Министерство образования и науки РФ Федеральное государственное автономное образовательное учреждение высшего образования

**Национальный исследовательский технологический университет**

**«МИСИС»**

Институт информационных технологий и компьютерных наук (ИТКН)

Кафедра автоматизированных систем управления (АСУ)

**КУРСОВАЯ РАБОТА**

по дисциплине: «Разработка клиент-серверных приложений»

на тему: «Форум, пользователь, посты, комментарии»

Выполнил:

студент группы БИВТ-21-10

Рустамов С. М.

Проверил:

Рзазаде Ульви Азар оглы

Москва, 2023

**Оглавление**

1. Предметная область – 3
2. Постановка задачи – 3
3. Описание архитектуры – 5
4. Описание структуры БД – 9
5. Описание серверной части – 10
6. Описание клиентской части – 11
7. Заключение – 12
8. Список литературы – 13

**Предметная область**

Форум представляет собой предметную область, в которой пользователи могут создавать посты и оставлять комментарии. Всё это происходит для обмена мыслями, идеями, и прочим.

Проблемы данной предметной области могут заключаться в следующем:

1. Каждый форум, ровно как и любое клиент-серверное приложение, нуждается в надёжной системе авторизации, аутентификации и учёта. Это необходимо для обеспечения безопасности пользователей форума и их личных данных.
2. Необходимость в ведении учёта постов и комментариев. Решение данной проблемы обеспечит хорошее времяпрепровождение пользователей на форуме.
3. Необходимость в частичном ограничении неавторизированных пользователей от части доступных на сервере данных.

**Постановка задачи**

По варианту номер 22, «Форум, пользователи, посты, комментарии», требуется создать клиент-серверное приложение с достаточным для пользования функционалом, включающее в себя:

* Возможность оставлять комментарии;
* Возможность производить поиск по пользователям, комментариям и постам основываясь на разных запросах;
* Функция регистрации новых пользователей с соблюдением уникальности почты пользователя;
* Возможность отображать на страницах сайта только ту информацию, к которой данный пользователь имеет доступ.

Каждая из упомянутых проблем непременно требует решений. Я стремился разработать свое клиент-серверное приложение таким образом, чтобы пребывание пользователей на форуме прошло наилучшим образом. Для решения этих проблем, я прибег к следующим решениям:

1. Реализовал систему авторизации, регистрации и аутентификации в приложении. Я добился данного результата при помощи гвардов и пайпов.
2. Ведение учета постов и комментариев обеспечивается базой данных, созданной специально под данное приложение, и реляционными связями между ними.
3. Частичное ограничение неавторизированных пользователей от информации происходит при помощи DTO, отдельной от DTO для авторизированных пользователей.

Помимо этого, приложение должно уметь создавать таблицы в реляционной БД для хранения информации о постах, пользователях и комментариях.

В рамках моей курсовой работы я разработал концепцию клиент-серверного приложения, которое основано на использовании PostgreSQL, NestJS и TypeORM для реализации функциональности форума, пользователей, постов и комментариев. В процессе разработки, я стремился создать эффективное и масштабируемое решение, которое позволит пользователям активно общаться и делиться мнениями на форуме. Использование PostgreSQL в качестве базы данных, NestJS для серверной части и TypeORM для взаимодействия с данными позволит создать платформу, на которой пользователи смогут создавать посты, оставлять комментарии и обмениваться информацией на интересующие их темы.

**Описание архитектуры**

1. **Клиент-серверные приложения**

Клиент-серверное приложение – это веб-приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле)[3].

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций).

Практические реализации такой архитектуры называются клиент-серверными технологиями.

Например, одной из них является двухзвенная архитектура – распределение трех базовых компонентов между двумя узлами (клиентом и сервером)[3]. Двухзвенная архитектура, схема которой показана на рисунке 1 используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме.

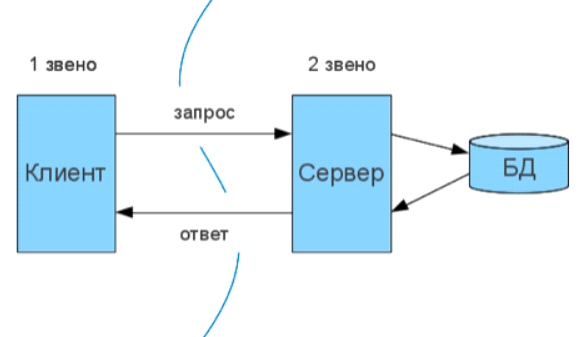


Рисунок 1. Схема двухзвенной архитектуры

При этом стоит сразу заметить, что база данных фактически не является частью веб-сервера, но большинство приложений просто не могут выполнять все возложенные на них функции без нее, так как именно в базе данных хранится вся динамическая информация приложения.

База данных – это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать. Базы данных функционируют под управлением так называемых систем управления базами данных. Самыми популярными СУБД являются MySQL, MS SQL Server, PostgreSQL, Oracle (все – клиент-серверные).

