Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Базы данных

Тема «Репетиционная база»

Лабораторная работа №1

Разработка серверной части прикладной программы

Студент: А.С. Бригадир

Преподаватель: С.С. Силич

МИНСК 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc208442049)

[1 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ 4](#_Toc208442050)

[1.1 Описание реляционной модели 4](#_Toc208442051)

[1.2 Описание таблиц 4](#_Toc208442052)

[1.3 Выделение справочных и основных таблиц 6](#_Toc208442053)

[1.4 Выделение прав доступа 7](#_Toc208442054)

[1.5 Определение требований к серверной части 7](#_Toc208442055)

[2 ПРОГРАММИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ 8](#_Toc208442056)

[2.1 Создание скриптов 8](#_Toc208442057)

[2.2 Реализация HTTP-сервера 11](#_Toc208442058)

[ЗАКЛЮЧЕНИЕ 15](#_Toc208442059)

# ВВЕДЕНИЕ

Данная лабораторная работа предполагает создание серверной части приложения, включая разработку спецификаций, реализацию HTTP-сервера и обеспечение взаимодействия с базой данных через стандартные методы REST API.

Работа опирается на результаты лабораторной работы №6 первого семестра, где была разработана начальная реляционная схема и реализована базовая структура базы данных.

В рамках текущего задания предполагается уточнение схемы, определение ролей пользователей, разработка технических требований и программирование серверной части. Особое внимание уделяется обеспечению безопасности доступа через разделение прав между обычными пользователями и суперпользователями, а также реализации операций резервного копирования и фильтрации данных.

# ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

## 1.1 Описание реляционной модели

Реляционная схема осталась без изменений и изображена в соответствии с UML на рисунке 1.1.



Рисунок 1.1 – Уточненная реляционная схема

## 1.2 Описание таблиц

Таблица rehearsal\_points включает основные данные о репетиционных базах. Описание имен таблицы rehearsal\_points:

– id: идентификатор репетиционной базы. Первичный ключ;

– rating: рейтинг базы;

– contact\_number: контактный номер телефона (обязательное поле);

– schedule: расписание (JSON);

– name: название репетиционной базы (обязательное поле);

– address: адрес репетиционной базы (обязательное поле).

Таблица rooms представляет данные системы. Описание имен таблицы rooms:

– id: идентификатор комнаты. Первичный ключ;

– name: название комнаты (обязательное поле);

– air\_conditioner: наличие кондиционера (по умолчанию FALSE);

– price: стоимость аренды (обязательное поле);

– recording\_support: поддержка записи (по умолчанию FALSE);

– area: площадь комнаты (обязательное поле);

–id\_rehearsal\_point: внешний ключ на таблицу rehearsal\_points.

Таблица service представляет данные о всех видах услуг. Описание имен таблицы service:

– id: идентификатор услуги. Первичный ключ;

– name: название услуги (обязательное поле);

– price: стоимость услуги (обязательное поле);

– type: тип услуги (обязательное поле);

– requirements: дополнительные требования;

–id\_rehearsal\_point: внешний ключ на таблицу rehearsal\_points.

Таблица equipment представляет данные о всем оборудовании репетиционных точек. Описание имен таблицы equipment:

– id: идентификатор оборудования. Первичный ключ;

– name: название оборудования (обязательное поле);

– type: тип оборудования (обязательное поле);

– brand: бренд оборудования (обязательное поле);

– model: модель оборудования (обязательное поле);

– condition: состояние оборудования (обязательное поле);

-id\_rehearsal\_point: внешний ключ на таблицу rehearsal\_points.

Таблица staff представляет данные о сотрудниках. Описание имен таблицы staff:

– id: идентификатор сотрудника. Первичный ключ;

– full\_name: ФИО сотрудника (обязательное поле);

– address: адрес сотрудника;

– experience: опыт работы (в годах);

– phone: номер телефона (обязательное поле);

– age: возраст сотрудника (обязательное поле);

– id\_rehearsal\_point: внешний ключ на таблицу rehearsal\_points.

Таблица users представляет данные о пользователях. Описание имен таблицы users:

– id: идентификатор пользователя. Первичный ключ;

– full\_name: ФИО пользователя (обязательное поле);

– phone: номер телефона (обязательное поле);

– email: адрес электронной почты (обязательное поле);

– registration\_date: дата регистрации (обязательное поле).

Таблица booking представляет данные о бронированиях. Описание имен таблицы booking:

– id: идентификатор бронирования. Первичный ключ;

– time: время бронирования (обязательное поле);

– duration: длительность бронирования;

– cost: стоимость (обязательное поле);

– creation\_date: дата создания (обязательное поле);

– status: статус бронирования (обязательное поле);

– number\_of\_people: количество людей (обязательное поле);

– id\_room: внешний ключ на таблицу rooms;

– id\_user: внешний ключ на таблицу users.

Таблица service\_booking представляет собой побочную таблицу связи many-to-many. Описание имен таблицы service\_booking:

– id\_service: внешний ключ на таблицу service. Первичный ключ;

– id\_booking: внешний ключ на таблицу booking. Первичный ключ.

Таблица equipment\_booking представляет собой побочную таблицу связи many-to-many. Описание имен таблицы equipment\_booking:

– id\_equipment: внешний ключ на таблицу equipment. Первичный ключ;

– id\_booking: внешний ключ на таблицу booking. Первичный ключ.

## 1.3 Выделение справочных и основных таблиц

В данной схеме в категорию справочных таблиц должны быть выделены: service\_booking и equipment\_booking, так как они содержат данные для сопоставления услуг и оборудования с бронированиями и изменяются только администратором.

В качестве основной таблицы должна быть выделена таблица rehearsal\_points, так как она содержит основные данные о точках репетиций, с которых начинается работа приложения.

## 1.4 Выделение прав доступа

Пользователь должен обладать правами просмотра, сохранения результатов запросов и редактирования всех таблиц, кроме справочных, а суперпользователь обладать теми же правами что и обычный пользователь, но с возможностью редактирования справочных таблиц и создания бэкапа базы данных. Для выполнения действий от имени суперпользователя приложение должно запрашивать пароль суперпользователя.

## 1.5 Определение требований к серверной части

Серверная часть прикладной программы должна быть реализована в виде HTTP-сервера. Тела ответов сервера, так же, как и тела запросов должны быть представлены в формате JSON.

Для взаимодействия с ресурсами (таблицами) должны использоваться стандартные HTTP-методы:

1. GET – получение данных о ресурсе;
2. POST – создание нового ресурса;
3. PUT – обновление существующего ресурса;
4. DELETE – удаление ресурса.

Каждый ресурс должен быть доступен по уникальному URL:

– /api/rehearsal\_points: таблица rehearsal\_points;

– /api/rooms: таблица rooms;

– /api/service: таблица service;

– /api/equipment: таблица equipment;

– /api/staff: таблица staff;

– /api/users: таблица users;

– /api/booking: таблица booking;

– /api/service\_booking: таблица service\_booking;

– /api/equipment\_booking: таблица equipment\_booking.

Серверная часть прикладной программы должна предоставлять следующие операции для работы с базой данных:

– просмотр таблиц;

– фильтрация содержимого таблиц;

– добавление записей в таблицы;

– обновление записей в таблицах;

– удаление записей из таблиц;

– выполнение специальных запросов;

– создание бэкапов базы данных;

– сохранение результатов запросов в файл.

# ПРОГРАММИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ

## 2.1 Создание скриптов

Для создания таблиц в базе данных используется следующий скрипт:

CREATE TABLE IF NOT EXISTS main.booking (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"Time" timestamp with time zone NOT NULL,

"Duration" integer,

"Cost" integer NOT NULL,

"CreationDate" timestamp with time zone NOT NULL,

"Status" text NOT NULL,

"NumberOfPeople" integer NOT NULL,

"IdRoom" integer,

"IdUser" integer

);

CREATE TABLE IF NOT EXISTS main.equipment (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"Name" text NOT NULL,

"Type" text NOT NULL,

"Brand" text NOT NULL,

"Model" text NOT NULL,

"Condition" text NOT NULL,

"IdRehearsalPoint" integer

);

CREATE TABLE IF NOT EXISTS main.equipment\_booking (

"IdEquipment" integer NOT NULL,

"IdBooking" integer NOT NULL

);

CREATE TABLE IF NOT EXISTS main.rehearsal\_points (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"Rating" real,

"ContactNumber" text NOT NULL,

"Schedule" text NOT NULL,

"Name" text NOT NULL,

"Address" text NOT NULL

);

CREATE TABLE IF NOT EXISTS main.rooms (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"Name" text NOT NULL,

"AirConditioner" boolean NOT NULL,

"Price" integer NOT NULL,

"RecordingSupport" boolean NOT NULL,

"Area" integer NOT NULL,

"IdRehearsalPoint" integer

);

CREATE TABLE IF NOT EXISTS main.service (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"Name" text NOT NULL,

"Price" integer NOT NULL,

"Type" text NOT NULL,

"Requirements" text,

"IdRehearsalPoint" integer

);

CREATE TABLE IF NOT EXISTS main.service\_booking (

"IdService" integer NOT NULL,

"IdBooking" integer NOT NULL

);

CREATE TABLE IF NOT EXISTS main.staff (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"FullName" text NOT NULL,

"Address" text,

"Experience" integer,

"Phone" text NOT NULL,

"Age" integer NOT NULL,

"IdRehearsalPoint" integer

);

CREATE TABLE IF NOT EXISTS main.users (

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

"FullName" text NOT NULL,

"Phone" text NOT NULL,

"Email" text NOT NULL,

"RegistrationDate" timestamp with time zone NOT NULL

);

ALTER TABLE ONLY main.booking ADD CONSTRAINT PK\_booking PRIMARY KEY ("Id");

ALTER TABLE ONLY main.equipment ADD CONSTRAINT PK\_equipment PRIMARY KEY ("Id");

ALTER TABLE ONLY main.equipment\_booking ADD CONSTRAINT PK\_equipment\_booking PRIMARY KEY ("IdEquipment", "IdBooking");

ALTER TABLE ONLY main.rehearsal\_points ADD CONSTRAINT PK\_rehearsal\_points PRIMARY KEY ("Id");

ALTER TABLE ONLY main.rooms ADD CONSTRAINT PK\_rooms PRIMARY KEY ("Id");

ALTER TABLE ONLY main.service ADD CONSTRAINT PK\_service PRIMARY KEY ("Id");

ALTER TABLE ONLY main.service\_booking ADD CONSTRAINT PK\_service\_booking PRIMARY KEY ("IdService", "IdBooking");

ALTER TABLE ONLY main.staff ADD CONSTRAINT PK\_staff PRIMARY KEY ("Id");

ALTER TABLE ONLY main.users ADD CONSTRAINT PK\_users PRIMARY KEY ("Id");

ALTER TABLE ONLY main.booking ADD CONSTRAINT FK\_booking\_rooms\_IdRoom FOREIGN KEY ("IdRoom") REFERENCES main.rooms("Id") ON DELETE SET NULL;

ALTER TABLE ONLY main.booking ADD CONSTRAINT FK\_booking\_users\_IdUser FOREIGN KEY ("IdUser") REFERENCES main.users("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.equipment\_booking ADD CONSTRAINT FK\_equipment\_booking\_booking\_IdBooking FOREIGN KEY ("IdBooking") REFERENCES main.booking("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.equipment\_booking ADD CONSTRAINT FK\_equipment\_booking\_equipment\_IdEquipment FOREIGN KEY ("IdEquipment") REFERENCES main.equipment("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.equipment ADD CONSTRAINT FK\_equipment\_rehearsal\_points\_IdRehearsalPoint FOREIGN KEY ("IdRehearsalPoint") REFERENCES main.rehearsal\_points("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.rooms ADD CONSTRAINT FK\_rooms\_rehearsal\_points\_IdRehearsalPoint FOREIGN KEY ("IdRehearsalPoint") REFERENCES main.rehearsal\_points("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.service\_booking ADD CONSTRAINT FK\_service\_booking\_booking\_IdBooking FOREIGN KEY ("IdBooking") REFERENCES main.booking("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.service\_booking ADD CONSTRAINT FK\_service\_booking\_service\_IdService FOREIGN KEY ("IdService") REFERENCES main.service("Id") ON DELETE CASCADE;

ALTER TABLE ONLY main.service ADD CONSTRAINT FK\_service\_rehearsal\_points\_IdRehearsalPoint FOREIGN KEY ("IdRehearsalPoint") REFERENCES main.rehearsal\_points("Id") ON DELETE CASCADE;

## 2.2 Реализация HTTP-сервера

Для создания серверной части был использован DB-first подход и ORM Entity Framework Core. Для всех таблиц были описаны сущности и репозитории, которые для взаимодействия с базой данных Postgresql использовали библиотеку libpq. Также для каждой сущности и для экспорта данных были созданы соответствующие контроллеры. Листинг кода приведен в приложении А.

Для тестирования GET запросов была использована поисковая строка браузера. В качестве примера на рисунках 2.1 и 2.2 отображены результаты выполнения запросов таблиц rehearsal\_points и equipment соответственно.

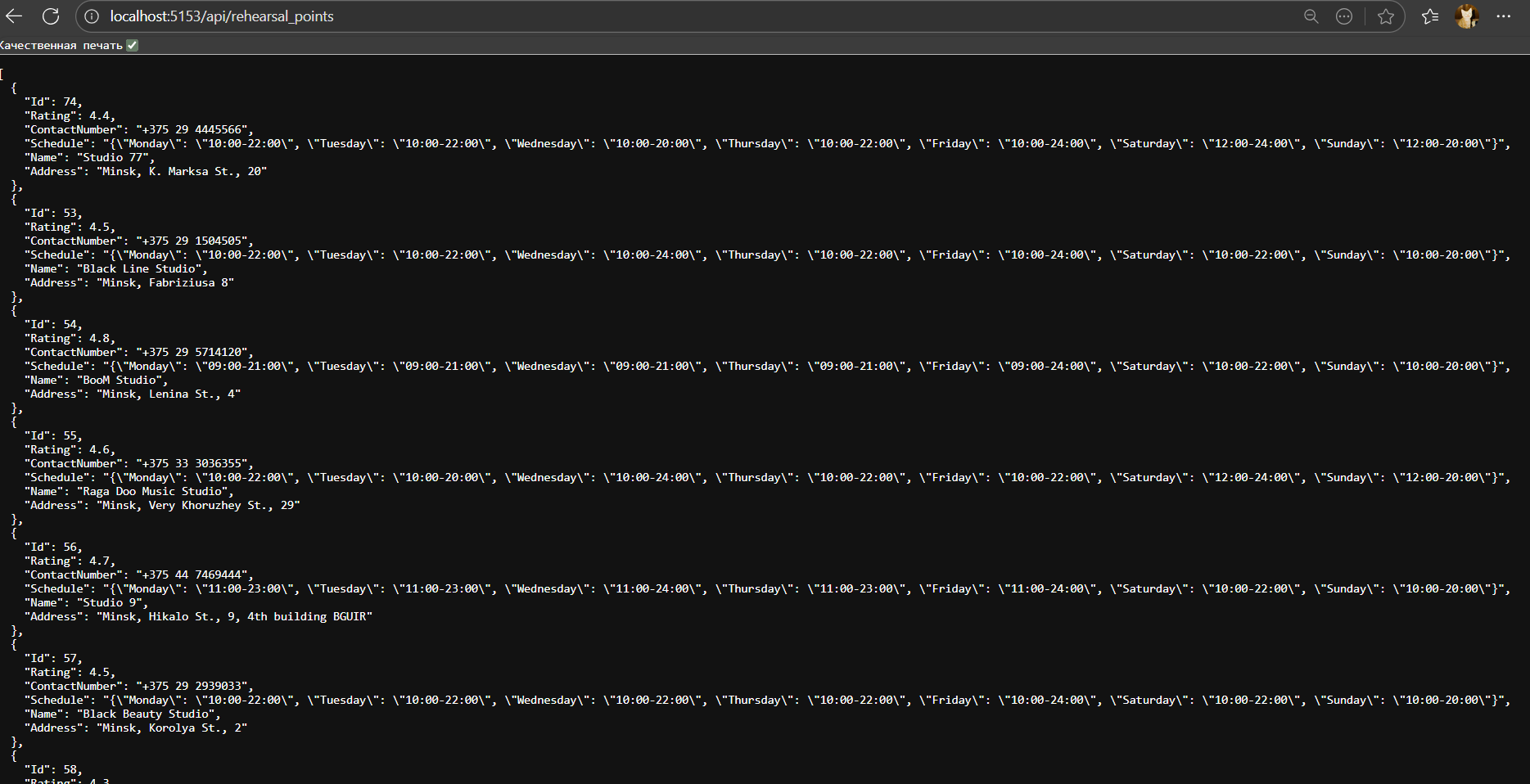


Рисунок 2.1 – Результат запроса таблицы rehearsal\_points

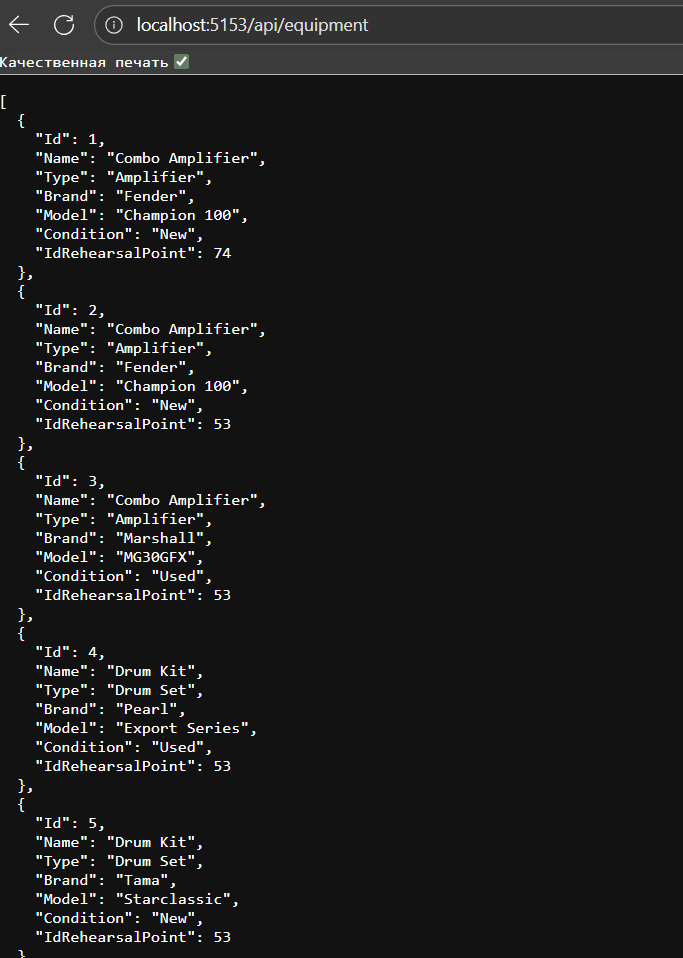


Рисунок 2.2 – Результат запроса таблицы equipment

Также была реализована фильтрация по одному или нескольким параметрам. В качестве примера на рисунке 2.3 приведен результат запроса таблицы service с примененной фильтрации по типу услуги, а именно услуги записи.

