**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И**

**ГОСУДАРСТВЕННОЙ СЛУЖБЫ   
при ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ - ФИЛИАЛ**

Кафедра информатики и информационных технологий

Направление подготовки (специальность): 09.03.03 Прикладная информатика

**ОТЧЕТ**

**о прохождении учебной практики (практики по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности)**

Соколова Дмитрий Александровича

2 курс обучения учебная группа № Ик-721

Место прохождения практики АНОО «Школа 800» Нижний Новгород, Южный б-р, 23

Срок прохождения практики: с «28» апреля 2025 г. по «26» мая 2025 г.

**Руководители по практической подготовке:**

От института: М.Л. Орлов Доцент

*(И.О. Фамилия) (должность)*

От профильной организации

*(при наличии)***:**  Д.А.Бердышев\_\_\_ Директор по

цифровизации

*(И.О. Фамилия) (должность)*

Отчет подготовлен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Соколов

*(подпись студента)*

**Представитель центра карьеры института-филиала**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О., должность, подпись)*

г. Нижний Новгород , 2025 г.

Содержание

[Библиографический список 33](#_Toc197879240)

Введение

В период с 28 апреля по 26 мая 2025 года, мной была пройдена учебная практика в автономной некоммерческой образовательной организации «Школа 800».

**Цель прохождения практики:** закрепление, расширение, углубление и систематизация знаний, полученных в ходе изучения дисциплин учебного плана в период теоретического обучения, получение первичных умений и навыков профессиональной деятельности.

**Задачи практики:**

формирование профессиональных знаний, умений и навыков, необходимых для успешного осуществления профессиональной деятельности с использованием современных информационно-коммуникационных технологий;

изучение нормативных правовых актов, регламентирующих деятельность объекта учебной практики;

анализ информационного обеспечения управления объектом учебной практики;

изучение технологий проектирования программных продуктов, разработанных в НИУ филиале РАНХиГС;

знакомство с основными этапами создания программного обеспечения;

разработка программного продукта, автоматизирующего процессы управления организацией, его отладка и тестирование.

**Планируемые результаты практики:**

закрепление, углубление и расширение знаний, полученных в ходе обучения;

приобретение первичных профессиональных умений и навыков;

получение первоначального практического опыта по основным видам профессиональной деятельности.

Глава 1. Характеристика организации

**1.1. Общая характеристика организации**

«Школа 800» — образовательный центр для детей в Нижнем Новгороде. Это школа полного дня, где ученики получают знания, погружаются в профильные предметы, развивают индивидуальность и учатся взаимодействовать в команде.

В организации есть несколько структурных подразделений. Практика проходила в IT-службе Школы 800.

Данное подразделение обеспечивает бесперебойную работу всех цифровых и информационных систем учреждения.

Основной вид деятельности структурного подразделения: обеспечение бесперебойной работы цифровой среды организации.

Основные задачи IT-службы Школы 800:

Аналитика данных и разработка программ;

Поддержка бесперебойной работы локальной сети, Wi-Fi по всей территории школы;

Обслуживание серверов, систем хранения данных и резервного копирования;

Обеспечение стабильной работы интерактивных досок, проекторов, умных панелей и другой техники;

Внедрение и сопровождение электронных дневников, платформ дистанционного обучения;

Разработка и поддержка образовательных приложений и программ;

Поддержка цифровых лабораторий, VR/AR-классов и симуляторов;

Настройка и мониторинг антивирусной защиты, межсетевых экранов, систем контроля доступа;

Защита персональных данных обучающихся и сотрудников;

Анализ и внедрение новых ИТ-решений.

Руководителем организации (лицом, имеющим право без доверенности действовать от имени юридического лица) с 22 июня 2020 г. является директор Сартан Марк Наумович (ИНН: 5262371883).

**1.2. Реквизиты организации**

Юридической наименование: автономная некоммерческая образовательная организация «Школа 800».

Юридический адрес: 603087, г. Нижний Новгород, ул. Нижне-Печерская, д. 4А.

***Таблица 1.***

**Сведения об организации**

|  |  |
| --- | --- |
| ИНН | 5262371883 |
| КПП | 526001001 |
| ОГРН | 1205200025073 |
| Дата образования | 22.06.2020 |

**1.3. Программное обеспечение организации**

Рабочее место офисного сотрудника оснащено персональным компьютером Ноутбук Acer Extensa 15 EX215-23-R8PN 15.6. Это ноутбук для ведения потоковых трансляций, офисной работы и просмотра контента онлайн в высоком разрешении.

Основные характеристики:

- Матовый IPS-экран разрешением 1920х1200;

- Процессор Ryzen 5 7520U частотой до 4,4 ГГц, 4 ядера и 8 потоков;

- 16 Гб оперативной памяти;

- 512 Гб внутренней памяти;

- Видеоядро AMD Radeon 610M;

- OC Windows 10;

- Встроенная HD-камера;

В программное обеспечение, которое используют сотрудники, входят такие программы как:

PostgreSQL – объектно-реляционная СУБД с открытым исходным кодом, предназначенная для хранения, обработки и управления данными, поддерживает сложные запросы и расширения.

Visual Studio – мощная среда разработки от Microsoft для создания программ на C#, C++, Python и других языках, включает инструменты отладки, тестирования и управления проектами.

PyCharm – профессиональная среда разработки для Python от JetBrains, с интеллектуальным автодополнением, встроенной отладкой и поддержкой фреймворков (Django, Flask и др.).

Power BI – приложение для создания интерактивных аналитических отчётов и панелей, позволяющее визуализировать данные из множества источников и делиться результатами онлайн.

dbdiagram.io – онлайн-инструмент для создания схем баз данных (ER-диаграмм), с возможностью экспорта SQL-кода и совместной работы.

Яндекс.Браузер – веб-браузер на базе Chromium с защитой данных, турбо-режимом и встроенным переводчиком, оптимизирован для российских сервисов.

