**Раздел 3. Проектирование и создание сайта предприятия «КАЧЕСТВЕННЫЕ ОКНА-СИМФЕРОПОЛЬ».**

**3.1 Развертывание Nest.js приложения и установка зависимостей для Backend части сайта предприятия.**

Чтобы начать, мы можем либо создать каркас проекта с помощью Nest CLI, либо клонировать стартовый проект (оба варианта приведут к одинаковому результату). [12]

Для установки Nest CLI глобально в систему необходимо в терминале ввести команду:

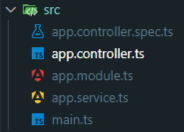
$ npm i -g @nestjs/cli

При этом в операционной системе уже должна быть установлена платформа Node.js от версии 16 и выше, которая позволит использовать NPM (Node Package Manager).

Чтобы создать каркас проекта с помощью Nest CLI, необходимо выполнить следующие команды. Это создаст новый каталог проекта и заполнит его исходными файлами ядра Nest и вспомогательными модулями, создавая обычную базовую структуру для проекта.

$ nest new server

Выполнение команды nest new приводит к генерации следующих файлов:

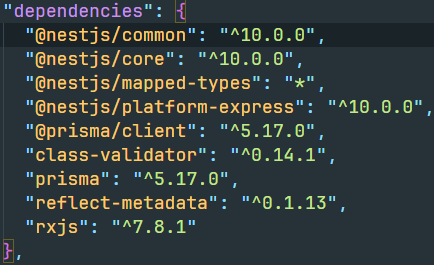


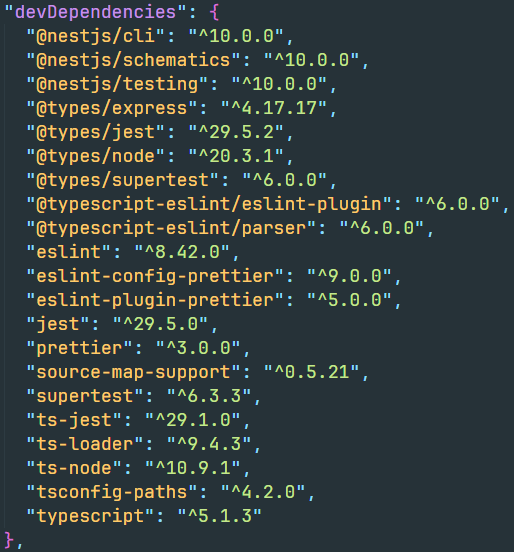
main.ts содержит асинхронную функцию, которая инициализирует (выполняет начальную загрузку, bootstrap) приложения:



В файле package.json можно увидеть в перечень настроек и установленных зависимостей:









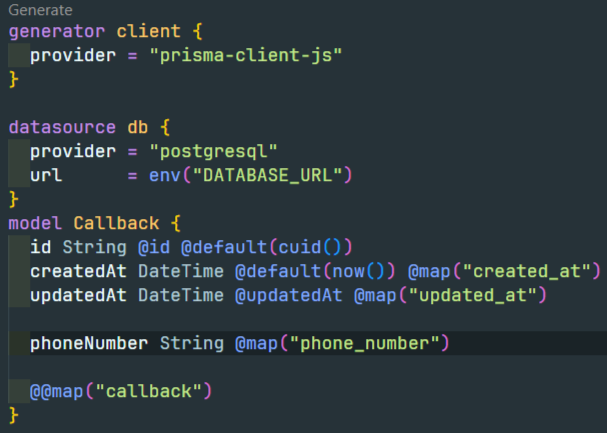
**3.2 Prisma и схема БД.**

Инициализируем проект Prisma:

$ npx prisma init

Prisma ORM создает файл .env по умолчанию в корневом каталоге ваших проектов. Вы можете заменить этот файл или создать новый в папке prisma, или, если вы решите переместить свой файл prisma.schema, то вместе с ним.

Далее в появившейся папке «prisma» в файле «shema.prisma» описываем модель таблицы в базе данных:



Установка и создание Prisma Client

Prisma Client — это типобезопасный клиент базы данных, созданный на основе определения вашей модели Prisma. Благодаря такому подходу Prisma Client может предоставлять доступ к CRUD-операциям, которые специально адаптированы к вашим моделям.

