан универсальный инструмент интерактивной визуализации графов.

Инструмент рассматривает обучающихся как вершины графа, а случаи заимствования как ребра графа, соединяющие двух различных обучающихся. В качестве веса ребра w используется значение процента соответствующего заимствования, которое лежит в полуинтервале (0,100]. Учитывая, что большие веса соответствуют меньшим расстояниям между двумя решениями в пространстве всех возможных решений некоторой задачи, необходимо в качестве длин ребер l использовать значения обратно пропорциональные их весу. При таком подходе на построенном графе можно визуально различать кластеры, каждый из которых определяет группу обучающихся, работы которых содержат значительно выделяющееся количество заимствований друг между другом. В случае отсутствия ярко выделенных кластеров, можно говорить о том, что примеров плагиата в работах найти не удалось или о том, что необходимо изменить параметры отображения графа.

В рамках проведения двух практических курсов по программированию на языке Java, включающем в общем сложности 12 заданий, с использованием оригинальной платформы было отмечено следующее явление: норма распределения процентов заимствованного кода для каждого задания уникальна и может значительно отличаться. При этом низкий процент заимствованного кода в части заданий с нормой распределения, смещенной к нулю, не обязательно означает сниженный уровень заимствования кода в этом задании.

Предположительно, в описанных случаях имеет смысл нормализовать веса, чтобы они укладывались в интервал [0, 1], причем правой границей должен соответствовать максимальный вес ребер из представленных. Для некоторых заданий, однако, подобный подход может оказаться непригодным и стоит нормализовать веса так, чтобы правой границе интервала соответствовали ребра с

весом равным 100. Следовательно, инструмент должен иметь возможность поддержки нескольких типов нормализации.

Дополнительный интерес представляет возможность пороговой нормализации, которая бы позволяла объединять в кластеры только таких обучающихся, у которых как минимум один из процентов заимствованного кода выше некоторого заданного порогового значения r.

Помимо прочего, существует ряд случаев, ведущих к необходимости как относительного, так и абсолютного масштабирования длин ребер графа. К примеру, при отключенной нормализации весов задания со смещенной нормой распределения процентов заимствованного кода влево будут представлены графом с очень короткими ребрами, на котором крайне сложно разобрать отдельные связи. В этом случае существование относительного масштабирования позволит сохранить пропорции длин ребер и при этом сделать их различимыми. В другом случае, при очень высоких значениях весов ребер, их длины стремятся к нулю, отчего вершины графа могут накладываться друг на друга и становиться трудноразличимыми. Возможность абсолютного масштабирования позволяет справиться с этой ситуацией, путем добавления некоторой фиксированной базовой длины всем ребрам.

2.1.2 Инструмент визуализации

Существует значительное количество инструментов для отображения графовых структур. Они отличаются интерфейсом, количеством базовых структур, документацией и инструментами разработки. Так, одним из возможных к использованию решений является библиотека data2viz, написанная на языке Kotlin и являющаяся адаптированной версий оригинальной библиотеки d3.js. Библиотека предоставляет возможность использовать создавать различного рода визуализации.

В связи с тем, что библиотека написана на языке Kotlin, обладающем широкой поддержкой интерактивными средствами разработки, то написание приложений с использованием data2viz является крайне прагматичным.

По описанным причинам разработка инструмента графовой визуализации, названного Data2Graph, производится с использованием упомянутой ранее библиотеки.

2.1.3 Интеграция инструмента

Отдельной задачей стоит интеграция инструмента визуализации с существующей инфраструктурой системы Flaxo. Сервис, являясь частью системы, должен быть достаточно независим для использования и в других сферах исследований, помимо визуализации результатов анализа плагиата.

Описанные требования накладывают определенные ограничения на архитектуру инструмента Data2Graph. Так, инструмент должен напрямую отправлять запросы на сервис.

Таким образом поддержать независимость инструмента является возможным, если ввести формальную структуру данных, которые инструмент получает на вход, а также определить адрес конечного сервера, отдающего эти данные, входным параметром.

Общая схема взаимодействия двух сервисов (Рисунок 3) должна происходить в несколько шагов. При запросе от пользователя на отображение результатов анализа плагиата серверная часть системы flaxo должна генерировать временную уникальную ссылку, по которой должен быть доступен результат анализа плагиата в виде, удовлетворяющем требованиям инструмента графовой визуализации. После того, как подходящая ссылка создана, серверная часть должна вернуть ее клиентской части приложения, которая начнет инициализировать графовый инструмент. По мере инициализации инструмент использует сгенерированную ссылку для получения графовой структуры и впоследствии сможет ее отобразить.

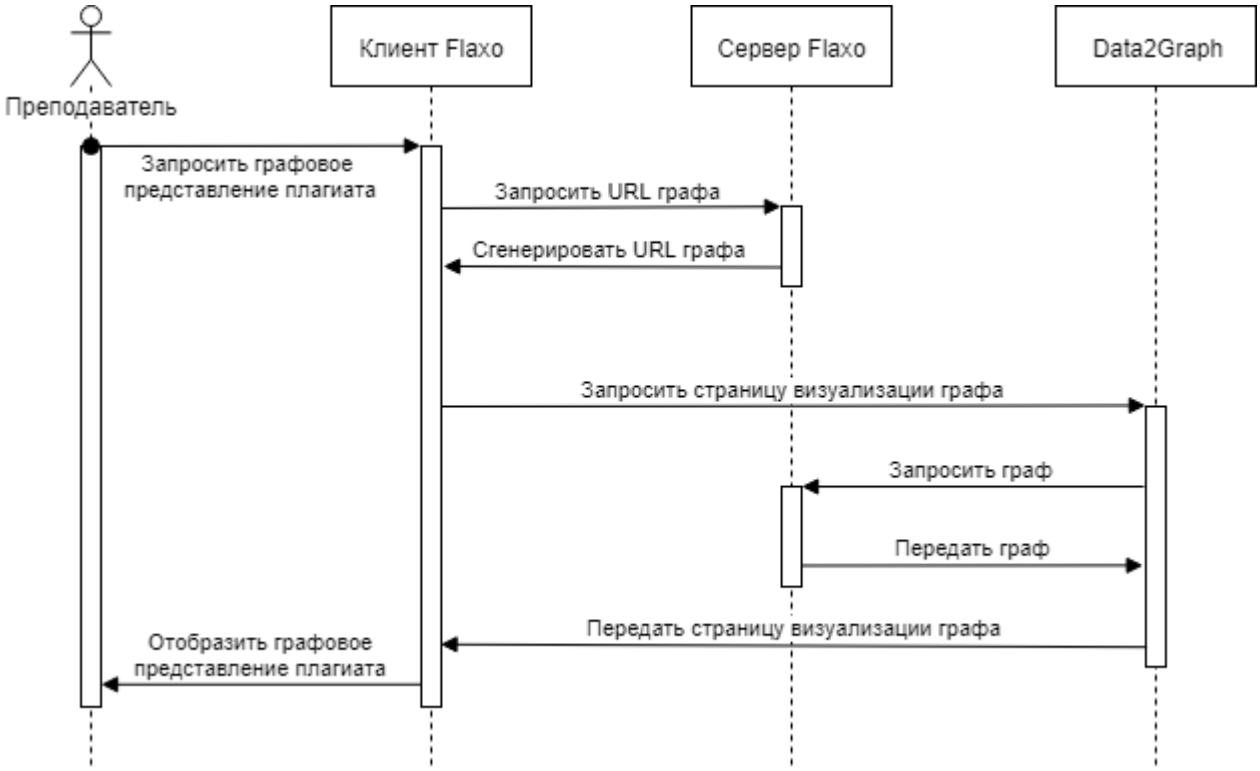

Рисунок 3 – Диаграмма последовательности работы инструмента графовой визуализации

2.2 Механизм анализа плагиата

Механизм анализа плагиата, существующий в системе Flaxo, является одной из самых важных составляющих всей платформы, которая позволяет преподавателям классифицировать решения на оригинальные и заимствованные. Тем не менее, анализ плагиата является сложным процессом, результат которого может сильно отличаться от одного практического курса к другому и даже от одного практического задания к другому.

Так, одной из задач настоящей работы, является развитие непосредственно механизма анализа плагиата. Наиболее весомым шагом в сторону его улучшения является интеграция с сервисом новейшим сервисом анализа плагиата GitPlag.

2.2.1 Интеграция GitPlag

Открытый сервис GitPlag, разработанный в 2019 годом нашим коллегой из Университета ИТМО, позволяет проводить анализ репозиториев открытых систем

контроля версий на плагиат и предоставляет синхронный и асинхронный программные интерфейсы.

Использование сервиса GitPlag открывает широкий спектр дополнительных возможностей по персонализации анализа плагиата системой Flaxo. Так, сервис позволяет использовать для каждого анализа один из трех доступных на сегодняшний день анализаторов плагиата:

- MOSS,
- JPlag,
- Combined.

В отличие от того, что предоставляет GitPlag, оригинальная система Flaxo имеет встроенную поддержку только одного анализатора плагиата, инструмента MOSS. Этот инструмент обладает рядом преимуществ перед другими анализаторами, однако не лишен своих недостатков.

Одним из новых анализаторов плагиата, поддержка которых появляется при интеграции GitPlag в инфраструктуру Flaxo, является JPlag, который имеет отличный от MOSS механизм работы, а также распространяется иным способом.

Существование другого независимого анализатора плагиата, который преподаватель может выбрать при изучении решений практического курса по программированию, позволяет расширить спектр применимости анализа плагиата. Теперь, в случаях, когда анализатор MOSS по той или иной причине мог работать недостаточно точно, можно использовать другой анализатор плагиата JPlag, который может вести себя иначе в тех же самых условиях.

Кроме того, сервис GitPlag предоставляет дополнительный синтетический анализатор плагиата Combined, который не является независимым инструментом как таковым, но является обобщением результатов работы друх других анализаторов – MOSS и JPlag. Подобное агрегирование результатов работы других анализаторов позволяет снизить общий риск ложных срабатываний и повысить общий уровень точности анализа плагиата.

В связи с тем, что каждый из трех анализаторов плагиата обладает теми или иными достоинствами и недостатками, выбор единственно оптимального не представляется возможным. Именно поэтому интеграция с сервисом GitPlag должна предоставлять преподавателю возможность использовать один из трех анализаторов плагиата при исследовании решений обучающихся на плагиат.

Другими преимуществами использования сервиса GitPlag является усовершенствованная система загрузки и кэширования файлов, которая позволяет увеличить общую скорость анализа плагиата.

Помимо прочего, поскольку сервис GitPlag целиком посвящен анализу плагиата и позволяет вынести часть функционала оригинальной платформы в отдельный сервис, что согласуется с принципом единой ответственности, то инфраструктура системы Flaxo должна целиком включить сервис GitPlag и его зависимости.

Подобное поглощение должно положительно повлиять на общую стабильность работы всей платформы, согласно все тому же принципу единой ответственности.