Microsoft Office 2021 – пакет офисных приложений от Microsoft для локальной установки, включающий классические версии Word, Excel, PowerPoint, Outlook и других программ (в зависимости от редакции).

Глава 2. Основная часть

**Получение практического задания**

В период с 28 апреля по 26 мая 2025 года, мной была пройдена учебная практика в автономной некоммерческой образовательной организации «Школа 800».

Деятельностьорганизации «Школа 800» разделена на 16 отделов: Администрация школы, Учебно-методический отдел, IT-служба, Отдел цифровизации и аналитики данных, Психолого-педагогическая служба, Библиотечно-информационный центр, Отдел дополнительного образования, Проектный и инновационный отдел, Отдел воспитательной работы и школьных инициатив, HR-служба (кадровый отдел), Финансово-экономический отдел, Юридическая служба, Служба безопасности и охраны, Хозяйственно-технический отдел, Медицинский кабинет, PR-отдел и работа с общественностью.

В рамках прохождения учебной практики я был направлена в отдел «IT-служба».

Мне было поручено задание: спроектировать модель базы данных, создать её, протестировать запросы к серверу для web-приложения «Интерактивное расписание школы». Необходимо учитывать, что в дальнейшем, приложение должно иметь интуитивно понятный интерфейс, разделение ролей, динамическое обновление информации и адаптацию под информационные устройства, которыми оснащена школа.

**Разработка функциональной модели**

Проанализировав техническое задание, было принято решение разработать функциональную модель, определив основной функционал приложения и взаимодействие его частей.

Функциональная модель — это основа проектирования web-приложения, в которой описываются все ключевые функции, роли пользователей и их взаимодействие с системой. Она помогает понять, какие задачи будет выполнять приложение, какие действия доступны каждому пользователю, и как будет организована логика работы сервиса.

В ходе разработки функциональной модели были описаны основные функциональные возможности приложения.

Определение основных ролей пользователей (разделение прав доступа):

Администратор системы: Управление всеми пользователями и ролями, настройка параметров школы (периоды, звонки, классы, аудитории), управление расписанием: создание, редактирование, удаление;

Заместитель директора / Учебная часть: Составление и утверждение расписания. назначение учителей, аудиторий и предметов, просмотр загруженности кабинетов, учителей, классов;

Учитель: Просмотр своего индивидуального расписания, просмотр расписания класса (в котором он ведёт), уведомления об изменениях;

Ученик: Просмотр расписания своего класса и кружков, получение уведомлений об изменениях;

Родитель: Просмотр расписания своего ребёнка, получение уведомлений;

Гость (без авторизации): Просмотр общедоступного расписания (например, по классам).

Основные функции системы:

Управление расписанием: Создание расписания по неделям/четвертям/триместрам. Ввод расписания вручную или через шаблон. Автоматическое распределение (в перспективе).

Фильтрация и поиск: фильтрация по классу, учителю, предмету, дню недели, времени. Быстрый поиск по фамилии учителя или названию предмета.

Отображение расписания: Табличный и карточный формат (адаптация под информационные устройства в школе – информационные доски).

Управление данными: Справочники: классы, предметы, учителя, кабинеты. Импорт/экспорт данных (например, из Excel).

Аналитика: Загруженность учителей/кабинетов. Конфликты в расписании (автоматическая проверка).

Требования к UX / UI:

Поддержка адаптивного дизайна.

Фирменные цвета и шрифты школы.

Простой, понятный интерфейс с акцентом на доступ к нужной информации за 1–2 клика.

**Проектирование информационной базы**

**2.3.1 Основные сущности (таблицы)**

На этом этапе я разработал структуру данных, которая ляжет в основу базы данных приложения. Определил ключевые сущности (таблицы), их атрибуты и связи между ними. Особое внимание уделяется модели пользователей: система должна различать роли (администратор, редактор, учитель, ученик, родитель) и ограничивать доступ к функциям в зависимости от этой роли. Также предусмотрел структуру для хранения расписания, классов, предметов, кабинетов и связей между ними.

***Таблица 2.***

**Сущность Users (Пользователи)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| full\_name | ФИО |
| email | Почта |
| password\_hash | Захешированный пароль (для безопасности) |
| role | Роли |
| created\_at, last\_login | Дата и время создания/обновления |

***Таблица 3.***

**Сущность Classes (Классы)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| name | Название класса |
| year | Год |
| class\_teacher\_id | ID учителя |

***Таблица 4.***

**Сущность Subjects (Предметы)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| name | Название предмета |

***Таблица 5.***

**Сущность Teachers\_Subjects (Учителя-предметы)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| teacher\_id | ID учителя (из Users) |
| subject\_id | ID предмета (из Subjects) |

***Таблица 6.***

**Сущность Schedule (Расписание)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| class\_id | ID класса (из Classes) |
| subject\_id | ID предмета (из Subjects) |
| teacher\_id | ID учителя (из Users) |
| room | Кабинет |
| day\_of\_week | День недели |
| lesson\_number | Номер урока |

***Таблица 7.***

**Сущность Parents\_Students (Родитель-ученик)**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| id | Уникальный идентификатор |
| parent\_id | ID родителя (из Users) |
| student\_id | ID ученика (из Users) |

**2.3.2 Права доступа по ролям**

***Таблица 8.***

**Права доступа**

|  |  |  |  |
| --- | --- | --- | --- |
| **Роль** | **Просмотр расписания** | **Редактирование** | **Управление пользователями** |
| Администратор | **+** | **+** | **+** |
| Редактор | **+** | **+** | **-** |
| Учитель | **+** | **-** | **-** |
| Ученик | **+** | **-** | **-** |
| Родитель | **+** | **-** | **-** |

**2.3.3 Связи между сущностями**

Один учитель может преподавать несколько предметов;

Один ученик состоит в одном классе;

Один родитель может быть связан с несколькими учениками;

Расписание связывает учителя, класс, предмет, день и время.

**2.3.4 Безопасность**

Хранение пароля в базе данных должно быть безопасным, поэтому я буду использовать библиотеку для хеширования.

