🚀 API Mottu - Challenge FIAP 2025 🛵 🛞

Bem-vindo à documentação oficial da API Mottu! Este projeto foi desenvolvido pela equipe **Metamind Solutions** para o Challenge da FIAP do 1º Semestre de 2025.

📜 **Versão:** 1.0

🗓️ **Data de Criação do Arquivo:** 2025-05-22 09:54:24

🔗 **Projeto no GitHub:** [Mottu Challenge 2025](https://github.com/carmipa/challenge_2025_1_semestre_mottu)

🎓 **Turma:** 2TDSPV / 2TDSPZ

👥 **Equipe Metamind Solutions:**

* 👨‍💻 Arthur Bispo de Lima (RM557568): RM557568@fiap.com.br | [GitHub](https://github.com/ArthurBispo00)

* 👨‍💻 João Paulo Moreira (RM557808): RM557808@fiap.com.br | [GitHub](https://github.com/joao1015)

* 👨‍💻 Paulo André Carminati (RM557881): RM557881@fiap.com.br | [GitHub](https://github.com/carmipa)

🗺️ Navegação Rápida (Índice)

🌟 Introdução

Esta API RESTful é a espinha dorsal do sistema de gerenciamento para o Challenge Mottu. Ela oferece um conjunto robusto de funcionalidades para administrar Clientes, Veículos, Endereços, Contatos, Pátios, Boxes, Zonas e Rastreamentos. A documentação interativa e detalhada dos endpoints está disponível através do Swagger UI, que é lançado automaticamente ao iniciar a aplicação.

> ✨ **Objetivo:** Facilitar a integração e o desenvolvimento de aplicações cliente, fornecendo uma interface clara e bem documentada para interagir com os dados e funcionalidades do sistema Mottu.

🏗️ Arquitetura do Projeto

O projeto adota uma arquitetura padrão para aplicações Spring Boot, promovendo organização e manutenibilidade. Os principais diretórios e seus propósitos são:

main/ ├── java/ │ └── br/com/fiap/mottu/ │ ├── config/ # 🔧 Configurações (CORS, OpenAPI) │ ├── controller/ # 🕹️ Controladores REST (Endpoints) │ ├── dto/ # ✉️ Data Transfer Objects (Request/Response) │ │ ├── box/ │ │ ├── cliente/ │ │ ├── contato/ │ │ ├── endereco/ │ │ ├── patio/ │ │ ├── rastreamento/ │ │ ├── veiculo/ │ │ └── zona/ │ ├── exception/ # 🚦 Exceções customizadas │ │ └── handler/ # 🛠️ Manipulador global de exceções │ ├── external/ # 🌍 Integrações com serviços externos (ViaCEP) │ │ └── viacep/ │ ├── filter/ # 🔍 Classes de filtro para buscas │ ├── mapper/ # 🔄 Mapeadores de dados (MapStruct) │ ├── model/ # 🧱 Entidades JPA (Models) │ │ └── relacionamento/ # 🔗 Entidades de junção para N:M │ ├── repository/ # 🗄️ Repositórios JPA (Acesso a Dados) │ │ └── relacionamento/ # 🔗 Repositórios para junções │ ├── service/ # 💡 Lógica de Negócios (Services) │ ├── specification/ # ⚙️ Especificações JPA para consultas complexas │ └── MottuApplication.java # 🚀 Classe principal da aplicação └──

resources/ ├── static/ # 🎨 Arquivos estáticos (CSS, JS, Imagens) └── templates/ # 📄 Templates de visualização

---

⚙️ Configurações Essenciais

Detalhes sobre as configurações cruciais que moldam o comportamento da API.

📜 OpenAPI & Swagger

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

* **Classe de Configuração:** `br.com.fiap.mottu.config.OpenApiConfig` [cite: 21, 24, 25]

* **Interface UI:** Acessível em `http://localhost:{server.port}/{context-path}/swagger-ui/index.html` [cite: 39, 41] (o caminho exato pode variar com base nas propriedades `springdoc.swagger-ui.path` [cite: 39]).

* **Informações Detalhadas:**

  * **Título:** Challenge-2025-FIAP-TEMMU-METAMIND SOLUTIONS [cite: 27]

  * **Versão:** 1.0 [cite: 27]

  * **Descrição:** API RESTful para o Challenge Mottu - Gestão de Clientes, Veículos, Endereços, Contatos e mais. [cite: 28]

  * **Contato:** Metamind Solution (RM557568@fiap.com.br) [cite: 32, 33]

  * **Servidor Padrão:** `http://localhost:8080` [cite: 35]

🌐 CORS (Cross-Origin Resource Sharing)

Permite que a API seja consumida por aplicações frontend hospedadas em diferentes domínios.

* **Classe de Configuração:** `br.com.fiap.mottu.config.CorsConfig` [cite: 17]

* **Rotas Afetadas:** Todas (`/**`) [cite: 17]

* **Origens Permitidas (Exemplos):** `http://localhost:3000`[cite: 17], `https://seu-dominio-de-producao.com` [cite: 18]

* **Métodos HTTP:** GET, POST, DELETE, PUT, PATCH, OPTIONS, HEAD [cite: 18]

* **Cabeçalhos Permitidos:** Todos (`*`) [cite: 19]

* **Suporte a Credenciais:** Sim [cite: 19]

🚀 Aplicação Principal (`MottuApplication`)

Ponto de partida da sua aplicação Spring Boot.