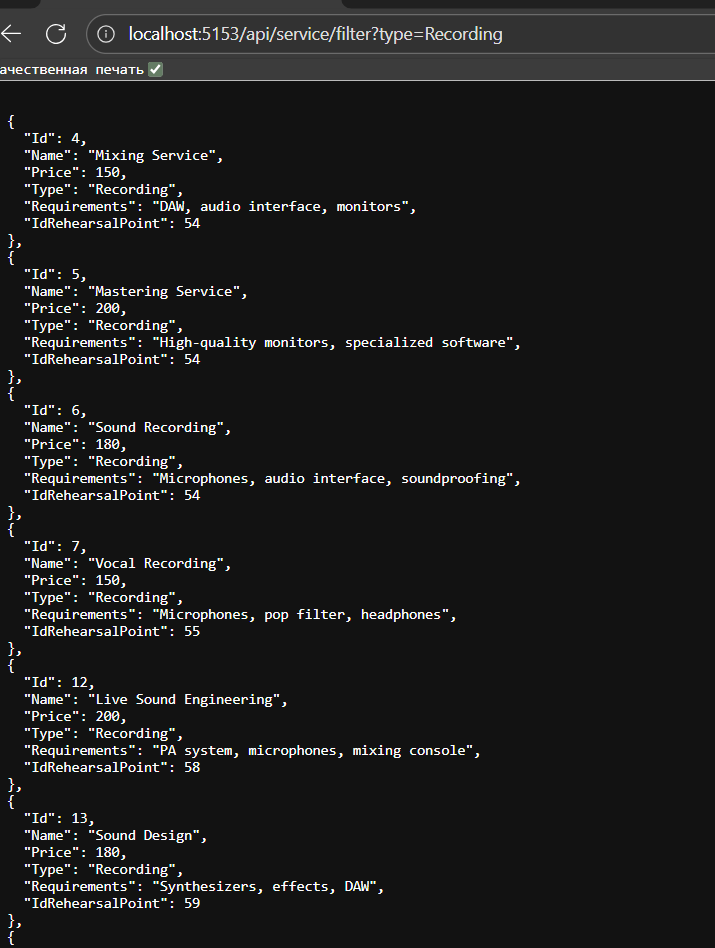


Рисунок 2.3 – Результат запроса таблицы service

Для тестирования POST запросов контроллера создания бэкапов был использован Swagger. В качестве примера на рисунке 2.4 отображен результат создания бэкапа.

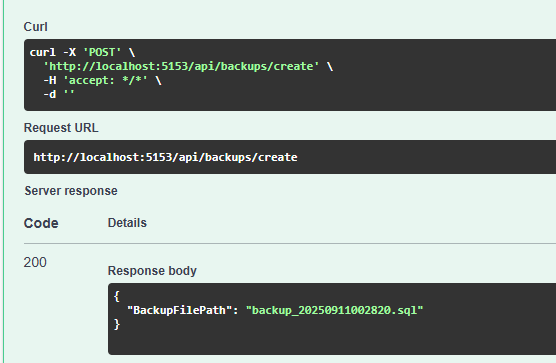


Рисунок 2.3 – Результат создания бэкапа базы данных

В качестве примера на рисунке 2.4 отображен результат POST запроса на создание бронирования.

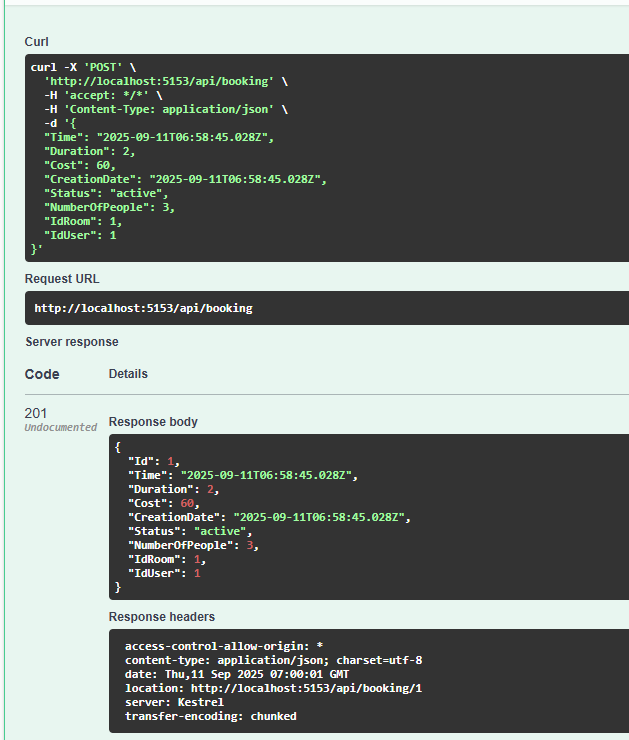


Рисунок 2.3 – Результат POST запроса на создание бронирования

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были успешно реализованы ключевые этапы разработки серверной части прикладной программы. На основе уточненной реляционной схемы на основе лабораторной работы №6 первого семестра были разработаны технические требования, включая определение ролей пользователей, основной таблицы и справочных таблиц. Серверная часть реализована в виде HTTP-сервера с использованием формата JSON для обмена данными, что обеспечило совместимость и удобство взаимодействия.

Реализованы стандартные HTTP-методы (GET, POST, PUT, DELETE) для работы с ресурсами, доступными по уникальным URL, а также широкий спектр операций: просмотр, фильтрация, добавление, обновление и удаление записей, выполнение специальных запросов, создание бэкапов и сохранение результатов.

Приложение А

**Листинг кода**

Файл Program.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Application.Services;

004 using RehearsalStudio.Infrastructure.Data;

005 using RehearsalStudio.Infrastructure.Repositories;

006 using Microsoft.OpenApi.Models;

007 using System.Reflection;

008 var builder = WebApplication.CreateBuilder(args);

009 // Add services to the container.

010 // Register Entity Framework Core with PostgreSQL

011 builder.Services.AddDbContext<RehearsalStudioDbContext>(options =>

012 options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")));

013 // Register repositories

014 builder.Services.AddScoped<IRehearsalPointRepository, RehearsalPointRepository>();

015 builder.Services.AddScoped<IRoomRepository, RoomRepository>();

016 builder.Services.AddScoped<IServiceRepository, ServiceRepository>();

017 builder.Services.AddScoped<IEquipmentRepository, EquipmentRepository>();

018 builder.Services.AddScoped<IStaffRepository, StaffRepository>();

019 builder.Services.AddScoped<IUserRepository, UserRepository>();

020 builder.Services.AddScoped<IBookingRepository, BookingRepository>();

021 builder.Services.AddScoped<IServiceBookingRepository, ServiceBookingRepository>();

022 builder.Services.AddScoped<IEquipmentBookingRepository, EquipmentBookingRepository>();

023 // Register services

024 builder.Services.AddScoped<IRehearsalPointService, RehearsalPointService>();

025 builder.Services.AddScoped<IRoomService, RoomService>();

026 builder.Services.AddScoped<IServiceService, ServiceService>();

027 builder.Services.AddScoped<IEquipmentService, EquipmentService>();

028 builder.Services.AddScoped<IStaffService, StaffService>();

029 builder.Services.AddScoped<IUserService, UserService>();

030 builder.Services.AddScoped<IBookingService, BookingService>();

031 builder.Services.AddScoped<IServiceBookingService, ServiceBookingService>();

032 builder.Services.AddScoped<IEquipmentBookingService, EquipmentBookingService>();

033 builder.Services.AddScoped<IBackupService, BackupService>();

034 // Add controllers

035 builder.Services.AddControllers()

036 .AddJsonOptions(options =>

037 {

038 options.JsonSerializerOptions.PropertyNamingPolicy = null; // Preserve property names as-is

039 });

040 // Configure CORS (optional, for front-end integration)

041 builder.Services.AddCors(options =>

042 {

043 options.AddPolicy("AllowAll", policy =>

044 {

045 policy.AllowAnyOrigin()

046 .AllowAnyMethod()

047 .AllowAnyHeader();

048 });

049 });

050 // Configure Swagger/OpenAPI

051 builder.Services.AddEndpointsApiExplorer();

052 builder.Services.AddSwaggerGen(c =>

053 {

054 c.SwaggerDoc("v1", new OpenApiInfo

055 {

056 Title = "RehearsalStudio API",

057 Version = "v1",

058 Description = "API for managing rehearsal studio resources"

059 });

060 });

061 // Build the application

062 var app = builder.Build();

063 // Configure the HTTP request pipeline

064 if (app.Environment.IsDevelopment())

065 {

066 app.UseSwagger();

067 app.UseSwaggerUI(c =>

068 {

069 c.SwaggerEndpoint("/swagger/v1/swagger.json", "RehearsalStudio API v1");

070 c.RoutePrefix = string.Empty; // Serve Swagger at root (/)

071 });

072 }

073 app.UseHttpsRedirection();

074 app.UseCors("AllowAll"); // Apply CORS policy

075 app.UseAuthorization();

076 app.MapControllers();

077 app.Run();

Файл BackupsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.Interfaces;

003 using System.Threading.Tasks;

004 namespace RehearsalStudio.Api.Controllers;

005 [Route("api/backups")]

006 [ApiController]

007 public class BackupsController : ControllerBase

008 {

009 private readonly IBackupService \_backupService;

010 public BackupsController(IBackupService backupService)

011 {

012 \_backupService = backupService;

013 }

014 [HttpPost("create")]

015 public async Task<IActionResult> CreateBackup()

016 {

017 var backupPath = await \_backupService.CreateDatabaseBackupAsync();

018 return Ok(new { BackupFilePath = backupPath });

019 }

020 [HttpPost("query-save")]

021 public async Task<IActionResult> SaveQueryResults([FromBody] QuerySaveRequest request)

022 {

023 var resultPath = await \_backupService.SaveQueryResultsToFileAsync(request.SqlQuery, request.FileFormat);

024 return Ok(new { ResultFilePath = resultPath });

025 }

026 }

027 public class QuerySaveRequest

028 {

029 public string SqlQuery { get; set; } = string.Empty;

030 public string FileFormat { get; set; } = "json";

031 }

Файл BookingsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/booking")]

007 [ApiController]

008 public class BookingsController : ControllerBase

009 {

010 private readonly IBookingService \_service;

011 public BookingsController(IBookingService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? status, [FromQuery] int? idRoom, [FromQuery] int? idUser)

031 {

032 var result = await \_service.GetFilteredAsync(status, idRoom, idUser);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] BookingDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] BookingDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл EquipmentBookingsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/equipment\_booking")]

007 [ApiController]

008 public class EquipmentBookingsController : ControllerBase

009 {

010 private readonly IEquipmentBookingService \_service;

011 public EquipmentBookingsController(IEquipmentBookingService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{idEquipment}/{idBooking}")]

022 public async Task<IActionResult> GetById(int idEquipment, int idBooking)

023 {

024 var result = await \_service.GetByIdAsync(idEquipment, idBooking);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] int? idEquipment, [FromQuery] int? idBooking)

031 {

032 var result = await \_service.GetFilteredAsync(idEquipment, idBooking);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] EquipmentBookingDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { idEquipment = result.IdEquipment, idBooking = result.IdBooking }, result);

040 }

041 [HttpDelete("{idEquipment}/{idBooking}")]

042 public async Task<IActionResult> Delete(int idEquipment, int idBooking)

043 {

044 await \_service.DeleteAsync(idEquipment, idBooking);

045 return NoContent();

046 }

047 }

Файл EquipmentController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/equipment")]

007 [ApiController]

008 public class EquipmentController : ControllerBase

009 {

010 private readonly IEquipmentService \_service;

011 public EquipmentController(IEquipmentService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? name, [FromQuery] string? type, [FromQuery] int? idRehearsalPoint)

031 {

032 var result = await \_service.GetFilteredAsync(name, type, idRehearsalPoint);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] EquipmentDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] EquipmentDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл RehearsalPointsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/rehearsal\_points")]

007 [ApiController]

008 public class RehearsalPointsController : ControllerBase

009 {

010 private readonly IRehearsalPointService \_service;

011 public RehearsalPointsController(IRehearsalPointService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? name, [FromQuery] float? minRating)

031 {

032 var result = await \_service.GetFilteredAsync(name, minRating);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] RehearsalPointDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] RehearsalPointDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл RoomsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/rooms")]

007 [ApiController]

008 public class RoomsController : ControllerBase

009 {

010 private readonly IRoomService \_service;

011 public RoomsController(IRoomService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? name, [FromQuery] int? minPrice, [FromQuery] int? idRehearsalPoint)

031 {

032 var result = await \_service.GetFilteredAsync(name, minPrice, idRehearsalPoint);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] RoomDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] RoomDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл ServiceBookingsController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/service\_booking")]

007 [ApiController]

008 public class ServiceBookingsController : ControllerBase

009 {

010 private readonly IServiceBookingService \_service;

011 public ServiceBookingsController(IServiceBookingService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{idService}/{idBooking}")]

022 public async Task<IActionResult> GetById(int idService, int idBooking)

023 {

024 var result = await \_service.GetByIdAsync(idService, idBooking);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] int? idService, [FromQuery] int? idBooking)

031 {

032 var result = await \_service.GetFilteredAsync(idService, idBooking);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] ServiceBookingDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { idService = result.IdService, idBooking = result.IdBooking }, result);

040 }

041 [HttpDelete("{idService}/{idBooking}")]

042 public async Task<IActionResult> Delete(int idService, int idBooking)

043 {

044 await \_service.DeleteAsync(idService, idBooking);

045 return NoContent();

046 }

047 }

Файл ServicesController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/service")]

007 [ApiController]

008 public class ServicesController : ControllerBase

009 {

010 private readonly IServiceService \_service;

011 public ServicesController(IServiceService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? name, [FromQuery] string? type, [FromQuery] int? idRehearsalPoint)

031 {

032 var result = await \_service.GetFilteredAsync(name, type, idRehearsalPoint);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] ServiceDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] ServiceDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл UsersController.cs:

001 using Microsoft.AspNetCore.Mvc;

002 using RehearsalStudio.Application.DTOs;

003 using RehearsalStudio.Application.Interfaces;

004 using System.Threading.Tasks;

005 namespace RehearsalStudio.Api.Controllers;

006 [Route("api/users")]

007 [ApiController]

008 public class UsersController : ControllerBase

009 {

010 private readonly IUserService \_service;

011 public UsersController(IUserService service)

012 {

013 \_service = service;

014 }

015 [HttpGet]

016 public async Task<IActionResult> GetAll()

017 {

018 var result = await \_service.GetAllAsync();

019 return Ok(result);

020 }

021 [HttpGet("{id}")]

022 public async Task<IActionResult> GetById(int id)

023 {

024 var result = await \_service.GetByIdAsync(id);

025 if (result == null)

026 return NotFound();

027 return Ok(result);

028 }

029 [HttpGet("filter")]

030 public async Task<IActionResult> GetFiltered([FromQuery] string? fullName, [FromQuery] string? email)

031 {

032 var result = await \_service.GetFilteredAsync(fullName, email);

033 return Ok(result);

034 }

035 [HttpPost]

036 public async Task<IActionResult> Create([FromBody] UserDto dto)

037 {

038 var result = await \_service.CreateAsync(dto);

039 return CreatedAtAction(nameof(GetById), new { id = result.Id }, result);

040 }

041 [HttpPut("{id}")]

042 public async Task<IActionResult> Update(int id, [FromBody] UserDto dto)

043 {

044 await \_service.UpdateAsync(id, dto);

045 return NoContent();

046 }

047 [HttpDelete("{id}")]

048 public async Task<IActionResult> Delete(int id)

049 {

050 await \_service.DeleteAsync(id);

051 return NoContent();

052 }

053 }

Файл .NETCoreApp,Version=v9.0.AssemblyAttributes.cs:

001 // <autogenerated />

002 using System;

003 using System.Reflection;

004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]

Файл RehearsalStudio.Api.AssemblyInfo.cs:

001 //------------------------------------------------------------------------------

002 // <auto-generated>

003 // This code was generated by a tool.

004 //

005 // Changes to this file may cause incorrect behavior and will be lost if

006 // the code is regenerated.

007 // </auto-generated>

008 //------------------------------------------------------------------------------

009 using System;

010 using System.Reflection;

011 [assembly: System.Reflection.AssemblyCompanyAttribute("RehearsalStudio.Api")]

012 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

013 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

014 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

015 [assembly: System.Reflection.AssemblyProductAttribute("RehearsalStudio.Api")]

016 [assembly: System.Reflection.AssemblyTitleAttribute("RehearsalStudio.Api")]

017 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

018 // Generated by the MSBuild WriteCodeFragment class.

Файл RehearsalStudio.Api.GlobalUsings.g.cs:

001 // <auto-generated/>

002 global using global::Microsoft.AspNetCore.Builder;

003 global using global::Microsoft.AspNetCore.Hosting;

004 global using global::Microsoft.AspNetCore.Http;

005 global using global::Microsoft.AspNetCore.Routing;

006 global using global::Microsoft.Extensions.Configuration;

007 global using global::Microsoft.Extensions.DependencyInjection;

008 global using global::Microsoft.Extensions.Hosting;

009 global using global::Microsoft.Extensions.Logging;

010 global using global::System;

011 global using global::System.Collections.Generic;

012 global using global::System.IO;

013 global using global::System.Linq;

014 global using global::System.Net.Http;

015 global using global::System.Net.Http.Json;

016 global using global::System.Threading;

017 global using global::System.Threading.Tasks;

Файл RehearsalStudio.Api.MvcApplicationPartsAssemblyInfo.cs:

001 //------------------------------------------------------------------------------

002 // <auto-generated>

003 // This code was generated by a tool.

