МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Кафедра информационных систем

КУРСОВАЯ РАБОТА

Тема: Разработка веб-приложения «Система управления контентом»

Работу выполнил студент: Бабинцев Алексей Кириллович **группы** М33122

(фамилия, имя, отчество) (номер группы)

Руководитель **Приискалов Роман Андреевич** *преподаватель практики*

(фамилия, имя, отчество)

Работа защищена "_____"_____ 2022 г. с **оценкой** _____

Подписи членов комиссии: _____

_____

САНКТ-ПЕТЕРБУРГ
2022
УНИВЕРСИТЕТ ИТМО

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Бабинцев Алексей Кириллович
(Фамилия, И., О.)
Факультет Информационных технологий и программирования
Кафедра Информационных систем Группа М33122
Направление (специальность) Информационные системы и технологии
Руководитель Приискалов Роман Андреевич., Университет ИТМО, преподаватель практики.
(Фамилия, И.О., должность, ученое звание, степень)
Дисциплина Web-программирование

Наименование темы Разработка веб-приложения «Система управления контентом»

Задание Разработать веб-приложение, проанализировать и смоделировать процессы, средства автоматизации, спроектировать архитектуру информационной системы и разработать пользовательский интерфейс

Краткие методические указания В ходе выполнения работы необходимо:

1. Описать структуру разрабатываемого приложения в целом, выделить модули и конкретизировать задачи для каждого модуля (авторизация, работа с базой данных и т.д.), а также указать основные информационные объекты, которые используются в каждом модуле (модели данных, сервисы, и прочие подобные сущности фреймворка). Выбрать методологию и в соответствии с ее правилами сформировать набор диаграмм, дающих формальное описание. Сделать выводы о функциональных требованиях к средствам автоматизации со стороны смоделированных процессов. При наличии возможность описать нефункциональные требования
2. Описать типовые функциональные возможности классов информационных систем, применяющихся для автоматизации определенных на предыдущем этапе процессов, обосновать выбор конкретного набора информационных систем, детально описать их функциональные возможности и сопоставить их с функциональными требованиями, полученными на предыдущем этапе.
3. Представить функциональную и информационную архитектуры ИС, включающие все выбранные на предыдущем этапе программные средства автоматизации. Функциональная архитектура представляется как распределение операций смоделированных процессов по функциональным компонентам отдельных программных средств. В случае взаимосвязанных процессов или распределения операций одного процесса по нескольким средствам автоматизации указывается передача данных между функциональными компонентами соответствующих модулей. Информационная архитектура представляется в виде сопоставления информационных объектов, выделенных на первом этапе с информационными объектами, реализованными в выбранных средствах автоматизации. Описать интеграцию систем на уровне совместного использования преобразования данных информационных объектов, обеспечение целостности данных и синхронизации выполняемых над ними операций.

Содержание пояснительной записки

1. Определение основных понятий
2. Анализ и моделирование процессов
3. Анализ средств автоматизации процессов
4. Проектирование архитектуры ИС
5. Реализация пользовательского интерфейса

Рекомендуемая литература

1. Мартин Фаулер - Архитектура корпоративных программных приложений. Издательский дом "Вильяме'. 2006 г.
2. Флэнаган, Дэвид. JavaScript. Полное руководство, 7-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2021. — 720 с .
3. Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.
4. Браун И.Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание. — СПб.: Питер, 2021. — 336 с.
5. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 04.05.2021.

Руководитель

Подпись, дата

Студент

Подпись, дата

УНИВЕРСИТЕТ ИТМО

АННОТАЦИЯ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Бабинцев Алексей Кириллович
(Фамилия, И.О.)
Факультет Информационных технологий и программирования
Кафедра Информационных систем Группа М33122
Направление (специальность) 09.03.02 «Информационные системы и технологии»
Руководитель Приискалов Роман Андреевич., Университет ИТМО, преподаватель практики.
(Фамилия, И.О., место работы, должность, ученое звание, степень)

Дисциплина Web-программирование
Наименование темы Разработка веб-приложения «Система управления контентом»

ХАРАКТЕРИСТИКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)

1. Цель и задачи работы

☐ Предложены студентом
☐ Сформулированы при участии студента
☐ Определены руководителем

Цель: Разработать веб-приложение.
Задачи:
1) Сформировать функциональные требования к веб-приложению.
2) Проанализировать функциональные возможности и выбрать набор средств автоматизации.
3) Сформировать функциональную и информационную архитектуру решения.

