Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ АНАЛИЗА СХОЖЕСТИ ИСХОДНОГО КОДА РЕШЕНИЙ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ В ОТКРЫТЫХ ОБРАЗОВАТЕЛЬНЫХ РЕПОЗИТОРИЯХ

Автор Степочкин Никита Андреевич _____
(Фамилия, Имя, Отчество) (Подпись)

Направление подготовки (специальность)_____
(код, наименование)
09.03.02 Информационные системы и технологии

Квалификация _____бакалавр_____
(бакалавр, магистр)*

Руководитель ВКР Ефимчик Е.А., к.т.н. _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

К защите допустить

Руководитель ОП Лисицина Л.С., д.т.н., профессор _____
(Фамилия, И.О., ученое звание, степень) (Подпись)

“_____”_____ 20 ____г.

Санкт-Петербург, 2019 г.

Студент__Степочкин Н.А____Группа_Р3420__ Факультет__ПИиКТ__
(Фамилия, И. О.)

Направленность (профиль), специализация _____
____Автоматизация и управление в образовательных системах_____

Консультант (ы):
а) _____ _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

б) _____ _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

ВКР принята “____”_____20 ____г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты “____”_____20 ____г.

Секретарь ГЭК _____ _____
(ФИО) (подпись)

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

УТВЕРЖДАЮ

Руководитель ОП

_____ _____
(Фамилия, И.О.) (подпись)

«____» «_____» 20____ г.

З А Д А Н И Е
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студенту Степочкину Никите Андреевичу **Группа** P3420 **Факультет** ПИиКТ

Руководитель ВКР Ефимчик Е.А., к.т.н., Университет ИТМО, доцент ПИиКТ
(ФИО, ученое звание, степень, место работы, должность)

1 Наименование темы: Разработка информационной системы анализа схожести исходного кода решений задач по программированию в открытых образовательных репозиториях

Направление подготовки (специальность) 09.03.02 Информационные системы и технологии

Направленность (профиль) Автоматизация и управление в образовательных системах

Квалификация бакалавр

2 Срок сдачи студентом законченной работы «_____» «_____» 20_____г.

3 Техническое задание и исходные данные к работе Требуется разработать автоматизированную систему анализа схожести исходного кода решений задач по прораммированию для анализа схожести решений задач по программированию, хранящихся в открытых образовательных репозиториях

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

___1. Анализ отрицательных факторов в механизмах работы существующих систем анализа схожести исходного кода_____

___2. Проектирование и разработка автоматизированной системы _____

___3. Проведение тестирования разработанной системы и описание результатов ее работы__

_____

_____

_____

_____

_____

_____

_____

5 Перечень графического материала (с указанием обязательного материала)

_____

_____

_____

_____

_____

_____

_____

6 Исходные материалы и пособия

_____

_____

_____

_____

_____

_____

7 Дата выдачи задания «____» «_____» 20____г.

Руководитель ВКР_____
(подпись)

Задание принял к исполнению_____ «____» «_____» 20____г.
(подпись)

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

АННОТАЦИЯ

ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент Степочкин Никита Андреевич
(ФИО)

Наименование темы ВКР: Разработка информационной системы анализа схожести исходного кода решений задач по программированию в открытых образовательных репозиториях

Наименование организации, где выполнена ВКР

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования Создание автоматизированной системы анализа схожести исходного кода решений задач по программированию

2 Задачи, решаемые в ВКР Анализ отрицательных факторов в механизмах работы с существующими системами анализа схожести исходного кода; проектирование и разработка автоматизированной системы анализа схожести исходного кода для выполнения анализа решений задач в открытых образовательныч репозиториях; проверка работоспособности разработанной системы и описание результатов ее тестирования

3 Число источников, использованных при составлении обзора 3

4 Полное число источников, использованных в работе 10

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
			7	1	2

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	**Параграф работы**
Moss	2.1
JPlag	2.2
Kotlin, Gradle, Spring	3.1
JavaScript, React, WebSocket, SockJS	3.2
PostgreSQL	3.3
Docker	3.4
Git, Webhook, Github, Gitlab, Bitbucket	3.5

8 Краткая характеристика полученных результатов В результате выполнения данной дипломной работы была спроектирована, разработана и протестирована система анализа

схожести_исходного кода решений задач по программированию, использующая для анализа существующие системы анализа схожести исходного кода, и работающая с открытыми образовательными репозиториями как с источниками решений задач, упрощающая анализ схожести исходного кода и автоматизирующая процессы загрузки решений, их подготовки и отправке на анализ существующим системам анализа схожести исходного кода

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

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

а) 1 _____
(Библиографическое описание публикаций)
2_____
3_____

б) 1_____
(Библиографическое описание выступлений на конференциях)
2_____
3_____

Студент__Степочкин Никита Андреевич____ _____
(ФИО) (подпись)

Руководитель ВКР___Ефимчик Евгений Александрович_ _____
(ФИО) (подпись)

“_____”_____20___г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Для выполнения практических задач по программированию часто используются открытые репозитории. Процесс работы с такими репозиториями в общем случае выглядит следующим образом: преподаватель создает репозиторий и добавляет в него заготовки файлов исходного кода, студент создает собственную копию репозитория преподавателя, дополняет заготовленные файлы собственным кодом, представляющим из себя решение данной практической задачи, и отправляет свое решение в репозиторий преподавателя. Далее преподаватель проверяет решение студента и ставит ему соответствующую оценку. Такой метод выполнения задач по программированию удобен, так как позволяет структурировать решения студентов в одном репозитории, осуществлять отправку решений в любое время, а такие публичные сервисы как Github визуализируют список решений, позволяют указать название задачи и имя студента, а также просмотреть файлы исходного кода решения задачи. При этом, если количество задач и количество студентов достаточно велико, то возникает трудность в сравнении решений задач от разных студентов на предмет схожести. А визуальное сравнение решений задач по программированию в большинстве случаев неспособно выявить сходства в исходном коде решений, так как решения могут отличаться по операторам, модификаторам, комментариям и другим структурам языков программирования, и при этом не отличаться друг от друга по самой структуре решений и по выполняемым операциям в исходном коде.

Существующие системы анализа схожести исходного кода неудобны для использования с образовательными репозиториями, так как взаимодействие с ними возможно только при помощи загрузки всех файлов решений из репозитория, их структурировании и отправке анализаторам, что неудобно и занимает продолжительное время. Сами же существующие системы такого анализа малоинформативны и предоставляют два типа визуализации результатов своей работы: пары названий отправленных файлов с соответствующей этим

файлам степени схожести в виде процента, и визуализацию двух файлов с указанием участков, подозрительных на схожесть.

Для создания и управления репозиториями, которые используются для образовательных целей, существует система Flaxo. Она автоматически создает репозитории, регистрирует отправленные решения студентов, запускает различные проверки решений, в том числе проверку на схожесть исходного кода при помощи системы Moss. При этом результат этой проверки представляет из себя максимальный процент совпадения данного решения с каким-либо из других решений той же задачи, а также ссылку на результат работы анализатора. Процесс анализа схожести решений студентов в системе Flaxo имеет ряд недостатков:

- Отчет об анализе системы Moss хранится только на сервере этой системы и сохраняется только в течение 14 дней, после чего удаляется;
- Каждый запуск анализа требует полной загрузки файлов решений из репозиториев, что может занимать длительное время и приводить к невозможности загрузки файлов из-за ограничений по количеству запросов к сервисам управления репозиториями;
- Отсутствие возможности сортировки результатов анализа по степени совпадения и группировки результатов по имени студента;
- Отсутствие возможности в проверке того, какие файлы решений будут отправляться на анализ, из-за чего анализ может быть неполным;
- Система Flaxo для анализа схожести может использовать только систему Moss.

Для визуализации результатов анализа схожести система Flaxo использует систему data2graph, которая визуализирует пары студентов в виде графа. В графе узлами являются студенты, а ребра – совпадением решений студентов, которые находятся на их узлах. При этом ребру соответствует процент схожести их решений, и в зависимости от заданного минимального процента совпадения на графе отображаются ребра, соответствующий процент которых больше заданного. Данный граф имеет несколько недостатков:

- Отсутствие направленности, в связи с чем при помощи графа невозможно определить, чье решение двух студентов было создано раньше;
- Граф отображает все узлы вне зависимости от того, имеют ли они ребра, выходящие из них, что при большом количестве входных данных может негативно влияет на его визуальное восприятие.

Целью данной дипломной работы является создание автоматизированной системы загрузки решений задач по программированию из образовательных репозиториев, анализа схожести их исходного кода при помощи существующих систем анализа и визуализации получаемых результатов. Также разрабатываемая система должна предоставлять возможность использования анализа схожести исходного кода сторонними системами, такими как системе Flaxo.

Для визуализации результатов анализа предлагается использовать существующую систему data2graph, добавив в нее функциональные элементы для более информативного отображения результатов анализа.

В данной дипломной работе приведен анализ существующих систем анализа схожести исходного кода, предложена автоматизированная система, позволяющая интегрировать между собой анализаторы исходного кода и системы управления репозиториями и предоставлять подробную визуализацию получаемых результатов анализа схожести исходного кода.

1 АНАЛИЗ ТРЕБОВАНИЙ И ОБЗОР АНАЛОГОВ

В открытом доступе существует небольшое количество систем анализа схожести исходного кода. На данный момент в работоспособном состоянии находятся два таких сервиса: Moss и JPlag.

2.1 Обзор системы Moss

Система Moss основана на анализе схожести исходного кода про помощи метода отпечатков: суть метода состоит в том, что исходный код переводится в зашифрованный хэш-код, из которого в примерно равноудаленных позициях берутся отметки, характеризующие данный код, и которые сравниваются с такими отметками из других решений на предмет совпадений.[1] Система Moss позволяет указать чувствительность анализа, по которой определяется, нужно ли указать найденное совпадение исходных кодов двух решений как схожее исходя из длины совпавших участков. Для анализа системой Moss используется два типа файлов: базовые файлы, исходный код которых игнорируется при анализе и не участвует в сравнении решений, и файлы решений, исходный код которых подвергается сравнению.

Исходный код системы находится в закрытом доступе, а взаимодействие с ней происходит про помощи отправки файлов с исходным кодом и параметров анализа на сервер. Система попарно сравнивает исходные файлы и возвращает веб-страницу со списком попарных сравнений файлов и соответствующим уровнем совпадения кода в виде процента совпадения, а также визуализацию пар текстов файлов с выделением совпавших участков. Из-за необходимости отправки файлов исходного кода на сервер, скорость анализа сравнительно медленная при отправке большого количества файлов, так как скорость анализа возрастает пропорционально количеству отправленных файлов из-за, во-первых,

1. Saul Schleimer, Daniel S. Wilkerson, Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. - 2003. - 10 с.

необходимости отправить сам файл на удаленный сервер и, во-вторых, из-за необходимости сравнивать каждый файл с каждым.

Система находится в состоянии активной разработки, поддерживает 23 языка программирования и имеет большое количество клиентов для взаимодействия с ней на разных языках программирования. Сама система предназначена в первую очередь для сравнения файлов исходного кода без дифференциации по автору самого кода, поэтому использование ее для анализа схожести одного множества файлов с другим затруднено из-за необходимости четко определять, какие файлы созданы одним автором, а какие – другим. Для этого необходимо вручную либо при помощи встроенных функций системы объединять файлы, созданные одним автором, в один файл для исключения возможности сравнения файлов, имеющих одного автора, друг с другом. Результат каждого анализа хранится на сервере анализатора с возможностью доступа в течение двух недель, после чего удаляется.

2.2 Обзор системы Jplag

Система JPlag основана на анализе схожести исходного кода при помощи конвертации кода в набор токенов, которые характеризуют функциональную роль каждого участка кода, и сравнении наборов токенов из разных файлов друг с другом.[2] Система JPlag также, как и система Moss, позволяет указать чувствительность анализа, и, исходя из размеров последовательностей совпавших токенов, добавляет совпавшие участки в отчет об анализе. Также, система JPlag, как и система Moss, использует два типа файлов для анализа: базовые файлы и файлы решений.

Исходный код системы находится в открытом доступе, взаимодействие с ней происходит локально и все результаты анализа сохраняются в виде веб-страниц на машине, на которой осуществляется запуск анализа. Веб-страницы

2. Lutz Prechelt, Guido Malpohl, Michael Philippsen, JPlag: Finding plagiarisms among a set of programs. – 2000. – 44 с.

результата анализа содержат, аналогично системе Moss, результаты сравнения пары решений в виде процента совпадений, а также пары содержимого файлов исходного кода с выделенными позициями совпадений кода этих файлов. Анализ на схожесть исходного кода происходит сравнительно быстро из-за отсутствия необходимости отправки файлов решений на удаленный сервер, но время анализа так же возрастает пропорционально в соответствии с количеством отправляемых решений. Система поддерживает всего 6 языков программирования. Эта система удобна для анализа файлов исходного кода, имеющих разных авторов, так как она распознает автора файла по названию директории, в который находятся файлы под его авторством.

