МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий (программирование интернет-приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Веб-приложение «itProger»

Выполнил студент Миневич Кристина Викторовна

(Ф.И.О.)

Руководитель проекта преп.-стаж. Дубовик М.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.  (учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты преп.-стаж. Дубовик М.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер преп.-стаж. Дубовик М.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2023

Содержание

[Введение 3](#_Toc135007200)

[1 Аналитический обзор литературы и постановка задачи 4](#_Toc135007201)

[1.1 Обзор прототипов 4](#_Toc135007202)

[1.1.1 SkillBox 4](#_Toc135007203)

[1.1.2 TeachMeSkills 5](#_Toc135007204)

[1.1.3 GeekBrains 6](#_Toc135007205)

[1.1.4 Udemy 6](#_Toc135007206)

[1.2 Постановка задачи 7](#_Toc135007207)

[1.3 Описание используемых технологий 8](#_Toc135007208)

[2 Разработка архитектуры проекта 10](#_Toc135007209)

[2.1 Обобщенная структура проекта 10](#_Toc135007210)

[2.2 Диаграмма UML 11](#_Toc135007211)

[2.3 Модель базы данных 12](#_Toc135007212)

[2.4 Идентификация и авторизация 14](#_Toc135007213)

[3 Создание (реализация) программного средства 16](#_Toc135007214)

[3.1 Реализация серверной части 16](#_Toc135007215)

[Приложение 20](#_Toc135007216)

# **Введение**

Электронное обучение или e-learning развивается, практикуется и совершенствуется уже много лет. Тема актуальна как в СМИ, так и среди научного и образовательного сообществ.

Многие компании, предоставляющие свободное обучение своим сотрудникам, имеют свою внутреннюю закрытую систему, в которой предоставлена информация о тренингах, которые может предложить для них компания. Чаще всего это компании, ориентированные на разработку ПО. Также существуют сервисы, находящиеся в свободном доступе, предназначенные для предоставления доступа к тренингам всем желающим на платной или бесплатной основе, например, Udemy, PluralSight и т.д..

Сегодня огромен выбор образовательных программ. Каждый желающий может записаться на онлайн-курсы по интересам, пройти различные дистанционные программы. Подобный формат позволяет освоить понравившуюся специальность огромному количеству людей, не имеющих достаточно времени и финансов для получения высшего образования классическим способом. Также онлайн-обучение дает возможность учиться у тех, кто находится от тебя территориально далеко и в любое время. Благодаря ему можно найти курс подходящего вам уровня и подходящего автора – выбор огромен.

Изучение программирования становится все более доступным благодаря непрерывному росту количества онлайн ресурсов, которые помогают в этом всем желающим. Плюс таких ресурсов — в неограниченном количестве знаний, которые они могут дать и в высокой квалификации преподавателей. Минус — мотивация должна возникнуть у вас сама и качество полученных знаний зависит лишь от того, сколько усилий вы приложили.

При этом важно понимать, что любые формы обучения имеют свои плюсы и минусы, и онлайн-курсы — не исключение. А значит, тема совершенствования электронного обучения будет оставаться актуальной и получать свое развитие как с теоретической, так и с технологической точек зрения. Именно в этом, я считаю, и заключается актуальность выбранной темы — разработка веб-приложения для просмотра доступных тренингов и обучения программированию itProger.

# **Аналитический обзор литературы и постановка задачи**

В наше время большой популярностью пользуются сервисы, упрощающие получение информации о той или иной предметной области. Множество пользователей предпочитают вместо посещения самих компаний, предоставляющие обучение, посещать веб-сервисы, где удобно расположена информация. Популярность подобных сервисов достигается тем, что их использование очень удобно и не требует особых усилий, а также в связи с тем, что получить доступ к информации можно из любой точки мира.

Основные достоинства интернет-приложений, которые делают их выгодными и все более популярными:

* экономия времени;
* удобность использования;
* доступность.

На сегодняшний день классические тренинг-центры становятся все менее популярными. На смену им приходят онлайн-сервисы.

## **1.1 Обзор прототипов**

Рассмотрим примеры web-приложений, на которых можно получить информацию о тренингах.

## **1.1.1 SkillBox**

Skillbox – образовательная платформа, в котором обучают онлайн 90+ программам для получения востребованных IT профессий во всём мире.