Авторизация будет использовать токены, которые будут находиться в Cookie браузера.

**2.3.5 Логическая модель базы-данных**

Для создания логической модели использовал современное web-средство dbdiagram.io. Схема приведена на рисунке 1.

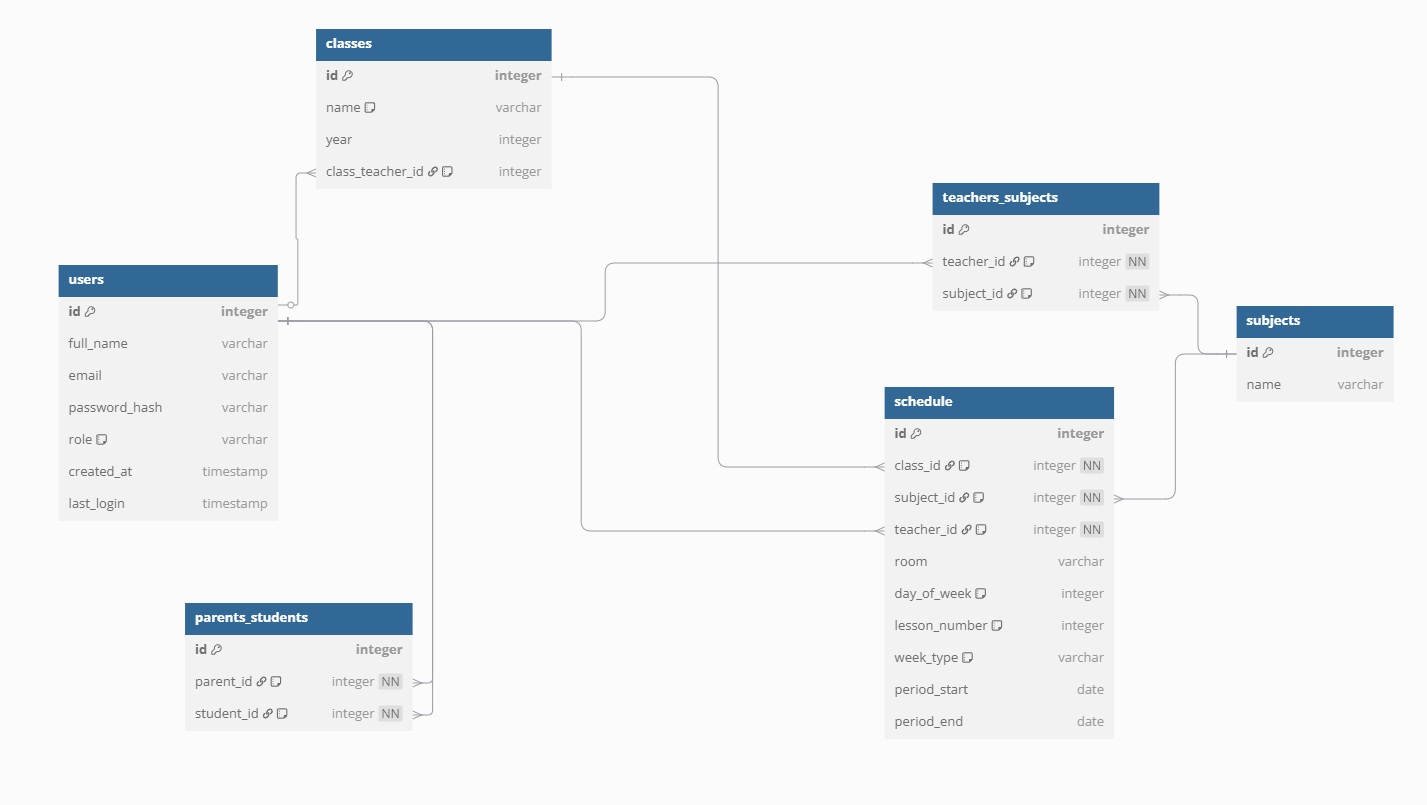


Рисунок 1 – Логическая модель базы данных

Представленная модель включает в себя 7 ключевых сущностей, отражающих структуру и логику функционирования информационной системы «Расписание школы»: Пользователи, Классы, Предметы, Учителя и предметы, Расписание, Родители и ученики, а также Роли, которые определяются прямо в сущности пользователей.

При входе в систему осуществляется идентификация пользователя по его учетным данным. Уровень доступа зависит от роли, которая хранится в поле role таблицы Users. Возможные роли: администратор, редактор, учитель, ученик, родитель. Эта структура позволяет гибко разграничивать функционал в системе. Например, редактор может управлять расписанием, а учитель и ученик — только просматривать.

Сущность Users является центральной в системе. Она содержит информацию о каждом участнике образовательного процесса: ФИО, контактные данные, роль, дату регистрации и последнего входа. Учителя, ученики и родители хранятся в этой таблице с различием только по роли, а связь между родителями и детьми реализована через отдельную сущность Parents\_Students.

Сущность Classes описывает учебные классы. Каждый класс имеет уникальное имя (например, «7.1»), учебный год и ссылку на классного руководителя — пользователя с ролью учителя. Это реализовано через внешний ключ class\_teacher\_id, ссылающийся на таблицу Users.

Сущность Subjects содержит перечень учебных дисциплин (например, математика, физика, английский язык). Информация о том, какой учитель преподаёт какой предмет, хранится в таблице Teachers\_Subjects, которая соединяет пользователей с ролью "учитель" с конкретными предметами. Это обеспечивает возможность одному учителю преподавать несколько предметов, и одному предмету быть преподаваемым несколькими учителями.

Сущность Schedule представляет собой расписание занятий. Она содержит сведения о том, в какой день недели и на какой урок (пара) проводится определённый предмет, в каком кабинете, с каким учителем и для какого класса.

Для реализации семейных связей используется сущность Parents\_Students, отражающая отношения «родитель — ученик». Эта таблица позволяет установить связь между несколькими детьми и одним родителем, а также учесть ситуацию, когда у одного ученика может быть два и более родителя.

