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 **4**

2. Обзор **5**

2.1. Обзор существующих решений 5

2.2. Обзор инструментов 6

3. Метод **8**

3.1. Проектирование 8

3.2. Реализация 9

4. Эксперимент **12**

Заключение **13**

Список литературы **14**

Введение

Одной из частых проблем в студенческой жизни являются плохое самочувствие и заболевания. Поэтому в СПбГУ для студентов, пропустивших по уважительной причине зачёт или экзамен, учебный отдел назначает индивидуальный график, который дает право на дополнительную попытку. Индивидуальный график включает в себя перечень дисциплин, которые необходимо сдать определенному студенту, а также устанавливает конкретные даты для сдачи, пересдачи и комиссии.

Составление индивидуальных графиков требует много времени и усилий учебного отдела, что негативно сказывается на их рабочем процессе, отвлекая от более приоритетных задач. Также к проблемам можно отнести отслеживание и анализ со стороны преподавателей индивидуальных графиков, которые зачастую приходят не одновременно, а по частям, тем самым приходится вручную проверять письма, отправленные учебным отделом, вместо того, чтобы потратить данное время на учебную и исследовательскую деятельность. Кроме того, ввиду огромного количества студентов-должников велика вероятность упустить кого-то из-за необходимости вручную просматривать текстовые документы индивидуальных графиков, что затем приведет к неприятным последствиям.

В связи с этим необходимо разработать инструмент для работы с индивидуальными графиками, который позволил бы упростить и автоматизировать процесс составления и анализа индивидуальных графиков, тем самым помогая сэкономить время учебного отдела и преподавателей, а также минимизировать риск пропустить кого-то.

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 макет и уточнить требования у учебного отдела.
    3. Реализовать часть учебного отдела.
    4. Провести тестирование и апробацию инструмента.

2. Обзор

2.1. Обзор существующих решений

В процессе разработки был проведен обзор существующих инструментов, позволяющих организовать составление индивидуальных графиков, а также их просмотр. Важными требованиями к инструменту по работе с индивидуальными графиками являются возможность с помощью Timetable[1], сервиса с расписанием в СПбГУ, подгружать расписание преподавателей, студентов и проверять свободна ли аудитория для проведения экзамена, экспортировать календарь экзаменов, экспортировать новые записи и составлять файл с распоряжением, а также возможность отображать все записи или фильтровать их по преподавателям или студентам.

- Системы для совместной работы и отслеживания задач (например, Jira[2]). Преимуществами таких систем являются большое количество различных инструментов для работы с документами, возможность создания мероприятий в календаре и их экспорт. Однако системы для совместной работы и отслеживания задач не позволяют автоматически составлять файл с распоряжением.

- Сервисы для планирования мероприятий (например, Яндекс.Календарь[3]). К плюсам такого решения можно отнести простоту в использовании, возможность учитывать занятость участников мероприятий, экспортировать мероприятия и проводить видеовстречи (представлено на Рис. 1). При этом такие сервисы обладают рядом недостатков, таких как невозможность работы с документами и отсутствие возможности формирования файла с распоряжением.

[1]Timetable. https://timetable.spbu.ru/ — [Дата обращения: 2023-12-08]
[2]Jira. https://www.atlassian.com/ru/software/jira — [Дата обращения: 2023-12-08]
[3]Яндекс.Календарь. https://calendar.yandex.ru — [Дата обращения: 2023-12-08]

