Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 20.Б10-мм

Пузанов Михаил Олегович

Реализация уведомлений, отправки их на почту и Telegram бот, тестирование сервисов уведомлений и решений для HwProj 2

Отчёт по учебной практике

в форме «Производственное задание»

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

Консультант:
инженер-программист JetBrains, А. В. Бережных

Санкт-Петербург
2022

Оглавление

1. Введение

Каждому человеку важно быть в курсе того, что происходит вокруг, и он должен быть осведомлен о каких-то изменениях, связанных с ним. Уведомления — важный аспект нашей жизни. Они также очень значимы в сфере образования. Каждый студент нуждается в осведомлении, особенно в настоящее время, когда образование частично перешло с полного очного формата на смешанный. Для преподавателей важно своевременно получать информацию о сдаче задачи студентов, а для студентов о публикации новой домашней работы. К сожалению, не каждый подобный сервис может предоставить комфортную систему уведомлений для пользователя.

На первом курсе обучения, задачи по программированию сдавались через BlackBoard Learn. У сервиса, конечно же, есть поддержка уведомлений и отправка их на почту, но внутри него они выглядят довольно неудобно, на почту уведомления приходили с большим опозданием, когда это было уже неактуально. В 3 семестре мы сдавали задачи через Microsoft Teams. Microsoft Teams не поддерживает отправку сообщений на почту, из-за этого всегда приходится открывать приложение для проверки новой информации. Для преподавателей неудобство заключается в том, что им не приходит уведомление о том, что студент сдал задачу. Также кафедра системного программирования использует HwProj. HwProj, как и BlackBoard Learn поддерживает уведомления внутри сервиса и отправку их на почту.

К сожалению, не один из сервисов не предоставляет возможность получение уведомлений в каком-либо из мессенджеров. Это очень неудобно, потому что множество студентов для общения используют мессенджеры. Интеграция с мессенджером позволит студентам, как можно быстрее получать информацию о новых домашних заданиях и об их оценивании, а отправить решения.

HwProj 2.0.1 — это студенческий проект для проверки домашних работ по программированию, над которым работает группа студентов. На HwProj 2.0.1 хотят устранить все недочеты представленных аналогов

в системе уведомлений и сделать Telegram бот, который будет реализовывать функциональность для получения информации о домашних работах и их решениях.

Также нельзя забывать, что программное обеспечение тестируют для проверки его работоспособности и характерного поведения в определенных условиях. Тестирование является неотъемлемой частью разработки любого приложения. Сервис уведомлений отвечает за обеспечение пользователя уведомлениями в самом веб-приложении, как было упомянуто ранее, уведомления очень важны в настоящие время. Сервис решения отвечает за отправку домашних заданий студентами и проверку их преподавателями. Данные сервисы выполняют большой объем работы, поэтому тестирование данных сервисов необходимо для проверки работоспособности.

2. Постановка задачи

Целью работы является добавление возможностей получать уведомления на почту и в Telegram бота, получение информации по курсу и отправку решений в Telegram боте, написание интеграционных тестов для сервисов уведомлений и решений. Для её выполнения были поставлены следующие задачи:

1. Провести обзор аналогов.
2. Разобраться в архитектуре web-сервиса.
3. Сформировать и реализовать список недостающих уведомлений.
4. Реализовать отправку уведомлений на почту и в Telegram бота.
5. Реализовать функциональность для получения информации по курсу в Telegram боте.
6. Реализовать функциональность для отправки решений через Telegram бот.
7. Провести тестовые сценарии для проверки работоспособности Telegram бота.
8. Сформировать сценарии интеграционных тестов для сервисов уведомлений и решений.
9. Реализовать интеграционные тесты.

3. Обзор

3.1. Обзор аналогов

Обзор аналогичных систем подразумевал исследование существующих программных продуктов и нахождение уже готовых решений.