2. Характер работы

☐ Расчет
☐ Моделирование
☐ Конструирование
☐ Другое, _____

4. Содержание работы

1) Определение основных понятий
2) Анализ и моделирование процессов
3) Анализ средств автоматизации процессов
4) Проектирование архитектуры ИС
5) Реализация пользовательского интерфейса

5. Выводы

Студент_____
(подпись)
Руководитель_____
(подпись)

«_____» _____2022 г.

УНИВЕРСИТЕТ ИТМО

О Т З Ы В
РУКОВОДИТЕЛЯ
о выполнении курсового проекта (работы)

Студент Бабинцев Алексей Кириллович
(Фамилия, И.О.)

Факультет Информационных технологий и программирования

Кафедра Информационных систем Группа М33122

Направление (специальность) 09.03.02 «Информационные системы и технологии»

Руководитель Приискалов Роман Андреевич, Университет ИТМО, преподаватель практики
(Фамилия, И.О., место работы, должность, ученое звание, степень)

Дисциплина Web-программирование

Наименование темы Разработка веб-приложения «Система управления контентом»

ОЦЕНКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)

№ п/п	Показатели	Оценка			
		5	4	3	0*
1.	Способность к работе с литературными источниками, справочной литературой, Интернет-ресурсами и т. п.				
2.	Использование иностранных источников				
3.	Способность к анализу и обобщению информационного материала				
4.	Владение базовыми знаниями в профессиональной области				
5.	Владение базовыми знаниями в смежных областях				
6.	Владение навыками решения технических задач				
7.	Способность применять знания на практике				
8.	Уровень и корректность использования в работе методов численного моделирования, инженерных расчетов и статистической обработки данных				
9.	Владение навыками использования современных пакетов компьютерных программ и технологий				
10.	Владение навыками оформления отчетных материалов с применением современных пакетов программ				
11.	Качество оформления пояснительной записки (общий уровень грамотности, стиль изложения, качество иллюстраций, корректность цитирования и пр.**)				
12.	Качество оформления презентации				
13.	Владение навыками публичного выступления и межперсональной коммуникации				
14.	Владение навыками планирования и управления временем при выполнении работы				
ИТОГОВАЯ ОЦЕНКА					

* - не оценивается (трудно оценить)

Отмеченные достоинства:_____

Отмеченные недостатки:_____

Заключение _____

Руководитель _____
(подпись)

Дата «_____»_____2022 г.

Оглавление

Введение

Объектом разработки является система управления контентом на языке программирования TypeScript с использованием фреймворка Nest – для серверной части приложения и языка программирования JavaScript, без использования фреймворков – для клиентской части приложения. Выбранный фреймворк один из самых поддерживаемых и распространённых.

Целью работы является разработка веб-приложения и пользовательского интерфейса, анализ требования и моделирование процессов, средств автоматизации и архитектуры информационной системы.

В ходе работы были получены следующие результаты:

- Серверная часть системы, принимающая запросы.
- Клиентская часть системы, предоставляющая интерфейс пользователя.
- База данных для хранения информации о пользователях, файлах, а также служебной информации внутри системы.

Определения, обобщения и сокращения

Браузер – прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач. В глобальной сети браузеры используют для запроса, обработки, манипулирования и отображения содержания веб-сайтов. Многие современные браузеры также могут использоваться для обмена файлами с серверами FTP, а также для непосредственного просмотра содержания файлов многих графических форматов (gif, jpeg, png, svg), аудио- и видеоформатов (mp3, mpeg), текстовых форматов (pdf, djvu) и других файлов.

Фреймворк - программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Heroku - облачная PaaS-платформа (предоставляет пользователю определённые функции и доступ к ПО, при этом её инфраструктура скрыта), поддерживающая ряд языков программирования. Данная платформа служила для развертывания нашего приложения (лабораторных работ). Она бесплатная и проста в использовании.

