Федеральное государственной образовательное учреждение

Высшего профессионального образования

Национальный исследовательский технологический университет «МИСиС»

Институт информационных технологий и компьютерных наук (ИТКН)

Курсовая работа

по курсу «Разработка клиент-серверных приложений»

Тема:

«Спортивный клуб»

Выполнила: студентка 2 курса,

группа БИВТ-21-6

Зайцева Дарья

Руководитель:

Рзазаде У. А.

МОСКВА 2023

**Оглавление**

[1. Введение 3](#_Toc136229819)

[2. Постановка задачи 4](#_Toc136229820)

[3. Описание используемых технических средств разработки 6](#_Toc136229821)

[4. Описание архитектуры проекта 8](#_Toc136229822)

[5. Описание структуры базы данных проекта 9](#_Toc136229823)

[6. Описание серверной части проекта 11](#_Toc136229824)

[7. Заключение 16](#_Toc136229825)

[8. Список литературы 17](#_Toc136229826)

# Введение

В наше время клиент-серверное приложение с базой данных для спортивного клуба является полезным инструментом для эффективного управления и организации деятельности клуба.

Доступ к информации становится удобным: приложение позволяет клиентам получать удобный и мгновенный доступ к необходимой им информации.

База данных позволяет хранить и управлять информацией о клиентах, позволяет создавать и управлять группами клиентов, назначать им тренеров

В целом, клиент-серверное приложение и база данных спортивного клуба помогает в организации и эффективном управлении различными аспектами деятельности клуба. Она облегчает работу с клиентами, тренерами, дисциплинами и группами, улучшает планирование и отчетность, а также способствует повышению качества предоставляемых услуг.

Клиент-серверные приложения играют важную роль в эффективной обработке данных и управлении ресурсами. Они позволяют эффективно распределить нагрузку между клиентскими и серверными компонентами, что обеспечивает более быстрое и надежное выполнение операций. Клиентская часть приложения обрабатывает пользовательский интерфейс и взаимодействует с сервером, который, в свою очередь, обеспечивает доступ к базе данных и другим ресурсам.

Одним из преимуществ клиент-серверных приложений является централизованное управление данными. Серверная часть приложения хранит и обрабатывает данные, что обеспечивает их целостность и безопасность. Клиенты получают доступ к данным через сервер, что позволяет контролировать доступ и обеспечивать защиту данных от несанкционированного доступа.

Таким образом, клиент-серверные приложения обеспечивают более эффективное управление данными, повышенную безопасность и надежность выполнения операций, что делает их незаменимыми инструментами для спортивных клубов и других организаций, требующих централизованного управления и обработки данных.

# Постановка задачи

В данной курсовой работе дана следующая задача: разработка клиент-серверного приложения для централизованного управления спортивным клубом. Для успешного выполнения этой задачи необходимо решить следующие подзадачи:

Для реализации клиент-серверного приложения использованы следующие подходы и технологии:

1. Разработка серверной части:
   * Использование фреймворков, сред выполнения для разработки серверной части: Node.js, Nest.js.
   * Определение маршрутов API для обработки запросов от клиентской части.
   * Реализация базы данных для хранения информации о клиентах, тренерах, дисциплинах и группах. Использование реляционной базы данных PostgreSQL.
   * Разработка логики обработки запросов, включая создание, чтение, обновление и удаление данных с использованием ORM (Object-Relational Mapping).
2. Обеспечение безопасности:
   * Реализация системы аутентификации и авторизации пользователей с использованием токенов.
   * Ограничение доступа к определенным функциональным возможностям приложения в зависимости от роли пользователя.
3. Тестирование:
   * Разработка и выполнение модульных тестов для проверки работоспособности отдельных компонентов приложения.
   * Выполнение интеграционных тестов для проверки взаимодействия между клиентской и серверной частями.
4. Документирование и развертывание:
   * Необходимо оформить документацию по разработке приложения, включая описание требований, архитектуры, базы данных и тестирования.

При разработке приложения важно также учитывать масштабируемость и поддерживаемость кода, чтобы в будущем можно было легко расширять функциональность и вносить изменения в систему.

# Описание используемых технических средств разработки

Для выполнения задания, указанного в детальных требованиях, мы выбрали

следующие технологии:

1. TypeScript: Язык программирования, являющийся надмножеством JavaScript.

TypeScript обеспечивает более структурированное и понятное написание кода и

предоставляет статическую типизацию, что помогает выявлять ошибки на этапе

компиляции. Он является основным инструментом для разработки клиент-

серверных приложений.