Сервис GitPlag распространяется в контейнеризованном виде, который естественным образом может быть использован почти в любых окружениях. Таким образом, его интеграция с проектируемым процессом развертывания (Рисунок 4) становится тривиальной. Более подробно процесс развертывания системы Flaxo, включающий сервис GitPlag со всеми его компонентами, описан ниже в настоящей главе.

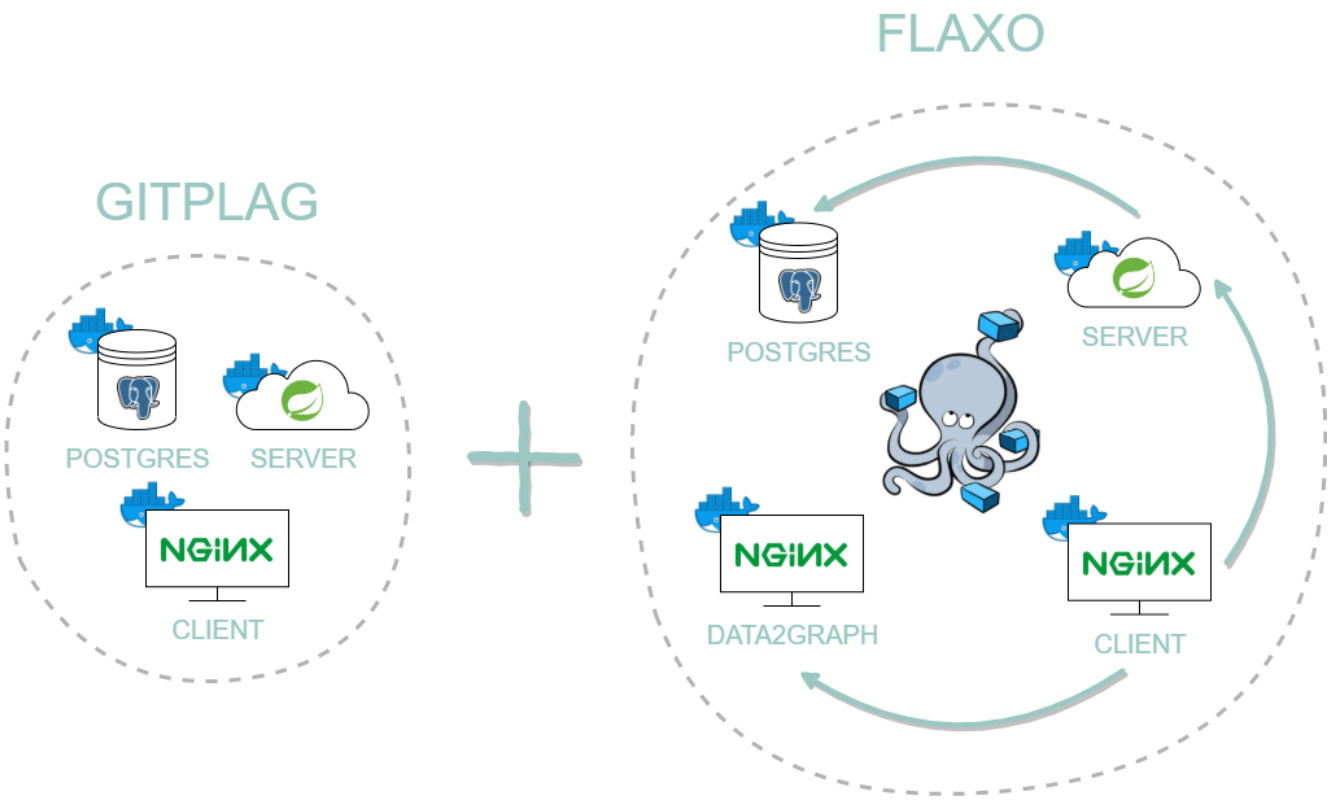

Рисунок 4 – Схема интеграции GitPlag в инфраструктуру Flaxo

2.2.2 Предобработка данных

Помимо интеграции с новейшими сервисами анализа плагиата часть задач по развитию механизма анализа плагиата платформы упирается в еще один основной этапа, а именно этап предобработки данных. Так, в процессе исследования процесса функционирования системы на практике, стало ясно, что в ряде случае анализатору плагиата на вход могут подаваться лишние файлы, использование которых может не являться для преподавателя очевидным.

В качестве решения проблемы с использованием анализатором плагиата лишних файлов, а также проблемы недоиспользования нужных файлов, принято решение создать дополнительную опцию для конфигурации паттерна имен файлов, который будет использоваться для фильтрации файлов. Новый паттерн может быть задан пользователем в виде регулярного выражения, которое по умолчанию будет соответствовать всем известным расширениям файлов выбранного языка программирования.

2.2.3 Индикация результатов

Одним из недостатков, выделенных в процессе исследования оригинальной системы, было отсутствие индикации результатов последнего анализа плагиата. Отсутствие подобной информации не позволяло преподавателю понять, необходим ли анализ плагиата в текущий момент времени и появились ли новые решения с момента последнего анализа плагиата.

Добавление индикации результатов последнего анализа плагиата с отдельным указанием на появление или отсутствие новых решений является значительным дополнением к уже существующей визуализации результатов анализа плагиата.

2.3 Платформенные изменения

Часть из обозначенных ключевых задач настоящей работы относятся непосредственно к функциональности оригинальной платформы, поэтому для удобства их описания они выделены в отдельную группу, описание которой приведено в ниже настоящем разделе.

2.3.1 Система оповещений

Одной из задач по оптимизации системы, обозначенных в первой главе, является решение проблемы получения обратной связи обучающимися. Другими словами, задача заключается в том, чтобы обеспечить обучающихся некоторым автоматизированным откликом на их действия.

Так в рамках решения поставленной задачи необходимо разработать одностороннюю систему оповещений обучающихся (Рисунок 5), которая позволит доносить до них информацию о некоторых процессах, происходящих внутри системы и ранее недоступную ни для кого, кроме преподавателя.

Например, в случае выставления преподавателем баллов обучающимся за задания важным является оповестить каждого студента в отдельности о произошедших изменениях, повысив тем самым открытость системы для всех участников образовательного процесса и сняв груз этой задачи с плеч преподавателя, который должен был ранее делать это вручную.

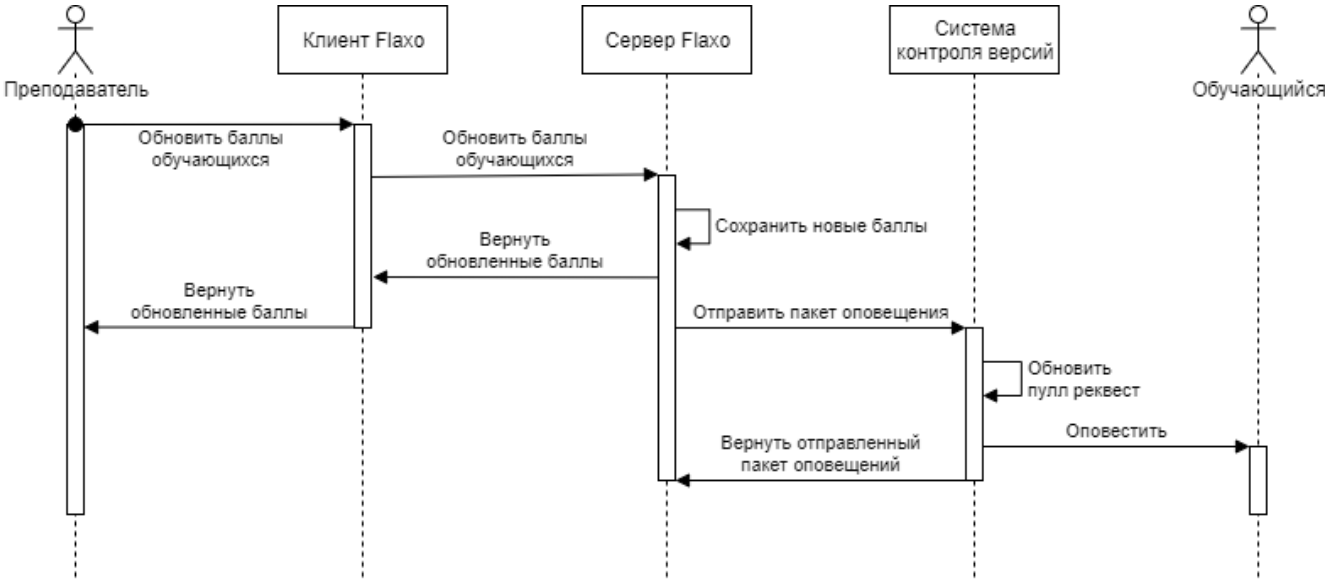

Рисунок 5 – Диаграмма последовательности функционирования системы оповещения

2.3.2 Системные события

Помимо необходимости существования системы оповещений крайне важным является мониторинг и обработка фоновых, автоматизированных системных событий. Как показал опыт исследования функционирования платформы Flaxo на практике, некоторые системные события тянут за собой череду рутинных задач, которые преподаватель должен проделывать каждый раз.

В связи с тем, что системные события могут быть зарегистрированы и обработаны самой системой, практичным может являться вариант, при котором все системные задачи регистрируются в некоторой единой очереди (Рисунок 6), которая в нужный момент может активировать эти события вместе со всеми необходимыми связанными задачи, которые преподавателю необходимо было ранее решать самостоятельно.

Так, одним примером системных событий может служить дедлайн задания в практическом курсе по программированию. является в случае выставления преподавателем баллов обучающимся за задания важным является оповестить каждого студента в отдельности о произошедших изменениях, повысив тем самым открытость системы для всех участников образовательного процесса и сняв груз этой задачи с плеч преподавателя, который должен был ранее делать это вручную.

