Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ МОРСКОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(СПБГМТУ)

Сообщение-доклад на тему выпускной квалификационной работы «»

Руководитель ВКР: Кудашов Р.Д.

Бакалавр: Иванов И.И.

Санкт-Петербург

2024 год.

ЗАДАНИЕ НА ВЫПУСКНУЮ РАБОТУ

Содержание

[Введение 3](#_Toc165904490)

[1 Постановка задачи 4](#_Toc165904491)

[2 Текущее состояние ВКР 5](#_Toc165904492)

[2.1 План написания ВКР 5](#_Toc165904493)

[2.2 Технологии, используемые для выполнения целей ВКР 5](#_Toc165904494)

[2.3 Ознакомление с ГОСТ 5](#_Toc165904495)

[2.4 Поиск и исследование информации 5](#_Toc165904496)

[2.5 Изучение основ работы с системами контроля версия 6](#_Toc165904497)

[2.6 Изучение инструментов 6](#_Toc165904498)

[3 Необходимые дальнейшие шаги 7](#_Toc165904499)

[3.1 Действия согласно учебному плану 7](#_Toc165904500)

[3.2 Технологии, которые требуют исследования 7](#_Toc165904501)

[3.3 Информация, которую необходимо изучить 7](#_Toc165904502)

[3.4 Практические задачи, которые необходимо будет реализовать 7](#_Toc165904503)

[3.5 Перспективы расширения работы, при достаточном времени 7](#_Toc165904504)

[Заключение 8](#_Toc165904505)

[Список использованных источников 9](#_Toc165904506)

[Приложение 10](#_Toc165904507)

Введение

Выпускная квалификационная работа (ВКР) посвящена разработке веб-интерфейса онлайн библиотеки университета. В современном информационном обществе непрерывно развиваются технологии, оптимизирующие различные процессы, в том числе и в сфере образования. В этом контексте особое значение приобретает разработка веб-приложений для онлайн библиотек. Такие приложения обеспечивают удобный доступ преподавателей и студентов к учебным материалам, повышая эффективность взаимодействия в образовательном процессе.

В рамках данной ВКР будет разработана система, позволяющая преподавателям загружать методические и лекционные материалы, а студентам осуществлять поиск и получение необходимых ресурсов через удобный веб-интерфейс с дополнительным функционалом фильтрации поискового запроса.

Актуальность разработки обусловлена повседневной необходимостью доступа к учебным материалам как для преподавателей, так и для студентов. Существующие методы хранения и обмена материалами, такие как бумажный формат или электронные ресурсы, часто ограничены по функционалу, объему или неудобны в использовании. Разработка веб-интерфейса онлайн библиотеки университета позволит устранить эти недостатки и значительно повысить доступность образовательного процесса.

Работа отличается оригинальным предложенным подходом к созданию веб-интерфейса для управления учебными материалами. Анализ аналогичных проектов показывает, что существующие решения не обладают необходимым функционалом или не предназначены для использования в контексте университетского обучения.

Таким образом, ВКР по разработке веб-интерфейса онлайн библиотеки университета является научно значимой и актуальной. Она направлена на создание продукта, предоставляющего возможность дополнительного обмена знаниями в университетской среде.

# Постановка задачи и методы её решения

## Цель работы и техническое задание

### Цель работы

Целью работы является разработка веб-приложения, которое предоставит удобный и эффективный доступ к учебным материалам для студентов университета и возможность их загрузки для преподавателей.

### Задачи

• Анализ существующих решений: Изучить существующие онлайн-библиотеки и аналогичные проекты, выявив их сильные и слабые стороны, а также определив ключевые требования к разрабатываемому приложению.

• Разработка архитектуры и дизайна: Создание архитектуры приложения, включающей в себя клиентскую и серверную часть, а также дизайн пользовательского интерфейса, удобного как для преподавателей, так и для студентов.

• Реализация клиентской части: Разработка веб-интерфейса, позволяющего пользователям:

* Находить учебные материалы по различным критериям (название, автор, тема, дата публикации и т.д.)
* Просматривать информацию о материалах (название, автор, описание)
* Скачивать материалы в удобном формате
* Отслеживать загруженные материалы

• Реализация серверной части: Создание серверной части приложения, обеспечивающей:

* Загрузку и хранение учебных материалов в базе данных
* Проверку авторизации пользователей (преподавателей и студентов)
* Обработку запросов от клиента

• Обеспечение безопасности: Внедрение мер безопасности для защиты данных, хранящихся в базе данных, а также предотвращения несанкционированного доступа к материалам.

• Тестирование и отладка: Проверка работоспособности приложения, устранение ошибок и недоработок.

• Подготовка документации: Создание документации по использованию приложения, включающей описание функционала, руководства пользователя и техническую документацию.

### Технические требования (Техническое задание)

Методы и способы решения разрабатываемого ПО:

• Язык программирования: JavaScript

• Фреймворк: Express.js (серверная часть), React.js (клиентская часть)

• База данных: PostgreSQL

• Система контроля версий: Git (с использованием платформы GitHub)

Особенности функционирования создаваемого ПО:

• Удобный интерфейс: Интуитивно понятный и простой в использовании интерфейс для всех пользователей.

• Функциональность: Возможность загрузки, поиска, скачивания и просмотра учебных материалов в различных форматах (PDF, DOCX, PPTX, видео).

• Безопасность: Защита данных от несанкционированного доступа и утечки информации.

• Мобильная адаптация: Веб-приложение должно корректно работать на различных устройствах (компьютеры, планшеты, смартфоны).

Интерфейс (дизайн) ПО:

• Современный дизайн: Использование современных элементов дизайна, обеспечивающих привлекательный внешний вид.

• Адаптивный дизайн: Отображение интерфейса в зависимости от размера экрана устройства.

• Интуитивная навигация: Простая и понятная навигация по меню, позволяющая легко находить необходимые материалы.

Требования по используемой аппаратуре и вспомогательному ПО:

• Сервер: Сервер с установленным PostgreSQL

• Операционная система: Windows

• Веб-браузер

# Обзор существующих программных средств

## Программные средства

Электронные библиотеки играют важную роль в современном образовательном процессе, предоставляя пользователям доступ к разнообразным учебным материалам, научным статьям, книгам и другим ресурсам. Существует множество веб-приложений электронных библиотек, каждое из которых предлагает свои уникальные функции и возможности.

Электронные библиотеки можно разделить на несколько категорий в зависимости от их целевой аудитории и функциональности:

1. Академические электронные библиотеки - предоставляют доступ к научным статьям, диссертациям, книгам и другим академическим ресурсам.
2. Общественные электронные библиотеки - содержат коллекции книг, журналов и других материалов для широкой аудитории.
3. Специализированные электронные библиотеки - ориентированы на конкретные области знаний или профессиональные сообщества.
4. Образовательные платформы с библиотеками - интегрируют функции электронных библиотек в образовательные платформы.

Примеры существующих программных средств:

1. eLibrary.ru

Описание: eLibrary.ru - это крупнейшая российская научная электронная библиотека, предоставляющая доступ к статьям, журналам, книгам и конференционным материалам.

Функции: Поиск по ключевым словам, доступ к полным текстам, создание персонализированных списков чтения.

Сравнение: eLibrary.ru предоставляет более широкий спектр академических ресурсов и обладает сложной системой поиска и фильтрации. Однако, в отличие от предлагаемой разработки, eLibrary.ru не ориентирован на загрузку и управление учебными материалами преподавателями.

1. КиберЛенинка

Описание: КиберЛенинка - это научная электронная библиотека, ориентированная на популяризацию науки и поддерживающая открытый доступ к научным публикациям.

Функции: Скачивание статей, поиск по автору и названию, доступ к полным текстам.

Сравнение: КиберЛенинка фокусируется на научных статьях и поддерживает открытый доступ, что делает её ресурсы доступными для всех. Тем не менее, как и eLibrary.ru, она не предоставляет функционал для преподавателей по загрузке и управлению учебными материалами.

1. Научная электронная библиотека диссертаций РГБ

Описание: Научная электронная библиотека диссертаций Российской государственной библиотеки предоставляет доступ к авторефератам и диссертациям.

Функции: Поиск по ключевым словам, доступ к полным текстам диссертаций.

Сравнение: Эта библиотека специализируется на диссертациях и авторефератах, предоставляя доступ к научным исследованиям. Однако она не предназначена для загрузки и управления учебными материалами преподавателями.

1. Электронная библиотека Московского государственного университета (МГУ)

Описание: Электронная библиотека МГУ предоставляет доступ к широкому спектру учебных и научных материалов, включая книги, статьи, учебные пособия и диссертации, созданные в университете.