2.3 Анализ требований к разрабатываемой системе

Процесс анализа исходного кода существующими системами анализа выглядит следующим образом:

1. На машине, на которой происходит запуск анализа, создается директория, содержащая базовые файлы, исходный код которых не участвует в анализе и игнорируется анализаторами, а также файлы, исходный код которых должен подвергнуться сравнению друг с другом;
2. При помощи исполняемого файла запускается анализ с указанием путей к базовым файлам, исходный код которых не участвует в анализе, и к анализируемым файлам;
3. В результате запуска анализатор выдает веб-страницы с результатами анализа.

Исходя из обзора существующих систем анализа схожести исходного кода, можно сделать выводы об их основных недостатках, которые можно устранить:

- Необходимость создавать директории, содержащие файлы исходного кода, в соответствии с необходимой структурой, а также с необходимой структурой самих файлов для того, чтобы обеспечить верный формат входных данных для предстоящего анализа;

- Сложность конфигурации анализа, состоящую в том, что каждый раз при вызове анализа необходимо явно указывать пути к каждому файлу, либо к каждой директории, которые необходимо проанализировать, а также в необходимости явно указывать конфигурационные параметры анализа;
- Немногословность выходных результатов анализа. Все системы указывают лишь проценты совпадений в исходном коде и конкретные позиции этих совпадений;
- Отсутствие возможности получения результатов анализа в виде данных, имеющих возможность дальнейшего использования в сторонних сервисах, таких как Flaxo.

Для выполнения задач по программированию часто используются публичные репозитории. Большинство систем управления репозиториями используют технологии Git, REST Api и Webhook. Технология Git предоставляет возможность создавать репозитории, копировать их, добавлять программный код и создавать запросы на изменения репозитория, от которого была создана копия, что является основными необходимыми функциями для репозиториев, предназначенных для образовательных целей. Технология Git имеет возможность создания веток в репозиториях, которые испольуются образовательными репозиториями как задачи, то есть название ветки является названием задачи, а сама ветка хранит в себе файлы, которые к ней относятся. Файлы ветки репозитория преподавателя являются базовыми файлами задачи, а файлы веток студентов в репозитории, который он скопировал от репозитория преподавателя, являются файлами решения задачи. Запрос на изменение из ветки репозитория студента в ветку репозитория преподавателя является отправкой решения задачи. Технология REST Api предоставляет возможность получения информации о репозиториях при помощи HTTP-запросов. Технология Webhook предоставляет возможность оповещения о событиях в репозиториях и отправки информации об этих событиях сторонним сервисам.

В данной работе предлагается автоматизированное решение, позволяющее интегрировать системы управления репозиториями с названными анализаторами

схожести исходного кода в одну систему. Проектируемая система должна иметь следующий функционал:

- Система должна иметь возможность подключения репозиториев для использования их данных и файлов для анализа, хранения параметров анализа и хранения результатов анализа;
- Репозиторий, который подключается к проектируемой системе, должен иметь возможность взаимодействовать с системой при помощи технологии Webhook. Это будет позволять автоматически загружать файлы преподавателя, которые он создает в своем репозитории, а также файлы решений, отправленных студентами в репозиторий преподавателя;
- Для подготовки файлов перед отправкой анализаторам система должна автоматически объединять файлы решений от каждого студента в один файл, чтобы их структура соответствовала требованиям анализаторов;
- Система должна отправлять подготовленные файлы анализаторам, извлекать из отчетов об анализе все данные этого анализа и сохранять эти данные в своей базе данных;
- Система должна предоставлять возможность использования себя сторонними системами, такими как Flaxo;
- Система должна предоставлять возможность визуального сравнения двух решений с указанием схожих участков, полученных от анализатора, и процентом сходства решений;
- Для визуализации результатов анализа система должна использовать систему data2graph. В эту систему должны быть добавлены две функциональные особенности: направления ребер графа, указывающих на решения, созданные позднее чем те, из которых выходят ребра, а также скрытие тех узлов графа, которые не имеют ребер.

2 ПРОЕКТИРОВАНИЕ

2.1 Пользовательские истории

Первым этапом проектирования системы является определение ролей пользователей системы, для каждой из которых описываются пользовательские истории, позволяющие определить, какие операции пользователь будет иметь возможность выполнить и какой результат работы системы пользователь должен получать.

В разрабатываемой системе предусмотрена одна роль – Преподаватель. Пользовательские истории преподавателя:

- Как преподаватель, я хочу создать в системе запись о существующем репозитории для того, чтобы система могла взаимодействовать с данным репозиторием.
- Как преподаватель, я хочу задать настройки анализа для того, чтобы они могли применяться анализаторами постоянно без необходимости указывать их при каждом анализе повторно.
- Как преподаватель, я хочу включить автоматическую загрузку создаваемых в репозитории файлов в систему для того, чтобы постоянно поддерживать загруженные в систему файлы в самом обновленном состоянии.
- Как преподаватель, я хочу просмотреть список загруженных в систему файлов решений и отсортировать их по именам студентов, именам файлов или именам задач для того, чтобы определить, все ли необходимые файлы загружены в систему, правильно ли они названы и к каким веткам репозитория они относятся.
- Как преподаватель, я хочу загрузить файлы репозитория в систему для того, чтобы они могли использоваться для анализа.
- Как преподаватель, я хочу запустить анализ подключенного репозитория на схожесть отправленных в него решений для того, чтобы получить результат данного анализа.

- Как преподаватель, я хочу просмотреть существующий результат анализа подключенного репозитория и отсортировать список совпадений по именам студентов для того, чтобы определить, насколько схожи решения студентов.
- Как преподаватель, я хочу просмотреть пару решений студентов с указанием участков, которые анализатор указал как подозрительные на схожесть для того, чтобы определить, действительно ли указанные участки кода имеют схожесть.
- Как преподаватель, я хочу удалить существующий результат анализа для того, чтобы данные о нем не занимали пространство на машине, на которой был произведен запуск этого анализа.
- Как преподаватель, я хочу удалить репозиторий для того, чтобы не хранить информацию о созданных отчетах анализа и файлы исходного кода, относящиеся к репозиторию, и чтобы не загружать файлы из этого репозитория в систему.
- Как преподаватель, я хочу изменить файлы репозитория для того, чтобы анализ схожести исходного кода проходил без использования содержимого этих файлов.

2.2 Варианты использования

На основе представленных пользовательских историй составлена диаграмма вариантов использования (Рисунок 1), описывающая то, как пользователь взаимодействует с системой, и какая функциональность ожидается от разрабатываемой системы.

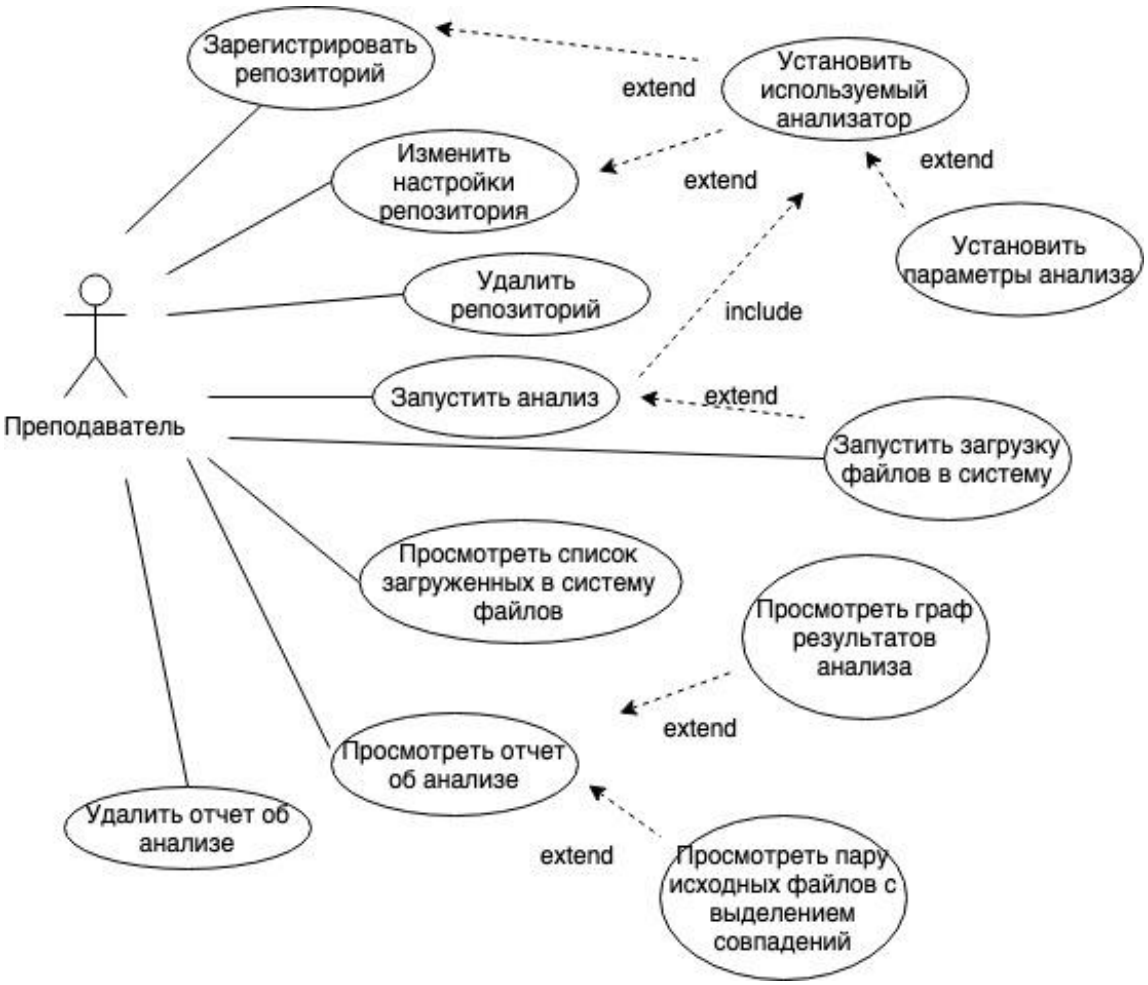

Рисунок 1 – Диаграмма вариантов использования

2.3 Сценарии работы

На основе пользовательских историй и вариантов использования составлены диаграммы последовательности, описывающие сценарии использования системы:

- Сценарий анализа схожести исходного кода репозитория (Рисунок 2);
- Сценарий автоматической загрузки файлов решения задачи (Рисунок 3);
- Сценарий автоматической загрузки файлов репозитория (Рисунок 4);
- Сценарий загрузки файлов репозитория и файлов решений задач (Рисунок 5).

