**Source Code:**

**Environment Setup:**

npm install @prisma/client prisma

npm install pg

**Define Prisma Models**

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

model User {

id String @id @default(uuid())

name String

email String @unique

createdAt DateTime @default(now())

projects Project[]

tasks Task[]

}

model

**Prisma schema definition**

model Project {

id String @id @default(uuid())

name String

description String

status ProjectStatus

createdAt DateTime @default(now())

userId String

user User @relation(fields: [userId], references: [id])

tasks Task[]

}

model Task {

id String @id @default(uuid())

title String

description String

status TaskStatus

createdAt DateTime @default(now())

projectId String

project Project @relation(fields: [projectId], references: [id])

assignedUserId String?

assignedUser User? @relation(fields: [assignedUserId], references: [id])

}

enum ProjectStatus {

PLANNED

ONGOING

COMPLETED

}

enum TaskStatus {

TODO

IN\_PROGRESS

DONE

}

**Run Migrations**

npx prisma migrate dev --name init

**Set Up NestJS Modules**

nest generate module users

nest generate module projects

nest generate module tasks

**Create DTOs (Data Transfer Objects)**

// src/users/dto/create-user.dto.ts

export class CreateUserDto {

name: string;

email: string;

}

// src/projects/dto/create-project.dto.ts

export class CreateProjectDto {

name: string;

description: string;

status: ProjectStatus;

userId: string;

}

// src/tasks/dto/create-task.dto.ts

export class CreateTaskDto {

title: string;

description: string;

status: TaskStatus;

projectId: string;

assignedUserId?: string;

}

**Implement CRUD Services and Controllers**

// src/users/users.service.ts

import { Injectable } from '@nestjs/common';

import { PrismaService } from '../prisma/prisma.service';

import { CreateUserDto } from './dto/create-user.dto';

@Injectable()

export class UsersService {

constructor(private prisma: PrismaService) {}

async create(createUserDto: CreateUserDto) {

return this.prisma.user.create({ data: createUserDto });

}

async findAll() {

return this.prisma.user.findMany();

}

// Other CRUD methods (update, delete)

}

**Filtering and Search**

// src/tasks/tasks.controller.ts

@Get()

async findAll(

@Query('status') status?: TaskStatus,

@Query('assignedUserId') assignedUserId?: string,

) {

return this.tasksService.findFiltered(status, assignedUserId);

}

**Create User Service**

import { Injectable } from '@nestjs/common';

import { PrismaService } from '../prisma/prisma.service';

import { CreateUserDto } from './dto/create-user.dto';

import { UpdateUserDto } from './dto/update-user.dto';

@Injectable()

export class UsersService {

constructor(private prisma: PrismaService) {}

async create(createUserDto: CreateUserDto) {

return this.prisma.user.create({ data: createUserDto });

}

async findAll() {

return this.prisma.user.findMany();

}

async findOne(id: string) {

return this.prisma.user.findUnique({ where: { id } });

}

async update(id: string, updateUserDto: UpdateUserDto) {

return this.prisma.user.update({

where: { id },

data: updateUserDto,

});

}

async remove(id: string) {

return this.prisma.user.delete({ where: { id } });

}

}

**Create User Controller**

import { Controller, Get, Post, Body, Param, Put, Delete } from '@nestjs/common';

import { UsersService } from './users.service';

import { CreateUserDto } from './dto/create-user.dto';

import { UpdateUserDto } from './dto/update-user.dto';

@Controller('users')

export class UsersController {

constructor(private readonly usersService: UsersService) {}

@Post()

create(@Body() createUserDto: CreateUserDto) {

return this.usersService.create(createUserDto);

}

@Get()

findAll() {

return this.usersService.findAll();

}

@Get(':id')

findOne(@Param('id') id: string) {

return this.usersService.findOne(id);

}

@Put(':id')

update(@Param('id') id: string, @Body() updateUserDto: UpdateUserDto) {

return this.usersService.update(id, updateUserDto);

}

@Delete(':id')

remove(@Param('id') id: string) {

return this.usersService.remove(id);

}

}