Функции: Поиск по ключевым словам, доступ к полным текстам, загрузка материалов для сотрудников и студентов университета.

Сравнение: Электронная библиотека МГУ предоставляет специализированный доступ к ресурсам университета и поддерживает загрузку и управление учебными материалами преподавателями, что делает её функционал более схожим с предлагаемой разработкой.

В отличие от существующих электронных библиотек, предлагаемая разработка предоставляет специфический функционал для преподавателей и студентов, что делает её более подходящей для образовательных учреждений. Данное веб приложение также поддерживает динамическое взаимодействие между преподавателями и студентами, включая функции личного кабинета и администрирования.

## Технологии, используемые для выполнения целей ВКР

Язык программирования:

* JavaScript: язык программирования, используемый для написания как клиентской, так и серверной части приложения. JavaScript популярен благодаря своей универсальности, широкой поддержке и большому сообществу разработчиков (<https://developer.mozilla.org/en-US/docs/Web/JavaScript>).

Библиотеки и фреймворки:

* Node.js: среда выполнения JavaScript, позволяющая запускать JavaScript-код на стороне сервера (<https://nodejs.org/en/>)
* Express.js: фреймворк для разработки веб-приложений на JavaScript, упрощающий создание серверной части и маршрутизацию запросов (<https://expressjs.com/>)
* React.js: библиотека JavaScript для создания пользовательских интерфейсов, известная своей простотой в использовании и высокой производительностью (<https://reactjs.org/>)
* PostgreSQL: мощная и гибкая система управления реляционными базами данных с открытым исходным кодом (<https://www.postgresql.org/>)

Система контроля версий:

* GitHub: онлайн-платформа для хостинга репозиториев Git и совместной работы над проектами (<https://github.com/>)

Операционная система:

* Windows: многопользовательская операционная система с графическим пользовательским интерфейсом, обладающая широкой распространенностью и поддержкой различных программных продуктов (<https://ru.wikipedia.org/wiki/Microsoft_Windows>)

## Java Script

JavaScript - это современный и многофункциональный язык программирования, который широко используется для разработки как клиентской, так и серверной части веб-приложений. JavaScript стал основным языком для создания интерактивных элементов на веб-страницах и с тех пор значительно эволюционировал, став одним из самых популярных и востребованных языков веб-разработки.

JavaScript относится к семейству языков, которые вдохновлены языком программирования Java, но отличается от него синтаксисом и возможностями. Он также имеет много общих черт с другими скриптовыми языками, такими как Python и Ruby.

Некоторые ключевые особенности и функции JavaScript, которые делают его мощным и эффективным языком программирования:

* Динамическая типизация: Переменные в JavaScript могут быть использованы для хранения различных типов данных, и их тип можно изменить во время выполнения программы.
* Функциональное программирование: JavaScript поддерживает функции как объекты первого класса, что означает, что функции могут быть переданы как аргументы другим функциям и возвращены из других функций.
* Асинхронное программирование: с помощью функций обратного вызова (callback) и промисов JavaScript обеспечивает эффективное управление асинхронными операциями, такими как запросы к серверу или обработка событий пользователя.
* Объектно-ориентированное программирование: JavaScript поддерживает концепции объектов, классов, наследования и инкапсуляции, что позволяет создавать структурированный и модульный код.
* Многофункциональность: JavaScript может быть использован для создания различных типов приложений, включая веб-приложения, мобильные приложения и игры.
* JavaScript также имеет широкую экосистему библиотек и фреймворков, таких как React.js, Angular и Vue.js, которые облегчают разработку сложных веб-приложений и улучшают производительность разработчиков.

## Node.js

Node.js – это среда выполнения JavaScript, которая позволяет запускать JavaScript-код на стороне сервера. Node.js значительно расширяет возможности JavaScript, позволяя разработчикам создавать высокопроизводительные и масштабируемые веб-приложения, работающие как на сервере, так и на клиенте. Благодаря асинхронной и неблокирующей архитектуре Node.js обеспечивает эффективную обработку большого количества запросов и событий одновременно. Node.js также имеет обширный набор модулей и библиотек, что делает его идеальным выбором для разработки серверных приложений, веб-сервисов и API.

## Express.js

Express.js – это минималистичный и гибкий фреймворк для разработки веб-приложений на JavaScript. Он строится поверх Node.js и упрощает создание серверной части и маршрутизацию запросов. Express.js предоставляет простой и интуитивно понятный интерфейс для обработки HTTP-запросов, определения маршрутов, управления сеансами и многое другое. Он позволяет разработчикам создавать эффективные и масштабируемые веб-приложения, используя минимум кода и максимум функциональности. Благодаря своей популярности и активному сообществу разработчиков, Express.js является одним из самых предпочтительных выборов для создания API и веб-серверов на Node.js.

## React.js

React.js – это мощная и гибкая библиотека JavaScript для создания пользовательских интерфейсов. Она известна своей простотой в использовании, высокой производительностью и модульной архитектурой. React.js позволяет разработчикам создавать интерактивные и динамичные веб-приложения, разбивая их на множество компонентов, которые могут быть переиспользованы и обновляются автоматически при изменении данных. Одной из ключевых особенностей React.js является виртуальный DOM (Document Object Model), который обеспечивает эффективное обновление пользовательского интерфейса и повышает производительность приложения. Благодаря своей популярности и обширному экосистеме компонентов и библиотек, React.js стал одним из ведущих инструментов для разработки современных веб-приложений.

## Sequelize

Sequelize – это популярный ORM (Object-Relational Mapping) для Node.js, который позволяет взаимодействовать с реляционными базами данных с использованием объектно-ориентированного подхода. Созданный для упрощения работы с базами данных, Sequelize предоставляет удобный API для определения моделей, выполнения запросов и управления связями между таблицами.

Основные возможности Sequelize включают:

* Модели: Определение моделей данных с использованием JavaScript-классов, что облегчает работу с таблицами и их полями.
* Ассоциации: Удобное управление связями между таблицами, включая "один к одному", "один ко многим" и "многие ко многим".
* Запросы: Возможность выполнения сложных SQL-запросов с использованием цепочек методов и других высокоуровневых функций.
* Миграции: Инструменты для управления версиями схемы базы данных, что упрощает процесс её изменения и обновления.
* Валидация и Преобразование: Встроенные механизмы для проверки данных и их преобразования перед сохранением в базу данных.

Sequelize поддерживает различные базы данных, такие как MySQL, PostgreSQL, SQLite и MSSQL, что делает его универсальным инструментом для разработки серверных приложений. Благодаря обширной документации и активному сообществу, Sequelize является идеальным выбором для разработчиков, стремящихся упростить взаимодействие с базами данных в своих проектах.

## Bootstrap

Bootstrap – это популярный фреймворк для разработки интерфейсов веб-приложений, созданный командой разработчиков из Twitter (популярная социальная сеть). Bootstrap предоставляет набор готовых CSS и JavaScript компонентов, которые помогают разработчикам быстро и эффективно создавать адаптивные и привлекательные веб-страницы. Фреймворк включает в себя стили для типографики, форм, кнопок, навигации и других элементов пользовательского интерфейса, а также инструменты для построения сетки, что обеспечивает удобство и гибкость в вёрстке страниц. Кроме того, Bootstrap имеет активное сообщество и обширную документацию, что облегчает процесс изучения и внедрения фреймворка в проекты.

