Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Гирин Алексей Романович

Веб-приложение для сдачи и проверки заданий по программированию

Курсовая работа

Научный руководитель:

к.т.н., доцент Литвинов Ю. В.

Санкт-Петербург

2019

Оглавление

Введение

Организация учебного процесса подразумевает выполнение студентами домашних и контрольных работ, поэтому существует необходимость в удобном сервисе, предоставляющем возможность сдачи и дистанционной проверки заданий. В настоящее время для этого используется проект HwProj [1], который разрабатывался с применением непопулярных на данный момент средств, что значительно затрудняет его поддержку. Таким образом, целью данной курсовой работы является реализация веб-приложения для сдачи и проверки заданий по программированию с использованием современных технологий и высоким потенциалом развития.

Для создания такого проекта были определены следующие задачи:

- Разработка удобной для внедрения новых возможностей архитектуры проекта
- Организация системы ролей пользователей
- Организация управления аккаунтами пользователей
- Организация управления курсами
- Организация взаимодействия студентов с курсами
- Организация способов проверки заданий и отслеживания прогресса
- Создание пользовательского интерфейса
- Настройка непрерывной интеграции
- Автоматизация развертывания в облачную платформу

1. Обзор существующих решений

- HwProj — веб-приложение, используемое для сдачи и проверки заданий по программированию на математико-механическом факультете СПБГУ. Является наиболее приближенной альтернативой к разрабатываемому проекту. Поддерживает возможность создания курсов, отправки решений и отслеживания прогресса. Имеет базовую функциональность для управления аккаунтом. Формат проекта предполагает, что теоретический материал читается на очных занятиях, а сам сервис используется непосредственно для сдачи практических заданий. В настоящее время проект не поддерживается разработчиками.

- Stepik [2] — образовательная платформа и конструктор онлайн-курсов. Поддерживает возможность создания курсов с применением видеоматериалов, отправки решений и отслеживания прогресса. Решения проверяются тестирующей системой. В основном данный проект используется для создания видео-курсов, ориентированных на широкую аудиторию. Студенты просматривают теоретический материал и выполняют задания различных типов. Курсы может создавать любой человек. У данного сервиса есть несколько проблем: во-первых, курсы состоят из модулей, которые в свою очередь состоят из уроков, и для того, чтобы создать курс, требуется выполнить ряд условий, например, наличие как минимум 10 уроков; во-вторых, ограниченный доступ к курсу является платной услугой; также форма сдачи заданий в курсах на stepik не является удобной для проверки задания преподавателем, а не тестирующей системой; отсутствует возможность уведомления студентов

о результатах проверки заданий, а преподавателей о факте сдачи студентом задания; неудобно следить за своими результатами и результатами остальных студентов.

Исходя из этого, разработка веб-приложения для сдачи и проверки заданий по программированию является актуальной задачей.

2. Описание решения

При разработке приложения использовалась микросервисная архитектура [3]. Это подход, при котором большое приложение разбивается на небольшие независимые компоненты — микросервисы. Каждый микросервис реализует узкий набор связанных функций. Микросервисы могут быть написаны на разных технологиях, пользоваться разными базами данных и запускаться на разных машинах. Это позволяет организовать разработку приложения наиболее эффективным образом и способствует удобному внедрению новых микросервисов при расширении приложения.

В разрабатываемом проекте было выделено 4 микросервиса:

- Микросервис курсов — отвечает за управление курсами
- Микросервис домашних работ — отвечает за управление домашними работами
- Микросервис решений — отвечает за управление решениями
- Микросервис авторизации — отвечает за управление учетными записями пользователей, авторизацию и аутентификацию

Каждый микросервис является REST-сервисом, написанном на веб-фреймворке ASP.NET Core. Выбор этого веб-фреймворка обусловлен его активным развитием, кроссплатформенностью, модульностью и легковесным конвейером обработки HTTP запросов. Каждый микросервис взаимодействует с базой данных посредством технологии Entity Framework (EF) Core. Это современная, кроссплатформенная версия библиотеки Entity Framework, которая является ORM-инструментом. То есть EF Core позволяет программисту

работать с данными в привычном для объектно-ориентированных программ виде.

2.1. Использование шлюза API Gateway

Изначально предполагалось использование архитектуры прямого взаимодействия клиента и сервисов (Рис. 1). При таком подходе у каждого сервиса есть общедоступная конечная точка, и клиент напрямую отправляет запросы к микросервисам. Однако такая организация сопряжена с рядом проблем. Это проблемы безопасности — микросервисы будут доступны извне, что значительно увеличивает уязвимую зону и сказывается на надежности приложения, и проблемы сквозной функциональности — каждый общедоступный сервис будет вынужден самостоятельно обрабатывать общие для всех микросервисов задачи, например, авторизацию.

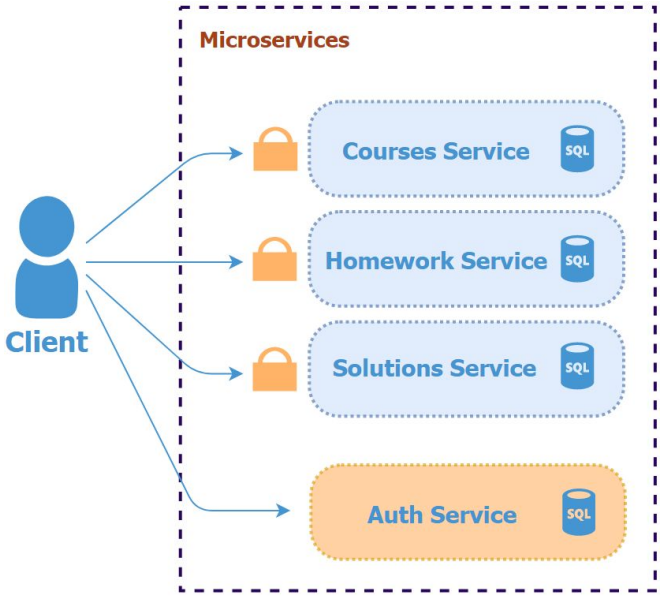

Рис. 1: Изначальная архитектура проекта

Для решениях этих проблем был создан шлюз API Gateway — Web API приложение, реализующее паттерн Backends for Frontends (Рис. 2). Во-первых, этот шлюз предоставляет единую конечную точку для клиента, перенаправляет клиентские запросы к конкретным сервисам, тем самым отделяя клиента от микросервисов. Во-вторых, предоставляет единую для всех сервисов авторизацию.

