Санкт-Петербургский государственный университет

Программная инженерия

Усик Владислав Сергеевич

Реализация элементов, расширяющих функциональные возможности HwProj 2.0

Отчёт по производственной практике

Научный руководитель:
доц. каф. СП, к.т.н. Ю.В. Литвинов

Санкт-Петербург
2021

Оглавление

Введение

В университетах студентам зачастую необходимо много взаимодействовать с преподавателями. Иногда оборот информации бывает очень большим, будь то задания, документы или какие-либо материалы. Нередко данные процессы стараются упростить и частично автоматизировать, используя различные средства.

HwProj — платформа, активно использующаяся на нашем факультете для упрощения и удобства взаимодействия между студентами и преподавателями в процессе обучения. К сожалению, у данной платформы существует несколько проблем. Во-первых, присутствуют некоторые ошибки и недочеты. Во-вторых, HwProj разрабатывался на устаревших технологиях, что делает его поддержку почти невозможной с имеющимися ресурсами. С целью это исправить, разрабатывается проект HwProj 2.0, использующий современный инструментарий, который позволяет без труда расширять имеющиеся возможности платформы и производить интеграцию по мере надобности. Проект разрабатывается на основе ASP.NET Core [5] и микросервисной архитектуры [1] и на данный момент продолжает развиваться.

Один из самых важных и сложных бизнес-процессов, происходящих на факультете — это работа над практиками. Он содержит большой документооборот и много взаимодействия между людьми. Данный процесс можно автоматизировать, что может упростить саму работу и сэкономить время. Для этого было решено разработать отдельную функциональность в виде сервиса, которая в будущем может стать отдельным приложением, интегрированным с HwProj.

Для правильного взаимодействия добавляемого сервиса с остальными необходима технология общения между сервисами. Для этого будем использовать событийно-ориентированную модель взаимодействия микросервисов. Также новый сервис должен уметь авторизовывать пользователей, для этого в HwProj 2.0 реализован отдельный сервис авторизации, который можно использовать с помощи специального шлюза API.

Данная работа посвящена созданию и внедрению нового сервиса производственных практик, помогающего автоматизировать процессы, связанные с курсовыми работами, в HwProj 2.0, а также реализации необходимых для внедрения и работоспособности сервиса технологий.

1. Постановка задачи

Цель данной работы — добавить новую функциональность в виде сервиса, предоставляющую упрощение проведения производственных практик. Для достижения итоговой цели потребуется выполнить следующие задачи:

- изучить предметную область и бизнес-процесс взаимодействия с производственными практиками;
- спроектировать сервис производственных практик;
- реализовать и подключить сервис к HwProj 2.0;
- изучить и выбрать технологию, реализующую событийно-ориентированную модель, а также применить ее;
- наладить авторизацию и осуществить синхронизацию данных пользователя между сервисами;
- протестировать основные сценарии использования.

2. Обзор предметной области и требований

2.1. Терминология

- Практика — работа, выполняемая студентом в процессе обучения. Работа включает в себя исследование, написание кода и другие элементы. Результатом являются доклад, презентация и выступление по проделанной работе.
- Куратор — роль, ответственная за сопровождение практик. В ее обязанности входит принятие работ, поиск и назначение рецензентов, организация публикаций тем и другое.
- Рецензент — сторонний для студента человек, проверяющий и оценивающий его работу.
- Биддинг — процесс выражения степени заинтересованности в практиках рецензентами в виде ставок.
- Ставка — мера, отображающая то, какой интерес имеет рецензент к практике.
- Дедлайн — крайняя дата, для выполнения какого-либо действия.

2.2. Предметная область

Бизнес-процесс проведения практик происходит в несколько этапов:

- на первом этапе преподаватели публикуют темы и направления своих практик, а кураторы публикуют практики не зарегистрировавшихся пользователей;
- кураторы назначают дедлайн на выбор практики для студентов, за направления которых они отвечают;
- в этот период студенты подают заявки, а преподаватели принимают студентов по заявкам на практики;

- после назначения студента на практику, исходя из курса и направления студента и кафедры преподавателя на практику также назначается куратор;
- после распределения студентов по практикам кураторы назначают дедлайн на сдачу текста. До дедлайна студенты должны загрузить текст;
- примерно после дедлайна по текстам каждый куратор выбирает среди всех пользователей с ролью рецензент тех, кто будет допущен к биддингу и рецензированию практик куратора;
- кураторы начинают период биддинга и назначают дедлайн для рецензентов для предложения ставок;
- рецензенты могут для каждой доступной практики сделать ставку на желание рецензировать работу. Ставки — “Да”, “Может быть”, “Нет”;
- после окончания биддинга кураторы запускают автоматическое распределение рецензентов по практикам. После распределения они корректируют вручную при надобности. Назначается дедлайн для рецензентов для загрузки рецензии.
- после загрузки рецензии, в зависимости от оценки, куратор может отдать текст студенту на переписывание и индивидуально переназначить дедлайн для студента, после чего снова идет рецензирование. Этот этап может повторяться несколько раз;
- если текст написан и положительно отрецензирован, а также загружены все дополнительные файлы и пройдена защита, куратор может объявить работу готовой для защиты.