* **Classe Principal:** `br.com.fiap.mottu.MottuApplication` [cite: 12]

* **Anotações Chave:**
  * `@SpringBootApplication`: Habilita a autoconfiguração do Spring Boot. [cite: 14]
  * `@EnableCaching`: Ativa o suporte a caching na aplicação. [cite: 14]
  * `@EnableJpaRepositories`: Descobre e configura os repositórios JPA. [cite: 14]
  * `@EnableJpaAuditing`: Permite o preenchimento automático de campos de auditoria (ex: `dataCadastro` em `Cliente` [cite: 813]). [cite: 14]

* **Beans Configurados:**
  * `RestTemplate`: Para chamadas HTTP síncronas. [cite: 13, 15]

  * `WebClient.Builder`: Para chamadas HTTP assíncronas/reativas (usado pelo `ViaCepService` [cite: 703]). [cite: 13, 16]

🖥️ Lançador do Navegador Swagger

Um utilitário para abrir a UI do Swagger automaticamente no navegador ao iniciar a aplicação.

* **Classe:** `br.com.fiap.mottu.config.SwaggerBrowserLauncher` [cite: 36]

* **Ativação:** Controlado pela propriedade `app.launch-swagger-on-startup:true` (padrão). [cite: 39]

* **Funcionamento:** Detecta o ambiente desktop e tenta abrir a URL do Swagger UI. [cite: 39, 42]

* **Logs:** Informa o status da tentativa de abertura. [cite: 42, 43, 44]

🧱 Entidades do Domínio (Models)

As entidades são o coração do modelo de dados, representando as tabelas do banco. Todas são anotadas com `@Entity` e utilizam Lombok para reduzir código boilerplate.

📦 Box

Representa um espaço de armazenamento ou vaga. [cite: 49, 580]

* 🔑 **ID:** `idBox` (Long, PK) [cite: 801]
* 🏷️ **Nome:** `nome` (String, 50) [cite: 801]
* 🚦 **Status:** `status` (String, 1 - 'L' Livre, 'O' Ocupado) [cite: 802]
* 🗓️ **Datas:** `dataEntrada`[cite: 803], `dataSaida` (LocalDate) [cite: 804]
* 📝 **Observação:** `observacao` (String, 100) [cite: 805]
* 🔗 **Relacionamentos:** `VeiculoBox` (1:N)[cite: 806], `ZonaBox` (1:N)[cite: 807], `PatioBox` (1:N) [cite: 808]

👤 Cliente

Informações detalhadas sobre os clientes. [cite: 110, 589, 812]

* 🔑 **ID:** `idCliente` (Long, PK) [cite: 813]
* 🗓️ **Cadastro:** `dataCadastro` (LocalDate, auditado - `@CreatedDate`) [cite: 811, 813]
* 👱 **Identificação:** `sexo` (String, 2 - 'M'/'H')[cite: 814], `nome` (String, 100)[cite: 815], `sobrenome` (String, 100) [cite: 816]
* 🎂 **Nascimento:** `dataNascimento` (LocalDate) [cite: 817]
* 🆔 **CPF:** `cpf` (String, 11, único) [cite: 818]
* 🛠️ **Profissional:** `profissao` (String, 50)[cite: 819], `estadoCivil` (String, 50) [cite: 820]
* 🔗 **Relacionamentos:** `Endereco` (N:1)[cite: 821], `Contato` (N:1)[cite: 822], `ClienteVeiculo` (1:N) [cite: 823]

📞 Contato

Dados de contato dos clientes ou pátios. [cite: 192, 195, 615, 825]

* 🔑 **ID:** `idContato` (Long, PK) [cite: 825]

* 📧 **Email:** `email` (String, 100, obrigatório) [cite: 826]

* ☎️ **Telefones:** `ddd` (Integer)[cite: 827], `ddi` (Integer)[cite: 828], `telefone1` (String, 20)[cite: 829], `telefone2`[cite: 830], `telefone3`[cite: 831], `celular` (String, 20, obrigatório) [cite: 832]

* 📝 **Extras:** `outro` (String, 100)[cite: 833], `observacao` (String, 200) [cite: 834]

* 🔗 **Relacionamentos:** `Cliente` (1:N via `clienteContatos` [cite: 835]), `ContatoPatio` (1:N) [cite: 836]

🏠 Endereco

Detalhes de endereçamento. [cite: 249, 251, 625]

* 🔑 **ID:** `idEndereco` (Long, PK) [cite: 839]

* 📮 **CEP:** `cep` (String, 9) [cite: 839]

* 🔢 **Número:** `numero` (Integer, 7 dígitos) [cite: 840]

* 🗺️ **Localização:** `logradouro` (String, 50)[cite: 841], `bairro` (String, 50)[cite: 842], `cidade` (String, 50)[cite: 843], `estado` (String, 2)[cite: 844], `pais` (String, 50) [cite: 845]

* 📝 **Extras:** `complemento` (String, 60)[cite: 846], `observacao` (String, 200) [cite: 847]

* 🔗 **Relacionamentos:** `Cliente` (1:N via `clienteEnderecos` [cite: 848]), `EnderecoPatio` (1:N) [cite: 849]