004 //

005 // Changes to this file may cause incorrect behavior and will be lost if

006 // the code is regenerated.

007 // </auto-generated>

008 //------------------------------------------------------------------------------

009 using System;

010 using System.Reflection;

011 [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Microsoft.AspNetCore.OpenApi")]

012 [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Swashbuckle.AspNetCore.SwaggerGen")]

013 // Создано классом WriteCodeFragment MSBuild.

Файл RehearsalStudioDbContext.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Domain.Entities;

003 using Npgsql.EntityFrameworkCore.PostgreSQL;

004 namespace RehearsalStudio.Infrastructure.Data;

005 public class RehearsalStudioDbContext : DbContext

006 {

007 public RehearsalStudioDbContext(DbContextOptions<RehearsalStudioDbContext> options)

008 : base(options)

009 {

010 }

011 public DbSet<RehearsalPoint> RehearsalPoints { get; set; }

012 public DbSet<Room> Rooms { get; set; }

013 public DbSet<Service> Services { get; set; }

014 public DbSet<Equipment> Equipment { get; set; }

015 public DbSet<Staff> Staff { get; set; }

016 public DbSet<User> Users { get; set; }

017 public DbSet<Booking> Bookings { get; set; }

018 public DbSet<ServiceBooking> ServiceBookings { get; set; }

019 public DbSet<EquipmentBooking> EquipmentBookings { get; set; }

020 protected override void OnModelCreating(ModelBuilder modelBuilder)

021 {

022 modelBuilder.HasDefaultSchema("main");

023 // Table names

024 modelBuilder.Entity<RehearsalPoint>().ToTable("rehearsal\_points");

025 modelBuilder.Entity<Room>().ToTable("rooms");

026 modelBuilder.Entity<Service>().ToTable("service");

027 modelBuilder.Entity<Equipment>().ToTable("equipment");

028 modelBuilder.Entity<Staff>().ToTable("staff");

029 modelBuilder.Entity<User>().ToTable("users");

030 modelBuilder.Entity<Booking>().ToTable("booking");

031 modelBuilder.Entity<ServiceBooking>().ToTable("service\_booking");

032 modelBuilder.Entity<EquipmentBooking>().ToTable("equipment\_booking");

033 // Primary keys

034 modelBuilder.Entity<RehearsalPoint>().HasKey(rp => rp.Id);

035 modelBuilder.Entity<Room>().HasKey(r => r.Id);

036 modelBuilder.Entity<Service>().HasKey(s => s.Id);

037 modelBuilder.Entity<Equipment>().HasKey(e => e.Id);

038 modelBuilder.Entity<Staff>().HasKey(s => s.Id);

039 modelBuilder.Entity<User>().HasKey(u => u.Id);

040 modelBuilder.Entity<Booking>().HasKey(b => b.Id);

041 modelBuilder.Entity<ServiceBooking>().HasKey(sb => new { sb.IdService, sb.IdBooking });

042 modelBuilder.Entity<EquipmentBooking>().HasKey(eb => new { eb.IdEquipment, eb.IdBooking });

043 // Auto-increment for IDs

044 modelBuilder.Entity<RehearsalPoint>().Property(rp => rp.Id).ValueGeneratedOnAdd();

045 modelBuilder.Entity<Room>().Property(r => r.Id).ValueGeneratedOnAdd();

046 modelBuilder.Entity<Service>().Property(s => s.Id).ValueGeneratedOnAdd();

047 modelBuilder.Entity<Equipment>().Property(e => e.Id).ValueGeneratedOnAdd();

048 modelBuilder.Entity<Staff>().Property(s => s.Id).ValueGeneratedOnAdd();

049 modelBuilder.Entity<User>().Property(u => u.Id).ValueGeneratedOnAdd();

050 modelBuilder.Entity<Booking>().Property(b => b.Id).ValueGeneratedOnAdd();

051 // No JSON type for Schedule

052 modelBuilder.Entity<RehearsalPoint>()

053 .Property(rp => rp.Schedule)

054 .HasColumnType("text");

055 // Foreign keys with inverse navigation

056 modelBuilder.Entity<Room>()

057 .HasOne(r => r.RehearsalPoint)

058 .WithMany(rp => rp.Rooms)

059 .HasForeignKey(r => r.IdRehearsalPoint)

060 .OnDelete(DeleteBehavior.Cascade);

061 modelBuilder.Entity<Service>()

062 .HasOne(s => s.RehearsalPoint)

063 .WithMany(rp => rp.Services)

064 .HasForeignKey(s => s.IdRehearsalPoint)

065 .OnDelete(DeleteBehavior.Cascade);

066 modelBuilder.Entity<Equipment>()

067 .HasOne(e => e.RehearsalPoint)

068 .WithMany(rp => rp.Equipment)

069 .HasForeignKey(e => e.IdRehearsalPoint)

070 .OnDelete(DeleteBehavior.Cascade);

071 modelBuilder.Entity<Staff>()

072 .HasOne(s => s.RehearsalPoint)

073 .WithMany(rp => rp.Staff)

074 .HasForeignKey(s => s.IdRehearsalPoint)

075 .OnDelete(DeleteBehavior.Cascade);

076 modelBuilder.Entity<Booking>()

077 .HasOne(b => b.Room)

078 .WithMany(r => r.Bookings)

079 .HasForeignKey(b => b.IdRoom)

080 .OnDelete(DeleteBehavior.SetNull);

081 modelBuilder.Entity<Booking>()

082 .HasOne(b => b.User)

083 .WithMany(u => u.Bookings)

084 .HasForeignKey(b => b.IdUser)

085 .OnDelete(DeleteBehavior.Cascade);

086 modelBuilder.Entity<ServiceBooking>()

087 .HasOne(sb => sb.Service)

088 .WithMany(s => s.ServiceBookings)

089 .HasForeignKey(sb => sb.IdService)

090 .OnDelete(DeleteBehavior.Cascade);

091 modelBuilder.Entity<ServiceBooking>()

092 .HasOne(sb => sb.Booking)

093 .WithMany(b => b.ServiceBookings)

094 .HasForeignKey(sb => sb.IdBooking)

095 .OnDelete(DeleteBehavior.Cascade);

096 modelBuilder.Entity<EquipmentBooking>()

097 .HasOne(eb => eb.Equipment)

098 .WithMany(e => e.EquipmentBookings)

099 .HasForeignKey(eb => eb.IdEquipment)

100 .OnDelete(DeleteBehavior.Cascade);

101 modelBuilder.Entity<EquipmentBooking>()

102 .HasOne(eb => eb.Booking)

103 .WithMany(b => b.EquipmentBookings)

104 .HasForeignKey(eb => eb.IdBooking)

105 .OnDelete(DeleteBehavior.Cascade);

106 }

107 }

Файл BookingDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class BookingDto

004 {

005 public int Id { get; set; }

006 public DateTime Time { get; set; }

007 public int? Duration { get; set; }

008 public int Cost { get; set; }

009 public DateTime CreationDate { get; set; }

010 public string Status { get; set; } = string.Empty;

011 public int NumberOfPeople { get; set; }

012 public int? IdRoom { get; set; }

013 public int? IdUser { get; set; }

014 }

Файл EquipmentBookingDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class EquipmentBookingDto

004 {

005 public int IdEquipment { get; set; }

006 public int IdBooking { get; set; }

007 }

Файл EquipmentDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class EquipmentDto

004 {

005 public int Id { get; set; }

006 public string Name { get; set; } = string.Empty;

007 public string Type { get; set; } = string.Empty;

008 public string Brand { get; set; } = string.Empty;

009 public string Model { get; set; } = string.Empty;

010 public string Condition { get; set; } = string.Empty;

011 public int? IdRehearsalPoint { get; set; }

012 }

Файл RehearsalPointDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class RehearsalPointDto

004 {

005 public int Id { get; set; }

006 public float? Rating { get; set; }

007 public string ContactNumber { get; set; } = string.Empty;

008 public string Schedule { get; set; } = string.Empty;

009 public string Name { get; set; } = string.Empty;

010 public string Address { get; set; } = string.Empty;

011 }

Файл RoomDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class RoomDto

004 {

005 public int Id { get; set; }

006 public string Name { get; set; } = string.Empty;

007 public bool AirConditioner { get; set; }

008 public int Price { get; set; }

009 public bool RecordingSupport { get; set; }

010 public int Area { get; set; }

011 public int? IdRehearsalPoint { get; set; }

012 }

Файл ServiceBookingDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class ServiceBookingDto

004 {

005 public int IdService { get; set; }

006 public int IdBooking { get; set; }

007 }

Файл ServiceDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class ServiceDto

004 {

005 public int Id { get; set; }

006 public string Name { get; set; } = string.Empty;

007 public int Price { get; set; }

008 public string Type { get; set; } = string.Empty;

009 public string? Requirements { get; set; }

010 public int? IdRehearsalPoint { get; set; }

011 }

Файл StaffDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class StaffDto

004 {

005 public int Id { get; set; }

006 public string FullName { get; set; } = string.Empty;

007 public string? Address { get; set; }

008 public int? Experience { get; set; }

009 public string Phone { get; set; } = string.Empty;

010 public int Age { get; set; }

011 public int? IdRehearsalPoint { get; set; }

012 }

Файл UserDto.cs:

001 using System;

002 namespace RehearsalStudio.Application.DTOs;

003 public class UserDto

004 {

005 public int Id { get; set; }

006 public string FullName { get; set; } = string.Empty;

007 public string Phone { get; set; } = string.Empty;

008 public string Email { get; set; } = string.Empty;

009 public DateTime RegistrationDate { get; set; }

010 }

Файл IBackupService.cs:

001 using System.Threading.Tasks;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IBackupService

004 {

005 Task<string> CreateDatabaseBackupAsync();

006 Task<string> SaveQueryResultsToFileAsync(string sqlQuery, string fileFormat = "json");

007 }

Файл IBookingRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IBookingRepository

004 {

005 Task<IEnumerable<Booking>> GetAllAsync();

006 Task<Booking?> GetByIdAsync(int id);

007 Task<IEnumerable<Booking>> GetFilteredAsync(string? status, int? idRoom, int? idUser);

008 Task<Booking> AddAsync(Booking booking);

009 Task UpdateAsync(Booking booking);

010 Task DeleteAsync(int id);

011 }

Файл IBookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IBookingService

005 {

006 Task<IEnumerable<BookingDto>> GetAllAsync();

007 Task<BookingDto?> GetByIdAsync(int id);

008 Task<IEnumerable<BookingDto>> GetFilteredAsync(string? status, int? idRoom, int? idUser);

009 Task<BookingDto> CreateAsync(BookingDto dto);

010 Task UpdateAsync(int id, BookingDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IEquipmentBookingRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IEquipmentBookingRepository

004 {

005 Task<IEnumerable<EquipmentBooking>> GetAllAsync();

006 Task<EquipmentBooking?> GetByIdAsync(int idEquipment, int idBooking);

007 Task<IEnumerable<EquipmentBooking>> GetFilteredAsync(int? idEquipment, int? idBooking);

008 Task<EquipmentBooking> AddAsync(EquipmentBooking equipmentBooking);

009 Task DeleteAsync(int idEquipment, int idBooking);

010 }

Файл IEquipmentBookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IEquipmentBookingService

005 {

006 Task<IEnumerable<EquipmentBookingDto>> GetAllAsync();

007 Task<EquipmentBookingDto?> GetByIdAsync(int idEquipment, int idBooking);

008 Task<IEnumerable<EquipmentBookingDto>> GetFilteredAsync(int? idEquipment, int? idBooking);

009 Task<EquipmentBookingDto> CreateAsync(EquipmentBookingDto dto);

010 Task DeleteAsync(int idEquipment, int idBooking);

011 }

Файл IEquipmentRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IEquipmentRepository

004 {

005 Task<IEnumerable<Equipment>> GetAllAsync();

006 Task<Equipment?> GetByIdAsync(int id);

007 Task<IEnumerable<Equipment>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint);

008 Task<Equipment> AddAsync(Equipment equipment);

009 Task UpdateAsync(Equipment equipment);

010 Task DeleteAsync(int id);

011 }

Файл IEquipmentService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IEquipmentService

005 {

006 Task<IEnumerable<EquipmentDto>> GetAllAsync();

007 Task<EquipmentDto?> GetByIdAsync(int id);

008 Task<IEnumerable<EquipmentDto>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint);

009 Task<EquipmentDto> CreateAsync(EquipmentDto dto);

010 Task UpdateAsync(int id, EquipmentDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IRehearsalPointRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IRehearsalPointRepository

004 {

005 Task<IEnumerable<RehearsalPoint>> GetAllAsync();

006 Task<RehearsalPoint?> GetByIdAsync(int id);

007 Task<IEnumerable<RehearsalPoint>> GetFilteredAsync(string? name, float? minRating);

008 Task<RehearsalPoint> AddAsync(RehearsalPoint rehearsalPoint);

009 Task UpdateAsync(RehearsalPoint rehearsalPoint);

010 Task DeleteAsync(int id);

011 }

Файл IRehearsalPointService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IRehearsalPointService

005 {

006 Task<IEnumerable<RehearsalPointDto>> GetAllAsync();

007 Task<RehearsalPointDto?> GetByIdAsync(int id);

008 Task<IEnumerable<RehearsalPointDto>> GetFilteredAsync(string? name, float? minRating);

009 Task<RehearsalPointDto> CreateAsync(RehearsalPointDto dto);

010 Task UpdateAsync(int id, RehearsalPointDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IRoomRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IRoomRepository

004 {

005 Task<IEnumerable<Room>> GetAllAsync();

006 Task<Room?> GetByIdAsync(int id);

007 Task<IEnumerable<Room>> GetFilteredAsync(string? name, int? minPrice, int? idRehearsalPoint);

008 Task<Room> AddAsync(Room room);

009 Task UpdateAsync(Room room);

010 Task DeleteAsync(int id);

011 }

Файл IRoomService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IRoomService

005 {

006 Task<IEnumerable<RoomDto>> GetAllAsync();

007 Task<RoomDto?> GetByIdAsync(int id);

008 Task<IEnumerable<RoomDto>> GetFilteredAsync(string? name, int? minPrice, int? idRehearsalPoint);

009 Task<RoomDto> CreateAsync(RoomDto dto);

010 Task UpdateAsync(int id, RoomDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IServiceBookingRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IServiceBookingRepository

004 {

005 Task<IEnumerable<ServiceBooking>> GetAllAsync();

006 Task<ServiceBooking?> GetByIdAsync(int idService, int idBooking);

007 Task<IEnumerable<ServiceBooking>> GetFilteredAsync(int? idService, int? idBooking);

008 Task<ServiceBooking> AddAsync(ServiceBooking serviceBooking);

009 Task DeleteAsync(int idService, int idBooking);

010 }

Файл IServiceBookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IServiceBookingService

005 {

006 Task<IEnumerable<ServiceBookingDto>> GetAllAsync();

007 Task<ServiceBookingDto?> GetByIdAsync(int idService, int idBooking);

008 Task<IEnumerable<ServiceBookingDto>> GetFilteredAsync(int? idService, int? idBooking);

009 Task<ServiceBookingDto> CreateAsync(ServiceBookingDto dto);

010 Task DeleteAsync(int idService, int idBooking);

011 }

Файл IServiceRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IServiceRepository

004 {

005 Task<IEnumerable<Service>> GetAllAsync();

006 Task<Service?> GetByIdAsync(int id);

007 Task<IEnumerable<Service>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint);

008 Task<Service> AddAsync(Service service);

009 Task UpdateAsync(Service service);

010 Task DeleteAsync(int id);

011 }

Файл IServiceService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IServiceService

005 {

006 Task<IEnumerable<ServiceDto>> GetAllAsync();

007 Task<ServiceDto?> GetByIdAsync(int id);

008 Task<IEnumerable<ServiceDto>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint);

009 Task<ServiceDto> CreateAsync(ServiceDto dto);

010 Task UpdateAsync(int id, ServiceDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IStaffRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IStaffRepository

004 {

005 Task<IEnumerable<Staff>> GetAllAsync();

006 Task<Staff?> GetByIdAsync(int id);

007 Task<IEnumerable<Staff>> GetFilteredAsync(string? fullName, int? minAge, int? idRehearsalPoint);

008 Task<Staff> AddAsync(Staff staff);

009 Task UpdateAsync(Staff staff);

010 Task DeleteAsync(int id);

011 }

Файл IStaffService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IStaffService

005 {

006 Task<IEnumerable<StaffDto>> GetAllAsync();

007 Task<StaffDto?> GetByIdAsync(int id);

008 Task<IEnumerable<StaffDto>> GetFilteredAsync(string? fullName, int? minAge, int? idRehearsalPoint);

009 Task<StaffDto> CreateAsync(StaffDto dto);

010 Task UpdateAsync(int id, StaffDto dto);

011 Task DeleteAsync(int id);

012 }

Файл IUserRepository.cs:

001 using RehearsalStudio.Domain.Entities;

002 namespace RehearsalStudio.Application.Interfaces;

003 public interface IUserRepository

004 {

005 Task<IEnumerable<User>> GetAllAsync();

006 Task<User?> GetByIdAsync(int id);

007 Task<IEnumerable<User>> GetFilteredAsync(string? fullName, string? email);

008 Task<User> AddAsync(User user);

009 Task UpdateAsync(User user);

010 Task DeleteAsync(int id);

011 }

Файл IUserService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using System.Threading.Tasks;

003 namespace RehearsalStudio.Application.Interfaces;