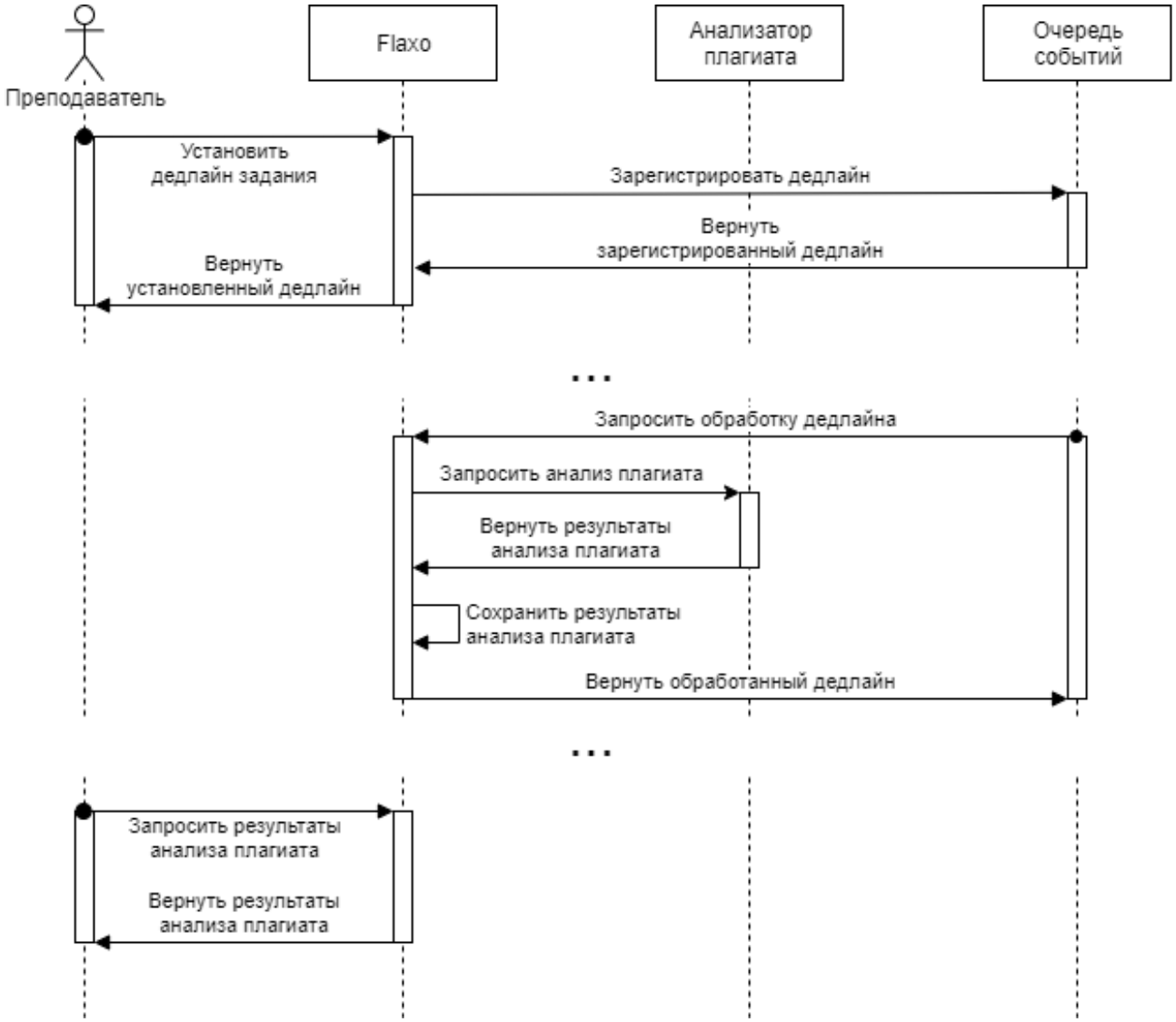

Рисунок 6 – Диаграмма последовательности функционирования системных событий

2.4 Развертывание

Одной из ключевых задач настоящей работы, как было обозначено ранее, является оптимизация процессов развертывания системы. Далее в настоящем разделе следует описание способа решения задачи оптимизации развертывания применительно к оригинальной системе Flaxo.

В настоящий момент при проектировании процессов разработки крупных и многомодульных приложений широко используется подход CI/CD или непрерывная интеграция и непрерывное поставка [7; 9], который предусматривает широкую автоматизацию процессов тестирования и сборки всего приложения.

Подобный подход позволяет минимизировать задержу между изменением системы и временем, когда становится возможным использовать измененную систему в клиентском окружении. Кроме того, непрерывная интеграция и поставка позволяют максимизировать эффективность разработки, поскольку позволяют централизовать автоматизированное тестирование и сборку приложения.

В качестве подхода к развертыванию системы Flaxo принято использовать описанный подход непрерывной интеграции и поставки.

2.4.1 Автоматическое тестирование

Одним из элементов процесса непрерывной интеграции и поставки является автоматизированное тестирование. Оригинальная система Flaxo обладает достаточным набором модульных и интеграционных тестов, необходимых для налаживания процесса автоматизированного тестирования.

Существует значительное число сервисов для проведения автоматизированного тестирования. Одним из наиболее распространенных и функциональных является сервис Travis, позволяющий решать большое число других задач подхода CI/CD.

В частности, сервис Travis не влияет на непосредственный процесс сборки приложения, который будет описан далее в настоящей главе, однако позволяет отображать статус каждой новой версии приложения в используемом сервисе контроля версий.

2.4.2 Контейнеризация

Помимо автоматизированного тестирования CI/CD предусматривает налаживание процессов автоматизированной сборки и публикации приложения. Поскольку современные приложения зачастую состоят из большого числа разнородных частей или сервисов, широко распространенным является подход с использованием различного рода контейнеризации [10]. Подобная подход предусматривает упаковку всех частей приложения в независимые контейнеры, содержащие внутри себя помимо самого приложения всей необходимой инфраструктуры для его работы.

Работа таких контейнеров обеспечивается широко развитыми возможностями виртуализации современных операционных систем. В связи с механизмом работы, контейнеры, являясь низкоуровневой абстракцией, имеют в большинстве случаев скорость работы, незначительно отличающуюся от скорости работы основной операционной системы.

Контейнеризация позволяет избавить конечного пользователя приложения от необходимости конфигурации окружения. Кроме того, появляется возможность запуска приложения на любых операционных системах, поддерживающих тот же механизм контейнеризации.

Одним из наиболее широко используемых инструментов контейнеризации является Docker, который позволяет собирать, запускать и публиковать контейнеры приложений. Далее в настоящей главе под контейнером будет пониматься Docker-контейнер.

Для оптимизации процессов развертывания системы Flaxo принято использовать инфраструктуру и контейнеры инструмента Docker.

2.4.3 Сервисы

Оригинальная система Flaxo состоит из двух сервисов – веб-клиента и сервера. Каждый из них достаточно независим для выделения отдельного контейнера. Кроме того, база данных, являющаяся частью сервера оригинальной системы, заслуживает быть выделенной в отдельный контейнер.

Помимо существующих сервисов, инструмент анализа плагиата, который описан ранее в настоящей работе, является достаточно независимым сервисом для выделения в отдельный контейнер.

Кроме того, интеграция с сервисом GitPlag требует использования двух дополнительных контейнеров – веб-клиента и сервиса, которые должны также находиться в одной сети с остальными контейнерами инфраструктуры Flaxo.

Суммируя вышеописанное, оригинальная система Flaxo состоит из 6 сервисов достаточно независимых для выделения в отдельные контейнеры.

2.4.4 Развертывание

Одним из важных элементов инфраструктуры Docker является инструмент Docker Compose, позволяющий конфигурировать и управлять целым набором из связанных контейнеров. Инструмент вводит абстракцию над группой связанных контейнеров, в связи с чем количество необходимых действий для развертывания приложения значительно снижается.

Так, для системы Flaxo конфигурация Docker Compose кластера (Рисунок 7) заключается в настройке 4-ех контейнеров каждого из сервисов системы: сервера, базы данных, клиента, инструмента графовой визуализации.

Развертывание всей инфраструктуры системы Flaxo в новом окружения с использованием технологии Docker Compose сводится к единственной команде запуска и необходимости конфигурации нескольких системных параметров приложения, а также нескольких ключей авторизации некоторых интегрированных сервисов.

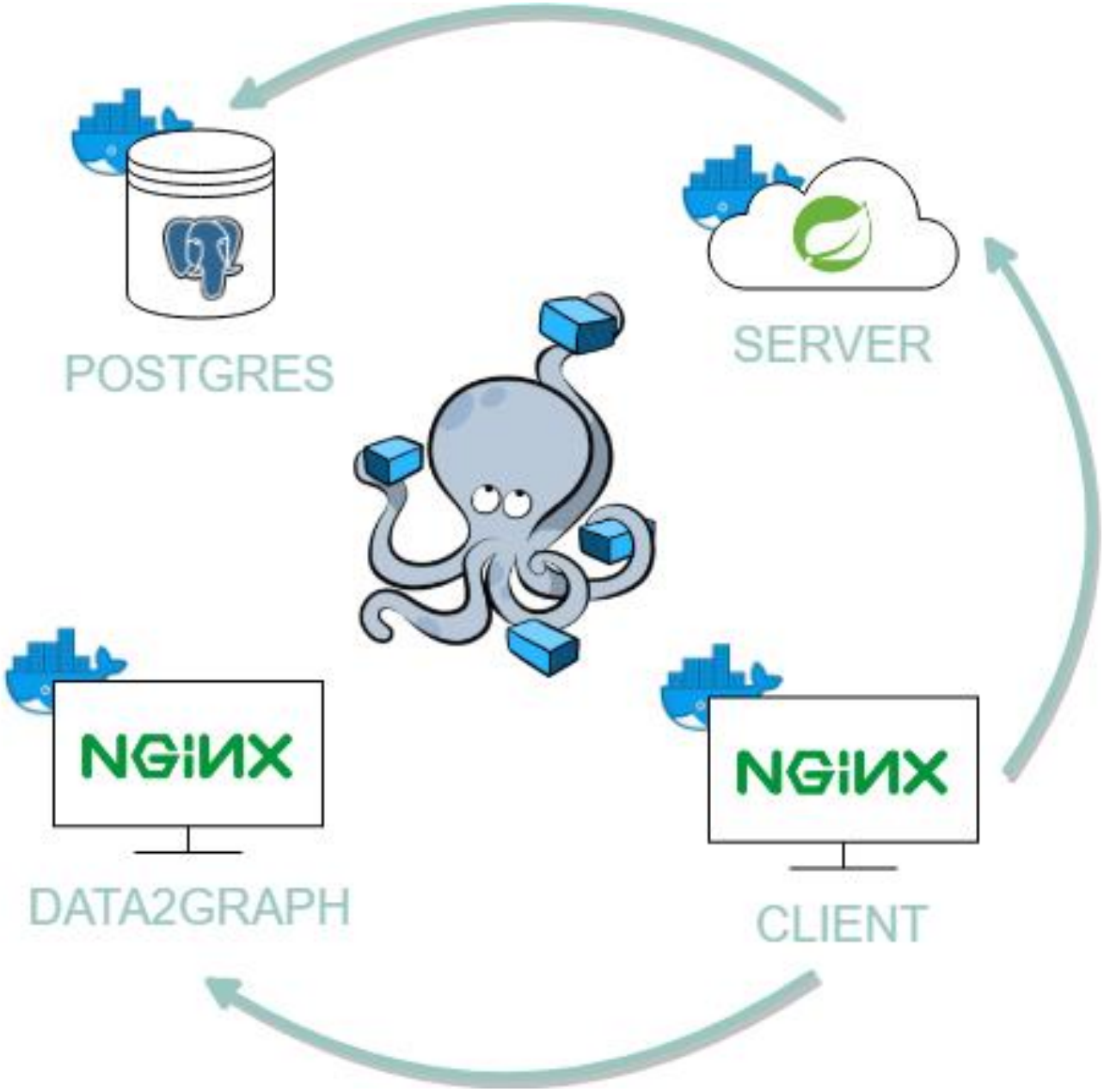

Рисунок 7 – Схема инфраструктуры системы Flaxo

2.4.5 Автоматические сборка и публикация

Инфраструктуры Docker также включает сервис DockerHub, который позволяет публиковать контейнеры, а также настраивать определенные процессы CI/CD. Так, одной из предоставляемых возможностей является возможность автоматической сборки версий контейнеров. Автоматизированная сборка запускается каждый раз, когда выпускается новая версия приложения в интегрированной системы контроля версий, в частности, на GitHub.

