

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б10-мм

Разработка инструмента для работы с индивидуальными графиками

Каргин Глеб Павлович

Отчёт по учебной практике
в форме «Производственное задание»

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

Санкт-Петербург
2024

Оглавление

Введение	3
1. Постановка задачи	5
2. Обзор	6
2.1. Обзор существующих решений	6
2.2. Обзор инструментов	8
3. Проектирование	11
4. Реализация	14
4.1. Серверная часть	14
4.2. Пользовательский интерфейс	16
5. Эксперимент	19
Заключение	22
Список литературы	23

Введение

Частыми явлениями в студенческой жизни являются различного рода уважительные причины, такие как плохое самочувствие, заболевания. В СПбГУ на такие случаи учебный отдел составляет индивидуальный график для студентов, которые пропустили зачёт, экзамен, пересдачи по уважительным причинам. Индивидуальный график состоит из перечня дисциплин и устанавливает конкретные даты, время, место для проведения сдачи, пересдачи и комиссии. Таким образом, индивидуальный график является довольно важной частью учебного процесса, так как предоставляет дополнительную попытку в случаях уважительной причины неявки в основной срок.

При составлении индивидуального графика необходимо учитывать следующие факторы: расписание студента, расписание преподавателя, занятость аудитории. Всё это требует достаточного количества времени учебного отдела и, как следствие, негативно сказывается на рабочем процессе. Кроме того, отслеживание новых индивидуальных графиков, которые зачастую приходят по частям и не сразу, и их разбор также занимает часть времени преподавателей, тем самым отвлекая от более приоритетных задач. Помимо перечисленного, стоит отметить, что из-за большого числа студентов-должников становится сложно отследить каждого, так как приходится вручную просматривать текстовые документы индивидуальных графиков.

В связи с этим необходимо разработать инструмент для работы с индивидуальными графиками, который поможет упростить и автоматизировать процесс составления, анализа индивидуальных графиков, позволит сэкономить время учебного отдела и преподавателей, а также снизит риск упустить кого-то из виду.

Таким образом, было принято решение о реализации веб-приложения, которое состоит из двух частей. Было решено начать разработку с преподавательской части с возможностью разбора файлов индивидуальных графиков, поступивших на почту, и просмотра графика, назначенного с использованием сервиса, а затем реализовать часть для учебного

отдела с пользовательским интерфейсом для создания индивидуального графика.

1. Постановка задачи

Целью работы является проектирование и реализация веб-приложения для составления и просмотра расписания индивидуальных графиков. Для её выполнения были поставлены следующие задачи.

- Осенний семестр:
 1. Провести обзор аналогичных инструментов.
 2. Спроектировать общую архитектуру сервиса и базы данных.
 3. Спроектировать пользовательский интерфейс.
 4. Реализовать инструмент для разбора .docx файлов.
 5. Реализовать API сервиса.
- Весенний семестр:
 1. Реализовать преподавательскую часть.
 2. Реализовать часть учебного отдела.
 3. Реализовать аутентификацию и авторизацию пользователей.
 4. Реализовать пользовательский интерфейс.
 5. Провести тестирование реализованного инструмента.

2. Обзор

2.1. Обзор существующих решений

Перед началом разработки веб-приложения были рассмотрены существующие на данный момент инструменты, которые позволили бы организовать процесс работы с индивидуальными графиками, а именно их составление и отображение. Инструмент для работы с индивидуальными графиками должен соответствовать следующим требованиям: возможность с помощью Timetable¹, сервиса с расписанием в СПбГУ, подгружать расписание преподавателей, студентов и проверять свободна ли аудитория, экспортировать календарь экзаменов, экспортировать новые записи и составлять файл с распоряжением, а также возможность отображать все записи или фильтровать их по преподавателям или студентам. Однако в силу специфичности разрабатываемой системы прямых аналогов не существует, поэтому были рассмотрены более общие технологии.

- Системы для совместной работы и отслеживания задач (например, Jira²). Такие системы имеют различные инструменты для работы с документами, в них можно создавать события в календаре и экспортировать их. В системах для совместной работы и ведения задач нет функции автоматического формирования распоряжений. Можно реализовать генератор распоряжений как плагин, однако это не сильно проще реализации системы с нуля, которая будет специализирована под составление индивидуальных графиков и не будет иметь лишней функциональности, перегружающей систему.
- Сервисы для планирования мероприятий (например, Яндекс.Календарь³). К преимуществам этого решения можно отнести простоту использования, возможность учитывать

¹Timetable. URL: <https://timetable.spbu.ru> (дата обращения: 2024-06-17)

²Jira. URL: <https://www.atlassian.com/ru/software/jira> (дата обращения: 2024-06-17)

³Яндекс.Календарь. URL: <https://calendar.yandex.ru> (дата обращения: 2024-06-17)

занятость участников мероприятий, возможность экспорта мероприятий и проведения видеовстреч (представлено на Рис. 1). При этом такие сервисы обладают рядом недостатков, таких как отсутствие возможности работы с документами и формирования файла с распоряжением. И в случае как и с системами для совместной работы возможно реализовать генератор, однако это будет сторонний сервис, который будет иметь ряд ограничений, например на количество вызовов API.

