Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б10-мм

Интеграция с GitHub в HwProj 2.0.1

Позиев Алексей Владимирович

Отчёт по учебной практике
в форме «Производственное задание»

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

Консультант:
инженер-программист А. В. Бережных

Санкт-Петербург
2024

Оглавление

Введение

На большинстве курсов по программированию и информатике важнейшими элементами эффективного обучения являются домашние задания и доступность материалов курса. Эти аспекты создают организационные сложности, такие как: публикация задач, их сдача, выставление оценок и контроль дедлайнов. Для решения этих проблем был разработан HwProj 2.0.1 — веб-сервис с микросервисной архитектурой, который предназначен для управления сдачей и проверкой домашних заданий и учитывает специфические потребности курсов кафедры системного программирования СПбГУ.

В силу того, что HwProj в основном используется в курсах, связанных с программированием, большая часть решений размещается на GitHub. Но, к сожалению, несмотря на его частое использование, HwProj 2.0.1 никак не использует его возможности. В связи с этим появляются некоторые недостатки сервиса.

Первым существенным недостатком является то, что на данный момент нет удобного механизма отправки решения задачи от лица студентов курса через API сервиса. Это, в случае курса Программной Инженерии, в котором задания сдаются в GitHub Classrooms и проходят тестирование решения через систему непрерывной интеграции, приводит к тому, что некоторые студенты вручную сдают неверные решения, которые могут быть достаточно легко найдены с помощью автоматической проверки, на сервисе HwProj. Поэтому преподаватели, которые начинают проверку работ, часто заходят на страницы неверных решений, что тратит их время и силы.

Следующей долгосуществующей проблемой является проверка своевременной сдачи контрольных работ в сервисе. Студенты часто сдают неготовые или пустые решения, опубликованные на GitHub, и уже позже добавляют новые коммиты, меняя системное время своей системы. Это приводит к тому, что студенты нечестным образом улучшают свои оценки в контрольных работах, а преподавателям достаточно тяжело и времязатратно это контролировать.

В сервисе HwProj также часто возникают баги и поступают пожелания пользователей по поводу новой функциональности. Баги необходимо исправлять, а пожелания — реализовывать. Это повышает качество работы сервиса и повышает его популярность среди преподавателей и студентов.

Данная работа посвящена интеграции HwProj 2.0.1 с GitHub и улучшению уже существующей функциональности.

1. Постановка задачи

Целью работы является интеграция сервиса HwProj 2.0.1 с GitHub и улучшению уже существующей функциональности.

1. Спроектировать функциональность для отправки решений задач от лица студентов из GitHub. Декомпозировать задачу и распределить части с другими разработчиками сервиса.

2. Реализовать подзадачи на стороне бэкенда и фронтенда HwProj 2.0.1.

3. Добавить функциональность по отслеживанию коммитов из GitHub в сданных контрольных работах в сервисе.

4. Исправить дефекты клиентской части и добавить новую функциональность, основанную на запросах пользователей, возникших по мере выполнения основных задач.

2. Обзор

2.1. Обзор используемых технологий

Перечисленные технологии до начала работы уже применялись в сервисе HwProj 2.0.1 и не изменялись.

- C# [2] — основной язык серверной части HwProj 2.0.1.
- ASP.NET Core [1] — кроссплатформенный фреймворк для разработки пользовательских веб-сервисов.
- Entity Framework Core [3] — объектно-ориентированная, расширяемая и легковесная технология для доступа к данным.
- Swagger [6] — набор инструментов, который позволяет автоматически описывать API на основе его кода.
- TypeScript [7] — сильнотипизированный язык на основе JavaScript. Основной язык клиентской части HwProj 2.0.1.
- React [5] — библиотека для создания пользовательского интерфейса с помощью компонентов.

2.2. Существующие решения

Обзор существующих решений проводился в целях определения преимуществ и недостатков интеграции с GitHub и возможностей автоматической сдачи работ для формирования наиболее удобной реализации.