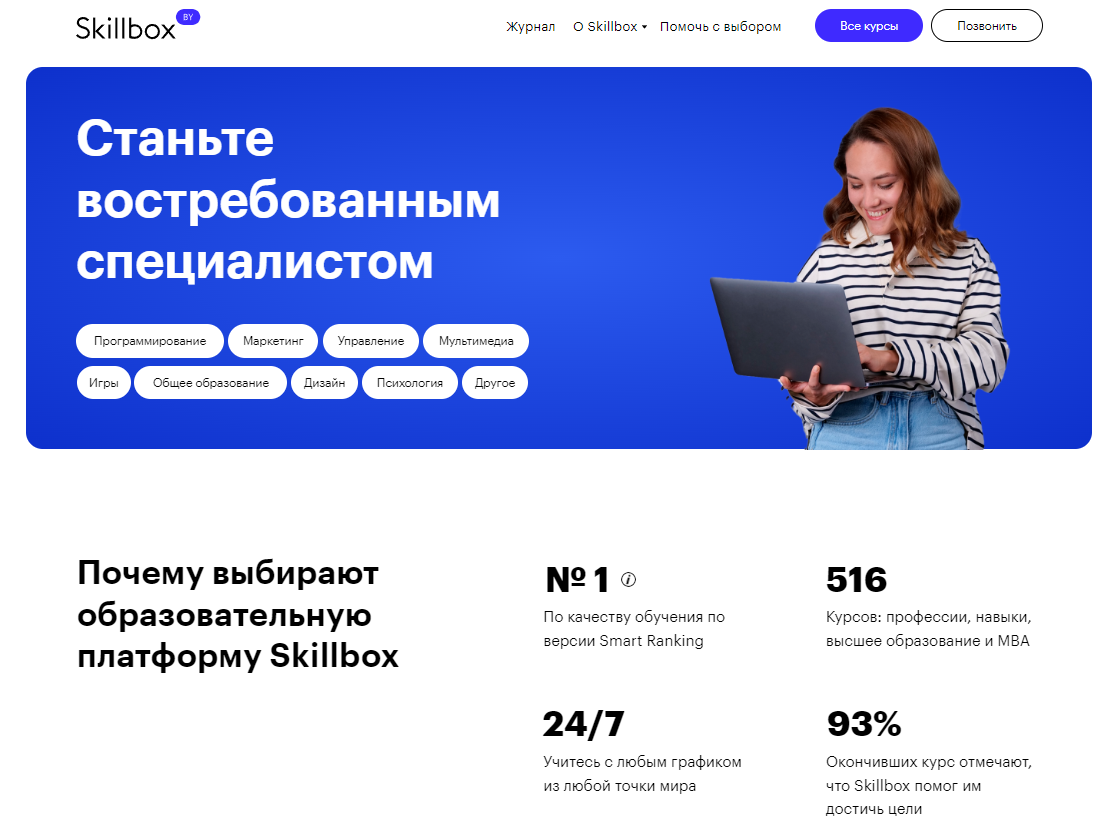


Рисунок 1.1.1 – Образовательная платформа Skillbox

Здесь обучают профессиям, которые будут всегда востребованы в мире IT. В Skillbox разноформатное обучение. Самым основным являются видеолекции. В процессе обучения на платформе пользователи закрепляют знания с помощью тестов и практических заданий, которые проверяет куратор.

Тут обучают больше 20+ профессиям, связанных с программированием и возможностью трудоустроиться по этому направлению. Вам на выбор предлагается стать разработчиками PHP, Android, Java, веб, Python, Frontend, C#, Fullstack, iOS, 1С, а также геймдизайнерами, тестировщиками или специалистами по анализу данных и кибербезопасности.

Сайт имеет интуитивно-понятный интерфейс, приятный дизайн и обладает высокой функциональностью.

## **1.1.2 TeachMeSkills**

TeachMeSkills –это школа программирования, где учат востребованным сегодня знаниям. Все программы составлены Senior- и Lead-разработчиками ведущих IT компаний специально для новичков в IT*.* Главная страница сайта представлена на рисунке 1.1.2.

