🚀 API Challenge Mottu (.NET) - Documentação Técnica 🏍️ 💨

Bem-vindo à documentação da API Challenge Mottu desenvolvida em C# .NET! Este documento detalha a arquitetura, endpoints e componentes chave do projeto.

📜 **Versão do Script de Conversão:** 1.1 [cite: 1]

🗓️ **Data de Criação do Arquivo Fonte:** 2025-05-22 10:25:12 [cite: 1]

🔗 **Projeto no GitHub (Referência Genérica):** [Challenge Mottu (conceitual)](https://github.com/carmipa/challenge_2025_1_semestre_mottu)

A equipe e turma mencionadas no projeto Java podem ser conceitualmente as mesmas.

🗺️ Navegação Rápida (Índice)

🌟 Introdução

Esta API, construída com C# e .NET 8, serve como backend para o sistema de gerenciamento do Challenge Mottu. Ela fornece funcionalidades CRUD (Create, Read, Update, Delete) e de busca para diversas entidades como Clientes, Veículos, Boxes, Pátios, entre outros. A documentação interativa via Swagger UI está configurada para auxiliar no desenvolvimento e testes. [cite: 1060, 1064, 1072]

📐 Estrutura do Projeto

O projeto está organizado da seguinte maneira dentro da pasta `ChallengeMuttuApi/`: [cite: 1]

ChallengeMuttuApi/ ├— ChallengeMuttuApi/ # Pasta principal do projeto C# │ ├— Controllers/ # 🕹️ Controladores da API (definem os endpoints) │ ├— Data/ # 💾 Contexto do Banco de Dados (AppDbContext) │ ├— Enums/ # 📊 Enumerações (ex: EstadoCivil) │ ├— Model/ # 🧱 Modelos de dados (Entidades do EF Core) │ ├— Properties/ # ⚙️ Configurações do projeto (ex: launchSettings.json) │ ├— bin/ # 🔩 Arquivos compilados (binários) │ ├— obj/ # 📄 Arquivos intermediários da compilação │ ├— wwwroot/ # 🌐 Raiz para arquivos estáticos (se houver) │ └— Program.cs # 🚀 Ponto de entrada e configuração da aplicação ├— .vs/ # 🛠️ Arquivos específicos do Visual Studio (geralmente ignorados pelo git) └— ChallengeMuttuApi.sln # 📄 Arquivo da Solução do Visual Studio (implícito)

Outras pastas dentro de `bin/` e `obj/` são geradas durante o processo de compilação e incluem arquivos de runtime, internacionalização, etc. [cite: 3, 4, 5, 6, 7, 8, 9, 10]

⚙️ Configuração da Aplicação (`Program.cs`) [cite: 12]

O arquivo `Program.cs` é o coração da configuração da aplicação .NET. Ele define como a aplicação é construída e executada.

🔧 Serviços Configurados

* **Controllers:** Adiciona suporte para controladores MVC/API. [cite: 1057]

* **API Explorer:** Necessário para a geração da documentação Swagger. [cite: 1057]

* **CORS (Cross-Origin Resource Sharing):** Configurado para permitir requisições de origens específicas (ex: `http://localhost:5181`, `https://localhost:7183`, `http://localhost:3000`). [cite: 1055, 1056]

* **DbContext (Oracle):** Configura o `AppDbContext` para usar o provedor Oracle com a string de conexão `OracleDb` do `appsettings.json`. [cite: 1058]

  * Logging do Entity Framework Core é habilitado para o console. [cite: 1058]

  * Log de dados sensíveis é habilitado (para desenvolvimento). [cite: 1058]

* **SwaggerGen:** Configura a geração da documentação OpenAPI (Swagger). [cite: 1060]

* **Logging:** Configurado para usar Console, Debug e EventSourceLogger. [cite: 1049, 1050]

* **HTTP Logging:** Adicionado para logar detalhes de requisições e respostas HTTP. [cite: 1051]

📜 Swagger & OpenAPI

A documentação da API é gerada usando Swagger (OpenAPI).

* **Configuração:** No `Program.cs`, através de `builder.Services.AddSwaggerGen()`. [cite: 1060]

* **Informações da API:**

  * **Título:** Challenge Muttu API [cite: 1060]

  * **Versão:** v1 [cite: 1060]

  * **Descrição:** Detalhes do projeto e equipe. [cite: 1060, 1061]

  * **Contato:** Equipe Metamind Solution. [cite: 1061, 1062]

  * **Licença:** Licença de Exemplo. [cite: 1062]

* **Comentários XML:** Tenta incluir comentários XML do assembly para enriquecer a documentação. [cite: 1064, 1065]

* **UI:** Disponível em `/swagger` (ex: `http://localhost:5181/swagger`). [cite: 1073]

🌐 CORS (Cross-Origin Resource Sharing)

Configurado para permitir que a API seja acessada de diferentes origens.

* **Política Padrão:** `AllowSpecificOrigins` (nome implícito). [cite: 1055]

* **Origens Permitidas:** `http://localhost:5181`, `https://localhost:7183`, `http://localhost:3000`. [cite: 1055]

* **Métodos e Cabeçalhos:** Permite qualquer método (`AllowAnyMethod()`) e qualquer cabeçalho (`AllowAnyHeader()`). [cite: 1055]

* **Middleware:** `app.UseCors()` é chamado para aplicar a política. [cite: 1075]

🗃️ Configuração do Banco de Dados (Oracle)

A aplicação utiliza Entity Framework Core para interagir com um banco de dados Oracle.