Новое событие

Название:

+ Описание

Телемост:

Время и дата: —
☐ Весь день ☐ Повторять

Участники:

Опциональные участники:
☐ Учитывать занятость опциональных участников

Занятость участников

Понедельник, 1 января															
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Я st098191@student.s...															

Место:

Уведомления: за час

Рис. 1: Создание мероприятия в Яндекс.Календарь

2.1.1. .NET

Платформа .NET [6] была выбрана ввиду поддержки большого числа языков программирования, которые ранее использовались при реализации различных сервисов для университета, что позволяет с легкостью интегрировать эти сервисы. Также .NET имеет обширную документацию и огромное сообщество. Для обеспечения дальнейшей поддержки проекта был выбран язык программирования C#, так как это широко используемый язык для разработки под .NET.

2.2. Обзор инструментов

2.2.1. ASP.NET Core

ASP.NET Core [1] — это фреймворк, который позволяет создавать веб-приложения на платформе .NET. В традиционном подходе ASP.NET применяются контроллеры, которые обрабатывают запросы к определённым URL-адресам (эндпойнтам). Каждый метод контроллера отвечает за обработку запросов к конкретному эндпойнту и часто возвращает веб-страницу, сгенерированную с помощью шаблонного движка Razor. Однако был выбран подход Minimal API, который особенно удобен для разработки небольших веб-сервисов, так как он позволяет формировать ответ как результат работы функции, который автоматически сериализуется в JSON. Необходимо лишь указать выбранный метод HTTP-запроса, путь и лямбда-выражение, которое будет вызываться при обращении по заданному пути с заданными HTTP-методом и параметрами.

2.2.2. Entity Framework

Entity Framework [5] — это ORM (Object-Relational Mapping) фреймворк, который позволяет разработчикам работать с базой данных, используя объектно-ориентированные модели. ORM-фреймворк автоматически управляет преобразованием данных между реляционной базой данных и объектами в коде, что упрощает создание и управление моделями, отношениями и связями. В фреймворке присутствует класс DbContext, который позволяет подключиться и делать запросы к базе данных. Кроме того, обычно определяются свойства у класса-наследника DbContext вида DbSet<TEntity>, где TEntity — это классы моделей, представляющих таблицы внутри базы данных. Всё это позволяет работать с базой данных на более высоком уровне абстракции, не беспокоясь о написании сложных SQL-запросов вручную, улучшая при этом поддерживаемость кода.

2.2.3. React

React [12] — библиотека, которая используется для разработки пользовательских интерфейсов. Данная библиотека была выбрана, так как автор уже имел опыт работы с ней, а также для данной библиотеки существует множество полезных при разработке интерфейсов библиотек. Для работы с этой библиотекой использовался язык программирования TypeScript, поскольку благодаря статической типизации языка легче отлавливать ошибки в коде.

2.2.4. Axios

Axios [2] — библиотека, предназначенная для выполнения HTTP-запросов в браузере и на стороне сервера. Библиотека обеспечивает удобную работу с запросами, поддерживает автоматическую трансформацию данных и включает встроенные механизмы обработки ошибок, что упрощает процесс разработки по сравнению с использованием нативных методов, таких как XMLHttpRequest или fetch.

2.2.5. PostgreSQL

PostgreSQL [11] — система управления базами данных с открытым исходным кодом. PostgreSQL широко применяется в индустрии, так как поддерживает сложные запросы и расширенные функции. Кроме того, автор уже имел опыт работы с данной системой управления базами данных.

2.2.6. Docker

Для развертывания базы данных, сервера, клиента на различных платформах требуется контейнеризация. Docker [3] позволяет автоматизировать процесс развертывания сервисов с помощью создания контейнеров. Полученные контейнеры зависимы, поэтому возникает необходимость в оркестраторе. В качестве оркестратора был выбран Docker Compose [4], потому что он не требует сложной настройки и подходит

для небольших проектов.

3. Проектирование

Для реализации сервиса сначала было необходимо спроектировать архитектуру веб-приложения, базу данных и макеты пользовательского интерфейса.

Архитектура веб-приложения (представлено на Рис. 2) состоит из пользовательского интерфейса с использованием React, серверной части с использованием ASP.NET Core, базы данных в PostgreSQL, адаптера для интеграции Timetable и сервисов календарей.