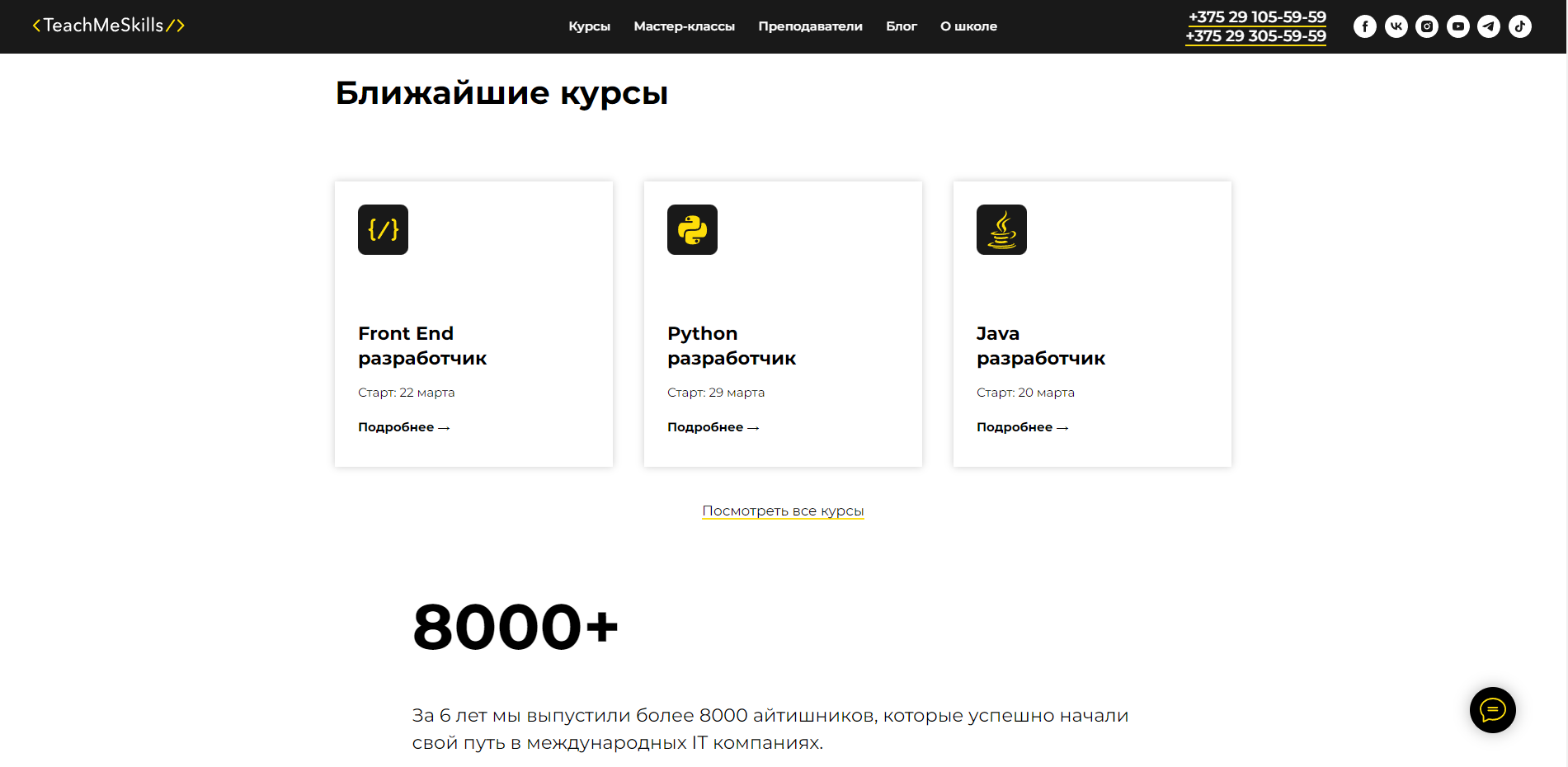


Рисунок 1.1.2 – Главная страница TeachMeSkills

Школа предлагает как онлайн-обучение, так и оффлайн, в Минске. Имеется возможность ознакомиться с преподавательским составом, а также почитать интересные статьи о сфере IT.

Сайт обладает динамическими стилями и информативным слайдером на главной странице.

## **1.1.3 GeekBrains**

В этой онлайн-школе можно найти практически всё, что касается прямо или косвенно IT-сферы. Тут обучают не только программированию или дизайну, но и интернет-маркетингу или системному администрированию. Какой курс GeekBrains не открой везде есть расписанная программа со всеми подробностями и отзывы от пользователей, проходивших тот или иной курс. Для студентов предусмотрена доступная программа стажировок и получение сертификатов о прохождении курсов. Главная страница сайта представлена на рисунке 1.1.3.

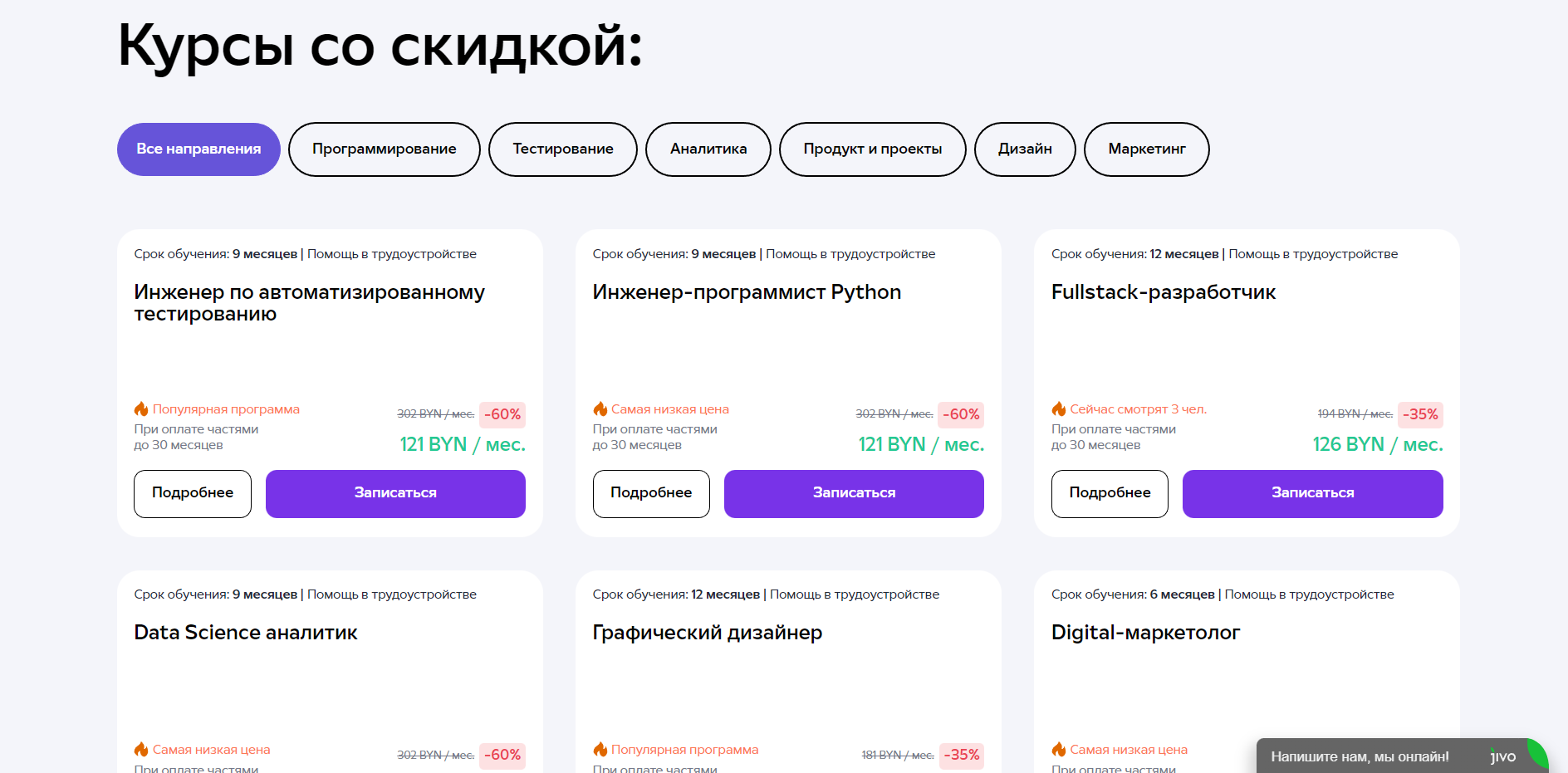


Рисунок 1.1.3 – Главная страница GeekBrains

Преимущество этого образовательного портала в доступе к множеству бесплатного контента, но самый козырь – это возможность обучения у топовых айтишников. Осилить современные профессии в мире диджитал можно на факультетах веб-, iOS-, Go-, Java- и Python-разработки, искусственного интеллекта, DevOps и других.