Следовательно, каждый раз при появлении новой версии Flaxo происходит автоматическая пересборка каждого из контейнеров системы. По завершении успешной сборки каждого из контейнеров публикацию приложения считается завершенной и приложение готов для использования пользователями.

2.4.6 Автоматическое развертывание

Помимо автоматических сборки и публикации системы Flaxo важным является также конфигурация автоматического развертывания приложения (Рисунок 8) в некотором тестовом или реальном окружении. На сегодняшний существует большое число сервисов, предоставляющих возможности по организации автоматических процессов сборки, публикации и развертывания. Одним из таких сервисов, является GitHub Actions, ставший доступным в 2019 году и успевший показать себя надежным и готовым к использованию средством для описания процессов сборки, публикации и развертывания.

Автоматическое развертывание срабатывает при каждом успешном завершении процессов автоматизированной сборки и публикации новых версий системы. Таким образом, настройка автоматического развертывания замыкает круг разработки, сборки, публикации, развертывания и сбора обратной связи, который определяет весь процесс развития системы.

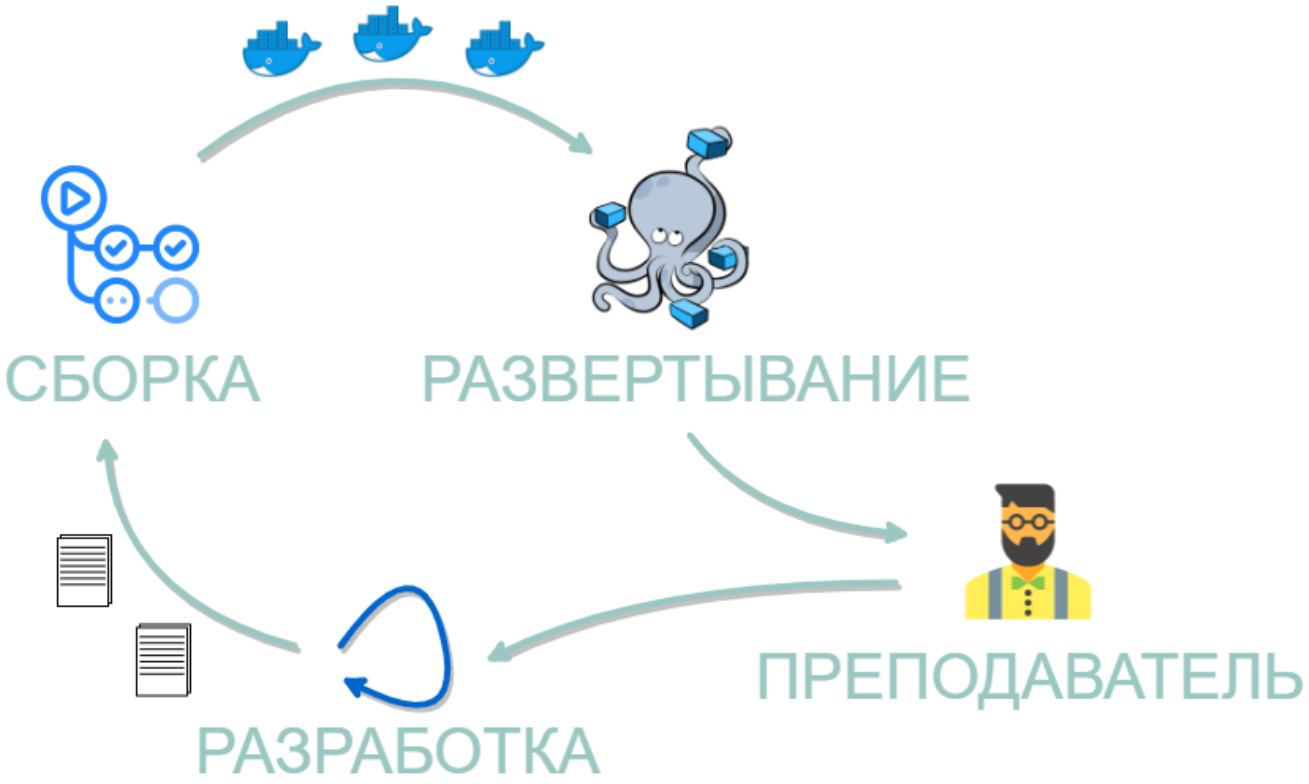

Рисунок 8 – Схема процессов автоматической сборки, публикации и развертывания

2.5 Документация

Последним направлением развития платформы является пользовательская документация, которая играет решающую роль, поскольку функциональность, не описанная в документации, зачастую, может быть просто недоступна для пользователя с точки зрения практического использования.

2.5.1 Автоматические сборка и публикация

В качестве инструмента для создания пользовательской документации могут использоваться различные движки. Одним из наиболее широко распространенных инструментов является mkdocs, позволяющий форматировать документацию, используя язык разметки Markdown, и компилировать ее в полноценную HTML-документацию.

Движок поддерживает множество пользовательских тем и предоставляет доступную поддержку для расширения. В качестве темы mkdocs в настоящей работе принято использовать тему material, требующую установку нескольких дополнительных пакетов.

Отдельный сервис readthedocs, использующий интеграции с системами контроля версий, в частности GitHub, и движок mkdocs, позволяет налаживать процессы автоматизированной сборки и публикации документации, а также ее версионирования.

Используя описанные технологии, является возможным создание и поддержка пользовательской документации для достаточно сложных информационных систем, в том числе системы Flaxo.

2.5.2 Содержание

Пользовательская документация системы Flaxo должна содержать, как уже было описано выше, описание всех основных концепций и бизнес-процессов системы, описание интегрированных с ней сервисов, авторизации и процесса развертывания системы.

3 ОПИСАНИЕ ВНЕДРЕННЫХ ИЗМЕНЕНИЙ

После того как все необходимые к выполнению задачи описаны и архитектура их решений спроектирована, следует этап их реализации и внедрения. Ниже в настоящей главе приводится описание реализованных и внедренных изменений в оригинальную систему Flaxo.

3.1 Графовая визуализация плагиата

Первым из изменений, описываемых в настоящей главе является создание инструмента графовой визуализации результатов анализа плагиата и интеграция его в инфраструктуру системы Flaxo.

3.1.1 Инструмент визуализации

Инструмент графовой визуализации представляет собой независимый сервис, позволяющий динамически отображать любые графовые структуры, соответствующие определенной модели. Исходный код графового инструмента визуализации, а также техническая документация размещены в открытом репозитории системы контроля версий по следующему адресу https://github.com/tcibinan/data2graph.

Сервис может свободно использоваться любыми исследователями для визуализации графовых структур. Для удобства развертывания сервис контейнеризован и доступен в виде единственного docker-контейнера. Кроме того, сервис может быть собран и запущен локально при необходимости.

В упакованной версии инструмента визуализации плагиата для отображения доступен стандартный тестовый набор данных, который может использоваться для исследования функциональности инструмента.

Пользовательский интерфейс инструмента графовой визуализации (Рисунок 9) является небольшим веб-приложением, которое можно использовать из любого

современного браузера. Основной интерфейс можно условно разделить на две основные части: рабочую область и панель управления.

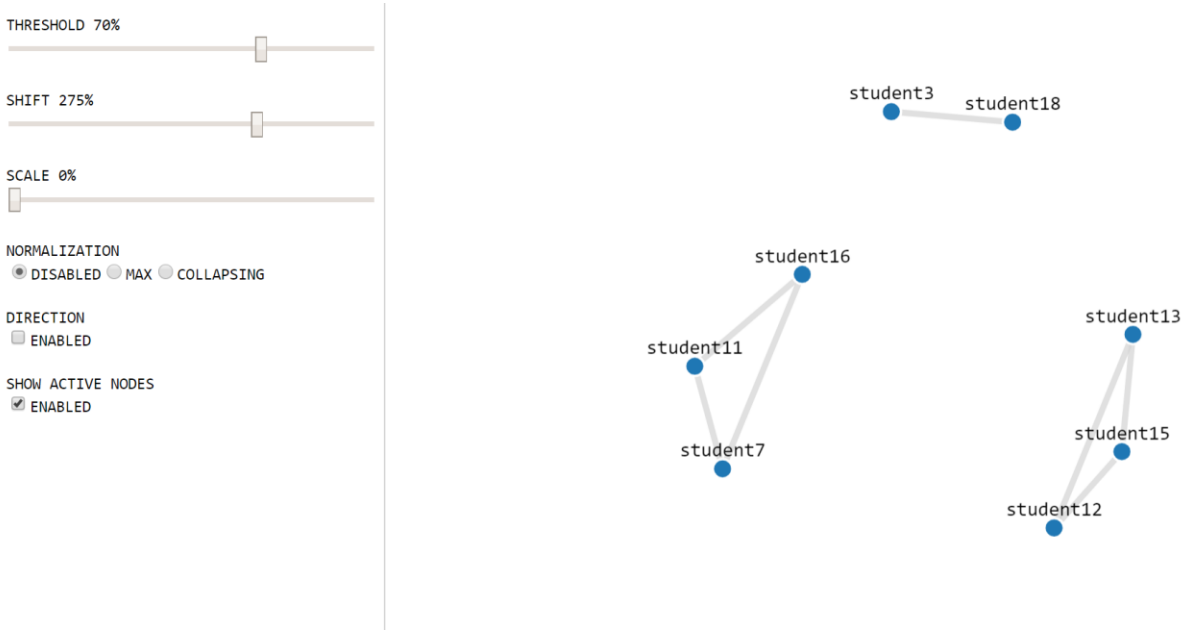

Рисунок 9 – Скриншот инструмента графовой визуализации

Рабочая область отвечает за отображение графа, а также является интерактивной и позволяет активировать некоторые события, привязанные к нажатию на вершины или ребра графа.

Каждая вершина графа отображается в виде небольшого круга с некоторым именем, а ребра отображаются в виде полупрозрачных линий. И вершины, и ребра графа являются интерактивными и при наведении пользователем курсора мыши меняют свой цвет и выделяются на графе. Кроме того, на наведенные вершины и ребра пользователь может кликнуть мышью, что повлечет за собой срабатывание некоторого привязанного действия.

Таким образом за отрисовку графа и обработку пользовательских действий по отношению к нему отвечает рабочая область. При этом задачи по конфигурации отображения графа лежат на панели управления, которая позволяет пользователю динамически конфигурировать всевозможные параметры отображения графа:

- Порог веса отдельного ребра, необходимый для его отображения. Параметр позволяет отображать отдельные подграфы на основании веса их ребер.
- Абсолютное смещение. Параметр позволяет увеличивать длину всех ребер графа на некоторую константную величину, не зависящую от их веса.
- Относительное смещение. Параметр позволяет увеличивать длину всех ребер графа на некоторую уникальную величину, зависящую от их веса.
- Тип нормализации веса. Параметр позволяет выбирать тип нормализации веса всех ребер графа.
- Отображение направлений ребер. Параметр контролирует отображение графа как ориентированного или нет.
- Отображение только связных вершин. Параметр контролирует отображение только связных вершин, т.е. таких, для которых отображается хотя бы одно ребро.