3.1.1. Уведомления

Существует множество технологий для рассылки уведомлений. Их можно рассылать на почту, мессенджер, телефон или сделать push-уведомления в браузере. Конечно, отправка уведомлений на почту — это часть, которую нет надобности убирать, потому что в основном на всех ресурсах регистрация проходит через почту, а HwProj 2.0.1 не исключение. Для рассылок на телефон надо указывать свой личный номер телефона, далеко не все люди будут согласны на это, поэтому это может стать проблемой. На push-уведомления в браузере также может согласиться маленький процент людей, потому что к этому аспекту есть недоверие, что также может стать проблемой. Мессенджер остается самым приоритетном вариантом, потому что в мессенджерах люди не бояться подписываться на разные каналы или группы. Также если рассматривать push-уведомления, как на телефон, так и в браузере, то это не подходит под специфику задачи, потому что помимо рассылки уведомлений в мессенджер, было целью реализовать функциональность получении информации по курсу.

3.1.2. Требования к аналогам

При обзоре аналогичных систем рассматривалась поддержка отправки уведомлений на почту, наличие интеграции с мессенджером, возможность получить информацию по курсу с помощью мессенджера и список всевозможных уведомлений, которые могут получить студент и преподаватель.

3.1.3. HwProj [8]

HwProj — сервис для проверки домашних заданий по программированию. Поддерживает отправку уведомлений на почту, но отсутствует поддержка интеграции с мессенджером.

3.1.4. Microsoft Teams [11]

Microsoft Teams — платформа, организующая работу для групп лиц, поддерживает проведение онлайн занятий, сдачу домашних работ и их проверку. В Microsoft Teams были замечены проблемы с поддержкой уведомлений для преподавателей, потому что сервер не поддерживает уведомления о том, что студент сдал домашнее задание, также отсутствует поддержка интеграции с мессенджером.

3.1.5. BlackBoard Learn [3]

BlackBoard Learn — это сервис для дистанционного обучения, позволяет создавать курсы, выкладывать материал, задания и проводить контрольные. Не все уведомления поддерживают разделение между преподавателями и студентами и также отсутствует интеграция с мессенджером, но присутствует отправление уведомлений на почту.

3.2. Обзор технологий

3.2.1. ASP.NET Core [1]

HwProj 2.0.1 разрабатывался на ASP.NET Core, поэтому выбор данной технологии очевиден. ASP.NET Core — фреймворк от компании Microsoft, предоставляющий технологии для разработки веб-приложений и работающий на платформе .Net.

3.2.2. C# [4]

Объектно-ориентированный, типобезопасный язык программирования, работающий на платформе .Net. Для реализации Telegram бота

был выбран C#, потому что для уведомлений и отправки их на почту использовался именно этот язык и вся сервисная часть HwProj 2.0.1 написана на C#. Как уже говорилось язык типобезопасный, а также обладает множеством библиотек и шаблонов.

3.2.3. MailKit [9]

SendGrid [16] и MailKit две популярные библиотеки для реализации отправки уведомлений на почту. Эти библиотеки, предоставляют технологии для отправки сообщений на почту. Был выбран MailKit, потому что для него нашлось больше материала для изучения.

3.2.4. Telegram [17]

Выбор мессенджера для бота был в пользу Telegram. Telegram обладает развитой инфраструктурой для создания ботов, а также в основном связь между студентами и преподавателями по программированию осуществляется через Telegram.

3.2.5. Telegram.Bot [18]

Для реализации Telegram бота была выбрана данная библиотека, потому что у нее присутствует подробная документация с множеством примеров реализации, также является одной из самых популярных библиотек для реализации телеграм бота на C# и на данный момент поддерживается разработчиками.

3.2.6. NUnit [13]

Фреймворк для тестирования в .Net. Довольно популярная библиотека для тестирования. Конечно, это не единственный фреймворк в данной сфере, есть еще xUnit, но фреймворки предоставляют одинаковые возможности. Был выбран фреймворк NUnit, так как опыт работы с ним присутствовал.