Если Вы новичок и не можете понять, что же нравится и подходит именно вам, команда GeekBrains готова безвозмездно помочь и определиться с будущим выбором.

Онлайн-уроки, разбор заданий вместе с преподавателями и возможность общение со своими одногруппниками – делает обучение похожее на офлайновую реальность. И только тогда, когда ученики подтверждают свои умения путем тестирования они получают сертификат.

## **1.1.4 Udemy**

Ещё одна платформа для изучения программирования – Udemy.com. Здесь доступно более двухсот тысяч онлайн-видеокурсов, а новые появляются каждый месяц! Имеется доступ к личному кабинету студента, подборка курсов на четырнадцати языках и возможность просматривать информацию об преподавателях, создавших курс.

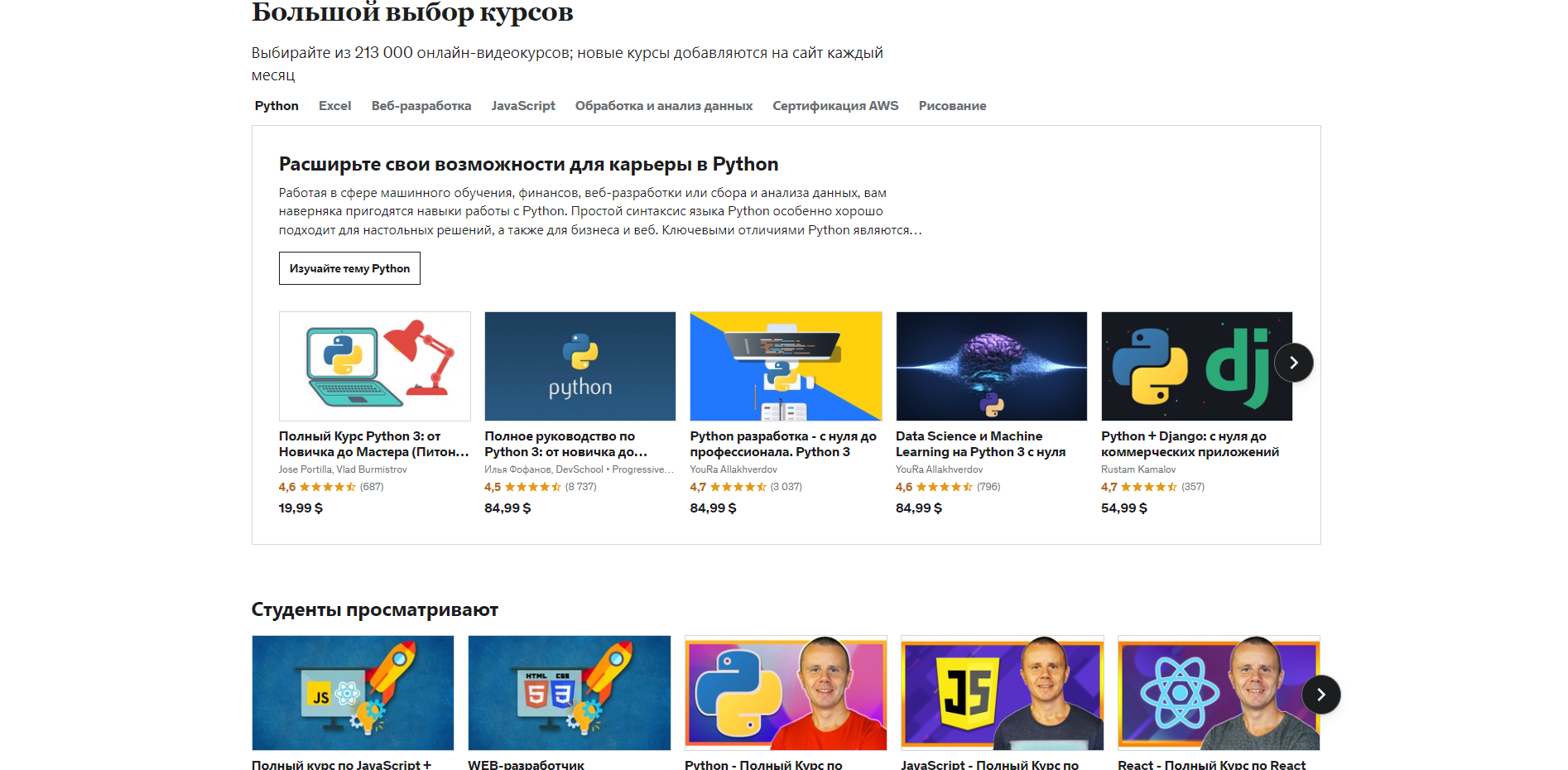


Рисунок 1.1.4 – Платформа Udemy.com

Ещё одним преимуществом данной платформы является то, что студент может проходить курс в удобном для него графике: временных ограничений нет.

Для того, чтобы понять, подходит для пользователя курс или нет, имеется возможность ознакомиться с описанием курса, а также в материалах курса можно воспользоваться предпросмотром некоторых лекций. Студенты, записанные на курс, имеют возможность оставлять отзывы, что тоже является преимуществом выбрать именно данную школу для изучения программирования.

## **1.2 Постановка задачи**

Анализ существующих сайтов и приложений позволил определиться с постановкой задачи.

Итогом разработки должно стать веб-приложение для просмотра информации о курсах, предлагаемых тренинг-центром IT компании и изменения (администрирования) данной информации.