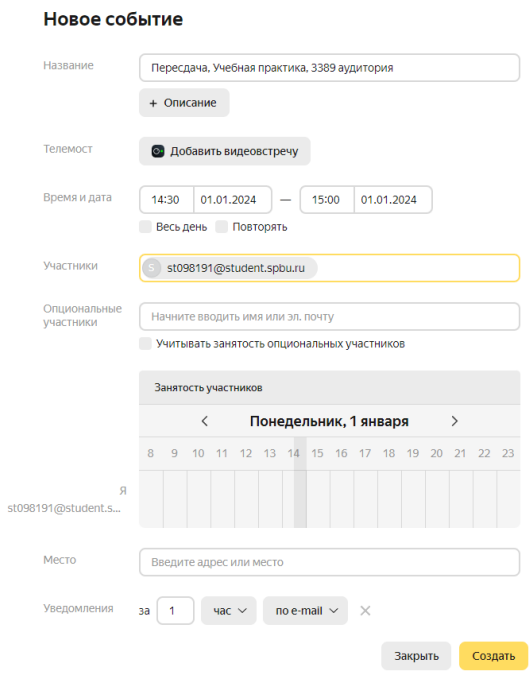

Рис. 1: Создание мероприятия в Яндекс.Календарь

2.2. Обзор инструментов

2.2.1. ASP.NET Core

ASP.NET Core [1] — фреймворк, который позволяет создавать веб-приложения на платформе .NET [4]. В качестве подхода к разработке API интерфейсов использовался Minimal API, который упрощает процесс разработки API за счет создания эндпойнтов без использования контроллеров и шаблонов, указав выбранный метод, путь и лямбда-выражение, которое вызывается при обращении по заданному пути с заданными методом и параметрами. Для обеспечения дальнейшей поддержки проекта был выбран язык программирования C#, так как это широко используемый язык для разработки под .NET.

2.2.2. React

React [10] — библиотека, которая используется для разработки пользовательских интерфейсов. Данная библиотека была выбрана, так как

[4].NET. https://dotnet.microsoft.com/en-us/ — [Дата обращения: 2023-12-08]

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 TypeScript, поскольку благодаря статической типизации языка легче отлавливать ошибки в коде.

2.2.3. Axios

Axios [2] — библиотека, предназначенная для выполнения HTTP-запросов в браузере и на стороне сервера. Данная библиотека выделяется своей простотой использования, удобством работы с запросами, а также обработкой ошибок.

2.2.4. PostgreSQL

PostgreSQL [9] — система управления базами данных с открытым исходным кодом. Она обеспечивает эффективное хранение и манипулирование данными, предоставляя поддержку сложных запросов. Данная система управления базами данных была выбрана поскольку автор имеет опыт работы с ней.

2.2.5. 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 инструменте Docker Compose [4], который позволяет объединить несколько контейнеров и конфигурировать их взаимодействие.

3. Метод

3.1. Проектирование

Перед реализацией сервиса 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.