🅿️ Patio

Áreas de estacionamento ou armazenamento de veículos. [cite: 307]

* 🔑 **ID:** `idPatio` (Long, PK) [cite: 851]

* 🏷️ **Nome:** `nomePatio` (String, 50) [cite: 852]

* 🗓️ **Datas:** `dataEntrada`[cite: 853], `dataSaida` (LocalDate) [cite: 854]

* 📝 **Observação:** `observacao` (String, 100) [cite: 855]

* 🔗 **Relacionamentos:** `ContatoPatio`[cite: 856], `EnderecoPatio`[cite: 857], `VeiculoPatio`[cite: 858], `ZonaPatio`[cite: 859], `PatioBox` (um-para-muitos, representando associações) [cite: 860]

🛰️ Rastreamento

Registros de geolocalização dos veículos. [cite: 402, 405, 645, 862]

* 🔑 **ID:** `idRastreamento` (Long, PK) [cite: 867]

* 📍 **IPS:** `ipsX` [cite: 870], `ipsY` [cite: 871], `ipsZ` (BigDecimal, precisão 7, escala 3) [cite: 872]

* 📡 **GPRS:** `gprsLatitude` (BigDecimal, 11,6)[cite: 873], `gprsLongitude` (BigDecimal, 11,6)[cite: 874], `gprsAltitude` (BigDecimal, 7,2) [cite: 875]

* 🕒 **Registro:** `dataHoraRegistro` (LocalDateTime, auditado - `@CreationTimestamp`) [cite: 876, 877]

* 🔗 **Relacionamentos:** `VeiculoRastreamento` (1:N) [cite: 877]

🏍️ Veiculo

Informações sobre os veículos gerenciados. [cite: 458, 461, 666, 879]

* 🔑 **ID:** `idVeiculo` (Long, PK) [cite: 879]

* 🏷️ **Identificadores:** `placa` (String, 10, único)[cite: 880], `renavam` (String, 11, único)[cite: 881], `chassi` (String, 17, único) [cite: 882]

* 🏭 **Detalhes:** `fabricante` (String, 50)[cite: 883], `modelo` (String, 60)[cite: 884], `motor` (String, 30)[cite: 885], `ano` (Integer) [cite: 886]

* ⛽ **Combustível:** `combustivel` (String, 20) [cite: 887]

* 🔗 **Relacionamentos:** `ClienteVeiculo`[cite: 888], `VeiculoBox`[cite: 889], `VeiculoPatio`[cite: 890], `VeiculoRastreamento`[cite: 891], `VeiculoZona` (um-para-muitos, representando associações) [cite: 892]

📍 Zona

Setores ou áreas específicas dentro de um pátio ou localidade. [cite: 524, 527, 674, 894]

* 🔑 **ID:** `idZona` (Long, PK) [cite: 895]
* 🏷️ **Nome:** `nome` (String, 50) [cite: 895]
* 🗓️ **Datas:** `dataEntrada`[cite: 896], `dataSaida` (LocalDate) [cite: 897]
* 📝 **Observação:** `observacao` (String, 100) [cite: 898]
* 🔗 **Relacionamentos:** `VeiculoZona`[cite: 899], `ZonaBox`[cite: 900], `ZonaPatio` (um-para-muitos, representando associações) [cite: 901]

🔗 Entidades de Relacionamento

Para gerenciar as cardinalidades Muitos-para-Muitos, o sistema utiliza tabelas de junção, cada uma representada por uma entidade com chave primária composta (`@EmbeddedId`).

* **Cliente 🤝 Veiculo:** `ClienteVeiculo` (ID: `ClienteVeiculoId`) [cite: 903, 909]

* **Contato 🤝 Patio:** `ContatoPatio` (ID: `ContatoPatioId`) [cite: 913, 914, 920]

* **Endereco 🤝 Patio:** `EnderecoPatio` (ID: `EnderecoPatioId`) [cite: 922, 923, 929]

* **Patio 🤝 Box:** `PatioBox` (ID: `PatioBoxId`) [cite: 931, 932, 939]

* **Veiculo 🤝 Box:** `VeiculoBox` (ID: `VeiculoBoxId`) [cite: 941, 942, 948]

* **Veiculo 🤝 Patio:** `VeiculoPatio` (ID: `VeiculoPatioId`) [cite: 950, 951, 957]

* **Veiculo 🤝 Rastreamento:** `VeiculoRastreamento` (ID: `VeiculoRastreamentoId`) [cite: 959, 960, 966]

* **Veiculo 🤝 Zona:** `VeiculoZona` (ID: `VeiculoZonaId`) [cite: 968, 969, 975]

* **Zona 🤝 Box:** `ZonaBox` (ID: `ZonaBoxId`) [cite: 977, 978, 984]

* **Zona 🤝 Patio:** `ZonaPatio` (ID: `ZonaPatioId`) [cite: 986, 987, 993]

🕹️ Endpoints da API (Controllers)

Os Controllers são a porta de entrada da API, responsáveis por receber requisições HTTP, delegar o processamento para a camada de serviço e retornar respostas adequadas.

🧊 `BoxController`

Endpoints para o gerenciamento de Boxes.