Разумеется, необходимо разработать несколько интерфейсов: для пользователя и для администратора.

Интерфейс пользователя упрощен. Он должен давать возможность посмотреть каталог курсов и лекций по ним, а также изменить личную информацию о самом пользователе.

Интерфейс администратора должен предусматривать всевозможные операции с содержимым сайта, описаниями. Сайт должен быть выполнен в спокойных ненавязчивых тонах.

Таким образом, в данном курсовом проекте требовалось реализовать следующие задачи:

– Сохранять пользовательскую информацию в базе данных;

– Получать информацию из базы данных;

– Создание пользовательского интерфейса для взаимодействия с базой данных;

– Регистрация и авторизация пользователей;

– Удаление при необходимости информации из базы данных;

– Просмотр уроков / лекций;

– Добавление материалов в урок;

– Добавление уроков.

## **1.3 Описание используемых технологий**

При проектировании программного средства в качестве программной платформы для сервера был выбран Node.js. Node.js – программная платформа для разработки серверных web-приложений на основе движка JavaScript Chrome V8. Является средой исполнения приложений на JavaScript. Также ориентирована на события, поддерживает асинхронность и является однопоточной. Также был использован пакет Express.js, который является популярным веб-фреймворком, написанным на JavaScript и работающий внутри среды исполнения node.js. Express не мешает общей производительности приложения т.к. представляет собой тонкий слой основных функций веб-приложений. Также в нём удобно настраивать маршруты приложения, поскольку он использует уже давно известные методы http.

Рассмотрим основные особенности Node.js.

– Скорость. Одной из основных привлекательных особенностей Node.js является скорость. JavaScript-код, выполняемый в среде Node.js, может быть в два раза быстрее, чем код, написанный на компилируемых языках, вроде C или Java, и на порядки быстрее интерпретируемых языков наподобие Python или Ruby.

– Простота. Платформа Node.js проста в освоении и использовании.

– Движок V8. В основе Node.js, помимо других решений, лежит опенсорсный JavaScript-движок V8 от Google, применяемый в браузере Google Chrome и в других браузерах.

– Асинхронность. JavaScript значительно упрощает написание асинхронного и неблокирующего кода с использованием единственного потока, функций обратного вызова и подхода к разработке, основанной на событиях.

Для проектирования базы данных используется система управления базами данных Prisma. Prisma - это объектно-реляционная система управления базами данных (ORM) с открытым исходным кодом, предназначенная для разработки веб-приложений. Она позволяет разработчикам работать с базами данных с помощью удобного и интуитивно понятного API на языке TypeScript или JavaScript. Prisma предоставляет множество преимуществ, которые делают его удобным инструментом для работы с базами данных:

1. Простота использования: Prisma обеспечивает интуитивно понятный и удобный интерфейс для работы с базами данных, что делает его простым в использовании даже для новичков.
2. Мощная система миграций: Система миграций Prisma позволяет легко изменять структуру баз данных, что делает ее удобной для поддержки и масштабирования приложений.
3. Высокая производительность: Prisma использует современные технологии и архитектуру, чтобы обеспечить высокую производительность работы с базами данных.
4. Гибкость: Prisma поддерживает работу с различными базами данных, что делает его удобным инструментом для разработки приложений с различными требованиями.
5. Автоматически генерируемый код: Prisma генерирует код на основе схемы базы данных, что позволяет сократить время и усилия, затрачиваемые на написание кода.

Для написания клиентской части использовалась библиотека JavaScript – ReactJS. React — это библиотека для создания пользовательских интерфейсов. Он поощряет создание многократно используемых компонентов пользовательского интерфейса, представляющие данные, которые со временем изменяются. Многие люди используют React как V в MVC. React абстрагирован от DOM, предлагая более простую модель программирования и лучшую производительность. React реализует односторонний реактивный поток данных, который уменьшает объем шаблонов и упрощает понимание кода, по сравнению с традиционной привязкой данных.

Преимущества React.

– Использует виртуальный DOM, который является объектом JavaScript. Это улучшает производительность приложений, поскольку виртуальный JavaScript DOM быстрее, чем обычный DOM.

– Может использоваться как на стороне клиента, так и на стороне сервера, а также на других платформах.

– Компоненты и модели данных улучшают читаемость, что помогает поддерживать более крупные приложения.

# **2 Разработка архитектуры проекта**

Разработка архитектуры проекта – важная задача в процессе работы над приложением, потому что в зависимости от неё определяется уровень связности между компонентами приложения, и насколько легко можно будет это приложение расширить.

Основные задачи разработки архитектуры проекта:

* Выделение программных подсистем и отображение на них внешних функций (заданных по внешнем описании) программного средства;
* Определение способов взаимодействия между выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

## **2.1 Обобщенная структура проекта**

Курсовой проект построен на клиент-серверной архитектуре. Клиент-серверная архитектура – это архитектура, которая подразумевает две компоненты: клиент и сервер. Клиент является инициатором соединения. В качестве сервера будет выступать приложение на Node.js. В качестве клиента будет выступать приложение c асинхронным UI (React). Схема развёртывания представлена на рисунке 2.1.