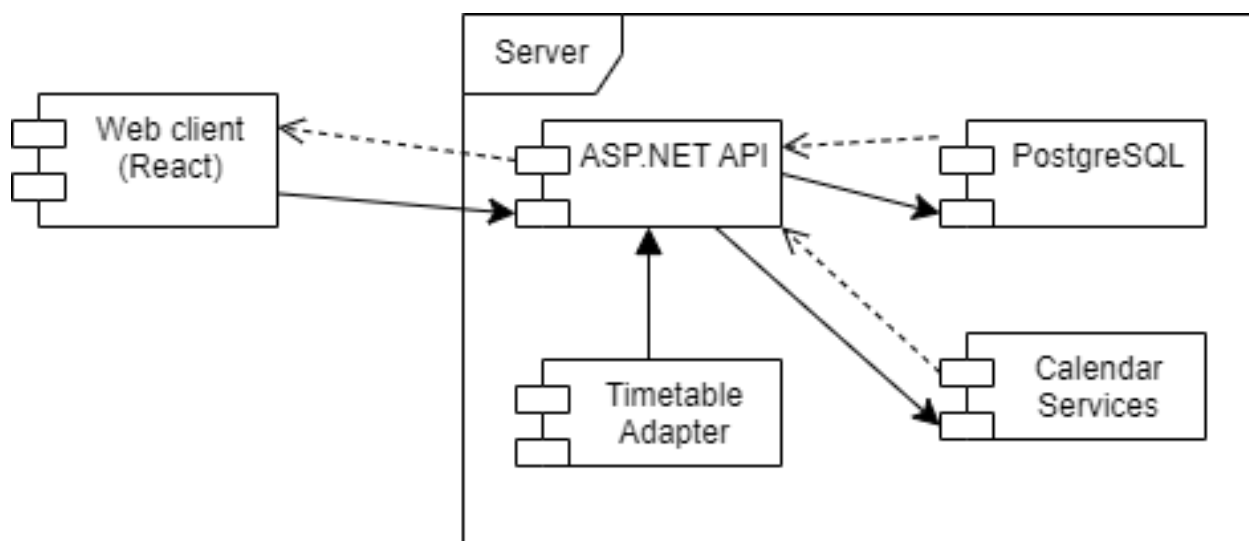


Рис. 2: Диаграмма компонентов приложения

Во избежание избыточности и несогласованности данных при проектировании базы данных (схема базы данных представлена на Рис. 3) была проведена нормализация. Здесь представлены таблицы Role, где определены роли персонала, Staff — данные персонала, Student — данные студента, Exam_Type — типы экзаменов (сдача, пересдача, комиссия), Location — адреса аудиторий. Кроме того, так как на экзамене могут присутствовать несколько преподавателей, возникает отношение «многие ко многим», которое порождает таблицу Exam_Lecturer, содержащую идентификаторы экзаменов и соответствующих им преподавателей.

Пользовательский интерфейс инструмента по работе с индивидуальными графиками состоит из преподавательской части и части учебно-

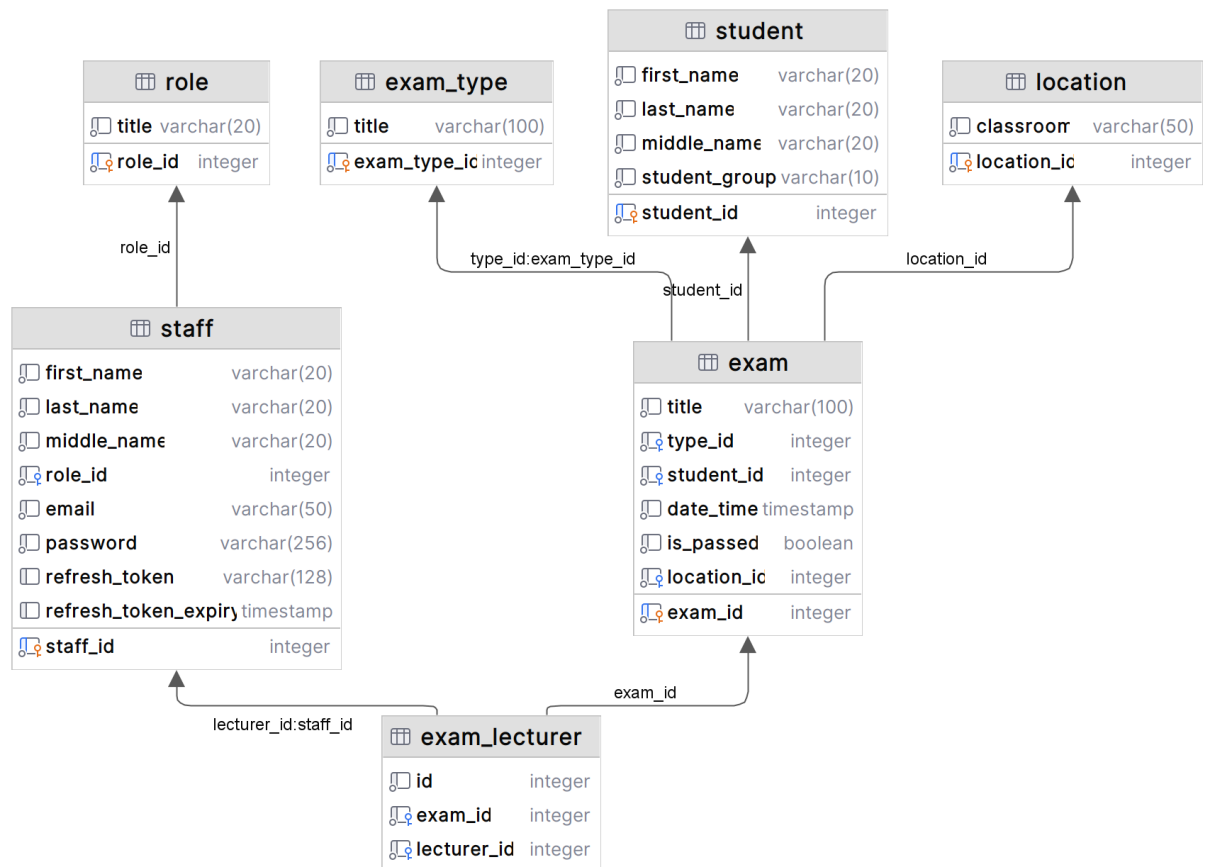


Рис. 3: Схема базы данных

го отдела, которые содержат страницы создания и отображения индивидуальных графиков. Поэтому ввиду малого количества страниц и простоты интерфейса было принято решение использовать сервис для проектирования (представлено на Рис. 4) Ninjamock [8], который позволяет сфокусироваться на пользовательском опыте и взаимодействии с интерфейсом.