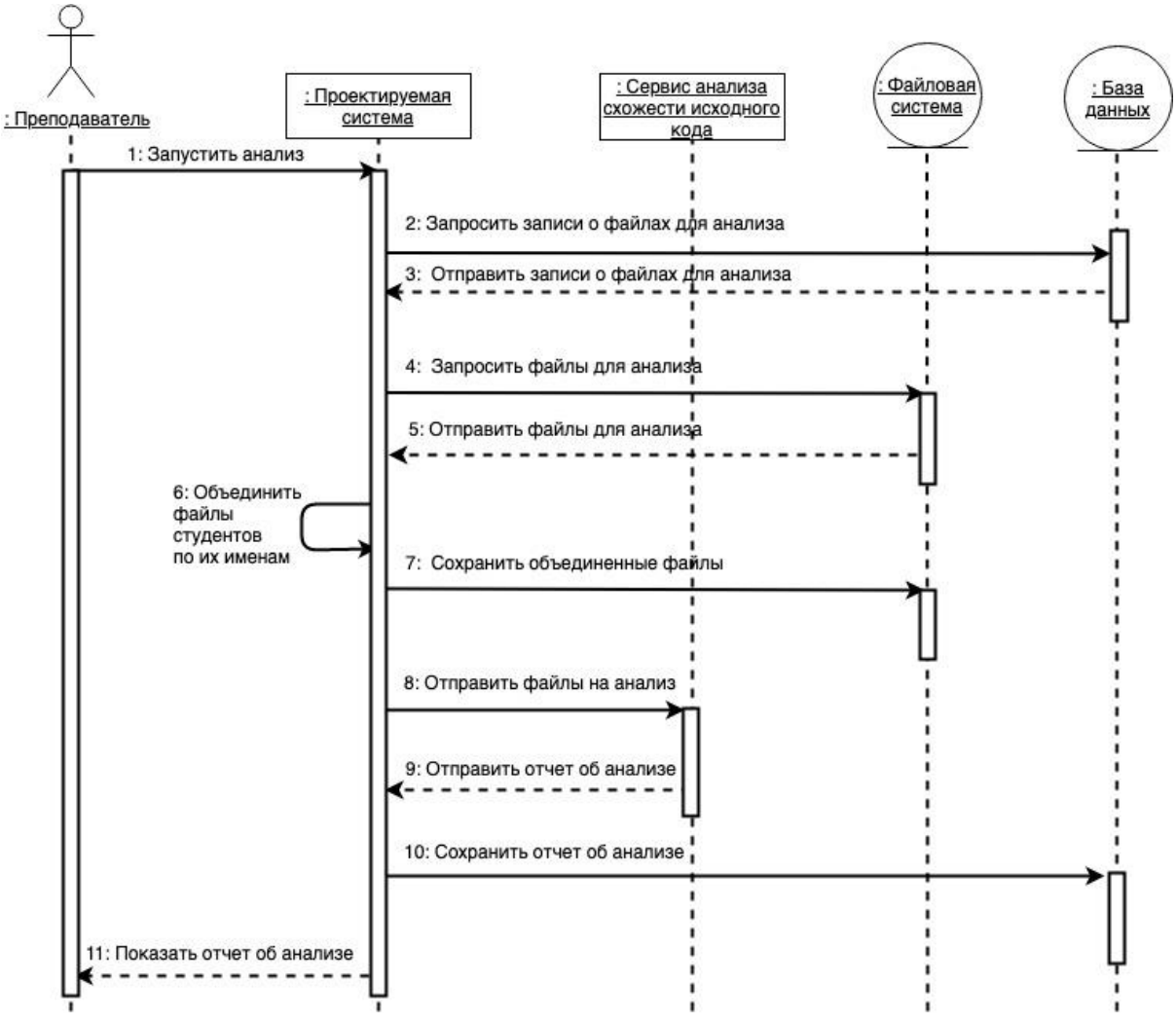

Рисунок 2 – Диаграмма последовательности для анализа репозитория

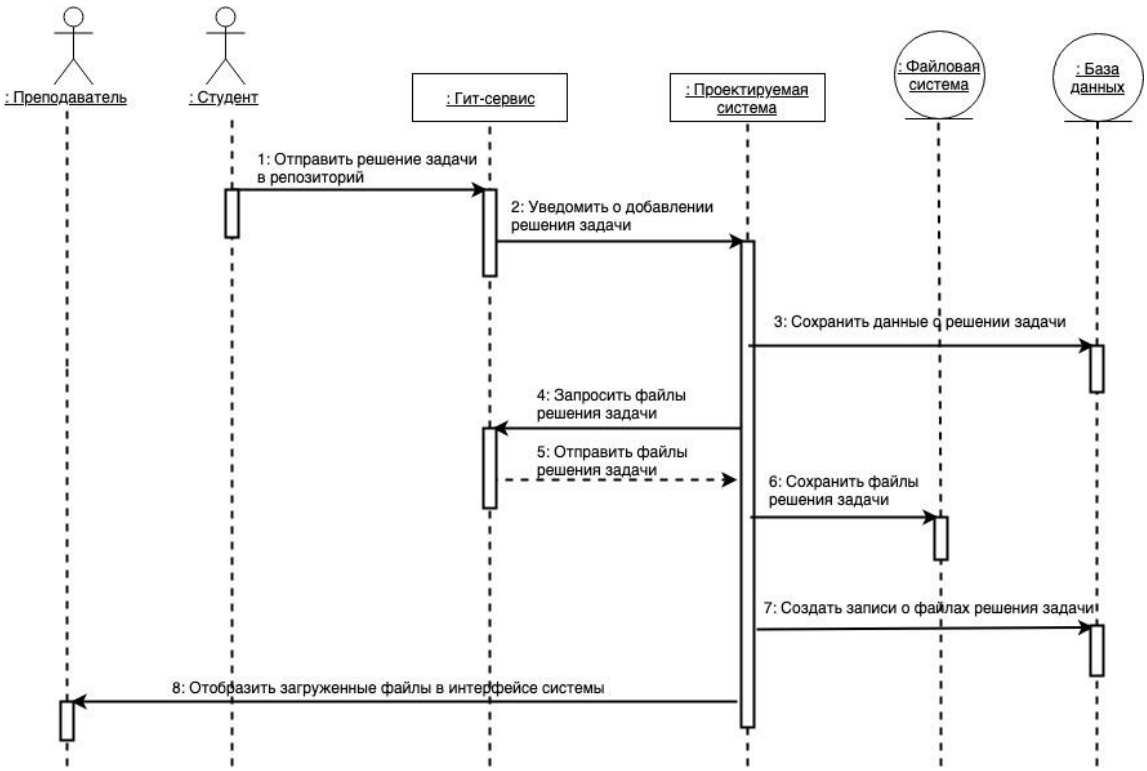

Рисунок 3 – Диаграмма последовательности для автоматической загрузки решения в систему

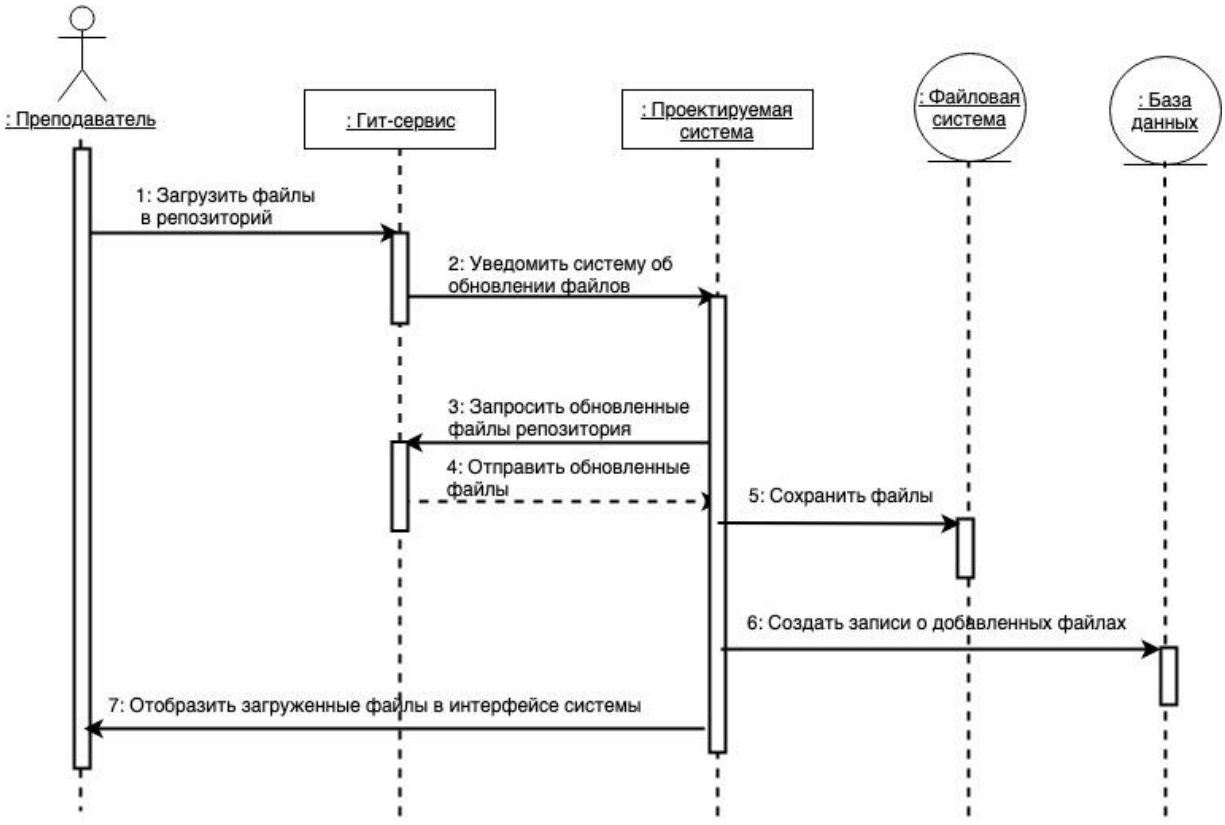

Рисунок 4 – Диаграмма последовательности для автоматической загрузки файлов репозитория в систему

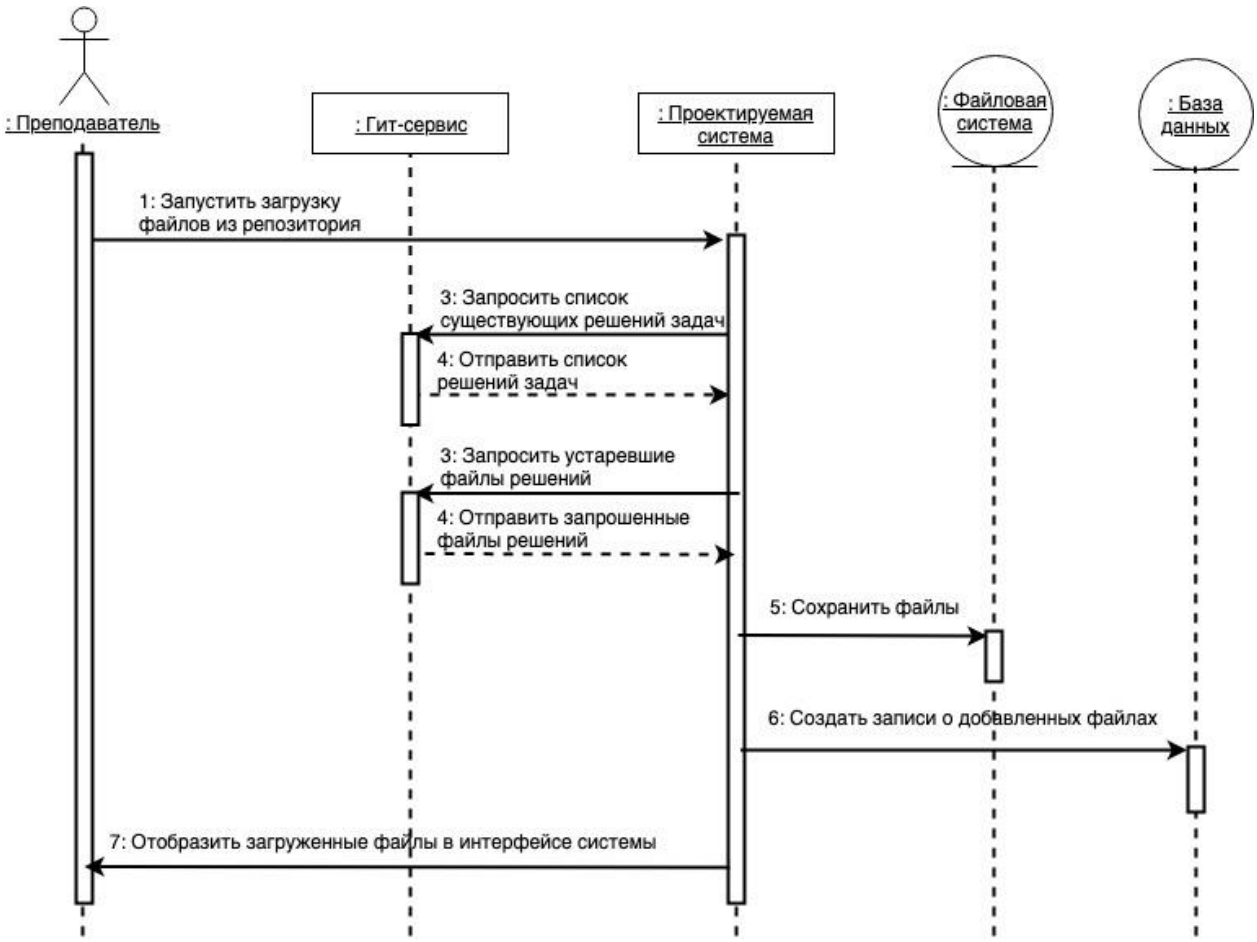

Рисунок 5 – Диаграмма последовательности для загрузки файлов репозитория и файлов решений в систему

2.4 Модель базы данных

На основе приведенных выше сценариев работы системы и технических требований создана модель базы данных.