MVC - расшифровывается как модель-представление-контроллер. Это архитектурный шаблон (паттерн), который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения (модель), другой отвечает за внешний вид (представление), а третий контролирует работу приложения (контроллер). Такое разделение Web-приложения на части упрощает структуру приложения за счет более строго разделения его уровней. Логика пользовательского интерфейса располагается в представлении, логика ввода-вывода – в контроллере, а бизнеслогика – в модели. Достигается полное отделение логики работы приложения от представления данных. Разработчик получает полный

контроль над формируемым HTML-документом. Облегчается задача выполнения тестирования приложения.

NestJS – это MVC-подобный фреймворк для создания эффективных, масштабируемых серверных приложений Node.js. Он использует прогрессивный JavaScript, построен и полностью поддерживает TypeScript (но при этом позволяет разработчикам писать код на чистом JavaScript) и сочетает в себе элементы ООП (объектно-ориентированного программирования), ФП (функционального программирования) и ФРП (функционально-реактивного программирования). Под капотом Nest использует надежные платформы HTTP-серверов, такие как Express (по умолчанию), и при желании также может быть настроен для использования Fastify. Nest обеспечивает уровень абстракции по сравнению с этими распространенными платформами Node.js (Express/Fastify), но также предоставляет свои API-интерфейсы непосредственно разработчику. Это дает разработчикам свободу использовать множество сторонних модулей, доступных для базовой платформы.

Node.js - программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Express - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

PostgreSQL - свободная объектно-реляционная система управления базами данных.

Prisma - это инструмент, позволяющий работать с реляционными (PostgreSQL, MySQL, SQL Server, SQLite) и нереляционной (MongoDB) базами данных с помощью JavaScript или TypeScript без использования SQL (хотя такая возможность имеется). Использовалась для работы с базой данных PostgreSQL.

Аутентификация - процедура проверки подлинности, в данной работе подразумевается проверка подлинности пользователя путём сравнения введённого им логина и пароля с логином и паролем, сохранённым в базе данных пользовательских логинов.

WebSocket - протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Gateway (в NestJS) – класс, аннотированный декоратором @WebSocketGateway; не зависит от платформы, что позволяет ему быть совместимым с любой WebSocket библиотекой при создании адаптера.

DTO (Data Transfer Object) – один из шаблонов проектирования, используемый для передачи данных между подсистемами приложения. Не должен содержать какого-либо поведения.

Валидация – процесс проверки данных по критериям корректности и полезности для конкретного применения. Nest предлагает ValidationPipe как инструмент для проверки входных данных.

JWT – открытый стандарт (RFC 7519) 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 клиенты, который в дальнейшем использует данный токен для подтверждения своей личности.

Описание предметной области

Описание прикладного процесса

Разрабатываемое веб-приложение, это система управления контентом. Благодаря таким системам можно, не имея навыков программирования, и не обращаясь к веб-разработчикам, самостоятельно управлять содержимым внутри сайта.

При работе с классическими системами управления контентом можно выделить такие типовые задачи как редактирование и удаление текущих, создание новых страниц или разделов.

Формирование требований

В ходе анализа прикладного процесса был получен следующий список функциональных требований:

- Возможность добавлять, удалять и обновлять посты
- Возможность аутентификации
- Отображение всех постов
- Возможность анонимного общения между пользователями посредством чата

Нефункциональные требования

Разрабатываемая система не является публичной, поэтому основным требованием выступает её закрытость, путем обязательного прохождения процесса авторизации и аутентификации.

Так же должна быть возможность работы с приложением напрямую, через общие программные интерфейсы, описанные по спецификации OpenAPI версии не ниже 3.0

Проектирование

Используемый стек технологий

Решение создания именно веб-приложения обусловлено тем, что необходимо было обеспечить доступ к системе с любого устройства, в любое время. Веб-приложение решает этот вопрос, а также снимает вопрос обновлений на стороне клиента.

Проект использует стек стандартных технологий, характерный для большинства веб-приложений: HTML, CSS, Javascript.

В качестве веб-сервера используется Express.

В качестве базы данных используется PostgreSQL, ввиду того, что данная СУБД является самой стабильной в данной связке. Библиотекой для работы с данными была выбрана Prisma.

В проекте используется свободная распределённая система управления версиями Git, хранилищем исходных кодов является крупнейший веб-сервис GitHub, а в качестве хостинга используется облачный сервис Heroku.

Системная архитектура

Рисунок 1. Системная архитектура приложения.