* **DbContext:** `AppDbContext` [cite: 543]

* **String de Conexão:** Obtida de `configuration.GetConnectionString("OracleDb")`. [cite: 1058]

* **Provedor:** `UseOracle()`. [cite: 1058]

* **Logging EF Core:** Habilitado para o console com nível `Information`. [cite: 1058]

* **Dados Sensíveis:** Log de dados sensíveis habilitado (recomendado apenas para desenvolvimento). [cite: 1058]

📝 Logging

A aplicação utiliza o sistema de logging do ASP.NET Core.

* **Provedores:**
  * Console [cite: 1049]
  * Debug [cite: 1049]
  * EventSourceLogger [cite: 1050]
* **HTTP Logging:**
  * Habilitado via `builder.Services.AddHttpLogging()`. [cite: 1051]

  * Loga todos os campos de requisição e resposta (`HttpLoggingFields.All`). [cite: 1051]

  * Limites para o corpo da requisição/resposta logado (4096 bytes). [cite: 1051]

  * Middleware `app.UseHttpLogging()` aplicado. [cite: 1071]

* **Debug CORS:** Um middleware customizado foi adicionado para logar informações sobre o processamento CORS das requisições. [cite: 1067, 1068, 1069, 1070]

🧱 Entidades do Domínio (Models)

Os modelos representam as estruturas de dados da aplicação, mapeadas para tabelas do banco de dados usando Entity Framework Core. Estão localizados no namespace `ChallengeMuttuApi.Model`. [cite: 11]

🧊 Box [cite: 11, 633]

Armazena informações sobre boxes ou compartimentos.

- 🔑 **`IdBox`** (int, PK, Identity) [cite: 640, 641]

- 🏷️ **`Nome`** (string, 50, Obrigatório) [cite: 642, 643]

- 🚦 **`Status`** (bool, Obrigatório - 'A' Ativo / 'I' Inativo no DB) [cite: 644, 645, 646]

- 🗓️ **`DataEntrada`** (DateTime, Obrigatório) [cite: 647, 648]

- 🗓️ **`DataSaida`** (DateTime, Obrigatório) [cite: 649, 650]

- 📝 **`Observacao`** (string?, 100) [cite: 651, 652]

- 🔗 **Relacionamentos:** `PatioBoxes` (ICollection<PatioBox>?), `VeiculoBoxes` (ICollection<VeiculoBox>?), `ZonaBoxes` (ICollection<ZonaBox>?) [cite: 653, 655, 657]

👤 Cliente [cite: 11, 659]

Informações sobre os clientes.

- 🔑 **`IdCliente`** (int, PK, Identity) [cite: 668, 669]

- 🗓️ **`DataCadastro`** (DateTime, Obrigatório, Padrão: Now) [cite: 670, 671, 672]

- 👱 **`Sexo`** (string, 2, Obrigatório, Validação: 'M', 'H', ou 'F') [cite: 673, 674, 675, 676, 679]

- 🏷️ **`Nome`** (string, 100, Obrigatório) [cite: 680, 681, 682]

- 🏷️ **`Sobrenome`** (string, 100, Obrigatório) [cite: 683, 684, 685]

- 🎂 **`DataNascimento`** (DateTime, Obrigatório) [cite: 686, 687]

- 🆔 **`Cpf`** (string, 11, Obrigatório, Único, Validação: 11 dígitos numéricos) [cite: 688, 689, 690, 691, 693]

- 🛠️ **`Profissao`** (string, 50, Obrigatório) [cite: 694, 695, 696]

- 👫 **`EstadoCivil`** (Enum: `EstadoCivil`, Obrigatório) [cite: 697, 698, 699, 700]

- 🔗 **Chaves Estrangeiras:** `TbEnderecoIdEndereco` (int, FK para Endereco), `TbContatoIdContato` (int, FK para Contato) [cite: 701, 702, 703, 704]

- 🔗 **Relacionamentos:** `Endereco` (Endereco?), `Contato` (Contato?), `ClienteVeiculos` (ICollection<ClienteVeiculo>?) [cite: 705, 707, 709]

🔗 ClienteVeiculo [cite: 11, 715]

Tabela de ligação entre Cliente e Veiculo.

- 🔑 **Chave Primária Composta:** `TbClienteIdCliente`, `TbClienteTbEnderecoIdEndereco`, `TbClienteTbContatoIdContato`, `TbVeiculoIdVeiculo` [cite: 587]

- 👤 **`TbClienteIdCliente`** (int, FK para Cliente) [cite: 722, 723]

- 🏠 **`TbClienteTbEnderecoIdEndereco`** (int, parte da PK composta, referenciando Endereco do Cliente) [cite: 725, 728]

- 📞 **`TbClienteTbContatoIdContato`** (int, parte da PK composta, referenciando Contato do Cliente) [cite: 730, 732]

- 🚗 **`TbVeiculoIdVeiculo`** (int, FK para Veiculo) [cite: 734, 736]

- 🔗 **Relacionamentos:** `Cliente` (Cliente?), `Veiculo` (Veiculo?) [cite: 737, 739]

📞 Contato [cite: 11, 741]

Informações de contato.