2.3. Бизнес-требования

Создаваемый сервис будет предназначен для использования на математико-механическом факультете СПбГУ и должен поддерживать до 500 одно-

временных запросов от пользователей, с расчетом на дальнейший рост. В процессе проведения практик участвуют люди с разными ролями (студент, преподаватель, рецензент, куратор), поэтому сервис должен поддерживать ролевую систему. Далее перечислены основные бизнес-требования, которые должны быть реализованы запросами:

- преподаватели должны уметь: создавать страницы практик; принимать/отклонять заявки студентов на практики, загружать отзыв (файл); видеть всю информацию по своим практикам; назначать роль преподавателя другому пользователю. После принятия заявки студента, сервис автоматически должен определить куратора и назначить его на практику;

- студенты должны уметь: просматривать свободные практики; подавать заявки на практики; загружать необходимые файлы, видеть в своей практике только информацию, предназначенную для студента;

- любой пользователь должен уметь выразить желание быть рецензентом. Все такие пользователи в период биддинга должны видеть списки практик, кураторы которых допустили пользователя до биддинга. После назначения рецензента на практику, рецензент должен уметь загружать рецензию и видеть дедлайн по рецензированию;

- куратор должен уметь все то, что умеет преподаватель, как от своего лица, так и от лица другого преподавателя. Также он должен уметь: заполнять данные об университете (кураторы, кафедры, направления и т.д.); назначать дедлайны; объявлять период биддинга; выбирать круг лиц, имеющих доступ к практикам куратора во время биддинга; после биддинга запускать автоматическое распределение рецензентов на практики; после автоматического распределения вручную переназначать рецензентов; ставить зачет за практику; назначать роль куратора пользователю, имеющему роль преподавателя.

2.4. Используемые технологии

- ASP.NET Core — современная кроссплатформенная среда для создания web-приложений и служб. С помощью этой технологии ранее реализованы существующие микросервисы в HwProj 2.0.

- Entity Framework Core [2] — технология ORM для работы с базами данных. Часто используется в связке с ASP.NET Core и также использовалась в реализованных микросервисах HwProj 2.0.

- Ocelot [6] — технология для создания шлюза API, работает только на .NET Core. С помощью нее реализован шлюз в HwProj 2.0.

- RabbitMQ [8] — очередь сообщений, преимуществами которой являются гибкая маршрутизация сообщений и надежность, также она проста в освоении. Она дает возможность создавать узлы со своей собственной очередью, принимать сообщения в очередь по определенным правилам и обрабатывать их. Среди похожих технологий можно выделить Apache Kafka, служебную шину Azure и NServiceBus. Apache Kafka не подходит по бизнес-целям, ее плюсы это возможность работы с большим количеством данных и гарантированный порядок доставки сообщений, но Kafka не гарантирует доставку сообщения, а это важно для текущей работы. Шина Azure и NServiceBus являются уже более высокоуровневыми очередями, предоставляющими большое количество возможностей. Они более тяжеловесны, а многие их функции не требуются для приложения. Поэтому выбор сделан в пользу RabbitMQ.

- Postman [7] — приложение для составления, отправки и тестирования запросов. Использовался для тестирования запросов сервиса практик.

3. Обзор существующих решений

3.1. HwProj

HwProj позволяет работать студентам с преподавателями в рамках какого-то курса. Преподаватели могут создавать курсы, выкладывать в них материалы и ставить задачи. Студенты загружают решения и после преподаватель должен проверить его и либо зачесть задачу, либо отправить на переписывание. На текущий момент нет никакой функциональности для сопровождения практик и поддержкой проекта никто не занимается. Приложение написано на языке Ruby старой версии, но на кафедре просто нет студентов, знающих старый Ruby или готовых развивать проект.

3.2. Stepik

Stepik — платформа для составления и прохождения курсов. Преподаватели могут через специальный конструктор создавать курсы, которые содержат в себе текстовые материалы, видеозаписи, задачи разных типов (тест, код, ответ). По прохождению курсов студенты получают сертификат на определенную оценку исходя из набранных баллов за выполнение задач. Stepik можно активно использовать для преподавания определенных дисциплин в университете. К сожалению, платформа не обладает необходимой функциональностью для проведения практик, так как изначальная цель платформы — именно свободное обучение пользователей.