Архитектура данных

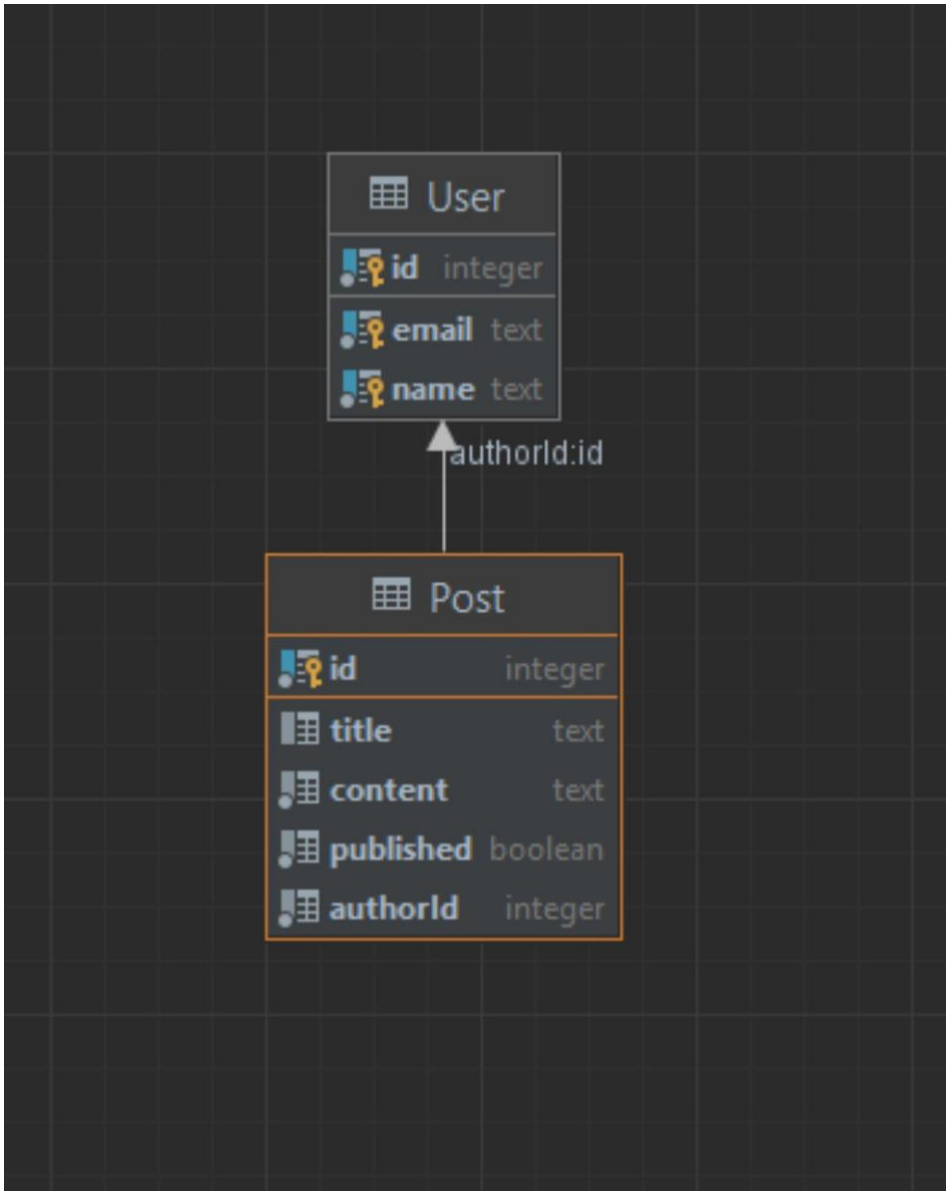

Рисунок 2. Схема таблиц базы данных

User – Предоставляет информацию о пользователе

Post – Предоставляет информацию о посте пользователя

Программная архитектура

Таблица 1. Отношения модулей и классов

Название модуля	**Название класса**	**Назначение класса**
Post	PostController	Сервис (Контроллер)
	CreatePostDto	Модель передачи данных
	PostService	Сервис (Модель)
	PostModule	Корневая точка модуля
User	UserController	Сервис (Контроллер)
	CreateUserDTO	Модель передачи данных
	UserService	Сервис (Модель)
	UserModule	Корневая точка модуля
Auth	SuperTokensService	Класс-обертка для аутентификации по JWT
	ConfigInterface	Класс-обертка над сервисом SuperTokens
	AuthGuard	Связывающее программное обеспечение (ограничитель доступа)
	AuthMiddleware	Связывающее программное обеспечение (ограничитель доступа)
	AuthFilter	Связывающее программное обеспечение (фильтр исключений)
	SessionDecorator	Связывающее программное обеспечение (проверка сессий)
	AuthModule	Корневая точка модуля