- 🔑 **`IdContato`** (int, PK, Identity) [cite: 748, 749]
- 📧 **`Email`** (string, 100, Obrigatório, Formato de Email) [cite: 750, 751]
- ☎️ **`Ddd`** (int, Obrigatório, Range: 11-99) [cite: 752, 753]
- 🌎 **`Ddi`** (int, Obrigatório, Range: 0-9999) [cite: 754, 755]
- 📞 **`Telefone1`** (string, 20, Obrigatório) [cite: 756, 757]
- 📞 **`Telefone2`** (string?, 20) [cite: 758, 759]
- 📞 **`Telefone3`** (string?, 20) [cite: 760, 761]
- 📱 **`Celular`** (string, 20, Obrigatório) [cite: 762, 763]
- 📝 **`Outro`** (string?, 100) [cite: 764, 765]
- 📝 **`Observacao`** (string?, 200) [cite: 766, 767]
- 🔗 **Relacionamentos:** `Clientes` (ICollection<Cliente>?), `ContatoPatios` (ICollection<ContatoPatio>?) [cite: 768, 771]

🔗 ContatoPatio [cite: 11, 773]

Tabela de ligação entre Contato e Patio.

- 🔑 **Chave Primária Composta:** `TbPatioIdPatio`, `TbContatoIdContato` [cite: 591]
- 🅿️ **`TbPatioIdPatio`** (int, FK para Patio) [cite: 776, 778]
- 📞 **`TbContatoIdContato`** (int, FK para Contato) [cite: 779, 781]
- 🔗 **Relacionamentos:** `Patio` (Patio?), `Contato` (Contato?) [cite: 782, 784]

🏠 Endereco [cite: 11, 786]

Detalhes de endereço.

- 🔑 **`IdEndereco`** (int, PK, Identity) [cite: 793, 794]
- 📮 **`Cep`** (string, 9, Obrigatório) [cite: 795, 796]
- 🔢 **`Numero`** (int, Obrigatório) [cite: 797, 798]
- 🛣️ **`Logradouro`** (string, 50, Obrigatório) [cite: 799, 800]
- 🏘️ **`Bairro`** (string, 50, Obrigatório) [cite: 801, 802]
- 🏙️ **`Cidade`** (string, 50, Obrigatório) [cite: 803, 804]
- 🗺️ **`Estado`** (string, 2, Obrigatório) [cite: 805, 806]
- 🌎 **`Pais`** (string, 50, Obrigatório) [cite: 807, 808]
- 📝 **`Complemento`** (string?, 60) [cite: 809, 810]
- 📝 **`Observacao`** (string?, 200) [cite: 811, 812]
- 🔗 **Relacionamentos:** `Clientes` (ICollection<Cliente>?), `EnderecoPatios` (ICollection<EnderecoPatio>?) [cite: 813, 816]

🔗 EnderecoPatio [cite: 11, 817]

Tabela de ligação entre Endereco e Patio.

- 🔑 **Chave Primária Composta:** `TbEnderecoIdEndereco`, `TbPatioIdPatio` [cite: 594]
- 🏠 **`TbEnderecoIdEndereco`** (int, FK para Endereco) [cite: 821, 823]
- 🅿️ **`TbPatioIdPatio`** (int, FK para Patio) [cite: 824, 826]
- 🔗 **Relacionamentos:** `Endereco` (Endereco?), `Patio` (Patio?) [cite: 827, 829]

🏢 Filial (Modelo Adicional) [cite: 11, 830]

Representa uma filial. (Este modelo não está listado nos DbSets do `AppDbContext` no trecho fornecido, então pode não estar persistido ou ser parte de outro contexto/uso).

- 🔑 **`FilialId`** (int) [cite: 831]

- 🏷️ **`Nome`** (string, 50, Obrigatório) [cite: 832, 833]

🅿️ Patio [cite: 11, 838]

Informações sobre pátios.

- 🔑 **`IdPatio`** (int, PK, Identity) [cite: 844, 845]
- 🏷️ **`NomePatio`** (string, 50, Obrigatório) [cite: 846, 847]
- 🗓️ **`DataEntrada`** (DateTime, Obrigatório) [cite: 848, 849]
- 🗓️ **`DataSaida`** (DateTime, Obrigatório) [cite: 850, 851]
- 📝 **`Observacao`** (string?, 100) [cite: 852, 853]
- 🔗 **Relacionamentos:** `ContatoPatios`, `EnderecoPatios`, `PatioBoxes`, `VeiculoPatios`, `ZonaPatios` (ICollection<T>?) [cite: 854, 856, 858, 860, 862]

🔗 PatioBox [cite: 11, 864]

Tabela de ligação entre Patio e Box.

- 🔑 **Chave Primária Composta:** `TbPatioIdPatio`, `TbBoxIdBox` [cite: 597]
- 🅿️ **`TbPatioIdPatio`** (int, FK para Patio) [cite: 867, 869]
- 🧊 **`TbBoxIdBox`** (int, FK para Box) [cite: 870, 872]
- 🔗 **Relacionamentos:** `Patio` (Patio?), `Box` (Box?) [cite: 873, 875]

🛰️ Rastreamento [cite: 11, 877]

Dados de rastreamento de veículos.