Подобное количество динамических параметров позволяет исследовать графовые данные с различными количеством вершин, количеством ребер, а также с различным распределения весов ребер. По умолчанию, порог веса ребра для отображения равен 75, абсолютное и относительные смещения равны 200, нормализация выключена, отображение направлений отключено, а также отображаются только связные вершины.

3.1.2 Интеграция инструмента

Инструмент графовой визуализации, являясь обособленным универсальным средством визуализации, на практике должен использоваться оригинальной системой Flaxo для улучшения пользовательского опыта преподавателя при его

работе с результатами анализа плагиата решений некоторого практического курса по программированию (Рисунок 10).

Для этого в системе Flaxo, позволяющей производить анализ плагиата решений обучающийся, появляется дополнительная возможность представлять результаты такого анализа в виде графа. Пользовательский интерфейс инструмента графовой визуализации встраивается в основной пользовательский интерфейс системы Flaxo в виде отдельного документа, заключенного в iframe. В итоге с точки зрения пользователя он продолжает работать на той же веб-странице, на самом деле взаимодействуя с отдельным сервисом.

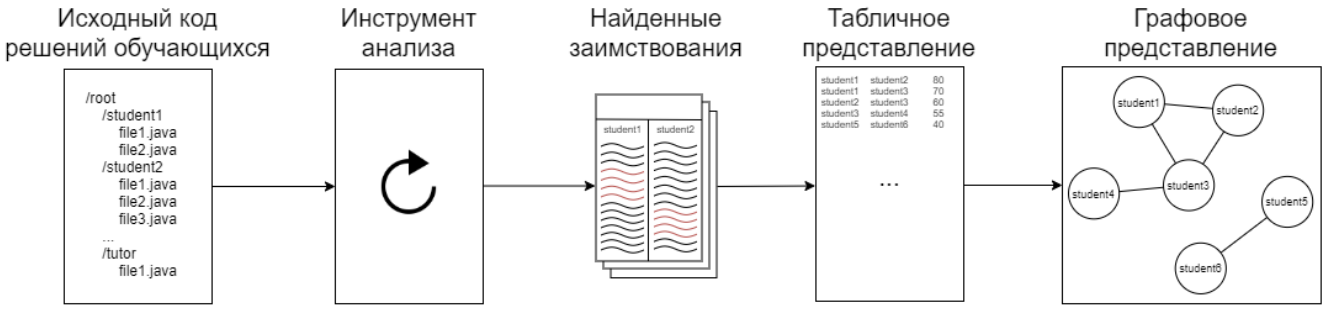

Рисунок 10 – Схема стадий анализа плагиата

3.2 Механизм анализа плагиата

Одной из ключевых задач настоящей работы является улучшение механизма анализа плагиата в системе Flaxo. На основании спроектированных изменений механизм анализа плагиата был в значительной степени преобразован. Ниже в настоящем разделе кратко описаны результаты внедрения связанных изменений.

3.2.1 Интеграция GitPlag

В частности, встроенный механизм анализа плагиата на основе инструмента MOSS был заменен на использование отдельного сервиса GitPlag (Рисунок 11), который берет на себя задачи по организации и проведению анализа плагиата исходного кода из распределенных репозиториев системы контроля версий.

Помимо прочего сервис GitPlag поддерживает интеграцию с разработанным инструментом графовой визуализации результатов анализа плагиата и предоставляет собственный пользовательский интерфейс, что расширяет возможности по конфигурации проводимых анализов плагиата.

Если раньше преподаватель мог на странице курса в системе Flaxo просмотреть отчет инструмента анализа плагиата, то теперь по тому же сценарию преподаватель может открыть интерфейс GitPlag, в котором помимо просмотра оригинального отчета инструмента анализа плагиата также доступен ряд дополнительных возможностей, как например, просмотр истории анализов плагиата некоторого репозитория.

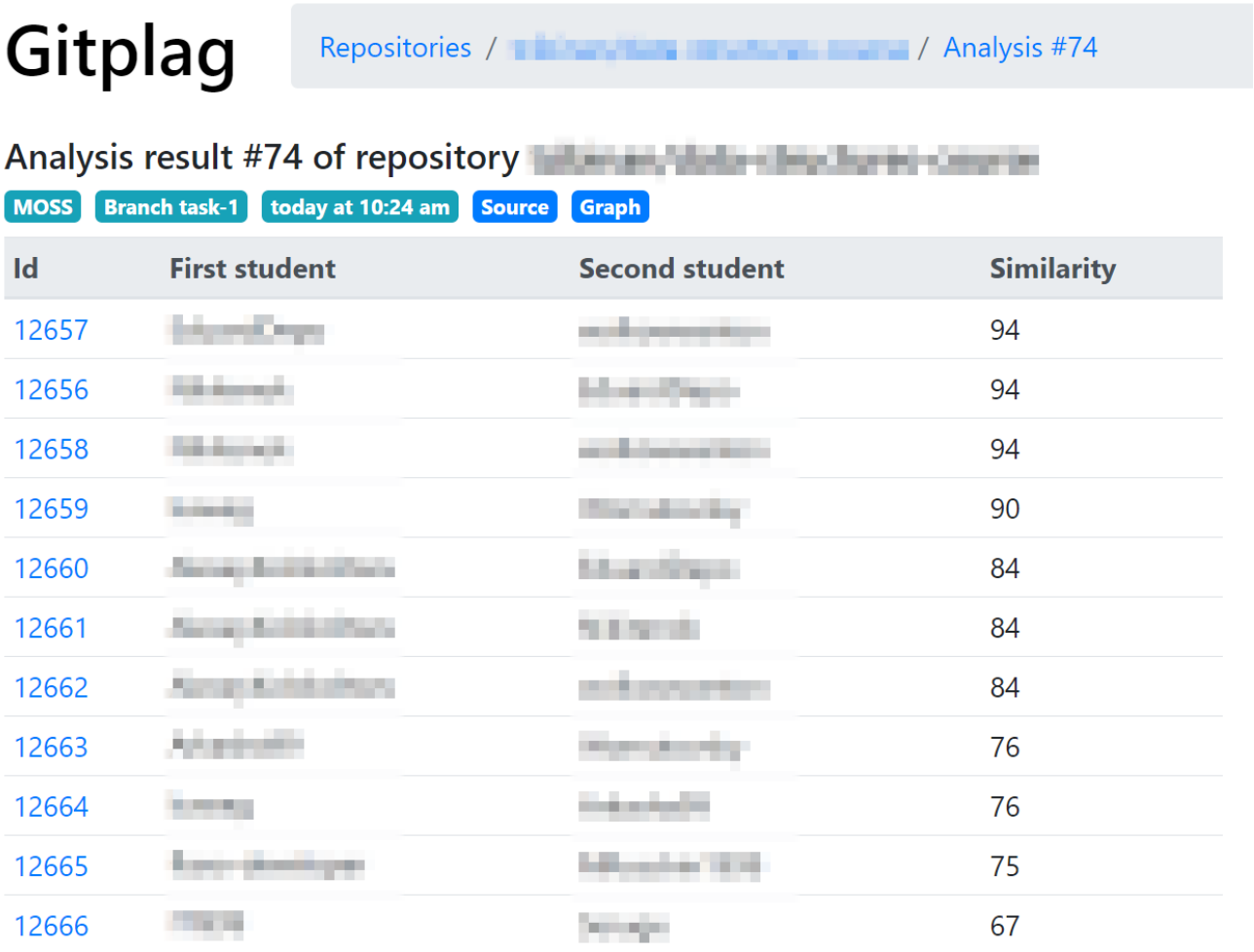

Рисунок 11 – Скриншот интегрированного сервиса GitPlag

3.2.2 Предобработка данных

Помимо изменения непосредственно инструмента анализа плагиата были также внедрены некоторые изменения, связанные с этапом предобработки данных. Так, например, с момента внедрения спроектированных изменений преподавателю стала доступна дополнительная настройка (Рисунок 12), которая позволяет сконфигурировать регулярное выражение, которое будет использоваться при фильтрации файлов, поступающих на вход инструмента анализа плагиата.

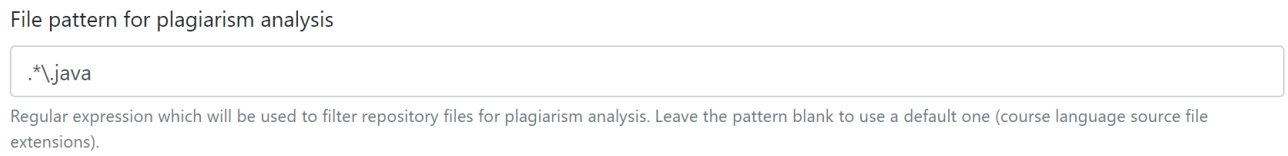

Рисунок 12 – Скриншот настройки паттерна файлов анализа плагиата

3.2.3 Индикация результатов

Кроме этапа предобработки данных было также обновлено отображение сводки по результатам анализа плагиата. Теперь преподаватель в любой момент времени может увидеть дату последнего анализа плагиата (Рисунок 13). Так, преподавателю не нужно запоминать, проведен ли уже анализ плагиата для этого задания или ему еще только предстоит начаться.

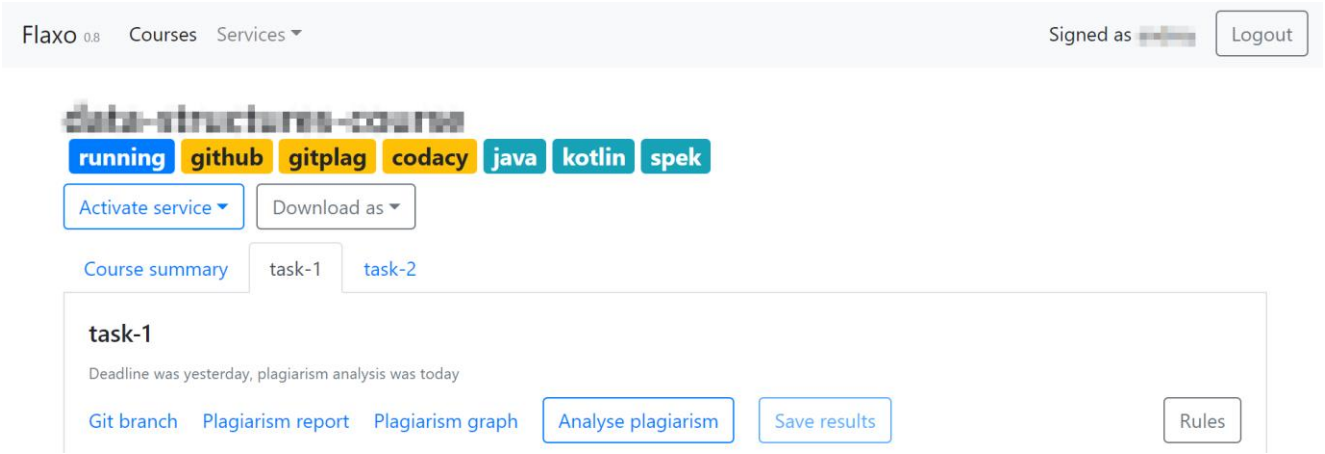

Рисунок 13 – Скриншот индикации времени последнего анализа плагиата

3.2.4 Анализ плагиата отдельных заданий

Отдельным внедренным улучшением также является добавление возможности анализа плагиата отдельных заданий практического курса. Прежде преподаватель мог запустить анализ плагиата всех заданий курса одновременно. Теперь стало возможным проведение независимых анализов плагиата заданий практических курсов.