WebSocket	AppGateway	Класс-обертка

Таблица 2. Описание классов

Название класса	**Описание класса**
PostController	Класс, обрабатывающий входящие запросы (показать все посты; создать пост; удалить пост; показать пост, для определенного пользователя; изменить пост) и возвращающий ответы.
CreatePostDTO	Класс, инкапсулирующий и валидирующий данные (название поста; содержимое поста; статус публикации поста) при передаче из одной подсистемы в другую.
PostService	Класс, используемый для работы с постами. Осуществляет поиск постов, отображение всех постов, добавить и удалить пост, а также изменить уже существующий пост.
PostModule	Класс, использующийся фреймворком Nest для организации внутренней структуры модуля.
UserController	Класс, обрабатывающий входящие запросы (добавление пользователя и просмотр пользователя) и возвращающий ответы.
CreateUserDTO	Класс, инкапсулирующий и валидирующий данные (имя пользователя; электронный адрес пользователя) при передаче из одной подсистемы в другую.
UserService	Класс, используемый для работы с пользователями. Осуществляет поиск пользователей и их добавление.
UserModule	Класс, использующийся фреймворком Nest для организации внутренней структуры модуля.
SuperTokensService	Класс-обертка, для реализации аутентификации по JWT.

ConfigInterface	Класс, служащий оберткой над классом (сервисом) Super Tokens из платформы Super Tokens.
AuthGuard	Класс, который в зависимости от того авторизован пользователь или нет возвращает true или false.
AuthMiddleware	Класс, чей основной функционал вызывается перед обработчиком маршрута и отфильтровывает неавторизованных пользователей.
AuthFilter	Класс, служащий как фильтр исключений и обрабатывает ошибки.
SessionDecorator	Класс, возвращающий сеанс, который прикреплен к запросу.
AuthModule	Класс, использующийся фреймворком Nest для организации внутренней структуры модуля.
AppGateway	Класс, обслуживающий WebSocket канал (обертка над WebsocketServer в Nest)

Разработка

Реализация серверного API

В качестве описания программного интерфейса был выбран инструмент, поддерживающий стандарт OAS 3.0 – Swagger. Далее представлена полученная документация API полученная автоматически по директивам, указанным в декораторах различных методов и структурах данных внутри разрабатываемой информационной системы.