- 🔑 **`IdRastreamento`** (int, PK, Identity) [cite: 882, 883]
- 📍 **`IpsX`**, **`IpsY`**, **`IpsZ`** (decimal, NUMBER(38,8), Obrigatório) [cite: 884, 886, 887, 889, 890, 892]
- 📡 **`GprsLatitude`**, **`GprsLongitude`**, **`GprsAltitude`** (decimal, NUMBER(38,8), Obrigatório) [cite: 893, 895, 896, 898, 899, 901]
- 🔗 **Relacionamentos:** `VeiculoRastreamentos` (ICollection<VeiculoRastreamento>?) [cite: 902]

🏍️ Veiculo [cite: 12, 904]

Informações sobre veículos.

- 🔑 **`IdVeiculo`** (int, PK, Identity) [cite: 912, 913]
- 🏷️ **`Placa`** (string, 10, Obrigatório, Único) [cite: 914, 915, 916]
- 🆔 **`Renavam`** (string, 11, Obrigatório, Único) [cite: 917, 918, 919]
- 🔩 **`Chassi`** (string, 17, Obrigatório, Único) [cite: 920, 921, 922]
- 🏭 **`Fabricante`** (string, 50, Obrigatório) [cite: 923, 924]
- 📋 **`Modelo`** (string, 60, Obrigatório) [cite: 925, 926]
- ⚙️ **`Motor`** (string?, 30) [cite: 927, 928]
- 🗓️ **`Ano`** (int, Obrigatório) [cite: 929, 930]
- ⛽ **`Combustivel`** (string, 20, Obrigatório) [cite: 931, 932]
- 🔗 **Relacionamentos:** `ClienteVeiculos`, `VeiculoBoxes`, `VeiculoPatios`, `VeiculoRastreamentos`, `VeiculoZonas` (ICollection<T>?) [cite: 933, 935, 937, 939, 941]

🔗 VeiculoBox [cite: 12, 943]

Tabela de ligação entre Veiculo e Box.

- 🔑 **Chave Primária Composta:** `TbVeiculoIdVeiculo`, `TbBoxIdBox` [cite: 600]
- 🏍️ **`TbVeiculoIdVeiculo`** (int, FK para Veiculo) [cite: 946, 948]
- 🧊 **`TbBoxIdBox`** (int, FK para Box) [cite: 949, 951]
- 🔗 **Relacionamentos:** `Veiculo` (Veiculo?), `Box` (Box?) [cite: 952, 954]

🔗 VeiculoPatio [cite: 12, 955]

Tabela de ligação entre Veiculo e Patio.

- 🔑 **Chave Primária Composta:** `TbVeiculoIdVeiculo`, `TbPatioIdPatio` [cite: 603]

- 🏍️ **`TbVeiculoIdVeiculo`** (int, FK para Veiculo) [cite: 959, 961]
- 🅿️ **`TbPatioIdPatio`** (int, FK para Patio) [cite: 962, 964]
- 🔗 **Relacionamentos:** `Veiculo` (Veiculo?), `Patio` (Patio?) [cite: 965, 967]

🔗 VeiculoRastreamento [cite: 12, 969]

Tabela de ligação entre Veiculo e Rastreamento.

- 🔑 **Chave Primária Composta:** `TbVeiculoIdVeiculo`, `TbRastreamentoIdRastreamento` [cite: 606]
- 🏍️ **`TbVeiculoIdVeiculo`** (int, FK para Veiculo) [cite: 972, 974]
- 🛰️ **`TbRastreamentoIdRastreamento`** (int, FK para Rastreamento) [cite: 975, 977]
- 🔗 **Relacionamentos:** `Veiculo` (Veiculo?), `Rastreamento` (Rastreamento?) [cite: 978, 980]

🔗 VeiculoZona [cite: 12, 981]

Tabela de ligação entre Veiculo e Zona.

- 🔑 **Chave Primária Composta:** `TbVeiculoIdVeiculo`, `TbZonaIdZona` [cite: 609]
- 🏍️ **`TbVeiculoIdVeiculo`** (int, FK para Veiculo) [cite: 985, 987]
- 📍 **`TbZonaIdZona`** (int, FK para Zona) [cite: 988, 990]
- 🔗 **Relacionamentos:** `Veiculo` (Veiculo?), `Zona` (Zona?) [cite: 991, 993]

📍 Zona [cite: 12, 995]

Informações sobre zonas.

- 🔑 **`IdZona`** (int, PK, Identity) [cite: 1002, 1003]
- 🏷️ **`Nome`** (string, 50, Obrigatório) [cite: 1004, 1005]
- 🗓️ **`DataEntrada`** (DateTime, Obrigatório) [cite: 1006, 1007]
- 🗓️ **`DataSaida`** (DateTime, Obrigatório) [cite: 1008, 1009]

- 📝 **`Observacao`** (string?, 100) [cite: 1010, 1011]

- 🔗 **Relacionamentos:** `VeiculoZonas`, `ZonaBoxes`, `ZonaPatios` (ICollection<T>?) [cite: 1012, 1014, 1016]

🔗 ZonaBox [cite: 12, 1018]

Tabela de ligação entre Zona e Box.

- 🔑 **Chave Primária Composta:** `TbZonaIdZona`, `TbBoxIdBox` [cite: 612]

- 📍 **`TbZonaIdZona`** (int, FK para Zona) [cite: 1021, 1023]

- 🧊 **`TbBoxIdBox`** (int, FK para Box) [cite: 1024, 1026]

- 🔗 **Relacionamentos:** `Zona` (Zona?), `Box` (Box?) [cite: 1027, 1029]

🔗 ZonaPatio [cite: 12, 1031]

Tabela de ligação entre Zona e Patio.