3.2.7. AutoFixture [2]

Библиотека для генерации объектов с различными данными.

3.2.8. Moq [12]

Библиотека, позволяющая имитировать разные объекты.

3.2.9. Microsoft SQL Server [10]

Разработанная Microsoft система управления реляционными базами данных. HwProj 2.0.1 использовал такие базы данных, поэтому они также использовались для Telegram бота.

4. Реализация

4.1. Архитектура HwProj 2.0.1

В сервисной части HwProj 2.0.1 выделены такие микросервисы как AuthService, CoursesService, NotificationsService, SolutionsService. Каждое название соответствует логике тому, за что они отвечают. Фронтенд реализован на React [15]. Общение между бэкендом и фронтендом происходит через API Gateway, который перенаправляет запросы с фронтенда на отдельный микросервис. Для уведомлений используется шина сообщений RabbitMQ [14].

На рисунке 1 представлена архитектура HwProj 2.0.1

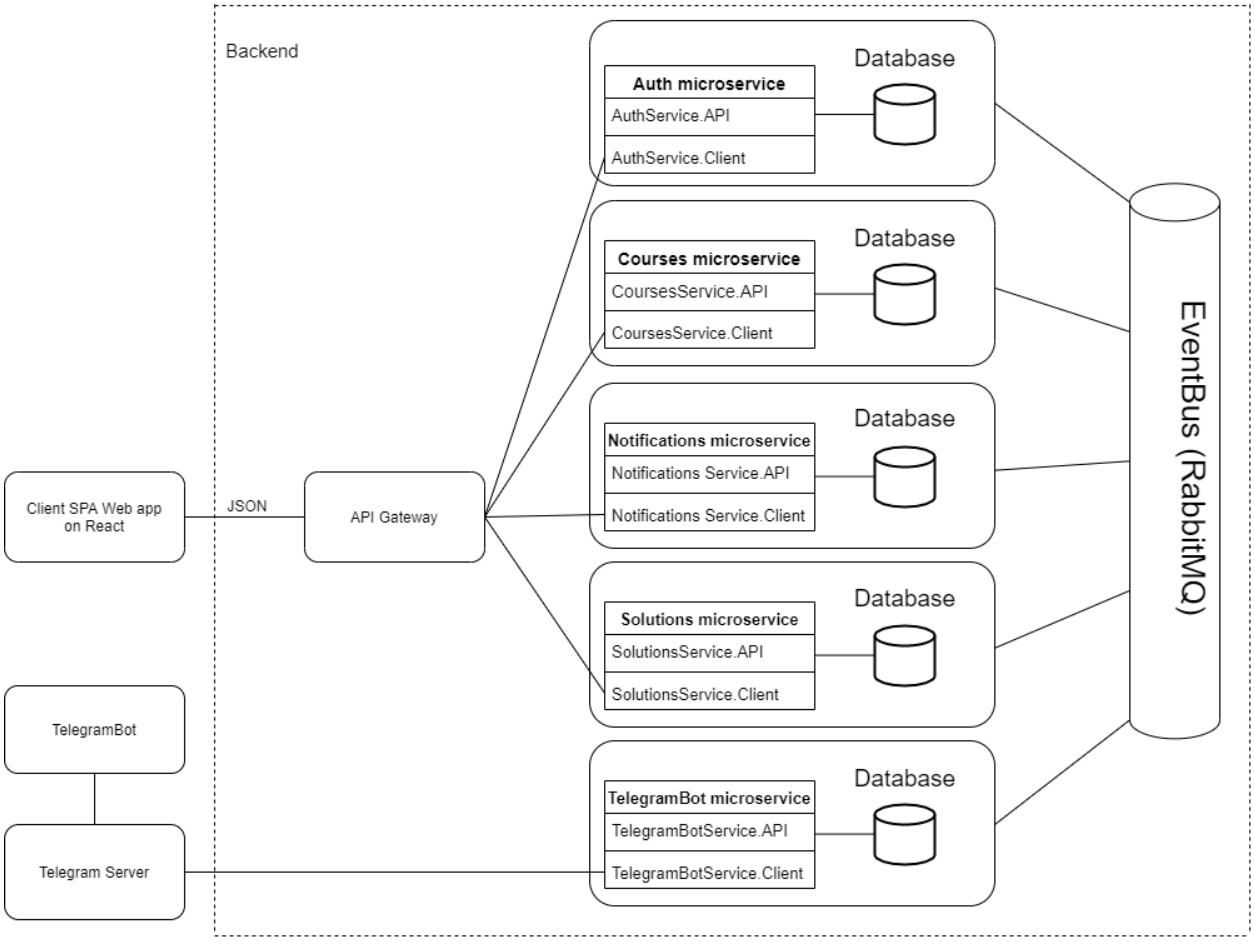

Рис. 1: Архитектура HwProj 2.0.1

4.2. Разработка уведомлений

В каждом микросервисе для каждого уведомления был реализован свой класс с определенным набором свойств. После создания уведомления в соответствущем методе в сервисной части, оно отправляется в шину сообщений. Пример такого класса можно увидеть на листинг 1.

После этого отправленное уведомление обрабатывается соответствующим обработчиком событий. В обработчике создается само уведомление и отправляется соответствующему пользователю на HwProj 2.0.1, на почту и в Telegram бот.

Листинг 1: класс уведомления

```
public class NewCourseMateEvent : Event
{