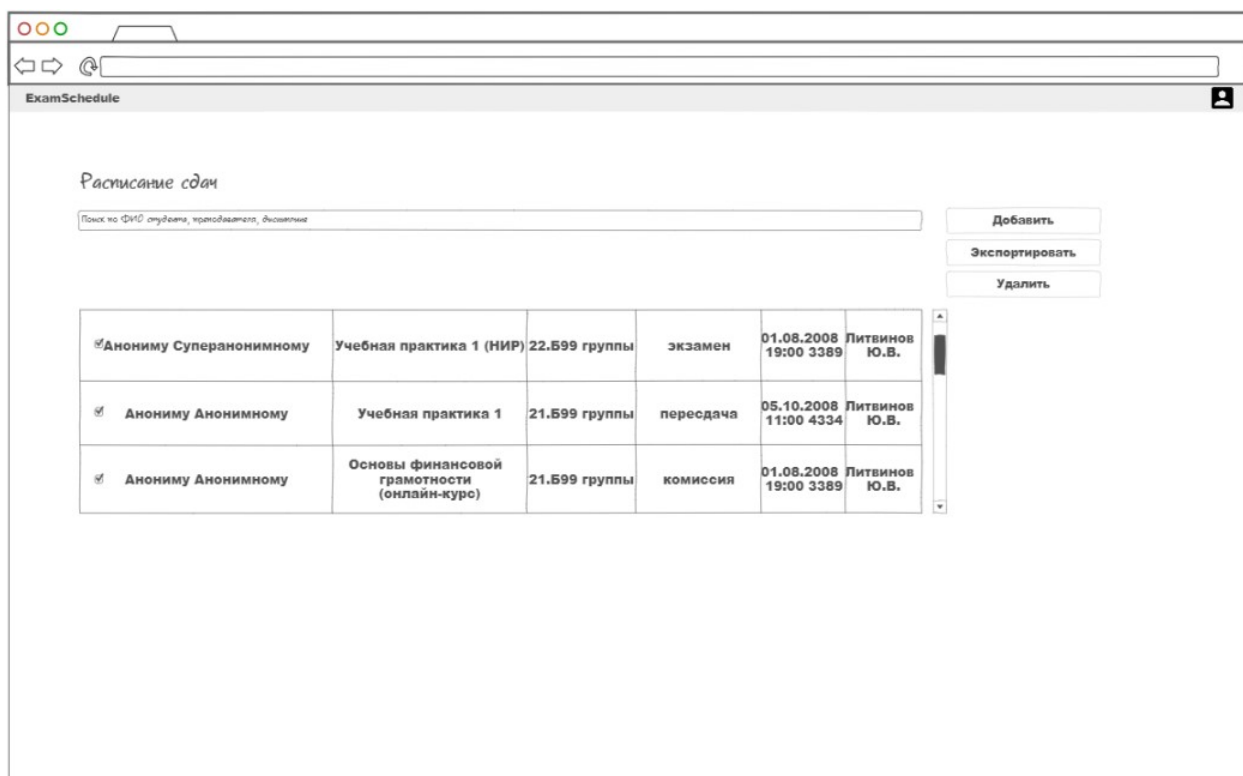


Рис. 4: Мокап страницы отображения расписания

4. Реализация

4.1. Серверная часть

Поскольку проблемы, связанные с индивидуальными графиками, являются актуальными, необходимо было создать часть реализации как можно быстрее. Чтобы не ждать готовности всего сервиса, было принято решение начать с реализации преподавательской части.

Для преподавателей были реализованы библиотека разбора файла индивидуального графика, соответствующий эндпойнт и консольное приложение, которое принимает на вход путь до таблицы, путь до файла формата .docx с индивидуальными графиками. Создается экземпляр класса `ScheduleParser` и в качестве аргумента передается файловый поток индивидуального графика. Затем из полученного файлового потока формируется документ, который разбивается на абзацы, содержащие данные студента и таблицу с индивидуальным графиком для него. После необходимые данные из документа извлекаются в соответствии с указанными полями и заносятся в экземпляр класса `DataTable`, который позволяет хранить таблицу с информацией. У экземпляра парсера вызывается метод `ParseToTable` и в качестве аргумента передается файловый поток таблицы, в которую необходимо внести изменения. Затем из полученного файлового потока вновь извлекаются данные в объект `DataTable` с целью избежания повторных записей. В результате данная таблица заполняется всеми данными из экземпляра `DataTable` и сохраняется уже с обновленными записями о сдачах.