2.2.1. Blackboard Learn

Blackboard Learn [8] — платформа предоставляет единую интерактивную среду для обучения и взаимодействия между обучаемыми или студентом и преподавателем. Имеет гибкий набор инструментов для управления курсами и домашними заданиями.

Публичное API Blackboard Learn не позволяет сдавать решения от лица студента, но Blackboard Learn поддерживает протокол LTI 1.3, позволяющий интегрироваться с GitHub Classrooms.

2.2.2. Stepik

Stepik [10] — образовательная платформа и конструктор онлайн-курсов.

Stepik аналогично Blackboard Learn поддерживает протокол LTI 1.3 и не позволяет сдавать решения от лица студента через публичный API. Присутствует возможность авторизации через GitHub.

2.2.3. Google Classrooms

Google Classrooms [9] — это платформа для управления курсами и взаимодействия между преподавателями и студентами. Обеспечивает удобный интерфейс для создания, распределения и оценки заданий, а также для общения и обмена материалами в рамках курса.

Google Classrooms в свою очередь не поддерживает LTI, а предоставляет необходимые инструменты в своем API для интеграции с другими сервисами.

2.3. Обзор архитектуры HwProj 2.0.1

Серверная часть HwProj 2.0.1 разделена на четыре микросервиса:

- AuthService отвечает за аутентификацию пользователей;
- NotificationsService отвечает за уведомления;
- SolutionsService — сервис, предназначенный для работы с решениями задач студентов в курсе, в частности публикации, выставления оценок;
- CoursesService — сервис, предназначенный для создания и редактирования курсов, а также содержащихся в них домашних заданий и задач.

Для связи между ними используется брокер сообщений RabbitMQ [4]. Общение между бэкендом и фронтендом происходит через API Gateway — REST API сервиса. С архитектурой проекта можно ознакомиться на Рис. 1.