- 🔑 **Chave Primária Composta:** `TbPatioIdPatio`, `TbZonaIdZona` [cite: 615]

- 🅿️ **`TbPatioIdPatio`** (int, FK para Patio) [cite: 1034, 1036]

- 📍 **`TbZonaIdZona`** (int, FK para Zona) [cite: 1037, 1039]

- 🔗 **Relacionamentos:** `Patio` (Patio?), `Zona` (Zona?) [cite: 1040, 1042]

📊 Enums

Enumerações utilizadas no projeto para representar conjuntos de valores constantes.

👫 EstadoCivil [cite: 11, 625]

Define os estados civis possíveis para um cliente.

- `Solteiro` [cite: 627]

- `Casado` [cite: 628]

- `Viuvo` (Representa 'Divorciado' no comentário original, mas o nome é Viuvo) [cite: 629]

- `Separado` [cite: 630]

- `Uniao_Estavel` (Mapeado de "Uni o Est vel" do DDL) [cite: 632]

💾 Contexto do Banco de Dados (`AppDbContext`) [cite: 2, 543]

O `AppDbContext` é a classe que gerencia a sessão com o banco de dados, permitindo consultar e salvar dados. Ele herda de `DbContext` do Entity Framework Core.

* **Construtor:** Recebe `DbContextOptions<AppDbContext>` para configuração. [cite: 545]

* **DbSets:** Define coleções `DbSet<T>` para cada entidade persistida, representando as tabelas do banco. [cite: 546, 548, 550, 552, 554, 556, 558, 560, 562, 564, 566, 568, 570, 572, 574, 576, 578, 580]

  * Exemplos: `public DbSet<Box> Boxes { get; set; }`[cite: 546], `public DbSet<Cliente> Clientes { get; set; }` [cite: 548]

* **`OnModelCreating(ModelBuilder modelBuilder)`:** Usado para configurar o modelo de dados via Fluent API. [cite: 582]

  * **Chaves Compostas:** Define as chaves primárias compostas para as tabelas de ligação (ex: `modelBuilder.Entity<ClienteVeiculo>().HasKey(...)`). [cite: 587, 591, 594, 597, 600, 603, 606, 609, 612, 615]

  * **Relacionamentos:** Configura os relacionamentos entre as entidades (ex: `HasOne(...).WithMany(...).HasForeignKey(...)`). [cite: 589, 590, 592, 593, 595, 596, 598, 599, 601, 602, 604, 605, 607, 608, 610, 611, 613, 614, 616, 617]

  * **Índices Únicos:** Define índices únicos para campos como `Cliente.Cpf`, `Veiculo.Placa`, etc. [cite: 618, 619, 620, 621]

* **Conversões de Propriedade:**
    * `Cliente.EstadoCivil` (Enum para string). [cite: 622]
    * `Box.Status` (bool para string 'A'/'I' e vice-versa). [cite: 623, 624]

🕹️ Endpoints da API (Controllers)

Os Controllers (`ChallengeMuttuApi.Controllers`) são responsáveis por expor os endpoints da API. [cite: 2] Eles utilizam o `AppDbContext` para interagir com o banco de dados. [cite: 18, 89, 164]

🧊 `BoxesController` [cite: 12, 17]

Gerencia operações CRUD para a entidade `Box`.

- **Rota Base:** `/api/Boxes` [cite: 18]

Método HTTP	Rota	📝 Descrição	📩 Request Body	📤 Response (Sucesso)	⚙️ Parâmetros
🟢 `GET`	`/`	Retorna lista de todos os boxes. [cite: 21, 22]	N/A	`IEnumerable<Box>` (200)[cite: 22, 25], (204) [cite: 22, 24]	N/A
🟢 `GET`	`/{id}`	Retorna um box pelo ID. [cite: 28, 29]	N/A	`Box` (200)[cite: 29, 32], (404) [cite: 29, 31]	`id` (Path, int) [cite: 29]
🟢 `GET`	`/search-by-name`	Pesquisa boxes por parte do nome. [cite: 35, 36]	N/A	`IEnumerable<Box>` (200)[cite: 37, 41], (204)[cite: 37, 40], (400) [cite: 37, 37]	`nome` (Query, string) [cite: 37]

| 🟢 `GET` | `/by-status/{status}` | Retorna boxes pelo status ('A' ou 'I'). [cite: 44, 45] | N/A | `IEnumerable<Box>` (200)[cite: 45, 55], (204)[cite: 46, 54], (400) [cite: 46, 46] | `status` (Path, string) [cite: 45, 46] |

| 🔵 `POST` | `/` | Cria um novo box. [cite: 58, 59] | `Box` [cite: 60] | `Box` (201 Criado) [cite: 59, 62] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um box existente. [cite: 67, 68] | `Box` [cite: 69] | N/A (204 Sem Conteúdo) [cite: 69, 75] | `id` (Path, int) [cite: 69]
|

| 🔴 `DELETE` | `/{id}` | Exclui um box pelo ID. [cite: 79, 80] | N/A | N/A (204 Sem Conteúdo) [cite: 80, 84] | `id` (Path, int) [cite: 80]
|

👤 `ClientesController` [cite: 11, 87]

Gerencia operações CRUD para a entidade `Cliente`.