3.3. BlackBoard

BlackBoard — платформа для обучения, общения и обмена информацией между студентами и преподавателями. Она имеет достаточно мощный инструментарий, поэтому с помощью нее можно автоматизировать многие учебные бизнес-процессы. Система позволяет вести курсы, создать тесты, делать объявления, управлять доступами, выкладывать

различные ресурсы и многие другие учебные процессы. BlackBoard имеет систему уведомлений и различные интеграции, например, со студенческой почтой. Плюсом можно выделить, что имеются некоторые возможности по сопровождению практик. Можно вести список практик и загружать необходимые файлы.

Вместе с большими возможностями BlackBoard имеет немного перегруженный интерфейс. Опрос, проведенный среди студентов и преподавателей, показал, что пользователи оценили удобность интерфейса в 35,5 баллов из 100, что является очень низкой оценкой и подтверждает неудобство интерфейса. Также, так как это сторонняя система с закрытым кодом, она закрыта для расширения, что делает невозможным введение дополнительных возможностей для автоматизации сопровождения практик, например, систему биддинга для распределения рецензентов.

3.4. EasyChair

EasyChair — система для управления конференциями. Она предоставляет множество гибких возможностей для процесса запуска конференций и организации сбора и проверки документов. Платформа позволяет подавать статьи для конференций, приглашать заинтересованных лиц, загружать необходимые файлы, вести обсуждение, получать отзывы и отвечать на них. Присутствует система рецензирования и система биддинга.

Платформу можно использовать для обучения студентов, в том числе применить к процессу сопровождения практик. Студент может загружать свой доклад и остальные документы, получать рецензирование и также вести обсуждения по работе. Однако система в первую очередь предназначена для помощи запуска различных по тематике конференций, в ней не фигурируют все объекты из области университета и проведения практик, такие как кураторы, кафедры, направления, дедлайны, поэтому для некоторых задач, характерных именно для бизнес-процесса проведения практик, использовать систему не получится или

это будет проблематично.

3.5. Moodle

Moodle — система управления обучением с открытым исходным кодом. Особенность Moodle заключается в очень гибкой настройке системы под определенные учебные процессы, большое количество плагинов с различной функциональностью, а также хорошая интегрируемость с другими продуктами. Система бесплатная и ее необходимо самостоятельно устанавливать на сервер, настраивать и сопровождать. С помощью администрирования и подключения плагинов можно установить необходимую функциональность, такую как создание курсов, ведение отчетности, обсуждения и другое. Можно гибко создавать роли и регулировать их права и доступы. Благодаря открытому исходному коду и возможности написания плагинов систему можно расширять и дополнять необходимыми возможностями. При должной настройке и сопровождении, а также написании плагинов для некоторых процессов Moodle можно применять и для сопровождения практик.

Moodle написан на PHP, но на нашем факультете не подразумевается изучение этого языка, в связи с этим найти необходимые ресурсы для разработки плагинов будет весьма сложно. Также Moodle уже слегка устарел относительно новых технологий. По этим причинам выбор был сделан в пользу разработки HwProj 2.0.

3.6. Текущее сопровождение практик

На текущий момент практики сопровождаются без автоматизации. Используется Google Docs для ведения практик, все остальные же моменты взаимодействия между заинтересованными лицами делаются напрямую. Это самый гибкий по бизнес-целям вариант, но вместе с этим на такое сопровождение практик уходит самое большое количество времени и усилий.

4. Описание реализации

В ходе работы был написан сервис практик, реализована очередь событий, налажены авторизация и общение между сервисом авторизации и сервисом практик. Многие архитектурные решения в HwProj 2.0 и сервиса практик взяты из приложения eShopOnContainers [11] с открытым исходным кодом. В результате работы в сервисе практик присутствует порядка 55 запросов, покрывающих исходные бизнес-требования. С кодом работы можно ознакомиться по ссылке https://github.com/InteIligeNET/HwProj-2.0.1/pull/49.