- **Rota Base:** `/api/boxes` [cite: 57]

Método HTTP	Rota	📝 Descrição	📥 Request Body	📤 Response Body (Sucesso 2xx)	⚙️ Parâmetros
🟢 `GET`	`/`	Lista boxes com paginação.	N/A	`Page<BoxResponseDto>`	`page` (Query, Int, opc, def:0)[cite: 64], `size` (Query, Int, opc, def:10)[cite: 64], `sort` (Query, Str, opc, ex:"nome,asc") [cite: 64]

| 🟢 `GET` | `/{id}` | Busca box por ID. | N/A | `BoxResponseDto` | `id` (Path, Long) [cite: 73]
|

| 🟢 `GET` | `/search` | Busca boxes por filtro e paginação. | N/A | `Page<BoxResponseDto>` | `BoxFilter` (Query, opc)[cite: 81], `page` (Query, Int, opc, def:0)[cite: 78], `size` (Query, Int, opc, def:10)[cite: 78], `sort` (Query, Str, opc) [cite: 78] |

| 🔵 `POST` | `/` | Cria um novo box. | `BoxRequestDto` [cite: 89] | `BoxResponseDto` (201 Criado) [cite: 86] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um box existente. | `BoxRequestDto` [cite: 98] | `BoxResponseDto` [cite: 94] | `id` (Path, Long) [cite: 98]
|

| 🔴 `DELETE` | `/{id}` | Deleta um box. | N/A | N/A (204 Sem Conteúdo) [cite: 102] | `id` (Path, Long) [cite: 104]
|

👤 `ClienteController`

Endpoints para o gerenciamento de Clientes e suas associações com Veículos.

- **Rota Base:** `/api/clientes` [cite: 121]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response Body (Sucesso 2xx) | ⚙️ Parâmetros
|

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

| 🟢 `GET` | `/` | Lista clientes com paginação. | N/A | `Page<ClienteResponseDto>` [cite: 133] | `page` (Query, Int, opc, def:0)[cite: 131], `size` (Query, Int, opc, def:10)[cite: 131], `sort` (Query, Str, opc, ex:"nome,asc") [cite: 131] |

| 🟢 `GET` | `/{id}` | Busca cliente por ID. | N/A | `ClienteResponseDto` [cite: 138] | `id` (Path, Long) [cite: 140] |

| 🟢 `GET` | `/search` | Busca clientes por filtro e paginação. | N/A | `Page<ClienteResponseDto>` [cite: 147] | `ClienteFilter` (Query, opc)[cite: 148], `page` (Query, Int, opc, def:0)[cite: 145], `size` (Query, Int, opc, def:10)[cite: 145], `sort` (Query, Str, opc) [cite: 145] |

| 🔵 `POST` | `/` | Cria novo cliente (pode criar/associar Endereço e Contato de forma reativa). | `ClienteRequestDto` [cite: 154] | `Mono<ClienteResponseDto>` (201) [cite: 153] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza cliente (pode atualizar Endereço e Contato de forma reativa). | `ClienteRequestDto` [cite: 162] | `Mono<ClienteResponseDto>` [cite: 161] | `id` (Path, Long) [cite: 162] |

| 🔴 `DELETE` | `/{id}` | Deleta um cliente. | N/A | N/A (204 Sem Conteúdo) [cite: 166] | `id` (Path, Long) [cite: 167] |

| 🔵 `POST` | `/{clienteId}/enderecos/{enderecoId}/contatos/{contatoId}/veiculos/{veiculoId}/associar` | Associa veículo a cliente (usa IDs atuais de endereço/contato do cliente). | N/A | `String` (201 Criado) [cite: 172] | `clienteId` (Path, Long)[cite:

172], `enderecoId` (Path, Long)[cite: 173], `contatoId` (Path, Long)[cite: 173], `veiculoId` (Path, Long) [cite: 173]
|

| 🔴 `DELETE` | `/{clienteId}/enderecos/{enderecoId}/contatos/{contatoId}/veiculos/{veiculoId}/desassociar` | Desassocia veículo de cliente (usa IDs atuais de endereço/contato do cliente). | N/A | N/A (204 Sem Conteúdo) [cite: 178] | `clienteId` (Path, Long)[cite: 178], `enderecoId` (Path, Long)[cite: 178], `contatoId` (Path, Long)[cite: 179], `veiculoId` (Path, Long) [cite: 179]
|

| 🟢 `GET` | `/{clienteId}/veiculos` | Lista veículos de um cliente. | N/A | `Set<VeiculoResponseDto>` [cite: 184] | `clienteId` (Path, Long) [cite: 185]
|

📞 `ContatoController`

Endpoints para o gerenciamento de Contatos.