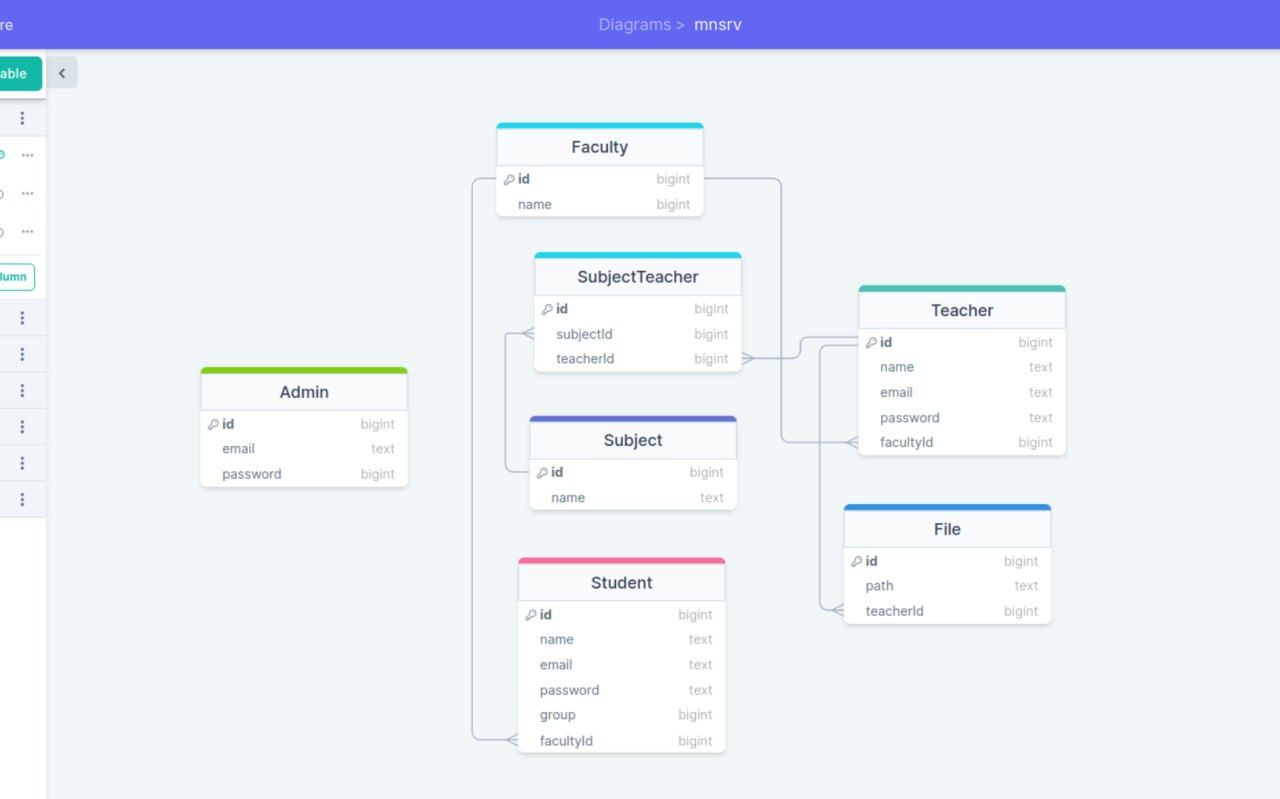
# Описание разработанных программных средств

## Ы

Третий раздел отчета посвящен описанию разработанного ПО. Здесь могут быть представлены:

* Общая схема работы приложения;
* Структура разработанного ПО;
* Схема модулей приложения;
* Описание разработанных модулей приложения;

## Схема базы данных



## Описание таблиц базы данных;

1. Admin (Администратор):

* id (идентификатор): bigint — Уникальный идентификатор администратора.
* email (электронная почта): text — Адрес электронной почты администратора.
* password (пароль): text — Пароль для входа администратора.

Таблица "Admin" хранит данные о администраторах, которые управляют системой.

2. Faculty (Факультет):

* id (идентификатор): bigint — Уникальный идентификатор факультета.
* name (название): text — Название факультета.

Таблица "Faculty" содержит информацию о различных факультетах университета.

3. Subject (Предмет):

* id (идентификатор): bigint — Уникальный идентификатор предмета.
* name (название): text — Название предмета.

Таблица "Subject" хранит данные о предметах, которые преподаются в университете.

4. Teacher (Преподаватель):

* id (идентификатор): bigint — Уникальный идентификатор преподавателя.
* name (имя): text — Имя преподавателя.
* email (электронная почта): text — Адрес электронной почты преподавателя.
* password (пароль): text — Пароль для входа преподавателя.
* facultyId (идентификатор факультета): bigint — Идентификатор факультета, к которому принадлежит преподаватель.

Таблица "Teacher" содержит данные о преподавателях, включая информацию о том, на каком факультете они работают.

5. SubjectTeacher (ПреподавательПредмет):

* id (идентификатор): bigint — Уникальный идентификатор записи.
* subjectId (идентификатор предмета) : bigint — Идентификатор предмета.
* teacherId (идентификатор преподавателя): bigint — Идентификатор преподавателя.

Таблица "SubjectTeacher" связывает преподавателей с предметами, которые они преподают.

6. Student (Студент):

* id (идентификатор): bigint — Уникальный идентификатор студента.
* name (имя): text — Имя студента.
* email (электронная почта): text — Адрес электронной почты студента.
* password (пароль): text — Пароль для входа студента.
* group (группа): text — Группа, в которой учится студент.
* facultyId (идентификатор факультета): bigint — Идентификатор факультета, на котором учится студент.

Таблица "Student" хранит данные о студентах, включая их группу и факультет.

7. File (Файл):

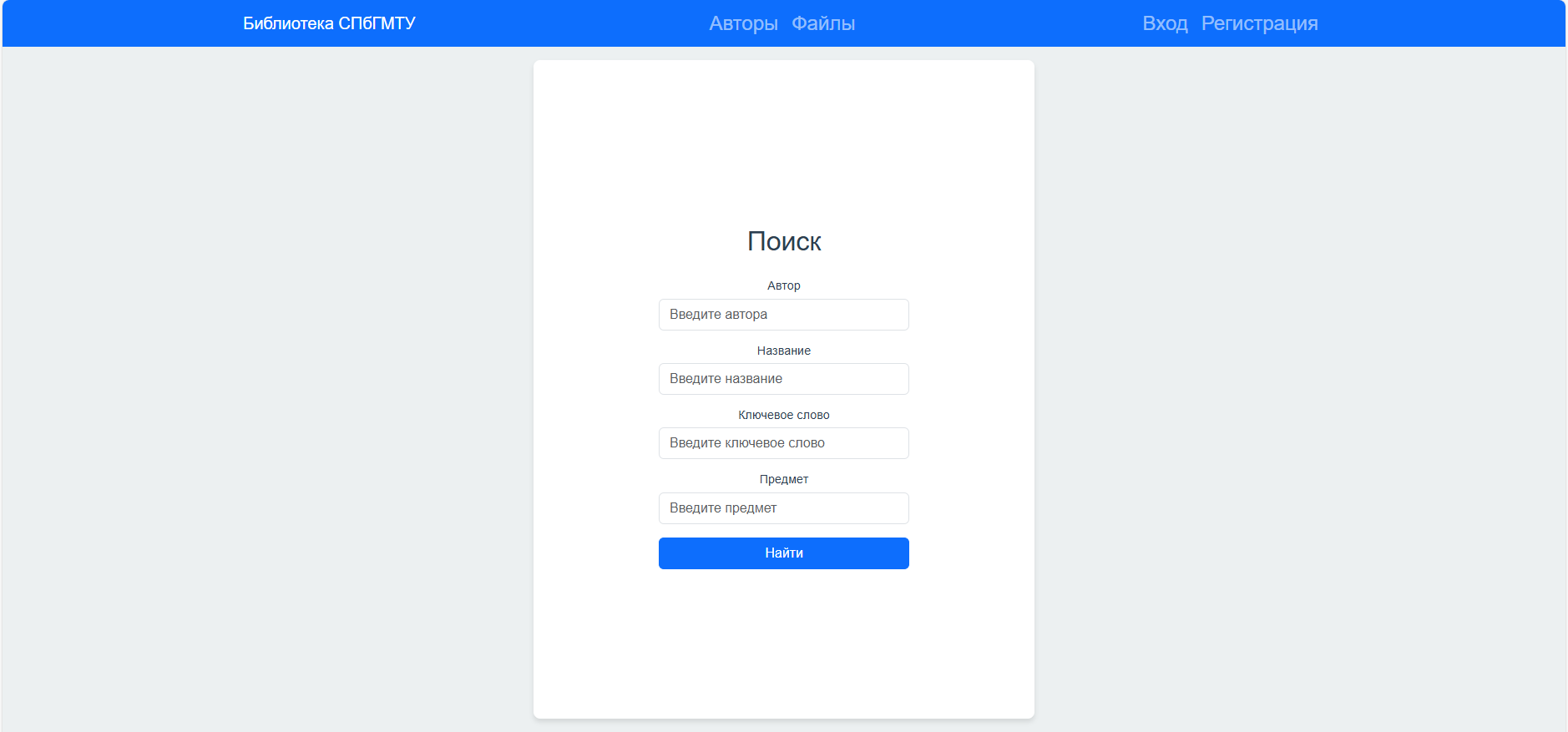
* id (идентификатор): bigint — Уникальный идентификатор файла.
* path (путь): text — Путь к файлу.
* teacherId (идентификатор преподавателя): bigint — Идентификатор преподавателя, загрузившего файл.

Таблица "File" содержит информацию о файлах, которые загружают преподаватели.

Каждая из таблиц играет важную роль в функционировании сайта онлайн-библиотеки, обеспечивая хранение и организацию данных о пользователях, предметах, факультетах и файлах. Связи между таблицами позволяют управлять отношениями между студентами, преподавателями, предметами и загруженными материалами.