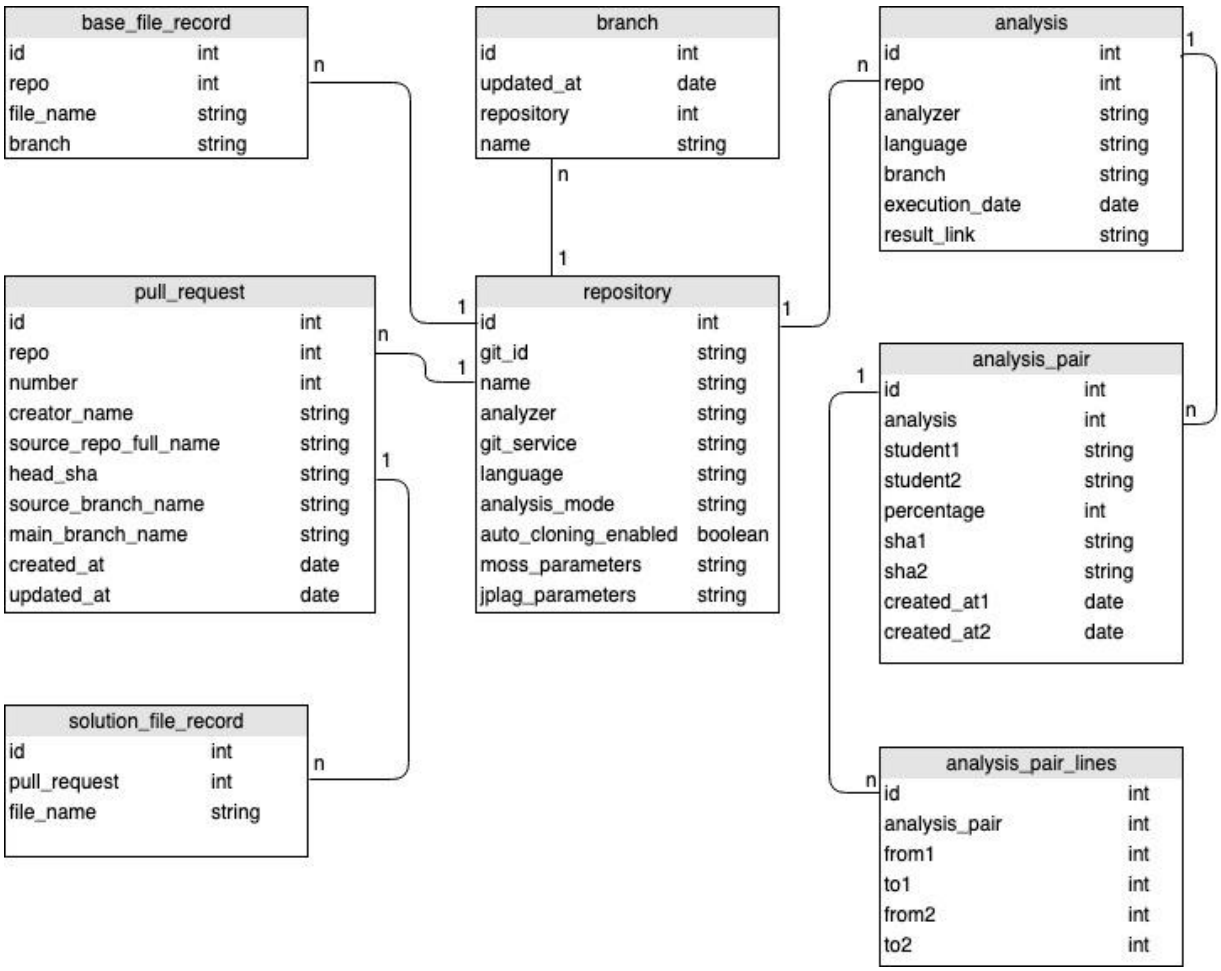

Рисунок 6 – Схема модели базы данных

- repository – репозиторий;
    - id – идентификатор репозитория;
    - git_id – идентификатор репозитория, полученный из системы, в которой он находится;
    - name – имя репозитория;
    - analyzer – имя анализатора, используемого по умолчанию;
    - git_service – гит-система, к которой относится репозиторий;
    - language – язык программирования данного репозитория;
    - analysis_mode – режим анализа, используемый данным репозиторием по умолчанию;
    - auto_cloning_enabled – параметр, указывающий на то, загружаются ли файлы из гит-репозитория при получении уведомления об изменении этих файлов ;

    - moss_parameters – параметры анализатора Moss, используемые данным репозиторием по умолчанию;
    - jplag_parameters – параметры анализатора JPlag, используемые данным репозиторием по умолчанию.
- pull_request – запрос на изменение файлов репозитория;
    - id – идентификатор запроса на изменение;
    - repo – идентификатор репозитория;
    - number – номер запроса, используемый системой этого репозитория;
    - creator_name – имя создателя запроса;
    - source_repo_full_name – имя репозитория, из которого сделан запрос;
    - head_sha – хэш-значение, являющееся идентификатором решения в гит-сервисе;
    - source_branch_name – имя ветки репозитория, из которой создан запрос;
    - main_branch_name – имя ветки репозитория, в которую создан запрос;
    - created_at – время создания запроса;
    - updated_at – время последнего обновления запроса.
- analysis – отчет об анализе схожести исходного кода;
    - id – идентификатор анализа;
    - repo – идентификатор репозитория;
    - analyzer – название использовавшегося анализатора;
    - language – язык программирования, использовавшийся для анализа;
    - branch – имя ветки репозитория, которая была проанализирована;
    - execution_date – время выполнения анализа;
    - result_link – ссылка на результат анализа, полученный от анализатора.
- analysis_pair – пара студентов, которая была получена из результата анализа;
    - id – идентификатор пары студентов;
    - analysis – идентификатор анализа;
    - student1 – имя первого из двух студентов;

    - student2 – имя второго из двух студентов;
    - percentage – процент схожести решений студентов;
    - sha1 – хэш-значение первого решения;
    - sha2 – хэш-значение второго решения;
    - created_at1 – время создания первого решения;
    - created_at2 – время создания второго решения.
- analysis_pair_lines – пара позиций в решениях двух студентов, которые анализатор отметил как схожие;
    - id – идентификатор пары позиций;
    - analysis_pair – идентификатор пары студентов;
    - from1 – начальная позиция совпадения в решении первого студента;
    - to1 – конечная позиция совпадения в решении первого студента;
    - from2 – начальная позиция совпадения в решении второго студента;
    - to2 – конечная позиция совпадения в решении второго студента.
- branch – ветка репозитория;
    - id – идентификатор ветки;
    - updated_at – время последнего обновления содержимого ветки;
    - repository – идентификатор репозитория;
    - name – имя репозитория.
- base_file_record – запись о файле репозитория;
    - id – идентификатор записи;
    - repo – идентификатор репозитория;
    - file_name – имя файла;
    - branch – имя ветки репозитория, к которой относится файл.
- solution_file_record – запись о файле решения задачи;
    - id – идентификатор записи;
    - pull_request – запрос на изменение, к которому относится файл решения;
    - file_name – имя файла.

2.5 Диаграмма развертывания

Система должна состоять из четырех компонентов:

- Клиентская часть приложения;
- Серверная часть приложения;
- База данных;
- Система data2graph.

Эти компоненты должны быть запущены на одной машине. Файлы системы должны храниться в файловой системе машины, на которой будет запущена система, и данные базы данных также должны храниться в файловой системе машины для того, чтобы не зависеть от состояния запущенных компонентов системы. Взаимодействие с системой должно происходить по протоколу HTTP. На основании этих требований представлена диаграмма развертывания системы (Рисунок 7).

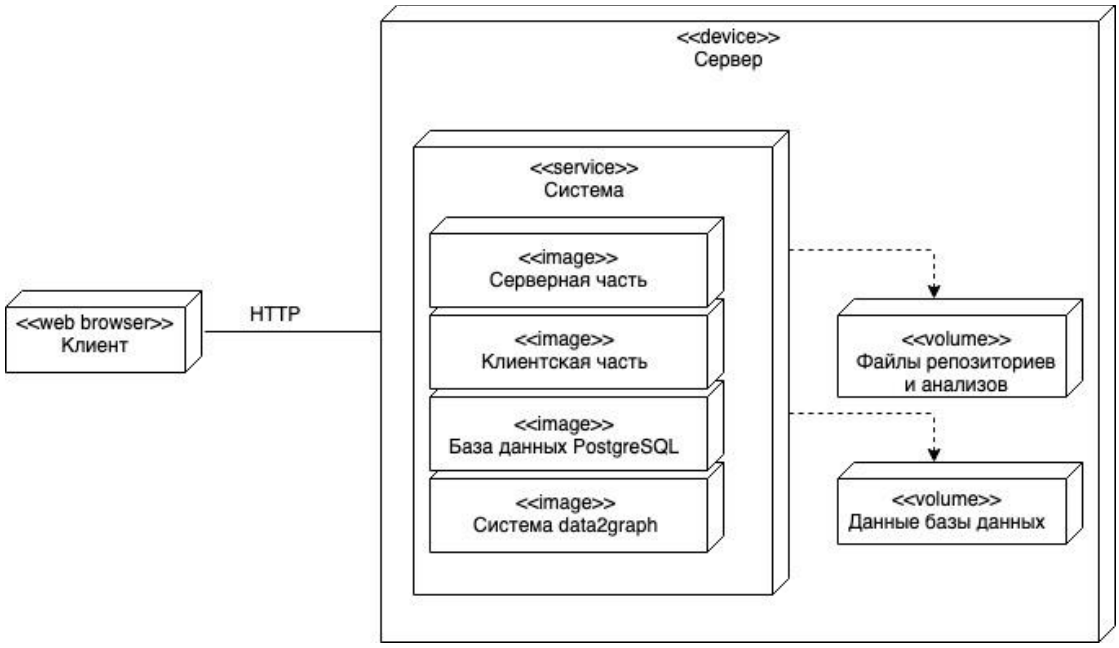

Рисунок 7 – Диаграмма развертывания системы

3 РАЗРАБОТКА

Разрабатываемая система построено на основе клиент-серверной архитектуры, и представляет из себя два модуля: клиентский и серверный. Основными преимуществом данной архитектуры является разграничение потребностей клиента и потребностей сервера, хранящего данные, что позволяет этим двум модулям развиваться независимо друг от друга, а также повышение безопасности системы из-за наличия централизованного сервера. Клиентская часть разрабатываемого приложения взаимодействует с серверной при помощи HTTP Api, построенного по архитектуре REST. Этот архитектурный стиль требует организовать доступ к ресурсам системы, используя определенный унифицированный набор операций, которые выполняются независимо от клиента и независимо от предыдущих выполненных операций, то есть без сохранения состояния.

3.1 Серверная часть

Для написания системы был выбран язык Kotlin. Разработка на языке Kotlin имеет ряд преимуществ:

- Высокий уровень поддержки функциональной парадигмы программирования, неизменяемых структур данных, поддержка написания собственных DSL-языков;
- Благодаря встроенным функциям языка, таким как поддержка безопасных типов объектов и делегирование свойств, Kotlin позволяет писать простой и понятный код, в значительной мере избегая необходимости в написании шаблонного кода;
- Kotlin является одним из JVM-языков, а также полностью поддерживает работу с кодом и библиотеками, написанными на языке Java, что значительно расширяет возможности внедрения языка Kotlin.

В качестве системы автоматизированной сборки проекта выбран Gradle. Gradle использует императивный синтаксис, что позволяет удобно и в краткой

форме создавать и изменять собственные задачи сборок, а также поддерживает большое количество встроенных плагинов, имеющих большое количество готовых задач, необходимых для большинства требований по автоматизированной сборке проекта.

Основным фреймворком, используемым для разработки серверной части приложения, является Spring. Spring является наиболее распространенным, свободно распространяемым фреймворком для разработки крупных приложений на языках Java и Kotlin, содержит большое количество подмодулей для решения разнообразных задач разработки и большое количество документации.

3.2 Клиентская часть

Для разработки клиентской части системы был выбран язык JavaScript. Этот язык был выбран, так как он поддерживает функциональную и объектно-ориентированную парадигмы, имеет большое количество поддерживаемых фреймворков и библиотек, а также большое количество документации.

Основным фреймворком, используемым для разработки клиентской части приложения, является React. React имеет простую структуру, позволяющую создавать независимые компоненты веб-интерфейса, реализовать веб-приложение по принципу SPA, то есть одностраничное приложение, состоящее из повторно используемых компонентов.

Для создания визуального оформления веб-интерфейса использовался фреймворк Bootstrap, позволяющий при помощи предоставляемых им настроек визуализации оформлять интерфейс в соответствии с конкретными требованиями к разрабатываемому веб-интерфейсу.

Для получения сообщений с сервера на клиентской части системы используется библиотека SockJS, которая использует протокол WebSocket. С помощью этой библиотеки клиент устанавливает постоянное соединение с сервером, что позволяет в режиме реального времени принимать сообщения с сервера и отображать их на пользовательском интерфейсе.

3.3 База данных

Для работы с базой данных была выбрана система управления базами данных (СУБД) PostgreSQL. Это свободно распространяемая СУБД, базирующаяся на языке SQL, имеющая высокопроизводительные механизмы транзакций и репликации, поддерживающая ACID-требования к транзакциям, обеспечивающие наиболее надёжную и предсказуемую работу СУБД. Также PostgreSQL поддерживает различные методы индексации базы данных и возможность создания в базе данных хранимых процедур.