- **Rota Base:** `/api/contatos` [cite: 203]

| Método HTTP | Rota | 📝 Descrição | 📥 Request Body | 📤 Response Body (Sucesso 2xx) | ⚙️ Parâmetros
|

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

---

| 🟢 `GET` | `/` | Lista contatos com paginação. | N/A | `Page<ContatoResponseDto>` [cite: 212] | `page` (Query, Int, opc, def:0)[cite: 210], `size` (Query, Int, opc, def:10)[cite: 210], `sort` (Query, Str, opc, ex:"email,asc") [cite: 210] |

| 🟢 `GET` | `/{id}` | Busca contato por ID. | N/A | `ContatoResponseDto` [cite: 217] | `id` (Path, Long) [cite: 219] |

| 🟢 `GET` | `/search` | Busca contatos por filtro e paginação. | N/A | `Page<ContatoResponseDto>` [cite: 226] | `ContatoFilter` (Query, opc)[cite: 227], `page` (Query, Int, opc, def:0)[cite: 224], `size` (Query, Int, opc, def:10)[cite: 224], `sort` (Query, Str, opc, ex:"email,asc") [cite: 224] |

| 🔵 `POST` | `/` | Cria um novo contato. | `ContatoRequestDto` [cite: 232] | `ContatoResponseDto` (201) [cite: 231] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza um contato existente. | `ContatoRequestDto` [cite: 238] | `ContatoResponseDto` [cite: 237] | `id` (Path, Long) [cite: 238] |

| 🔴 `DELETE` | `/{id}` | Deleta um contato. | N/A | N/A (204 Sem Conteúdo) [cite: 242] | `id` (Path, Long) [cite: 243] |

🏠 `EnderecoController`

Endpoints para o gerenciamento de Endereços, com integração ViaCEP.

- **Rota Base:** `/api/enderecos` [cite: 260]

Método HTTP	Rota	📝 Descrição	📥 Request Body	📤 Response Body (Sucesso 2xx)	⚙️ Parâmetros
🟢 `GET`	`/`	Lista endereços com paginação.	N/A	`Page<EnderecoResponseDto>` [cite: 270]	`page` (Query, Int, opc, def:0)[cite: 268], `size` (Query, Int, opc, def:10)[cite: 268], `sort` (Query, Str, opc, ex:"cep,asc") [cite: 268]
🟢 `GET`	`/{id}`	Busca endereço por ID.	N/A	`EnderecoResponseDto` [cite: 275]	`id` (Path, Long) [cite: 276]
🟢 `GET`	`/search`	Busca endereços por filtro e paginação.	N/A	`Page<EnderecoResponseDto>` [cite: 283]	`EnderecoFilter` (Query, opc)[cite: 284], `page` (Query, Int, opc, def:0)[cite: 281], `size` (Query, Int, opc, def:10)[cite: 281], `sort` (Query, Str, opc, ex:"cep,asc") [cite: 281]
🔵 `POST`	`/`	Cria novo endereço (consulta ViaCEP de forma reativa).	`EnderecoRequestDto` [cite: 289]	`Mono<EnderecoResponseDto>` (201) [cite: 288]	N/A
🟠 `PUT`	`/{id}`	Atualiza endereço (pode consultar ViaCEP se CEP mudar, de forma reativa).	`EnderecoRequestDto` [cite: 295]	`Mono<EnderecoResponseDto>` [cite: 294]	`id` (Path, Long) [cite: 295]
🔴 `DELETE`	`/{id}`	Deleta um endereço.	N/A	N/A (204 Sem Conteúdo) [cite: 299]	`id` (Path, Long) [cite: 300]

🅿️ `PatioController`

Endpoints para o gerenciamento de Pátios e suas diversas associações.

- **Rota Base:** `/api/patios` [cite: 309]

Método HTTP	Rota	📝 Descrição	📥 Request Body	📤 Response Body (Sucesso 2xx)	⚙️ Parâmetros
🟢 `GET`	`/`	Lista pátios com paginação.	N/A	`Page<PatioResponseDto>` [cite: 322]	`page` (Query, Int, opc, def:0)[cite: 322], `size` (Query, Int, opc, def:10)[cite: 322], `sort` (Query, Str, opc, def:"nomePatio,asc") [cite: 322]
🟢 `GET`	`/{id}`	Busca pátio por ID.	N/A	`PatioResponseDto` [cite: 327]	`id` (Path, Long) [cite: 327]
🟢 `GET`	`/search`	Busca pátios por filtro e paginação.	N/A	`Page<PatioResponseDto>` [cite: 331]	`PatioFilter` (Query, opc)[cite: 332], `page` (Query, Int, opc, def:0)[cite: 331], `size` (Query, Int, opc, def:10)[cite: 331], `sort` (Query, Str, opc) [cite: 331]

| 🔵 `POST` | `/` | Cria um novo pátio. | `PatioRequestDto` [cite: 335] | `PatioResponseDto` (201) [cite: 336] | N/A |

| 🟠 `PUT` | `/{id}` | Atualiza um pátio existente. | `PatioRequestDto` [cite: 340] | `PatioResponseDto` [cite: 340] | `id` (Path, Long) [cite: 340] |

| 🔴 `DELETE` | `/{id}` | Deleta um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 343] | `id` (Path, Long) [cite: 344] |

| 🔵 `POST` | `/{patioId}/veiculos/{veiculoId}/associar` | Associa veículo a um pátio. | N/A | `String` (201 Criado) [cite: 346] | `patioId` (Path, Long)[cite: 346], `veiculoId` (Path, Long) [cite: 346] |

| 🔴 `DELETE` | `/{patioId}/veiculos/{veiculoId}/desassociar` | Desassocia veículo de um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 350] | `patioId` (Path, Long)[cite: 350], `veiculoId` (Path, Long) [cite: 350] |

| 🟢 `GET` | `/{patioId}/veiculos` | Lista veículos de um pátio. | N/A | `Set<VeiculoResponseDto>` [cite: 354] | `patioId` (Path, Long) [cite: 354] |

