МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П. М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра информационных технологий и управления бизнесом

КУРСОВОЙ ПРОЕКТ

по дисциплине «Технологии взаимодействия с базами данных»

РАЗРАБОТКА ИНФОРМАЦИОНной Системы «АВТОБАЗА» с

использованием Nestjs

.

Царенко Антон Дмитриевич,

3 курс, 34 группа

Научный руководитель:

Адаменко Наталья Дмитриевна,

доцент кафедры информационных технологий и управления бизнесом,

кандидат педагогических наук

Витебск, 2022

**Реферат**

Курсовая работа 16 стр., 5 рис., 8 листингов, 6 источников

TYPESCRIPT, NESTJS, POSTGRESQL, REST, BACKEND, TYPEORM.

**Объект исследования** – технологии для разработки приложений с использованием платформы NestJS.

**Предмет исследования** – алгоритмы проектирования и разработки приложений с использованием платформы NestJS.

**Цель работы** – разработка информационной системы «Автобаза».

**Задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере разработки серверных приложений.
2. Ознакомиться с особенностями реализации приложения на основе NestJS (язык программирования TypeScript).
3. Подготовить план разработки приложения.
4. Разработать приложение.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем, общенаучные методы исследования (описание, анализ и т.д.).

**Теоретическая значимость и элементы новизны**: были проанализированы публичные источники информации, уже существующие аналоги разрабатываемых систем и алгоритмов.

**Практическая значимость**: созданная backend часть приложения может быть использована для дальнейшего расширения приложения.

Содержание

[Введение 4](#_Toc121410656)

[1 Обоснование выбора средства реализации и его краткая характеристика 5](#_Toc121410657)

[2 Общая характеристика и функционал разрабатываемого приложения. Этапы разработки 6](#_Toc121410658)

[2.1 Общая характеристика и функционал разрабатываемого приложения 6](#_Toc121410659)

[2.2 Описание этапов разработки 6](#_Toc121410660)

[Заключение 15](#_Toc121410661)

[Список использованных источников 16](#_Toc121410662)

Введение

**Цель работы** – разработка backend приложения, реализующего информационную систему «Автобаза».

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере создания серверных backend приложений.
2. Ознакомиться с особенностями разработки приложения на основе NestJS.
3. Подготовить схему базы данных.
4. Составить план разработки приложения и подготовить необходимые шаблоны и библиотеки.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем разработки редакторов, общенаучные методы исследования (описание, анализ, классификация, сравнение, аналогия).

Работа включает введение, 2 главы, заключение, список использованных источников. В первой главе предоставлено обоснование выбора средства и платформы для разработки, а также языка программирования. Во второй главе приводится общая характеристика, функционал приложения и описание этапов разработки.

1 Обоснование выбора средства реализации и его краткая характеристика

NestJS – это платформа для создания серверных приложений Node.js. NestJS построен с использованием TypeScript и полностью поддерживает его.

Использование NestJS дает следующие преимущества:

1. Проверки на типы данных (TypeScript – строго типизированный);
2. Декораторы;
3. Классические импорты TypeScript – ускорение разработки, возможность писать большой код командой;
4. Четкое разделение функциональной нагрузки по элементам.

Плюсы NestJS:

1. Легко использовать, учиться и осваивать;
2. Подробная и отлаженная документация с примерами;
3. Открытый исходный код;
4. Простые приложения для модульного тестирования.

NestJS предоставляет готовую архитектуру приложений, которая позволяет разработчикам и командам создавать хорошо тестируемые, масштабируемые, слабо связанные и легко обслуживаемые приложения.

В качестве СУБД будет использован PostreSQL. PostgreSQL — это система управления объектно-реляционными базами данных (ORDBMS), разработанная в Калифорнийском университете на факультете компьютерных наук Беркли.

Для тестирования приложения будет использоваться Postman. Postman является средой разработки для создания, тестирования, контроля и публикации API-документации. Назначение Postman — тестирование отправки запросов с клиента на сервер и получения ответа от сервера.

2 Общая характеристика и функционал разрабатываемого приложения. Этапы разработки

2.1 Общая характеристика и функционал разрабатываемого приложения

Целью разрабатываемого приложения является создание информационной системы «Автобаза».