3.4 Средства развертывания

Для развертывания разработанной системы была выбрана система Docker. Docker позволяет объединить любую программную систему с подходящей для нее средой выполнения, окружением и зависимостями в независимый контейнер, который будет иметь возможность контактировать с другими контейнерами. Основные преимущества использования Docker – это независимость от среды выполнения и изоляция рабочей системы от не относящегося к ней окружения.

Система Docker имеет встроенное средство docker-compose, предназначенное для определения и запуска системы, состоящей из нескольких контейнеров. Для конфигурации docker-compose используется конфигурационный файл docker-compose.yml, в котором должны быть определены все компоненты системы, переменные среды и порядок запуска контейнеров. Запуск разрабатываемой системы происходит при помощи docker-compose.

3.5 Сервис системы контроля версий

Разрабатываемое приложение должно пользоваться сервисами системы контроля версий для работы с анализируемыми файлами. Самыми распространенными такими сервисами являются:

- Github;
- Gitlab;

- Bitbucket.

В целях обеспечения возможности использования разрабатываемой системы независимо от используемого средства системы контроля версий было решено обеспечить поддержку всех трех систем.

3.6 Пользовательский интерфейс

Первоначально для создания интерфейса системы создаются макеты страниц, обозначаются основные компоненты. При формировании интерфейса системы важно учитывать такие параметры, как простота и понятность визуализации, удобство использования, наглядность компонентов интерфейса.

Пользовательский интерфейс системы построен по принципу одностраничного приложения, состоящего из обновляемых компонентов.

Интерфейс системы содержит несколько страниц:

- Страница списка подключенных репозиториев (Рисунок 8);
- Страница подключения нового репозитория (Рисунок 9);
- Страница подключенного репозитория (Рисунок 10);
- Страница изменения подключенного репозитория (Рисунок 11);
- Страница загруженных файлов репозитория в систему (Рисунок 12);
- Страница запуска анализа подключенного репозитория (Рисунок 13);
- Страница результата анализа (Рисунок 14);
- Страница сравнения двух решений (Рисунок 15).

Большая часть страниц разработанного пользовательского интерфейса содержат верхний блок указателя пути к текущей странице, что позволяет удобно перемещаться между страницами системы.

Страница списка подключенных репозиториев (Рисунок 8) содержит список репозиториев с указанием используемого средства системы контроля версий. При нажатии на название репозитория происходит переход на его страницу.

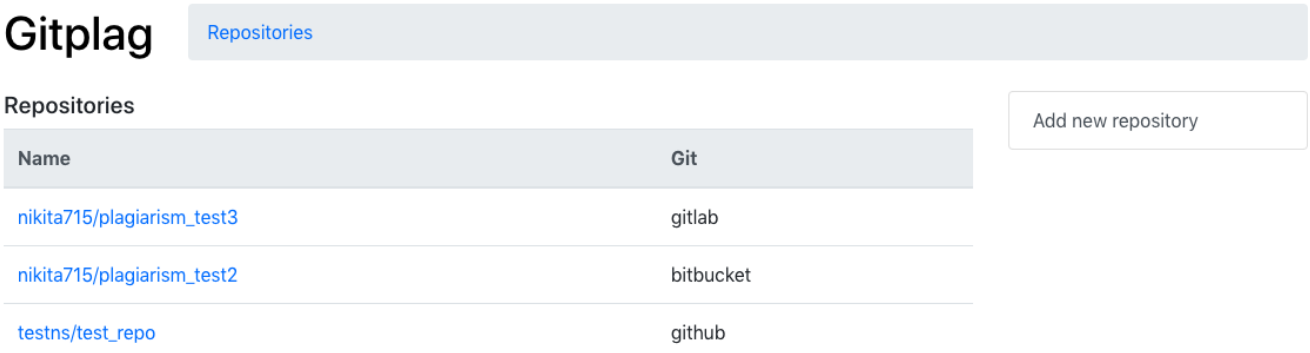

Рисунок 8 - Страница списка подключенных репозиториев

Страница подключения нового репозитория (Рисунок 9) содержит поля выбора средства системы контроля версий, используемых по умолчанию анализатора, параметров анализаторов, а также список регулярных выражений, по которым определяется, должен ли файл репозитория быть загружен в систему.

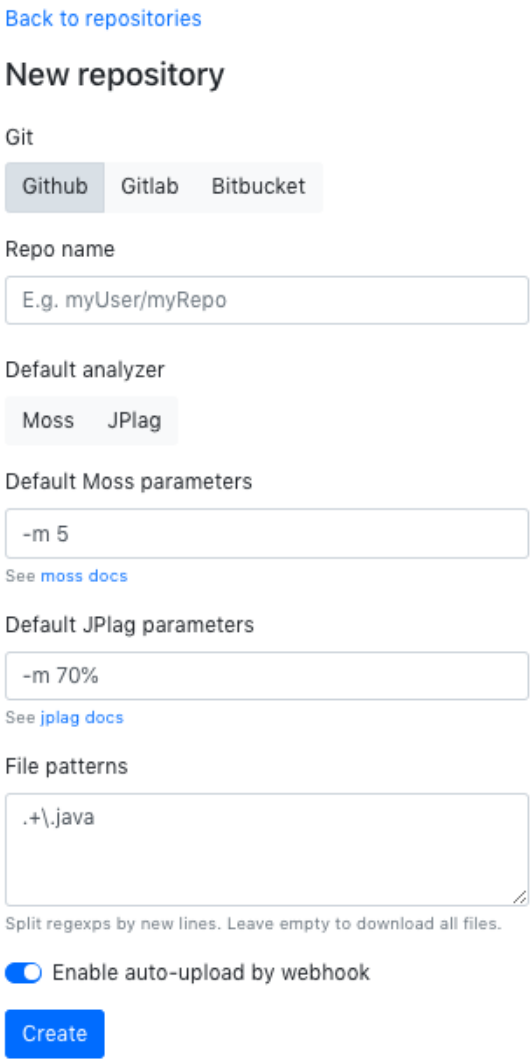

Рисунок 9 - Страница подключения нового репозитория

Страница подключенного репозитория (Рисунок 10) содержит список проведенных анализов репозитория с указанием названия задачи, времени выполнения анализа и использовавшегося анализатора. При нажатии на номер результата анализа происходит переход на его страницу. Также на этой странице можно удалить результат анализа, запустить загрузку файлов из репозитория в систему, перейти на страницы запуска анализа, списка загруженных файлов и изменения репозитория.

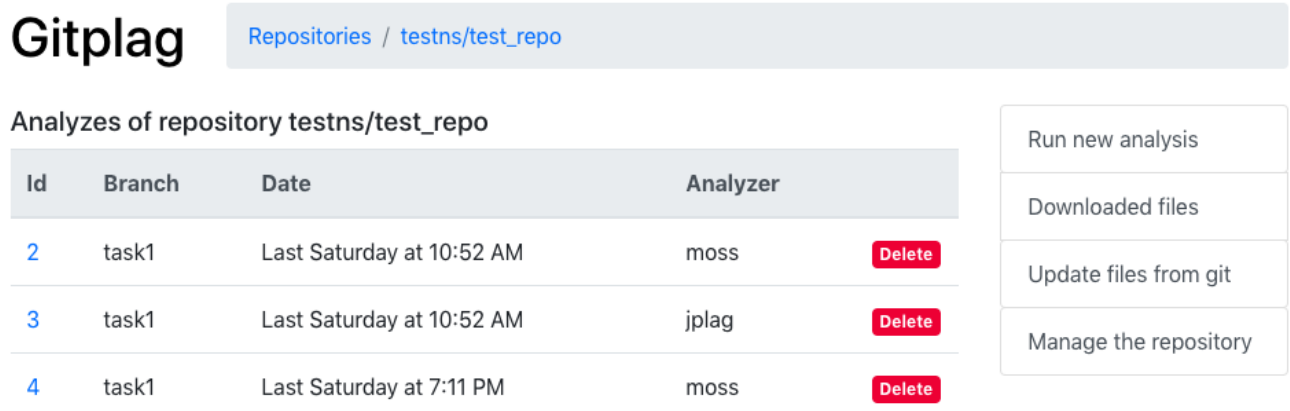

Рисунок 10 - Страница подключенного репозитория

На странице изменения подключенного репозитория (Рисунок 11) можно удалить этот репозиторий, изменить настройки репозитория, используемые при анализе по умолчанию, установить параметр автоматической загрузки файлов репозитория в систему при помощи уведомлений репозитория об изменениях, изменить список регулярных выражений имен файлов.

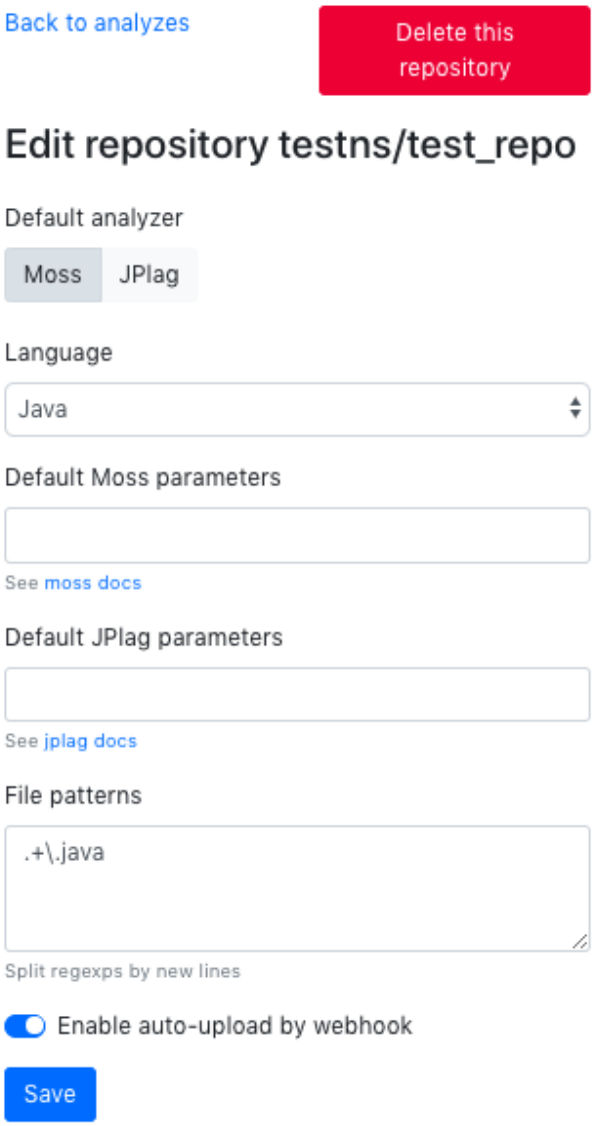

Рисунок 11 - Страница изменения подключенного репозитория

На странице загруженных в систему файлов (Рисунок 12) отображены список файлов репозитория и список файлов решений. В списке файлов репозитория для каждого файла указаны название, название задачи и время последнего обновления файла. В списке файлов решений для каждого файла указаны имя студента, название, название задачи и время последнего обновления файла. Имеется возможность найти интересующий файл при помощи поиска регулярным выражением, а также возможность отсортировать списки по одному из столбцов при нажатии на него.