4.1. Архитектура приложения

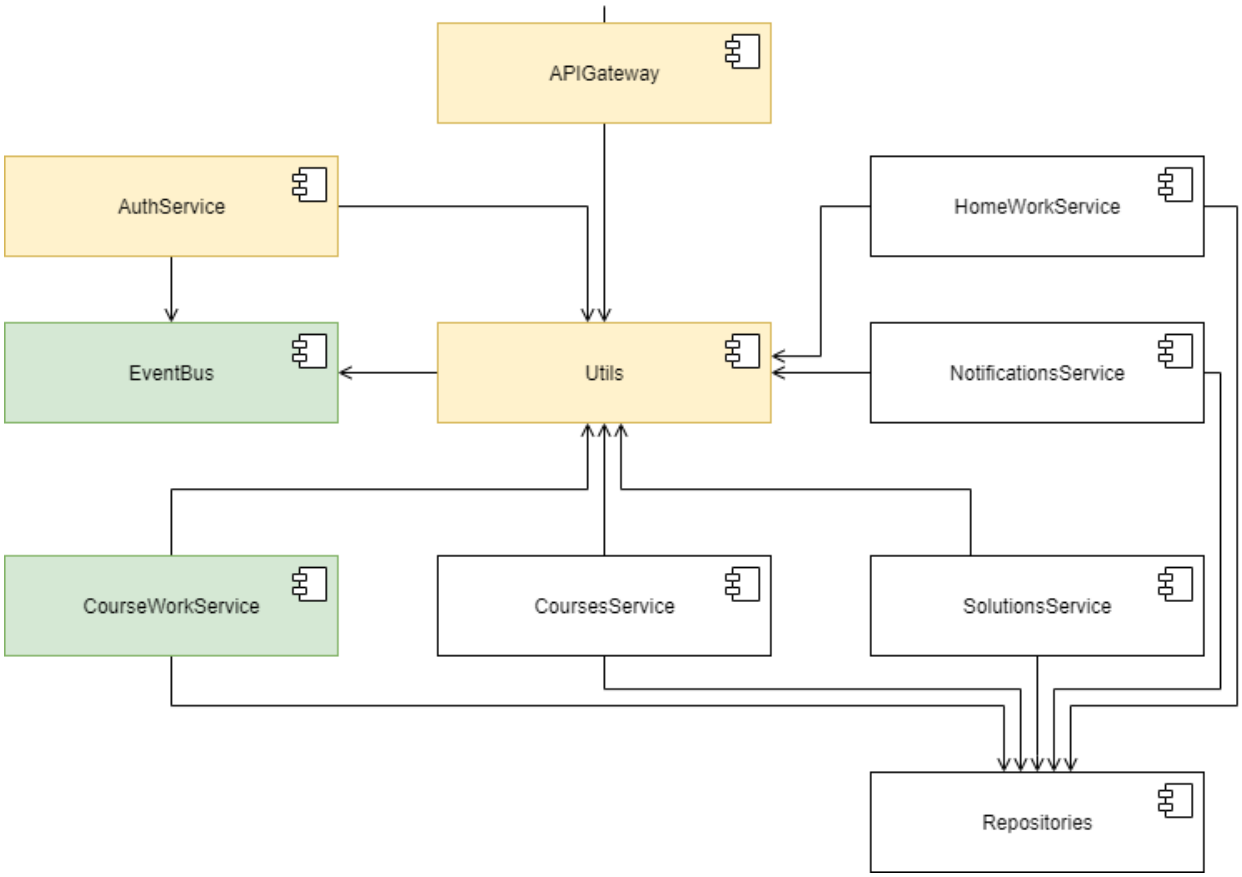

Рис. 1: Диаграмма компонентов

С диаграммой компонентов можно ознакомиться на рисунке 1. В ходе работы зеленые компоненты были созданы с нуля, желтые были изменены или дополнены, белые не изменились. Компонент Utils

содержит в себе общие методы для использования и настройки конфигурации микросервисов, от него зависят все микросервисы в HwProj 2.0. Компоненты AuthService и CourseWorkService представляют собой микросервисы авторизации и практик соответственно. Они содержат соответствующие запросы и всю логику по их обработке. APIGateway — микросервис шлюза API, он принимает запросы, проверяет авторизацию и роль и при успехе перенаправляет запрос на другие микросервисы. Repositories — библиотека классов, в ней находятся базовые интерфейсы репозиториев и их реализации. EventBus — библиотека классов для работы с очередью событий, инкапсулирующая в себе более низкоуровневое использование очереди RabbitMQ. Компоненты HomeWorkService, NotificationsService, CoursesService и SolutionService являются микросервисами HwProj 2.0 разработанные ранее до текущей работы. Они содержат запросы для работы с курсами, домашними задачами, решениями и уведомлениями.

4.2. Архитектура сервиса

Сам сервис практик представляет собой классический микросервис в ASP.Net Core. Есть несколько контроллеров — общий и отдельные, запросы в которых предназначены для определенной роли. В контроллерах происходят проверки на доступность операции, валидация данных и возврат статусных кодов. Для выполнения бизнес-логики контроллеры обращаются к классам сервисов. Они в свою очередь обращаются к классам репозиториев, для работы с данными. Классы репозиториев [4] предоставляют методы работы с данными, как простыми запросами, так и сложными, и так же поддерживают целостность данных. Все классы сервисов и репозиториев используются через интерфейсы и технологию внедрения зависимостей (DI).