Информационная система будет содержать сведения об автомобилях, их ремонтах, маршрутах, а также распределении водителей и автомобилей по маршрутным листам.

В приложении предусмотрены следующие роли:

1. Администратор;
2. Диспетчер;
3. Водитель.

2.2 Описание этапов разработки

Этапы проекта в соответствии с каскадной моделью:

1. Подготовка и проектирование.

Первым шагом в разработке является подготовка, в процессе которой изучается документация, выдвигаются технические требования, процесс разделяется на ряд мелких задач, ставятся сроки их выполнения.

Техническое задание определяет название средства, описание, основное его назначение, требования к выполнению и результатам работы. В ходе проектирования для каждой роли в приложении были выдвинуты определенные требования.

Функции администратора:

1. Создание, удаление учетных записей;
2. Назначение ролей учетным записям.

Функции диспетчера:

1. Создание маршрутных листов;
2. Распределение водителей на маршруты;
3. Назначение автомобилей на маршруты.

Водитель может:

1. Сделать заявку на ремонт автомобиля;
2. Сделать отметку о выполнении рейса.

Приложение будет содержать следующие модули:

1. CoreModule – содержит сущности, сервисы и контроллеры, реализующие описанные выше функции приложения.
2. AuthModule – содержит сервисы и контроллеры, необходимые для авторизации и аутентификации пользователей.

Также в этап подготовки входит создание схемы базы данных:

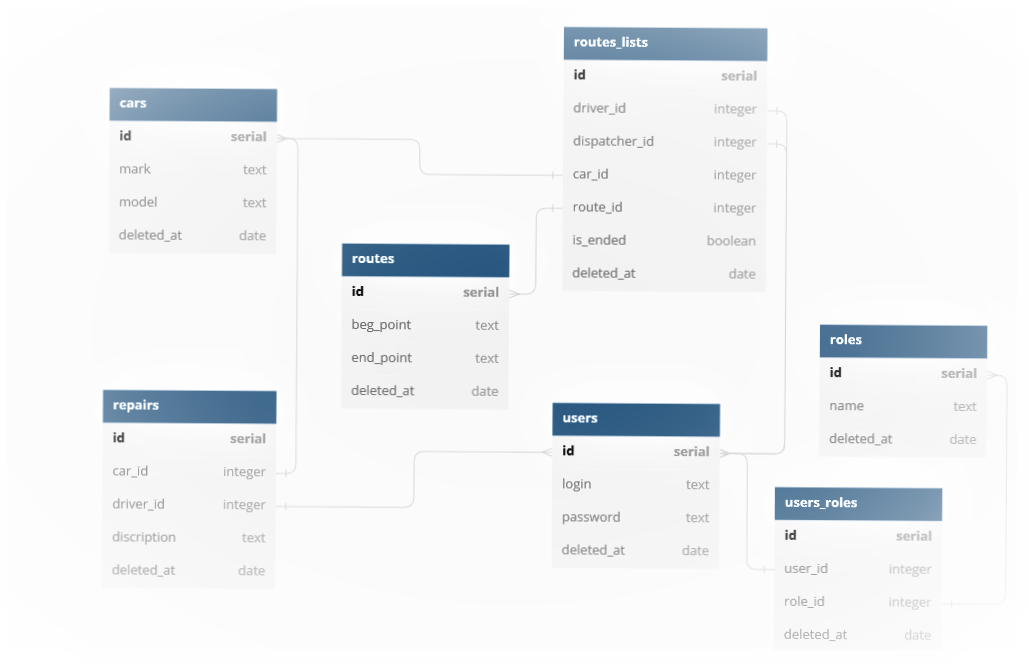


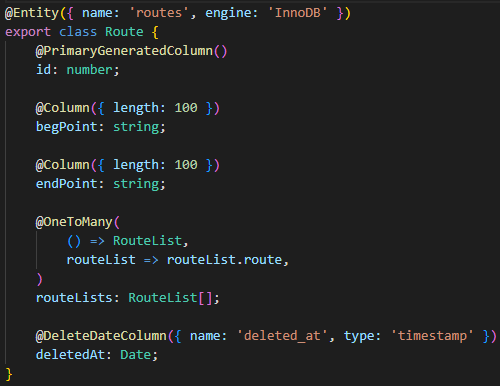
Рисунок 1 – схема базы данных приложения.