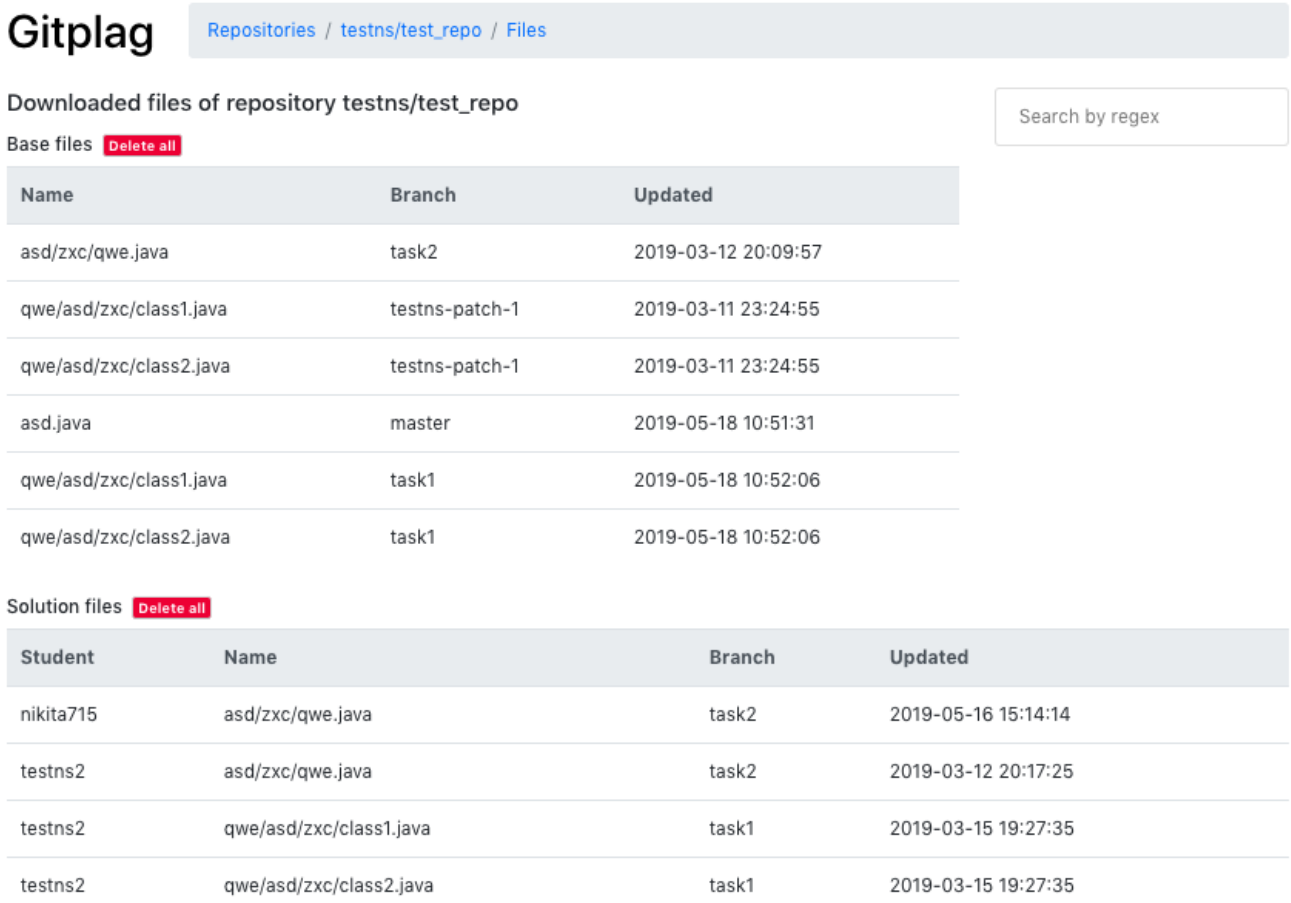

Рисунок 12 - Страница загруженных файлов репозитория в систему

На странице запуска анализа (Рисунок 13) указывается название задачи, используемый анализатор, параметры анализатора и параметр, указывающий системе, необходимо ли дополнительно загрузить файлы репозитория перед анализом.

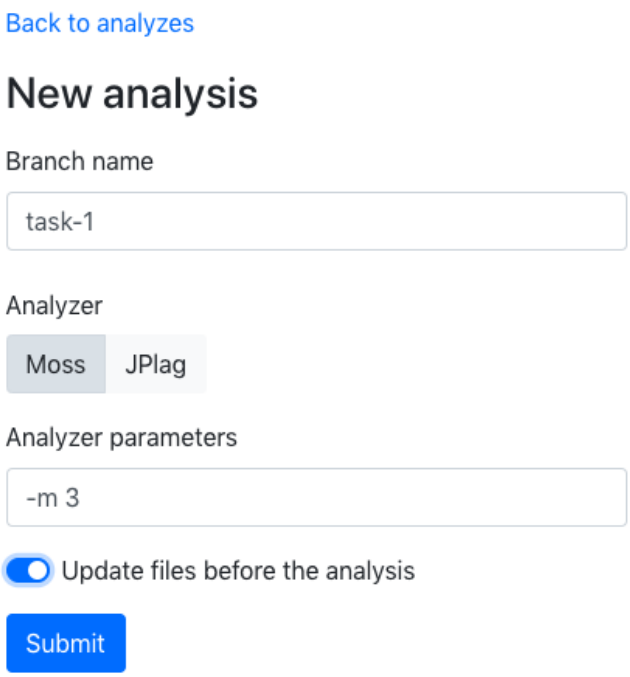

Рисунок 13 - Страница запуска анализа подключенного репозитория

На странице результата анализа (Рисунок 14) отображен список пар студентов, полученный из результата работы системы анализа. В списке указаны имена студентов и процент схожести их решений. По нажатию на номер пары происходит переход на страницу сравнения файлов студентов. Над списком указаны использовавшийся анализатор, название проанализированной задачи, время проведения анализа, ссылка на исходный результат анализа, полученнный от системы анализа схожести исходного кода, ссылка на граф результатов анализа.

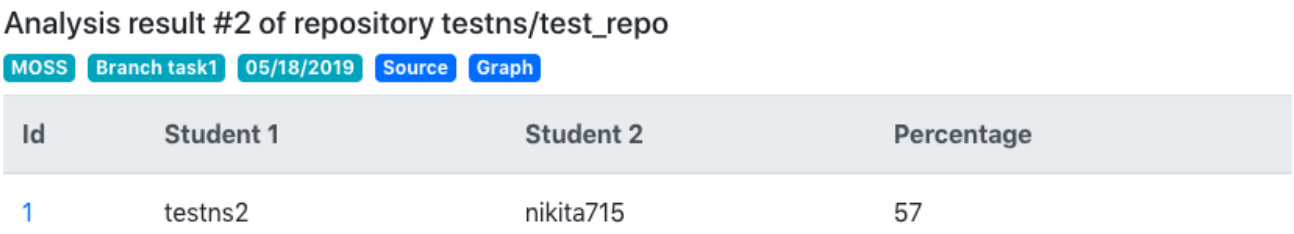

Рисунок 14 - Страница результата анализа

Страница сравнения двух решений (Рисунок 15) содержит исходный код двух проанализированных файлов с указанием номеров строк и указанием позиций совпадений. При нажатии на один из выделенных участков происходит его выравнивание с парным участком. Также при нажатии на номер строки происходит ее выравнивание по верхней границе интерфейса. Вверху страницы указаны имена студентов, время создания решений и процент схожести.

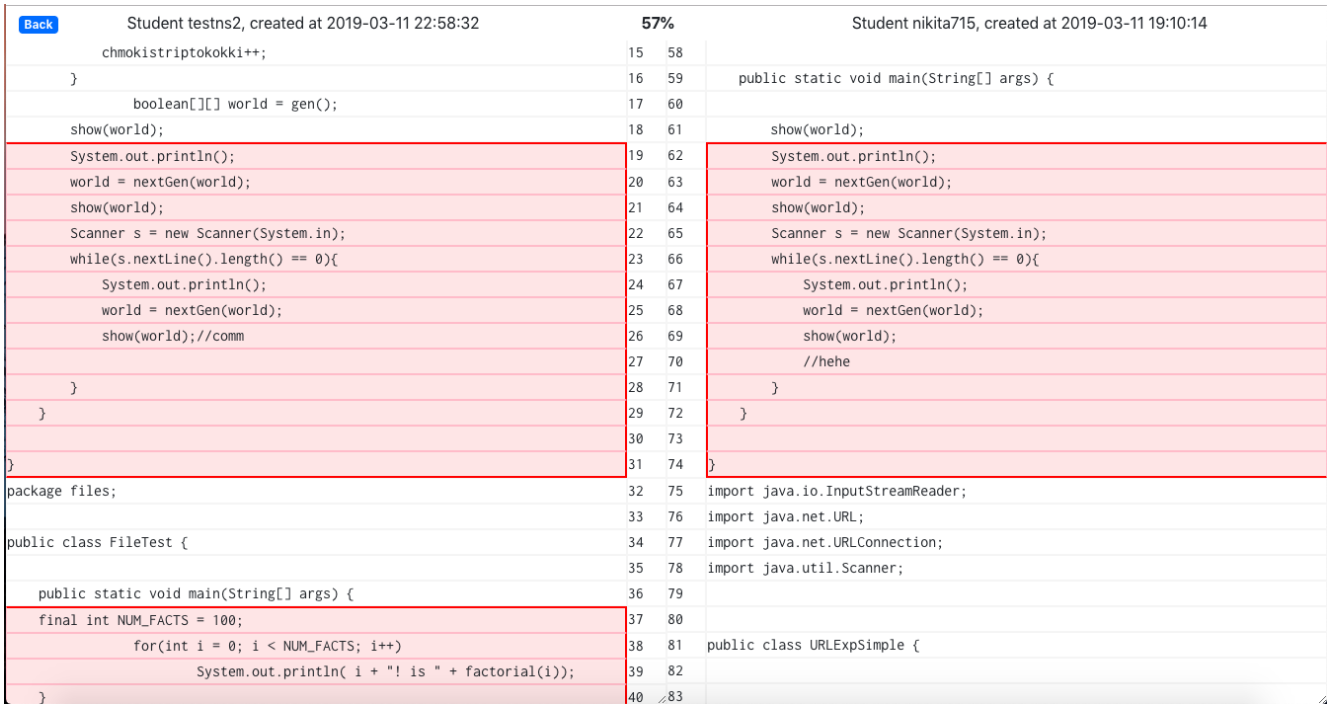

Рисунок 15 - Страница сравнения двух решений

Клиентская часть приложения при помощи библиотеки SockJS устанавливает соединение с сервером по протоколу WebSocket, и сообщения с сервера отображаются на пользовательском интерфейсе. Сообщения отображаются на всех страницах клиента в правом верхнем углу (Рисунок 16).

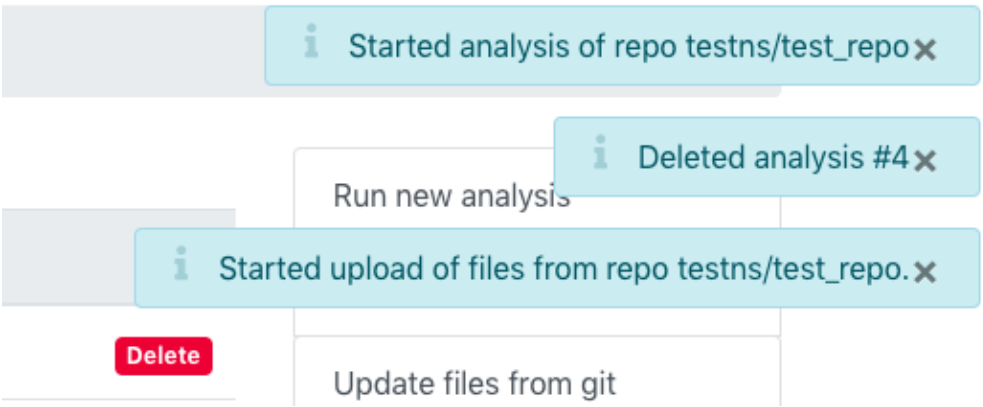

Рисунок 16 – Визуализация сообщений от сервера

3.7 Интерфейс системы data2graph

Система data2graph является системой с открытым исходным кодом, поэтому любой желающий может добавить в нее полезную функциональность при согласии ее владельца. Таким образом в эту систему были добавлены изменения, которые требовались для визуализации результатов анализов из разрабатываемой системы: направленность графа и скрытие неиспользуемых узлов (Рисунок 17, рисунок 18).