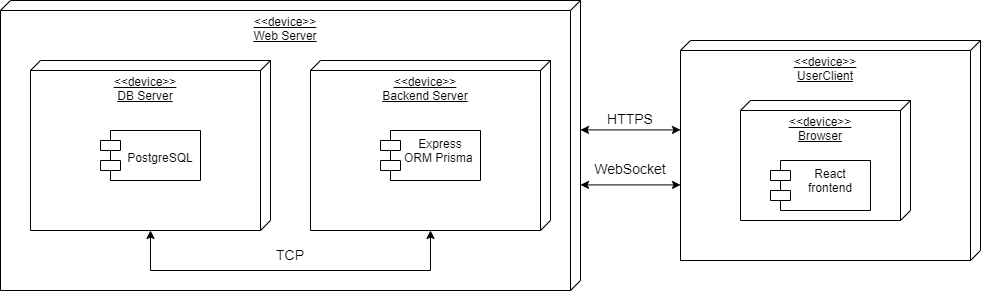


Рисунок 2.1 – Схема развёртывания

Выбрана клиент-серверная архитектура, так как она хорошо подходит для разработки web-приложений, и нагрузка будет распределена между клиентом и сервером. Также отсутствует дублирование кода сервера на клиенте. Все данные хранятся на сервере, который обычно защищен лучше большинства клиентов. Еще, так как все вычисления выполняются на сервере, снижаются системные требования к клиенту.

Таким образом, приложение должно быть выполнено в клиент-серверной архитектуре. Сервер принимает запрос клиента, обрабатывает запрос, работает с базой данных и возвращает клиенту ответ. Клиент будет являться инициатором соединения с сервером.

## **2.2 Диаграмма UML**

ДиаграммаUML–это графическое представление набора элементов,изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграмма использования представлена на рисунке 2.2.

В языке UML вариант использования изображается в виде овала, помеченного именем представляемого варианта. Варианты использования могут быть связаны с участвующими в них действующими лицами (actors), изображаемыми в виде человечков и представляющими различные роли пользователей системы или внешние системы, взаимодействующие с ней.

Варианты использования могут быть связаны друг с другом тремя видами связей: обобщением (generalization), расширением (extend relationship) и включением (include relationship).

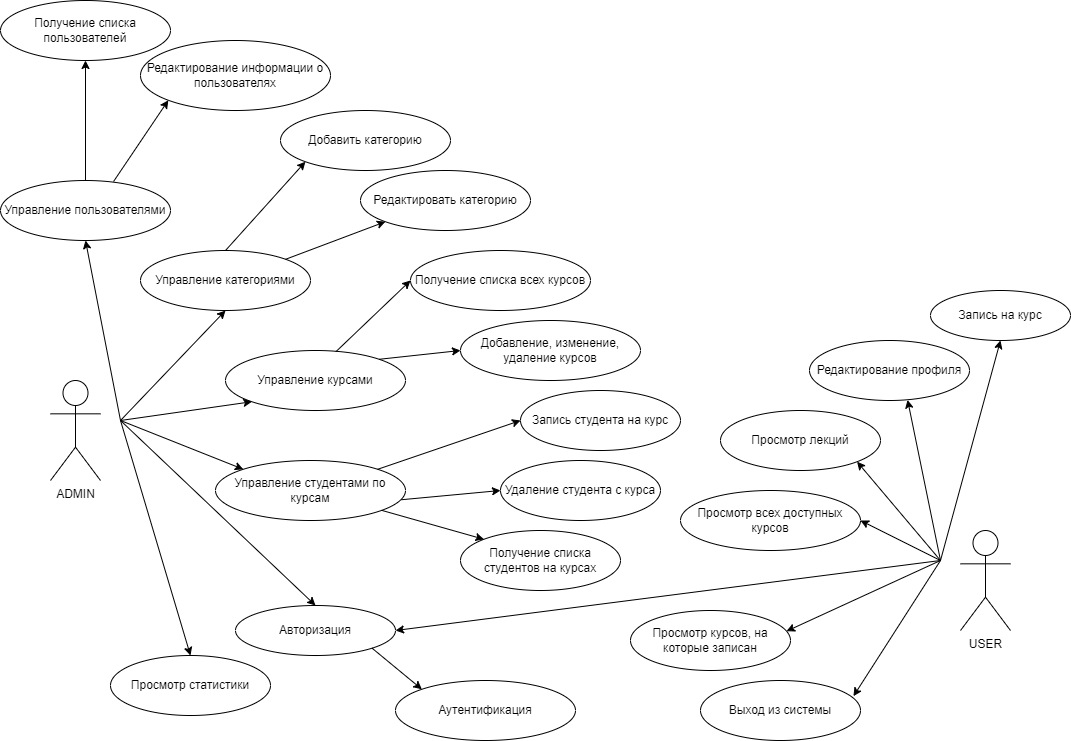


Рисунок 2.2 – Диаграмма вариантов использования

На данной диаграмме представлены 2 варианта юзеров, которые имеют различные возможности.

В первом варианте (ADMIN) есть возможности авторизации, просмотра статистики по записям на курсы и количеству курсов по каждой категории, управления пользователями, категориями, курсами и записям студентов на курсы.

Во втором же варианте (USER) можно авторизоваться, просмотреть все доступные курсы, созданные администратором, записаться на курсы, просмотреть лекции по курсам, создать или обновить профиль пользователя, а также просмотреть курсы, на которые записан текущий пользователь.

## **2.3 Модель базы данных**

Для реализации поставленной в курсовом проектировании задачи была создана база данных itProger. Для её создания использовалась система управления реляционными базами данных PostgreSQL.

Выбор данного СУБД произошел по нескольким очевидным причинам: производительность и масштабируемость (он может обрабатывать сложные запросы эффективно и обеспечивать стабильную работу даже при высоких нагрузках), расширяемость (множество встроенных и сторонних расширений, которые позволяют расширять функциональность базы данных), безопасность (обладает широким набором механизмов безопасности, включая аутентификацию, авторизацию и шифрование данных), надёжность (обеспечивает механизмы сохранности данных, включая транзакции, резервное копирование и восстановление, что позволяет избежать потери данных и обеспечить их целостность).