Модель данных представлена на рисунке 2. Объект User имеет ссылки на объекты StudentProfile, LecturerProfile, ReviewerProfile и CuratorProfile, которые содержат информацию о пользователе относительно соответствующей роли, при ее наличии. Объекты Department

и Direction — кафедры и направления соответственно, нужны для автоматического назначения кураторов на практики и для назначения дедлайнов на направления (выбор тем). Объект Bid представляет собой ставку и связана с объектами практики и рецензента. CuratorProfile и ReviewerProfile имеют связь многие ко многим, в базе данных связь разрешается дополнительной таблицей ReviewersInCuratorsBidding, связь означает, что рецензент допущен к биддингу куратора. Аналогично связь многие ко многим имеют объекты User и Role, что характеризует принадлежность роли пользователю.

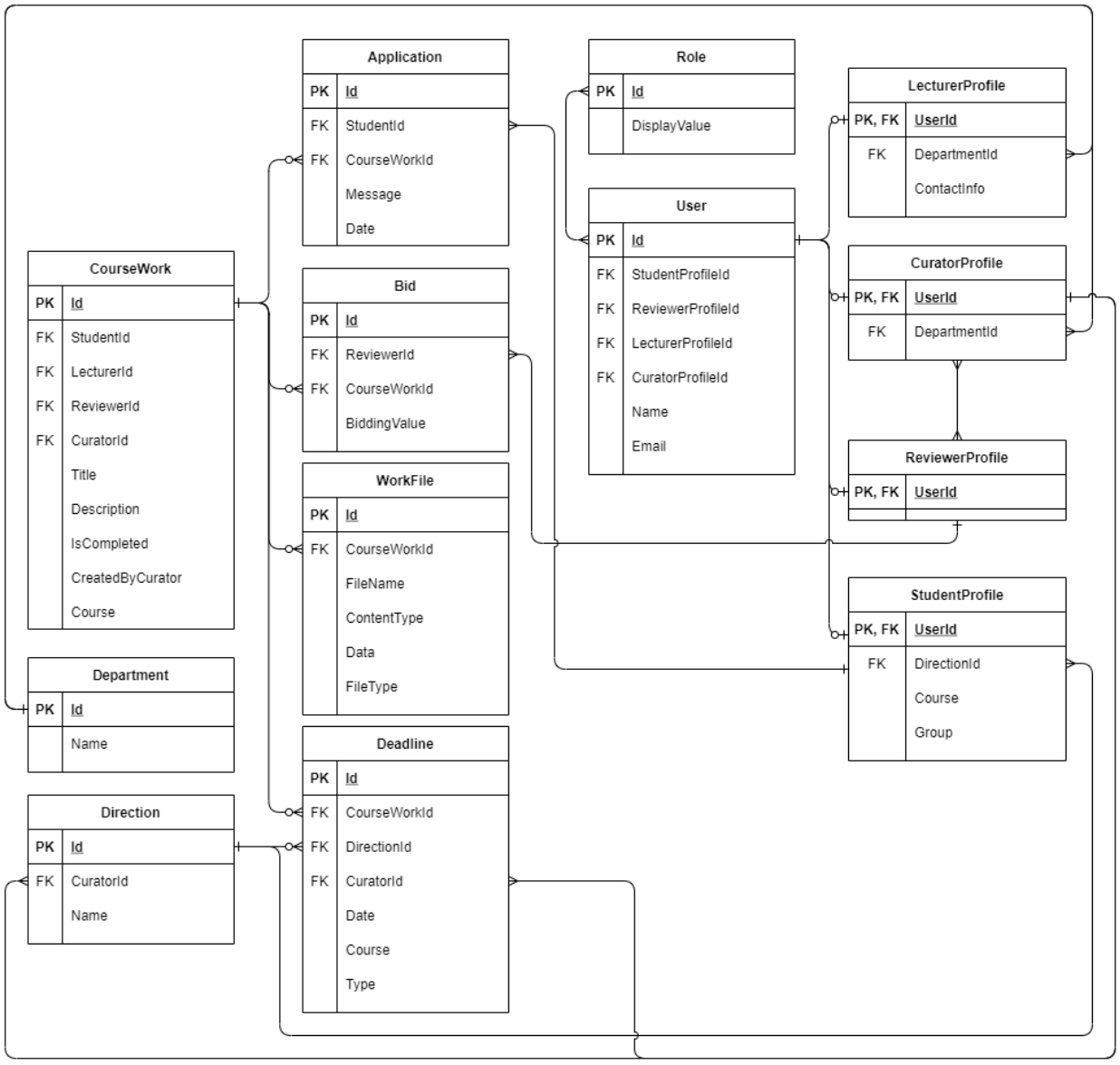

Рис. 2: Модель данных

Диаграмма классов находится на рисунке 3, на ней изображены основные классы сервисов и репозиториев. Из основных можно выделить:

- UserRepository — позволяет делать операции с пользователями, данными пользователей и ролями, при этом сохраняя целостность и правильные взаимосвязи;
- CourseWorkService — предоставляет все основные операции над практиками и файлами;
- ApplicationService — содержит методы для работы с заявками, также здесь происходит автоматическое определение куратора после принятия заявки;
- UniversityService — включает в себя операции над кафедрами, направлениями, дедлайнами. Содержит логику отображения нужных дедлайнов. Позволяет назначать дедлайны на направление(дедлайн выбора тем), на все свои курсовые и также отделено переопределять дедлайны для конкретных практик;
- ReviewService — здесь сосредоточены операции над рецензентами, допущенных к биддингу. Также здесь реализован алгоритм распределения рецензентов.

4.3. Очередь событий

Был создан единый класс очереди событий, инкапсулирующий всю работу с очередями и событиями при помощи RabbitMQ. Этот класс подключен к каждому микросервису, который нуждается во взаимодействии с другими. Внешне этот класс предоставляет простой интерфейс, дающий возможность подписаться на определенный тип событий и отправить событие. Выглядит это все как единая очередь, которая сама знает куда и кому отправлять сообщение.

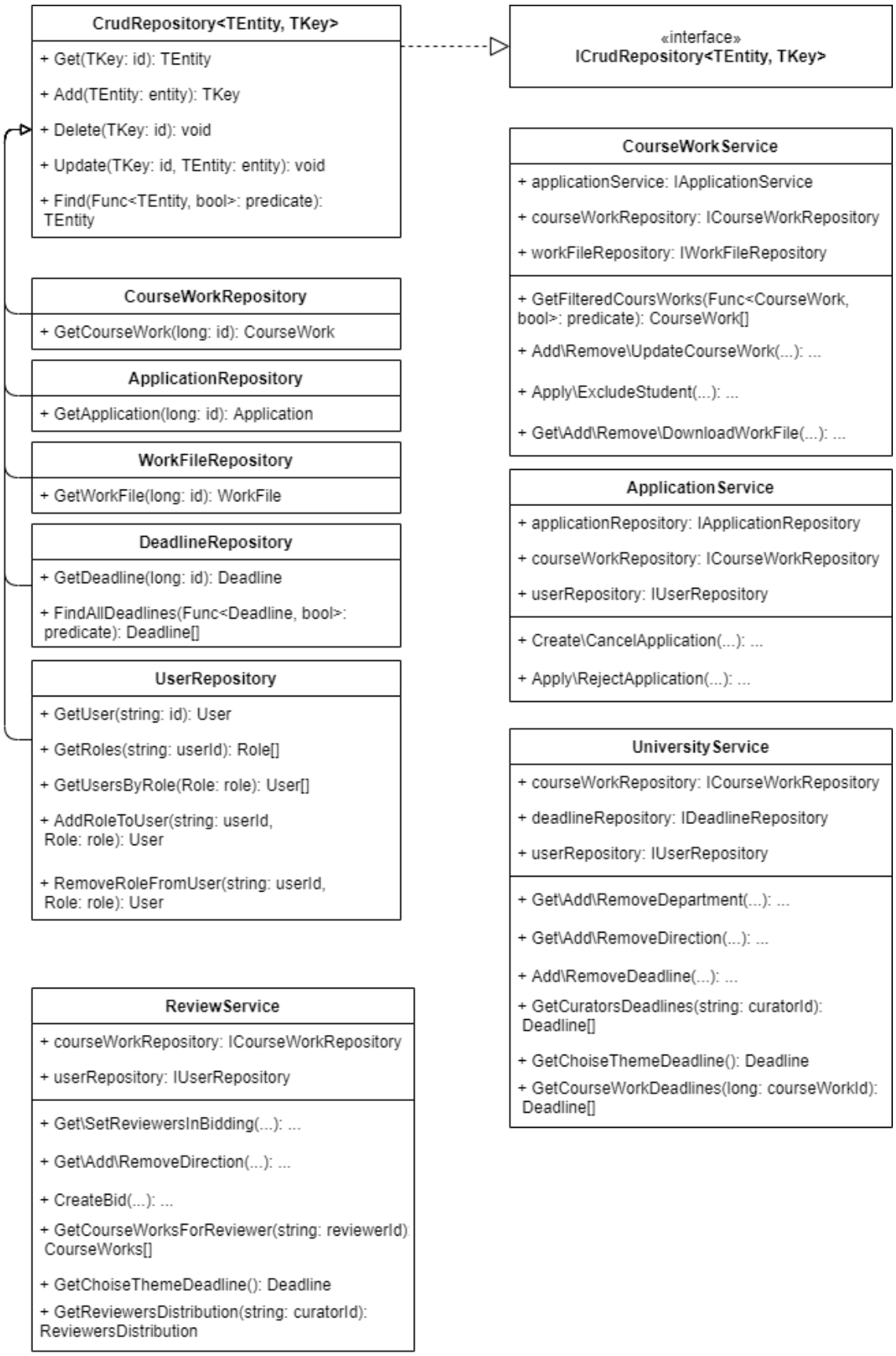

Рис. 3: Частичная диаграмма классов

Внутри класс устроен следующим образом: каждый микро-сервис имеет свой узел и собственную очередь; при запуске сервиса он подписывается на все типы событий, которые хочет получать, а также указывает как их обрабатывать; при публикации сообщения RabbitMQ отправляет сообщения только подписанным на этот тип события очередям; отправка является отказоустойчивой, если сообщение не дошло, происходит повторная отправка сообщения с увеличением интервала перед следующей попыткой.

В обработчиках событий могут выполняться разные действия, в том числе обращение к базе данных, поэтому важно продуманно использовать контекст базы данных и следить за его временем жизни [3]. В микросервисах scope и контекст создаются автоматически отдельно для каждого запроса, но обработка событий в очереди происходит вне запросов. Поэтому для каждой обработки события вручную создается свой scope и контекст базы данных.

4.4. Авторизация и данные пользователей

В процессе сопровождения практик участвуют многие люди, в том числе пользователи могут быть никак не связаны с университетом и не иметь никаких университетских акканутов, поэтому авторизацию было решено делать собственную, с дальнейшей возможностью добавлять интеграции с разными платформами. В HwProj 2.0 ранее уже был создан микросервис для авторизации пользователей, использующий JWT Bearer токены [9], поэтому надобности создавать еще одну авторизацию нет. Но с другой стороны, сервис практик довольно сильно отличается по бизнес-целям от текущего HwProj 2.0 и работает с другой информацией о пользователях и другими ролями. Так же сервис практик может считаться самостоятельным, отдельным от HwProj 2.0 приложением.