1. **Выбор технических средств**

На основании задач, поставленных передо мной, я решил использовать объектно-ориентированный язык typescript и фреймворк Nest. В работе я использовал следующие компоненты:

* TypeORM – необходима для работы с базами данных и создания схемы БД
* Nestjs-form-data – формат передачи данных между клиентским и серверным приложением
* Dotenv – переменные окружения
* Crossenv – переменные окружения
* Swagger – утилита для тестирования и автодокументирования api запросов
* Pg – бибилиотека для работы с БД postgresql
* Argon2 – использовался для хэширования паролей

Разработанное клиент-серверное приложение представляет собой систему управления магазином. Архитектура приложения основана на классическом подходе клиент-серверного взаимодействия, где клиентская часть обеспечивает пользовательский интерфейс и взаимодействие с сервером, а серверная часть обрабатывает запросы от клиентов, управляет данными и выполняет бизнес-логику.

Основные сущности приложения:

Пользователи:

1. Сущность "Пользователь" хранит информацию о пользователях, включая имя, никнейм, почту и пароль.

2. DTO "UserDTO" содержит поля "name", "user\_name", "email" и "password" для передачи данных о продукте между клиентом и сервером.

3. Модуль "Пользователи" отвечает за управление информацией о пользователях, включая добавление, изменение и удаление продуктов.

4. Сервис "Пользователи" обеспечивает операции CRUD (создание, чтение, обновление, удаление) для пользователей.

Посты:

1. Сущность "Посты" хранит информацию о постах на форуме, включая ник автора, тему поста, и сам текст.

2. DTO "PostDTO" содержит поля "id", "name", "topic " и "text" для передачи данных о посте между клиентом и сервером.

3. Модуль "Посты" отвечает за управление информацией о постах, включая добавление, изменение и удаление постов.

4. Сервис "Посты" обеспечивает операции CRUD для постов.

Комментарии:

1. Сущность "Комментарий" хранит информацию о комментарии, включая его текст и автора.

2. DTO "CommentDTO" содержит поля "text" и "user\_name" для передачи данных при создании комментария.

3. Модуль "Комментарии" отвечает за управление информацией о комментариях, включая добавление, изменение и удаление комментариев.

4. Сервис "Комментарии" обеспечивает операции добавления, изменения и удаления комментариев.

Архитектура приложения предусматривает взаимодействие клиентской части с серверной частью, построенной с использованием фреймворка NestJS и базы данных PostgreSQL. Клиентская часть предоставляет пользовательский интерфейс, позволяющий пользователям просматривать информацию о постах, комментариях и пользователях, а также выполнять операции управления данными. Серверная часть обрабатывает запросы от клиентов, выполняет необходимые операции с данными, взаимодействует с базой данных и предоставляет клиентам необходимую информацию.

Благодаря данной архитектуре приложение обеспечивает удобный интерфейс для клиентов магазина, позволяет эффективно управлять информацией о продуктах, работниках и пользователях, а также обеспечивает надежное хранение данных и высокую производительность при обработке запросов от пользователей.

**Описание структуры БД**

База данных – Форум.

Сущности:

1. Посты
2. Комментарии
3. Пользователи.

Рассмотрим каждую их них подробно:

Посты:

1. Первичным ключом выступает поле id [number]
2. Name: имя автора поста [string]
3. User\_name: никнейм автора поста [string]
4. Topic: тема поста (ограничен в 40 символов) [string]
5. Text: основной текст поста (ограничен в 2800 символов) [string]

Образована связь OneToMany с сущностью комментариев и связь ManyToOne с сущностью пользователей.

Комментарии:

1. Первичным ключом выступает поле id [number]
2. Name: имя автора комментария [string]
3. User\_name: никнейм автора комментария [string]
4. Text: текст комментария (ограничен в 1400 символов) [string]

Образована связь ManyToOne с сущностью пользователей.

Пользователи:

1. Первичным ключом выступает поле id [number]
2. Name: имя пользователя [string]
3. User\_name: никнейм пользователя [string]
4. Email: электронная поста пользователя [string]
5. Password: зашифрованный пароль пользователя [string]
6. About\_me: короткое описание пользователя [string]

Образована связь OneToMany с сущностью комментариев и связь OneToMany с сущностью постов.