- **Rota Base:** `/api/Clientes` [cite: 89]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------- | :---------- | :---------- | :---------- | :---------- |

| 🟢 `GET` | `/` | Retorna lista de todos os clientes. [cite: 93, 95] | N/A | `IEnumerable<Cliente>` (200)[cite: 95, 98], (204) [cite: 95, 97] | N/A
|

| 🟢 `GET` | `/{id}` | Retorna um cliente pelo ID. [cite: 100, 102] | N/A | `Cliente` (200)[cite: 102, 105], (404) [cite: 103, 104] | `id` (Path, int) [cite: 102] |

| 🟢 `GET` | `/by-cpf/{cpf}` | Retorna um cliente pelo CPF. [cite: 107, 109] | N/A | `Cliente` (200)[cite: 110, 113], (404)[cite: 110, 113], (400) [cite: 110, 111] | `cpf` (Path, string) [cite: 109] |

| 🟢 `GET` | `/search-by-name` | Pesquisa clientes por parte do nome. [cite: 115, 116] | N/A | `IEnumerable<Cliente>` (200)[cite: 116, 121], (204)[cite:

116, 119], (400) [cite: 117, 117] | `nome` (Query, string) [cite: 117]
|

| 🔵 `POST` | `/` | Cria um novo cliente. [cite: 123, 126] | `Cliente` [cite: 127] | `Cliente` (201 Criado) [cite: 126, 133] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um cliente existente. [cite: 138, 141] | `Cliente` [cite: 142] | N/A (204 Sem Conteúdo) [cite: 142, 150] | `id` (Path, int) [cite: 142] |

| 🔴 `DELETE` | `/{id}` | Exclui um cliente pelo ID. [cite: 154, 156] | N/A | N/A (204 Sem Conteúdo) [cite: 156, 160] | `id` (Path, int) [cite: 156]
|

📞 `ContatosController` [cite: 11, 162]

Gerencia operações CRUD para a entidade `Contato`.

- **Rota Base:** `/api/Contatos` [cite: 164]

| Método HTTP | Rota | 📝 Descrição | 📩 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :-------------------- | :-- | :-------------- | :---------------------- | :-- |

| 🟢 `GET` | `/` | Retorna lista de todos os contatos. [cite: 167, 168] | N/A | `IEnumerable<Contato>` (200)[cite: 168, 171], (204) [cite: 168, 170] | N/A |

| 🟢 `GET` | `/{id}` | Retorna um contato pelo ID. [cite: 173, 174] | N/A | `Contato` (200)[cite: 174, 177], (404) [cite: 174, 176] | `id` (Path, int) [cite: 174] |

| 🟢 `GET` | `/by-email/{email}` | Retorna um contato pelo email. [cite: 179, 180] | N/A | `Contato` (200)[cite: 180, 184], (404)[cite: 181, 183], (400) [cite: 181, 181] | `email` (Path, string) [cite: 180] |

| 🟢 `GET` | `/search-by-celular` | Pesquisa contatos por parte do celular. [cite: 187, 188] | N/A | `IEnumerable<Contato>` (200)[cite: 189, 192], (204)[cite:

189, 191], (400) [cite: 189, 189] | `celular` (Query, string) [cite: 189]
|

| 🔵 `POST` | `/` | Cria um novo contato. [cite: 195, 196] | `Contato` [cite: 197] | `Contato` (201 Criado) [cite: 196, 201] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um contato existente. [cite: 205, 206] | `Contato` [cite: 207] | N/A (204 Sem Conteúdo) [cite: 207, 215] | `id` (Path, int) [cite: 207] |

| 🔴 `DELETE` | `/{id}` | Exclui um contato pelo ID. [cite: 219, 220] | N/A | N/A (204 Sem Conteúdo) [cite: 220, 224] | `id` (Path, int) [cite: 220]
|

🏠 `EnderecosController` [cite: 11, 225]

Gerencia operações CRUD para a entidade `Endereco`.

- **Rota Base:** `/api/Enderecos` [cite: 228]

| Método HTTP | Rota | 📝 Descrição | 📩 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------- | :---------- | :---------- | :---------- | :---------- |

| 🟢 `GET` | `/` | Retorna lista de todos os endereços. [cite: 231, 232] | N/A | `IEnumerable<Endereco>` (200)[cite: 232, 235], (204) [cite: 232, 234] | N/A |

| 🟢 `GET` | `/{id}` | Retorna um endereço pelo ID. [cite: 237, 238] | N/A | `Endereco` (200)[cite: 238, 241], (404) [cite: 239, 240] | `id` (Path, int) [cite: 238] |

| 🟢 `GET` | `/by-cep/{cep}` | Retorna endereços pelo CEP. [cite: 243, 244] | N/A | `IEnumerable<Endereco>` (200)[cite: 244, 248], (204)[cite: 245, 247], (400) [cite: 245, 245] | `cep` (Path, string) [cite: 244] |

| 🟢 `GET` | `/search-by-location` | Pesquisa endereços por cidade e estado. [cite: 251, 252] | N/A | `IEnumerable<Endereco>` (200)[cite: 253, 258],

(204)[cite: 253, 257], (400) [cite: 253, 254] | `cidade` (Query, string), `estado` (Query, string) [cite: 253] |

| 🔵 `POST` | `/` | Cria um novo endereço. [cite: 261, 262] | `Endereco` [cite: 263] | `Endereco` (201 Criado) [cite: 262, 265] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza um endereço existente. [cite: 269, 270] | `Endereco` [cite: 271] | N/A (204 Sem Conteúdo) [cite: 271, 277] | `id` (Path, int) [cite: 271] |

| 🔴 `DELETE` | `/{id}` | Exclui um endereço pelo ID. [cite: 280, 281] | N/A | N/A (204 Sem Conteúdo) [cite: 281, 285] | `id` (Path, int) [cite: 281] |

🅿️ `PatiosController` [cite: 11, 286]

Gerencia operações CRUD para a entidade `Patio`.