По этим причинам было решено сделать собственную таблицу данных пользователей, оставив авторизацию через старый микросервис. Для этого во время всех основных операций по добавлению/изменению/удалению пользователей в сервисе авторизации в оче-

редь отправляются определенные события, сервис практик получает эти события и регистрирует такие же изменения в своей базе. После в сервисе практик заполняется дополнительная информация о новых пользователях. Таким образом у обоих сервисов в своих базах данных хранится один и тот же список с основной информацией о пользователях, но разной из-за бизнес-целей дополнительной информацией.

4.5. Алгоритм распределения рецензентов

Задача алгоритма

В период биддинга, все рецензенты делают на практики ставки — “да”, “может быть”, “нет”, которые означают соответственно +1, +0.5, -1 балл в желании рецензировать данную практику. По окончанию биддинга для всех ситуаций, где рецензент не сделал никакой ставки на практику, считается, что он сделал нейтральную ставку с баллом 0.

Требуется по данным с биддинга для всех практик куратора распределить рецензентов на практики так, чтобы это было достаточно оптимальным образом. Оптимальность считается следующим образом: для каждой практики, на которую назначен рецензент, берется ставка, которая была у этого рецензента на эту практику. Все эти ставки суммируются по баллам. Чем выше получившаяся сумма, тем более оптимально распределение. То есть алгоритм должен находить распределение с такой максимальной суммой. При этом один рецензент может отвечать за несколько практик, чем больше практик у одного рецензента, тем хуже, это тоже стоит учесть.

Алгоритм:

- Алгоритм строится на основе задачи о назначениях и венгерского метода [10]. В качестве матрицы будет выступать матрица ставок, где строки – практики, столбцы – рецензенты, а ячейки – ставка рецензента на практику.

- Так как на все практики рецензент должен назначаться обязательно и независимо от кол-ва рецензентов, то чтобы применить

венгерский метод количество рецензентов в матрице должно быть больше или равно количеству практик. Но в реальном процессе такое бывает далеко не всегда, поэтому делается следующее: если рецензентов оказалось меньше чем практик, то добавляем в матрицу дополнительные столбцы рецензентов пока столбцов не будет больше или равно строкам; столбцы добавляются следующим образом — в первоначальном виде матрицы было N столбцов рецензентов, при каждом добавлении копируем все N начальных столбцов и добавляем их в матрицу. Таким образом одному рецензенту будет соответствовать несколько столбцов.