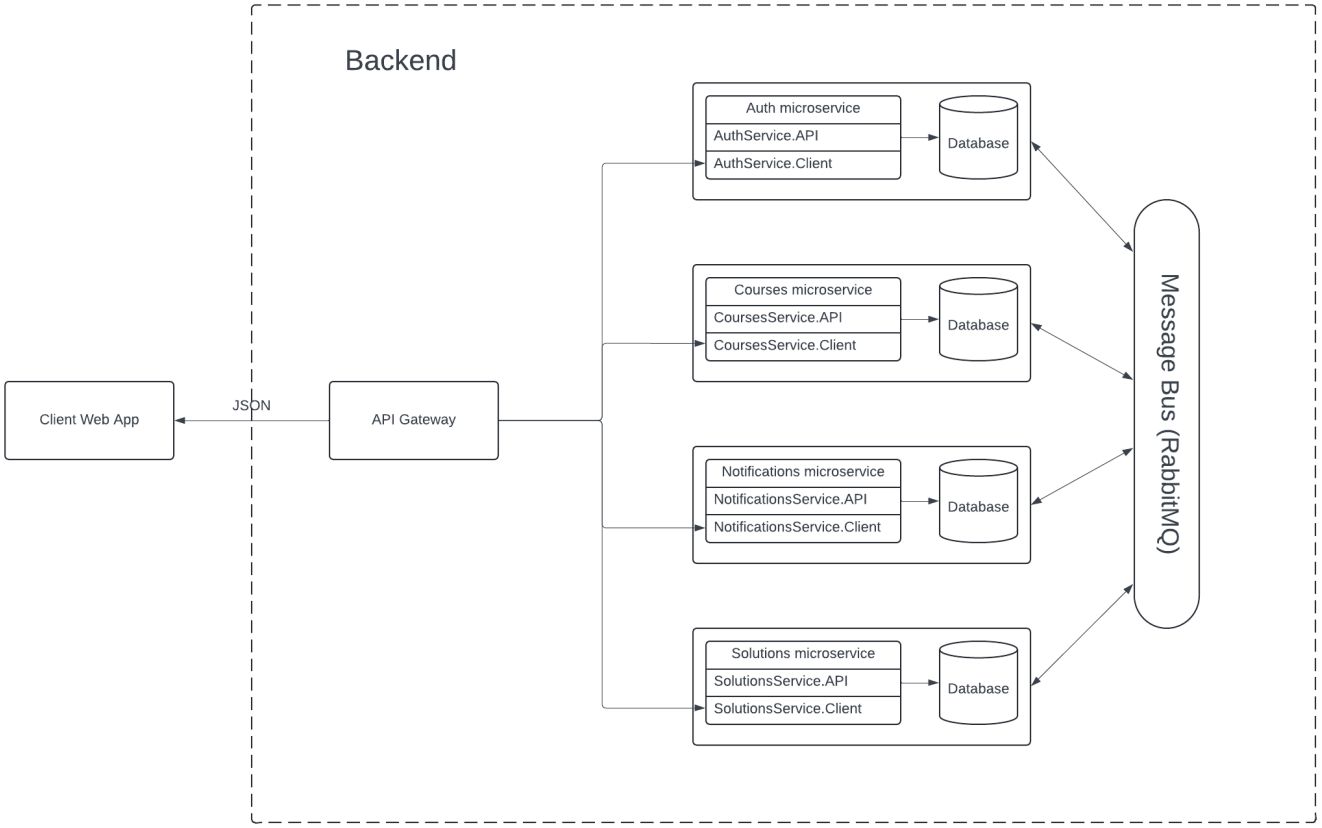

Рис. 1: Архитектура серверной части

3. Реализация

3.1. Проектирование функциональности отправки решений задач от лица студентов

Первоначальной идеей реализации была поддержка протокола LTI 1.3, что позволило бы обмениваться данными о решениях, студентах и курсах не только с GitHub Classrooms, но и с другими сервисами, поддерживающими LTI. Однако, от данной идеи пришлось отказаться, так как она решает задачи намного шире изначальной проблемы, требует большого объема работы в рамках одного семестра, а также могут возникнуть сложности в дальнейшем сопровождении кода, связанные с необходимостью ознакомления разработчиков с данным протоколом. Поэтому было решено найти более быстрое решение, которое также смогло бы решить необходимую задачу.

В связи с решением не поддерживать протокол LTI, очевидным решением стало предоставление необходимых инструментов со стороны API сервиса HwProj 2.0.1 для отправки решений задач от лица студентов. Это позволит другим сервисам, таким как GitHub, публиковать решения студентов только в ситуациях, когда их корректность и готовность к проверке подтверждены. Например, в случае с GitHub Classrooms решения будут отправляться только после прохождения всех тестов.

Задача декомпозируется на следующие подзадачи:

1. Добавить возможность идентификации студента по его GitHub логину.

2. Добавить шаблон команды для отправки решений в HwProj 2.0.1 из GitHub Actions.

3. Поддержать новую функциональность на стороне фронтенда.

4. Добавить новый эндпоинт для публикации решений от лица студента, включая специфику групповых решений.

5. Создание JWT-токенов для авторизации автоматической сдачи решений.

Так как задача выполнялась в группе из двух человек, то моей задачей стала реализация первых трех пунктов из списка подзадач.

3.2. Реализация подзадач на стороне бэкенда и фронтенда

3.2.1. Идентификация студента по логину GitHub

Идентификация студента происходит по логину, а не ID пользователя, так как логин, хоть и не постоянное значение, но часто используется в названиях репозиториев, пулл реквестов, из которых его можно парсить и отправлять сервису. Но сервис не содержал данных о GitHub логинах пользователей, поэтому появилась необходимость добавить возможность привязать свой GitHub логин к аккаунту сервиса HwProj.

Первоначальной идеей было добавить текстовое поле, в которое студенты могли бы написать свой логин. Но сразу стали видны недостатки:

- Человек может ошибиться при написании своего логина;
- Пользователь может написать чужой логин и получать не свои решения.