- **Rota Base:** `/api/Patios` [cite: 289]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------- | :---------- | :---------- | :---------- | :---------- |

| 🟢 `GET` | `/` | Retorna lista de todos os pátios. [cite: 292, 293] | N/A | `IEnumerable<Patio>` (200)[cite: 293, 296], (204) [cite: 293, 295] | N/A |

| 🟢 `GET` | `/{id}` | Retorna um pátio pelo ID. [cite: 298, 299] | N/A | `Patio` (200)[cite: 299, 302], (404) [cite: 300, 301] | `id` (Path, int) [cite: 299] |

| 🟢 `GET` | `/search-by-name` | Pesquisa pátios por parte do nome. [cite: 304, 305] | N/A | `IEnumerable<Patio>` (200)[cite: 306, 309], (204)[cite: 306, 308], (400) [cite: 306, 306] | `nomePatio` (Query, string) [cite: 306] |

| 🟢 `GET` | `/by-date` | Retorna pátios por data de entrada/saída. [cite: 312, 313] | N/A | `IEnumerable<Patio>` (200)[cite: 314, 319], (204)[cite: 314,

319], (400) [cite: 314, 315] | `date` (Query, DateTime), `type` (Query, string - 'entrada' ou 'saida') [cite: 314] |

| 🔵 `POST` | `/` | Cria um novo pátio. [cite: 322, 323] | `Patio` [cite: 324] | `Patio` (201 Criado) [cite: 323, 326] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza um pátio existente. [cite: 330, 331] | `Patio` [cite: 332] | N/A (204 Sem Conteúdo) [cite: 332, 338] | `id` (Path, int) [cite: 332] |

| 🔴 `DELETE` | `/{id}` | Exclui um pátio pelo ID. [cite: 341, 342] | N/A | N/A (204 Sem Conteúdo) [cite: 342, 346] | `id` (Path, int) [cite: 342] |

🛰️ `RastreamentosController` [cite: 11, 348]

Gerencia operações CRUD para a entidade `Rastreamento`.

- **Rota Base:** `/api/Rastreamentos` [cite: 350]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------------------- | :-- | :---------- | :-------------------------- | :-- |

| 🟢 `GET` | `/` | Retorna lista de todos os rastreamentos. [cite: 353, 354] | N/A | `IEnumerable<Rastreamento>` (200)[cite: 354, 357], (204) [cite: 354, 356] | N/A |

| 🟢 `GET` | `/{id}` | Retorna um rastreamento pelo ID. [cite: 359, 360] | N/A | `Rastreamento` (200)[cite: 360, 363], (404) [cite: 361, 362] | `id` (Path, int) [cite: 360] |

| 🟢 `GET` | `/search-by-coordinates` | Pesquisa por range de Latitude/Longitude. [cite: 365, 366] | N/A | `IEnumerable<Rastreamento>` (200)[cite: 367, 373], (204)[cite: 367, 372], (400) [cite: 367, 369] | `minLat`, `maxLat`, `minLong`, `maxLong` (Query, decimal) [cite: 367] |

| 🟢 `GET` | `/by-ips-range` | Pesquisa por range de coordenadas IPS (x,y). [cite: 376, 377] | N/A | `IEnumerable<Rastreamento>` (200)[cite: 378, 384], (204)[cite: 378, 383], (400) [cite: 378, 380] | `minX`, `maxX`, `minY`, `maxY` (Query, decimal) [cite: 378] |

| 🔵 `POST` | `/` | Cria um novo rastreamento. [cite: 387, 388] | `Rastreamento` [cite: 389] | `Rastreamento` (201 Criado) [cite: 388, 391] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza um rastreamento existente. [cite: 395, 396] | `Rastreamento` [cite: 397] | N/A (204 Sem Conteúdo) [cite: 397, 403] | `id` (Path, int) [cite: 397] |

| 🔴 `DELETE` | `/{id}` | Exclui um rastreamento pelo ID. [cite: 406, 407] | N/A | N/A (204 Sem Conteúdo) [cite: 407, 411] | `id` (Path, int) [cite: 407] |

🏍️ `VeiculosController` [cite: 11, 413]

Gerencia operações CRUD para a entidade `Veiculo`.

- **Rota Base:** `/api/Veiculos` [cite: 415]