Все эти преимущества делают PostgreSQL привлекательным выбором для разработки и управления базами данных, включая курсовые проекты. Он предлагает гибкость, производительность и надежность, необходимые для успешной работы с данными и решения сложных задач.

Для базы данных itProger было разработано 6 таблиц:

* User – для хранения информации о пользователе;
* Profile – данные для профиля пользователя;
* Category – названия категорий для курсов;
* Course – информация о курсе;
* Lecture – данные из лекции;
* Enrollment – информация о студентах, записанных на определенные курсы.

Взаимосвязь всех таблиц проектируемой базы данных представлена на рисунке 2.3.

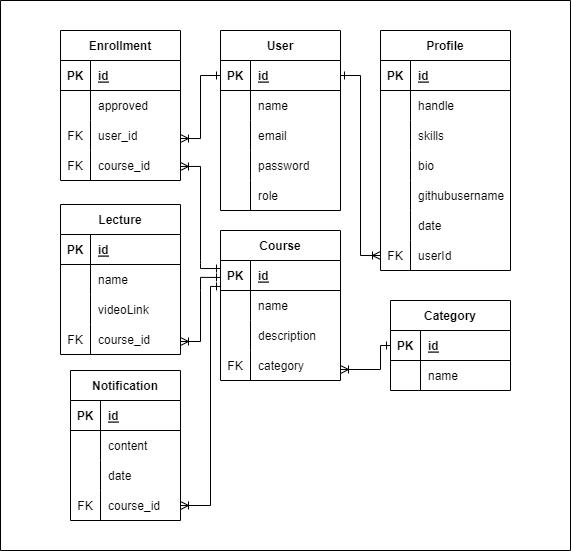


Рисунок 2.3 – Взаимосвязь таблиц базы данных

Листинг создания базы данных, используя ORM Prisma, представлен в приложении. Рассмотрим каждую таблицу по отдельности.

Таблица **User** представляет собой данные о пользователе, состоит из столбцов:

– id – идентификатор пользователя, первичный ключ;

– name – имя пользователя;

– email – адрес электронной почты пользователя;

– password – пароль пользователя;

– role – роль (USER или ADMIN).

Таблица **Category** представляет собой названия категорий курсов, состоит из столбцов:

– id – идентификатор категории, первичный ключ;

– name – название категории.

Таблица **Course** представляет собой данные о курсе, состоит из столбцов:

– id – идентификатор курса, первичный ключ;

– name – название курса;

– description – описание курса;

– category – категория, к которой относится данный курс, внешний ключ.

В таблице **Lecture** хранятся данные о лекции, она состоит из столбцов:

– id – идентификатор лекции, первичный ключ;

– name – название лекции;

– course\_id – идентификатор курса, к которому относится данная лекция, внешний ключ;

– videoLink – ссылка на видео данной лекции.

Таблица **Profile** представляет собой профиль пользователя, состоит из столбцов:

– id – идентификатор профиля пользователя, первичный ключ;

– userId – идентификатор пользователя, заполняющего профиль, внешний ключ;

– handle – заголовок профиля;

– skills – список навыков пользователя;

– bio – описание пользователя;

– githubusername – никнейм пользователя на GitHub;

– date – время создания и заполнения профиля.

Таблица **Enrollment** предназначена для хранения записей пользователей на курсы, состоит из столбцов:

– id – идентификатор записи, первичный ключ;

– user\_id – идентификатор пользователя, внешний ключ;

– course\_id – идентификатор курса, внешний ключ;

– approved – идентификатор записи, показывающий записан ли пользователь на курс или нет;

Таблица **Notification** предназначена для хранения уведомлений администратора для пользователя, состоит из столбцов:

– id – идентификатор уведомления, первичный ключ;

– courseId – идентификатор курса, внешний ключ;

– content – текст уведомления;

– date – дата создания уведомления;

Таким образом были созданы все необходимые для работы приложения таблицы в базе данных.

## **2.4 Идентификация и авторизация**

Для выполнения задач при разработке приложения, существует необходимость регистрации, авторизации и идентификации, для этой задачи используется JWT.

JSON Web Token (JWT) — это открытый стандарт для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

Состоит из трех частей: заголовка, полезной нагрузки и зашифрованных данных, которые и являются уникальными благодаря секретным ключам.

Аутентификация – процедура проверки подлинности идентификации пользователя. В данном случае происходит проверка путем сравнения введенного пароля с паролем, который сохранен в базе данных и соответствует логину. Функция принимает логин и пароль, который вводит пользователь, дальше идет поиск пользователя с введенным логином в базе данных и проверка пароля с помощью библиотеки BCrypt. В случае успеха происходит формирование JSON Web Token (JWT), который в дальнейшем будет использоваться для доступа к приложению, и отправка его пользователю.

На стороне клиента, при получении токена, он сохраняется в localStorage и устанавливается в заголовок Authorization. После чего мы можем декодировать токен и получить оттуда информацию о пользователе. Методы, выполняющие эти функции, представлены в приложении Б.

# **3 Создание (реализация) программного средства**

Решение представлено двумя проектами: backend (серверная часть на Node.js) и frontend (клиентская часть – ReactJS). Для старта сначала запускается серверная часть, затем – клиентская.

## **3.1 Реализация серверной части**

При написании сервера на NodeJS используется фреймворк Express.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Express не мешает общей производительности приложения, т.к. представляет собой тонкий слой основных функций веб-приложений. Также в нём удобно реализована маршрутизация. Имея в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, создать надежный API можно быстро и легко.

Проект разделен на несколько основных частей, которые расположены по разным директориям. На рисунке 3.1 представлена структура проекта.