Модель построена таким образом, чтобы обеспечить гибкость, масштабируемость и возможность расширения — например, для добавления новых ролей или внедрения системы уведомлений.

2.4 Архитектура приложения

**2.4.1 Общие сведения**

В архитектуре web-приложения «Расписание школы» frontend и backend взаимодействуют через REST API. Это разделение позволяет независимо развивать клиентскую и серверную части и масштабировать приложение при необходимости.

REST API (Representational State Transfer Application Programming Interface) — это архитектурный стиль взаимодействия между клиентом и сервером через стандартные HTTP-запросы. REST API позволяет frontend (например, React) и backend (например, Express) обмениваться данными, как правило, в формате JSON.

Почему я выбрал REST API:

Вся система строится вокруг ресурсов (например, расписание, пользователи, классы, предметы).

Каждый ресурс доступен по уникальному URL.

Разные HTTP-методы соответствуют разным действиям с ресурсами:

***Таблица 9.***

Методы REST API

|  |  |  |
| --- | --- | --- |
| **HTTP-метод** | **Назначение** | **Пример URL** |
| GET | |  | | --- | |  |  |  | | --- | | Получить данные | | /api/schedule – список расписаний |
| POST | Создать ресурс | /api/schedule – добавить запись |
| PUT | |  | | --- | |  |  |  | | --- | | Обновить ресурс | | |  | | --- | |  |  |  | | --- | | /api/schedule/12 – обновить запись с ID 12 | |
| DELETE | Удалить ресурс | /api/schedule/12 – удалить запись |

Данный подход позволяет:

Ограничивать доступ (например, только редактор может редактировать расписание),

Возвращать статусы запросов (200 OK, 201 Created, 401 Unauthorized, 404 Not Found),

Работать с фильтрами и параметрами: /api/schedule?classId=5&day=2.

Преимущества:

Прост в реализации;

Поддерживается всеми браузерами и клиентами;

Хорошо подходит для SPA (одностраничных приложений) на React;

Не требует сложных протоколов — только HTTP и JSON.

**2.4.2 Используемые технологии и их взаимодействие**

**Frontend (React)**

React — это JavaScript-библиотека, разработанная компанией Facebook, предназначенная для создания пользовательских интерфейсов. React используется для разработки одностраничных приложений (SPA), где интерфейс не перезагружается при переходе между страницами, а обновляется только нужная его часть.

В моем проекте «Расписание школы» React позволяет строить интерфейс, который динамически меняется в зависимости от данных пользователя и его роли (ученик, родитель, учитель, редактор).

React компоненты — это независимые, повторно используемые блоки интерфейса, такие как форма авторизации, таблица расписания или выпадающий список классов.

С помощью React планирую организовать маршрутизацию, фильтрацию, отправку запросов к серверу и визуализацию данных, полученных из backend-системы.

Функции React:

Отвечает за отображение интерфейса: расписания, фильтров, форм, ролей, уведомлений.

Получает и отправляет данные с помощью HTTP-запросов (обычно через fetch или axios).

Управляет маршрутизацией (например, с помощью React Router).

Обрабатывает авторизацию, рендерит интерфейс в зависимости от роли (учитель, ученик, родитель и т.д.).

**Backend (Express + Prisma)**

Express — это серверный фреймворк для платформы Node.js, предназначенный для построения веб-сервисов и API. В моем приложении Express выполняет роль "мозга" серверной части. Он принимает HTTP-запросы от клиентской части (написанной на React), обрабатывает их, применяет бизнес-логику, проверяет авторизацию, и формирует ответы в виде JSON. Express легко масштабировать и дополнять, к тому же он хорошо интегрируется с другими библиотеками, такими как middleware для работы с токенами (например, JWT), загрузкой файлов, логированием и т.п. Все маршруты вроде /api/schedule, /api/users, /api/login реализованы именно в Express.

Prisma — это современная ORM (Object-Relational Mapping) библиотека, которая обеспечивает удобную и типобезопасную работу с базой данных. Prisma позволяет разработчику использовать SQL-базу (в моем случае PostgreSQL) через JavaScript-код, не вручную составляя SQL-запросы. Вся структура базы описывается в виде модели (Prisma Schema), а затем автоматически генерируется клиент, через который удобно получать, создавать, обновлять и удалять записи. Prisma встраивается в Express и позволяет, например, за пару строк получить все расписания для конкретного класса или добавить новую запись в таблицу Schedule. Также Prisma позволяет использовать связи между таблицами и вложенные запросы, что особенно важно для сложных сущностей, вроде связки «учитель-предмет-класс-время».

PostgreSQL — это надежная, объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. Она используется в моем проекте как хранилище всей информации: пользователей, ролей, предметов, классов, связей между родителями и детьми, и расписания. PostgreSQL идеально подходит для структурированных данных, обладает высокой производительностью, поддерживает транзакции, внешние ключи, индексы и сложные запросы. В паре с Prisma она позволяет вам безопасно и эффективно управлять школьной информацией, сохраняя целостность данных даже при множественных связях.

Функции Backend:

Принимает запросы от frontend, обрабатывает их, взаимодействует с базой данных через Prisma и возвращает JSON-ответ.

Реализует авторизацию (например, по JWT), валидацию данных и бизнес-логику.

Управляет CRUD-операциями: добавление, обновление, удаление и получение расписания, пользователей, классов и т.д.

**2.4.3 Пример взаимодействия Frontend и Backend**

**Шаг 1: Аутентификация**

Пользователь заходит в систему → React отображает форму входа.

По нажатию на кнопку "Войти" → frontend отправляет POST /api/auth/login с логином и паролем.

Backend проверяет данные, генерирует JWT-токен и отправляет его обратно.

Токен сохраняется на клиенте (httpOnly cookie).

**Шаг 2: Запрос расписания**

После входа React делает GET /api/schedule?userId=123, передавая токен в заголовке Authorization: Bearer <token>.

Backend извлекает userId, проверяет роль (student) и находит расписание, связанное с его классом.