| 🔵 `POST` | `/{patioId}/zonas/{zonaId}/associar` | Associa zona a um pátio. | N/A | `String` (201 Criado) [cite: 357] | `patioId` (Path, Long)[cite: 357], `zonaId` (Path, Long) [cite: 357] |

| 🔴 `DELETE` | `/{patioId}/zonas/{zonaId}/desassociar` | Desassocia zona de um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 361] | `patioId` (Path, Long)[cite: 361], `zonaId` (Path, Long) [cite: 361] |

| 🟢 `GET` | `/{patioId}/zonas` | Lista zonas de um pátio. | N/A | `Set<ZonaResponseDto>` [cite: 365] | `patioId` (Path, Long) [cite: 365] |

| 🔵 `POST` | `/{patioId}/contatos/{contatoId}/associar` | Associa contato a um pátio. | N/A | `String` (201 Criado) [cite: 368] | `patioId`

(Path, Long)[cite: 368], `contatoId` (Path, Long) [cite: 368]
|

| 🔴 `DELETE` | `/{patioId}/contatos/{contatoId}/desassociar` | Desassocia contato de um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 372] | `patioId` (Path, Long)[cite: 372], `contatoId` (Path, Long) [cite: 372]
|

| 🟢 `GET` | `/{patioId}/contatos` | Lista contatos de um pátio. | N/A | `Set<ContatoResponseDto>` [cite: 376] | `patioId` (Path, Long) [cite: 376]
|

| 🔵 `POST` | `/{patioId}/enderecos/{enderecoId}/associar` | Associa endereço a um pátio. | N/A | `String` (201 Criado) [cite: 379] | `patioId` (Path, Long)[cite: 379], `enderecoId` (Path, Long) [cite: 379]
|

| 🔴 `DELETE` | `/{patioId}/enderecos/{enderecoId}/desassociar` | Desassocia endereço de um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 383] | `patioId` (Path, Long)[cite: 383], `enderecoId` (Path, Long) [cite: 383]
|

| 🟢 `GET` | `/{patioId}/enderecos` | Lista endereços de um pátio. | N/A | `Set<EnderecoResponseDto>` [cite: 387] | `patioId` (Path, Long) [cite: 387]
|

| 🔵 `POST` | `/{patioId}/boxes/{boxId}/associar` | Associa box a um pátio. | N/A | `String` (201 Criado) [cite: 390] | `patioId` (Path, Long)[cite: 390], `boxId` (Path, Long) [cite: 390]
|

| 🔴 `DELETE` | `/{patioId}/boxes/{boxId}/desassociar` | Desassocia box de um pátio. | N/A | N/A (204 Sem Conteúdo) [cite: 394] | `patioId` (Path, Long)[cite: 394], `boxId` (Path, Long) [cite: 394]
|

| 🟢 `GET` | `/{patioId}/boxes` | Lista boxes de um pátio. | N/A | `Set<BoxResponseDto>` [cite: 398] | `patioId` (Path, Long) [cite: 398]
|

🛰️ `RastreamentoController`

Endpoints para o gerenciamento de Rastreamentos de Veículos.

- **Rota Base:** `/api/rastreamentos` [cite: 413]

Método HTTP	Rota	📝 Descrição	📩 Request Body	📤 Response Body (Sucesso 2xx)	⚙️ Parâmetros
🟢 `GET`	`/`	Lista rastreamentos com paginação.	N/A	`Page<RastreamentoResponseDto>` [cite: 422]	`page` (Query, Int, opc, def:0)[cite: 420], `size` (Query, Int, opc, def:10)[cite: 420], `sort` (Query, Str, opc, def:"dataHoraRegistro,desc") [cite: 420]
🟢 `GET`	`/{id}`	Busca rastreamento por ID.	N/A	`RastreamentoResponseDto` [cite: 427]	`id` (Path, Long) [cite: 428]
🟢 `GET`	`/search`	Busca rastreamentos por filtro e paginação.	N/A	`Page<RastreamentoResponseDto>` [cite: 435]	`RastreamentoFilter` (Query, opc)[cite: 436], `page` (Query, Int, opc, def:0)[cite: 433], `size` (Query, Int, opc, def:10)[cite: 433], `sort` (Query, Str, opc, def:"dataHoraRegistro,desc") [cite: 433]

| 🔵 `POST` | `/` | Cria um novo rastreamento. | `RastreamentoRequestDto` [cite: 441] | `RastreamentoResponseDto` (201) [cite: 440] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um rastreamento existente. | `RastreamentoRequestDto` [cite: 447] | `RastreamentoResponseDto` [cite: 446] | `id` (Path, Long) [cite: 447]
|

| 🔴 `DELETE` | `/{id}` | Deleta um rastreamento. | N/A | N/A (204 Sem Conteúdo) [cite: 451] | `id` (Path, Long) [cite: 452]
|

🏍️ `VeiculoController`

Endpoints para o gerenciamento de Veículos, incluindo sua localização.