    public long CourseId { get; set; }
    public string CourseName { get; set; }
    public string MentorIds { get; set; }
    public string StudentId { get; set; }
    public bool IsAccepted { get; set; }
}
```

4.3. Реализация Telegram бота

Архитектура Telegram бота представлена на рисунке 2.

При получении новых сообщений сервер Telegram отправляет update боту, который принимается в контролере. Update — это объект, который содержит большое количество информации о сообщении: данные о пользователе, отправившем это сообщение, текст сообщения, его вид и чат id.

Далее update передается в CommandService, где происходит определение того, выполнения какой команды ожидает пользователь, отправляя данное сообщение.

В каждой команде выполняется простая логика: в ней собираются данные, которые надо предоставить пользователю, затем строится

клавиатура. Клавиатура будет привязана к сообщению, а при нажатии на кнопку, боту отправляется update с информацией, которую мы кладем в CallbackData каждой кнопки. Каждая команда унаследована от абстрактного класса Commands. Абстрактный класс Commands можно увидеть на листинг 2.

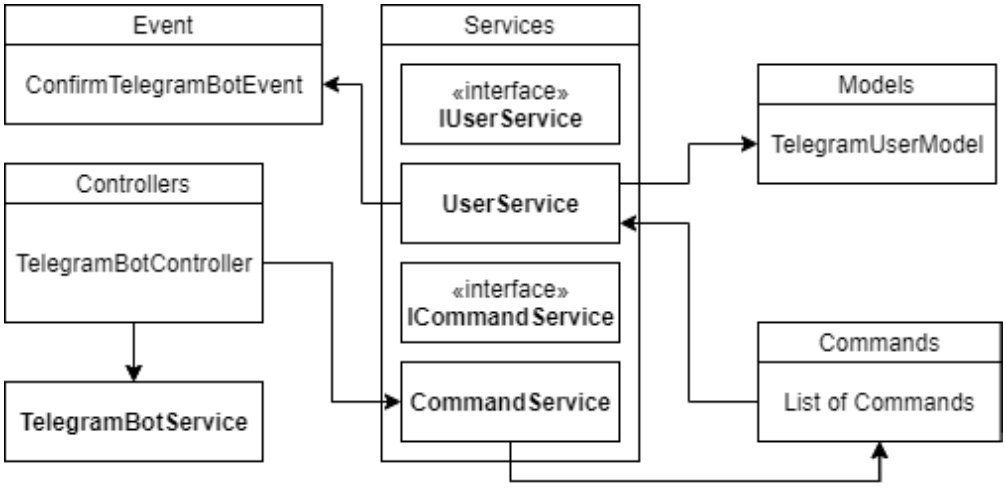

Рис. 2: Архитектура Telegram бота

Листинг 2: класс Commands

```
public abstract class Commands
{
    public abstract string Name { get; }

    public abstract Task ExecuteAsync(Update update);

    protected InlineKeyboardButton GetButton(string text,
    string callbackData)
        => new()
        {
            Text = text,
            CallbackData = callbackData
        };
}
```