## Описание разработанных модулей приложения

### Главная страница



На изображении представлена главная страница сайта онлайн-библиотеки университета СПбГМТУ. На этой странице пользователям предоставлена возможность искать учебные материалы, загруженные преподавателями, по различным критериям. Опишем функции и содержимое страницы подробнее.

1. **Содержимое страницы**

* **Форма поиска** состоит из четырех полей ввода и кнопки для выполнения поиска. Пользователи могут заполнить данные поля для более точного поиска учебных материалов.
* **Навигационная панель** обеспечивает легкий доступ к основным разделам сайта, позволяя пользователям быстро переходить к нужной информации.

1. **Навигационная панель**

В верхней части страницы находится навигационная панель, включающая следующие элементы:

* **Библиотека СПбГМТУ** — название сайта, которое является ссылкой на главную страницу.
* **Авторы** — ссылка на страницу, где пользователи могут просмотреть список преподавателей, загружающих материалы.
* **Файлы** — ссылка на страницу, где пользователи могут просмотреть все доступные для скачивания файлы.
* **Вход** — ссылка для перехода на страницу авторизации пользователей.
* **Регистрация** — ссылка для перехода на страницу регистрации новых пользователей.

При авторизации пользователя, ссылки "Вход" и "Регистрация" заменяются на:

* **Панель Администратора** — для администраторов сайта. Ссылка для перехода в административную панель, где можно управлять контентом и пользователями.
* **Личный кабинет** — для преподавателей. Ссылка для перехода в личный кабинет, где преподаватели могут управлять своими материалами.
* **Выйти** — для всех видов пользователей. Кнопка для реализации выхода из учетной записи.

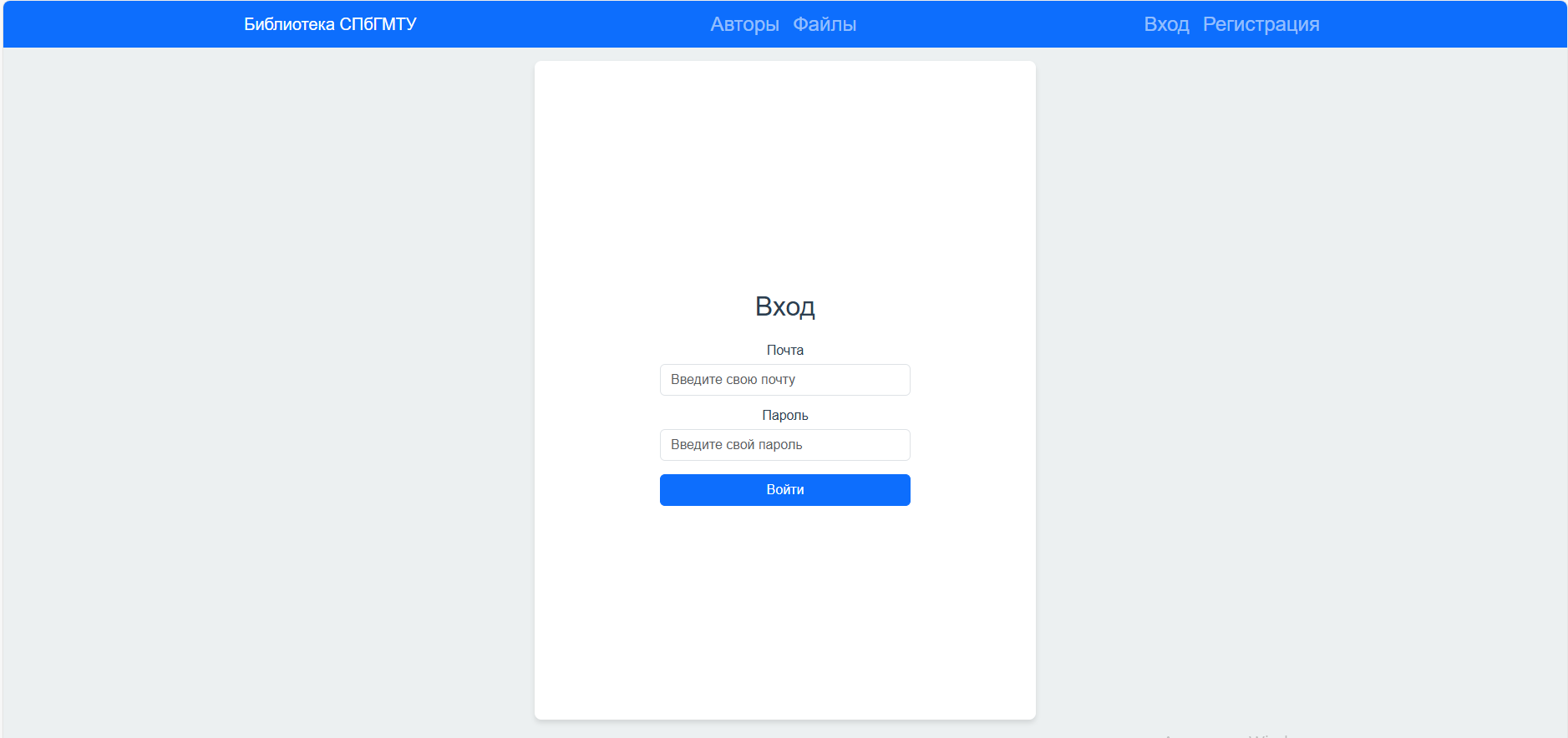
1. **Поисковая форма**

Центральная часть страницы занимает форма поиска, предназначенная для нахождения файлов по различным параметрам. Форма включает следующие поля ввода:

* **Автор** — поле для ввода имени автора (преподавателя), по которому будет выполняться поиск.
* **Название** — поле для ввода названия материала.
* **Ключевое слово** — поле для ввода ключевых слов, присутствующих в описании файлов. Поиск по ключевым словам позволяет найти материалы, которые содержат заданные термины в описании.
* **Предмет** — поле для ввода названия предмета, для которого ищется материал.
* **Найти** — кнопка, которая запускает процесс поиска по введенным критериям.

Ввод данных можно осуществлять одновременно в несколько полей для уточнения запроса. В этом случае параметры поиска объединяются, и в результат попадают только те элементы, которые удовлетворяют всем указанным условиям.