Prisma выполняет нужный SQL-запрос и возвращает расписание в JSON-формате.

React отображает его в виде таблицы/карточек по дням недели.

2.5 Практическая часть

**2.5.1 Инициализация Backend. Настройка сервера**

Для начала была создана базовая структура проекта (рис 2.).

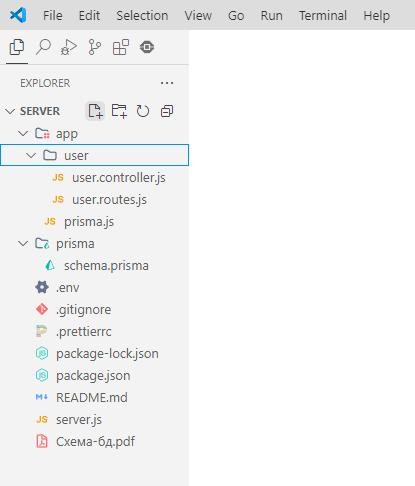


Рисунок 2 – Структура проекта backend

**Файл server.js**

Это входная точка в приложение. Здесь происходит инициализация сервера Express, подключение к базе данных (через Prisma Client), настройка middleware (например, express.json(), cors), маршрутов (app.use('/api/users', userRoutes)) и запуск сервера на указанном порту. Этот файл запускается командой npm run dev. Код файла приведен на рис.3

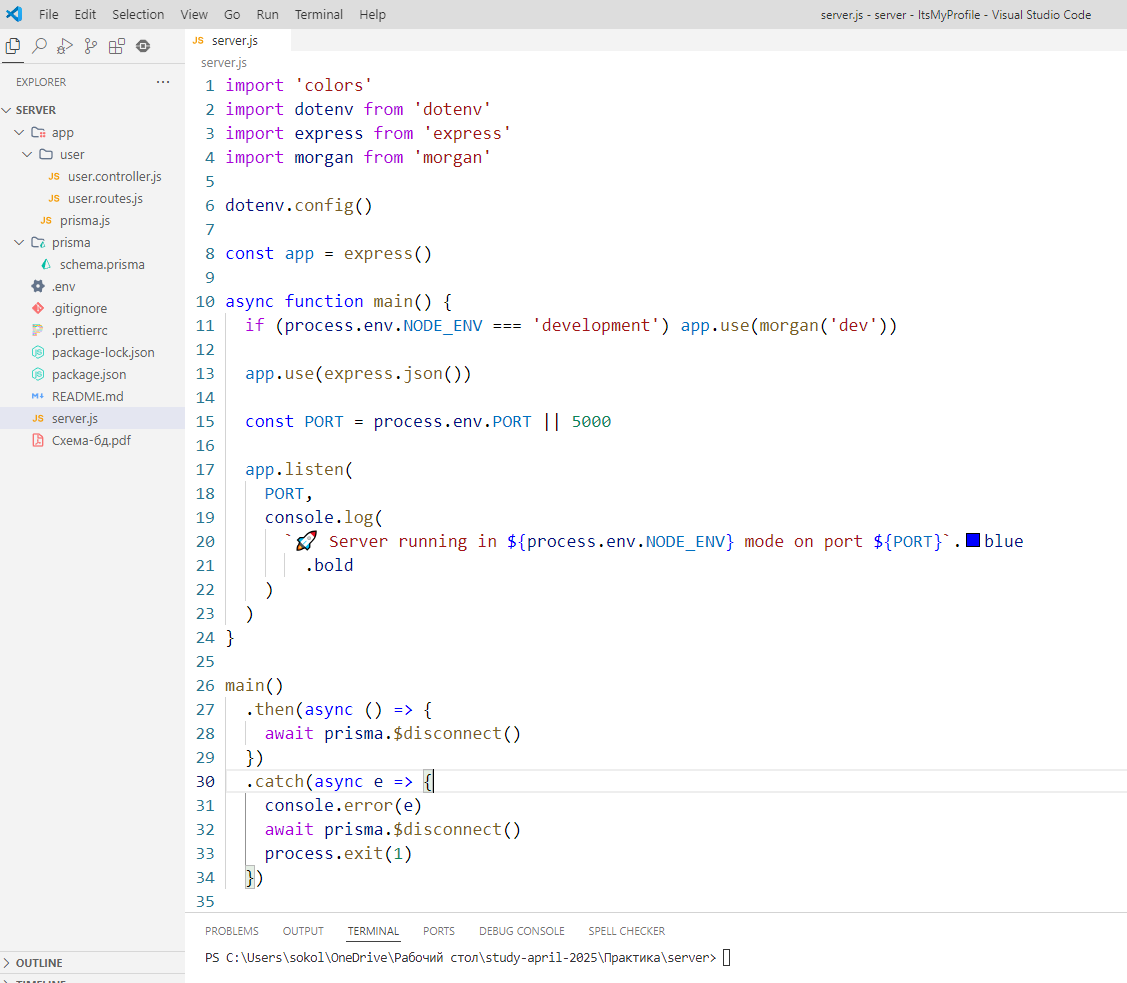


Рисунок 3 – Основной файл backend

**Остальные файлы в корне проекта**

В корне проекта находится множество служебных файлов, они необходимы для работы основных частей.

.env – содержит в себе общие переменные, которые мы можем в любой момент получить в любой части приложения. Обязательно указываем там адрес сервера и JWT токен (рис 4).

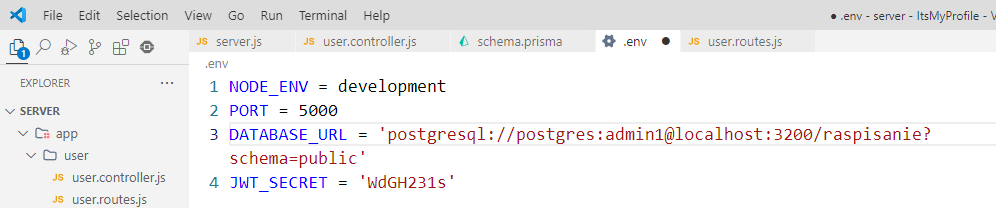


Рисунок 4 – Файл .env

Файл prettier – форматирование кода, package.json – информация о проекте, стартовые команды и список зависимостей (необходимые библиотеки), gitignore – запрещает выгрузку некоторых файлов в репозиторий в целях оптимизации.