- **Rota Base:** `/api/veiculos` [cite: 470]

| Método HTTP | Rota | 📝 Descrição | 📩 Request Body | 📤 Response Body (Sucesso 2xx) | ⚙️ Parâmetros
|

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

| 🟢 `GET` | `/` | Lista veículos com paginação. | N/A | `Page<VeiculoResponseDto>` [cite: 478] | `page` (Query, Int, opc, def:0)[cite: 476], `size` (Query, Int, opc, def:10)[cite: 476], `sort` (Query, Str, opc, ex:"placa,asc") [cite: 476] |

| 🟢 `GET` | `/{id}` | Busca veículo por ID. | N/A | `VeiculoResponseDto` [cite: 483] | `id` (Path, Long) [cite: 485]
|

| 🟢 `GET` | `/search` | Busca veículos por filtro e paginação. | N/A | `Page<VeiculoResponseDto>` [cite: 491] | `VeiculoFilter` (Query, opc)[cite: 492], `page` (Query, Int, opc, def:0)[cite: 489], `size` (Query, Int, opc, def:10)[cite: 489], `sort` (Query, Str, opc) [cite: 489] |

| 🔵 `POST` | `/` | Cria um novo veículo. | `VeiculoRequestDto` [cite: 500] | `VeiculoResponseDto` (201) [cite: 497] | N/A
|

| 🟠 `PUT` | `/{id}` | Atualiza um veículo existente. | `VeiculoRequestDto` [cite: 508] | `VeiculoResponseDto` [cite: 504] | `id` (Path, Long) [cite: 508]
|

| 🔴 `DELETE` | `/{id}` | Deleta um veículo. | N/A | N/A (204 Sem Conteúdo) [cite: 511] | `id` (Path, Long) [cite: 512]
|

| 🟢 `GET` | `/{id}/localizacao` | Obtém localização de um veículo. | N/A | `VeiculoLocalizacaoResponseDto` [cite: 517] | `id` (Path, Long) [cite: 519]
|

📍 `ZonaController`

Endpoints para o gerenciamento de Zonas.

- **Rota Base:** `/api/zonas` [cite: 535]

Método HTTP	Rota	📝 Descrição	📥 Request Body	📤 Response Body (Sucesso 2xx)	⚙️ Parâmetros
🟢 `GET`	`/`	Lista zonas com paginação.	N/A	`Page<ZonaResponseDto>` [cite: 544]	`page` (Query, Int, opc, def:0)[cite: 542], `size` (Query, Int, opc, def:10)[cite: 542], `sort` (Query, Str, opc, ex:"nome,asc") [cite: 542]
🟢 `GET`	`/{id}`	Busca zona por ID.	N/A	`ZonaResponseDto` [cite: 549]	`id` (Path, Long) [cite: 550]
🟢 `GET`	`/search`	Busca zonas por filtro e paginação.	N/A	`Page<ZonaResponseDto>` [cite: 557]	`ZonaFilter` (Query, opc)[cite: 558], `page` (Query, Int, opc, def:0)[cite: 555], `size` (Query, Int, opc, def:10)[cite: 555], `sort` (Query, Str, opc) [cite: 555]
🔵 `POST`	`/`	Cria uma nova zona.	`ZonaRequestDto` [cite: 563]	`ZonaResponseDto` (201) [cite: 562]	N/A
🟠 `PUT`	`/{id}`	Atualiza uma zona existente.	`ZonaRequestDto` [cite: 569]	`ZonaResponseDto` [cite: 568]	`id` (Path, Long) [cite: 569]
🔴 `DELETE`	`/{id}`	Deleta uma zona.	N/A	N/A (204 Sem Conteúdo) [cite: 573]	`id` (Path, Long) [cite: 574]

🛠️ Camada de Serviço (Services)

A camada de serviço (`br.com.fiap.mottu.service`) é onde reside a lógica de negócios da aplicação. Cada serviço (ex: `ClienteService`[cite: 1086], `VeiculoService` [cite: 1363]) orquestra operações, interage com os repositórios para acesso a dados, utiliza mappers para conversão de DTOs, aplica regras de validação e de negócios, e gerencia transações (`@Transactional` [cite: 1065]).

✨ **Principais Responsabilidades:**

* Implementar os casos de uso do sistema.

* Coordenar o acesso a dados através dos repositórios.

* Garantir a integridade e consistência dos dados.

* Gerenciar transações (commit/rollback).

* Aplicar caching (`@Cacheable`[cite: 1063], `@CachePut`[cite: 1062], `@CacheEvict` [cite: 1062]) para otimizar consultas.

* Lançar exceções customizadas (`ResourceNotFoundException`[cite: 1061], `DuplicatedResourceException`[cite: 1061], `InvalidInputException` [cite: 1094]) para sinalizar erros de negócio.

* Interagir com serviços externos (ex: `EnderecoService` [cite: 1199] utilizando `ViaCepService` [cite: 1200]).

🗄️ Camada de Acesso a Dados (Repositories)

Os repositórios (`br.com.fiap.mottu.repository`) são interfaces que estendem `JpaRepository` e `JpaSpecificationExecutor` do Spring Data JPA. Eles abstraem a complexidade do acesso ao banco de dados, fornecendo uma API simples para operações CRUD (Criar, Ler, Atualizar, Deletar) e consultas mais avançadas.

✨ **Características:**

* **Operações CRUD Prontas:** Métodos como `save()`, `findById()`, `findAll()`, `deleteById()` são herdados.

* **Queries Derivadas:** O Spring Data JPA cria implementações de queries baseadas nos nomes dos métodos (ex: `findByCpf(String cpf)` no `ClienteRepository` [cite: 1004]).