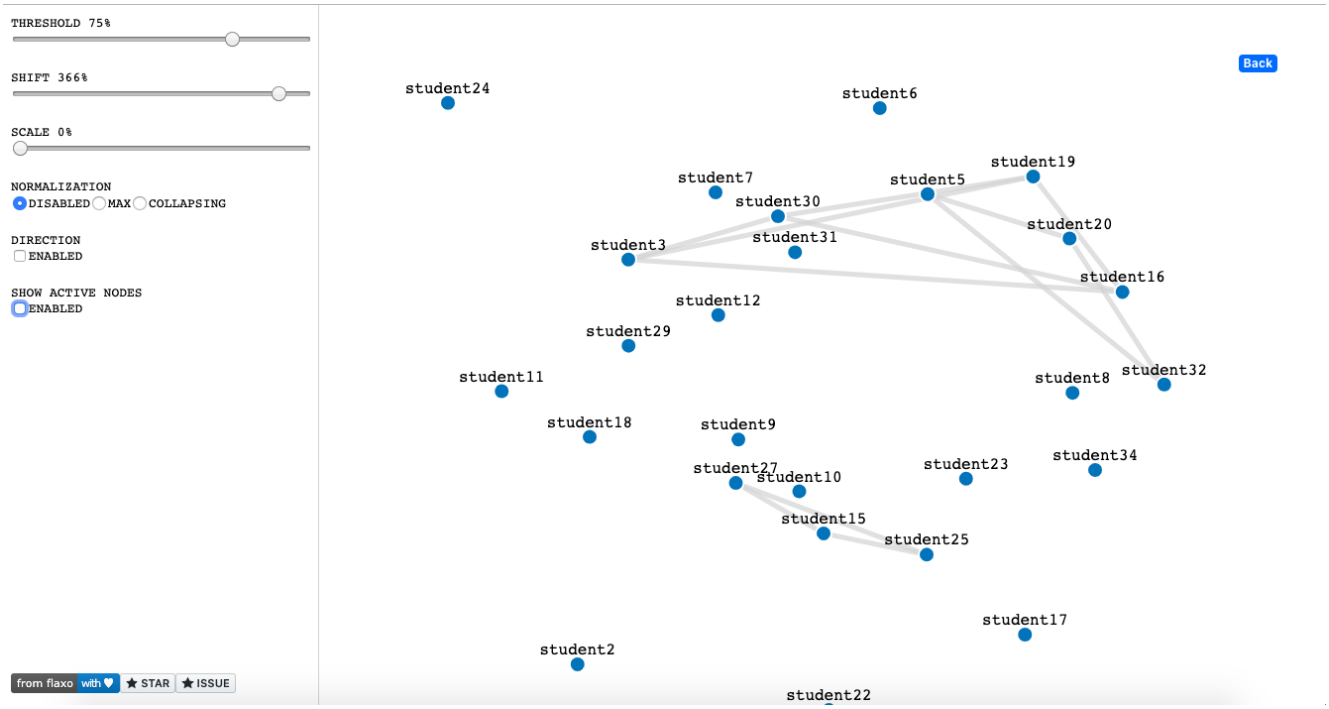

Рисунок 17 – Интерфейс системы data2graph без добавленной функциональности

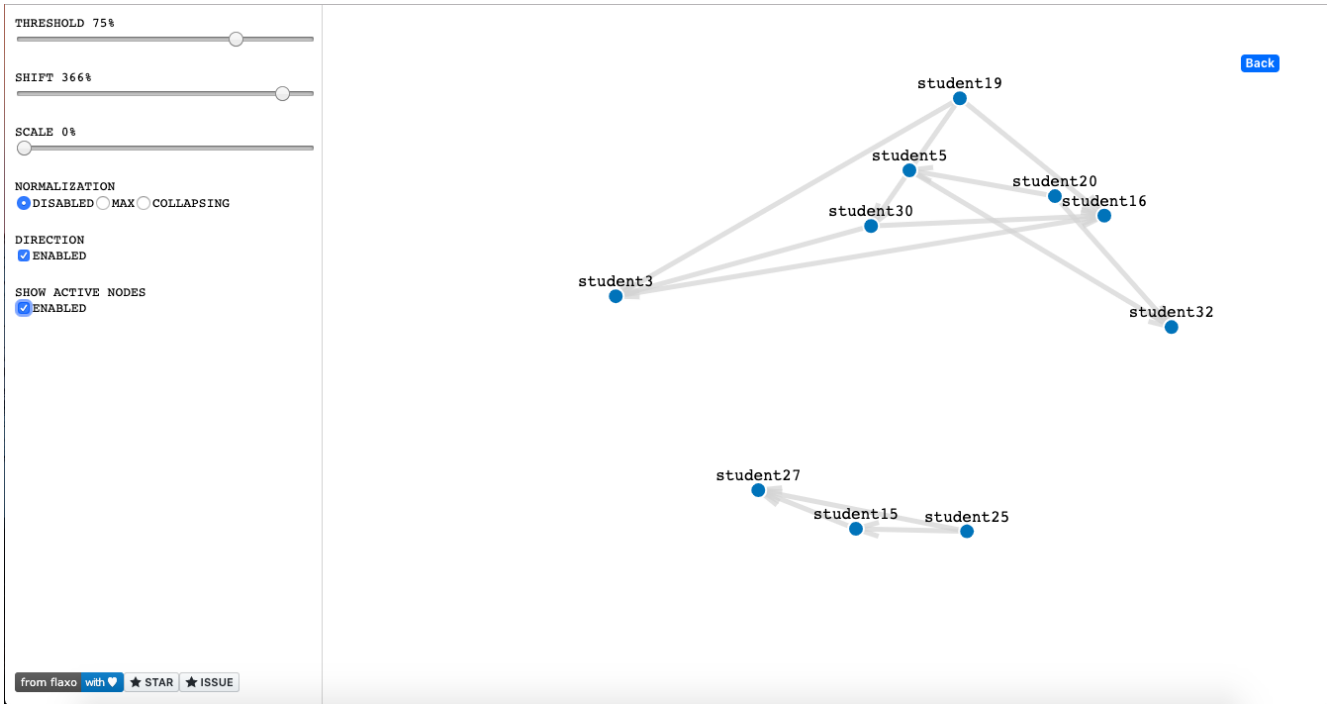

Рисунок 18 – Интерфейс системы data2graph с добавленной функциональностью

4 ТЕСТИРОВАНИЕ

Завершающим этапом создания системы является тестирование системы на реальных данных. Для этого был взят репозиторий, использующийся студентами первого курса для решения практических задач по дисциплине «Программирование». Запись о репозитории была добавлена в разработанную систему, и в репозитории была установлена настройка отправки оповещений системе при помощи Webhook для поддержания загруженных файлов репозитория в постоянно обновленном состоянии. Для репозитория была установлена настройка файлов, путь к которым из корневой директории репозитория начинается с src/main, так как файлы, которые используются для решения задач находятся только внутри этой директории, а тесты, путь к которым начинается с src/test, не изменяются и используются студентами для тестирования собственных решений, поэтому должны игнорироваться системой. После подключения репозитория была запущена загрузка уже имеющихся в репозитории файлов в систему. Далее этого файлы обновлялись при помощи оповещений репозитория.

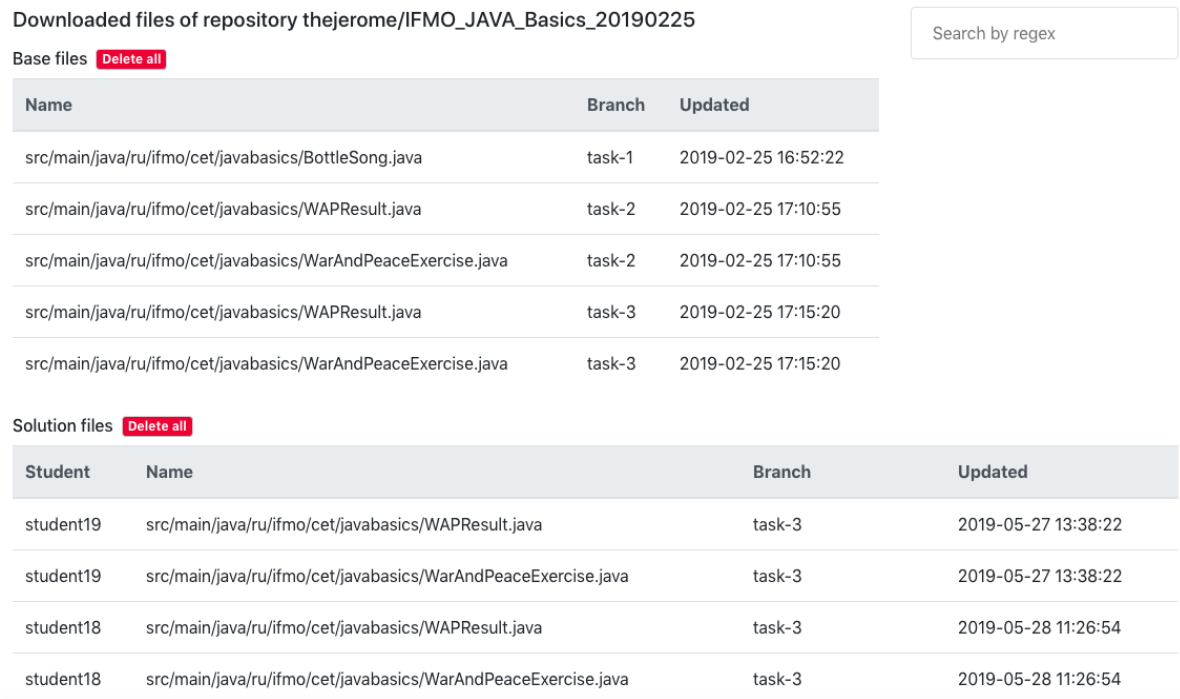

Рисунок 19 – Фрагмент страницы загруженных в систему файлов

Использовавшийся репозиторий содержит в себе три задачи, каждая из которых содержит наборы решений, отправленных разными студентами. Для трех задач был запущен анализ обеими системами анализа (Рисунок 20).

Analyzes of repository thejerome/IFMO_JAVA_Basics_20190225

Id	Branch	Date	Analyzer	
29	task-3	Today at 6:49 PM	moss	Delete
30	task-1	Today at 6:49 PM	moss	Delete
31	task-2	Today at 6:49 PM	moss	Delete
32	task-1	Today at 6:57 PM	jplag	Delete
33	task-2	Today at 6:57 PM	jplag	Delete
34	task-3	Today at 6:57 PM	jplag	Delete

Рисунок 20 – Список проведенных анализов

Анализ задачи при помощи системы Moss показал очень большой уровень схожести решений (Рисунок 21), система JPlag также указала очень большую схожесть (Рисунок 22). При визуальном осмотре пар решений со степенью схожести на уровне 99-100% было обнаружено, что некоторые из этих решений полностью идентичны, остальные отличается только именами переменных, переносами строк и их выравниванием.

Analysis result #30 of repository thejerome/IFMO_JAVA_Basics_20190225

MOSS | Branch task-1 | last sunday at 6:49 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1410	student5	student20	99
1411	student32	student5	99
1412	student32	student20	99
1414	student16	student19	99
1415	student16	student3	99
1416	student3	student19	99
1417	student27	student25	99
1418	student27	student15	99
1419	student15	student25	99
1420	student16	student30	84
1421	student3	student30	84
1422	student30	student19	84

Рисунок 21 – Результаты анализа задачи 1 системой Moss

Analysis result #32 of repository thejerome/IFMO_JAVA_Basics_20190225

JPLAG | Branch task-1 | last sunday at 6:57 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1686	student3	student30	100
1687	student19	student3	100
1688	student16	student3	100
1692	student25	student15	100
1693	student25	student27	100
1697	student5	student32	100
1698	student5	student20	100
1699	student20	student32	100
1700	student19	student30	100
1701	student19	student16	100
1704	student16	student30	100
1707	student9	student31	100

Рисунок 22 – Результаты анализа задачи 1 системой JPlag

На графе результатов анализа задачи 1 системой Moss был установлен порог процента схожести в 80%. В итоге студенты на нем распределены на 3 группы (Рисунок 23).

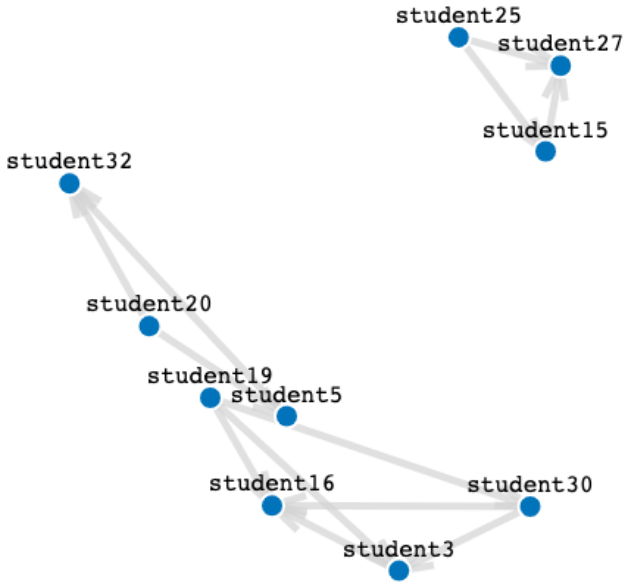

Рисунок 23 – Граф результатов анализа задачи 1 системой Moss

На графе результатов анализа задачи 1 системой JPlag также был установлен порог процента схожести в 80%. В итоге студенты на нем распределены на 5 групп (Рисунок 24).