Чтобы установить Prisma Client в свой проект, выполните следующую команду в своем терминале:

$ npm install @prisma/client

Использование Prisma Client в сервисах Nest JS.

Теперь мы можем отправлять запросы к базе данных с помощью Prisma Client. При настройке приложения Nest JS мы захотим абстрагироваться от клиентского API Prisma для запросов к базе данных в рамках сервиса. Для начала мы можем создать новый сервис Prisma, который позаботится о создании экземпляра клиента Prisma и подключении к вашей базе данных.

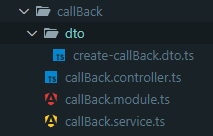
Внутри каталога src создадим новый файл с именем prisma.service.ts и добавим в него следующий код:



**3.3 Создание обработчика для сервиса обратной связи.**

Модуль - это класс, аннотированный с помощью декоратора @Module(). Декоратор @Module() предоставляет метаданные, которые Nest использует для организации структуры приложения.

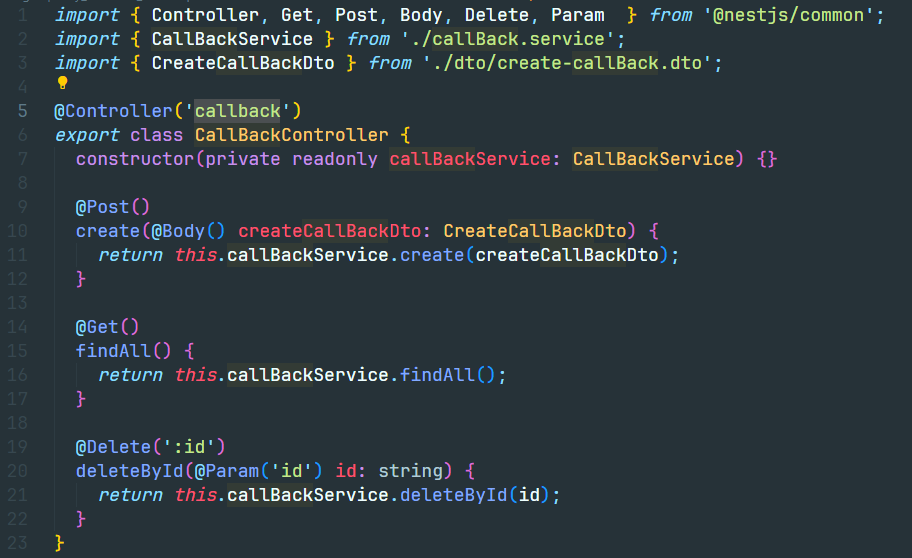
Чтобы создать модуль с помощью интерфейса командной строки, нужно выполнить команду $ nest g module callBack.



Контроллеры отвечают за обработку входящих запросов и возврат ответов клиенту. Назначение контроллера - получать конкретные запросы для приложения. Механизм маршрутизации определяет, какой контроллер получает какие запросы. Часто у каждого контроллера есть несколько маршрутов, и разные маршруты могут выполнять разные действия.  
Для создания базового контроллера мы используем классы и декораторы. Декораторы связывают классы с требуемыми метаданными и позволяют Nest создавать карту маршрутизации (привязывать запросы к соответствующим контроллерам).

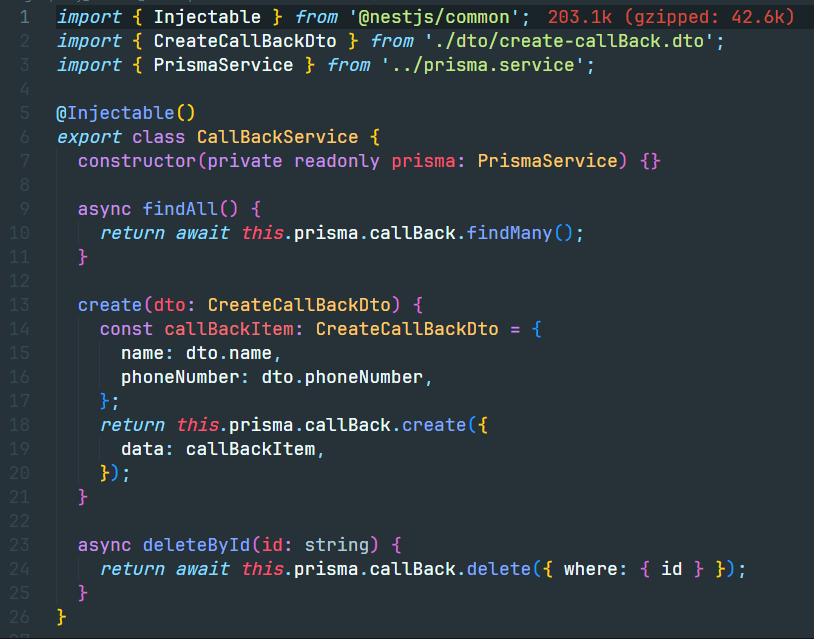
В Nest.js, **сервис** (service) представляет собой класс, предназначенный для обработки бизнес-логики приложения. Сервисы служат для того, чтобы инкапсулировать и отделить логику работы с данными от контроллеров, которые занимаются обработкой HTTP-запросов и передачей данных в представление.