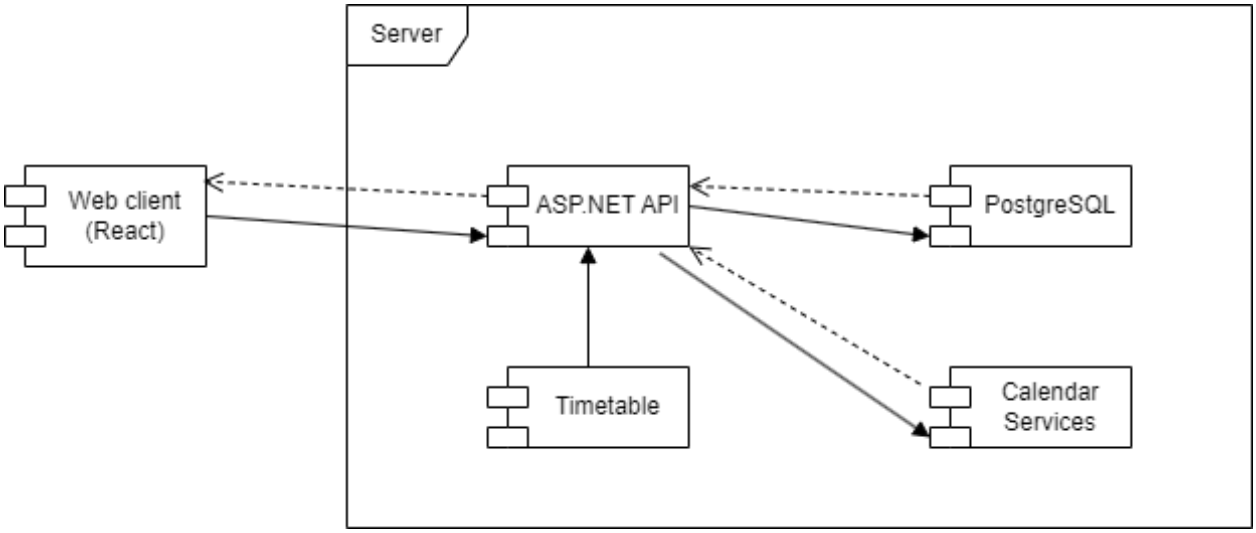

Рис. 2: Диаграмма компонентов приложения

При проектировании базы данных (схема базы данных представлена на Рис. 3) была проведена нормализация для избежания избыточности и несогласованности данных. Таблица Exam является основным объектом данного веб-приложения и содержит такие свойства как название, тип (сдача, пересдача, комиссия), студент, дата и время, место проведения. Так как на экзамене могут присутствовать несколько преподавателей, возникает отношение «многие ко многим», которое разбивается на три таблицы.