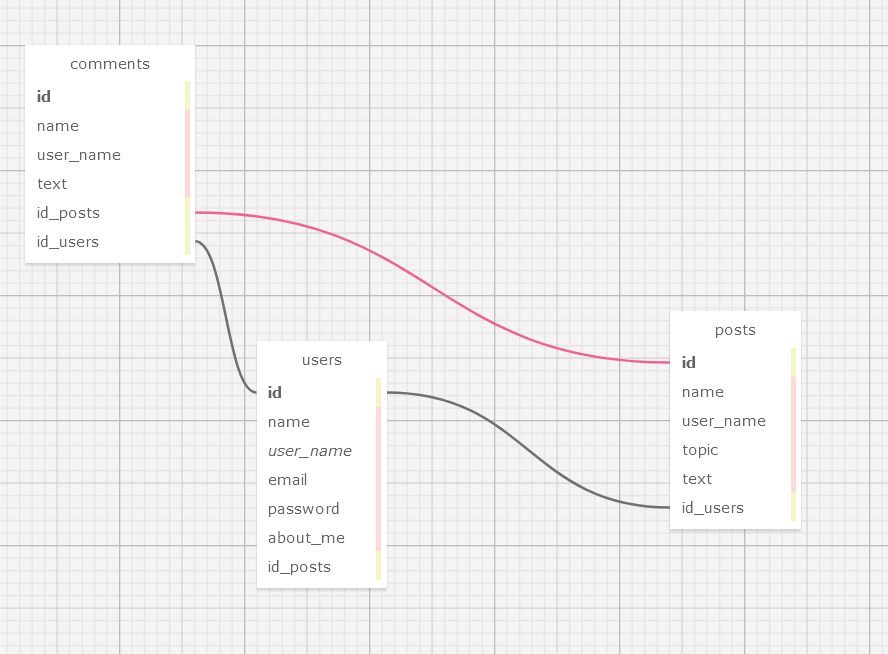


Рисунок 2. Реляционные связи БД

**Описание серверной части**

Серверная часть приложения "Форум" построена на архитектурном шаблоне MVC (Model-View-Controller) и включает в себя следующие основные компоненты:

Контроллеры (Controllers):

PostsController: отвечает за обработку запросов, связанных с операциями над постами (например, создание, чтение, обновление, удаление постов).

UserController: обрабатывает запросы, связанные с операциями над пользователями (например, создание, чтение, обновление, удаление пользователей).

CommentController: отвечает за обработку запросов, связанных с операциями над комментариями (например, создание, чтение, обновление, удаление комментариев).

Модули (Modules):

PostsModule: модуль, связанный с постами. Включает в себя сервис PostsService и контроллер PostsController. Также имеет зависимость от TypeOrmModule для работы с базой данных и включает сущность Posts.

UserModule: модуль, связанный с пользователями. Включает в себя сервис UserService и контроллер UserController. Также имеет зависимость от TypeOrmModule для работы с базой данных и включает сущность User.

CommentModule: модуль, связанный с комментариями. Включает в себя сервис CommentService и контроллер CommentController.

Каждый контроллер обрабатывает соответствующие запросы, вызывая соответствующие методы сервисов. Модули импортируют необходимые зависимости, объединяют сервисы и контроллеры, и экспортируют их для использования в других частях приложения.

**Описание клиентской части**

Клиентская часть приложения представлена в виде спецификации API, предоставляемого приложением "Форум ". Ниже приведены описания API-маршрутов, связанных с сущностью "Users", основанные на вашем контроллере:

Получение списка работников:

1. Endpoint: GET /api/users

2. Описание: Получить список всех пользователей.

3. Параметры запроса: Отсутствуют.

4. Ответ: JSON-массив с информацией о каждом пользователе.

Создание работника:

1. Endpoint: POST /api/users

2. Описание: Создать нового пользователя.

3. Параметры запроса: JSON-объект с информацией о новом пользователе.

4. Ответ: JSON-объект с информацией о созданном пользователе.

Получение информации о конкретном работнике:

1. Endpoint: GET /api/users/{id}

2. Описание: Получить информацию о пользователе с заданным идентификатором.

3. Параметры запроса: Идентификатор пользователя.

4. Ответ: JSON-объект с информацией о пользователе.

Удаление работника:

1.Endpoint: DELETE /api/users/{id}

2. Описание: Удалить пользователя с заданным идентификатором.

3. Параметры запроса: Идентификатор пользователя.

4. Ответ: Пустой ответ.

При создании пользователя используется валидация данных с помощью ValidationPipe.

**Заключение**

Предметная область данного проекта включает в себя форум, ведущий учёт по действиям пользователей. Проблемами, рассматриваемыми в проекте являются управление данными форума, потребность в валидации и регистрации.

Для решения данной проблемы было разработано клиент-серверное приложение. Серверная часть приложения была построена с использованием фреймворка Nest.js и базы данных PostgreSQL. Была создана структура БД, включающая сущности "Posts", "User" и "Comment", а также связи между ними.

В рамках курсовой работы было создано клиент-серверное приложение на тематику «Форум», что и является главным результатом проекта.

**Список литературы**

1. С. А. Орлов «Программная инженерия. Технологии разработки программного обеспечения». СПб: Санкт-Петербург, 2016–640 с.
2. Daniel Jacobson, Greg Brail, Dan Woods. APIs: A Strategy Guide. 2011.
3. Моисеев Антон, Яков Файн. TypeScript быстро. 2021.
4. Danny Adams. Learn TypeScript – The Ultimate Beginners Guide [Электронный ресурс]. – Режим доступа: <https://www.freecodecamp.org/news/learn-typescript-beginners-guide/>, свободный. – (дата обращения: 16.05.2023).
5. Clara Ekekenta. Full-stack app tutorial with NestJS and React [Электронный ресурс]. – Режим доступа: <https://blog.logrocket.com/full-stack-app-tutorial-nestjs-react/>, свободный. – (дата обращения: 16.05.2023).
6. Documentation | NestJS [Электронный ресурс]. – режим доступа: <https://nestjs.ru/security/authentication>, свободный. – (дата обращения: 29.05.2023).