Основой реализации была разработка логики веб-приложения. Для этого были созданы модели сущностей, класс `ScheduleContext` как потомок класса `DbContext`, а также определены свойства вида `DbSet<TEntity>`. Строка подключения к базе данных хранится в файле «appsettings.json». Всё это необходимо для работы с базой данных и реализации API. Затем были реализованы классы, содержащие необходимые запросы к базе данных для конкретных сущностей, а также добавлены соответствующие пути в API (представлено на Рис. 5).

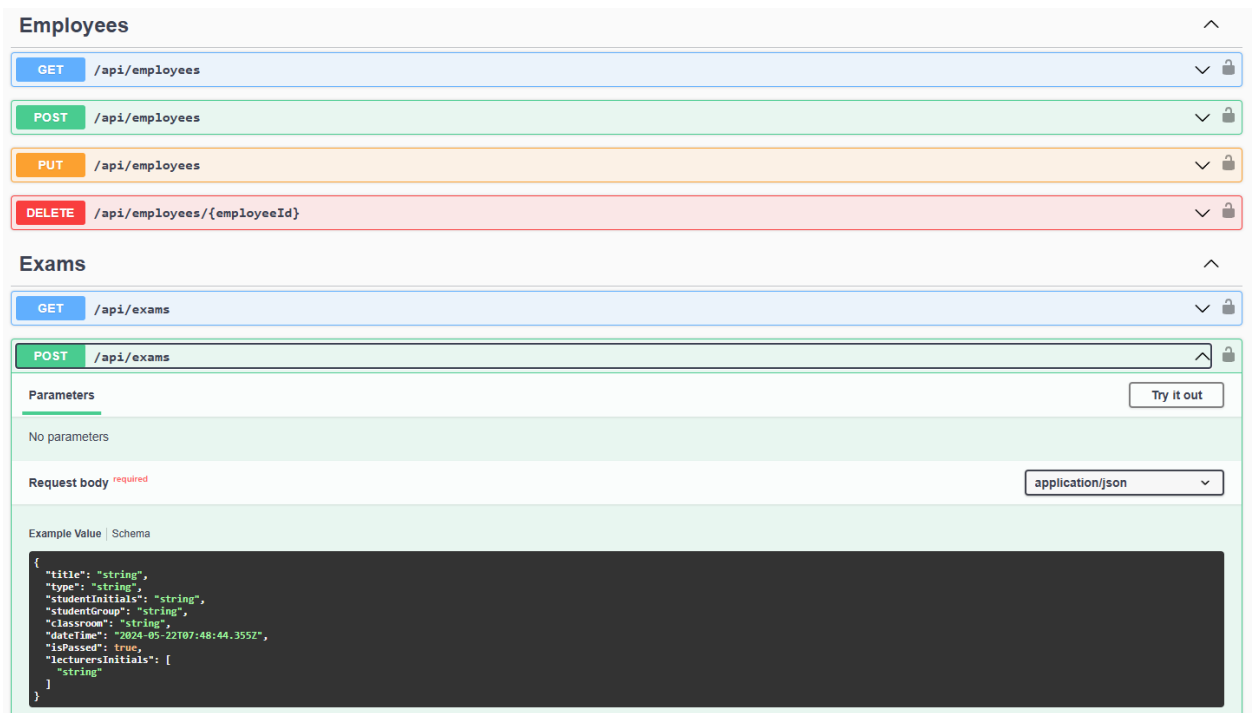


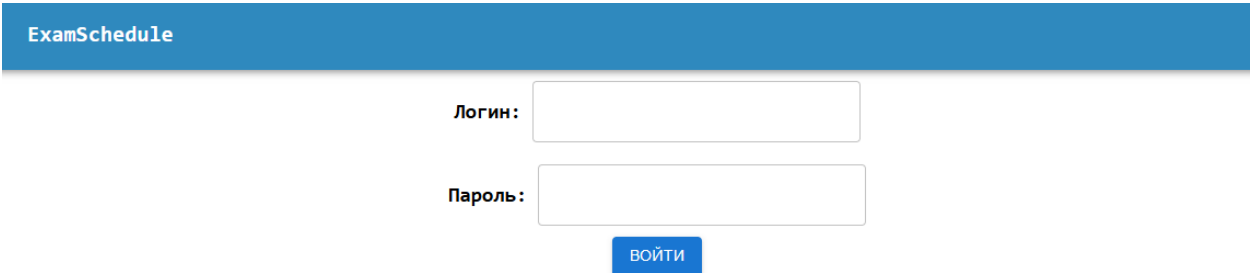
Рис. 5: Документация реализованного API

Немаловажной частью приложения являются авторизация и аутентификация. Было решено реализовать аутентификацию с использованием JWT-токенов, поскольку автор уже имел опыт работы с данным типом аутентификации. Для ее реализации потребовалось добавить NuGet [9] пакет Microsoft.AspNetCore.Authentication.JwtBearer, который предоставляет промежуточное программное обеспечение для реализации аутентификации пользователя. Затем были прописаны соответствующие политики, накладывающие ограничения на запросы, а также было внедрено промежуточное программное обеспечение в веб-приложение и добавлены требования авторизации и выполнения политик. Аутентификация с использованием JWT-токенов подразумевает, что при аутентификации пользователь получает access-токен, который в дальнейшем используется при обращении к различным эндпоинтам. Однако проблемой является тот факт, что время у access-токена ограничено, причем рекомендуется устанавливать его время жизни как можно меньше в целях безопасности. Всё это приводит к тому, что пользователю приходится довольно часто заново получать access-токен для работы