2. NestJS: Фреймворк для разработки серверных приложений на TypeScript. Он

предоставляет разработчикам множество инструментов и функций,

позволяющих быстро создавать высокопроизводительные и масштабируемые

приложения. NestJS использует модульную архитектуру и предоставляет

интеграции с другими технологиями, такими как TypeORM и PostgreSQL.

3. TypeORM: ORM (Object-Relational Mapping) для TypeScript и JavaScript,

который позволяет разработчикам работать с базами данных, используя

объектно-ориентированный подход. Он предоставляет функции, такие как

миграции баз данных и связи между таблицами. TypeORM интегрируется с

NestJS и PostgreSQL, что делает его идеальным выбором для разработки

складского учета.

4. PostgreSQL: Реляционная база данных с открытым исходным кодом,

обладающая высокой производительностью и надежностью. Она используется

для хранения данных, связанных со складским учетом, таких как информация о

продуктах, заказах и поставщиках. PostgreSQL обладает множеством функций и

поддерживает многопоточность, что делает его идеальным выбором для

разработки приложений с большим объемом данных.

5. Git: Система контроля версий, позволяющая разработчикам отслеживать

изменения в коде, вносить исправления и улучшения, а также сотрудничать в

команде. Git является одним из наиболее популярных инструментов для

управления исходным кодом и используется при разработке клиент-серверных

приложений.

6. Node.js: Платформа, основанная на JavaScript, используемая для выполнения

серверного кода. Node.js позволяет разработчикам создавать серверные

приложения с использованием JavaScript, что обеспечивает единый язык

программирования на клиентской и серверной сторонах. Он обладает высокой

производительностью и масштабируемостью, что делает его идеальным

выбором для разработки клиент-серверных приложений и взаимодействия с

базой данных и другими ресурсами. Node.js в сочетании с выбранными

технологиями, такими как TypeScript, NestJS и TypeORM, обеспечит

эффективное развертывание и функционирование серверной части приложения

для автоматизации складского учета товаров.

# Описание архитектуры проекта

Архитектура проекта для курсовой работы по разработке клиент-серверных приложений на тему "Спортивный клуб" базируется на использовании TypeScript, PostgreSQL, Git, NestJS, TypeORM и NodeJS.

Для клиентской части приложения используется фреймворк NestJS, который позволяет создавать масштабируемые и модульные приложения на NodeJS.

Для работы с базой данных PostgreSQL используется ORM TypeORM, который позволяет работать с данными в объектно-ориентированном стиле.

В проекте используется Git для контроля версий, что позволяет разработчикам работать с кодом в команде и контролировать изменения.

Архитектура проекта основана на принципе MVC (Model-View-Controller), который позволяет разделить приложение на три основных компонента: модель, представление и контроллер.

Модель представляет собой объекты, которые отображают данные в базе данных. Представление отвечает за отображение данных на клиентской стороне приложения. Контроллер является посредником между моделью и представлением, управляя логикой приложения и обрабатывая запросы от клиента.

Приложение разбито на модули, что позволяет разделить его на логические части и упростить масштабирование.

В целом, архитектура проекта базируется на использовании современных технологий и принципов, что позволяет создать эффективное и масштабируемое приложение для управления спортивным клубом.

# Описание структуры базы данных проекта

Структура базы данных для спортивного клуба содержит четыре таблицы: clients, trainers, disciplines, groups.

Таблица clients содержит информацию о клиентах, посещающих спортивный клуб. Таблица включает их уникальный идентификатор (id), имя пользователя (fullname), их пол (gendre) и дату рождения (birthday).

Таблица trainers содержит информацию о тренерах, включая их уникальный идентификатор (id), полное имя (fullname), количество рабочих часов (workhours) и зарплату (salary).

Таблица disciplines содержит информацию о спортивных дисциплинах клуба, включая их уникальный идентификатор (id), название (name) и уровень сложности (level).

Таблица groups содержит информацию о группах, включая их уникальный идентификатор (id) и название (name)

Также были созданы join-таблицы со связью многие-ко-многим: client\_discipline для того, чтобы видеть на какие дисциплины записан клиент; client\_group для распределения клиентов по группам; trainer\_discipline для распределения дисциплин для тренеров; trainer\_group для распределения групп для тренеров.