- После выполнения венгерского метода в матрице будет выбрано M ячеек, никакие две из которых не стоят на одной строке или столбце. Эти ячейки и характеризуют назначение рецензента на курсовую.

- Алгоритм выше описан только в рамках распределения допущенных рецензентов одного куратора до практик этого куратора. Но кураторов и таких распределений несколько. Практики разных кураторов не могут пересекаться, но рецензенты могут быть допущены к практикам и первого и второго куратора, а значит при текущем алгоритме, рецензенты, допущенные к распределению обоих кураторов, будут по итогу назначаться на большее количество практик. Чтобы учитывать это, сделаем модификацию в алгоритме. При построении матрицы в каждой ячейке будем писать не чистый балл ставки, а $k - 0.5 * n$, где k – балл ставки, а n – количество практик, на которые рецензент уже назначен. Таким образом, оптимальный результат будет учитывать количество практик, за которые отвечают рецензенты.

4.6. Тестирование

Для тестирования основных сценариев работы использовалось приложение Postman. Оно предоставляет удобный интерфейс для составле-

ния и отправки запросов. Запросы можно настроить необходимым образом, указав заголовки, тело, токены и остальное при необходимости. Также Postman позволяет писать тесты, включающие последовательности запросов. Тестирование происходило преимущественно вручную, а также через несколько автоматических тестов в Postman. Запросы делались через шлюз API и с проверкой авторизации и доступов, запущены были сервис авторизации, сервис практик и шлюз API. Среди автоматических тестов Postman были следующие сценарии:

- проверка авторизации студента и последующего доступа к разрешенному запросу и запрещенному (запросы — получить список практик студента и получить список практик преподавателя);
- проверка добавления практики и последующего чтения данных этой практики;
- проверка сценария автоматического назначения куратора. То есть создание практики, подача заявки на практику, принятие заявки и после проверка куратора из данных практики;
- проверка отображения дедлайна на выбор темы для студента;
- проверка переопределения дедлайна для студента.

Заключение

В ходе данной работы было выполнено следующее:

- изучена предметная область и требования к сервису практик;
- спроектирована архитектура сервиса;
- реализован сервис практик с запросами, покрывающими бизнес-требования;
- реализована очередь событий для общения между микросервисами;
- подключена базовая авторизация HwProj 2.0 к сервису практик, а также налажена синхронизация данных о пользователях с помощью очереди событий;
- были протестированы все основные сценарии использования.

С кодом работы можно ознакомиться по ссылке https://github.com/InteIligeNET/HwProj-2.0.1/pull/49.

Список литературы

[1] Cesar de la Torre Bill Wagner Mike Rousos. .NET Microservices: Architecture for Containerized .NET Applications // MSDN. — 2020. — Access mode: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/ (online; accessed: 13.10.2020).

[2] Entity Framework Core Documentation // MSDN. — 2020. — Access mode: https://docs.microsoft.com/en-us/ef/core/ (online; accessed: 15.11.2020).

[3] GUEDDARI MEHDI EL. Managing DbContext with Entity Framework // blog. — 2014. — Access mode: https://mehdi.me/ambient-dbcontext-in-ef6/ (online; accessed: 12.01.2021).

[4] Kudchikar Shadman. Repository Pattern C // blog. — 2019. — Access mode: https://codewithshadman.com/repository-pattern-csharp/ (online; accessed: 17.11.2020).

[5] Microsoft. Introduction to ASP.NET Core // MSDN. — 2020. — Access mode: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1 (online; accessed: 05.10.2020).

[6] Ocelot API Gateway Guide // Ocelot Documentation Archive. — Access mode: https://ocelot.readthedocs.io/en/latest/introduction/gettingstarted.html (online; accessed: 23.11.2020).

[7] Postman Guide // Postman Documentation Archive. — Access mode: https://learning.postman.com/docs/getting-started/introduction/ (online; accessed: 29.11.2020).

[8] RabbitMQ .Net Guide // RabbitMQ Documentation Archive. — Access mode: https://www.rabbitmq.com/dotnet-api-guide.html (online; accessed: 02.11.2020).

[9] Vinicios Marcos. Authorization and Authentication with Bearer and JWT // blog. — 2020. — Access mode: https://medium.com/the-innovation/asp-net-core-3-authorization-and-authentication-with-bearer-and (online; accessed: 07.12.2020).

[10] Wikipedia. Hungarian algorithm // Wikipedia, the free encyclopedia. — 2019. — Access mode: https://en.wikipedia.org/wiki/Hungarian_algorithm (online; accessed: 15.04.2021).

[11] eShopOnContainers app // github. — 2020. — Access mode: https://github.com/dotnet-architecture/eShopOnContainers (online; accessed: 14.10.2020).