Поэтому было решено добавить необходимость авторизации в GitHub при помощи OAuth App, что позволит получать актуальный логин пользователя. Также данная реализация позволяет дополнительно сохранять GitHub ID пользователя, который будет использоваться для авторизации в сервисе HwProj через GitHub. Авторизация происходит со скоупом *user:read*, что гарантирует пользователям сервис, что HwProj не сможет получить никакой информацию, кроме публичных данных профиля.

Так как для работы с OAuth App необходимы секреты, то их нужно где-то хранить. Все секреты OAuth App сервиса HwProj хранятся на стороне бэкенда. В продакшене секреты получаются из окружения

процесса, а при локальной разработке секреты добавляются в файл *secrets.json*, команда для добавления описана в документации проекта.

Возможность привязки GitHub аккаунта была добавлена на стороне фронтенда на страницу редактирования данных пользователя (Рис. 2).

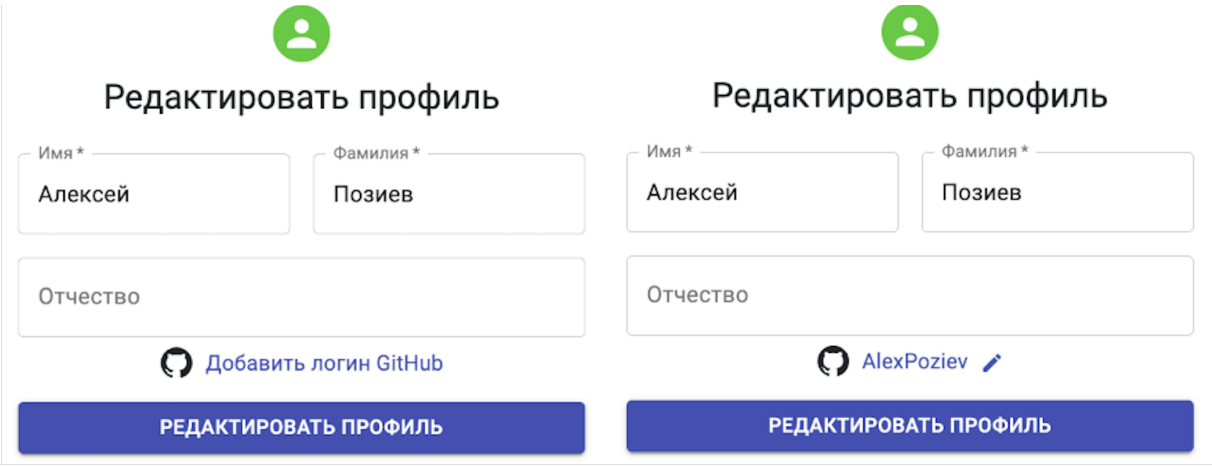

Рис. 2: GitHub

3.2.2. Создание шаблона команды для отправки решений в HwProj 2.0.1 и выдача для публикации автоматических решений

Шаблон был реализован на bash с использованием команды curl. Клиентская часть сервиса самостоятельно, при выдаче шаблона, подставляет ID задачи, для которой хотят получить шаблон. Чтобы получить шаблон, необходимо нажать на логотип GitHub в задаче (Рис. 3).

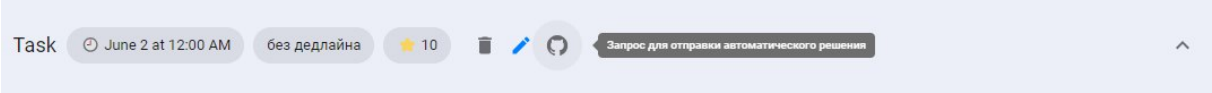

Рис. 3: GitHub логин

Шаблон команды посылает запрос в HwProj для публикации решения, получает код запроса и, в случае успеха, продолжает работу (Рис. 4).

bash

```
response=$(curl --location -i -s -o /dev/null
-w "%{http_code}" 'http://localhost:3000/api/Solutions/automatic'
--header 'Content-Type: application/json'
--header 'Authorization: Bearer $Token'
--data-raw '{
    "GithubId": $GITHUB_LOGIN,
    "SolutionUrl": $SOLUTION_URL,
    "TaskId": 1
}')
if [[ ${response:0:1} != "2" ]]; then
    echo "Failed"
    exit 1
else
    echo "Success"
fi
```

Рис. 4: bash команда

Для локального тестирования того, что шаблон корректно работает в GitHub Actions, был создан тестовый репозиторий с настроенным CI. Для того, чтобы запросы из GitHub могли приходить в локально запущенный сервис, использовалась утилита «ngrok».