004 public interface IUserService

005 {

006 Task<IEnumerable<UserDto>> GetAllAsync();

007 Task<UserDto?> GetByIdAsync(int id);

008 Task<IEnumerable<UserDto>> GetFilteredAsync(string? fullName, string? email);

009 Task<UserDto> CreateAsync(UserDto dto);

010 Task UpdateAsync(int id, UserDto dto);

011 Task DeleteAsync(int id);

012 }

Файл 20250910175704\_InitialCreate.cs:

001 ﻿using System;

002 using Microsoft.EntityFrameworkCore.Migrations;

003 using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

004 #nullable disable

005 namespace RehearsalStudio.Application.Migrations

006 {

007 /// <inheritdoc />

008 public partial class InitialCreate : Migration

009 {

010 /// <inheritdoc />

011 protected override void Up(MigrationBuilder migrationBuilder)

012 {

013 migrationBuilder.EnsureSchema(

014 name: "main");

015 migrationBuilder.CreateTable(

016 name: "rehearsal\_points",

017 schema: "main",

018 columns: table => new

019 {

020 Id = table.Column<int>(type: "integer", nullable: false)

021 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

022 Rating = table.Column<float>(type: "real", nullable: true),

023 ContactNumber = table.Column<string>(type: "text", nullable: false),

024 Schedule = table.Column<string>(type: "text", nullable: false),

025 Name = table.Column<string>(type: "text", nullable: false),

026 Address = table.Column<string>(type: "text", nullable: false)

027 },

028 constraints: table =>

029 {

030 table.PrimaryKey("PK\_rehearsal\_points", x => x.Id);

031 });

032 migrationBuilder.CreateTable(

033 name: "users",

034 schema: "main",

035 columns: table => new

036 {

037 Id = table.Column<int>(type: "integer", nullable: false)

038 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

039 FullName = table.Column<string>(type: "text", nullable: false),

040 Phone = table.Column<string>(type: "text", nullable: false),

041 Email = table.Column<string>(type: "text", nullable: false),

042 RegistrationDate = table.Column<DateTime>(type: "timestamp with time zone", nullable: false)

043 },

044 constraints: table =>

045 {

046 table.PrimaryKey("PK\_users", x => x.Id);

047 });

048 migrationBuilder.CreateTable(

049 name: "equipment",

050 schema: "main",

051 columns: table => new

052 {

053 Id = table.Column<int>(type: "integer", nullable: false)

054 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

055 Name = table.Column<string>(type: "text", nullable: false),

056 Type = table.Column<string>(type: "text", nullable: false),

057 Brand = table.Column<string>(type: "text", nullable: false),

058 Model = table.Column<string>(type: "text", nullable: false),

059 Condition = table.Column<string>(type: "text", nullable: false),

060 IdRehearsalPoint = table.Column<int>(type: "integer", nullable: true)

061 },

062 constraints: table =>

063 {

064 table.PrimaryKey("PK\_equipment", x => x.Id);

065 table.ForeignKey(

066 name: "FK\_equipment\_rehearsal\_points\_IdRehearsalPoint",

067 column: x => x.IdRehearsalPoint,

068 principalSchema: "main",

069 principalTable: "rehearsal\_points",

070 principalColumn: "Id",

071 onDelete: ReferentialAction.Cascade);

072 });

073 migrationBuilder.CreateTable(

074 name: "rooms",

075 schema: "main",

076 columns: table => new

077 {

078 Id = table.Column<int>(type: "integer", nullable: false)

079 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

080 Name = table.Column<string>(type: "text", nullable: false),

081 AirConditioner = table.Column<bool>(type: "boolean", nullable: false),

082 Price = table.Column<int>(type: "integer", nullable: false),

083 RecordingSupport = table.Column<bool>(type: "boolean", nullable: false),

084 Area = table.Column<int>(type: "integer", nullable: false),

085 IdRehearsalPoint = table.Column<int>(type: "integer", nullable: true)

086 },

087 constraints: table =>

088 {

089 table.PrimaryKey("PK\_rooms", x => x.Id);

090 table.ForeignKey(

091 name: "FK\_rooms\_rehearsal\_points\_IdRehearsalPoint",

092 column: x => x.IdRehearsalPoint,

093 principalSchema: "main",

094 principalTable: "rehearsal\_points",

095 principalColumn: "Id",

096 onDelete: ReferentialAction.Cascade);

097 });

098 migrationBuilder.CreateTable(

099 name: "service",

100 schema: "main",

101 columns: table => new

102 {

103 Id = table.Column<int>(type: "integer", nullable: false)

104 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

105 Name = table.Column<string>(type: "text", nullable: false),

106 Price = table.Column<int>(type: "integer", nullable: false),

107 Type = table.Column<string>(type: "text", nullable: false),

108 Requirements = table.Column<string>(type: "text", nullable: true),

109 IdRehearsalPoint = table.Column<int>(type: "integer", nullable: true)

110 },

111 constraints: table =>

112 {

113 table.PrimaryKey("PK\_service", x => x.Id);

114 table.ForeignKey(

115 name: "FK\_service\_rehearsal\_points\_IdRehearsalPoint",

116 column: x => x.IdRehearsalPoint,

117 principalSchema: "main",

118 principalTable: "rehearsal\_points",

119 principalColumn: "Id",

120 onDelete: ReferentialAction.Cascade);

121 });

122 migrationBuilder.CreateTable(

123 name: "staff",

124 schema: "main",

125 columns: table => new

126 {

127 Id = table.Column<int>(type: "integer", nullable: false)

128 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

129 FullName = table.Column<string>(type: "text", nullable: false),

130 Address = table.Column<string>(type: "text", nullable: true),

131 Experience = table.Column<int>(type: "integer", nullable: true),

132 Phone = table.Column<string>(type: "text", nullable: false),

133 Age = table.Column<int>(type: "integer", nullable: false),

134 IdRehearsalPoint = table.Column<int>(type: "integer", nullable: true)

135 },

136 constraints: table =>

137 {

138 table.PrimaryKey("PK\_staff", x => x.Id);

139 table.ForeignKey(

140 name: "FK\_staff\_rehearsal\_points\_IdRehearsalPoint",

141 column: x => x.IdRehearsalPoint,

142 principalSchema: "main",

143 principalTable: "rehearsal\_points",

144 principalColumn: "Id",

145 onDelete: ReferentialAction.Cascade);

146 });

147 migrationBuilder.CreateTable(

148 name: "booking",

149 schema: "main",

150 columns: table => new

151 {

152 Id = table.Column<int>(type: "integer", nullable: false)

153 .Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

154 Time = table.Column<DateTime>(type: "timestamp with time zone", nullable: false),

155 Duration = table.Column<int>(type: "integer", nullable: true),

156 Cost = table.Column<int>(type: "integer", nullable: false),

157 CreationDate = table.Column<DateTime>(type: "timestamp with time zone", nullable: false),

158 Status = table.Column<string>(type: "text", nullable: false),

159 NumberOfPeople = table.Column<int>(type: "integer", nullable: false),

160 IdRoom = table.Column<int>(type: "integer", nullable: true),

161 IdUser = table.Column<int>(type: "integer", nullable: true)

162 },

163 constraints: table =>

164 {

165 table.PrimaryKey("PK\_booking", x => x.Id);

166 table.ForeignKey(

167 name: "FK\_booking\_rooms\_IdRoom",

168 column: x => x.IdRoom,

169 principalSchema: "main",

170 principalTable: "rooms",

171 principalColumn: "Id",

172 onDelete: ReferentialAction.SetNull);

173 table.ForeignKey(

174 name: "FK\_booking\_users\_IdUser",

175 column: x => x.IdUser,

176 principalSchema: "main",

177 principalTable: "users",

178 principalColumn: "Id",

179 onDelete: ReferentialAction.Cascade);

180 });

181 migrationBuilder.CreateTable(

182 name: "equipment\_booking",

183 schema: "main",

184 columns: table => new

185 {

186 IdEquipment = table.Column<int>(type: "integer", nullable: false),

187 IdBooking = table.Column<int>(type: "integer", nullable: false)

188 },

189 constraints: table =>

190 {

191 table.PrimaryKey("PK\_equipment\_booking", x => new { x.IdEquipment, x.IdBooking });

192 table.ForeignKey(

193 name: "FK\_equipment\_booking\_booking\_IdBooking",

194 column: x => x.IdBooking,

195 principalSchema: "main",

196 principalTable: "booking",

197 principalColumn: "Id",

198 onDelete: ReferentialAction.Cascade);

199 table.ForeignKey(

200 name: "FK\_equipment\_booking\_equipment\_IdEquipment",

201 column: x => x.IdEquipment,

202 principalSchema: "main",

203 principalTable: "equipment",

204 principalColumn: "Id",

205 onDelete: ReferentialAction.Cascade);

206 });

207 migrationBuilder.CreateTable(

208 name: "service\_booking",

209 schema: "main",

210 columns: table => new

211 {

212 IdService = table.Column<int>(type: "integer", nullable: false),

213 IdBooking = table.Column<int>(type: "integer", nullable: false)

214 },

215 constraints: table =>

216 {

217 table.PrimaryKey("PK\_service\_booking", x => new { x.IdService, x.IdBooking });

218 table.ForeignKey(

219 name: "FK\_service\_booking\_booking\_IdBooking",

220 column: x => x.IdBooking,

221 principalSchema: "main",

222 principalTable: "booking",

223 principalColumn: "Id",

224 onDelete: ReferentialAction.Cascade);

225 table.ForeignKey(

226 name: "FK\_service\_booking\_service\_IdService",

227 column: x => x.IdService,

228 principalSchema: "main",

229 principalTable: "service",

230 principalColumn: "Id",

231 onDelete: ReferentialAction.Cascade);

232 });

233 migrationBuilder.CreateIndex(

234 name: "IX\_booking\_IdRoom",

235 schema: "main",

236 table: "booking",

237 column: "IdRoom");

238 migrationBuilder.CreateIndex(

239 name: "IX\_booking\_IdUser",

240 schema: "main",

241 table: "booking",

242 column: "IdUser");

243 migrationBuilder.CreateIndex(

244 name: "IX\_equipment\_IdRehearsalPoint",

245 schema: "main",

246 table: "equipment",

247 column: "IdRehearsalPoint");

248 migrationBuilder.CreateIndex(

249 name: "IX\_equipment\_booking\_IdBooking",

250 schema: "main",

251 table: "equipment\_booking",

252 column: "IdBooking");

253 migrationBuilder.CreateIndex(

254 name: "IX\_rooms\_IdRehearsalPoint",

255 schema: "main",

256 table: "rooms",

257 column: "IdRehearsalPoint");

258 migrationBuilder.CreateIndex(

259 name: "IX\_service\_IdRehearsalPoint",

260 schema: "main",

261 table: "service",

262 column: "IdRehearsalPoint");

263 migrationBuilder.CreateIndex(

264 name: "IX\_service\_booking\_IdBooking",

265 schema: "main",

266 table: "service\_booking",

267 column: "IdBooking");

268 migrationBuilder.CreateIndex(

269 name: "IX\_staff\_IdRehearsalPoint",

270 schema: "main",

271 table: "staff",

272 column: "IdRehearsalPoint");

273 }

274 /// <inheritdoc />

275 protected override void Down(MigrationBuilder migrationBuilder)

276 {

277 migrationBuilder.DropTable(

278 name: "equipment\_booking",

279 schema: "main");

280 migrationBuilder.DropTable(

281 name: "service\_booking",

282 schema: "main");

283 migrationBuilder.DropTable(

284 name: "staff",

285 schema: "main");

286 migrationBuilder.DropTable(

287 name: "equipment",

288 schema: "main");

289 migrationBuilder.DropTable(

290 name: "booking",

291 schema: "main");

292 migrationBuilder.DropTable(

293 name: "service",

294 schema: "main");

295 migrationBuilder.DropTable(

296 name: "rooms",

297 schema: "main");

298 migrationBuilder.DropTable(

299 name: "users",

300 schema: "main");

301 migrationBuilder.DropTable(

302 name: "rehearsal\_points",

303 schema: "main");

304 }

305 }

306 }

Файл 20250910175704\_InitialCreate.Designer.cs:

001 ﻿// <auto-generated />

002 using System;

003 using Microsoft.EntityFrameworkCore;

004 using Microsoft.EntityFrameworkCore.Infrastructure;

005 using Microsoft.EntityFrameworkCore.Migrations;

006 using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

007 using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

008 using RehearsalStudio.Infrastructure.Data;

009 #nullable disable

010 namespace RehearsalStudio.Application.Migrations

011 {

012 [DbContext(typeof(RehearsalStudioDbContext))]

013 [Migration("20250910175704\_InitialCreate")]

014 partial class InitialCreate

015 {

016 /// <inheritdoc />

017 protected override void BuildTargetModel(ModelBuilder modelBuilder)

018 {

019 #pragma warning disable 612, 618

020 modelBuilder

021 .HasDefaultSchema("main")

022 .HasAnnotation("ProductVersion", "9.0.9")

023 .HasAnnotation("Relational:MaxIdentifierLength", 63);

024 NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

025 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

026 {

027 b.Property<int>("Id")

028 .ValueGeneratedOnAdd()

029 .HasColumnType("integer");

030 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

031 b.Property<int>("Cost")

032 .HasColumnType("integer");

033 b.Property<DateTime>("CreationDate")

034 .HasColumnType("timestamp with time zone");

035 b.Property<int?>("Duration")

036 .HasColumnType("integer");

037 b.Property<int?>("IdRoom")

038 .HasColumnType("integer");

039 b.Property<int?>("IdUser")

040 .HasColumnType("integer");

041 b.Property<int>("NumberOfPeople")

042 .HasColumnType("integer");

043 b.Property<string>("Status")

044 .IsRequired()

045 .HasColumnType("text");

046 b.Property<DateTime>("Time")

047 .HasColumnType("timestamp with time zone");

048 b.HasKey("Id");

049 b.HasIndex("IdRoom");

050 b.HasIndex("IdUser");

051 b.ToTable("booking", "main");

052 });

053 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

054 {

055 b.Property<int>("Id")

056 .ValueGeneratedOnAdd()

057 .HasColumnType("integer");

058 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

059 b.Property<string>("Brand")

060 .IsRequired()

061 .HasColumnType("text");

062 b.Property<string>("Condition")

063 .IsRequired()

064 .HasColumnType("text");

065 b.Property<int?>("IdRehearsalPoint")

066 .HasColumnType("integer");

067 b.Property<string>("Model")

068 .IsRequired()

069 .HasColumnType("text");

070 b.Property<string>("Name")

071 .IsRequired()

072 .HasColumnType("text");

073 b.Property<string>("Type")

074 .IsRequired()

075 .HasColumnType("text");

076 b.HasKey("Id");

077 b.HasIndex("IdRehearsalPoint");

078 b.ToTable("equipment", "main");

079 });

080 modelBuilder.Entity("RehearsalStudio.Domain.Entities.EquipmentBooking", b =>

081 {

082 b.Property<int>("IdEquipment")

083 .HasColumnType("integer")

084 .HasColumnOrder(0);

085 b.Property<int>("IdBooking")

086 .HasColumnType("integer")

087 .HasColumnOrder(1);

088 b.HasKey("IdEquipment", "IdBooking");

089 b.HasIndex("IdBooking");

090 b.ToTable("equipment\_booking", "main");

091 });

092 modelBuilder.Entity("RehearsalStudio.Domain.Entities.RehearsalPoint", b =>

093 {

094 b.Property<int>("Id")

095 .ValueGeneratedOnAdd()

096 .HasColumnType("integer");

097 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

098 b.Property<string>("Address")

099 .IsRequired()

100 .HasColumnType("text");

101 b.Property<string>("ContactNumber")

102 .IsRequired()

103 .HasColumnType("text");

104 b.Property<string>("Name")

105 .IsRequired()

106 .HasColumnType("text");

107 b.Property<float?>("Rating")

108 .HasColumnType("real");

109 b.Property<string>("Schedule")

110 .IsRequired()

111 .HasColumnType("text");

112 b.HasKey("Id");

113 b.ToTable("rehearsal\_points", "main");

114 });

115 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

116 {

117 b.Property<int>("Id")

118 .ValueGeneratedOnAdd()

119 .HasColumnType("integer");

120 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

121 b.Property<bool>("AirConditioner")

122 .HasColumnType("boolean");

123 b.Property<int>("Area")

124 .HasColumnType("integer");

125 b.Property<int?>("IdRehearsalPoint")

126 .HasColumnType("integer");

127 b.Property<string>("Name")

128 .IsRequired()

129 .HasColumnType("text");

130 b.Property<int>("Price")

131 .HasColumnType("integer");

132 b.Property<bool>("RecordingSupport")

133 .HasColumnType("boolean");

134 b.HasKey("Id");

135 b.HasIndex("IdRehearsalPoint");

136 b.ToTable("rooms", "main");

137 });

138 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

139 {

140 b.Property<int>("Id")

141 .ValueGeneratedOnAdd()

142 .HasColumnType("integer");

143 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

144 b.Property<int?>("IdRehearsalPoint")

145 .HasColumnType("integer");

146 b.Property<string>("Name")

147 .IsRequired()

148 .HasColumnType("text");

149 b.Property<int>("Price")

150 .HasColumnType("integer");

151 b.Property<string>("Requirements")

152 .HasColumnType("text");

153 b.Property<string>("Type")

154 .IsRequired()

155 .HasColumnType("text");

156 b.HasKey("Id");

157 b.HasIndex("IdRehearsalPoint");

158 b.ToTable("service", "main");

159 });

160 modelBuilder.Entity("RehearsalStudio.Domain.Entities.ServiceBooking", b =>

161 {

162 b.Property<int>("IdService")

163 .HasColumnType("integer")

164 .HasColumnOrder(0);

165 b.Property<int>("IdBooking")

166 .HasColumnType("integer")

167 .HasColumnOrder(1);

168 b.HasKey("IdService", "IdBooking");

169 b.HasIndex("IdBooking");

170 b.ToTable("service\_booking", "main");

171 });

172 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Staff", b =>

173 {

174 b.Property<int>("Id")

175 .ValueGeneratedOnAdd()

176 .HasColumnType("integer");

177 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

178 b.Property<string>("Address")

179 .HasColumnType("text");

180 b.Property<int>("Age")

181 .HasColumnType("integer");

182 b.Property<int?>("Experience")

183 .HasColumnType("integer");

184 b.Property<string>("FullName")

185 .IsRequired()