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

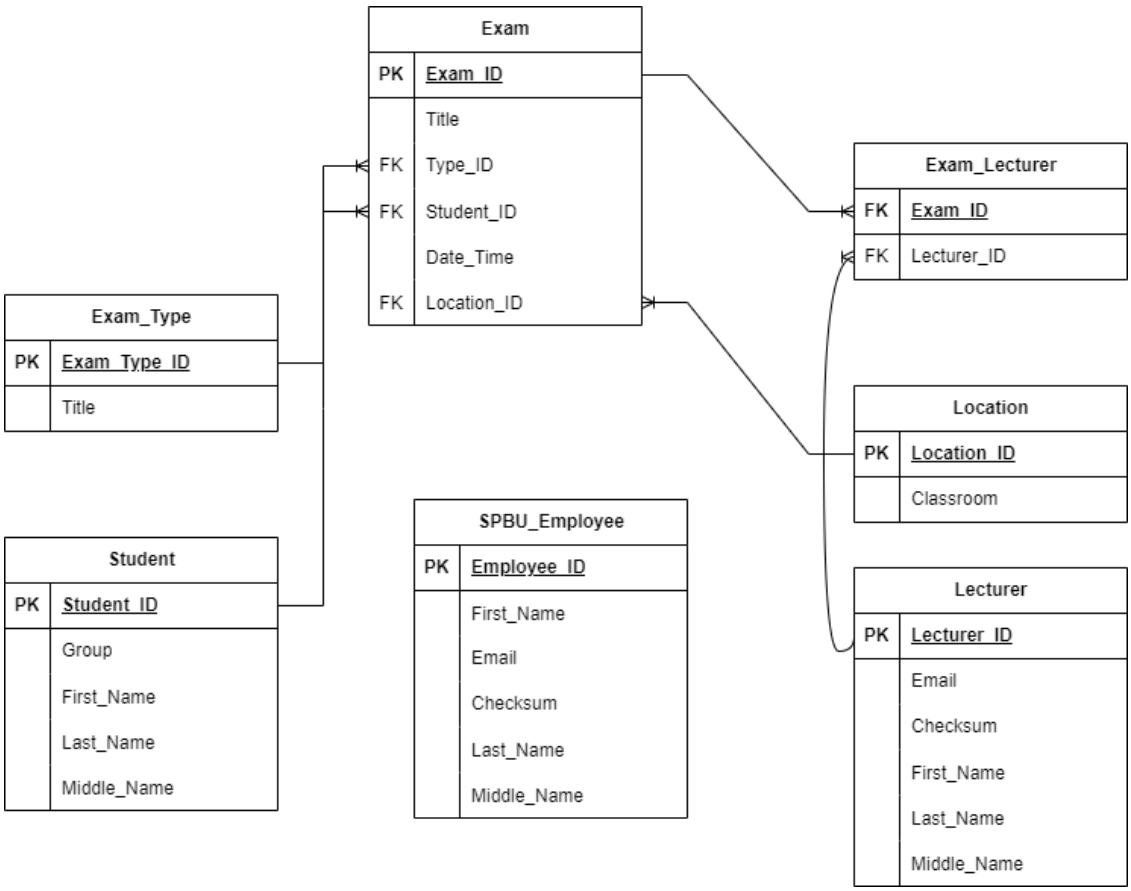

Рис. 3: Схема базы данных

отдела, которые содержат страницы создания и отображения индивидуальных графиков. Ввиду малого количества страниц и простоты интерфейса для его проектирования (представлено на Рис. 4) использовался сервис Ninjamock [6], который позволяет сфокусироваться на пользовательском опыте и взаимодействии.

3.2. Реализация

В связи с актуальностью проблем индивидуальных графиков была необходимость создать часть реализации как можно скорее. Поэтому во избежание ожидания готовности всего сервиса было принято решение начать с реализации преподавательской части.

Для преподавателей был реализован инструмент, который принимает на вход OAuth-токен пользователя для работы с Яндекс.Диск REST