Следующим этапом после подготовки идёт проектирование, в ходе которого разрабатываются системы и алгоритмы взаимодействия данных, а также структурные компоненты.

1. Реализация приложения.

Создание приложения начинается с описания сущностей (entities). Entity – класс, который предназначен для хранения информации об объектах информационной системы. В нашем случае сущности используются для представления таблиц из базы данных в приложении.

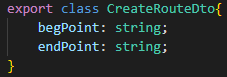
Листинг 1 – пример создания сущности Route (маршрут).



Для облегчения написания кода в NestJS используются декораторы. В примере выше используются декораторы @Entity() (содержит название таблицы в БД и тип движка БД), @PrimaryGeneratedColumn() (указывает, что данное поле в таблице – первичный ключ), @Column() (указывает, что полю класса соответствует столбец в таблице), @DeletedDateColumn() (поле, в которое записывается дата удаления объекта (мягкое удаление)), @OneToMany() (используется для указания связей с другими сущностями; также существуют @OneToOne(), @ManyToOne(), @ManyToMany() декораторы).

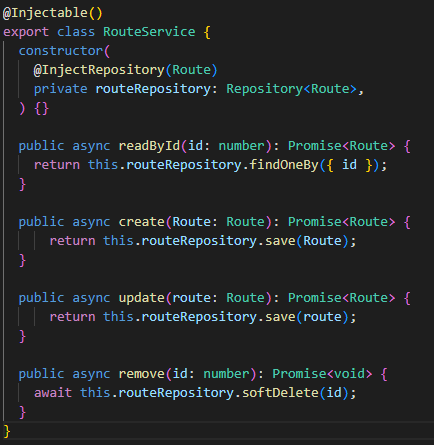
Для каждой сущности создаются data transfer objects (DTO). DTO – объект, который описывает, как данные передаются в сети. Например, при добавлении маршрута не нужно указывать id (он создается автоматически в БД), поэтому для создания маршрута будет использоваться следующий объект:

Листинг 2 – DTO для создания сущности Route.



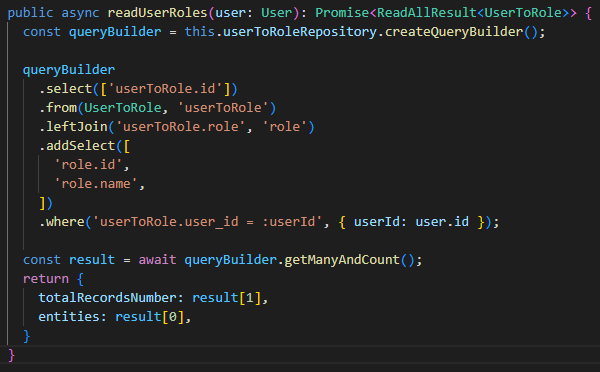
Следующим шагом создаются сервисы. Сервисы вызывают методы репозиториев, а репозитории отправляют запросы к БД.

Листинг 3 – сервис для сущности Route.



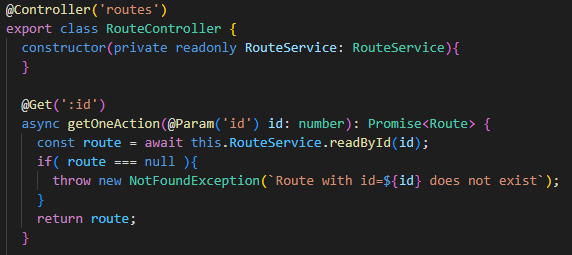
@InjectRepository() позволяет автоматически создать репозиторий, который содержит множество встроенных методов (например, save() – сохраняет переданный объект в БД). Если встроенных методов недостаточно, то можно использовать QueryBuilder, который позволяет создавать запросы к БД, используя его методы.

Листинг 4 – пример использования QueryBuilder для получения всех ролей пользователя.