186 .HasColumnType("text");

187 b.Property<int?>("IdRehearsalPoint")

188 .HasColumnType("integer");

189 b.Property<string>("Phone")

190 .IsRequired()

191 .HasColumnType("text");

192 b.HasKey("Id");

193 b.HasIndex("IdRehearsalPoint");

194 b.ToTable("staff", "main");

195 });

196 modelBuilder.Entity("RehearsalStudio.Domain.Entities.User", b =>

197 {

198 b.Property<int>("Id")

199 .ValueGeneratedOnAdd()

200 .HasColumnType("integer");

201 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

202 b.Property<string>("Email")

203 .IsRequired()

204 .HasColumnType("text");

205 b.Property<string>("FullName")

206 .IsRequired()

207 .HasColumnType("text");

208 b.Property<string>("Phone")

209 .IsRequired()

210 .HasColumnType("text");

211 b.Property<DateTime>("RegistrationDate")

212 .HasColumnType("timestamp with time zone");

213 b.HasKey("Id");

214 b.ToTable("users", "main");

215 });

216 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

217 {

218 b.HasOne("RehearsalStudio.Domain.Entities.Room", "Room")

219 .WithMany("Bookings")

220 .HasForeignKey("IdRoom")

221 .OnDelete(DeleteBehavior.SetNull);

222 b.HasOne("RehearsalStudio.Domain.Entities.User", "User")

223 .WithMany("Bookings")

224 .HasForeignKey("IdUser")

225 .OnDelete(DeleteBehavior.Cascade);

226 b.Navigation("Room");

227 b.Navigation("User");

228 });

229 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

230 {

231 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

232 .WithMany("Equipment")

233 .HasForeignKey("IdRehearsalPoint")

234 .OnDelete(DeleteBehavior.Cascade);

235 b.Navigation("RehearsalPoint");

236 });

237 modelBuilder.Entity("RehearsalStudio.Domain.Entities.EquipmentBooking", b =>

238 {

239 b.HasOne("RehearsalStudio.Domain.Entities.Booking", "Booking")

240 .WithMany("EquipmentBookings")

241 .HasForeignKey("IdBooking")

242 .OnDelete(DeleteBehavior.Cascade)

243 .IsRequired();

244 b.HasOne("RehearsalStudio.Domain.Entities.Equipment", "Equipment")

245 .WithMany("EquipmentBookings")

246 .HasForeignKey("IdEquipment")

247 .OnDelete(DeleteBehavior.Cascade)

248 .IsRequired();

249 b.Navigation("Booking");

250 b.Navigation("Equipment");

251 });

252 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

253 {

254 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

255 .WithMany("Rooms")

256 .HasForeignKey("IdRehearsalPoint")

257 .OnDelete(DeleteBehavior.Cascade);

258 b.Navigation("RehearsalPoint");

259 });

260 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

261 {

262 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

263 .WithMany("Services")

264 .HasForeignKey("IdRehearsalPoint")

265 .OnDelete(DeleteBehavior.Cascade);

266 b.Navigation("RehearsalPoint");

267 });

268 modelBuilder.Entity("RehearsalStudio.Domain.Entities.ServiceBooking", b =>

269 {

270 b.HasOne("RehearsalStudio.Domain.Entities.Booking", "Booking")

271 .WithMany("ServiceBookings")

272 .HasForeignKey("IdBooking")

273 .OnDelete(DeleteBehavior.Cascade)

274 .IsRequired();

275 b.HasOne("RehearsalStudio.Domain.Entities.Service", "Service")

276 .WithMany("ServiceBookings")

277 .HasForeignKey("IdService")

278 .OnDelete(DeleteBehavior.Cascade)

279 .IsRequired();

280 b.Navigation("Booking");

281 b.Navigation("Service");

282 });

283 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Staff", b =>

284 {

285 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

286 .WithMany("Staff")

287 .HasForeignKey("IdRehearsalPoint")

288 .OnDelete(DeleteBehavior.Cascade);

289 b.Navigation("RehearsalPoint");

290 });

291 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

292 {

293 b.Navigation("EquipmentBookings");

294 b.Navigation("ServiceBookings");

295 });

296 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

297 {

298 b.Navigation("EquipmentBookings");

299 });

300 modelBuilder.Entity("RehearsalStudio.Domain.Entities.RehearsalPoint", b =>

301 {

302 b.Navigation("Equipment");

303 b.Navigation("Rooms");

304 b.Navigation("Services");

305 b.Navigation("Staff");

306 });

307 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

308 {

309 b.Navigation("Bookings");

310 });

311 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

312 {

313 b.Navigation("ServiceBookings");

314 });

315 modelBuilder.Entity("RehearsalStudio.Domain.Entities.User", b =>

316 {

317 b.Navigation("Bookings");

318 });

319 #pragma warning restore 612, 618

320 }

321 }

322 }

Файл RehearsalStudioDbContextModelSnapshot.cs:

001 ﻿// <auto-generated />

002 using System;

003 using Microsoft.EntityFrameworkCore;

004 using Microsoft.EntityFrameworkCore.Infrastructure;

005 using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

006 using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

007 using RehearsalStudio.Infrastructure.Data;

008 #nullable disable

009 namespace RehearsalStudio.Application.Migrations

010 {

011 [DbContext(typeof(RehearsalStudioDbContext))]

012 partial class RehearsalStudioDbContextModelSnapshot : ModelSnapshot

013 {

014 protected override void BuildModel(ModelBuilder modelBuilder)

015 {

016 #pragma warning disable 612, 618

017 modelBuilder

018 .HasDefaultSchema("main")

019 .HasAnnotation("ProductVersion", "9.0.9")

020 .HasAnnotation("Relational:MaxIdentifierLength", 63);

021 NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

022 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

023 {

024 b.Property<int>("Id")

025 .ValueGeneratedOnAdd()

026 .HasColumnType("integer");

027 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

028 b.Property<int>("Cost")

029 .HasColumnType("integer");

030 b.Property<DateTime>("CreationDate")

031 .HasColumnType("timestamp with time zone");

032 b.Property<int?>("Duration")

033 .HasColumnType("integer");

034 b.Property<int?>("IdRoom")

035 .HasColumnType("integer");

036 b.Property<int?>("IdUser")

037 .HasColumnType("integer");

038 b.Property<int>("NumberOfPeople")

039 .HasColumnType("integer");

040 b.Property<string>("Status")

041 .IsRequired()

042 .HasColumnType("text");

043 b.Property<DateTime>("Time")

044 .HasColumnType("timestamp with time zone");

045 b.HasKey("Id");

046 b.HasIndex("IdRoom");

047 b.HasIndex("IdUser");

048 b.ToTable("booking", "main");

049 });

050 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

051 {

052 b.Property<int>("Id")

053 .ValueGeneratedOnAdd()

054 .HasColumnType("integer");

055 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

056 b.Property<string>("Brand")

057 .IsRequired()

058 .HasColumnType("text");

059 b.Property<string>("Condition")

060 .IsRequired()

061 .HasColumnType("text");

062 b.Property<int?>("IdRehearsalPoint")

063 .HasColumnType("integer");

064 b.Property<string>("Model")

065 .IsRequired()

066 .HasColumnType("text");

067 b.Property<string>("Name")

068 .IsRequired()

069 .HasColumnType("text");

070 b.Property<string>("Type")

071 .IsRequired()

072 .HasColumnType("text");

073 b.HasKey("Id");

074 b.HasIndex("IdRehearsalPoint");

075 b.ToTable("equipment", "main");

076 });

077 modelBuilder.Entity("RehearsalStudio.Domain.Entities.EquipmentBooking", b =>

078 {

079 b.Property<int>("IdEquipment")

080 .HasColumnType("integer")

081 .HasColumnOrder(0);

082 b.Property<int>("IdBooking")

083 .HasColumnType("integer")

084 .HasColumnOrder(1);

085 b.HasKey("IdEquipment", "IdBooking");

086 b.HasIndex("IdBooking");

087 b.ToTable("equipment\_booking", "main");

088 });

089 modelBuilder.Entity("RehearsalStudio.Domain.Entities.RehearsalPoint", b =>

090 {

091 b.Property<int>("Id")

092 .ValueGeneratedOnAdd()

093 .HasColumnType("integer");

094 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

095 b.Property<string>("Address")

096 .IsRequired()

097 .HasColumnType("text");

098 b.Property<string>("ContactNumber")

099 .IsRequired()

100 .HasColumnType("text");

101 b.Property<string>("Name")

102 .IsRequired()

103 .HasColumnType("text");

104 b.Property<float?>("Rating")

105 .HasColumnType("real");

106 b.Property<string>("Schedule")

107 .IsRequired()

108 .HasColumnType("text");

109 b.HasKey("Id");

110 b.ToTable("rehearsal\_points", "main");

111 });

112 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

113 {

114 b.Property<int>("Id")

115 .ValueGeneratedOnAdd()

116 .HasColumnType("integer");

117 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

118 b.Property<bool>("AirConditioner")

119 .HasColumnType("boolean");

120 b.Property<int>("Area")

121 .HasColumnType("integer");

122 b.Property<int?>("IdRehearsalPoint")

123 .HasColumnType("integer");

124 b.Property<string>("Name")

125 .IsRequired()

126 .HasColumnType("text");

127 b.Property<int>("Price")

128 .HasColumnType("integer");

129 b.Property<bool>("RecordingSupport")

130 .HasColumnType("boolean");

131 b.HasKey("Id");

132 b.HasIndex("IdRehearsalPoint");

133 b.ToTable("rooms", "main");

134 });

135 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

136 {

137 b.Property<int>("Id")

138 .ValueGeneratedOnAdd()

139 .HasColumnType("integer");

140 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

141 b.Property<int?>("IdRehearsalPoint")

142 .HasColumnType("integer");

143 b.Property<string>("Name")

144 .IsRequired()

145 .HasColumnType("text");

146 b.Property<int>("Price")

147 .HasColumnType("integer");

148 b.Property<string>("Requirements")

149 .HasColumnType("text");

150 b.Property<string>("Type")

151 .IsRequired()

152 .HasColumnType("text");

153 b.HasKey("Id");

154 b.HasIndex("IdRehearsalPoint");

155 b.ToTable("service", "main");

156 });

157 modelBuilder.Entity("RehearsalStudio.Domain.Entities.ServiceBooking", b =>

158 {

159 b.Property<int>("IdService")

160 .HasColumnType("integer")

161 .HasColumnOrder(0);

162 b.Property<int>("IdBooking")

163 .HasColumnType("integer")

164 .HasColumnOrder(1);

165 b.HasKey("IdService", "IdBooking");

166 b.HasIndex("IdBooking");

167 b.ToTable("service\_booking", "main");

168 });

169 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Staff", b =>

170 {

171 b.Property<int>("Id")

172 .ValueGeneratedOnAdd()

173 .HasColumnType("integer");

174 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

175 b.Property<string>("Address")

176 .HasColumnType("text");

177 b.Property<int>("Age")

178 .HasColumnType("integer");

179 b.Property<int?>("Experience")

180 .HasColumnType("integer");

181 b.Property<string>("FullName")

182 .IsRequired()

183 .HasColumnType("text");

184 b.Property<int?>("IdRehearsalPoint")

185 .HasColumnType("integer");

186 b.Property<string>("Phone")

187 .IsRequired()

188 .HasColumnType("text");

189 b.HasKey("Id");

190 b.HasIndex("IdRehearsalPoint");

191 b.ToTable("staff", "main");

192 });

193 modelBuilder.Entity("RehearsalStudio.Domain.Entities.User", b =>

194 {

195 b.Property<int>("Id")

196 .ValueGeneratedOnAdd()

197 .HasColumnType("integer");

198 NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("Id"));

199 b.Property<string>("Email")

200 .IsRequired()

201 .HasColumnType("text");

202 b.Property<string>("FullName")

203 .IsRequired()

204 .HasColumnType("text");

205 b.Property<string>("Phone")

206 .IsRequired()

207 .HasColumnType("text");

208 b.Property<DateTime>("RegistrationDate")

209 .HasColumnType("timestamp with time zone");

210 b.HasKey("Id");

211 b.ToTable("users", "main");

212 });

213 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

214 {

215 b.HasOne("RehearsalStudio.Domain.Entities.Room", "Room")

216 .WithMany("Bookings")

217 .HasForeignKey("IdRoom")

218 .OnDelete(DeleteBehavior.SetNull);

219 b.HasOne("RehearsalStudio.Domain.Entities.User", "User")

220 .WithMany("Bookings")

221 .HasForeignKey("IdUser")

222 .OnDelete(DeleteBehavior.Cascade);

223 b.Navigation("Room");

224 b.Navigation("User");

225 });

226 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

227 {

228 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

229 .WithMany("Equipment")

230 .HasForeignKey("IdRehearsalPoint")

231 .OnDelete(DeleteBehavior.Cascade);

232 b.Navigation("RehearsalPoint");

233 });

234 modelBuilder.Entity("RehearsalStudio.Domain.Entities.EquipmentBooking", b =>

235 {

236 b.HasOne("RehearsalStudio.Domain.Entities.Booking", "Booking")

237 .WithMany("EquipmentBookings")

238 .HasForeignKey("IdBooking")

239 .OnDelete(DeleteBehavior.Cascade)

240 .IsRequired();

241 b.HasOne("RehearsalStudio.Domain.Entities.Equipment", "Equipment")

242 .WithMany("EquipmentBookings")

243 .HasForeignKey("IdEquipment")

244 .OnDelete(DeleteBehavior.Cascade)

245 .IsRequired();

246 b.Navigation("Booking");

247 b.Navigation("Equipment");

248 });

249 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

250 {

251 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

252 .WithMany("Rooms")

253 .HasForeignKey("IdRehearsalPoint")

254 .OnDelete(DeleteBehavior.Cascade);

255 b.Navigation("RehearsalPoint");

256 });

257 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

258 {

259 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

260 .WithMany("Services")

261 .HasForeignKey("IdRehearsalPoint")

262 .OnDelete(DeleteBehavior.Cascade);

263 b.Navigation("RehearsalPoint");

264 });

265 modelBuilder.Entity("RehearsalStudio.Domain.Entities.ServiceBooking", b =>

266 {

267 b.HasOne("RehearsalStudio.Domain.Entities.Booking", "Booking")

268 .WithMany("ServiceBookings")

269 .HasForeignKey("IdBooking")

270 .OnDelete(DeleteBehavior.Cascade)

271 .IsRequired();

272 b.HasOne("RehearsalStudio.Domain.Entities.Service", "Service")

273 .WithMany("ServiceBookings")

274 .HasForeignKey("IdService")

275 .OnDelete(DeleteBehavior.Cascade)

276 .IsRequired();

277 b.Navigation("Booking");

278 b.Navigation("Service");

279 });

280 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Staff", b =>

281 {

282 b.HasOne("RehearsalStudio.Domain.Entities.RehearsalPoint", "RehearsalPoint")

283 .WithMany("Staff")

284 .HasForeignKey("IdRehearsalPoint")

285 .OnDelete(DeleteBehavior.Cascade);

286 b.Navigation("RehearsalPoint");

287 });

288 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Booking", b =>

289 {

290 b.Navigation("EquipmentBookings");

291 b.Navigation("ServiceBookings");

292 });

293 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Equipment", b =>

294 {

295 b.Navigation("EquipmentBookings");

296 });

297 modelBuilder.Entity("RehearsalStudio.Domain.Entities.RehearsalPoint", b =>

298 {

299 b.Navigation("Equipment");

300 b.Navigation("Rooms");

301 b.Navigation("Services");

302 b.Navigation("Staff");

303 });

304 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Room", b =>

305 {

306 b.Navigation("Bookings");

307 });

308 modelBuilder.Entity("RehearsalStudio.Domain.Entities.Service", b =>

309 {

310 b.Navigation("ServiceBookings");

311 });

312 modelBuilder.Entity("RehearsalStudio.Domain.Entities.User", b =>

313 {

314 b.Navigation("Bookings");

315 });

316 #pragma warning restore 612, 618

317 }

318 }

319 }

Файл .NETCoreApp,Version=v9.0.AssemblyAttributes.cs:

001 // <autogenerated />

002 using System;

003 using System.Reflection;

004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]

Файл RehearsalStudio.Application.AssemblyInfo.cs:

001 //------------------------------------------------------------------------------

002 // <auto-generated>

003 // This code was generated by a tool.

004 //

005 // Changes to this file may cause incorrect behavior and will be lost if

006 // the code is regenerated.

007 // </auto-generated>

008 //------------------------------------------------------------------------------

009 using System;

010 using System.Reflection;

011 [assembly: System.Reflection.AssemblyCompanyAttribute("RehearsalStudio.Application")]

012 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

013 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

014 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

015 [assembly: System.Reflection.AssemblyProductAttribute("RehearsalStudio.Application")]

016 [assembly: System.Reflection.AssemblyTitleAttribute("RehearsalStudio.Application")]

017 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

018 // Generated by the MSBuild WriteCodeFragment class.