В фаил callback.controller.ts добавляем код:



Данный контроллер будет обрабатывать запросы по адресу <http://localhost:3000/api/callBack>. В зависимости от метода запроса действия контроллера будут разными. Метод POST позволит создать в базе данных новую запись. Метод GET получить все записи из базы данных. Метод DELETE дает возможность удалить запись по предоставленному идентификатору.

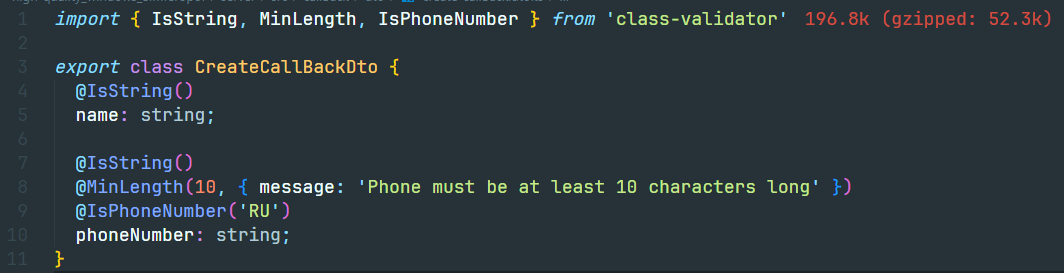
Для работы методов контроллера, их реализацию необходимо внести в фаил callback.service.ts:



При помощи принципа Dependency injection в конструктор класса помещается PrismaService, который позволяет использовать Prisma ORM как посредник между сервером и базой данных.

Функция findAll запускает функцию призмы findMany, которая вернет все записи в базе данных.

Функция create запускает одноименную функцию призмы create, которая создаст и в случае положительного результата вернет запись и базы PostgresSQL. Функция, в качестве параметра, ожидает получить данные структура которых определена в файле create-callBack.dto.ts:



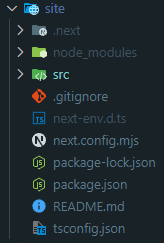
Функция deleteById запускает функцию призмы delete({ where: { id } }), которая позволяет удалить запись по id предоставленному в параметрах.

**3.4 Развертывание Next.js приложения и установка зависимостей для Frontend части сайта предприятия.**

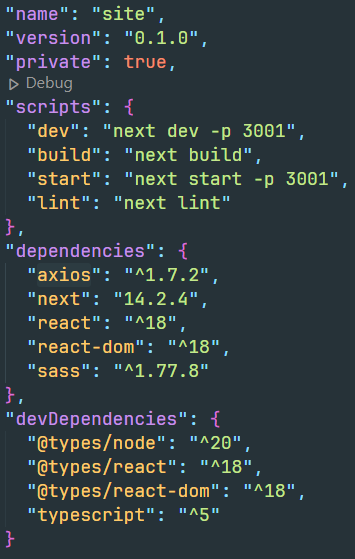
Команда разработчиков NEXT.JS рекомендует начать новое Next.js приложение, использующее create-next-app, которое автоматически настраивает все за нас. Чтобы создать проект, необходимо запустить:

npx create-next-app@latest

Тем самым мы получим следующую структуру проекта:



Список необходимых зависимостей отражен в package.json:



**3.5 Верстка сайта.**

В качестве макета для сайта предприятия использован Figma макет https://www.figma.com/design/8q4xFpFrwlqXpnLbO3G4hE/Impost-(Copy)?node-id=440-589&t=Quhgit2q8mZ2HhTn-0.