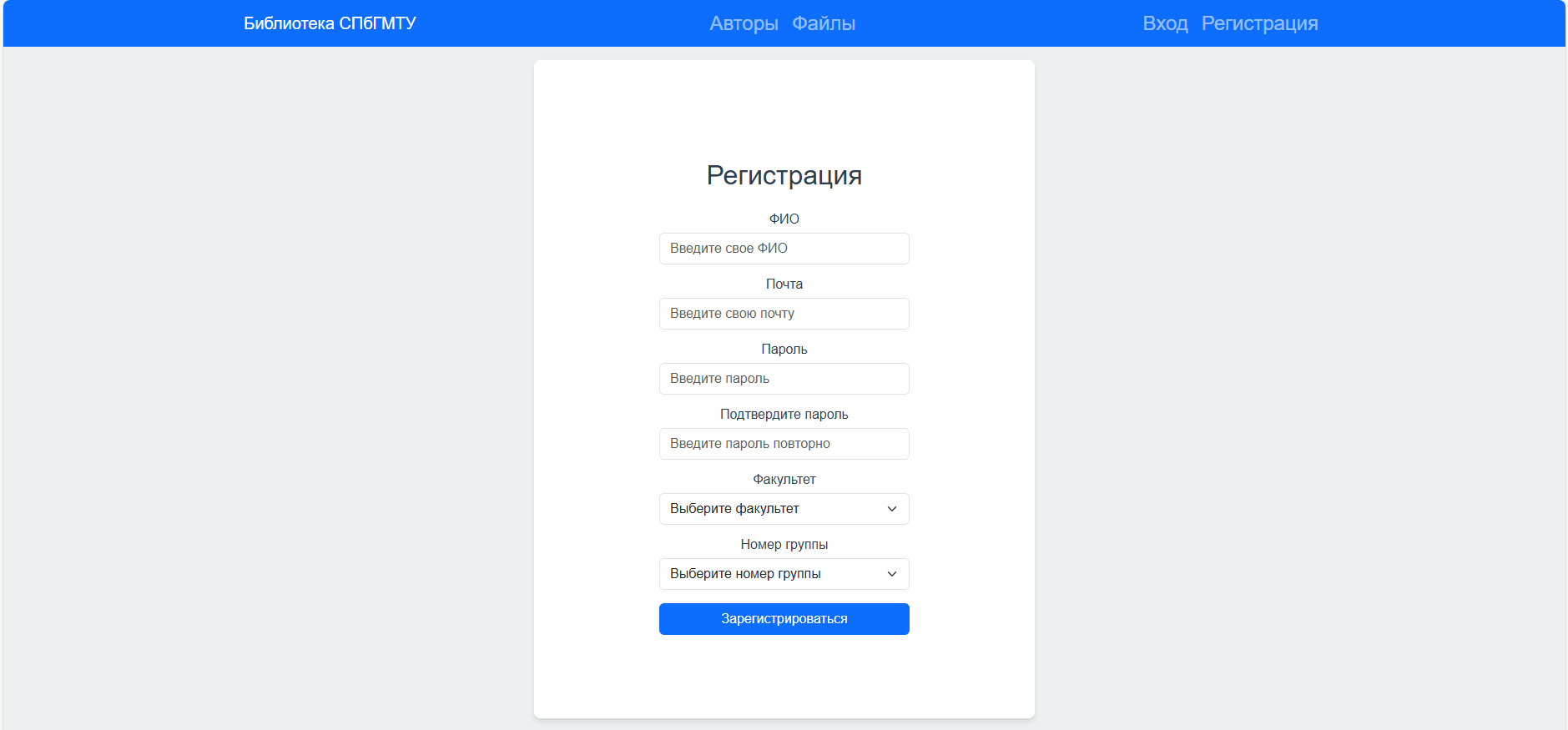
### Форма авторизации



Центральная часть страницы занимает форма входа, предназначенная для авторизации пользователей. Форма включает следующие поля ввода:

* **Почта** — поле для ввода электронной почты пользователя. Пользователь вводит свою почту для авторизации.
* **Пароль** — поле для ввода пароля пользователя. Пользователь вводит свой пароль для авторизации.
* **Войти** — кнопка, которая запускает процесс авторизации по введенным данным. При нажатии происходит проверка введенных почты и пароля по базе данных. Если данные корректны, пользователь перенаправляется на его личный кабинет (для студентов и преподавателей) или административную панель (для администраторов).

### Форма регистрации студента



На странице, описанной вами, центральная часть занимает форма входа, предназначенная для авторизации пользователей. Давайте разберемся подробнее:

* **Почта** — поле для ввода электронной почты пользователя. Пользователь вводит свою почту для авторизации.
* **Пароль** — поле для ввода пароля для авторизации. Это обязательное поле, которое обеспечивает безопасность доступа к аккаунту.
* **Войти** — кнопка для запуска процесса авторизации на основе введенных данных. При нажатии происходит проверка введенных почты и пароля по базе данных.

Если данные корректны, пользователь перенаправляется на свой личный кабинет (для студентов и преподавателей) или административную панель (для администраторов).

Эта форма представляет собой важный элемент для обеспечения безопасности и удобства

пользователей при входе на сайт.

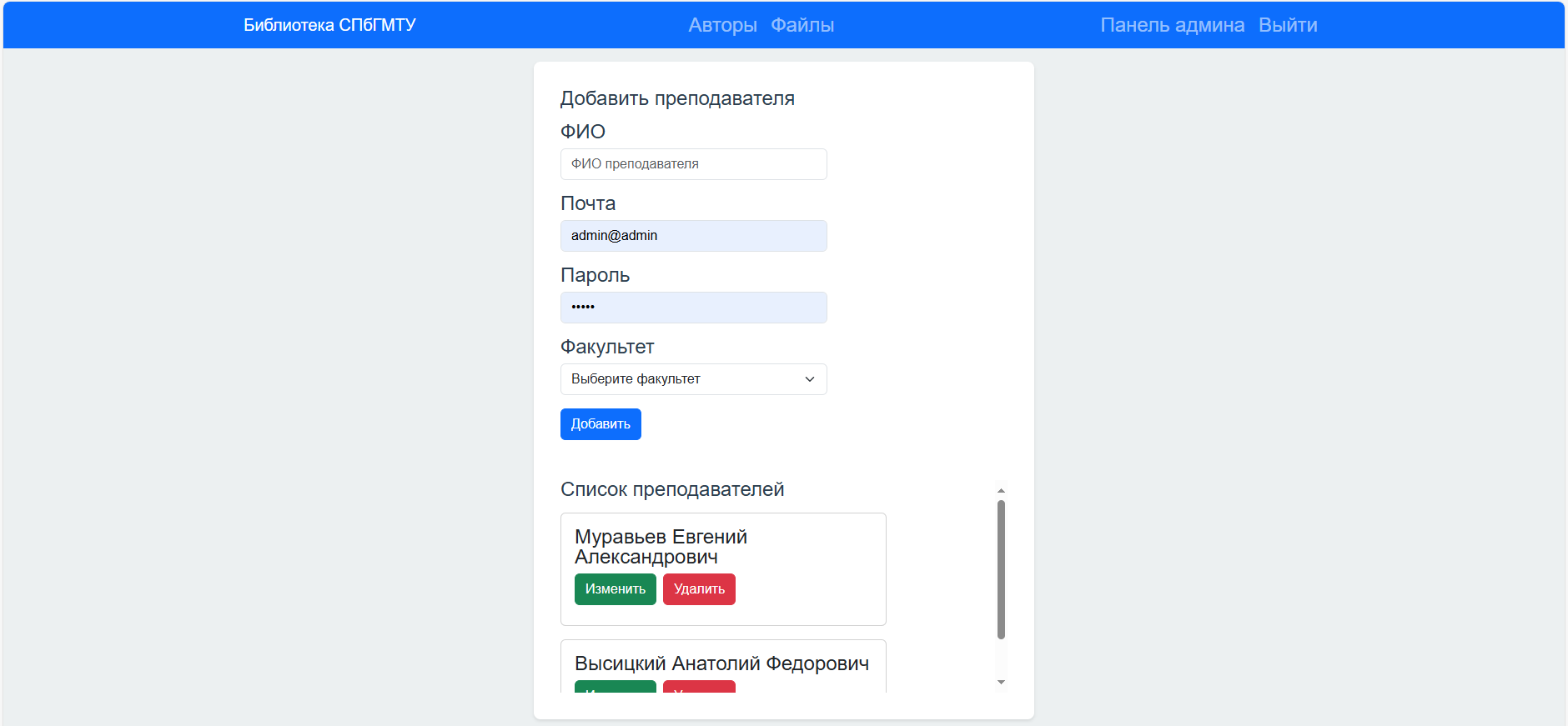
### Форма загрузки материалов

### 

Центральная часть страницы занимает форма загрузки материалов, предназначенная для преподавателей. Форма включает следующие поля ввода:

* **Название** — поле для ввода названия материала. Преподаватель вводит название загружаемого материала.
* **Описание** — поле для ввода описания материала. Преподаватель вводит краткое описание загружаемого материала.
* **Предмет** — выпадающий список для выбора предмета. Пользователь выбирает предмет, к которому относится добавляемый материал.
* **Добавить файл** — кнопка для выбора файла на компьютере пользователя. Пользователь выбирает файл, который хочет загрузить в систему.
* **Загрузить** — кнопка, которая запускает процесс загрузки материала с введенными данными. После нажатия кнопки создается отдельная страница для загруженного материала, где будут отображены его название, описание, автор и дата загрузки. Этот материал станет доступным для поиска на сайте и будет отображаться в общем списке материалов. Кроме того, он появится в личном кабинете преподавателя.

### Панель администратора и форма добавления преподавателя



Центральная часть страницы занимает форма добавления преподавателя, предназначенная для администраторов. Форма включает следующие поля ввода:

* **ФИО** — поле для ввода полного имени преподавателя. Администратор вводит фамилию, имя и отчество нового преподавателя.
* **Почта** — поле для ввода электронной почты преподавателя. Администратор вводит почту, которая будет использоваться для авторизации преподавателя на сайте.
* **Пароль** — поле для ввода пароля преподавателя. Администратор вводит пароль, который преподаватель будет использовать для входа на сайт.
* **Факультет** — выпадающий список для выбора факультета. Администратор выбирает факультет, к которому будет привязан преподаватель.
* **Добавить** — кнопка, которая запускает процесс добавления в базу данных нового преподавателя с введенными данными. После нажатия кнопки преподаватель будет зарегистрирован в системе и получит возможность авторизоваться на сайте.

После успешного добавления преподавателя он сможет войти в свою учетную запись и управлять своими материалами.