Далее разберем файл schema.prisma – в нем описана вся структура нашей базы данных. Данный файл по команде генерирует в postgresql всю базу (рис. 5-11).

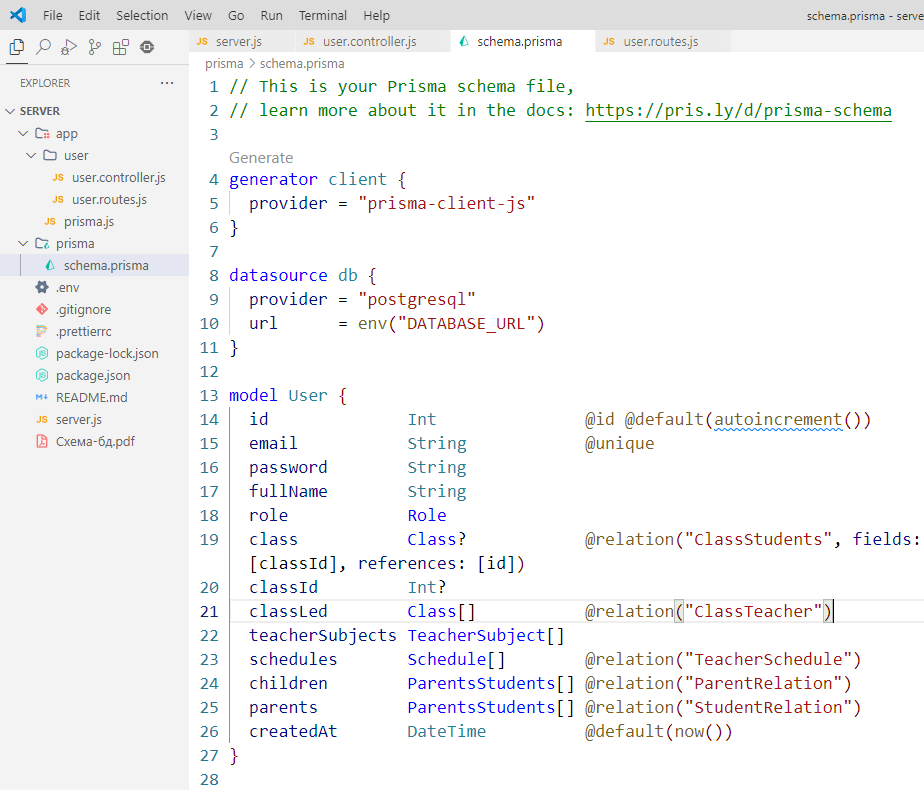


Рисунок 5 - Модель User

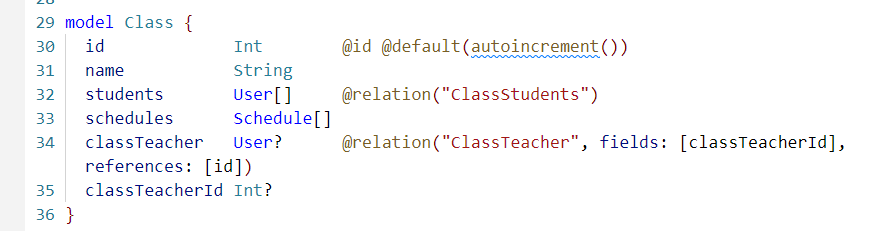


Рисунок 6 - Модель Class

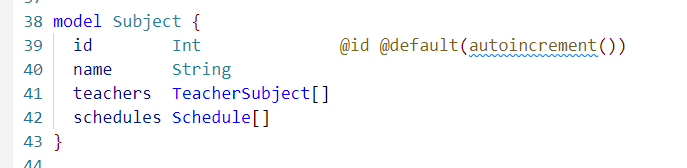


Рисунок 7 - Модель Subject

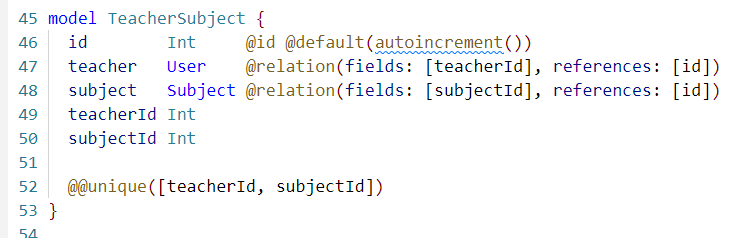


Рисунок 8 - Модель TeacherSubject

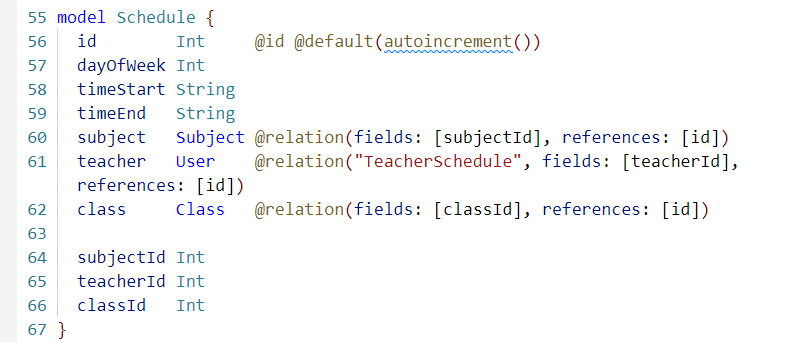


Рисунок 9 - Модель Schedule

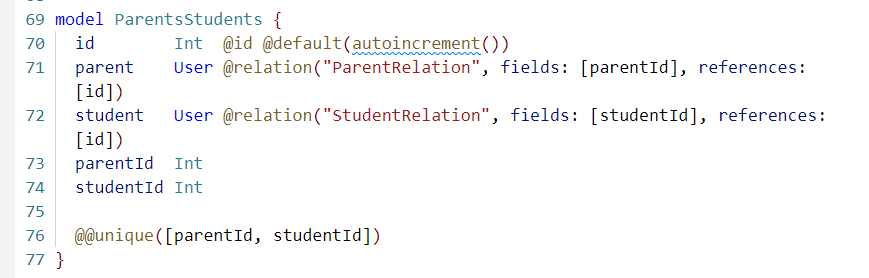


Рисунок 10 - Модель ParentsStudents

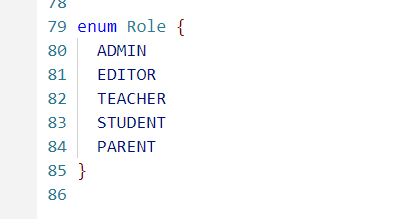


Рисунок 11 - Возможные роли в приложении

**2.5.2 Генерация Prisma Schema**

После создания схемы её можно сгенерировать - npx prisma generate, после воспользоваться npx prisma db push, что создаст все таблицы в базе, предварительно нужно создать базу raspisanie, т.к. я уже указал это название в файле .env.

