## **Guia de Testes e Setup do Projeto AgendaPro**

Este documento descreve dois processos fundamentais para o desenvolvimento e validação do projeto: como realizar testes manuais nos endpoints da API usando a interface do Swagger e quais comandos foram utilizados para criar e inicializar o banco de dados.

### **1. Testando a Web API com a Interface Swagger**

Swagger UI é uma ferramenta integrada à nossa API que gera uma documentação interativa, permitindo a execução de chamadas para cada endpoint diretamente pelo navegador. É o nosso principal "playground" para validar o comportamento da API.

#### **Passo a Passo para Testar:**

1. **Iniciar a API:** Primeiro, certifique-se de que a sua Web API está em execução. No terminal, na raiz do projeto, execute o comando:  
   dotnet run --project AgendaPro.API  
     
   O seu navegador deverá abrir automaticamente na página do Swagger.
2. **Testar a Criação de Dados (POST)**
   * Expanda a secção do método POST do controller que deseja testar (ex: POST /api/Salas).
   * Clique no botão **"Try it out"**.
   * No campo de texto **"Request body"**, preencha o JSON com os dados do novo item (ex: nome e capacidade da sala).
   * Clique em **"Execute"**.
   * **Resultado esperado:** Uma resposta com o código **201 Created** e, no corpo da resposta, o id do novo item criado.
3. **Testar a Listagem de Dados (GET)**
   * Expanda a secção GET (ex: GET /api/Salas).
   * Clique em **"Try it out"** e depois em **"Execute"**.
   * **Resultado esperado:** Uma resposta com o código **200 OK** e, no corpo da resposta, uma lista em formato JSON com todos os itens cadastrados. **Copie o id** de um desses itens para usar nos testes seguintes.
4. **Testar a Busca por ID (GET /{id})**
   * Expanda a secção GET /{id}.
   * Clique em **"Try it out"**.
   * No campo de parâmetro **id**, cole o id que você copiou no passo anterior.
   * Clique em **"Execute"**.
   * **Resultado esperado:** Uma resposta 200 OK com os dados do item específico.
5. **Testar a Edição de Dados (PUT /{id})**
   * Expanda a secção PUT /{id}.
   * Clique em **"Try it out"**.
   * Preencha o parâmetro **id** com o ID do item a ser editado.
   * No **"Request body"**, preencha o JSON com os dados atualizados.
   * Clique em **"Execute"**.
   * **Resultado esperado:** Uma resposta **204 No Content**, indicando que a atualização foi bem-sucedida.
6. **Testar a Exclusão de Dados (DELETE /{id})**
   * Expanda a secção DELETE /{id}.
   * Clique em **"Try it out"**.
   * Preencha o parâmetro **id** com o ID do item a ser excluído.
   * Clique em **"Execute"**.
   * **Resultado esperado:** Uma resposta **204 No Content**. Se você tentar listar os itens novamente, este não deverá mais aparecer.

### **2. Comandos de Criação do Banco de Dados (Entity Framework Core)**

Para criar a estrutura do banco de dados a partir das nossas classes de Domínio (abordagem Code-First), utilizamos as ferramentas de linha de comando do Entity Framework Core.

#### **Pré-requisito: Instalação da Ferramenta**

Antes de executar os comandos, é necessário ter a ferramenta dotnet-ef instalada.

dotnet tool install --global dotnet-ef

#### **Comando 1: Criar a "Migration" (A Planta Baixa)**

Este comando lê o nosso DbContext e as nossas entidades e gera um ficheiro de código C# que descreve como criar as tabelas e relacionamentos.

* **Comando Executado:**  
  dotnet ef migrations add InitialCreate --project AgendaPro.Infraestrutura --startup-project AgendaPro.API
* **O que ele faz:**
  + migrations add InitialCreate: Cria uma nova migração com o nome "InitialCreate".
  + --project AgendaPro.Infraestrutura: Especifica que o DbContext e os ficheiros de migração devem ser criados no projeto de Infraestrutura.
  + --startup-project AgendaPro.API: Usa o projeto da API como ponto de partida para encontrar as configurações, como a string de conexão no appsettings.json.

#### **Comando 2: Aplicar a Migration (Construir o Banco)**

Este comando pega na "planta baixa" gerada pelo comando anterior e executa-a no SQL Server para criar ou atualizar o banco de dados.

* **Comando Executado:**  
  dotnet ef database update --startup-project AgendaPro.API
* **O que ele faz:**
  + database update: Aplica todas as migrações pendentes ao banco de dados especificado na sua string de conexão. Se o banco não existir, ele é criado.

Após a execução bem-sucedida destes dois comandos, a base de dados AgendaProDb e todas as suas tabelas foram criadas e estão prontas para serem utilizadas pela API.