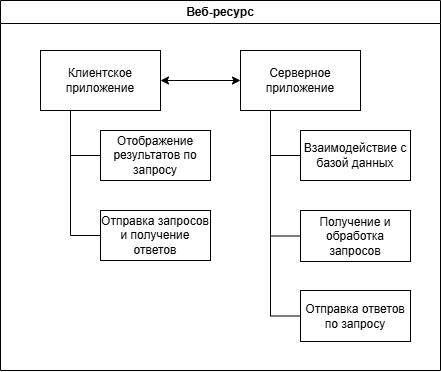
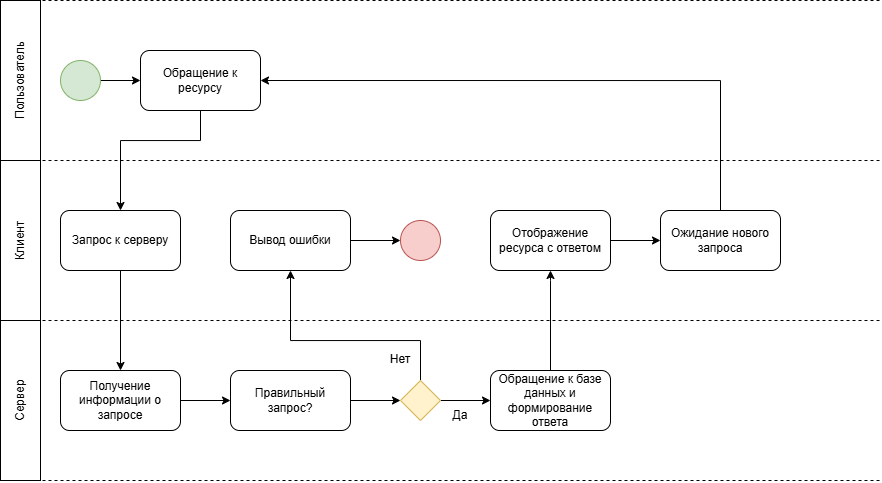
Ниже формы добавления преподавателя расположен список преподавателей, в котором видно их фамилию, имя, отчество, и есть возможность выполнить действия путем нажатия кнопок:

• **Изменить** — зеленая кнопка, которая позволяет администратору редактировать данные преподавателя.

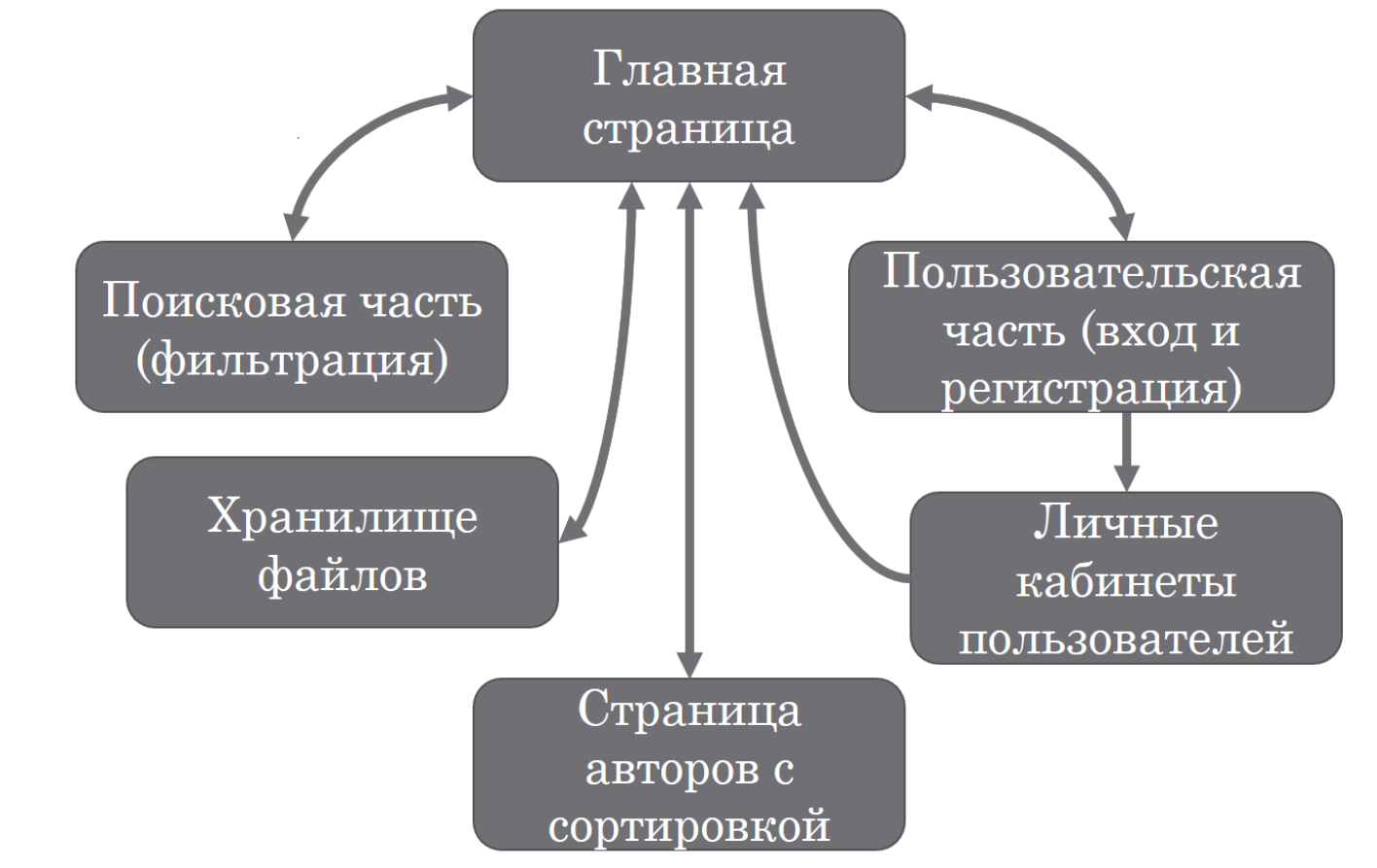
• **Удалить** — красная кнопка, которая позволяет администратору удалить преподавателя из системы.

# Логическая и физическая структуры разработанного сайта или Интернет-проекта.

bpmn диаграмма



## Логическая структура



### Регистрация (доступна только для студентов, так как преподавателей регистрирует администратор)

1. Студент вводит свои данные, выбирает факультет, и от выбранного факультета зависит выбор группы. При выборе факультета срабатывает скрипт на JavaScript, который отображает список доступных групп, связанных с выбранным факультетом.

2. Студент нажимает кнопку регистрации, после чего введенные данные отправляются с клиентской стороны на сервер.

3. На сервере проводятся следующие проверки:

3.1. Правильность повторно введенного пароля. Если пароль введен неверно, формируется ответ с ошибкой и соответствующим сообщением, который отправляется на клиент.

3.2. Уникальность электронной почты. Если студент с такой почтой уже существует, формируется ответ с ошибкой, который отправляется на клиент.

3.3. При успешном прохождении всех проверок, пароль хэшируется и происходит запись нового студента в базу данных.

4. Сервер формирует ответ и отправляет его на клиент. В случае успешной регистрации клиент перенаправляет студента на страницу авторизации.

### Авторизация

1. Пользователь вводит свои данные (электронную почту и пароль).

2. Пользователь нажимает кнопку авторизации, отправляя введенные данные на сервер.

3. Клиент собирает и отправляет данные на сервер.

4. На сервере проводятся следующие проверки:

4.1. Если пользователь не найден, формируется ответ с ошибкой.

4.2. Проверяется роль пользователя (студент, преподаватель, администратор).

4.3. Сравнивается хэшированный пароль с паролем в базе данных. Если пароли не совпадают, формируется ответ с ошибкой.

5. В случае успешной проверки сервер создает два JWT токена с разным временем жизни, включающие данные о пользователе, и загружает эти токены в cookies браузера для дальнейшего использования.

6. Сервер формирует и отправляет ответ на клиент.

7. Клиент получает положительный ответ и перенаправляет авторизованного пользователя на главную страницу.

### Поиск файлов

1. Пользователь заполняет необходимые поля в форме поиска и нажимает кнопку.

2. Клиент собирает данные и отправляет запрос на сервер.

3. Сервер принимает данные и производит поиск в базе данных по указанным параметрам. Если данные найдены, сервер формирует ответ и отправляет его на клиент.

4. Клиент получает данные и отображает результаты поиска.

### Вывод преподавателей по факультетам

1. Пользователь переходит на вкладку «Авторы».

2. Клиент отправляет запрос на сервер.

3. Сервер обращается к базе данных, выполняет поиск всех факультетов и связанных с ними преподавателей, формирует ответ и отправляет его на клиент.

4. Клиент получает данные и отображает их.

### Отображение всех файлов

1. Пользователь переходит на вкладку «Файлы».

2. Клиентская часть отправляет запрос на сервер.

3. Сервер обращается к базе данных, осуществляет поиск всех файлов и связанных с ними преподавателей, затем формирует ответ и отправляет его клиенту.

4. Клиент принимает данные и отображает их на странице.