3.2.3. Уведомление о записи на групповое решение на стороне фронтенда и получение токена

В силу того, что задачи сдаются автоматически, студенты не могут создать группу при сдаче решения, поэтому это необходимо делать уже после сдачи работы. В подобных случаях студент может забыть прикрепиться к групповому решению. Чтобы избежать таких случаев,

был добавлен новый компонент *GroupWorkTasks* (Рис. 5). Компонент получает уже подготовленный серверной частью список командных задач, в которых студент не состоит ни в одной команде.

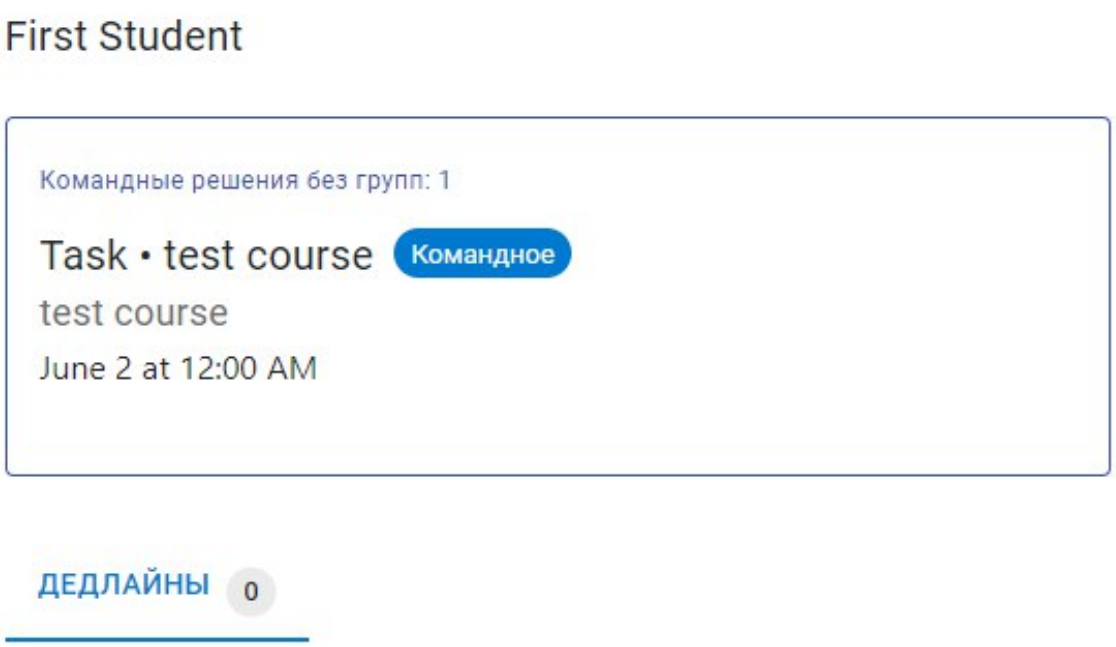

Рис. 5: Уведомление о групповом решении

Для получения токена, который позволяет сдавать решения задач от лица студентов курса, и показа шаблона команды был создан компонент *CodeWindow* (Рис. 6).

Рис. 6: Получение токена

3.3. Отслеживание коммитов из GitHub в сданных контрольных работах сервиса

Для проверки актуальности последнего коммита сданного решения необходимо их где-то сохранять. Для этого была создана новая сущность *GithubSolutionCommit* и соответствующая ей таблица в базе данных. *GithubSolutionCommit* состоит из *SolutionId* и хэша последнего коммита на момент сдачи решения. Этого достаточно, так как даже если будет добавлено новое решение с тем же пулл реквестом, *SolutionId* будет отличаться.