Файл RehearsalStudio.Application.GlobalUsings.g.cs:

001 // <auto-generated/>

002 global using global::System;

003 global using global::System.Collections.Generic;

004 global using global::System.IO;

005 global using global::System.Linq;

006 global using global::System.Net.Http;

007 global using global::System.Threading;

008 global using global::System.Threading.Tasks;

Файл BookingRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class BookingRepository : IBookingRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public BookingRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<Booking>> GetAllAsync()

014 {

015 return await \_context.Bookings.ToListAsync();

016 }

017 public async Task<Booking?> GetByIdAsync(int id)

018 {

019 return await \_context.Bookings.FindAsync(id);

020 }

021 public async Task<IEnumerable<Booking>> GetFilteredAsync(string? status, int? idRoom, int? idUser)

022 {

023 var query = \_context.Bookings.AsQueryable();

024 if (!string.IsNullOrEmpty(status))

025 query = query.Where(b => b.Status.Contains(status));

026 if (idRoom.HasValue)

027 query = query.Where(b => b.IdRoom == idRoom.Value);

028 if (idUser.HasValue)

029 query = query.Where(b => b.IdUser == idUser.Value);

030 return await query.ToListAsync();

031 }

032 public async Task<Booking> AddAsync(Booking booking)

033 {

034 \_context.Bookings.Add(booking);

035 await \_context.SaveChangesAsync();

036 return booking;

037 }

038 public async Task UpdateAsync(Booking booking)

039 {

040 \_context.Bookings.Update(booking);

041 await \_context.SaveChangesAsync();

042 }

043 public async Task DeleteAsync(int id)

044 {

045 var booking = await \_context.Bookings.FindAsync(id);

046 if (booking != null)

047 {

048 \_context.Bookings.Remove(booking);

049 await \_context.SaveChangesAsync();

050 }

051 }

052 }

Файл EquipmentBookingRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class EquipmentBookingRepository : IEquipmentBookingRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public EquipmentBookingRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<EquipmentBooking>> GetAllAsync()

014 {

015 return await \_context.EquipmentBookings.ToListAsync();

016 }

017 public async Task<EquipmentBooking?> GetByIdAsync(int idEquipment, int idBooking)

018 {

019 return await \_context.EquipmentBookings.FindAsync(idEquipment, idBooking);

020 }

021 public async Task<IEnumerable<EquipmentBooking>> GetFilteredAsync(int? idEquipment, int? idBooking)

022 {

023 var query = \_context.EquipmentBookings.AsQueryable();

024 if (idEquipment.HasValue)

025 query = query.Where(eb => eb.IdEquipment == idEquipment.Value);

026 if (idBooking.HasValue)

027 query = query.Where(eb => eb.IdBooking == idBooking.Value);

028 return await query.ToListAsync();

029 }

030 public async Task<EquipmentBooking> AddAsync(EquipmentBooking equipmentBooking)

031 {

032 \_context.EquipmentBookings.Add(equipmentBooking);

033 await \_context.SaveChangesAsync();

034 return equipmentBooking;

035 }

036 public async Task DeleteAsync(int idEquipment, int idBooking)

037 {

038 var equipmentBooking = await \_context.EquipmentBookings.FindAsync(idEquipment, idBooking);

039 if (equipmentBooking != null)

040 {

041 \_context.EquipmentBookings.Remove(equipmentBooking);

042 await \_context.SaveChangesAsync();

043 }

044 }

045 }

Файл EquipmentRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class EquipmentRepository : IEquipmentRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public EquipmentRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<Equipment>> GetAllAsync()

014 {

015 return await \_context.Equipment.ToListAsync();

016 }

017 public async Task<Equipment?> GetByIdAsync(int id)

018 {

019 return await \_context.Equipment.FindAsync(id);

020 }

021 public async Task<IEnumerable<Equipment>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint)

022 {

023 var query = \_context.Equipment.AsQueryable();

024 if (!string.IsNullOrEmpty(name))

025 query = query.Where(e => e.Name.Contains(name));

026 if (!string.IsNullOrEmpty(type))

027 query = query.Where(e => e.Type.Contains(type));

028 if (idRehearsalPoint.HasValue)

029 query = query.Where(e => e.IdRehearsalPoint == idRehearsalPoint.Value);

030 return await query.ToListAsync();

031 }

032 public async Task<Equipment> AddAsync(Equipment equipment)

033 {

034 \_context.Equipment.Add(equipment);

035 await \_context.SaveChangesAsync();

036 return equipment;

037 }

038 public async Task UpdateAsync(Equipment equipment)

039 {

040 \_context.Equipment.Update(equipment);

041 await \_context.SaveChangesAsync();

042 }

043 public async Task DeleteAsync(int id)

044 {

045 var equipment = await \_context.Equipment.FindAsync(id);

046 if (equipment != null)

047 {

048 \_context.Equipment.Remove(equipment);

049 await \_context.SaveChangesAsync();

050 }

051 }

052 }

Файл RehearsalPointRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class RehearsalPointRepository : IRehearsalPointRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public RehearsalPointRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<RehearsalPoint>> GetAllAsync()

014 {

015 return await \_context.RehearsalPoints.ToListAsync();

016 }

017 public async Task<RehearsalPoint?> GetByIdAsync(int id)

018 {

019 return await \_context.RehearsalPoints.FindAsync(id);

020 }

021 public async Task<IEnumerable<RehearsalPoint>> GetFilteredAsync(string? name, float? minRating)

022 {

023 var query = \_context.RehearsalPoints.AsQueryable();

024 if (!string.IsNullOrEmpty(name))

025 query = query.Where(rp => rp.Name.Contains(name));

026 if (minRating.HasValue)

027 query = query.Where(rp => rp.Rating >= minRating.Value);

028 return await query.ToListAsync();

029 }

030 public async Task<RehearsalPoint> AddAsync(RehearsalPoint rehearsalPoint)

031 {

032 \_context.RehearsalPoints.Add(rehearsalPoint);

033 await \_context.SaveChangesAsync();

034 return rehearsalPoint;

035 }

036 public async Task UpdateAsync(RehearsalPoint rehearsalPoint)

037 {

038 \_context.RehearsalPoints.Update(rehearsalPoint);

039 await \_context.SaveChangesAsync();

040 }

041 public async Task DeleteAsync(int id)

042 {

043 var rehearsalPoint = await \_context.RehearsalPoints.FindAsync(id);

044 if (rehearsalPoint != null)

045 {

046 \_context.RehearsalPoints.Remove(rehearsalPoint);

047 await \_context.SaveChangesAsync();

048 }

049 }

050 }

Файл RoomRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class RoomRepository : IRoomRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public RoomRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<Room>> GetAllAsync()

014 {

015 return await \_context.Rooms.ToListAsync();

016 }

017 public async Task<Room?> GetByIdAsync(int id)

018 {

019 return await \_context.Rooms.FindAsync(id);

020 }

021 public async Task<IEnumerable<Room>> GetFilteredAsync(string? name, int? minPrice, int? idRehearsalPoint)

022 {

023 var query = \_context.Rooms.AsQueryable();

024 if (!string.IsNullOrEmpty(name))

025 query = query.Where(r => r.Name.Contains(name));

026 if (minPrice.HasValue)

027 query = query.Where(r => r.Price >= minPrice.Value);

028 if (idRehearsalPoint.HasValue)

029 query = query.Where(r => r.IdRehearsalPoint == idRehearsalPoint.Value);

030 return await query.ToListAsync();

031 }

032 public async Task<Room> AddAsync(Room room)

033 {

034 \_context.Rooms.Add(room);

035 await \_context.SaveChangesAsync();

036 return room;

037 }

038 public async Task UpdateAsync(Room room)

039 {

040 \_context.Rooms.Update(room);

041 await \_context.SaveChangesAsync();

042 }

043 public async Task DeleteAsync(int id)

044 {

045 var room = await \_context.Rooms.FindAsync(id);

046 if (room != null)

047 {

048 \_context.Rooms.Remove(room);

049 await \_context.SaveChangesAsync();

050 }

051 }

052 }

Файл ServiceBookingRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class ServiceBookingRepository : IServiceBookingRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public ServiceBookingRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<ServiceBooking>> GetAllAsync()

014 {

015 return await \_context.ServiceBookings.ToListAsync();

016 }

017 public async Task<ServiceBooking?> GetByIdAsync(int idService, int idBooking)

018 {

019 return await \_context.ServiceBookings.FindAsync(idService, idBooking);

020 }

021 public async Task<IEnumerable<ServiceBooking>> GetFilteredAsync(int? idService, int? idBooking)

022 {

023 var query = \_context.ServiceBookings.AsQueryable();

024 if (idService.HasValue)

025 query = query.Where(sb => sb.IdService == idService.Value);

026 if (idBooking.HasValue)

027 query = query.Where(sb => sb.IdBooking == idBooking.Value);

028 return await query.ToListAsync();

029 }

030 public async Task<ServiceBooking> AddAsync(ServiceBooking serviceBooking)

031 {

032 \_context.ServiceBookings.Add(serviceBooking);

033 await \_context.SaveChangesAsync();

034 return serviceBooking;

035 }

036 public async Task DeleteAsync(int idService, int idBooking)

037 {

038 var serviceBooking = await \_context.ServiceBookings.FindAsync(idService, idBooking);

039 if (serviceBooking != null)

040 {

041 \_context.ServiceBookings.Remove(serviceBooking);

042 await \_context.SaveChangesAsync();

043 }

044 }

045 }

Файл ServiceRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class ServiceRepository : IServiceRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public ServiceRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<Service>> GetAllAsync()

014 {

015 return await \_context.Services.ToListAsync();

016 }

017 public async Task<Service?> GetByIdAsync(int id)

018 {

019 return await \_context.Services.FindAsync(id);

020 }

021 public async Task<IEnumerable<Service>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint)

022 {

023 var query = \_context.Services.AsQueryable();

024 if (!string.IsNullOrEmpty(name))

025 query = query.Where(s => s.Name.Contains(name));

026 if (!string.IsNullOrEmpty(type))

027 query = query.Where(s => s.Type.Contains(type));

028 if (idRehearsalPoint.HasValue)

029 query = query.Where(s => s.IdRehearsalPoint == idRehearsalPoint.Value);

030 return await query.ToListAsync();

031 }

032 public async Task<Service> AddAsync(Service service)

033 {

034 \_context.Services.Add(service);

035 await \_context.SaveChangesAsync();

036 return service;

037 }

038 public async Task UpdateAsync(Service service)

039 {

040 \_context.Services.Update(service);

041 await \_context.SaveChangesAsync();

042 }

043 public async Task DeleteAsync(int id)

044 {

045 var service = await \_context.Services.FindAsync(id);

046 if (service != null)

047 {

048 \_context.Services.Remove(service);

049 await \_context.SaveChangesAsync();

050 }

051 }

052 }

Файл StaffRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class StaffRepository : IStaffRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public StaffRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<Staff>> GetAllAsync()

014 {

015 return await \_context.Staff.ToListAsync();

016 }

017 public async Task<Staff?> GetByIdAsync(int id)

018 {

019 return await \_context.Staff.FindAsync(id);

020 }

021 public async Task<IEnumerable<Staff>> GetFilteredAsync(string? fullName, int? minAge, int? idRehearsalPoint)

022 {

023 var query = \_context.Staff.AsQueryable();

024 if (!string.IsNullOrEmpty(fullName))

025 query = query.Where(s => s.FullName.Contains(fullName));

026 if (minAge.HasValue)

027 query = query.Where(s => s.Age >= minAge.Value);

028 if (idRehearsalPoint.HasValue)

029 query = query.Where(s => s.IdRehearsalPoint == idRehearsalPoint.Value);

030 return await query.ToListAsync();

031 }

032 public async Task<Staff> AddAsync(Staff staff)

033 {

034 \_context.Staff.Add(staff);

035 await \_context.SaveChangesAsync();

036 return staff;

037 }

038 public async Task UpdateAsync(Staff staff)

039 {

040 \_context.Staff.Update(staff);

041 await \_context.SaveChangesAsync();

042 }

043 public async Task DeleteAsync(int id)

044 {

045 var staff = await \_context.Staff.FindAsync(id);

046 if (staff != null)

047 {

048 \_context.Staff.Remove(staff);

049 await \_context.SaveChangesAsync();

050 }

051 }

052 }

Файл UserRepository.cs:

001 using Microsoft.EntityFrameworkCore;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using RehearsalStudio.Infrastructure.Data;

005 namespace RehearsalStudio.Infrastructure.Repositories;

006 public class UserRepository : IUserRepository

007 {

008 private readonly RehearsalStudioDbContext \_context;

009 public UserRepository(RehearsalStudioDbContext context)

010 {

011 \_context = context;

012 }

013 public async Task<IEnumerable<User>> GetAllAsync()

014 {

015 return await \_context.Users.ToListAsync();

016 }

017 public async Task<User?> GetByIdAsync(int id)

018 {

019 return await \_context.Users.FindAsync(id);

020 }

021 public async Task<IEnumerable<User>> GetFilteredAsync(string? fullName, string? email)

022 {

023 var query = \_context.Users.AsQueryable();

024 if (!string.IsNullOrEmpty(fullName))

025 query = query.Where(u => u.FullName.Contains(fullName));

026 if (!string.IsNullOrEmpty(email))

027 query = query.Where(u => u.Email.Contains(email));

028 return await query.ToListAsync();

029 }

030 public async Task<User> AddAsync(User user)

031 {

032 \_context.Users.Add(user);

033 await \_context.SaveChangesAsync();

034 return user;

035 }

036 public async Task UpdateAsync(User user)

037 {

038 \_context.Users.Update(user);

039 await \_context.SaveChangesAsync();

040 }

041 public async Task DeleteAsync(int id)

042 {

043 var user = await \_context.Users.FindAsync(id);

044 if (user != null)

045 {

046 \_context.Users.Remove(user);

047 await \_context.SaveChangesAsync();

048 }

049 }

050 }

Файл BackupService.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.IO;

004 using System.Text;

005 using System.Threading.Tasks;

006 using Dapper;

007 using Microsoft.EntityFrameworkCore;

008 using Npgsql;

009 using RehearsalStudio.Application.Interfaces;

010 using RehearsalStudio.Infrastructure.Data;

011 using System.Text.Json;

012 using System.Linq;

013 namespace RehearsalStudio.Application.Services;

014 public class BackupService : IBackupService

015 {

016 private readonly RehearsalStudioDbContext \_context;

017 public BackupService(RehearsalStudioDbContext context)

018 {

019 \_context = context;

020 }

021 public async Task<string> CreateDatabaseBackupAsync()

022 {

023 var connectionString = \_context.Database.GetConnectionString();

024 var backupFilePath = $"backup\_{DateTime.Now:yyyyMMddHHmmss}.sql";

025 using var connection = new NpgsqlConnection(connectionString);

026 await connection.OpenAsync();

027 var backupScript = new StringBuilder();

028 // List of tables to back up

029 var tables = new[] { "rehearsal\_points", "rooms", "service", "equipment", "staff", "users", "booking", "service\_booking", "equipment\_booking" };

030 foreach (var table in tables)

031 {

032 // Generate table structure using information\_schema

033 var columns = await connection.QueryAsync<ColumnInfo>(

034 @"SELECT column\_name, data\_type, is\_nullable, character\_maximum\_length

035 FROM information\_schema.columns

036 WHERE table\_schema = 'main' AND table\_name = @TableName",

037 new { TableName = table });

038 // Start CREATE TABLE statement

039 backupScript.AppendLine($"DROP TABLE IF EXISTS main.{table} CASCADE;");

040 backupScript.AppendLine($"CREATE TABLE main.{table} (");

041 var columnDefinitions = columns.Select(c =>

042 {

043 var dataType = c.data\_type switch

044 {

045 "integer" => "INTEGER",

046 "real" => "REAL",

047 "boolean" => "BOOLEAN",

048 "text" => "TEXT",

049 "timestamp with time zone" => "TIMESTAMP WITH TIME ZONE",

050 \_ => c.data\_type.ToUpper()

051 };

052 var nullable = c.is\_nullable == "YES" ? "" : " NOT NULL";

053 return $" {c.column\_name} {dataType}{nullable}";

054 });

055 backupScript.AppendLine(string.Join(",\n", columnDefinitions));

056 // Add primary key constraints

057 var primaryKeys = await connection.QueryAsync<string>(

058 @"SELECT a.attname

059 FROM pg\_index i

060 JOIN pg\_attribute a ON a.attrelid = i.indrelid AND a.attnum = ANY(i.indkey)

061 JOIN pg\_class c ON c.oid = i.indrelid

062 JOIN pg\_namespace n ON n.oid = c.relnamespace

063 WHERE n.nspname = 'main' AND c.relname = @TableName AND i.indisprimary",

064 new { TableName = table });

065 if (primaryKeys.Any())

066 {

067 backupScript.AppendLine($", PRIMARY KEY ({string.Join(", ", primaryKeys)})");

068 }

069 backupScript.AppendLine(");");

070 backupScript.AppendLine();

071 // Export table data

072 using var reader = await connection.ExecuteReaderAsync($"SELECT \* FROM main.{table}");

073 var columnNames = Enumerable.Range(0, reader.FieldCount).Select(reader.GetName).ToList();

074 while (await reader.ReadAsync())

075 {

076 var values = new List<string>();

077 for (int i = 0; i < reader.FieldCount; i++)

078 {

079 var value = reader.GetValue(i);

080 if (value == DBNull.Value)

081 values.Add("NULL");

082 else if (reader.GetFieldType(i) == typeof(DateTime))

083 values.Add($"'{(DateTime)value:yyyy-MM-dd HH:mm:ss.fffz}'");

084 else if (reader.GetFieldType(i) == typeof(string))

085 values.Add($"'{value.ToString().Replace("'", "''")}'");

086 else

087 values.Add(value.ToString());

088 }

089 backupScript.AppendLine($"INSERT INTO main.{table} ({string.Join(", ", columnNames)}) VALUES ({string.Join(", ", values)});");

090 }

091 backupScript.AppendLine();

092 }

093 await File.WriteAllTextAsync(backupFilePath, backupScript.ToString());

094 return backupFilePath;

095 }

096 public async Task<string> SaveQueryResultsToFileAsync(string sqlQuery, string fileFormat = "json")

097 {

098 var connectionString = \_context.Database.GetConnectionString();

099 var resultFilePath = $"query\_results\_{DateTime.Now:yyyyMMddHHmmss}.{fileFormat}";

100 using var connection = new NpgsqlConnection(connectionString);

101 await connection.OpenAsync();

102 var results = await connection.QueryAsync<dynamic>(sqlQuery);

103 if (fileFormat.ToLower() == "json")

104 {

105 var json = JsonSerializer.Serialize(results);

106 await File.WriteAllTextAsync(resultFilePath, json);

107 }

108 else if (fileFormat.ToLower() == "csv")

109 {

110 var csv = new StringBuilder();

111 if (results.Any())

112 {

113 var columns = ((IDictionary<string, object>)results.First()).Keys;

114 csv.AppendLine(string.Join(",", columns));

115 foreach (var row in results)

116 {

117 var dict = (IDictionary<string, object>)row;

118 var values = columns.Select(c => dict[c]?.ToString() ?? string.Empty).Select(v => v.Contains(",") ? $"\"{v}\"" : v);

119 csv.AppendLine(string.Join(",", values));

120 }

121 }

122 await File.WriteAllTextAsync(resultFilePath, csv.ToString());