3.3 Платформенные изменения

Ряд изменений был проведен в отношении непосредственно функций платформы. Например, были расширены возможности по автоматическому созданию инфраструктуры курса (Рисунок 14), добавлена возможность импорта курса на основании существующего репозитория в системе контроля версий (Рисунок 15).

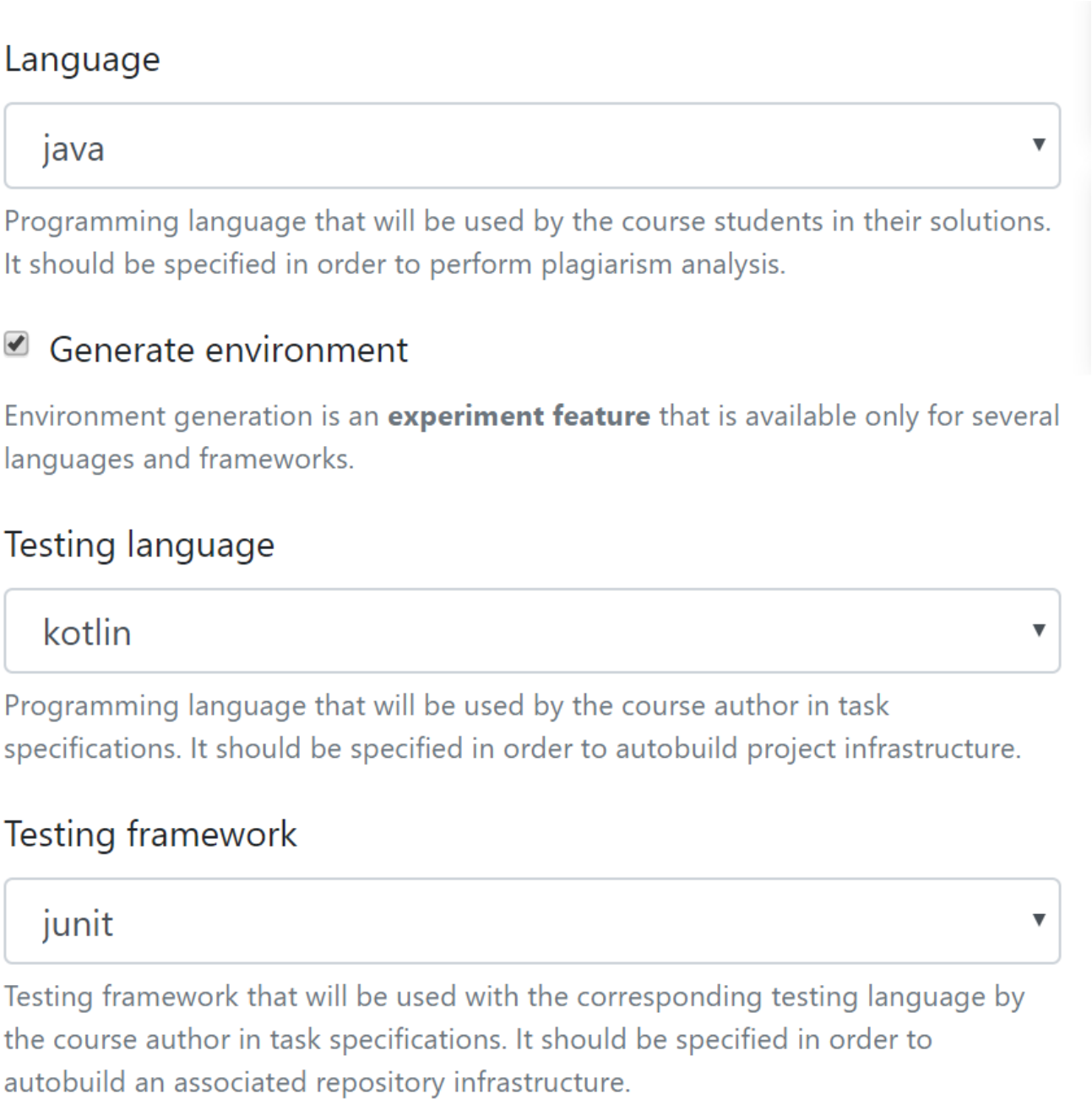

Рисунок 14 – Скриншот интерфейса генерации инфраструктуры курса

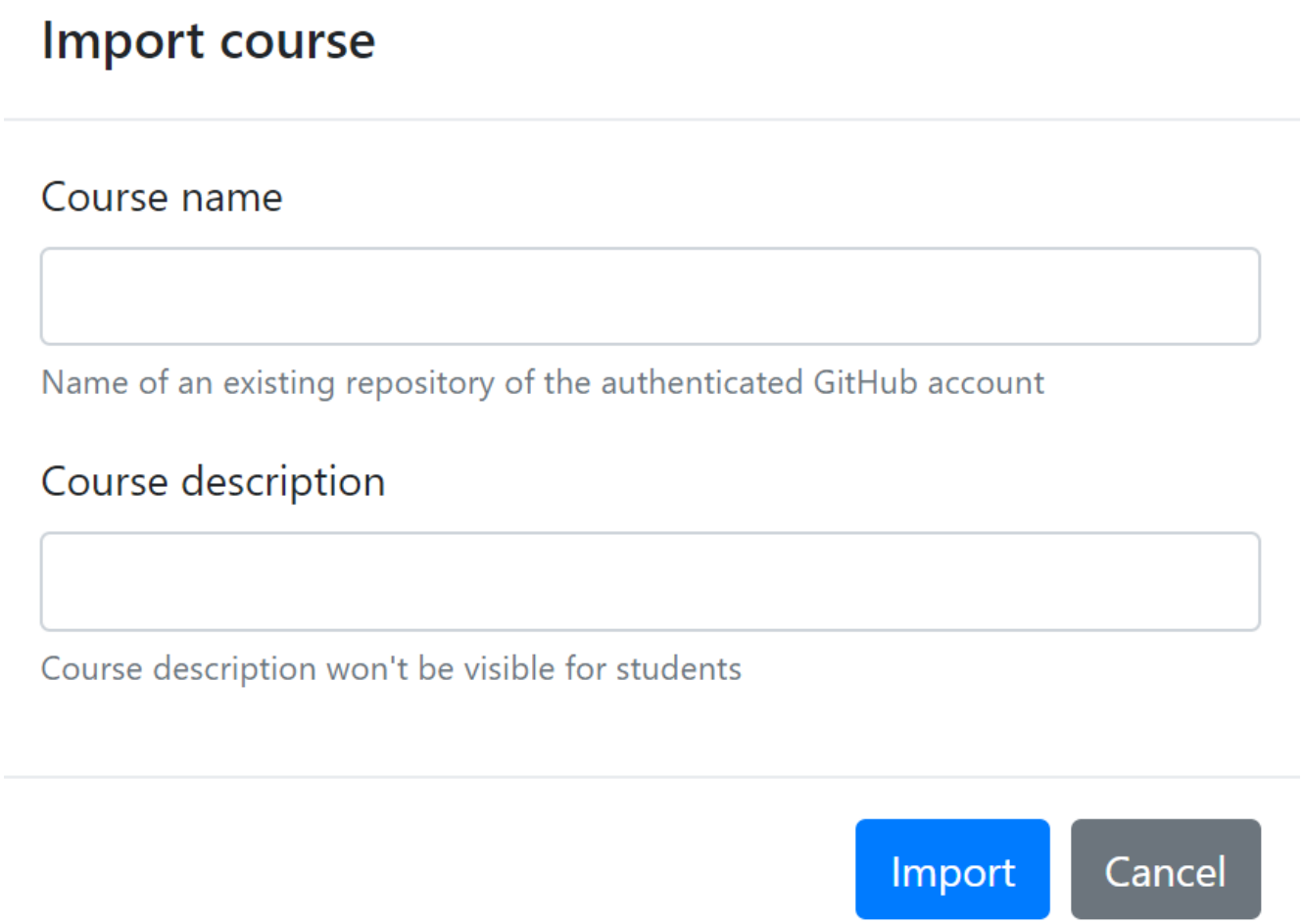

Рисунок 15 – Скриншот интерфейса импорт практического курса

3.3.1 Система оповещений

Отдельно можно выделить добавление поддержки системы оповещений, которая позволяет автоматически связывать некоторые действия преподавателя в системе с оповещением связанных обучающихся. Так, например, выставление баллов за решение обучающегося вызывает создание нового комментария в соответствующем пулл реквесте (Рисунок 16).

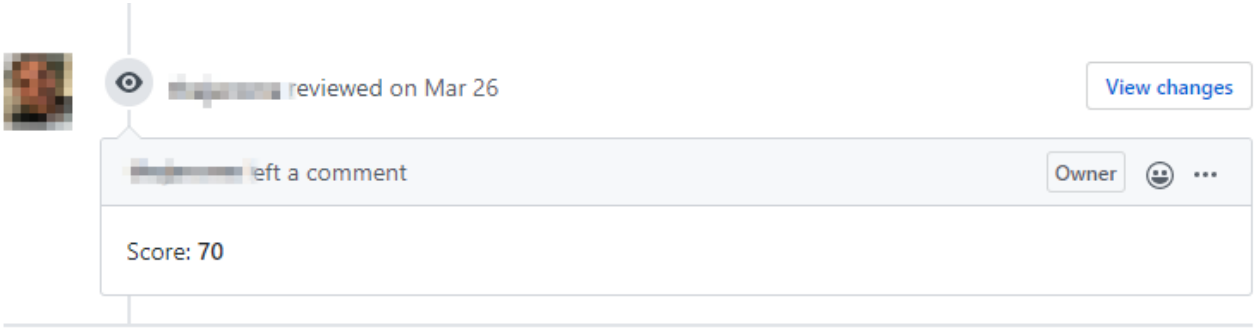

Рисунок 16 – Скриншот оповещения об изменении балла за решение

Содержание такого рода оповещений, сообщающих об изменении балла за решение, может быть изменено преподавателем путем использования соответствующей настройки курса (Рисунок 17). Кроме того, такие оповещения по умолчания отключены и для изменения этого необходимо также использовать дополнительную существующую настройку.