Создал пустую базу raspisanie в postgres (рис.12).

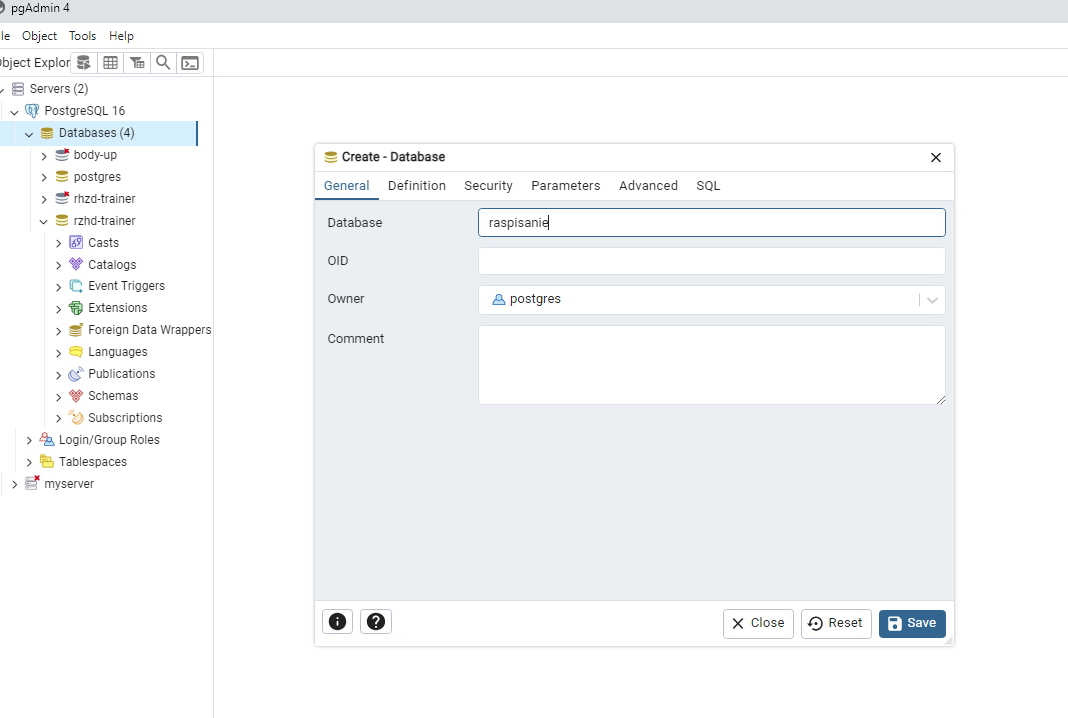


Рисунок 12 – Создание пустой базы

Сгенерировал схему prisma – обязательный шаг для данной технологии. Ввел команду npx prisma db push – схема загружена, таблицы созданы (рис. 13)

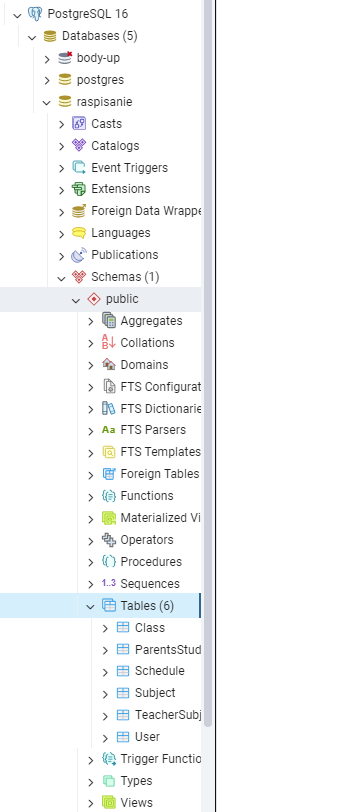


Рисунок 13 – Таблицы

**2.5.3 Создание первого контроллера и запрос к серверу**

После создания базы данных можно заносить в нее данные и делать запросы. Для этого создал в папке app первую сущность user.

Каждая сущность подразумевает 2 файла: routes – пути по которым будут идти запросы и controller – логика, которая запрашивает данные с сервера или передает на него (рис. 14).

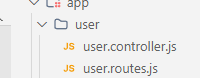


Рисунок 14 – Структура сущности

В user.routes.js прописал пути и методы get для получения и post для создания пользователя. Также указал какой код должен отработать при вызове конкретного пути (рис. 15).

