Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б10-мм.

Интеграция с GitHub в HwProj 2.0.1

Бахориков Егор Олегович

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

В обучающих системах ключевую роль играют домашние работы. Неудобные механизмы публикации, сдачи и проверки задач создают организационные трудности для всех сторон образовательного процесса. Для решения этих проблем был разработан HwProj 2.0.1 [5] — сервис взаимодействия студентов и преподавателей, который создавался с целью упрощения процесса сдачи и проверки домашних заданий, учитывая специфические потребности курсов кафедры системного программирования СПбГУ.

В силу специфики кафедры, на HwProj 2.0.1 в основном размещены курсы по программированию, поэтому большинство решений представляют собой ссылки на ресурсы внешних сервисов, чаще всего на GitHub [3] репозитории. Связав HwProj 2.0.1 и GitHub, можно было бы пользоваться преимуществами обоих сервисов, что облегчило бы жизнь преподавателей и студентов, а также повысило бы их продуктивность. Например, если предоставить возможность публиковать решения напрямую из GitHub, можно было бы использовать Github Actions для публикации только тех решений, которые прошли необходимые тесты, это снизило бы нагрузку на проверяющих.

Ряд существующих систем уже предоставляют возможность работы со сторонними сервисами. Но в то время как одни из них закрывают доступ пользователям из РФ, другие ограничивают возможности взаимодействия между сервисами. Хотелось бы иметь платформу, которая была бы доступна в нашей стране и подстраивалась под нужды кафедры системного программирования СПбГУ.

1. Постановка задачи

Целью работы является предоставление возможности автоматически публиковать решения в сервисе HwProj 2.0.1 [5] напрямую из GitHub [3]. Для её выполнения были поставлены следующие задачи:

1. Выбрать подход для реализации отправки решений задачи от лица студентов из GitHub.
2. Разделить основную задачу на подзадачи.
3. Распределить подзадачи между разработчиками сервиса.
4. Реализовать подзадачи на серверной стороне HwProj 2.0.1.
5. Провести апробацию решения с консультантом.

2. Обзор

2.1. Используемые технологии

Все представленные ниже технологии уже использовались в проекте на момент начала работы. Не было необходимости в поиске новых технологий для реализации задачи, потому что имеющиеся средства предлагали достаточную функциональность. Переход на другие решение привёл бы к большим временным затратам и к необходимости масштабного рефакторинга кодовой базы.

- **ASP.NET Core**
  ASP.NET Core [1] — это кроссплатформенный фреймворк для создания веб-приложений на платформе .NET.
  Фреймворк предоставляет возможность настраивать поток обработки запросов путём внедрения компонентов middleware, одним из них является компонент ответственный за аутентификацию.
  В рамках задачи фреймворк использовался в основном для изменения аутентификации путём добавления нового токена и роли. В проекте используется аутентификация на основе JWT токенов.

- **Entity Framework Core**
  Entity Framework Core [2] — это объектно-ориентированная технология доступа к данным, является решением для объектно-реляционного отображения (objectrelational mapping) для .NET Core от компании Microsoft.
  Фреймворк позволяет работать с объектами базы данных как с экземплярами классов, что упрощает процесс разработки. Entity Framework Core предполагает создание общего контекста с базой данных, таким образом можно получать данные, добавлять таблицы и генерировать миграции без необходимости погружаться в детали работы с базой данных.
  В рамках задачи фреймворк использовался для добавления в структуру базы данных таблицы с токенами курсов, получения

необходимых данных, а также для генерации миграций.
В проекте взаимодействие с базой данных реализовано полностью посредством Entity Framework Core.

- **Swagger**
  Swagger [10] — это набор инструментов, который позволяет генерировать и документировать API.
  Использовался для локального тестирования нового API.

2.2. Существующие решения

- **Stepik**
  Stepik [9] — это образовательная платформа и конструктор онлайн курсов.
  Имеется публичный API, но не поддерживается сдача решения от лица студента. Поддерживается протокол LTI (Learning Tools Interoperability) 1.3 [6], определяющий методы интеграции обучающих систем с внешними сервисами, что позволяет настраивать интеграцию с Github Classrooms.

- **Google Classrooms**
  Google Classrooms [4] - это сервис для образовательных учреждений, разработанный Google в целях создания, распространения и оценки учебных материалов.
  Протокол LTI не поддерживается, но API сервиса позволяет сдавать решения от лица студента.

- **Moodle**
  Moodle [7] - это популярная система управления образовательными электронными курсами с открытым исходным кодом.
  Поддерживается протокол LTI 1.3. Публичный API не позволяет сдавать решения от лица студента.