4.4. Авторизация в Telegram боте

Авторизация в Telegram боте происходит следующим образом: при запуске бота, пользователя просят ввести почту, которая авторизированна на HwProj 2.0.1. После отправки почты, пользователь получает код, состоящий из букв и цифр на свой профиль в HwProj 2.0.1. Пользователь отправляет боту код, который тот проверяет, если все правильно, то авторизация пройдена успешно.

4.5. Отправка решений в Telegram боте

Отправка решений также происходит в несколько этапов. Пользователь должен выбрать нужную задачу и нажать на кнопку "Отправить решение", бот запоминает задачу и просит отправить ссылку на решение, а потом ввести комментарий. Бот создает модель решения и отправляет ее через клиент в сервис решений.

5. Эксперимент

5.1. Telegram бот

Для проверки Telegram бота также был составлен тестовый сценарий:

- Запустить бота и попытаться получить информацию о своих курсах, без авторизации пользователь не может получить никакой информации.
- После авторизации узнать информацию о своих курсах, пользователь должен увидеть только свои курсы.
- При попытке узнать домашние задание по Курсу "A", пользователь должен увидеть все задания по курсу "A".
- При попытке узнать статистику по курсу "A", пользователь должен увидеть количество своих баллов из всех возможных по курсу "A".
- Выбрать задачу и выполнить нужные действия для отправки решения, оно должно появиться в списке отправленных решений.

5.2. Отправка уведомлений

Для тестирования данной функциональности был составлен следующий тестовый сценарий:

- Выполнить набор действий для каждого уведомления и проверить отправку на почту и в Telegram бота, для которого уведомления могут получать только авторизованные пользователи.

6. Тестирование

6.1. Подход к тестированию

Для тестирования сервиса, мы используем клиент самого сервиса. Отправляем через него запросы к сервису, а затем делаем проверку произошедшего, чтобы удостовериться в том, что все выполнилось правильно.

6.2. Сервис уведомлений

Для сервиса уведомлений были составлены и реализованы, следующие тестовые сценарии:

- Отправка одиночных уведомлений: регистрации, приглашении в преподаватели, заявку студента на вступление в курс приняли или отклонили, преподаватель оценил решение.
- Отправка групповых уведомлений: студент подал заявку на курс, студент выложил решение, преподаватель выложил домашнее задание, преподаватель обновил домашнее задание или задачу
- Просмотр уведомления.

Листинг 3: Пример теста просмотр сообщения

```
[Test]
public async Task CheckMarkAsSeenTest()
{
    var notificationClient = CreateNotificationsServiceClient();

    var (userId, _) = await CreateAndRegisterUser();
    var notificationBefore = await notificationClient.Get(userId,
    new NotificationFilter());
    var notificationId = new long[1]{notificationBefore[0].Id};
    await notificationClient.MarkAsSeen(userId, notificationId);
```

```
    var notificationAfter = await notificationClient.Get(userId,
    new NotificationFilter());

    notificationBefore[0].HasSeen.Should().BeFalse();
    notificationAfter[0].HasSeen.Should().BeTrue();
}
```

Тестовое покрытие клиента равно 100%.

6.3. Сервис решений

Для сервиса решений были составлены и реализованы, следующие тестовые сценарии. В каждом тестовом сценарии создается курс, домашняя работа, преподаватель и студент, но в некоторых сценариях создается несколько преподавателей или студентов.