Данные о пулл реквесте берутся из ссылки на сданное решение при помощи регулярного выражения, после чего серверная часть HwProj делает запрос к API GitHub для получения информации о коммитах и берет хэш последнего из них.

Так как обращение к API другого сервиса — достаточно долгая операция, то было принято создать отдельный эндпоинт *GetSolutionActuality* для проверки актуальности решения, а не делать проверку для каждого возвращаемого из сервиса решения. Эндпоинт возвращает *ActualityDto*, в котором содержится информация о том, актуально ли решение, и соответствующее сообщение, если студент добавил новые коммиты или полностью изменил историю коммитов.

Также потребовалось сделать изменения в клиентской части сервиса. На странице решений делается запрос в эндпоинт для проверки актуальности последнего решения и в случае нарушений оповещает преподавателя (Рис. 7).

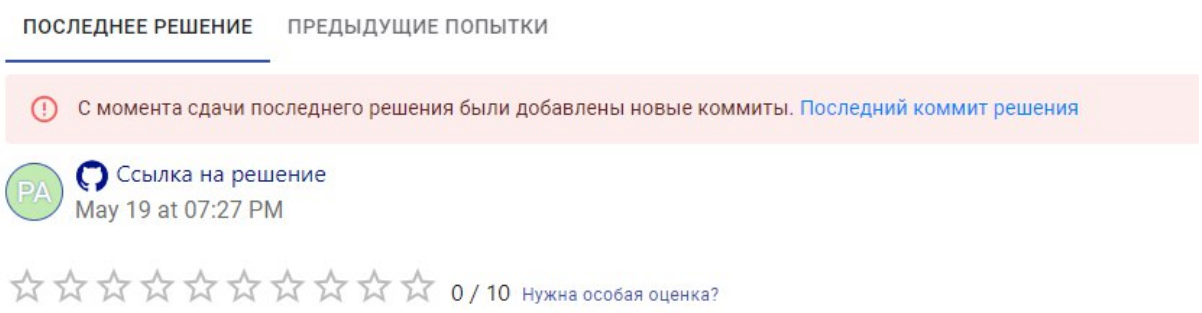

Рис. 7: Уведомление об изменении истории коммитов

3.4. Исправление дефектов клиентской части и добавление новой функциональности, основанной на запросах пользователей

Небольших исправлений и нововведений в клиентской части сервиса в ходе работы над основными задачами было достаточно много:

- исправление верстки страницы редактирования курса;
- исправление верстки страницы редактирования домашних заданий и задач;
- добавление возможности просмотра максимального количества баллов за задачу на странице дедлайнов;
- добавление минимальной ширины ячеек таблицы решений для избежания слишком «тонких ячеек»;
- добавление удобного способа выгрузки всех студентов курса.

По причине их большого количества, будет описана только часть изменений.

3.4.1. Исправление страницы редактирования курса

По мере реализации выдачи токенов для автоматической сдачи решений было обнаружено, что страница редактирования курса находится в плачевном состоянии: компоненты располагались в разных концах страницы, все компоненты съезжали вниз при открытии списка преподавателей курса (Рис. 8).