2.3. Архитектура серверной части

Серверная часть состоит из четырёх сервисов:

- **AuthService** отвечает за авторизацию и аутентификацию пользователей.
- **CoursesService** отвечает за создание и редактирование курсов, домашних работ и задач.
- **SolutionsService** отвечает за сдачу и обработку решений задач.
- **NotificationsService** отвечает за обработку уведомлений в системе.

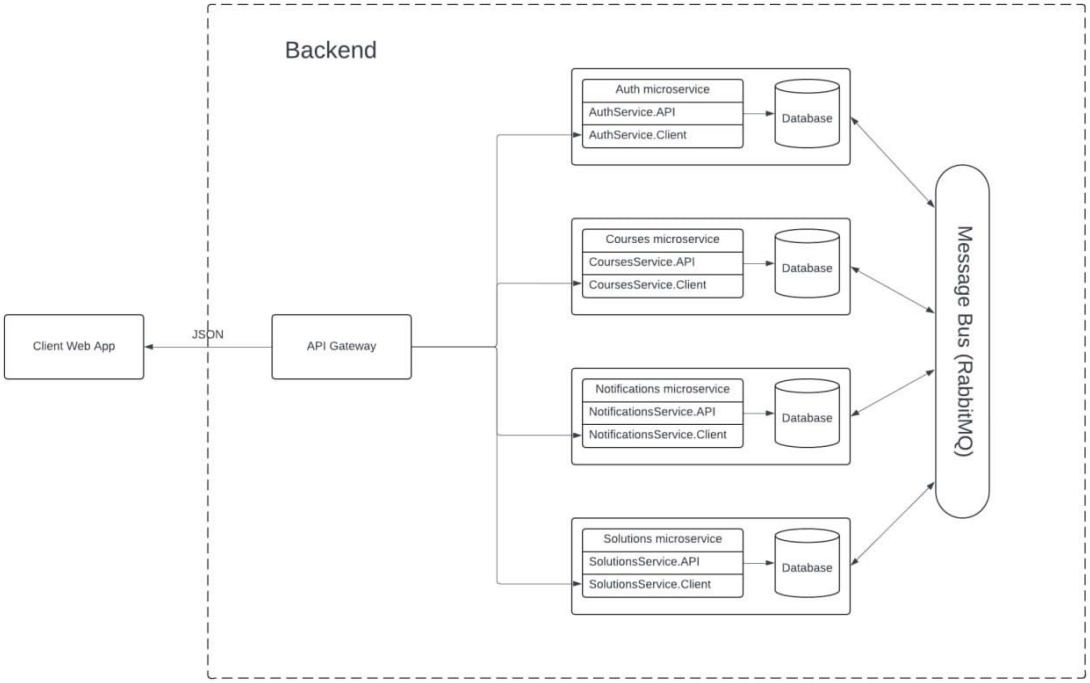

Рис. 1: Архитектура серверной части

Общение клиентской и серверной части реализовано посредством REST API, запросы обрабатывает API Gateway. Общение между микросервисами организовано с помощью брокера сообщений RabbitMQ [8].

3. Реализация

3.1. Выбор подхода к реализации отправки решений от лица студентов

На данном этапе проводились встречи с преподавателями для получения информации о предпочтительном способе интеграции сервиса с GitHub [3].

Изначально рассматривался вариант реализовать поддержку протокола LTI [6], что позволило бы интегрироваться с множеством сервисов, которые работают с этим протоколом, в том числе и с GitHub Classrooms. Но в ходе обсуждений стало ясно, что данное решение выходит далеко за рамки поставленной задачи. Реализация протокола потребовала бы больше времени, а также усложнила бы кодовую базу, поэтому было решено не идти по этому пути.
Очевидным для нас решением стало расширение API сервиса под конкретную задачу — отправка решений из внешних сервисов от лица студентов.

3.2. Разделение на подзадачи

Основная задача была разделена на следующие подзадачи:

- Добавить возможность создания токенов для проверки наличия прав при автоматической отправке решений.
- Добавить возможность хранения токенов.
- Расширение API сервиса для поддержки автоматической отправки решений.
- Добавить возможность идентификации студента по его GitHub логину.
- Добавить шаблон команды для отправки решений в HwProj 2.0.1 [5] из GitHub Actions.

- Поддержать новую функциональность на клиентской стороне.

3.3. Распределение подзадач между разработчиками

Работа над основной задачей велась в команде из двух человек. Мною были решены следующие задачи:

- Добавить возможность создания токенов для проверки наличия прав при автоматической отправке решений.
- Добавить возможность хранения токенов.
- Расширение API сервиса для поддержки автоматической отправки решений.