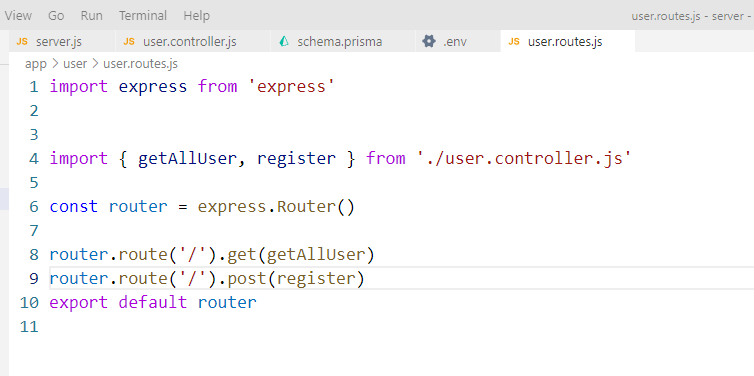


Рисунок 15 – user.routes.js

В контроллере создал два метода register – для создания пользователя и getAll – для получения всех пользователей. Важно, что при создании пользователя идет проверка на существование такого же, также производится хеширование пароля и в базе он хранится в нечитаемом виде (рис. 16).



Рисунок 16 – user.controller.js

Затем добавил пути к основному файлу, после чего можно сделать запрос на сервер с помощью insomnia – имитирует запросы на сервер (рис 17.).

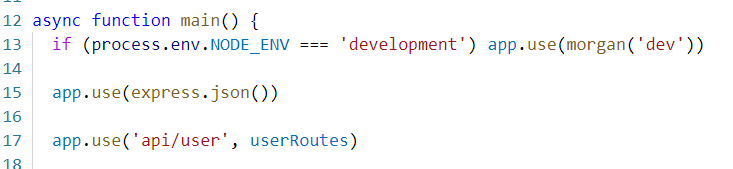


Рисунок 17 – Добавление путей на сервер

Затем я запустил сервер (рис 18.) Это означает, что можно сделать первый запрос.

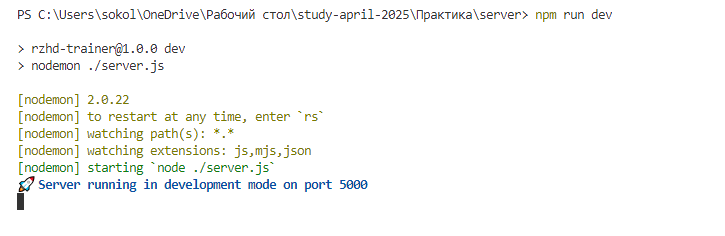


Рисунок 18 - Сервер

В приложении insomnia указал путь http://localhost:5000/api/user к get запросу и получил пустой массив, это означает, что сервер работает корректно, но данных в базе пока нет (рис. 19).

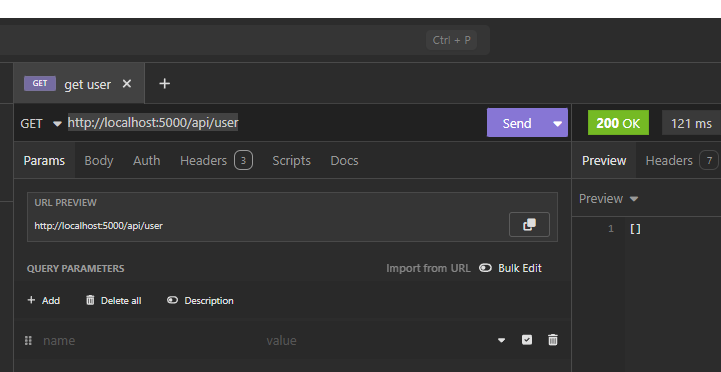


Рисунок 19 - Запрос в Insomnia

Заключение

В рамках прохождения **учебной** практики мною были применены теоретические знания и практические навыки, полученные в процессе обучения. В ходе работы была выполнена базовая настройка серверной части веб-приложения с использованиемExpress.js, Prisma ORM и PostgreSQL. Произведена настройка структуры проекта, подключение к базе данных, создание схемы данных с использованием Prisma, а также реализация базовых маршрутов и их корректное подключение к серверу.

Успешный запуск сервера и выполнение запросов подтвердили работоспособность конфигурации и позволили убедиться в корректной настройке всей серверной логики. Все задачи, поставленные в рамках **учебной** практики, были выполнены в полном объеме.

В результате я получил практический опыт в проектировании и создании серверной части современных веб-приложений, а также закрепил навыки работы с базой данных и архитектурой REST API. Этот опыт стал ценным этапом профессионального развития и значительно расширил мои технические компетенции.

# Библиографический список

Белов, И. П. Современные подходы к разработке backend-систем: Express.js и базы данных / И. П. Белов. — Новосибирск: СибАК, 2023. — 142 с.

Бражников, С. В. Практика программирования на JavaScript и Node.js / С. В. Бражников. — М.: БХВ-Петербург, 2023. — 320 с.

Власов, С. А. Информационные системы и базы данных / С. А. Власов. — СПб.: Питер, 2023. — 256 с.

Головастикова, А. А. SQL и базы данных: учебник и практикум для вузов / А. А. Головастикова. — М.: Юрайт, 2023. — 312 с. — (Высшее образование).

Дьяконов, В. Ю. PostgreSQL 15: руководство администратора и разработчика / В. Ю. Дьяконов. — М.: ДМК Пресс, 2023. — 464 с.

Ефимов, П. Р. Информационные технологии в прикладном программировании / П. Р. Ефимов. — СПб.: Питер, 2023. — 244 с.

Козлова, Е. С. Проектирование информационных систем: учебник для вузов / Е. С. Козлова. — М.: КноРус, 2023. — 289 с.

Кузнецов, А. В. Проектирование реляционных баз данных: учебное пособие / А. В. Кузнецов. — М.: Инфра-М, 2023. — 174 с.

Лакман, Э. Архитектура баз данных: руководство по проектированию / Э. Лакман. — М.: ДМК Пресс, 2023. — 368 с.

Шишкин, А. А. Современные веб-приложения на Express.js: основы и примеры / А. А. Шишкин. — Екатеринбург: Уральский рабочий, 2023. — 198 с.

|  |  |
| --- | --- |
|  |  |