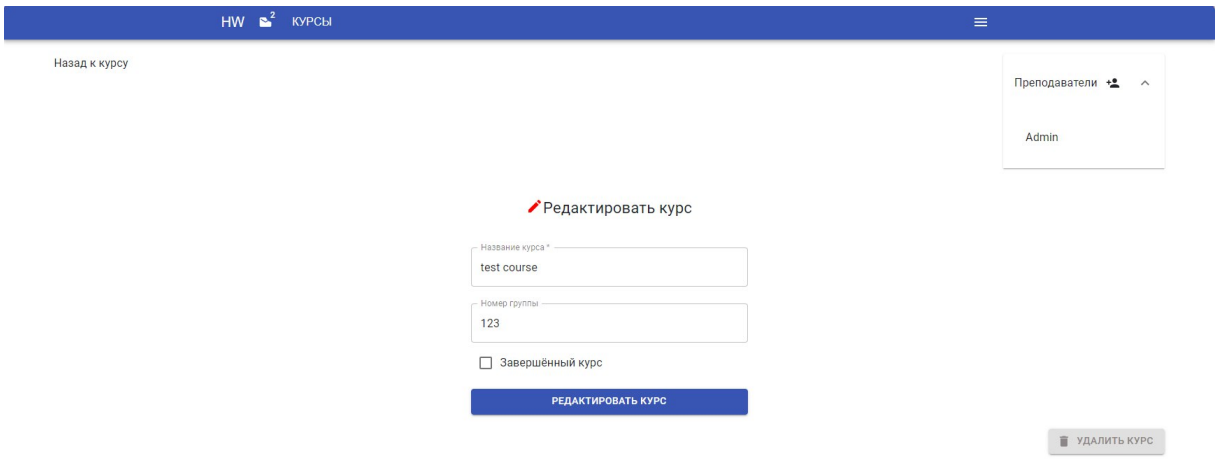

Рис. 8: Старая версия страницы редактирования курса

В силу того, что данная страница достаточно важная и ей будут пользоваться еще чаще для получения токенов, было решено исправить данные недочеты (Рис. 9).

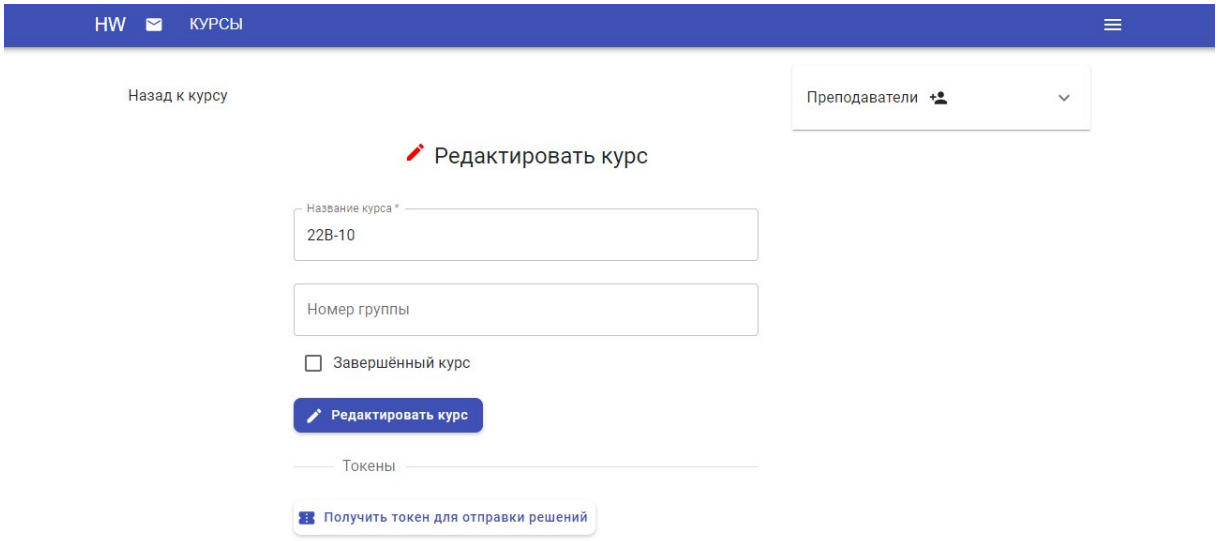

Рис. 9: Новая версия страницы редактирования курса

3.4.2. Выгрузка всех студентов курса

Задача выполнялась в рамках GitHub issue[1].