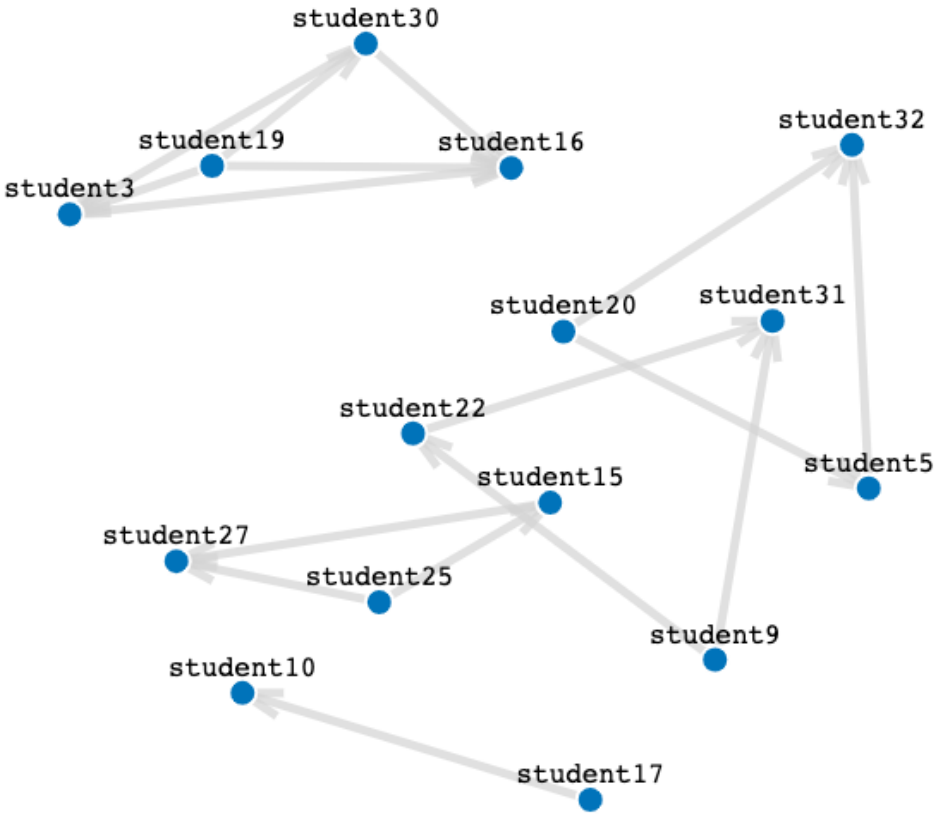

Рисунок 24 – Граф результатов анализа задачи 1 системой JPlag

На обоих графах есть три идентичные группы студентов, что указывает на то, что обе системы анализа верно находят схожие решения задач. Отличия в результатах анализа могут быть связаны с тем, что системы используют разные методы анализа, либо с тем, что системы по-разному рассчитывают проценты схожести.

Результаты анализа задачи два приведены на рисунках 25, 26. По визуальному осмотру нескольких первых сравнений решений этой задачи выяснено, что решения отличаются по размерам пробелов и табуляции, названиям модификаторов, но функциональность кода аналогична.

Analysis result #31 of repository thejerome/IFMO_JAVA_Basics_20190225

MOSS | Branch task-2 | last sunday at 6:49 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1498	student10	student8	83
1499	student2	student20	77
1497	student15	student14	73
1501	student10	student30	67
1502	student8	student30	67
1500	student25	student20	66
1507	student2	student3	65
1503	student27	student12	64
1506	student3	student20	63
1504	student9	student25	62
1505	student9	student20	62
1508	student3	student25	61

Рисунок 25 – Результаты анализа задачи 2 системой Moss

Analysis result #33 of repository thejerome/IFMO_JAVA_Basics_20190225

JPLAG | Branch task-2 | last sunday at 6:57 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1718	student14	student15	100
1721	student10	student8	100
1725	student25	student9	99
1731	student3	student2	97
1734	student12	student27	91
1732	student20	student3	88
1737	student25	student20	88
1726	student9	student20	87
1722	student30	student10	86
1741	student30	student8	86
1743	student20	student2	85
1733	student25	student3	75

Рисунок 26 – Результаты анализа задачи 2 системой JPlag

Графы анализов обоих анализаторов (Рисунки 27, 28) на минимальном уровне в 75% схожи по двум ребрам, но граф системы JPlag отображает намного больше ребер. При уменьшении минимального уровня схожести на графе системы Moss те ребра, которые присутствуют на графе системы JPlag, отображаются на графе системы Moss, что говорит о том, что одна из систем анализа либо приуменьшает, либо преувеличивает процент схожести.

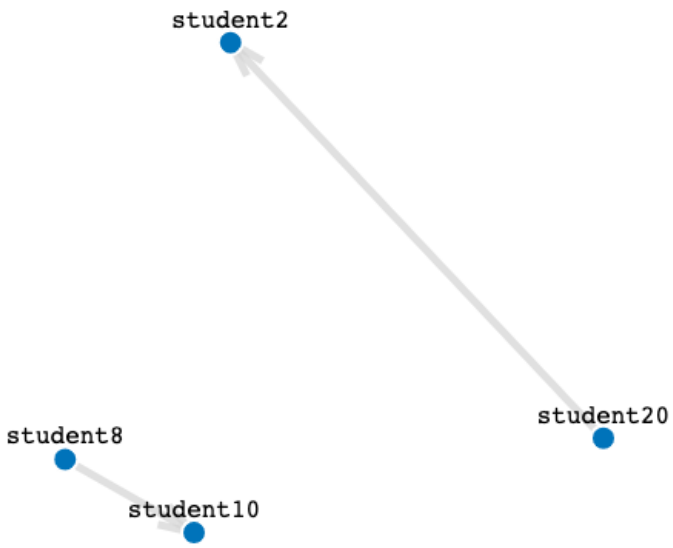

Рисунок 27 – Граф результатов анализа задачи 2 системой Moss

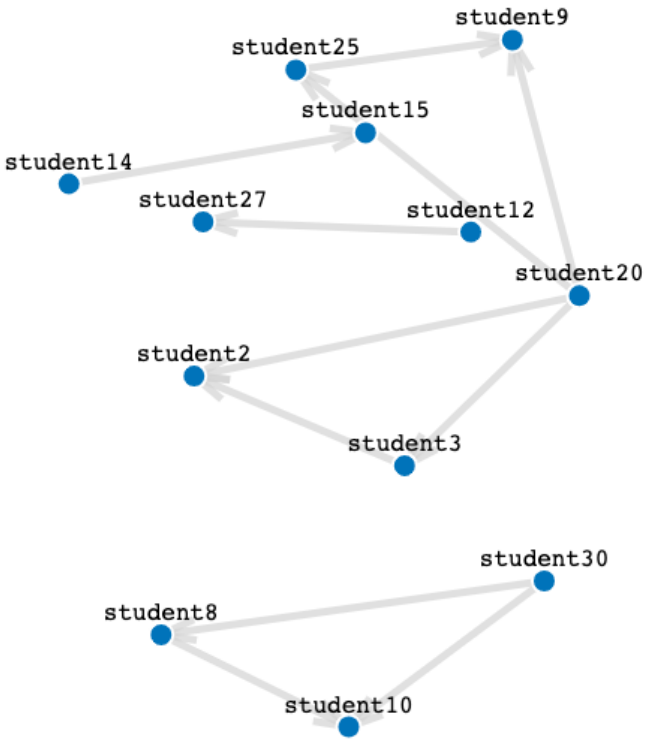

Рисунок 28 – Граф результатов анализа задачи 2 системой JPlag

В результате анализа задачи номер три было найдено только одна пара решений с высокой степенью схожести (Рисунки 29, 30). Исходный код этих решений отличался только по тому, что в одном из них отсутствовали отступы в началах строк. Хотя решения полностью схожи, система Moss указывает 84% схожести, в отличие от системы JPlag, которая указывает 100%. Можно сделать вывод о том, что именно система Moss преуменьшает процент схожести.

Analysis result #29 of repository thejerome/IFMO_JAVA_Basics_20190225

MOSS | Branch task-3 | last sunday at 6:49 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1382	student24	student26	84
1383	student24	student29	67
1384	student26	student29	67
1385	student25	student26	65
1387	student24	student25	65
1388	student25	student29	55
1386	student27	student30	51
1403	student28	student25	30
1389	student25	student27	27
1404	student28	student23	24
1405	student28	student27	24
1406	student28	student30	24

Рисунок 29 – Результаты анализа задачи 3 системой Moss

Analysis result #34 of repository thejerome/IFMO_JAVA_Basics_20190225

JPLAG | Branch task-3 | last sunday at 6:57 pm | Source | Graph

Id	Student 1	Student 2	Percentage
1749	student24	student26	100
1750	student25	student24	67
1755	student25	student26	67
1751	student24	student29	64
1756	student29	student26	64
1760	student25	student29	55
1761	student23	student29	51
1764	student27	student30	39
1752	student23	student24	36
1757	student23	student26	36
1767	student25	student23	35
1765	student23	student30	25

Рисунок 30 – Результаты анализа задачи 3 системой JPlag

Графы результатов анализов задачи три (Рисунки 31, 32) на уровне 50% схожи по четырем из пяти узлов и шести из семи ребер.

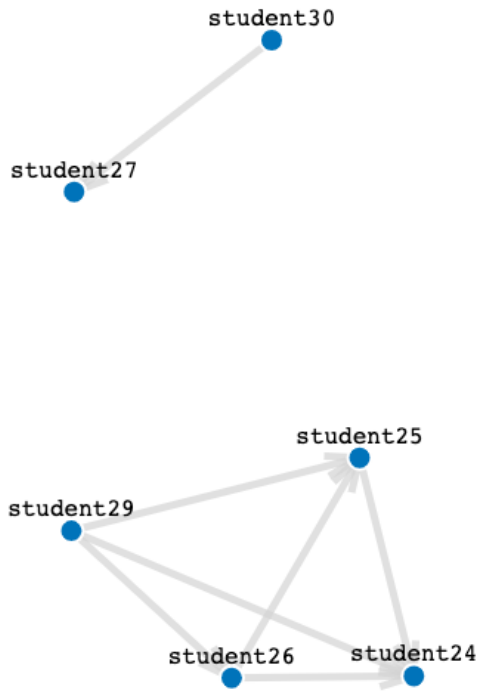

Рисунок 31 – Граф результатов анализа задачи 3 системой Moss

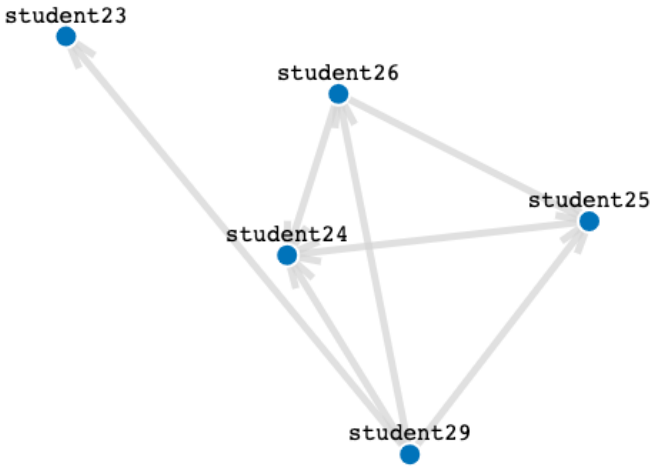

Рисунок 32 – Граф результатов анализа задачи 3 системой JPlag

ЗАКЛЮЧЕНИЕ

В результате выполнения данной дипломной работы была спроектирована, разработана и протестирована автоматизированная система анализа схожести исходного кода решений задач по программированию в открытых образовательных репозиториях, связывающая между собой существующие системы управления репозиториями с системами анализа схожести исходного кода, автоматизирующая процессы подготовки и отправки на анализ решений задач, предоставляющая удобную визуализацию результатов анализов в виде списка пар решений с указанием авторства и степеней схожести, в виде сравнения пар решений с указанием схожих участков и в виде направленного ациклического графа.

Разработанная система была протестирована на реальных данных, и из результатов анализа сделан вывод о том, что система функционирует так, как и планировалось, также были сделаны выводы о том, как функционируют системы анализа и чем отличаются результаты их работы.

Разработанная система может использоваться для сравнения решений как самостоятельная система, так и как сторонняя система для систем, у которых есть потребность в анализе решений из образовательных репозиториев на предмет схожести. Разрабатывается план внедрения данной системы в систему Flaxo.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения:

SPA – Single Page Application

REST – Representational State Transfer

JVM – Java Virtual Machine

API – Application programming interface

СПИСОК ЛИТЕРАТУРЫ

1. Ho, Clarence, Harrop, Rob, Schaefer, Chris. Pro Spring // Apress. - 2014. - 728 с.
2. Dmitry Jemerov, Svetlana Isakova. Kotlin in Action // Manning. - 2017. – 360 с.
3. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship // Prentice Hall. - 2008. – 464 с.
4. Joshua Bloch. Effective Java // Addison-Wesley Professional. - 2018. - 412с.
5. Scott Chacon, Ben Straub. Pro Git // Apress. - 2014. – 456 с.
6. Spring Framework Documentation. URL: https://docs.spring.io/autorepo/docs/spring-framework/5.0.0.M1/spring-framework-reference/pdf/spring-framework-reference.pdf (дата обращения: 07.04.2019). - 855 с.
7. Kotlin Language Documentation. URL: https://kotlinlang.org/docs/kotlin-docs.pdf (дата обращения: 07.04.2019). - 491с.
8. Saul Schleimer, Daniel S. Wilkerson, Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. - 2003. - 10 с.
9. A. Omar Portillo-Dominguez, Vanessa Ayala-Rivera, Evin Murphy, John Murphy. A Unified Approach to Automate the Usage of Plagiarism Detection Tools in Programming Courses. – 2017. – 6 с.
10. Lutz Prechelt, Guido Malpohl, Michael Philippsen, JPlag: Finding plagiarisms among a set of programs. – 2000. – 44 с.