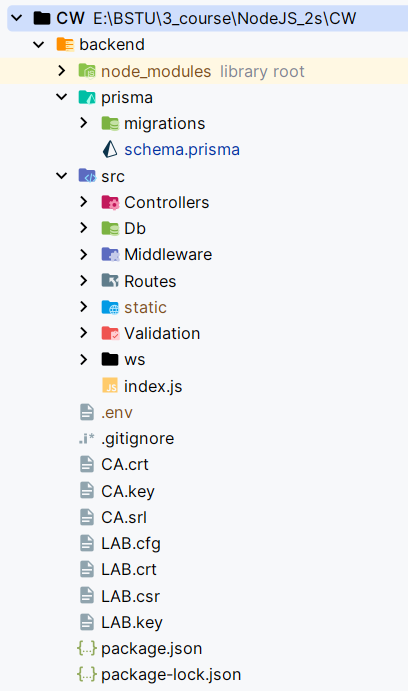


Рисунок 3.1 – Структура проекта сервера

В директории «Сontrollers» расположены контроллеры. Контроллеры обеспечивают взаимодействие пользователя и системы, обрабатывая запросы. Они используют модель для реализации необходимого действия. Содержимое директория представлена на рисунке 3.2.

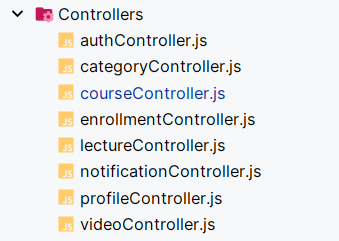


Рисунок 3.2 – Структура директории «Сontrollers»

Например, контроллер authController реализует вход, выход и регистрацию в приложении, проверяют данные для регистрации при вводе в форму, проверяется адрес электронной почты, является ли свободным, так как наличие двух пользователей с одинаковым email в приложении невозможна.

Пример одного из методов контроллера, а именно функция логина, приведена в листинге 3.1.

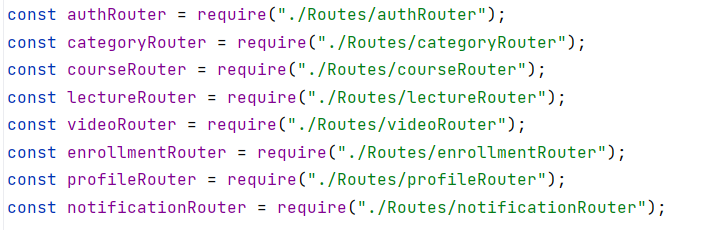
async login(req, res) {  
 const { errors, isValid } = validateLoginInput(req.body);  
  
 // Check Validation  
 if (!isValid) {  
 return res.status(400).json(errors);  
 }  
  
 const { email, password } = req.body;  
 try {  
 const user = await DbClient.user.findUnique({  
 where: {  
 email: email,  
 },  
 });  
 if (!user) {  
 const errors = { email: `User with email ${email} does not exist` };  
 return res.status(400).json(errors);  
 }  
 const validPassword = await bcrypt.compareSync(password, user.password);  
 if (!validPassword) {  
 const errors = { password: "Invalid password" };  
 return res.status(400).json(errors);  
 }  
 const token = generateAccessToken(user.id, user.role);  
 return res.json({ token });  
 } catch (err) {  
 console.log(err);  
 console.log(err)  
 res.status(400).json({ message: "Login error" });  
 }  
}

Листинг 3.1 – Реализация функции login

Директория «Middleware» содержит промежуточные обработчики которые нужны для проверки JWT токена и доступности действия исходя из роли запрашивающего.

Директория «prisma» в файле schema.prisma содержит модели, необходимые для генерации таблиц в базе данных. Приложение использует ORM-библиотеку «Prisma» для взаимодействия с базой данных. Конфигурация подключения импортируется из файла .env. Благодаря правильно прописанной модели, изменить базу данных не сложно, достаточно сделать миграцию.

Директория «Routes» содержит роутеры, которые реализуют маршрутизацию. Ниже на Листинге 3.2 представлено добавление роутеров с учётом их расположения в директориях проекта. Также мы настраиваем приложение на использование роутеров по определённому пути.



Листинг 3.2 – Регистрация роутеров

Директория «static» содержит статические файлы: видео, использующиеся на стороне клиента, которые необходимы для создания лекций по курсу.

В директории «Validation» разработаны различные промежуточные проверки на корректный ввод данных. Например, чтобы поля были непустыми, проверка на количество символов в пароле и т.д.. Содержимое директория представлена на рисунке 3.3.



Рисунок 3.3 – Структура директории «Validations»

index.js – основной файл приложения, он же и запускается.

На рисунке 3.1.2 показана структура файлов и папок в корне проекта client.

…

Рисунок 3.1.2 – Структура корня проекта frontend

Описание приведенной структуры представлено в таблице 3.1.2.

Таблица 3.1.2 – Описание файлов и папок в корне проекта frontend.

…

Структура директории src того же проекта представлена на рисунке 3.1.3.

…

Рисунок 3.1.3 – Структура директории src проекта frontend

Описание приведенной структуры представлено в таблице 3.1.3.

Таблица 3.1.3 – Описание файлов и папок в корне проекта frontend

…

Следует отметить, что весь пользовательский интерфейс строится из компонентов. React разработан вокруг концепции многоразовых компонентов. Мы определяем небольшие компоненты, и объединяем их, чтобы сформировать более крупные компоненты. Все компоненты, маленькие или большие, могут использоваться повторно, даже в разных проектах.

Каждый компонент — это JavaScript-функция, которая возвращает кусок кода, представляющего фрагмент страницы. Для формирования страницы мы вызываем эти функции в определённом порядке, собираем вместе результаты вызовов и показываем их пользователю.

Таким образом, сформированные таблицы помогают понять общую структура проектов проектируемого программного средства.

# **Приложение**