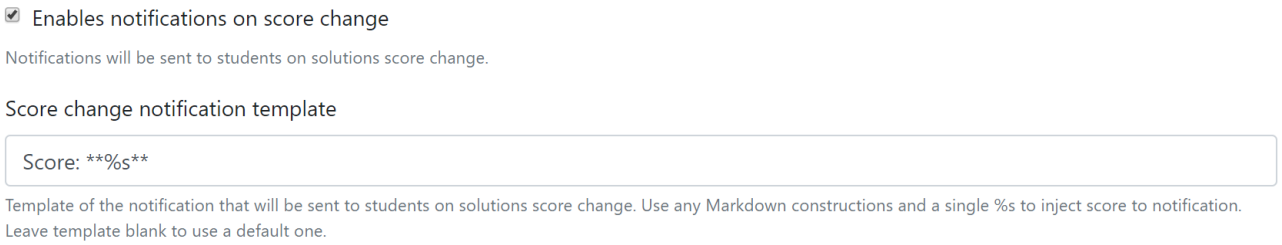

Рисунок 17 – Скриншот настроек оповещения обучающихся

3.4 Развертывание

Одной из ключевых задач настоящей работы, связанной непосредственно с функционированием технической поддержки системы Flaxo, является конфигурация процессов автоматической сборки, публикации и развертывания инфраструктуры платформы.

3.4.1 Контейнеризация

Первым шагов при переходе от мануального к автоматизированному развертыванию в настоящей работе является контейнеризация существующих элементов инфраструктуры для улучшения возможностей портирования все платформы.

Так, в рамках решения обозначенной задачи каждый из отдельных элементов системы был выделен в независимый контейнер со своими зависимостями. Каждый из сформированных контейнеров имеет минимизированный размер с целью уменьшения общего размера дистрибутива.

3.4.2 Сборка, публикация и развертывание

Процессы сборки, публикации и развертывания контейнеризованной системы Flaxo объединены в единую группу задач, которые автоматически выполняются сервисом GitHub Actions при любом обновлении исходного кода системы или релизе новой версии.

Так при обновлении исходного кода основной ветки разработки системы Flaxo, все контейнеризованные сервисы автоматически собираются и свободно публикуются на платформе DockerHub. Кроме того при релизе новой версии приложения происходит автоматическая сборка, публикация и развертывание всей инфраструктуры новой версии системы в некоторое тестовое окружение.

Конфигурационные файлы всех процессов сборки, публикации и развертывания опубликованы в основном репозитории системы на платформе GitHub. Однако в целях безопасности большинство конфигурационных параметров не доступны публично с целью ограничения доступа к тестовому окружению.

3.5 Документация

Последней из крупных задач настоящей работы было создание пользовательской документации (Рисунок 18). Помимо непосредственного

составления материалов документации эта задача включала в себя также настройки автоматической сборки и публикации материалов документа.

Проведенная работа позволяет дополнять и обновлять материалы пользовательской документации без необходимости мануальной сборки и публикации. Эти процессы работают автоматически согласно тем же принципам, на которых основываются процессы автоматической сборки, публикации и развертывания основного кода приложения.

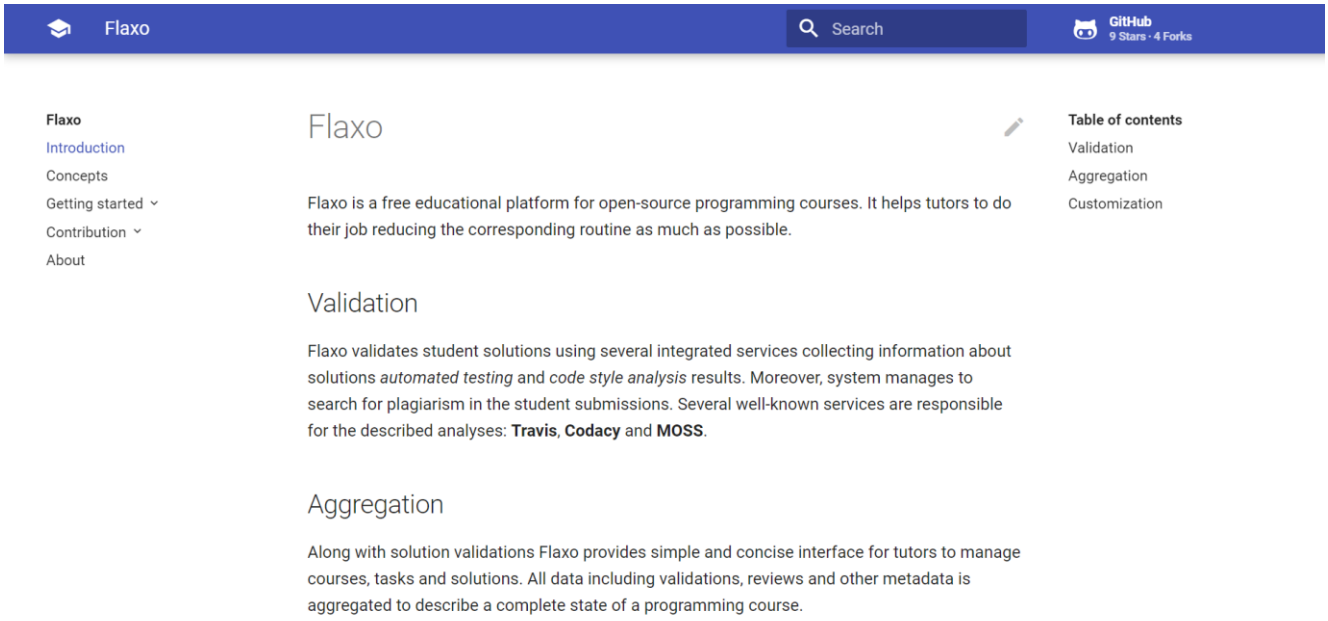

Рисунок 18 – Скриншот страницы пользовательской документации

4 АПРОБАЦИЯ

Последним этапом настоящей работы следует этап апробации, на котором все спроектированные и внедренные изменения проверяются на практике. По каждому улучшению приводятся разнообразные данные, в том числе статистические, если таковые имеются, и анализирует процесс функционирования. В итоге на основании приведенных проанализированных данных, делаются выводы о качестве внедренных изменений и их эффективности. Построенные выводы формируют базу для составления заключения по проведенному исследованию.

Большинство изменений носят качественный характер, вследствие чего апробация результатов их внедрения может носит только качественную оценку. Тем не менее часть изменений, результаты внедрения которых можно оценить и количественно, описаны ниже в настоящей главе с приведенными статистическими данные по результатам внедрения новой функциональности.

Апробация отдельных внедренных изменений проводилась довольно длительный период времени. Исходная система применялась на протяжении последних 4 семестров для проведения 10 практических курсов по программированию со средней численностью обучающихся порядка 40 человек, общим числом заданий порядка 50 и суммарным числом обработанных решений более 8 тысяч (Таблица 1).

Курс, №	Число заданий	Число обучающихся	Число обработанных решений
1	9	18	445
2	10	46	1370
3	2	24	109
4	3	35	259
5	3	19	133
6	2	60	680
7	10	58	4387
8	5	57	612
9	2	52	291

Таблица 1 – Сводные данные по части проведенных практических курсов с использованием системы Flaxo

4.1 Графовая визуализация плагиата

Одним из крупных изменений, внедренных в систему Flaxo, стало появление графовой визуализации результатов анализа плагиата. Добавление подобной возможности расширило возможности преподавателя по исследование случаев возможных заимствований исходного кода между обучающимися.

Внедренный инструмент графовой визуализации использовался на протяжение последнего года для исследования результатов анализа плагиата для 10 практических курсов по программированию. Инструмент позволяет эффективно использование его для практической курсов с различным числом обучающихся, с различными распределениями процента общего кода решений, а также с результатами анализа плагиата различными инструментов анализа плагиата.

Без использования графового инструмента визуализации результатов анализа плагиата преподавателю в рамках предварительного исследования необходимо было изучать отдельные найденные заимствования на основании единой сводной таблицы. Для определения групп взаимосвязанных обучающихся и классификации

работ, как оригинальных или заимствованных, преподавателю было необходимо учитывать время создания решений, а также приоритетность заимствований.

Ситуация особенно остро осложняется для больших групп обучающихся, когда их число становится более 50 человек. В таких ситуациях изучения результатов анализа плагиата становится очень сложным процессом, требующим высокой степени внимания.

Использование внедренного инструмента графовой визуализации драматически упрощает решение описанных выше задач. Так, преподавателю больше не требуется проделывать дополнительных действий как для выделения связанных групп обучающихся, так и для определения какие из связанных решений были раньше или позже других.

4.2 Механизм анализа плагиата

Помимо появления инструмента графовой визуализации результатов анализа плагиата также была внедрена интеграция с инструментов GitPlag, который, как уже было описано выше в настоящей работе, предоставляет универсальный интерфейс работы с инструментами анализа плагиата, такими как MOSS и JPlag.

Помимо расширения существующей функциональности анализа плагиата в системе Flaxo интеграция с инструментом GitPlag оптимизирует процесс агрегации решений обучающихся, что непосредственно влияет на скорость анализа плагиата. В частности, приведены сводные замеры времени работы процесса анализа плагиата (Таблица 2) для различных конфигураций инструментов для одного задания практического курса по программированию на языке программирования Java с 54 обучающимися.

Запуск, №	Конфигурация			
	Flaxo	Flaxo + GitPlag		
	MOSS, с	MOSS, с	JPlag, с	Combined, с
1	336	67	1	82
2	328	62	1	66
3	325	61	1	63
4	325	63	1	63
5	720	61	1	67
6	650	66	1	77
7	683	62	1	67
8	280	64	1	67
9	340	62	1	63
10	344	60	1	66
Среднее	433	63	1	68

Таблица 2 – Время анализа плагиата практического курса
с использованием различных конфигураций

На основании собранных данных можно сделать вывод о значительно сниженном времени работы процесса анализа плагиата или значительном увеличении скорости работы процесса анализа плагиата. В частности, подобное ускорение работы можно объяснить проводимой оптимизацией процессов агрегации и кэширования решений обучающихся перед анализом плагиата в системе GitPlag.

Можно также увидеть, что использование анализатора JPlag, предоставляемого сервисом GitPlag, позволяет проводить анализ плагиата за константное время равное 1 секунде. Подобное преимущество позволяет преподавателю проводить более регулярный анализ плагиата и выявлять недобросовестные заимствования как можно раньше.

Используя приведенные данные, можно также заключить, что использование агрегированного инструмента анализа плагиата Combined уникального для

системы GitPlag позволяет получать сводный результат анализа плагиата за время, сравнимое со временем работы при использовании только одного анализатора MOSS. Подобная возможность позволяет использовать более надежные результаты анализа плагиата, которые согласно некоторым источникам обладают сниженным числом ложных срабатываний.

Учитывая перечисленные выше преимущества, можно говорить о более эффективном функционировании системы анализа плагиата в системе Flaxo в связи с внедрением описанных изменений, в частности, интеграцией с системой GitPlag.

Отдельного упоминания заслуживает внедренная возможность запуска анализа плагиата отдельных заданий курса. Ранее, преподаватель мог провести анализ плагиата всего практического курса, что было непрактично в некоторых случаях. Внедрение поддержки анализа плагиата отдельных заданий сделало возможным независимое исследование решений обучающимися заданий с различными сроками дедлайнов.