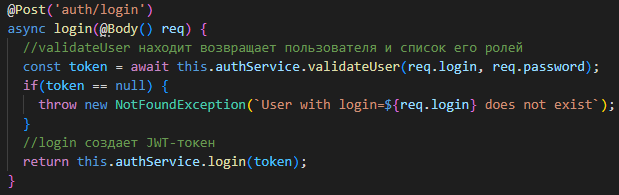
Для того, чтобы приложение могло принимать запросы и отправлять ответы используются контроллеры. Контроллер представляет собой класс, который содержит обработчики маршрутов, которые соответствуют методам HTTP-запроса и связанным маршрутам. В контролерах используются декораторы @Controller() (указывает общую часть маршрута для всех обработчиков), @Get() (указывает, что следующий метод обрабатывает запрос Get на определенном маршруте; подобные декораторы существуют для всех типов HTTP-запросов), @Param() (извлекает параметр из запроса) и т.д.

Листинг 5 – пример контроллера для маршрута /routes с обработчиком запроса Get на маршрут /route/[id].



Для авторизации и аутентификации в приложении будут использоваться JWT-токены. JWT-токен состоит из частей: заголовок, полезная нагрузка, подпись. Токен создается сервером при успешной аутентификации, передается клиенту, который он в дальнейшем использует для подтверждения подлинности. В полезной нагрузке токена будет храниться логин пользователя, а также список его ролей.

Листинг 6 – контролер для аутентификации пользователя.



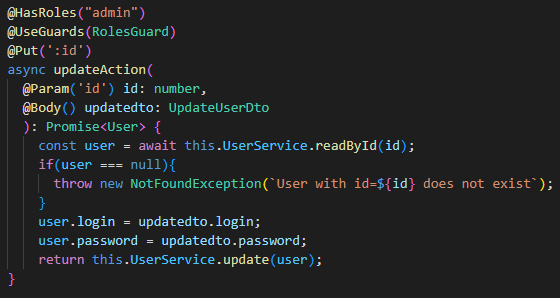
Для авторизации будет создан декоратор @HasRoles(), в который передается список ролей, и класс RolesGuard, который получает список ролей из декоратора @HasRoles() и список ролей пользователя из JWT-токена, затем сравнивает их, и при совпадении хотя бы одной роли разрешает доступ.

Листинг 7 – класс RolesGuard.



Для ограничения доступа используется декоратор @UseGuards().

Листинг 8 – использование @UseGuards() для разрешения доступа только пользователям с ролью admin.



1. Тестирование.

Тестирование будет происходить с помощью Postman.

Для начала проверим запрет на доступ к ресурсам неавторизованным пользователям:

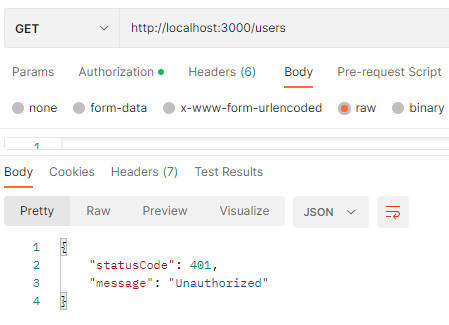


Рисунок 2 – попытка получить доступ неавторизованным пользователем.

Для авторизации отправим логин и пароль серверу.