с веб-приложением, тем самым вызывая дискомфорт в его использовании. Поэтому было решено добавить к стандартной реализации refresh-токен, который также ограничен по времени, но имеет больший период жизни и позволяет получить новый access-токен. Таким образом, были добавлены пути «/login» и «/refresh», которые принимают на вход адрес электронной почты, пароль и refresh, access-токен соответственно и представляют аутентификацию, выдачу токенов, а также получение нового access-токена.

4.2. Пользовательский интерфейс

Для реализации пользовательского интерфейса использовался инструмент сборки Vite [13], при помощи которого был создан проект с использованием библиотеки React. Был создан объект Axios для получения данных посредством HTTP запроса, настроен заголовок, содержащий access-токен, а также авторизация и аутентификация на клиенте сервиса. Для этого была реализована страница входа (представлено на Рис. 6), которая принимает на вход логин и пароль пользователя, а затем через ранее реализованную функцию обращается к API для получения access и refresh-токенов.



ExamSchedule

Логин:

Пароль:

Рис. 6: Страница входа

Кроме того, были реализованы страницы отображения расписания индивидуальных графиков (представлено на Рис. 7). Над таблицей расположена поисковая строка, где можно отсортировать записи по введенным туда ФИО, дисциплине, группе или же типу экзамена. Ниже

располагаются две таблицы «Расписание сдач» и «Сдано», где соответственно располагаются предстоящие сдачи и уже неактуальные. У каждой записи рядом располагаются кнопки, с помощью которых можно отметить экзамен как уже сданный/несданный или просто удалить его. При нажатии на иконку пользователя появляется меню, в котором можно выйти из учетной записи, нажав на кнопку «Выйти».

ExamSchedule

Поиск по ФИО, дисциплине, группе, типу экзамена

Профиль

Выйти

Расписание сдач

ДОБАВИТЬ

Сдан/не сдан	Id	Студент	Дисциплина	Группа	Тип	Место проведения	Дата	Преподаватели
↑↓	1	Аноним Анонимов	Экзамен	22.Б22	Сдача	3389	30.08.2022 10:10	Лекторов Лектор Лекторович
↑↓	2	Аноним Анонимов	Другой экзамен	22.Б22	Сдача	3389	30.10.2022 10:10	Лекторов Лектор Лекторович

СДАНО ^

Сдан/не сдан	Id	Студент	Дисциплина	Группа	Тип	Место проведения	Дата	Преподаватели
--------------	----	---------	------------	--------	-----	------------------	------	---------------

Рис. 7: Страница отображения расписания для сотрудников учебного отдела

Для сотрудников учебного отдела на странице отображения расписания находится кнопка «Создать», которая открывает диалоговое окно по созданию записи (представлено на Рис. 8). В данном диалоговом окне сотрудник вводит ФИО студента, его группу, дисциплину, ФИО преподавателя, тип экзамена, дату, время и место проведения. При вводе ФИО преподавателя, а также места проведения справа отображается соответствующее расписание, которое позволяет проанализировать, в какой день назначить индивидуальный график. После ввода указанных данных необходимо нажать кнопку «Создать», после чего всплывет окно и проинформирует об успешности операции.

17

ФИО студента:

Анонимов Аноним Анонимович

Группа:

22.510

Дисциплина:

Структуры и алгоритмы компьютерн

ФИО преподавателя:

Литвинов Юрий Викторович

Тип экзамена:

Комиссия

Дата и время проведения:

05/22/2024 10:00

Место проведения:

Университетский проспект, д. 28, ли

Среда

Архитектура и проектирование информационных систем, экзамен (передача)
Время: 11:15-12:45
Даты: 27.3
Группа: 068705 Архитектура и проектирование информационных систем (Основной курс, Траектория 1 семестра) (год начала обучения - 2023)
Экзетив. Архитектура и проектирование систем, экзамен (передача)
Время: 11:15-12:45
Даты: 27.3
Группа: 20.506-мех
Защита выпускной квалификационной работы, апелляционная комиссия
Время: 15:00-17:00
Даты: 22.5
Группа: 20.511-мех
Экзетив. Языки программирования и виртуальные машины, зачёт (комиссия)
Время: 18:00-19:30
Даты: 24.4
Группа: 20.518-мех

Среда

Структуры и алгоритмы компьютерной обработки данных, контрольная работа
Время: 9:30-11:05
Даты: 29.5
Группа: 22.609-мех [080187] Структуры и алгоритмы компьютерной обработки данных (подгруппа 2)
Преподаватель: Черноусов А. В., ассистент
Программирование, зачёт
Время: 13:40-15:15
Даты: 29.5
Группа: 23.607-мех [0802212] Программирование (подгруппа 1)
Преподаватель: Черноусов А. В., преподаватель
Программирование, зачёт
Время: 15:15-16:50
Даты: 29.5
Группа: 23.С83-мех [0803301] Программирование (подгруппа 1)
Преподаватель: Васильев А. А., младший научный сотрудник