4.3 Платформенные изменения

Переходя к непосредственно платформенным изменениям необходимо перечислить основные внедренные изменения, а именно:

- создание системы оповещений обучающихся,
- добавление возможности импорта практических курсов на основании существующих репозиториев в системе контроля версий,
- расширение возможностей по автоматическому созданию инфраструктуры курса.

Внедренная система оповещений обучающихся позволяет привязывать к некоторым действиям преподавателя рассылку оповещений связанным обучающимся. Так, в частности событие выставления баллов решению преподавателем теперь вызывает автоматическое оповещение обучающегося, опубликовавшего данное решение.

Оригинальный педагогический процесс не предусматривал непосредственного взаимодействия системы Flaxo с обучающимися, что на практике оказалось достаточно неэффективным. Ранее обучающийся мог узнать о проверках, проведенных преподавателем, только в момент их финального опубликования, а доступны ему были только результаты автоматизированных проверок.

С момента добавления системы оповещений каждое изменение преподавателем баллов за решения обучающихся отражается в соответствующих пулл реквестах в виде комментариев с указанием на количество выставленных баллов. Таким образом, время обратной связи между преподавателем и обучающихся уменьшается драматически. Помимо прочего, преподавателю не нужно совершать дополнительных действий для оповещения обучающихся кроме непосредственного проставления баллов.

Преподаватель имеет возможность включать и отключать автоматические оповещения, а также настраивать шаблон оповещений, что увеличивает гибкость описанного подхода. Таким образом для некоторых ситуаций, в которых результат проверок должен оставаться скрытым, оповещения могут быть отключены.

Помимо внедрения системы оповещений в настоящем разделе рассматривается также добавление возможности импорта практических курсов на основании репозиториев в системе контроля версий.

Подобное улучшение позволяет преподавателю использовать уже существующие репозитории в качестве практических курсов в системе Flaxo без необходимости их пересоздания в системе Flaxo с нуля. Это значительно упрощает пользовательский опыт преподавателей, которые уже имели похожую инфраструктуру для проведения практических курсов перед началом использования системы Flaxo.

Последним изменением, затронутом в настоящем разделе, является расширение возможностей автоматического создания инфраструктуры курса. Ранее преподаватель мог создавать курсы с использованием только предустановленных заранее конфигураций окружения. Внедренные изменения

снимают эти ограничения и позволяют создавать практические курса без автоматически создаваемого окружения. Подобное улучшение дает возможность создавать и автоматизировать любые практические курсы по программированию.

4.4 Развертывание

Процесс развертывания системы Flaxo был в значительной степени преобразован в рамках настоящей работы. Ранее все процессы сборки и развертывания были описаны в технической документации и требовали проведения большого числа мануальных операций. Любой исследователь, желающий использовать платформу в собственном окружении, был вынужден самостоятельно настраивать все окружение, устанавливать зависимости, запускать отдельные сервисы.

После того, как процессы автоматизированной сборки, публикации и развертывания были настроены для системы Flaxo, требования, накладываемые на необходимое для развертывание окружение, свелись к минимуму. В частности, единственным требованием является установка инфраструктуры Docker и Docker Compose.

Используя предварительно собранные контейнеры всех сервисов приложения и опубликованные конфигурации, любой исследователь способен развернуть всю систему, включающую базу данных, разработанный инструмент анализа плагиата, а также сервис GitPlag, в течении нескольких минут.

Публикующиеся контейнеры содержат сборки приложений, которые прошли стадию автоматизированного тестирования, так что можно говорить о некотором уровне надежности автоматически публикуемых версий системы.

Настроенный процесс автоматического развертывания новых версий системы в тестовое окружение позволяет, как это уже было описано ранее, замкнуть процессы получения обратной связи и выработки новых требований в циклический процесс, в котором работа инженеров поддержки сведена к минимуму и ограничивается только имплементацией изменений.

4.5 Документация

Последним изменением, затрагиваемом в настоящей главе, будет добавление пользовательской документации, которая на текущий момент включает описание только основных процессов системы.

Несмотря на то, что многие аспекты работы системы в настоящий момент не покрыты пользовательской документацией, внедренные изменения составляют значительный фундамент для последующего развития документации. Так, большой ценностью являются настроенные процессы автоматизированной сборки и публикации пользовательской документации, аналогичные описанным выше для исходного кода.

Любое изменение содержания пользовательской документации вызывает автоматическое срабатывание процессов ее автоматической сборки и публикации. Пользовательская документация размещается публично на открытом ресурсе для размещения документации в соответствующем домене.

Содержание пользовательской документации разбито на отдельные разделы и подразделы с целью более четкой организации структуры документации. Кроме того, редактирования содержания документации происходит также, как и изменения исходного кода систем путем внесения изменений в репозиторий проекта в системе контроля версий.

ЗАКЛЮЧЕНИЕ

Последней главой настоящей выпускной квалификационной работы является заключение, в котором приводятся результаты проделанной работы и формируются выводы, заключающие настоящее исследование.

В рамках движения к поставленной в настоящей работе цели об исследовании и оптимизации практического применения системы Flaxo для проведения практических курсов по программированию была проведена достаточная широкая работа, включающая исследование бизнес-процессов, выделение недостатков их функционирования, формирование задач по оптимизации, их проектирование, реализацию и апробацию.

В частности, был исследован педагогический процесс, построенный оригинальной системой Flaxo, в рамках оптимизации которого был решен спектр задач оптимизации процесса анализа плагиата, включающий создание инструмента графовой визуализации результатов анализа плагиата, интеграцию системы анализа плагиата GitPlag, предобработку данных для анализа плагиата и индикацию результатов его работы.

Кроме того, часть задач оптимизации педагогического процесса включала развитие системы в направлениях, не связанных непосредственно с процессом анализа плагиат. Так, были решены следующие задачи: создание системы оповещения обучающихся, поддержка импорта практических курсов из существующих репозиториев систем контроля версий, а также расширение возможностей инициализации практических курсов по программированию.

Помимо исследования педагогического процесса, большой частью настоящей работы было изучение процесса технической поддержки системы Flaxo. Были решены задачи по настройке процессов автоматической сборки, публикации и развертывания инфраструктуры платформы, которые драматически упрощают техническое сопровождение системы, а также возможности по развертывания системы в любых пользовательских окружениях.

Другой стороной технической поддержки исследованной в рамках настоящей работы было создание пользовательской документации, которая включает в себя описание основных элементов системы, и которая также как и инфраструктура приложения автоматически собирается и публикуется при любых изменениях.

Описанные выше задачи были сформированы, спроектированы, разработаны, внедрены и апробированы в рамках настоящей работы. Большинство изменений имеет только качественную оценку, которая однако ощутимо влияет на процесс практического использования системы, как преподавателями, так и инженерами поддержки.

Проведенная работа по исследованию бизнес-процессов системы Flaxo и оптимизации их работы оказывает значительный эффект на процесс практического использования системы Flaxo для проведения практических курсов по программированию.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей работе используются следующие сокращения.

CI – Continuous integration

CD – Continuous delivery

СПИСОК ИЛЛЮСТРАТИВНОГО МАТЕРИАЛА

Список иллюстраций

Список таблиц

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Aiken Alex. Moss, a system for detecting software plagiarism. - 2002. - 10 с.
2. Aivaliotis Dimitri. Mastering Nginx // Packt Publishing. - 2013. - 322 с.
3. Bhramadeo Vishnu Deokate, Dinesh Bhagwan Hanchate. Software Source Code Plagiarism Detection Using Latent Semantic Analysis // International Journal of Science and Research. - 2016. - 5 с.
4. Boris Lesner, Romain Brixtel, Cyril Bazin, and Guillaume Bagan. A novel framework to detect source code plagiarism: Now, students have to work for real! // 2010 ACM Symposium on Applied Computing. - 2010. - 2 с.
5. Chunhui Wang, Zhiguo Liu, and Dongsheng Liu. Preventing and detecting plagiarism in programming course // International Journal of Security and Its Applications. - 2013.
6. Dieter Pawelczak. Effects of plagiarism in introductory programming courses on the learning outcomes // Fifth International Conference on Higher Education Advances. - 2019.
7. Duvall Paul M. Continuous Integration: Improving Software Quality and Reducing Risk // Addison-Wesley Professional. - 2007. - 318 с.
8. George Obaido, Pravesh Ranchod, and Richard Klein. Constructing and analysing plagiarism in student programs using graphs // Proceedings of the Second International Conference on the Internet, Cyber Security and Information Systems. - 2017. - 8 с.
9. Jez Humble, David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation // Addison-Wesley Professional. - 2010. - 512 с.
10. Mouat Adrian. Using Docker: Developing and Deploying Software with Containers // O'Reilly Media. - 2016. - 354 с.
11. Omar Portillo, Vanessa Ayala-Rivera, Evin Murphy, and John Murphy. A unified approach to automate the usage of plagiarism detection tools in

programming courses // 12th International Conference on Computer Science and Education. - 2017. - 6 с.

12. Oscar Karnalim and Gisela Kurniawati. Programming style on source code plagiarism and collusion detection. - 2020. - 12 с.
13. Rene Bonifacio. Effects of plagiarism intervention program in the research papers of central mindanao university students // International Journal of Scientific & Technology Research. - 2020. - 7 с.
14. Ricardo Franclinton, Oscar Karnalim. A language-independent library for observing source code plagiarism // Journal of Information Systems Engineering and Business Intelligence. - 2019. - 10 с.
15. Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. - 2003. - 10 с.
16. Vitor Martins, D. Fonte, Pedro Rangel Henriques, Pedro & Daniela da Cruz. Plagiarism detection: A tool survey and comparison // OpenAccess Series in Informatics. - 2014. - 15 с.
17. Zenon Gniazdowski. Detection of a source code plagiarism in a student programming competition // Zeszyty Naukowe. - 2019. - 21 с.
18. Е.Г. Белых, А.А. Калинин, Л.А. Бардонова, В.А. Бывальцев, И.А. Степанов. Плагиат и академическая добросовестность в науке // Вестник Российской академии медицинских наук. - 2017. - 6 с.
19. Степочкин Н.А., Ефимчик. Е.А. Разработка информационной системы анализа схожести исходного кода решений задач по программированию в открытых образовательных репозиториях. - 2019. - 55 с.
20. Цибин А.И., Ефимчик Е.А. Графовая визуализация плагиата в практических курсах по программированию // Сборник тезисов докладов конгресса молодых ученых. Электронное издание [https://kmu.itmo.ru/file/download/application/13166]. - Режим доступа: ссылка на страницу с тезисом, своб. - 2020. - 2 с.

21. Цибин А.И., Ефимчик Е.А. Разработка информационной системы создания, сдачи и проверки заданий по дисциплине ”Программирование” // Университет ИТМО. - 2018. - 50 с.
22. Цибин А.И., Ефимчик Е.А. Распределенная интегрированная информационная система организации сдачи и проверки заданий по программированию: применение и особенности технической реализации // Сборник тезисов докладов конгресса молодых ученых. Электронное издание [http://openbooks.ifmo.ru/ru/file/7344/7344.pdf]. - Режим доступа: ссылка на страницу с тезисом, своб. - 2018. - 2 с.