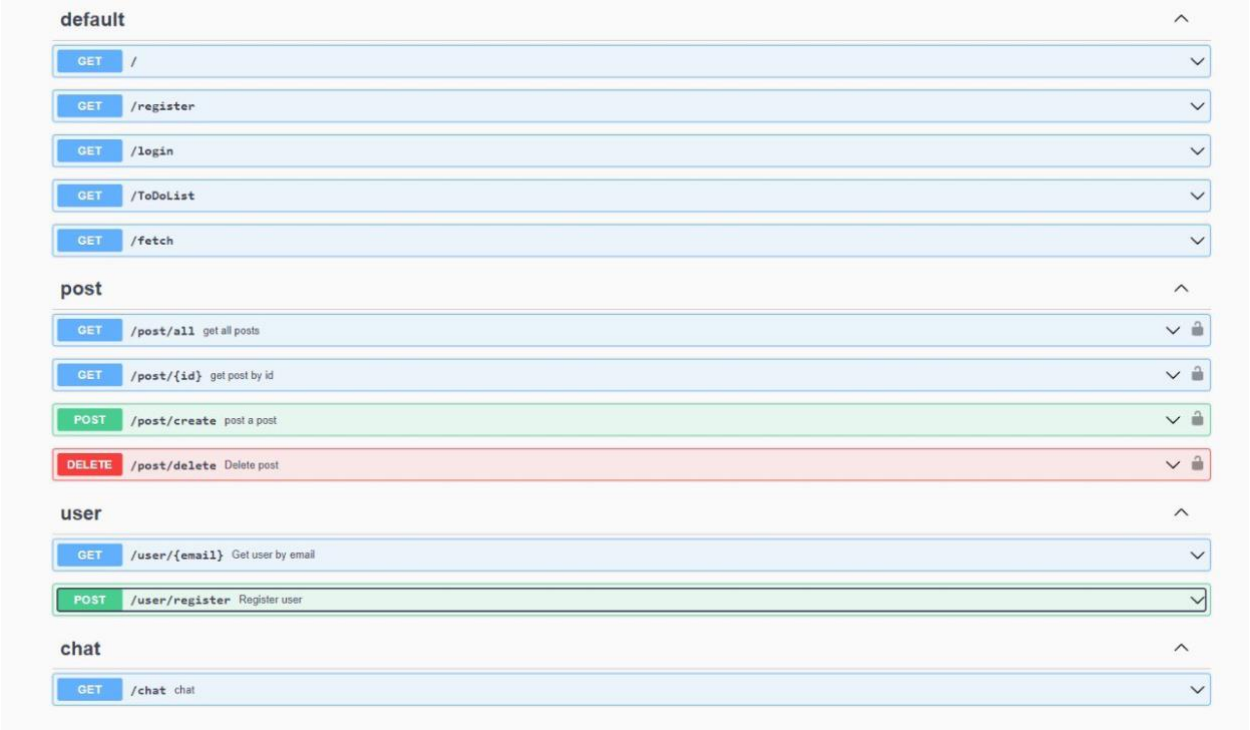

Рисунок 3.1. Программный интерфейс серверного API.

Рисунок 3.2. Программный интерфейс серверного API.