123 }

124 else

125 {

126 throw new ArgumentException("Unsupported file format. Use 'json' or 'csv'.");

127 }

128 return resultFilePath;

129 }

130 private class ColumnInfo

131 {

132 public string column\_name { get; set; }

133 public string data\_type { get; set; }

134 public string is\_nullable { get; set; }

135 public int? character\_maximum\_length { get; set; }

136 }

137 }

Файл BookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class BookingService : IBookingService

009 {

010 private readonly IBookingRepository \_repository;

011 public BookingService(IBookingRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<BookingDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new BookingDto

019 {

020 Id = e.Id,

021 Time = e.Time,

022 Duration = e.Duration,

023 Cost = e.Cost,

024 CreationDate = e.CreationDate,

025 Status = e.Status,

026 NumberOfPeople = e.NumberOfPeople,

027 IdRoom = e.IdRoom,

028 IdUser = e.IdUser

029 });

030 }

031 public async Task<BookingDto?> GetByIdAsync(int id)

032 {

033 var entity = await \_repository.GetByIdAsync(id);

034 if (entity == null) return null;

035 return new BookingDto

036 {

037 Id = entity.Id,

038 Time = entity.Time,

039 Duration = entity.Duration,

040 Cost = entity.Cost,

041 CreationDate = entity.CreationDate,

042 Status = entity.Status,

043 NumberOfPeople = entity.NumberOfPeople,

044 IdRoom = entity.IdRoom,

045 IdUser = entity.IdUser

046 };

047 }

048 public async Task<IEnumerable<BookingDto>> GetFilteredAsync(string? status, int? idRoom, int? idUser)

049 {

050 var entities = await \_repository.GetFilteredAsync(status, idRoom, idUser);

051 return entities.Select(e => new BookingDto

052 {

053 Id = e.Id,

054 Time = e.Time,

055 Duration = e.Duration,

056 Cost = e.Cost,

057 CreationDate = e.CreationDate,

058 Status = e.Status,

059 NumberOfPeople = e.NumberOfPeople,

060 IdRoom = e.IdRoom,

061 IdUser = e.IdUser

062 });

063 }

064 public async Task<BookingDto> CreateAsync(BookingDto dto)

065 {

066 if (string.IsNullOrEmpty(dto.Status) || dto.Cost <= 0 || dto.NumberOfPeople <= 0)

067 throw new ArgumentException("Status, Cost, and NumberOfPeople are required and must be valid.");

068 var entity = new Booking

069 {

070 Time = dto.Time,

071 Duration = dto.Duration,

072 Cost = dto.Cost,

073 CreationDate = dto.CreationDate,

074 Status = dto.Status,

075 NumberOfPeople = dto.NumberOfPeople,

076 IdRoom = dto.IdRoom,

077 IdUser = dto.IdUser

078 };

079 var created = await \_repository.AddAsync(entity);

080 return new BookingDto

081 {

082 Id = created.Id,

083 Time = created.Time,

084 Duration = created.Duration,

085 Cost = created.Cost,

086 CreationDate = created.CreationDate,

087 Status = created.Status,

088 NumberOfPeople = created.NumberOfPeople,

089 IdRoom = created.IdRoom,

090 IdUser = created.IdUser

091 };

092 }

093 public async Task UpdateAsync(int id, BookingDto dto)

094 {

095 if (string.IsNullOrEmpty(dto.Status) || dto.Cost <= 0 || dto.NumberOfPeople <= 0)

096 throw new ArgumentException("Status, Cost, and NumberOfPeople are required and must be valid.");

097 var entity = await \_repository.GetByIdAsync(id);

098 if (entity == null)

099 throw new KeyNotFoundException($"Booking with ID {id} not found.");

100 entity.Time = dto.Time;

101 entity.Duration = dto.Duration;

102 entity.Cost = dto.Cost;

103 entity.CreationDate = dto.CreationDate;

104 entity.Status = dto.Status;

105 entity.NumberOfPeople = dto.NumberOfPeople;

106 entity.IdRoom = dto.IdRoom;

107 entity.IdUser = dto.IdUser;

108 await \_repository.UpdateAsync(entity);

109 }

110 public async Task DeleteAsync(int id)

111 {

112 var entity = await \_repository.GetByIdAsync(id);

113 if (entity == null)

114 throw new KeyNotFoundException($"Booking with ID {id} not found.");

115 await \_repository.DeleteAsync(id);

116 }

117 }

Файл EquipmentBookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class EquipmentBookingService : IEquipmentBookingService

009 {

010 private readonly IEquipmentBookingRepository \_repository;

011 public EquipmentBookingService(IEquipmentBookingRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<EquipmentBookingDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new EquipmentBookingDto

019 {

020 IdEquipment = e.IdEquipment,

021 IdBooking = e.IdBooking

022 });

023 }

024 public async Task<EquipmentBookingDto?> GetByIdAsync(int idEquipment, int idBooking)

025 {

026 var entity = await \_repository.GetByIdAsync(idEquipment, idBooking);

027 if (entity == null) return null;

028 return new EquipmentBookingDto

029 {

030 IdEquipment = entity.IdEquipment,

031 IdBooking = entity.IdBooking

032 };

033 }

034 public async Task<IEnumerable<EquipmentBookingDto>> GetFilteredAsync(int? idEquipment, int? idBooking)

035 {

036 var entities = await \_repository.GetFilteredAsync(idEquipment, idBooking);

037 return entities.Select(e => new EquipmentBookingDto

038 {

039 IdEquipment = e.IdEquipment,

040 IdBooking = e.IdBooking

041 });

042 }

043 public async Task<EquipmentBookingDto> CreateAsync(EquipmentBookingDto dto)

044 {

045 if (dto.IdEquipment <= 0 || dto.IdBooking <= 0)

046 throw new ArgumentException("IdEquipment and IdBooking must be valid.");

047 var entity = new EquipmentBooking

048 {

049 IdEquipment = dto.IdEquipment,

050 IdBooking = dto.IdBooking

051 };

052 var created = await \_repository.AddAsync(entity);

053 return new EquipmentBookingDto

054 {

055 IdEquipment = created.IdEquipment,

056 IdBooking = created.IdBooking

057 };

058 }

059 public async Task DeleteAsync(int idEquipment, int idBooking)

060 {

061 var entity = await \_repository.GetByIdAsync(idEquipment, idBooking);

062 if (entity == null)

063 throw new KeyNotFoundException($"EquipmentBooking with IdEquipment {idEquipment} and IdBooking {idBooking} not found.");

064 await \_repository.DeleteAsync(idEquipment, idBooking);

065 }

066 }

Файл EquipmentService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class EquipmentService : IEquipmentService

009 {

010 private readonly IEquipmentRepository \_repository;

011 public EquipmentService(IEquipmentRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<EquipmentDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new EquipmentDto

019 {

020 Id = e.Id,

021 Name = e.Name,

022 Type = e.Type,

023 Brand = e.Brand,

024 Model = e.Model,

025 Condition = e.Condition,

026 IdRehearsalPoint = e.IdRehearsalPoint

027 });

028 }

029 public async Task<EquipmentDto?> GetByIdAsync(int id)

030 {

031 var entity = await \_repository.GetByIdAsync(id);

032 if (entity == null) return null;

033 return new EquipmentDto

034 {

035 Id = entity.Id,

036 Name = entity.Name,

037 Type = entity.Type,

038 Brand = entity.Brand,

039 Model = entity.Model,

040 Condition = entity.Condition,

041 IdRehearsalPoint = entity.IdRehearsalPoint

042 };

043 }

044 public async Task<IEnumerable<EquipmentDto>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint)

045 {

046 var entities = await \_repository.GetFilteredAsync(name, type, idRehearsalPoint);

047 return entities.Select(e => new EquipmentDto

048 {

049 Id = e.Id,

050 Name = e.Name,

051 Type = e.Type,

052 Brand = e.Brand,

053 Model = e.Model,

054 Condition = e.Condition,

055 IdRehearsalPoint = e.IdRehearsalPoint

056 });

057 }

058 public async Task<EquipmentDto> CreateAsync(EquipmentDto dto)

059 {

060 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Type) || string.IsNullOrEmpty(dto.Brand) ||

061 string.IsNullOrEmpty(dto.Model) || string.IsNullOrEmpty(dto.Condition))

062 throw new ArgumentException("Name, Type, Brand, Model, and Condition are required.");

063 var entity = new Equipment

064 {

065 Name = dto.Name,

066 Type = dto.Type,

067 Brand = dto.Brand,

068 Model = dto.Model,

069 Condition = dto.Condition,

070 IdRehearsalPoint = dto.IdRehearsalPoint

071 };

072 var created = await \_repository.AddAsync(entity);

073 return new EquipmentDto

074 {

075 Id = created.Id,

076 Name = created.Name,

077 Type = created.Type,

078 Brand = created.Brand,

079 Model = created.Model,

080 Condition = created.Condition,

081 IdRehearsalPoint = created.IdRehearsalPoint

082 };

083 }

084 public async Task UpdateAsync(int id, EquipmentDto dto)

085 {

086 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Type) || string.IsNullOrEmpty(dto.Brand) ||

087 string.IsNullOrEmpty(dto.Model) || string.IsNullOrEmpty(dto.Condition))

088 throw new ArgumentException("Name, Type, Brand, Model, and Condition are required.");

089 var entity = await \_repository.GetByIdAsync(id);

090 if (entity == null)

091 throw new KeyNotFoundException($"Equipment with ID {id} not found.");

092 entity.Name = dto.Name;

093 entity.Type = dto.Type;

094 entity.Brand = dto.Brand;

095 entity.Model = dto.Model;

096 entity.Condition = dto.Condition;

097 entity.IdRehearsalPoint = dto.IdRehearsalPoint;

098 await \_repository.UpdateAsync(entity);

099 }

100 public async Task DeleteAsync(int id)

101 {

102 var entity = await \_repository.GetByIdAsync(id);

103 if (entity == null)

104 throw new KeyNotFoundException($"Equipment with ID {id} not found.");

105 await \_repository.DeleteAsync(id);

106 }

107 }

Файл RehearsalPointService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class RehearsalPointService : IRehearsalPointService

009 {

010 private readonly IRehearsalPointRepository \_repository;

011 public RehearsalPointService(IRehearsalPointRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<RehearsalPointDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new RehearsalPointDto

019 {

020 Id = e.Id,

021 Rating = e.Rating,

022 ContactNumber = e.ContactNumber,

023 Schedule = e.Schedule,

024 Name = e.Name,

025 Address = e.Address

026 });

027 }

028 public async Task<RehearsalPointDto?> GetByIdAsync(int id)

029 {

030 var entity = await \_repository.GetByIdAsync(id);

031 if (entity == null) return null;

032 return new RehearsalPointDto

033 {

034 Id = entity.Id,

035 Rating = entity.Rating,

036 ContactNumber = entity.ContactNumber,

037 Schedule = entity.Schedule,

038 Name = entity.Name,

039 Address = entity.Address

040 };

041 }

042 public async Task<IEnumerable<RehearsalPointDto>> GetFilteredAsync(string? name, float? minRating)

043 {

044 var entities = await \_repository.GetFilteredAsync(name, minRating);

045 return entities.Select(e => new RehearsalPointDto

046 {

047 Id = e.Id,

048 Rating = e.Rating,

049 ContactNumber = e.ContactNumber,

050 Schedule = e.Schedule,

051 Name = e.Name,

052 Address = e.Address

053 });

054 }

055 public async Task<RehearsalPointDto> CreateAsync(RehearsalPointDto dto)

056 {

057 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Address) || string.IsNullOrEmpty(dto.ContactNumber))

058 throw new ArgumentException("Name, Address, and ContactNumber are required.");

059 var entity = new RehearsalPoint

060 {

061 Rating = dto.Rating,

062 ContactNumber = dto.ContactNumber,

063 Schedule = dto.Schedule,

064 Name = dto.Name,

065 Address = dto.Address

066 };

067 var created = await \_repository.AddAsync(entity);

068 return new RehearsalPointDto

069 {

070 Id = created.Id,

071 Rating = created.Rating,

072 ContactNumber = created.ContactNumber,

073 Schedule = created.Schedule,

074 Name = created.Name,

075 Address = created.Address

076 };

077 }

078 public async Task UpdateAsync(int id, RehearsalPointDto dto)

079 {

080 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Address) || string.IsNullOrEmpty(dto.ContactNumber))

081 throw new ArgumentException("Name, Address, and ContactNumber are required.");

082 var entity = await \_repository.GetByIdAsync(id);

083 if (entity == null)

084 throw new KeyNotFoundException($"RehearsalPoint with ID {id} not found.");

085 entity.Rating = dto.Rating;

086 entity.ContactNumber = dto.ContactNumber;

087 entity.Schedule = dto.Schedule;

088 entity.Name = dto.Name;

089 entity.Address = dto.Address;

090 await \_repository.UpdateAsync(entity);

091 }

092 public async Task DeleteAsync(int id)

093 {

094 var entity = await \_repository.GetByIdAsync(id);

095 if (entity == null)

096 throw new KeyNotFoundException($"RehearsalPoint with ID {id} not found.");

097 await \_repository.DeleteAsync(id);

098 }

099 }

Файл RoomService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class RoomService : IRoomService

009 {

010 private readonly IRoomRepository \_repository;

011 public RoomService(IRoomRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<RoomDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new RoomDto

019 {

020 Id = e.Id,

021 Name = e.Name,

022 AirConditioner = e.AirConditioner,

023 Price = e.Price,

024 RecordingSupport = e.RecordingSupport,

025 Area = e.Area,

026 IdRehearsalPoint = e.IdRehearsalPoint

027 });

028 }

029 public async Task<RoomDto?> GetByIdAsync(int id)

030 {

031 var entity = await \_repository.GetByIdAsync(id);

032 if (entity == null) return null;

033 return new RoomDto

034 {

035 Id = entity.Id,

036 Name = entity.Name,

037 AirConditioner = entity.AirConditioner,

038 Price = entity.Price,

039 RecordingSupport = entity.RecordingSupport,

040 Area = entity.Area,

041 IdRehearsalPoint = entity.IdRehearsalPoint

042 };

043 }

044 public async Task<IEnumerable<RoomDto>> GetFilteredAsync(string? name, int? minPrice, int? idRehearsalPoint)

045 {

046 var entities = await \_repository.GetFilteredAsync(name, minPrice, idRehearsalPoint);

047 return entities.Select(e => new RoomDto

048 {

049 Id = e.Id,

050 Name = e.Name,

051 AirConditioner = e.AirConditioner,

052 Price = e.Price,

053 RecordingSupport = e.RecordingSupport,

054 Area = e.Area,

055 IdRehearsalPoint = e.IdRehearsalPoint

056 });

057 }

058 public async Task<RoomDto> CreateAsync(RoomDto dto)

059 {

060 if (string.IsNullOrEmpty(dto.Name) || dto.Price <= 0 || dto.Area <= 0)

061 throw new ArgumentException("Name, Price, and Area are required and must be valid.");

062 var entity = new Room

063 {

064 Name = dto.Name,

065 AirConditioner = dto.AirConditioner,

066 Price = dto.Price,

067 RecordingSupport = dto.RecordingSupport,

068 Area = dto.Area,

069 IdRehearsalPoint = dto.IdRehearsalPoint

070 };

071 var created = await \_repository.AddAsync(entity);

072 return new RoomDto

073 {

074 Id = created.Id,

075 Name = created.Name,

076 AirConditioner = created.AirConditioner,

077 Price = created.Price,

078 RecordingSupport = created.RecordingSupport,

079 Area = created.Area,

080 IdRehearsalPoint = created.IdRehearsalPoint

081 };

082 }

083 public async Task UpdateAsync(int id, RoomDto dto)

084 {

085 if (string.IsNullOrEmpty(dto.Name) || dto.Price <= 0 || dto.Area <= 0)

086 throw new ArgumentException("Name, Price, and Area are required and must be valid.");

087 var entity = await \_repository.GetByIdAsync(id);

088 if (entity == null)

089 throw new KeyNotFoundException($"Room with ID {id} not found.");

090 entity.Name = dto.Name;

091 entity.AirConditioner = dto.AirConditioner;

092 entity.Price = dto.Price;

093 entity.RecordingSupport = dto.RecordingSupport;

094 entity.Area = dto.Area;

095 entity.IdRehearsalPoint = dto.IdRehearsalPoint;

096 await \_repository.UpdateAsync(entity);

097 }

098 public async Task DeleteAsync(int id)

099 {

100 var entity = await \_repository.GetByIdAsync(id);

101 if (entity == null)

102 throw new KeyNotFoundException($"Room with ID {id} not found.");

103 await \_repository.DeleteAsync(id);

104 }

105 }

Файл ServiceBookingService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class ServiceBookingService : IServiceBookingService

009 {

010 private readonly IServiceBookingRepository \_repository;

011 public ServiceBookingService(IServiceBookingRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<ServiceBookingDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new ServiceBookingDto

019 {

020 IdService = e.IdService,

021 IdBooking = e.IdBooking

022 });

023 }

024 public async Task<ServiceBookingDto?> GetByIdAsync(int idService, int idBooking)

025 {

026 var entity = await \_repository.GetByIdAsync(idService, idBooking);

027 if (entity == null) return null;

028 return new ServiceBookingDto

029 {

030 IdService = entity.IdService,

031 IdBooking = entity.IdBooking

032 };

033 }

034 public async Task<IEnumerable<ServiceBookingDto>> GetFilteredAsync(int? idService, int? idBooking)

035 {

036 var entities = await \_repository.GetFilteredAsync(idService, idBooking);

037 return entities.Select(e => new ServiceBookingDto

038 {

039 IdService = e.IdService,

040 IdBooking = e.IdBooking

041 });

042 }

043 public async Task<ServiceBookingDto> CreateAsync(ServiceBookingDto dto)

044 {

045 if (dto.IdService <= 0 || dto.IdBooking <= 0)

046 throw new ArgumentException("IdService and IdBooking must be valid.");

047 var entity = new ServiceBooking

048 {

049 IdService = dto.IdService,

050 IdBooking = dto.IdBooking

051 };

052 var created = await \_repository.AddAsync(entity);

053 return new ServiceBookingDto

054 {

055 IdService = created.IdService,

056 IdBooking = created.IdBooking

057 };