### Сортировка файлов

1. Пользователь выбирает критерии сортировки, после чего скрипт отображает выбор сортировки (по возрастанию или убыванию). Пользователь нажимает кнопку для подтверждения.

2. Клиентская часть собирает данные о необходимых файлах и отправляет запрос на сервер.

3. Сервер принимает данные, выполняет поиск всех файлов в базе данных, сортирует их по заданным критериям, формирует ответ и отправляет его клиенту.

4. Клиент принимает данные и отображает отсортированные результаты.

### Предпросмотр файлов

1. Пользователь нажимает на файл для его просмотра.

2. Клиентская часть отправляет запрос на сервер с ID выбранного файла.

3. Сервер ищет файл по ID в базе данных, собирает всю информацию о файле, формирует ответ и отправляет его клиенту.

4. Клиент принимает данные и отображает их для предварительного просмотра.

### Скачивание файлов

1. Пользователь нажимает на кнопку скачивания.

2. Клиентская часть отправляет запрос на сервер с указанием пути к файлу.

3. Сервер предоставляет файл по указанному пути и отправляет его клиенту.

4. Клиент принимает файл и предлагает пользователю сохранить его.

### Редактирование файлов

1. Преподаватель нажимает кнопку редактирования.

2. Клиентская часть отправляет запрос на сервер.

3. Сервер ищет файл в базе данных, проверяет права доступа преподавателя к этому файлу. Если файл не принадлежит преподавателю, формируется ответ с ошибкой. Если принадлежит, сервер отправляет положительный ответ с данными о файле.

4. Клиент принимает данные и перенаправляет преподавателя на страницу редактирования.

5. Преподаватель вносит изменения и нажимает кнопку сохранения.

6. Клиентская часть отправляет измененные данные на сервер.

7. Сервер принимает данные, вносит изменения в базу данных и отправляет ответ клиенту.

8. Клиент принимает положительный ответ и возвращает пользователя на предыдущую страницу.

### Удаление файлов

1. Преподаватель нажимает кнопку удаления.

2. Клиентская часть отправляет запрос на сервер с ID файла.

3. Сервер ищет файл в базе данных, проверяет права доступа преподавателя. Если файл не принадлежит преподавателю, формируется ответ с ошибкой. Если принадлежит, сервер удаляет запись из базы данных и файл с сервера, затем отправляет ответ клиенту.

4. Клиент принимает положительный ответ и возвращает преподавателя на предыдущую страницу.

### Добавление файлов

1. Преподаватель вводит данные и загружает файл через клиентскую часть.

2. Клиентская часть отправляет данные и файл на сервер.

3. Сервер получает данные и файл, сохраняет файл в определенное место в проекте, создает запись в базе данных и отправляет ответ клиенту.

4. Клиент принимает положительный ответ и отображает новый добавленный файл.

### Регистрация преподавателей

1. Администратор вводит данные и нажимает кнопку регистрации.

2. Клиентская часть отправляет данные на сервер.

3. Сервер создает новую запись в базе данных и отправляет ответ клиенту.

4. Клиент принимает ответ и отображает информацию о новом пользователе.

### Редактирование преподавателей

1. Администратор нажимает кнопку редактирования.

2. Клиентская часть отправляет запрос на сервер.

3. Сервер ищет информацию о преподавателе в базе данных, проверяет права доступа администратора на редактирование данной записи. Если права доступа отсутствуют, формируется ответ с ошибкой. Если права доступа подтверждены, сервер отправляет положительный ответ с данными о преподавателе.

4. Клиент принимает данные и перенаправляет администратора на страницу редактирования.

5. Администратор вносит изменения и нажимает кнопку сохранения.

6. Клиентская часть отправляет измененные данные на сервер.

7. Сервер принимает данные, вносит изменения в базу данных и отправляет ответ клиенту.

8. Клиент принимает положительный ответ и возвращает пользователя на предыдущую страницу.

### Удаление преподавателей

1. Администратор нажимает кнопку удаления.
2. Клиентская часть отправляет запрос на сервер с id выбранного преподавателя.
3. Сервер получает запрос, ищет в базе данных преподавателя по id, проверяет права администратора на удаление данной записи. Если права отсутствуют, сервер отправляет ответ с ошибкой. Если права подтверждены, сервер обращается к базе данных и удаляет запись о преподавателе. После этого сервер формирует и отправляет ответ клиенту.
4. Клиент принимает положительный ответ и перенаправляет администратора на предыдущую страницу.

### Основные принципы работы клиент-серверной архитектуры

1. Клиентская часть отправляет запросы на сервер, сервер определяет тип запроса и соответствующие действия.

2. Сервер обрабатывает запросы, взаимодействует с базой данных для получения или записи данных, затем формирует и отправляет ответ клиенту.

3. Клиентская часть принимает ответы сервера и отображает полученную информацию на странице.

4. Прямой доступ клиента к базе данных невозможен; все взаимодействие происходит через сервер.

### Реализация поиска

Поиск осуществляется путем выполнения запросов к базе данных по введенным пользователем полям. Каждое поле поиска соответствует определенному столбцу в базе данных. Например, поиск по ключевым словам выполняется по описанию файла, где ключевые слова ищутся в тексте описания для нахождения совпадений. Все поля поиска могут комбинироваться для сужения результатов поиска, что особенно полезно при большом количестве файлов.

### Реализация сортировки

Сортировка осуществляется путем отправки запроса к базе данных, в котором указывается, по каким столбцам и в каком направлении (по возрастанию или убыванию) следует сортировать данные.

### Хранение и загрузка файлов

Файлы загружаются в статическую папку проекта, а в базе данных сохраняется путь к этим файлам. Загрузка файлов реализована с использованием вспомогательной библиотеки Multer.

### Редактирование и удаление файлов или преподавателей

Редактирование и удаление файлов или преподавателей осуществляется через стандартные HTTP-запросы с использованием методов PUT или DELETE. Эти операции выполняются посредством запросов к базе данных, которые обновляют или удаляют соответствующие записи.

## Описание скриптов

* **auth.js** — скрипт принимает данные из формы авторизации или регистрации и отправляет их на сервер, а также получает ответ с сервера

**Регистрация:**

1. Находит элемент формы регистрации (.registration).
2. При отправке формы отменяет стандартное поведение и извлекает значения полей.
3. Отправляет POST-запрос на сервер по адресу /api/auth/sign-up с данными формы (имя, email, пароль, подтверждение пароля, факультет, группа).
4. Получает ответ от сервера и в зависимости от содержимого ответа:
   * Если сообщение равно "success", уведомляет пользователя об успешной регистрации и перенаправляет на страницу входа.
   * Если сообщение равно "error", уведомляет пользователя, что такой пользователь уже существует.
   * Если сообщение равно "trim", уведомляет пользователя о некорректно введенных данных.
   * Если сообщение равно "password", уведомляет пользователя, что пароли не совпадают.
   * Если другое сообщение, выводит его как есть.
5. Обрабатывает и показывает ошибки, если они возникают.

**Авторизация:**

1. Находит элемент формы авторизации (.authorization).

2. При отправке формы отменяет стандартное поведение и извлекает значения полей.

3. Отправляет POST-запрос на сервер по адресу /api/auth/sign-in с данными формы (email, пароль).

4. Получает ответ от сервера и в зависимости от содержимого ответа:

* + Если сообщение равно "success", перенаправляет пользователя на главную страницу.
  + Если другое сообщение, выводит его как есть.

1. Обрабатывает и показывает ошибки, если они возникают.

* **file.js** — скрипт обрабатывает клик на кнопку для удаления файла и отправляет запрос на сервер, после чего получает ответ с сервера

**Удаление файла**

1. Находит элемент кнопки удаления (.buttonDelete).
2. Если элемент существует, добавляет обработчик события click для этой кнопки.
3. При клике на кнопку показывает пользователю диалог подтверждения (confirm) с вопросом "Удалить данный файл?".
   * Если пользователь подтверждает удаление:
   * Получает id файла из ближайшего родительского элемента с классом .main-stack (значение атрибута data-id).
   * Отправляет DELETE-запрос на сервер по адресу /api/file/delete/${id}.
   * Ждет ответ от сервера и преобразует его в формат JSON.
   * Если сообщение в ответе равно "success", перезагружает страницу, перенаправляя пользователя на предыдущую страницу (document.referrer).
   * Если сообщение отличается от "success", выводит это сообщение как есть.
   * Если пользователь не подтверждает удаление, ничего не происходит.
   * Обрабатывает и показывает ошибки, если они возникают.