Авторизация в приложении организована через токены. Когда пользователь выполняет запрос на вход в систему, для него на сервисе авторизации генерируется токен доступа формата JSON Web Token с определенным временем жизни. Данный объект указывает на личность пользователя и подписывается секретным ключом, что предупреждает подмену данных. Токен доступа помещается в локальное хранилище на стороне клиента и добавляется в заголовок каждого запроса, требующего авторизацию. На этапе шлюза токен валидируется, из него изымается информация о пользователе, которая также проверяется. Если пользователь удовлетворяет всем условиям, то запрос перенаправляется к конкретному сервису.

Для реализации шлюза использовался проект Ocelot [4]. Он не требует много ресурсов, работает быстро и хорошо подходит для небольших приложений с микросервисной архитектурой.

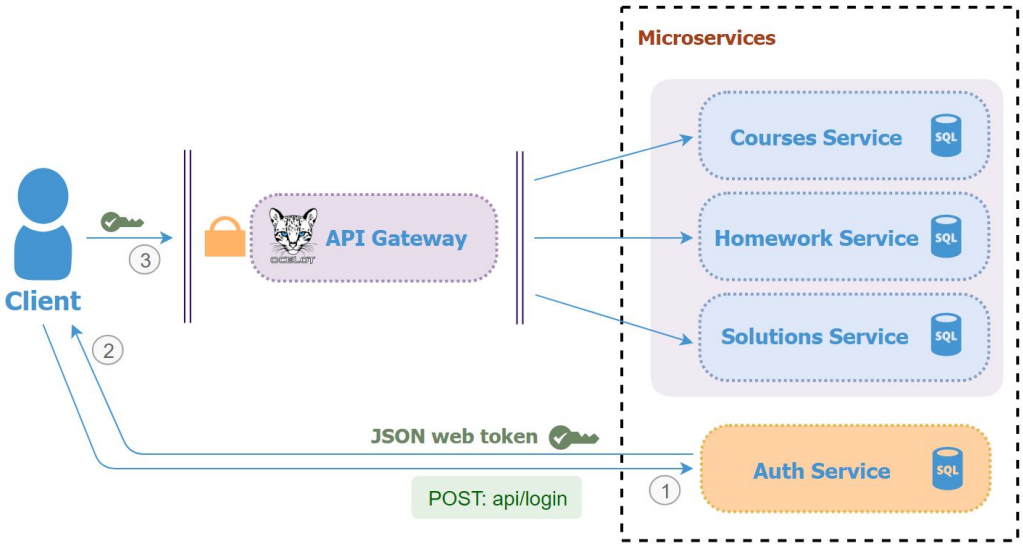

Рис. 2: Архитектура проекта. Итог

2.2. Сервис авторизации

Сервис авторизации отвечает за управление аккаунтами пользователей, авторизацию и аутентификацию. Для этого используется система ASP.NET Core Idenity. Поддерживается базовая функциональность для ведения профиля: регистрация, изменение данных учетной записи, удаление. Через службу оповещений, которая посылает ссылки подтверждения на почту пользователя, проверяется достоверность адреса почты при регистрации. Система ролей в приложении представлена студентами и преподавателями. Сервис авторизации обеспечивает возможность пользователю стать преподавателем после получения приглашения от другого преподавателя. Также данный микросервис производит генерацию токенов доступа с помощью механизма HMAC-SHA256, что обеспечивает надежную передачу данных пользователей остальным микросервисам.

Добавлена возможность авторизации в системе через аккаунт GitHub [5]. Для этого используется протокол OAuth. После выполнения запроса на вход в

систему через стороннего провайдера пользователь перенаправляется для выполнения аутентификации на стороне GitHub. После этого сервис авторизации получает доступ к данным GitHub аккаунта пользователя, выполняет проверку и предоставляет аккаунт приложения. Выбор GitHub обусловлен высокой популярностью этого веб-сервиса. На данный момент пользовательский интерфейс позволяет выполнять вход в систему только через аккаунт приложения. Добавление авторизации через GitHub на клиентской стороне планируется после запуска проекта.

Заключение

В рамках данной курсовой работы было реализовано веб-приложение для сдачи и проверки заданий по программированию. В частности, были решены следующие задачи:

- Организация авторизации и аутентификации приложения
- Организация управления аккаунтами
- Создание шлюза API Gateway

Список литературы

[1] HwProj. — 2016. — URL: http://hwproj.me/ (дата обращения: 31.05.2019).

[2] Stepik. — 2013. — URL: https://stepik.org/ (дата обращения: 31.05.2019).

[3] Cesar de la Torre Bill Wagner Mike Rousos. .NET Microservices: Architecture for Containerized .NET Applications. — Redmond, Washington : One Microsoft Way, 2018.

[4] Ocelot. — 2016. — URL: https://github.com/ThreeMammals/Ocelot/.

[5] GitHub. — 2008. — URL: https://github.com/ (дата обращения: 31.05.2019).