058 }

059 public async Task DeleteAsync(int idService, int idBooking)

060 {

061 var entity = await \_repository.GetByIdAsync(idService, idBooking);

062 if (entity == null)

063 throw new KeyNotFoundException($"ServiceBooking with IdService {idService} and IdBooking {idBooking} not found.");

064 await \_repository.DeleteAsync(idService, idBooking);

065 }

066 }

Файл ServiceService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class ServiceService : IServiceService

009 {

010 private readonly IServiceRepository \_repository;

011 public ServiceService(IServiceRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<ServiceDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new ServiceDto

019 {

020 Id = e.Id,

021 Name = e.Name,

022 Price = e.Price,

023 Type = e.Type,

024 Requirements = e.Requirements,

025 IdRehearsalPoint = e.IdRehearsalPoint

026 });

027 }

028 public async Task<ServiceDto?> GetByIdAsync(int id)

029 {

030 var entity = await \_repository.GetByIdAsync(id);

031 if (entity == null) return null;

032 return new ServiceDto

033 {

034 Id = entity.Id,

035 Name = entity.Name,

036 Price = entity.Price,

037 Type = entity.Type,

038 Requirements = entity.Requirements,

039 IdRehearsalPoint = entity.IdRehearsalPoint

040 };

041 }

042 public async Task<IEnumerable<ServiceDto>> GetFilteredAsync(string? name, string? type, int? idRehearsalPoint)

043 {

044 var entities = await \_repository.GetFilteredAsync(name, type, idRehearsalPoint);

045 return entities.Select(e => new ServiceDto

046 {

047 Id = e.Id,

048 Name = e.Name,

049 Price = e.Price,

050 Type = e.Type,

051 Requirements = e.Requirements,

052 IdRehearsalPoint = e.IdRehearsalPoint

053 });

054 }

055 public async Task<ServiceDto> CreateAsync(ServiceDto dto)

056 {

057 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Type) || dto.Price <= 0)

058 throw new ArgumentException("Name, Type, and Price are required and must be valid.");

059 var entity = new Service

060 {

061 Name = dto.Name,

062 Price = dto.Price,

063 Type = dto.Type,

064 Requirements = dto.Requirements,

065 IdRehearsalPoint = dto.IdRehearsalPoint

066 };

067 var created = await \_repository.AddAsync(entity);

068 return new ServiceDto

069 {

070 Id = created.Id,

071 Name = created.Name,

072 Price = created.Price,

073 Type = created.Type,

074 Requirements = created.Requirements,

075 IdRehearsalPoint = created.IdRehearsalPoint

076 };

077 }

078 public async Task UpdateAsync(int id, ServiceDto dto)

079 {

080 if (string.IsNullOrEmpty(dto.Name) || string.IsNullOrEmpty(dto.Type) || dto.Price <= 0)

081 throw new ArgumentException("Name, Type, and Price are required and must be valid.");

082 var entity = await \_repository.GetByIdAsync(id);

083 if (entity == null)

084 throw new KeyNotFoundException($"Service with ID {id} not found.");

085 entity.Name = dto.Name;

086 entity.Price = dto.Price;

087 entity.Type = dto.Type;

088 entity.Requirements = dto.Requirements;

089 entity.IdRehearsalPoint = dto.IdRehearsalPoint;

090 await \_repository.UpdateAsync(entity);

091 }

092 public async Task DeleteAsync(int id)

093 {

094 var entity = await \_repository.GetByIdAsync(id);

095 if (entity == null)

096 throw new KeyNotFoundException($"Service with ID {id} not found.");

097 await \_repository.DeleteAsync(id);

098 }

099 }

Файл StaffService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class StaffService : IStaffService

009 {

010 private readonly IStaffRepository \_repository;

011 public StaffService(IStaffRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<StaffDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new StaffDto

019 {

020 Id = e.Id,

021 FullName = e.FullName,

022 Address = e.Address,

023 Experience = e.Experience,

024 Phone = e.Phone,

025 Age = e.Age,

026 IdRehearsalPoint = e.IdRehearsalPoint

027 });

028 }

029 public async Task<StaffDto?> GetByIdAsync(int id)

030 {

031 var entity = await \_repository.GetByIdAsync(id);

032 if (entity == null) return null;

033 return new StaffDto

034 {

035 Id = entity.Id,

036 FullName = entity.FullName,

037 Address = entity.Address,

038 Experience = entity.Experience,

039 Phone = entity.Phone,

040 Age = entity.Age,

041 IdRehearsalPoint = entity.IdRehearsalPoint

042 };

043 }

044 public async Task<IEnumerable<StaffDto>> GetFilteredAsync(string? fullName, int? minAge, int? idRehearsalPoint)

045 {

046 var entities = await \_repository.GetFilteredAsync(fullName, minAge, idRehearsalPoint);

047 return entities.Select(e => new StaffDto

048 {

049 Id = e.Id,

050 FullName = e.FullName,

051 Address = e.Address,

052 Experience = e.Experience,

053 Phone = e.Phone,

054 Age = e.Age,

055 IdRehearsalPoint = e.IdRehearsalPoint

056 });

057 }

058 public async Task<StaffDto> CreateAsync(StaffDto dto)

059 {

060 if (string.IsNullOrEmpty(dto.FullName) || string.IsNullOrEmpty(dto.Phone) || dto.Age <= 0)

061 throw new ArgumentException("FullName, Phone, and Age are required and must be valid.");

062 var entity = new Staff

063 {

064 FullName = dto.FullName,

065 Address = dto.Address,

066 Experience = dto.Experience,

067 Phone = dto.Phone,

068 Age = dto.Age,

069 IdRehearsalPoint = dto.IdRehearsalPoint

070 };

071 var created = await \_repository.AddAsync(entity);

072 return new StaffDto

073 {

074 Id = created.Id,

075 FullName = created.FullName,

076 Address = created.Address,

077 Experience = created.Experience,

078 Phone = created.Phone,

079 Age = created.Age,

080 IdRehearsalPoint = created.IdRehearsalPoint

081 };

082 }

083 public async Task UpdateAsync(int id, StaffDto dto)

084 {

085 if (string.IsNullOrEmpty(dto.FullName) || string.IsNullOrEmpty(dto.Phone) || dto.Age <= 0)

086 throw new ArgumentException("FullName, Phone, and Age are required and must be valid.");

087 var entity = await \_repository.GetByIdAsync(id);

088 if (entity == null)

089 throw new KeyNotFoundException($"Staff with ID {id} not found.");

090 entity.FullName = dto.FullName;

091 entity.Address = dto.Address;

092 entity.Experience = dto.Experience;

093 entity.Phone = dto.Phone;

094 entity.Age = dto.Age;

095 entity.IdRehearsalPoint = dto.IdRehearsalPoint;

096 await \_repository.UpdateAsync(entity);

097 }

098 public async Task DeleteAsync(int id)

099 {

100 var entity = await \_repository.GetByIdAsync(id);

101 if (entity == null)

102 throw new KeyNotFoundException($"Staff with ID {id} not found.");

103 await \_repository.DeleteAsync(id);

104 }

105 }

Файл UserService.cs:

001 using RehearsalStudio.Application.DTOs;

002 using RehearsalStudio.Application.Interfaces;

003 using RehearsalStudio.Domain.Entities;

004 using System;

005 using System.Collections.Generic;

006 using System.Threading.Tasks;

007 namespace RehearsalStudio.Application.Services;

008 public class UserService : IUserService

009 {

010 private readonly IUserRepository \_repository;

011 public UserService(IUserRepository repository)

012 {

013 \_repository = repository;

014 }

015 public async Task<IEnumerable<UserDto>> GetAllAsync()

016 {

017 var entities = await \_repository.GetAllAsync();

018 return entities.Select(e => new UserDto

019 {

020 Id = e.Id,

021 FullName = e.FullName,

022 Phone = e.Phone,

023 Email = e.Email,

024 RegistrationDate = e.RegistrationDate

025 });

026 }

027 public async Task<UserDto?> GetByIdAsync(int id)

028 {

029 var entity = await \_repository.GetByIdAsync(id);

030 if (entity == null) return null;

031 return new UserDto

032 {

033 Id = entity.Id,

034 FullName = entity.FullName,

035 Phone = entity.Phone,

036 Email = entity.Email,

037 RegistrationDate = entity.RegistrationDate

038 };

039 }

040 public async Task<IEnumerable<UserDto>> GetFilteredAsync(string? fullName, string? email)

041 {

042 var entities = await \_repository.GetFilteredAsync(fullName, email);

043 return entities.Select(e => new UserDto

044 {

045 Id = e.Id,

046 FullName = e.FullName,

047 Phone = e.Phone,

048 Email = e.Email,

049 RegistrationDate = e.RegistrationDate

050 });

051 }

052 public async Task<UserDto> CreateAsync(UserDto dto)

053 {

054 if (string.IsNullOrEmpty(dto.FullName) || string.IsNullOrEmpty(dto.Phone) || string.IsNullOrEmpty(dto.Email))

055 throw new ArgumentException("FullName, Phone, and Email are required.");

056 var entity = new User

057 {

058 FullName = dto.FullName,

059 Phone = dto.Phone,

060 Email = dto.Email,

061 RegistrationDate = dto.RegistrationDate

062 };

063 var created = await \_repository.AddAsync(entity);

064 return new UserDto

065 {

066 Id = created.Id,

067 FullName = created.FullName,

068 Phone = created.Phone,

069 Email = created.Email,

070 RegistrationDate = created.RegistrationDate

071 };

072 }

073 public async Task UpdateAsync(int id, UserDto dto)

074 {

075 if (string.IsNullOrEmpty(dto.FullName) || string.IsNullOrEmpty(dto.Phone) || string.IsNullOrEmpty(dto.Email))

076 throw new ArgumentException("FullName, Phone, and Email are required.");

077 var entity = await \_repository.GetByIdAsync(id);

078 if (entity == null)

079 throw new KeyNotFoundException($"User with ID {id} not found.");

080 entity.FullName = dto.FullName;

081 entity.Phone = dto.Phone;

082 entity.Email = dto.Email;

083 entity.RegistrationDate = dto.RegistrationDate;

084 await \_repository.UpdateAsync(entity);

085 }

086 public async Task DeleteAsync(int id)

087 {

088 var entity = await \_repository.GetByIdAsync(id);

089 if (entity == null)

090 throw new KeyNotFoundException($"User with ID {id} not found.");

091 await \_repository.DeleteAsync(id);

092 }

093 }

Файл Class1.cs:

001 ﻿namespace RehearsalStudio.Domain;

002 public class Class1

003 {

004 }

Файл Booking.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class Booking

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public DateTime Time { get; set; }

013 public int? Duration { get; set; }

014 [Required]

015 public int Cost { get; set; }

016 [Required]

017 public DateTime CreationDate { get; set; }

018 [Required]

019 public string Status { get; set; } = string.Empty;

020 [Required]

021 public int NumberOfPeople { get; set; }

022 public int? IdRoom { get; set; }

023 [ForeignKey("IdRoom")]

024 public Room? Room { get; set; }

025 public int? IdUser { get; set; }

026 [ForeignKey("IdUser")]

027 public User? User { get; set; }

028 public List<ServiceBooking> ServiceBookings { get; set; } = new();

029 public List<EquipmentBooking> EquipmentBookings { get; set; } = new();

030 }

Файл Equipment.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class Equipment

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public string Name { get; set; } = string.Empty;

013 [Required]

014 public string Type { get; set; } = string.Empty;

015 [Required]

016 public string Brand { get; set; } = string.Empty;

017 [Required]

018 public string Model { get; set; } = string.Empty;

019 [Required]

020 public string Condition { get; set; } = string.Empty;

021 public int? IdRehearsalPoint { get; set; }

022 [ForeignKey("IdRehearsalPoint")]

023 public RehearsalPoint? RehearsalPoint { get; set; }

024 public List<EquipmentBooking> EquipmentBookings { get; set; } = new();

025 }

Файл EquipmentBooking.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class EquipmentBooking

007 {

008 [Key]

009 [Column(Order = 0)]

010 public int IdEquipment { get; set; }

011 [Key]

012 [Column(Order = 1)]

013 public int IdBooking { get; set; }

014 [ForeignKey("IdEquipment")]

015 public Equipment? Equipment { get; set; }

016 [ForeignKey("IdBooking")]

017 public Booking? Booking { get; set; }

018 }

Файл RehearsalPont.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class RehearsalPoint

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 public float? Rating { get; set; }

012 [Required]

013 public string ContactNumber { get; set; } = string.Empty;

014 public string Schedule { get; set; } = string.Empty;

015 [Required]

016 public string Name { get; set; } = string.Empty;

017 [Required]

018 public string Address { get; set; } = string.Empty;

019 public List<Room> Rooms { get; set; } = new();

020 public List<Service> Services { get; set; } = new();

021 public List<Equipment> Equipment { get; set; } = new();

022 public List<Staff> Staff { get; set; } = new();

023 }

Файл Room.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class Room

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public string Name { get; set; } = string.Empty;

013 public bool AirConditioner { get; set; } = false;

014 [Required]

015 public int Price { get; set; }

016 public bool RecordingSupport { get; set; } = false;

017 [Required]

018 public int Area { get; set; }

019 public int? IdRehearsalPoint { get; set; }

020 [ForeignKey("IdRehearsalPoint")]

021 public RehearsalPoint? RehearsalPoint { get; set; }

022 public List<Booking> Bookings { get; set; } = new();

023 }

Файл Service.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class Service

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public string Name { get; set; } = string.Empty;

013 [Required]

014 public int Price { get; set; }

015 [Required]

016 public string Type { get; set; } = string.Empty;

017 public string? Requirements { get; set; }

018 public int? IdRehearsalPoint { get; set; }

019 [ForeignKey("IdRehearsalPoint")]

020 public RehearsalPoint? RehearsalPoint { get; set; }

021 public List<ServiceBooking> ServiceBookings { get; set; } = new();

022 }

Файл ServiceBooking.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class ServiceBooking

007 {

008 [Key]

009 [Column(Order = 0)]

010 public int IdService { get; set; }

011 [Key]

012 [Column(Order = 1)]

013 public int IdBooking { get; set; }

014 [ForeignKey("IdService")]

015 public Service? Service { get; set; }

016 [ForeignKey("IdBooking")]

017 public Booking? Booking { get; set; }

018 }

Файл Staff.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class Staff

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public string FullName { get; set; } = string.Empty;

013 public string? Address { get; set; }

014 public int? Experience { get; set; }

015 [Required]

016 public string Phone { get; set; } = string.Empty;

017 [Required]

018 public int Age { get; set; }

019 public int? IdRehearsalPoint { get; set; }

020 [ForeignKey("IdRehearsalPoint")]

021 public RehearsalPoint? RehearsalPoint { get; set; }

022 }

Файл User.cs:

001 using System;

002 using System.Collections.Generic;

003 using System.ComponentModel.DataAnnotations;

004 using System.ComponentModel.DataAnnotations.Schema;

005 namespace RehearsalStudio.Domain.Entities;

006 public class User

007 {

008 [Key]

009 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

010 public int Id { get; set; }

011 [Required]

012 public string FullName { get; set; } = string.Empty;

013 [Required]

014 public string Phone { get; set; } = string.Empty;

015 [Required]

016 public string Email { get; set; } = string.Empty;

017 [Required]

018 public DateTime RegistrationDate { get; set; }

019 public List<Booking> Bookings { get; set; } = new();

020 }

Файл .NETCoreApp,Version=v9.0.AssemblyAttributes.cs:

001 // <autogenerated />

002 using System;

003 using System.Reflection;

004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]

Файл RehearsalStudio.Domain.AssemblyInfo.cs:

001 //------------------------------------------------------------------------------

002 // <auto-generated>

003 // This code was generated by a tool.

004 //

005 // Changes to this file may cause incorrect behavior and will be lost if

006 // the code is regenerated.

007 // </auto-generated>

008 //------------------------------------------------------------------------------

009 using System;

010 using System.Reflection;

011 [assembly: System.Reflection.AssemblyCompanyAttribute("RehearsalStudio.Domain")]

012 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

013 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

014 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

015 [assembly: System.Reflection.AssemblyProductAttribute("RehearsalStudio.Domain")]

016 [assembly: System.Reflection.AssemblyTitleAttribute("RehearsalStudio.Domain")]

017 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

018 // Generated by the MSBuild WriteCodeFragment class.

Файл RehearsalStudio.Domain.GlobalUsings.g.cs:

001 // <auto-generated/>

002 global using global::System;

003 global using global::System.Collections.Generic;

004 global using global::System.IO;

005 global using global::System.Linq;

006 global using global::System.Net.Http;

007 global using global::System.Threading;

008 global using global::System.Threading.Tasks;

Файл Class1.cs:

001 ﻿namespace RehearsalStudio.Infrastructure;

002 public class Class1

003 {

004 }

Файл .NETCoreApp,Version=v9.0.AssemblyAttributes.cs:

001 // <autogenerated />

002 using System;

003 using System.Reflection;

004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]

Файл RehearsalStudio.Infrastructure.AssemblyInfo.cs:

001 //------------------------------------------------------------------------------

002 // <auto-generated>

003 // This code was generated by a tool.

004 //

005 // Changes to this file may cause incorrect behavior and will be lost if

006 // the code is regenerated.

007 // </auto-generated>

008 //------------------------------------------------------------------------------

009 using System;

010 using System.Reflection;

011 [assembly: System.Reflection.AssemblyCompanyAttribute("RehearsalStudio.Infrastructure")]

012 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

013 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

014 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

015 [assembly: System.Reflection.AssemblyProductAttribute("RehearsalStudio.Infrastructure")]

016 [assembly: System.Reflection.AssemblyTitleAttribute("RehearsalStudio.Infrastructure")]

017 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

018 // Generated by the MSBuild WriteCodeFragment class.

Файл RehearsalStudio.Infrastructure.GlobalUsings.g.cs:

001 // <auto-generated/>

002 global using global::System;

003 global using global::System.Collections.Generic;

004 global using global::System.IO;

005 global using global::System.Linq;

006 global using global::System.Net.Http;

007 global using global::System.Threading;

008 global using global::System.Threading.Tasks;