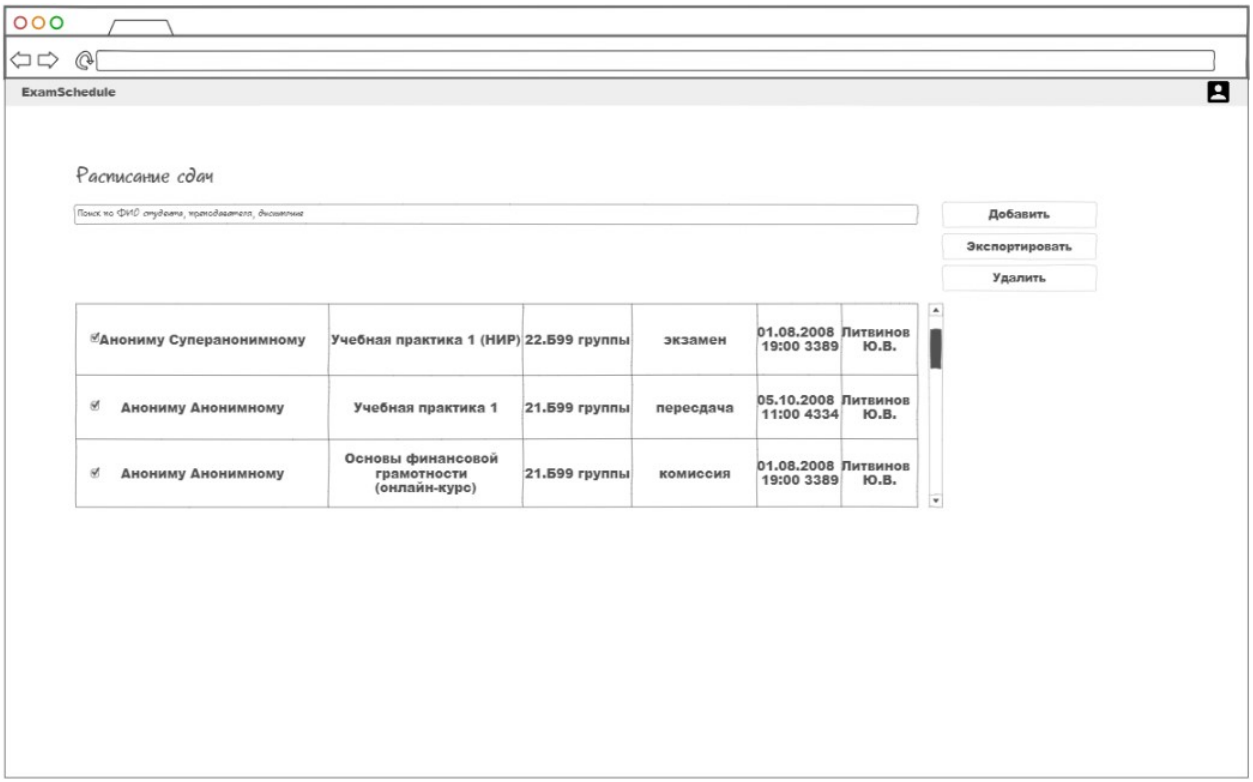

Рис. 4: Мокап страницы отображения расписания

API [12], путь до файла с таблицей в сервисе Яндекс.Диск[5] и файл формата .docx с индивидуальными графиками, который рассылается преподавателям на почту. Отправленный запрос обрабатывается на сервере, где с помощью библиотеки Open XML SDK [7] разбирается файл формата .docx. После открытия полученный документ разбивается на абзацы, которые содержат данные студента и таблицу с индивидуальным графиком для него, данные из файла извлекаются в соответствии с указанными полями и вносятся изменения в скачанную таблицу. Затем полученная таблица посредством REST API выгружается на Яндекс.Диск.

При помощи инструмента сборки Vite [11] был создан проект с использованием библиотеки React для пользовательского интерфейса. Кроме того, была создана страница отображения расписания сдач, отображающая данные, полученные посредством HTTP запроса библиотеки Axios к реализованному API. В созданном проекте были добавлены ком-

[5]Яндекс.Диск. https://360.yandex.ru/disk/ — [Дата обращения: 2023-12-08]

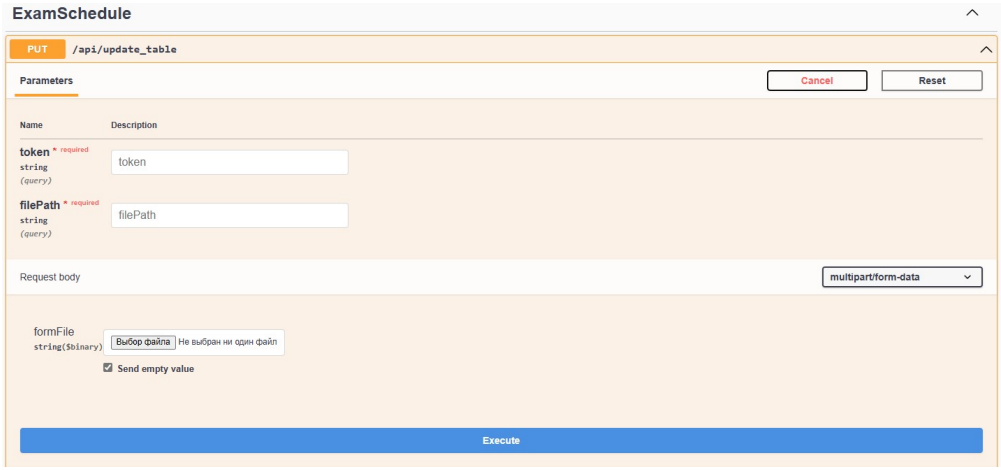

Рис. 5: Документация запроса для разбора файла и обновление таблицы сдач

понеты для дальнейшей реализации интерфейса.