- Отправка решения без дедлайна.
- Отправка решения с не строгим дедлайном.
- Отправка решения со строгим дедлайном.
- Оценка решения.
- Удаление решения.
- Получение статистики за преподавателя.
- Получение статистики за студента.
- Получить статистику по курсу.

Листинг 4: Пример теста на удаление решения

```
[Test]
public async Task PostByStudentNotFromThisCourseTest()
{
    var (studentId, lectureId) = await CreateUserAndLecture();
    var lectureClient = CreateCourseServiceClient(lectureId);
```

```
    var (_, hwId, taskId) =
    await CreateCourseHwTaskWithOutDeadLine(lectureClient, lectureId);
    var solutionClient = CreateSolutionsServiceClient();
    var solutionViewModel = GenerateSolutionViewModel(studentId);

    Assert.ThrowsAsync<ForbiddenException>(async ()
    => await solutionClient.PostSolution(solutionViewModel, taskId));
}
```

Тестовое покрытие клиента равно 83%.

7. Заключение

В ходе учебной практики были выполнены следующие задачи:

- Проведен обзор существующих аналогов.
- Проведен обзор и разбор архитектуры проекта.
- Сформирован и реализован список недостающих уведомления на сервисе.
- Реализована отправка уведомлений на почту и в Telegram бота.
- Реализована функциональность для получения информации по курсу в Telegram боте.
- Реализована функциональность для отправки решений в Telegram боте.
- Были проведены тестовые сценарии для проверки исправности работы Telegram бота.
- Сформировано и реализовано интеграционное тестирование сервиса уведомлений.
- Сформировано и реализовано интеграционное тестирование сервиса решений.

С кодом можно ознакомиться в соответствующих пулл рек-вестах репозитория на GitHub [5][6][7].

- Telegram Bot[5].
- Тестирование сервиса уведомлений[7].
- Тестирование сервиса решений[6].

Список литературы

[1] ASP.NET. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-6.0/ (online; accessed: 2020-12-14).

[2] AutoFixture. — URL: https://github.com/AutoFixture/AutoFixture (online; accessed: 2020-12-14).

[3] BlackBoard Learn. — URL: https://bb.spbu.ru/ (online; accessed: 2020-12-14).

[4] C#. — URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (online; accessed: 2020-12-14).

[5] Github. — URL: https://github.com/InteIIigeNET/HwProj-2.0.1/pull/133/ (online; accessed: 2020-12-14).

[6] Github. — URL: https://github.com/InteIIigeNET/HwProj-2.0.1/pull/160/ (online; accessed: 2020-12-14).

[7] Github. — URL: https://github.com/InteIIigeNET/HwProj-2.0.1/pull/161 (online; accessed: 2020-12-14).

[8] HwProj. — URL: http://hwproj.me/ (online; accessed: 2020-12-14).

[9] Mailkit. — URL: https://github.com/jstedfast/MailKit/ (online; accessed: 2020-12-14).

[10] Microsoft SQL Server. — URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019 (online; accessed: 2020-12-14).

[11] Microsoft Teams. — URL: https://www.microsoft.com/ru-ru/microsoft-teams/log-in (online; accessed: 2020-12-14).

[12] Moq. — URL: https://github.com/moq/moq4 (online; accessed: 2020-12-14).

[13] NUnit. — URL: https://nunit.org/ (online; accessed: 2020-12-14).

[14] RabbitMQ. — URL: https://www.rabbitmq.com/ (online; accessed: 2020-12-14).

[15] React. — URL: https://reactjs.org/ (online; accessed: 2020-12-14).

[16] SendGrid. — URL: https://github.com/sendgrid/sendgrid-csharp/ (online; accessed: 2020-12-14).

[17] Telegram. — URL: https://telegram.org/ (online; accessed: 2020-12-14).

[18] Telegram.Bot. — URL: https://github.com/TelegramBots/Telegram.Bot/ (online; accessed: 2020-12-14).