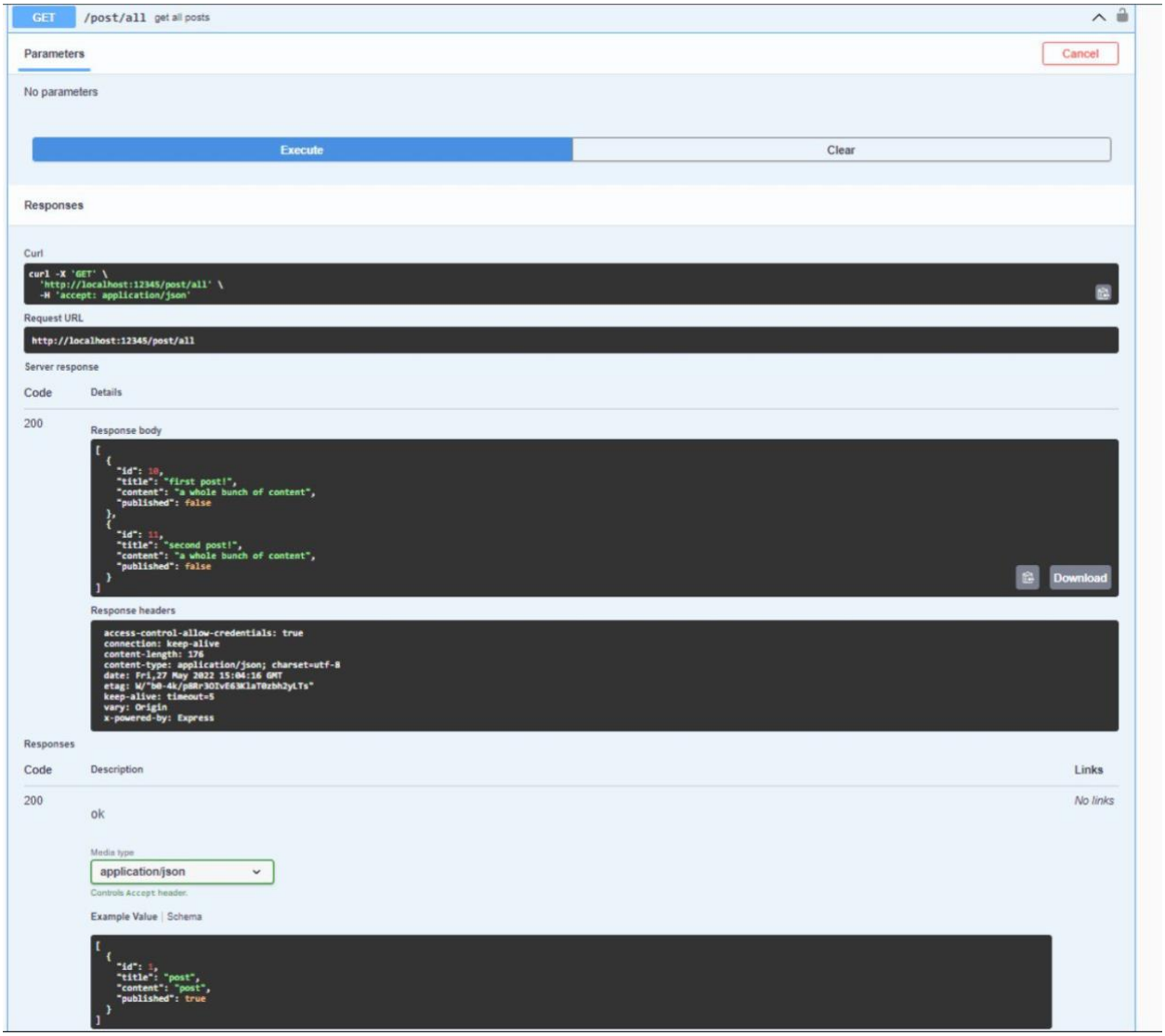

Рисунок 3.3. Программный интерфейс серверного API.

Реализация пользовательского интерфейса

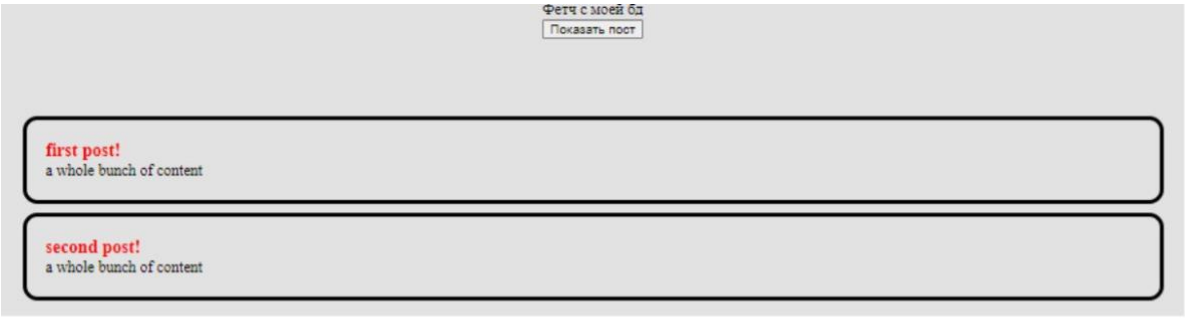

Рисунок 4. Пользовательский интерфейс взаимодействия с разделом постов.

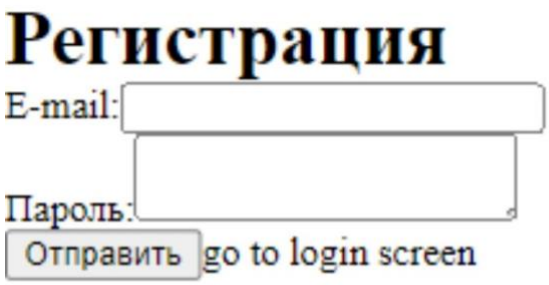

Рисунок 5. Пользовательский интерфейс (регистрация пользователя).