* **select.js** — скрипт для динамического обновления списка групп на основе выбранного факультета и управления отображением параметров сортировки

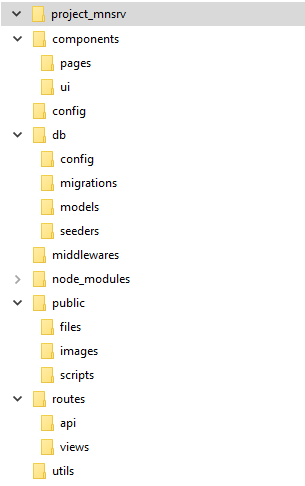
**Динамическое обновление списка групп**

* + 1. Находит элемент выпадающего списка факультетов (#facultySelect).
    2. Если элемент существует, добавляет обработчик события change для этого элемента.
    3. При изменении выбранного факультета:
  + Получает значение выбранного факультета.
  + Находит элемент выпадающего списка групп (#groupSelect).
  + Удаляет все существующие элементы внутри списка групп.
    1. Создает массив групп на основе выбранного факультета:
  + Если выбран факультет с id = 1, массив групп будет ['20490', '20491'].
  + Если выбран факультет с id = 2, массив групп будет ['3470', '3471'].
  + Для других значений факультета массив групп будет пустым.
    1. Проходит по массиву групп и добавляет каждый элемент как опцию в список групп.
    2. Обрабатывает и показывает ошибки, если они возникают.
    3. Управление отображением параметров сортировки
    4. Находит элементы, связанные с выбором сортировки:
  + Выпадающий список сортировки (#sortSelect).
  + Контейнер для порядка сортировки (#sortOrder).
  + Радиокнопки для восходящей и нисходящей сортировки (#ascending и #descending).

Если все элементы существуют, добавляет обработчик события change для выпадающего списка сортировки.

* + 1. При изменении выбранной опции в списке сортировки проверяет, выбрана ли одна из опций с value = '1', '2', или '3'.
  + Если выбрана одна из этих опций, отображает контейнер для порядка сортировки.
  + Если выбрана любая другая опция, скрывает контейнер для порядка сортировки.
    1. Сбрасывает состояние радиокнопок для восходящей и нисходящей сортировки (обе становятся невыбранными).

## Физическая структура



## Таблица разработанных файлов сайта и их описание

|  |  |
| --- | --- |
| **Файл** | **Описание** |
| AdminPage.jsx | Страница администратора. |
| Authorization.jsx | Страница авторизации. |
| Authors.jsx | Страница с информацией об авторах. |
| FileChange.jsx | Страница изменения файла. |
| Files.jsx | Страница файлов. |
| FileView.jsx | Страница просмотра файла. |
| Main.jsx | Главная страница. |
| Profile.jsx | Страница профиля пользователя. |
| RegistrationChange.jsx | Страница изменения регистрации. |
| TeacherChange.jsx | Страница изменения данных учителя. |
| Layout.jsx | Компонент макета (UI). |
| normalize.css | Файл для нормализации стилей веб страницы. |
| reset.css | Файл для сброса стилей HTML-элементов к единому базовому состоянию. |
| style.css | Файл пользовательских стилей. |

# Отладка, тестирование и результаты разработанного ПО

Тестирование программы проводилось на нескольких компьютерах с операционными системами Windows 10 и Windows 7. Запуск сайта осуществлялся через браузеры Yandex Browser и Microsoft Internet Explorer.

## Тестирование интерфейса

Тест 1.

Сортировка столбца списка файлов. Щелчок по выпадающему списку «Сортировка», выбор нужных параметров, щелчок по кнопке «Показать»

Результат: файлы упорядочены по алфавиту без обновления страницы.

Тест 2.

Добавление материала при незаполненных полях.

Результат: появление надписи с требованием заполнить нужные поля.

Тест 3.

Регистрация пользователя с вводом несовпадающих паролей в поля «Пароль» и «Повторите пароль».

Результат: появление сообщения с текстом «Пароли не совпадают!».

Тест 4.

Изменение описания уже существующего файла в личном кабинете преподавателя. Щелчок по кнопке «Изменить», ввод новых параметров, щелчок по кнопке «Сохранить»

Результат: выполнен автоматический переход на страницу файла с обновленными данными. Поля, которые не изменяли, остаются прежними.

Тест 5.

Переход по ссылкам. Нажатие на ссылки сайта: авторы, файлы.

Результат: происходит переход по всем ссылкам сайта

Тест 6.

Скачивание файла. Щелчок по кнопке «Скачать» на странице нужного файла.

Результат: происходит скачивание файла.

Тест 7.

Поиск. Ввод данных в текстовое поле нужного из поисковых параметров (одного или нескольких), нажатие кнопки "Поиск".

Результат: на странице выводятся только те файлы, где присутствуют данные, соответствующие требуемым.

Тест 8.

Выход из личного кабинета пользователя. Щелчок по кнопке «Выйти».

Результат: перенаправление на главную страницу сайта. Отсутствие профиля и возможность входа/регистрации присутствует.

## Тестирование внутренней структуры

Тест 9.

Учетные политики. Преподаватель не может удалить или изменить файлы, загруженные другими пользователями.

Результат: кнопка удаления и изменения отсутствует как таковая.

Тест 10.

Целостность таблиц. Удаление из таблицы «Преподаватели» автора, у которого есть загруженные файлы.

Результат: вывод сообщения о нарушении ограничения внешнего ключа. Преподавателя удалить невозможно, сначала нужно удалить его файлы.

## Тестирование вводимых данных

Тест 11.

Попытка регистрации с уже зарегистрированным ранее адресом почты.

Результат: вывод сообщения с текстом: «Пользователь уже существует!».

Тест 12.

Попытка входа с неправильно введенным паролем.

Результат: вывод сообщения с текстом: «Неверно введены данные».

## Тестирование на соответствие постановке задачи

Тест 13.

Соответствие постановке задачи. Щелчки мышкой по гиперссылкам, выполненным в виде соответствующей подписи к ней на сервисы: авторы, файлы, библиотека спбгмту. Результат: требуемые части страниц подгружаются без обновления всей страницы.

В процессе тестирования в виде спланированной последовательности шагов было выявлено, что они привели к успешному построению программной системы.

# Руководство пользователя

## Ы

В пятом разделе отчета приводится руководство пользователя разработанного приложения. Рекомендуется описание действий различных видов пользователей пояснять скриншотами – копиями экранов с примерами выбора пунктов меню, установки параметров, ввода исходных данных и т.д. Скриншоты оформляются в виде графических изображений с подрисуночными подписями.

# Руководство по установке и сопровождению ПО

## ы

Шестой раздел отчета представляет руководство по установке и сопровождению ПО, позволяющее управлять приложением после его создания, изменять и расширять его функциональность. В разделе могут быть описаны следующие моменты:

# требования к разработчику,

# программные требования,

# хранение данных,

# руководство по установке разработанного ПО,

# защита приложения после установки,

# перенос приложения на другой компьютер/сервер,

# обновление приложения.

Заключение

1-2 страницы

Общие слова, связанные с результатами текущих исследований, какие источники используются чаще всего

Общая оценка состояния готовности ВКР (можно в процентах - можно иными словами)

Перспективы работ по теме ВКР на ближайшие 2 месяца

Перспективы развития темы на магистратуре (даже если она не планируется, можно добавить то, что вам было бы интересно, что хотелось бы добавить, попробовать и исследовать, возможно другие технологии, возможно, оптимизировать или реализовать иным способом)

Заключение отчета должно содержать краткие выводы по результатам выполненной работы и оценку полноты решения поставленной задачи. Если работа носила научно-исследовательский

характер, то делаются выводы о практической значимости полученных результатов и области их применения; о целесообразности

продолжения работы, с указанием направлений дальнейших исследований

Список использованных источников

1. Электронная библиотека eLibrary.ru - URL: https://www.elibrary.ru/ (дата обращения 31.05.2024).
2. КиберЛенинка - URL: https://cyberleninka.ru/ (дата обращения 31.05.2024).
3. Электронная библиотека РУКОНТ - URL: https://www.rucont.ru/ (дата обращения 31.05.2024).
4. Научная электронная библиотека диссертаций Российской государственной библиотеки (РГБ) - URL: https://diss.rsl.ru/ (дата обращения 31.05.2024).
5. Электронная библиотека Московского государственного университета - URL: https://library.msu.ru/ (дата обращения 31.05.2024).
6. SkillFactory. "Глоссарий JavaScript: основные термины и понятия". - URL: <https://blog.skillfactory.ru/glossary/javascript/> (дата обращения 27.05.2024)

Приложение

Сюда по желанию можно закидывать куски кода или таблицы/данные, на которые вы можете ссылать в тексте

Сейчас это не обязательно, но можно потренироваться перед дипломом

Оформление указано в ГОСТ

Вопросы ВКР

1. Зачем это сделано?