создать

закрыть

Рис. 8: Диалоговое окно создания записи

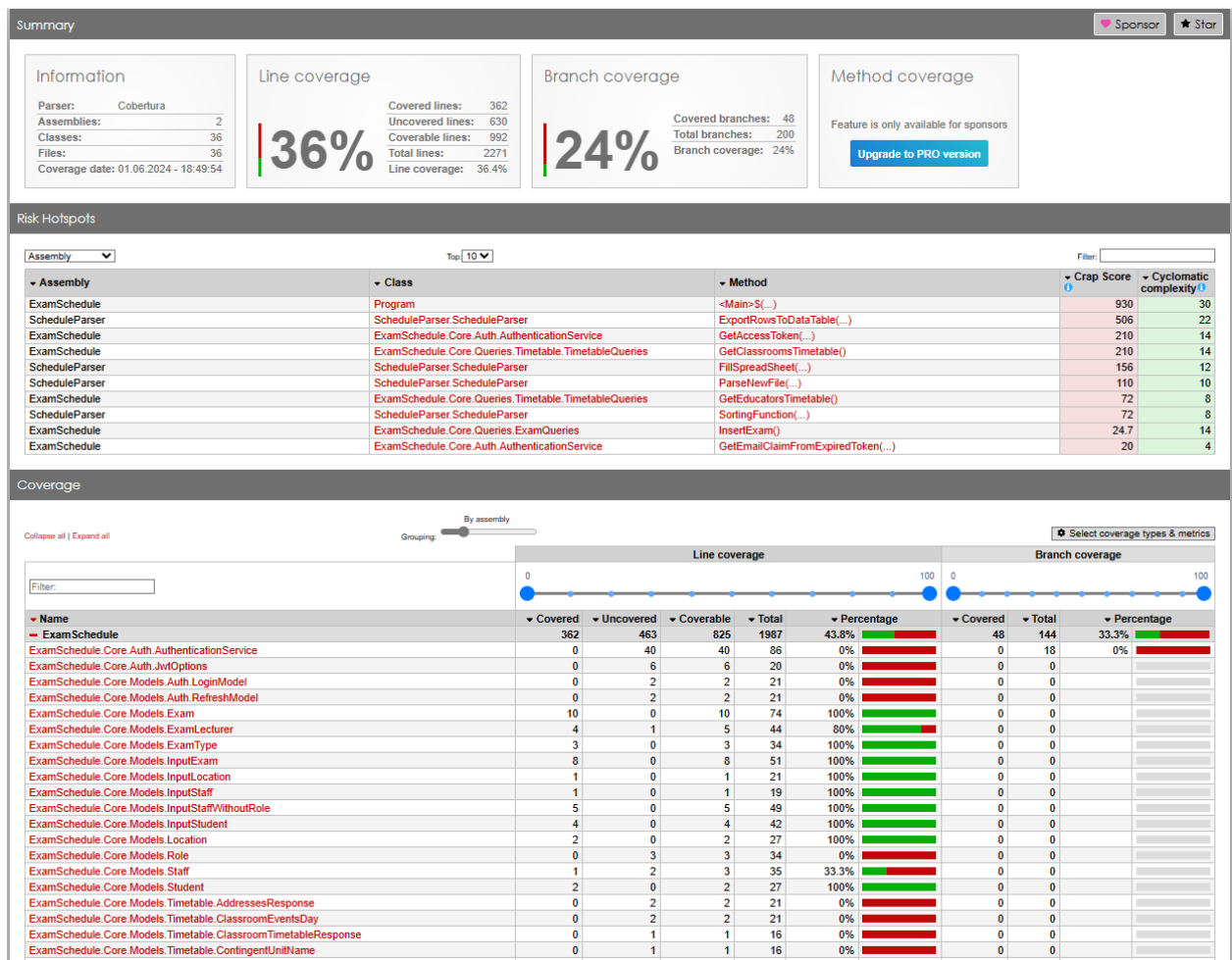
18

5. Эксперимент

Для обеспечения качества реализованного веб-приложения было проведено тестирование — модульное тестирование серверной части приложения, а также сквозное тестирование пользовательского интерфейса.

В качестве библиотеки для модульного тестирования использовался NUnit [7]. Стоит отметить, что были реализованы лишь основные тесты, которые проверяют работоспособность системы в целом, именно поэтому процент покрытия составляет 36% (представлено на Рис. 9). Реализованные модульные тесты были добавлены в непрерывную интеграцию, тем самым делая данные тесты регрессионными, которые в случае нововведений в кодовой базе обнаружат ошибку в ранее протестированных сценариях.

Кроме того, для автоматизации сквозного тестирования использовалась наиболее популярная библиотека Playwright [10]. Был проведен ряд тестов (представлено на Рис. 10), которые позволяют протестировать пользовательский опыт и реакцию реализованного интерфейса на действия пользователя.