3.4. Реализация подзадач

3.4.1. Создание и хранение JWT токенов

Использование токена позволяет проверять, есть ли у отправителя необходимые права. Получить токен может только лектор для предотвращения несанкционированной отправки решений. Токен подписывается секретным ключом, а также сгорает по истечении определенного времени.

- В компонент middleware, ответственный за аутентификацию, был добавлен новый тип JWT токена.
- В сервис была добавлена новая роль — AutomaticRole
- API сервиса было расширено методом получения токена для курса.
- Был добавлен сервис и репозиторий для работы с токенами курса.
- В контекст базы данных сервиса курсов была добавлена новая сущность — CourseToken, хранящая информацию о текущем токене курса.

- Была добавлена миграция базы данных для добавления таблицы с токенами.

Листинг 1: Пример создания токена для курса

```
var token = new JwtSecurityToken(
    issuer: _configuration["ApiName"],
    notBefore: DateTime.Now,
    expires: expires,
    claims: new[]
    {
    new Claim("_course_Id", courseId.ToString()),
    new Claim(ClaimTypes.Role, Roles.AutomaticRole)
    },
    signingCredentials: new SigningCredentials(securityKey,
       SecurityAlgorithms.HmacSha256));
```

Получить токен для курса преподаватель может перейдя в компонент редактирования курса.

3.5. Расширение API HwProj 2.0.1

Для того чтобы у внешнего сервиса появилась возможность отправить решение от лица студента необходимо расширить API методом, к которому внешний сервис мог бы обращаться.

Для решения подзадачи был добавлен метод AutomaticPostSolution в ApiGateway, проверяющий права отправителя, а также сохраняющий решение в системе. Далее идет переход в SolutionService и используется уже существующая логика этого сервиса.

В модель PostSolutionModel было добавлено свойство IsAutomatic, это необходимо для фильтрации решений. Например, после публикации решения через GitHub нужна возможность указать группу, по умолчанию группу можно указать только в момент публикации.

Листинг 2: Url путь к методу

```
/api/solutions/automatic
```

Листинг 3: Сигнатура и атрибуты метода

```
[HttpPost("automatic")]
[Authorize(AuthenticationSchemes = "Automatic", Roles = Roles.
    AutomaticRole)]
[ProducesResponseType(typeof(long), (int)HttpStatusCode.OK)]
public async Task<IActionResult> AutomaticPostSolution([FromBody]
    AutomaticSolution model)
```

4. Апробация

Апробация была проведена на консультанте с целью демонстрации возможности автоматической публикации решений в HwProj 2.0.1 [5] через GitHub [5]. Пулл-реквест с реализацией будет влит в основную ветку после ещё одного этапа ревью.

Заключение

Выполненные в ходе учебной практики задачи

- Выбран подход для реализации отправки решений задачи от лица студентов из GitHub [3].
- Выполнено разделение основной задачи на подзадачи между разработчиками сервиса.
- Подзадачи были распределены между разработчиками сервиса.
- На стороне сервера были выполнены подзадачи, которые расширили API HwProj 2.0.1 [5] для отправки решений из внешних сервисов.
- Проведена апробация решения с консультантом.

Ссылка на pull request с реализацией: [11]

Список литературы

[1] Asp.Net Core. — URL: https://github.com/dotnet/aspnetcore (дата обращения: 2024-9-16).

[2] Entity Framework Core. — URL: https://github.com/dotnet/efcore (дата обращения: 2024-9-16).

[3] GitHub. — URL: https://github.com/ (дата обращения: 2024-9-16).

[4] Google Classrooms. — URL: https://edu.google.com/workspace-for-education/classroom/ (дата обращения: 2024-9-16).

[5] HwProj 2.0.1. — URL: https://github.com/InteIligeNET/HwProj-2.0.1 (дата обращения: 2024-9-16).

[6] LTI. — URL: https://en.wikipedia.org/wiki/Learning_Tools_Interoperability (дата обращения: 2024-9-17).

[7] Moodle. — URL: https://github.com/moodle/moodle (дата обращения: 2024-9-16).

[8] RabbitMQ. — URL: https://www.rabbitmq.com (дата обращения: 2024-9-16).

[9] Stepik. — URL: https://stepik.org/ (дата обращения: 2024-9-16).

[10] Swagger. — URL: https://swagger.io/ (дата обращения: 2024-9-16).

[11] Реализация. — URL: https://github.com/InteIligeNET/HwProj-2.0.1/pull/368 (дата обращения: 2024-9-164).