Método HTTP	Rota	📝 Descrição	📩 Request Body	📤 Response (Sucesso)	⚙️ Parâmetros
🟢 `GET`	`/`	Retorna lista de todos os veículos. [cite: 418, 419]	N/A	`IEnumerable<Veiculo>` (200)[cite: 419, 422], (204) [cite: 419, 421]	N/A

| 🟢 `GET` | `/{id}` | Retorna um veículo pelo ID. [cite: 424, 425] | N/A | `Veiculo` (200)[cite: 425, 428], (404) [cite: 426, 427] | `id` (Path, int) [cite: 425] |

| 🟢 `GET` | `/by-placa/{placa}` | Retorna um veículo pela placa. [cite: 430, 431] | N/A | `Veiculo` (200)[cite: 432, 436], (404)[cite: 432, 435], (400) [cite: 432, 433] | `placa` (Path, string) [cite: 431] |

| 🟢 `GET` | `/search-by-model` | Pesquisa veículos por parte do modelo. [cite: 439, 440] | N/A | `IEnumerable<Veiculo>` (200)[cite: 441, 445], (204)[cite: 441, 444], (400) [cite: 442, 442] | `modelo` (Query, string) [cite: 441]
|

| 🔵 `POST` | `/` | Cria um novo veículo. [cite: 448, 449] | `Veiculo` [cite: 450] | `Veiculo` (201 Criado) [cite: 450, 454] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um veículo existente. [cite: 457, 458] | `Veiculo` [cite: 459] | N/A (204 Sem Conteúdo) [cite: 459, 468] | `id` (Path, int) [cite: 459] |

| 🔴 `DELETE` | `/{id}` | Exclui um veículo pelo ID. [cite: 471, 472] | N/A | N/A (204 Sem Conteúdo) [cite: 472, 476] | `id` (Path, int) [cite: 472]
|

📍 `ZonasController` [cite: 12, 482]

Gerencia operações CRUD para a entidade `Zona`.

- **Rota Base:** `/api/Zonas` [cite: 484]

| Método HTTP | Rota | 📝 Descrição | 📩 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------- | :---------- | :---------- | :---------- | :---------- |
|

| 🟢 `GET` | `/` | Retorna lista de todas as zonas. [cite: 487, 488] | N/A | `IEnumerable<Zona>` (200)[cite: 488, 491], (204) [cite: 488, 490] | N/A
|

| 🟢 `GET` | `/{id}` | Retorna uma zona pelo ID. [cite: 493, 494] | N/A | `Zona` (200)[cite: 494, 497], (404) [cite: 495, 496] | `id` (Path, int) [cite: 494] |

| 🟢 `GET` | `/search-by-name` | Pesquisa zonas por parte do nome. [cite: 499, 500] | N/A | `IEnumerable<Zona>` (200)[cite: 501, 504], (204)[cite: 501, 503], (400) [cite: 501, 501] | `nome` (Query, string) [cite: 501]
|

| 🟢 `GET` | `/by-date` | Retorna zonas por data de entrada/saída. [cite: 507, 508] | N/A | `IEnumerable<Zona>` (200)[cite: 509, 514], (204)[cite: 509, 514], (400) [cite: 509, 510] | `date` (Query, DateTime), `type` (Query, string - 'entrada' ou 'saida') [cite: 509] |

| 🔵 `POST` | `/` | Cria uma nova zona. [cite: 517, 518] | `Zona` [cite: 519] | `Zona` (201 Criado) [cite: 518, 521] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza uma zona existente. [cite: 525, 526] | `Zona` [cite: 527] | N/A (204 Sem Conteúdo) [cite: 527, 533] | `id` (Path, int) [cite: 527] |

| 🔴 `DELETE` | `/{id}` | Exclui uma zona pelo ID. [cite: 536, 537] | N/A | N/A (204 Sem Conteúdo) [cite: 537, 541] | `id` (Path, int) [cite: 537] |

⛈️ `WeatherForecastController` (Exemplo) [cite: 12, 477]

Um controlador de exemplo, geralmente incluído em novos projetos ASP.NET Core Web API.

- **Rota Base:** `/[controller]` (ex: `/WeatherForecast`) [cite: 478]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response (Sucesso) | ⚙️ Parâmetros |

| :---------- | :---------------------- | :---------------------------------- | :-------------- | :--------------------- | :------------ |

| 🟢 `GET` | `GetWeatherForecast` | Retorna uma previsão do tempo. [cite: 480] | N/A | `IEnumerable<WeatherForecast>` | N/A |

🚦 Tratamento de Erros (Nos Controllers)

A aplicação lida com erros diretamente nos métodos dos controllers, utilizando blocos `try-catch` e retornando respostas HTTP apropriadas:

* **`Ok(data)`:** Retorna status `200 OK` com os dados. [cite: 25, 32, 41]

* **`NoContent()`:** Retorna status `204 No Content` quando uma lista está vazia ou uma operação de atualização/exclusão é bem-sucedida sem retornar dados. [cite: 24, 40, 75]

* **`NotFound("mensagem")`:** Retorna status `404 Not Found` quando um recurso específico não é encontrado. [cite: 31, 73, 82]

* **`BadRequest("mensagem")` ou `BadRequest(ModelState)`:** Retorna status `400 Bad Request` para dados de entrada inválidos ou falhas de validação do modelo. [cite: 37, 60, 70]

* **`CreatedAtAction(...)`:** Retorna status `201 Created` após a criação bem-sucedida de um recurso, incluindo a URI do novo recurso. [cite: 62, 133, 201]

* **`StatusCode(500, "mensagem")`:** Retorna status `500 Internal Server Error` para exceções inesperadas ou erros de banco de dados (`DbUpdateException`, `DbUpdateConcurrencyException`). [cite: 27, 34, 64]

Logs de erro são escritos no console (`Console.WriteLine`) em caso de exceções, com a sugestão de usar `ILogger` para um logging mais robusto em produção. [cite: 26, 33, 65]

Um middleware global adicional em `Program.cs` também captura exceções não tratadas que possam escapar dos controllers, logando-as e retornando uma resposta genérica de erro `500`. [cite: 1077, 1078]