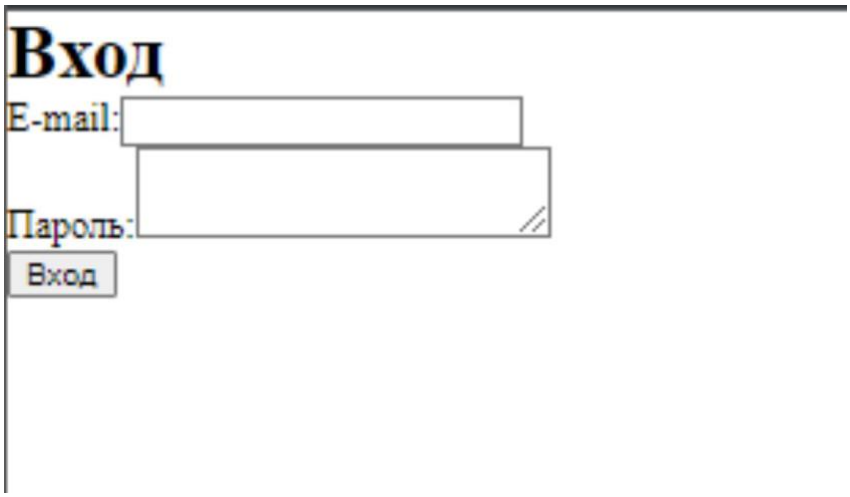

Рисунок 6. Пользовательский интерфейс авторизации (вход в аккаунт).

Рисунок 7. Пользовательский интерфейс авторизации.

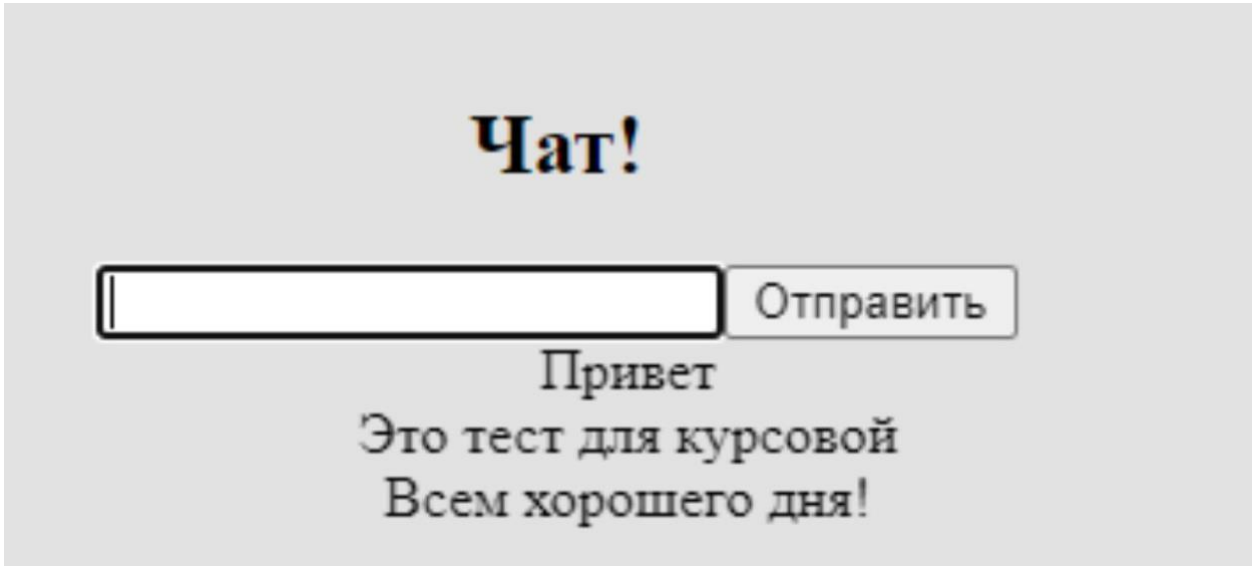

Рисунок 8. Пользовательский интерфейс чата.

Заключение

В ходе выполнения курсовой работы был проведён анализ работы классических систем управления сайтами, исходя из которого были выявлены и сформированы требования к разрабатываемому веб-приложению.

Исходя из выбранной архитектуры и наложенных ограничений были сформированы требования к используемым технологиям внутри модулей. Была спроектирована архитектура данных, программная и системная архитектура в виде набора диаграмм в нотации UML.

Опираясь на выше изложенные требования и стек технологий было разработано веб-приложение и пользовательский интерфейс в рамках дисциплины «Web-программирование».

Таким образом, все поставленные ранее цели были выполнены.

Разработанное приложение является результатом данной курсовой работы.

Список использованной литературы

1. Мартин Фаулер - Архитектура корпоративных программных приложений. Издательский дом "Вильяме'. 2006 г.

2. Флэнаган, Дэвид. JavaScript. Полное руководство, 7-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2021. — 720 с .

3. Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.

4. Браун И.Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание. — СПб.: Питер, 2021. — 336 с.

5. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 04.05.2021.