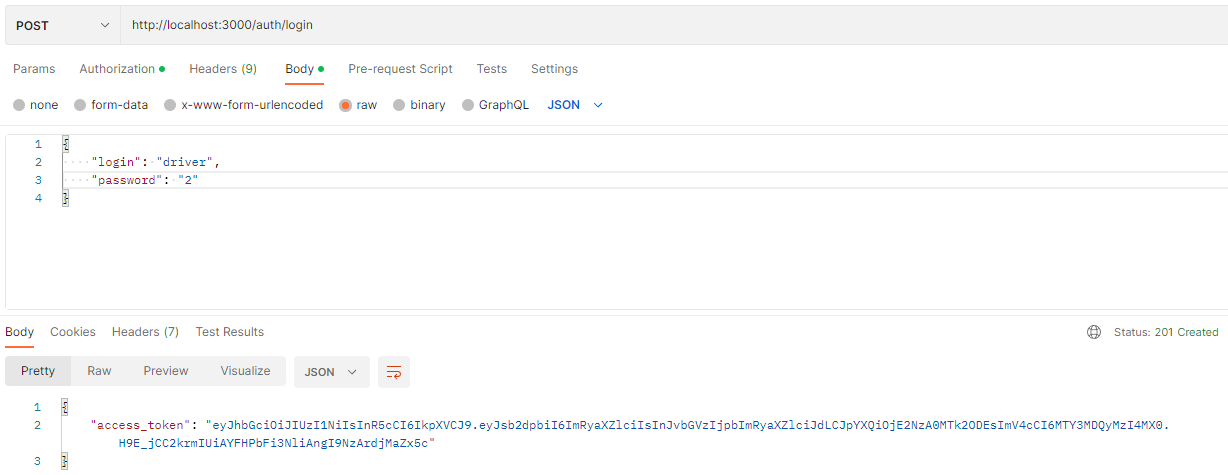


Рисунок 3 – успешная аутентификация пользователя, токен получен.

После получения токена, копируем его в заголовок запроса. Проверяем доступ:

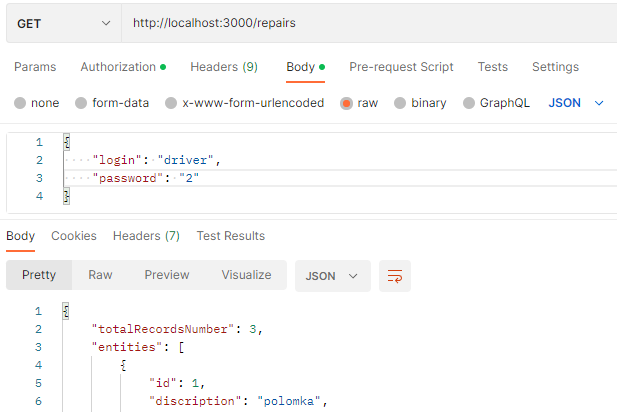


Рисунок 4 – успешная аутентификация, доступ получен.

Вошедший пользователь имеет только роль «водитель». Проверим, может ли он создавать маршрутные листы (функционал «диспетчера»):

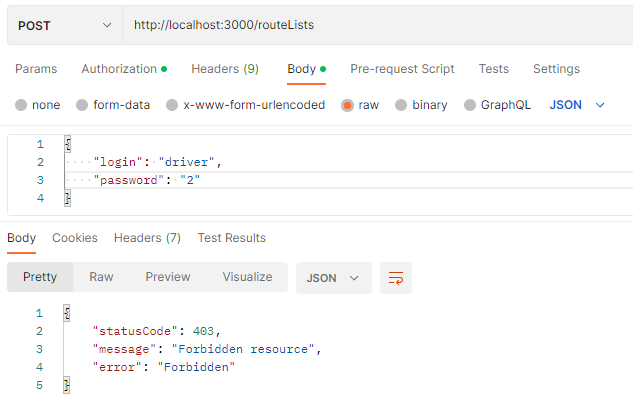


Рисунок 5 – авторизация работает успешно, доступ запрещен.

1. *Написание документации.*

Завершающим этапом разработки редактора является документирование результатов работы. В случае курсовой работы документированием является детальное изложение каждого этапа разработки, описание технического задания.

Заключение

В данной курсовой работе выполнена разработка приложения, реализующего информационную систему «Автобаза».

В результате проделанной работы были решены следующие задачи:

1. Проанализированы открытые источники информации, литература по теме курсовой работы.
2. Изучены особенности создания приложений с помощью NestJS.
3. Подготовлен план разработки приложения.
4. Реализовано приложение.

Созданное приложение с заданным набором функций свидетельствует о выполнении поставленных задач и о том, что цель работы достигнута.

Список использованных источников

1. Authentication – Mode of access: <https://docs.nestjs.com/> security/authentication – Date of access: 06.12.2022
2. First steps – Mode of access: <https://docs.nestjs.com/first-steps> - Date of access: 06.12.2022
3. Controllers – Mode of access: <https://docs.nestjs.com/controllers> - Date of access: 06.12.2022
4. Providers – Mode of access: <https://docs.nestjs.com/providers> - Date of access: 06.12.2022
5. Черный Борис Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений / Борис Черный — СПб.: Питер, 2021. — 352 с.
6. Вандеркам Дэн Эффективный TypeScript: 62 способа улучшить код / Дэн Вандеркам – СПб.: Питер, 2020. – 288 с.