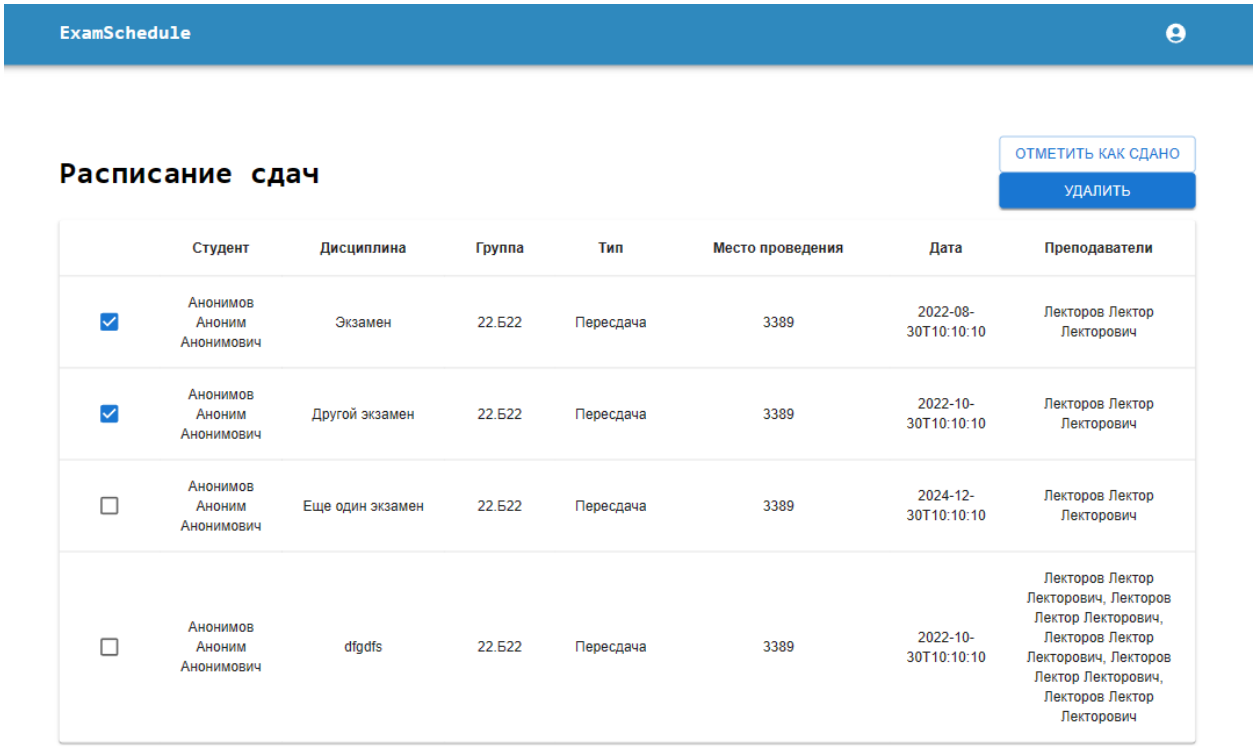

Рис. 6: Страница отображения расписания сдач

4. Эксперимент

Для обеспечения качества реализованного веб-приложения будет проведено тестирование — модульное тестирование серверной части приложения, а также сквозное тестирование пользовательского интерфейса. В качестве библиотеки для модульного тестирования будет использоваться NUnit [5], а для автоматизации сквозного тестирования была выбрана наиболее популярная библиотека Playwright [8].

Помимо этого планируется провести апробацию среди преподавателей и сотрудников учебного отдела с использованием System Usability Scale.

Будет предложено испытать реализованную систему в реальных условиях. Таким образом планируется протестировать:

- Удобство пользовательского интерфейса;
- Соответствие представленным требованиям;
- Отказоустойчивость реализованной системы.

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.
- Создан React-проект и начата разработка пользовательского интерфейса.

Код доступен в репозитории на GitHub[6].

[6] 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: 2024-01-03).

[2] Axios. — URL: https://axios-http.com/docs/intro (дата обращения: 2024-01-03).

[3] Docker. — URL: https://www.docker.com/ (дата обращения: 2024-01-03).

[4] Docker Compose. — URL: https://docs.docker.com/compose/ (дата обращения: 2024-01-03).

[5] NUnit. — URL: https://nunit.org/ (дата обращения: 2024-01-03).

[6] Ninjamock. — URL: https://ninjamock.com/ (дата обращения: 2024-01-03).

[7] Open XML SDK. — URL: https://github.com/dotnet/Open-XML-SDK (дата обращения: 2024-01-03).

[8] Playwright. — URL: https://playwright.dev/ (дата обращения: 2024-01-03).

[9] PostgreSQL. — URL: https://www.postgresql.org/ (дата обращения: 2024-01-03).

[10] React. — URL: https://reactjs.org/ (дата обращения: 2024-01-03).

[11] Vite. — URL: https://vitejs.dev/ (дата обращения: 2024-01-03).

[12] Яндекс.Диск REST API. — URL: https://yandex.ru/dev/disk/api/ (дата обращения: 2024-01-03).