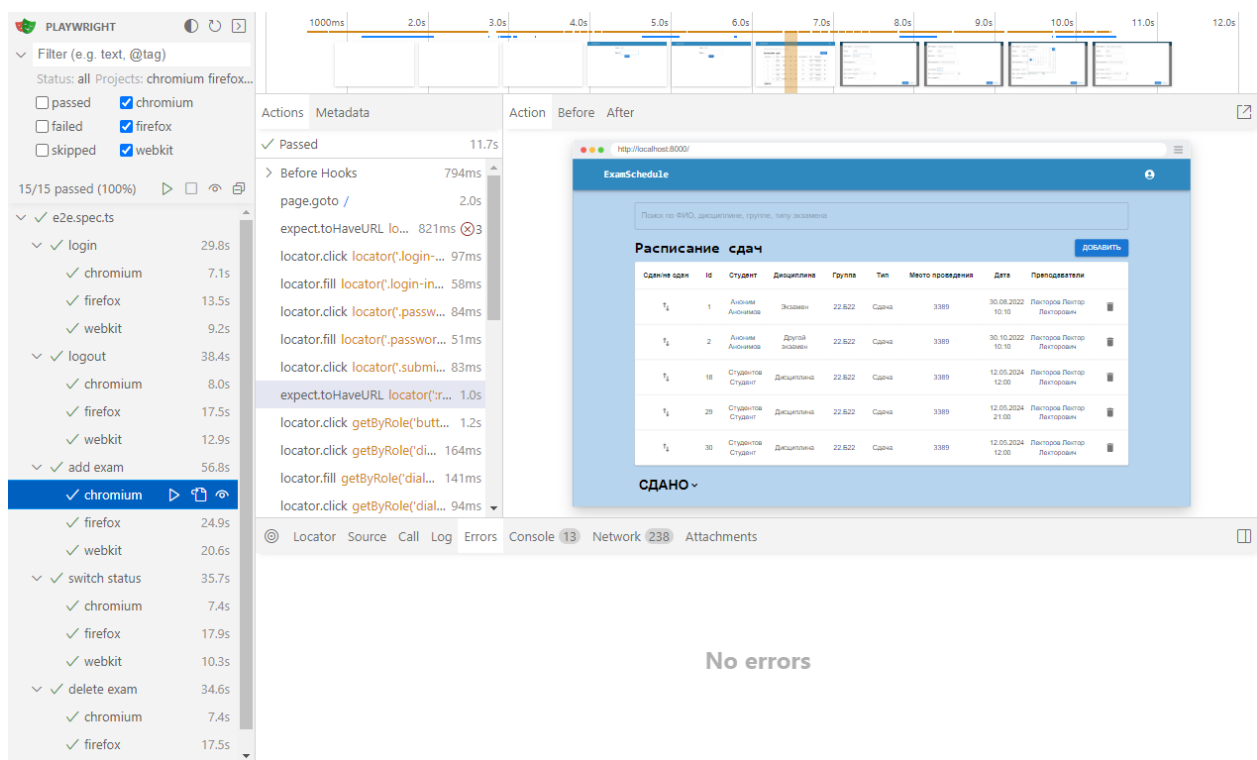


Рис. 10: Сквозное тестирование веб-приложения

Заключение

В ходе выполнения данной работы были получены следующие результаты.

- Проведен обзор аналогичных инструментов.
- Спроектирована общая архитектура сервиса и базы данных.
- Спроектирован пользовательский интерфейс.
- Реализован инструмент для разбора .docx файлов.
- Реализован API сервиса.
- Реализована преподавательская часть.
- Реализована часть учебного отдела.
- Реализована аутентификация и авторизация пользователей.
- Реализован пользовательский интерфейс.
- Проведено тестирование реализованного инструмента.

Код доступен в репозитории на GitHub⁴.

⁴<https://github.com/yurii-litvinov/ExamSchedule>. [Имя аккаунта: Belgrak]

Список литературы

- [1] ASP.NET Core. — URL: <https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-8.0> (дата обращения: 2024-06-17).
- [2] Axios. — URL: <https://axios-http.com/docs/intro> (дата обращения: 2024-06-17).
- [3] Docker. — URL: <https://www.docker.com> (дата обращения: 2024-06-17).
- [4] Docker Compose. — URL: <https://docs.docker.com/compose> (дата обращения: 2024-06-17).
- [5] Entity Framework. — URL: <https://learn.microsoft.com/ru-ru/ef> (дата обращения: 2024-06-17).
- [6] .NET. — URL: <https://dotnet.microsoft.com/ru-ru> (дата обращения: 2024-06-17).
- [7] NUnit. — URL: <https://nunit.org> (дата обращения: 2024-06-17).
- [8] Ninjamock. — URL: <https://ninjamock.com> (дата обращения: 2024-06-17).
- [9] NuGet. — URL: <https://www.nuget.org/packages> (дата обращения: 2024-06-17).
- [10] Playwright. — URL: <https://playwright.dev> (дата обращения: 2024-06-17).
- [11] PostgreSQL. — URL: <https://www.postgresql.org> (дата обращения: 2024-06-17).
- [12] React. — URL: <https://reactjs.org> (дата обращения: 2024-06-17).
- [13] Vite. — URL: <https://vitejs.dev> (дата обращения: 2024-06-17).