Ниже (рис.1) можно увидеть базу данных для спортивного клуба и связь между таблицами.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис.1 База данных для спортивного клуба

Таким образом, структура базы данных для спортивного клуба позволит удобно хранить и обрабатывать информацию о клиентах, тренерах, дисциплинах и группах.

# Описание серверной части проекта

Серверная часть приложения представляет собой RESTful API, которое обрабатывает запросы от клиентской части и взаимодействует с базой данных PostgreSQL.

Архитектура серверной части проекта представлена на рисунке 2.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рис. 2. Архитектура сервера

Также для удобства контроля и отслеживания запросов была использована документация Swagger UI, представленная на рисунках 3-4.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок 3. Документация Swagger UI

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 4. Документация Swagger UI

В приложении реализованы следующие функции: просмотр списка тренеров, клиентов, дисциплин, групп, их выборка по уникальному идентификатору, возможность осуществлять CRUD-операции. Для каждой из этих функций разработаны соответствующие контроллеры и сервисы.

Для примера, в папке clients, как видно на рисунке 5, реализованы сервис, контроллер и модуль. И entity, которая описывает как выглядит сущность (в БД). Внутри clients.dto реализованы дополнительные возможности передачи данных (исходя из нужд конечного клиента).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5. Содержимое папки clients

База данных PostgreSQL содержит таблицы для хранения информации о клиентах, тренерах, дисциплинах и группах. TypeORM используется для создания таблиц и связей между ними, а также для выполнения запросов к базе данных. На рисунке 6. Представлен результат выполнения этих запросов.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 6. Результат выполнения запроса к серверу

Также реализованы аутентификация и авторизация, что обеспечивает безопасность.

Была выполнена JWT авторизация на NestJS с использованием Passport.js. JWT используется для создания токенов авторизации, которые могут быть переданы между клиентом и сервером для аутентификации пользователя.

А авторизация реализована на основе разделения ролей (RBAC implementation)

Работу авторизации и аутентификации можно увидеть в swagger на рисунках 7-8.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7. Выдача токена

Изображение выглядит как текст, снимок экрана, Прямоугольник

Автоматически созданное описание

Рисунок 8. Получение ролей (авторизация) на основе токена

В целом, серверная часть проекта "Спортивный клуб" представляет собой надежную и масштабируемую систему, которая позволяет управлять спортивным клубом с помощью интерфейса.

# Заключение

Разработка клиент-серверных приложений для спортивного клуба является важным шагом в упрощении и автоматизации процессов управления спортивным клубом. Эти приложения позволяют быстро и эффективно обрабатывать информацию, относящуюся к клиентам, тренерам, дисциплинам и группам.

При разработке таких приложений необходимо учитывать различные факторы, в том числе надежность и безопасность передачи данных, простоту использования для конечных пользователей и масштабируемость для будущего роста.

Для этого использовались современные технологии, такие как базы данных и клиент-серверная архитектура, а также программные библиотеки для реализации функциональности приложений. Было проведено тщательное тестирование для обеспечения высокой производительности и надежности приложения.

В целом разработанное приложение подходит для использования в различных спортивных клубах и подобных организациях, так как оптимизирует рабочие процессы, повышает эффективность и упрощает управление.

# Список литературы

1. Карпов А.А. "Разработка клиент-серверных приложений на языке TypeScript". Москва: ДМК Пресс, 2019.
2. Корнелюк А. "PostgreSQL. Проектирование и разработка баз данных". Москва: ДМК Пресс, 2019.
3. Чакрыгин М. "Git. Система контроля версий". Москва: ДМК Пресс, 2019.
4. Кузнецов В. "NestJS. Разработка масштабируемых веб-приложений на Node.js". Москва: ДМК Пресс, 2020.
5. Козлов А. "TypeORM. Разработка баз данных на TypeScript". Москва: ДМК Пресс, 2020.
6. Фримен А., Робсон Э. "Node.js. Полное руководство". Москва: ДМК Пресс, 2019.
7. Макфарланд А., Бутстрап Т. "Bootstrap 4. Руководство разработчика". Москва: ДМК Пресс, 2018.
8. Крылов Д.А. "Fullstack React: разработка с использованием React 16 и Node.js". Москва: Питер, 2020.

Ссылка на гит-репозиторий: https://github.com/newmockingjay/couscovaya