Первоначальной, уже выполненной реализацией стало переиспользование модального окна *CodeWindow*, которое уже было реализовано при интеграции с GitHub, добавив в него возможность редактирования текста. Но данное решение имело недостаток — новая ответственность компонента модального окна в виде возможности редактирования текста, которую можно перенести на конечный инструмент, в который копируются данные о студентах.

Поэтому вместо изменений существующих компонентов была добавлена кнопка для копирования списка студентов из таблицы решений (Рис. 10).

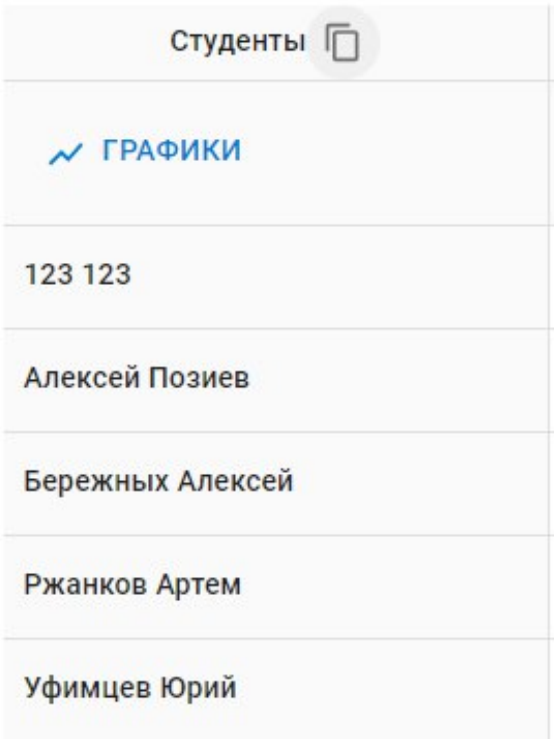

Рис. 10: Копирования студентов курса

[1] https://github.com/InteIligeNET/HwProj-2.0.1/issues/399 (дата доступа: 1 июня 2024 г.).

4. Апробация

Большая часть пулл реквестов уже интегрированы в сервис. Пулл-реквест добавления эндпоинта для сдачи решений от лица студентов готов для слияния с основной веткой, однако было решено отложить интеграцию до окончания зачетов. Его апробация была проведена на консультанте с целью демонстрации работы по публикации решений студентов вне сервиса HwProj 2.0.1.

Заключение

В рамках данной работы были выполнены следующие задачи:

- Спроектирована и декомпозирована функциональность для отправки решений задач от лица студентов из GitHub.
- Реализованы подзадачи на стороне бэкенда и фронтенда HwProj 2.0.1.
- Добавлена возможность отслеживания коммитов из GitHub в сданных контрольных работах в сервисе.
- Исправлены дефекты клиентской части и добавлены новая функциональность, основанная на запросах пользователей, возникших по мере выполнения основных задач.

С кодом реализации можно ознакомиться в списке пулл реквестов за данный семестр.

Список литературы

[1] ASP.NET Core. — URL: https://dotnet.microsoft.com/en-us/learn/aspnet (дата обращения: 28 мая 2024 г.).

[2] C#. — URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата обращения: 5 2024 г.).

[3] Entity Framework Core. — URL: https://learn.microsoft.com/en-us/ef/core (дата обращения: 28 мая 2024 г.).

[4] RabbitMQ. — URL: https://www.rabbitmq.com (дата обращения: 28 мая 2024 г.).

[5] React. — URL: https://react.dev (дата обращения: 28 мая 2024 г.).

[6] Swagger. — URL: https://swagger.io/ (дата обращения: 28 мая 2024 г.).

[7] Typescript. — URL: https://www.typescriptlang.org (дата обращения: 28 мая 2024 г.).

[8] Сайт Blackboard Learn. — URL: https://bb.spbu.ru/ (дата обращения: 5 2024 г.).

[9] Сайт Google Classrooms. — URL: https://sites.google.com/view/classroom-workspace/ (дата обращения: 1 июня 2024 г.).

[10] Сайт Stepik. — URL: https://stepik.org/ (дата обращения: 28 мая 2024 г.).