* **Suporte a Specifications:** Permite a construção de queries dinâmicas e complexas usando a Criteria API do JPA, ideal para filtros.

* **Repositórios por Entidade:** Cada entidade principal e entidade de relacionamento possui sua interface de repositório (ex: `BoxRepository`[cite: 995], `ClienteVeiculoRepository` [cite: 1037]).

🔄 Mapeadores de Dados (Mappers)

O projeto utiliza **MapStruct** para automatizar o mapeamento entre Entidades JPA e DTOs. As interfaces de mapper (localizadas em `br.com.fiap.mottu.mapper`) são anotadas com `@Mapper(componentModel = "spring")`[cite: 731], permitindo que o MapStruct gere implementações que são beans Spring.

✨ **Benefícios:**

* **Redução de Código Boilerplate:** Evita a escrita manual de código de conversão.

* **Segurança de Tipos:** Verificações em tempo de compilação.

* **Performance:** Gera código Java simples e otimizado.

* **Principais Métodos Gerados:**

* `toEntity(RequestDto dto)`: Converte DTO de requisição para entidade (para criação). [cite: 732]

* `partialUpdate(RequestDto dto, @MappingTarget Entidade entidade)`: Atualiza uma entidade existente com dados de um DTO (ignora nulos no DTO). [cite: 733]

* `toResponseDto(Entidade entidade)`: Converte entidade para DTO de resposta. [cite: 734]

✉️ Objetos de Transferência de Dados (DTOs)

DTOs (`br.com.fiap.mottu.dto`) são classes simples usadas para encapsular dados que são transferidos entre as camadas da aplicação, especialmente para definir a "forma" dos dados nas requisições e respostas da API.

✨ **Tipos Comuns:**

* 🤌 **Request DTOs (`*RequestDto`):**

* Usados para receber dados do cliente (corpo de requisições POST/PUT). [cite: 46, 579]

    * Contêm validações (`jakarta.validation.constraints` como `@NotBlank`[cite: 580], `@Size`[cite: 580], `@NotNull`[cite: 582], `@Email` [cite: 605]).

    * Exemplo: `ClienteRequestDto` [cite: 588] contém campos para criar ou atualizar um cliente, incluindo `EnderecoRequestDto` [cite: 596] e `ContatoRequestDto` [cite: 597] aninhados.

* 📩 **Response DTOs (`*ResponseDto`):**

    * Usados para enviar dados da API para o cliente. [cite: 46, 586]

    * Definem quais campos da entidade são expostos.

    * Exemplo: `ClienteResponseDto` [cite: 599] mostra os dados de um cliente, incluindo seus `EnderecoResponseDto` [cite: 601] e `ContatoResponseDto`[cite: 601].

* 🗺️ **DTOs Específicos:**

    * `VeiculoLocalizacaoResponseDto`[cite: 650]: Estrutura específica para retornar a localização de um veículo[cite: 517], combinando dados do veículo, seu último rastreamento e associações atuais[cite: 651, 652, 653, 654, 655].

Todos os DTOs utilizam a anotação `@Value` do Lombok[cite: 580, 586], o que os torna imutáveis e gera automaticamente construtores, getters, `equals()`, `hashCode()` e `toString()`.

🔍 Filtros e Especificações de Consulta

Para buscas avançadas e flexíveis, o projeto emprega uma combinação de classes de Filtro e Especificações JPA.

* **Filtros (`br.com.fiap.mottu.filter.*Filter`):** [cite: 47]
  * São `records` Java que encapsulam os critérios de pesquisa que um cliente pode enviar via query parameters. [cite: 711, 713]
  * Exemplo: `ClienteFilter` [cite: 712] permite filtrar clientes por nome[cite: 713], CPF[cite: 713], cidade do endereço[cite: 714], placa do veículo[cite: 714], etc.

* **Especificações (`br.com.fiap.mottu.specification.*Specification`):** [cite: 1061]
  * Implementam a interface `Specification<T>` do Spring Data JPA. [cite: 1444]
  * Convertem os objetos `Filter` em `Predicate` da Criteria API do JPA, construindo dinamicamente as cláusulas `WHERE` das consultas SQL. [cite: 1445, 1446]
  * Permitem a criação de lógicas de filtro complexas, incluindo `JOINs` para buscar em entidades relacionadas. [cite: 1472]
  * Usam `query.distinct(true)` para evitar resultados duplicados ao usar `JOINs` em coleções. [cite: 1474]

Esta abordagem torna os endpoints de busca (`/search`) poderosos e flexíveis, permitindo combinações variadas de critérios de filtro.

🚦 Tratamento Global de Erros

O sistema possui um mecanismo centralizado para tratamento de exceções, localizado em `br.com.fiap.mottu.exception.handler.GlobalExceptionHandler`. [cite: 689, 690] Esta classe, anotada com `@ControllerAdvice`, intercepta exceções lançadas pelos controllers ou services e as converte em respostas HTTP padronizadas.

✨ **Exceções Customizadas e Seus Handlers:**

* ⚠️ **`ResourceNotFoundException`** ([caminho](br/com/fiap/mottu/exception/ResourceNotFoundException.java)) [cite: 683]

  * **Quando:** Um recurso específico (ex: Cliente com ID 999) não é encontrado. [cite: 686]

  * **HTTP Status:** `404 Not Found`. [cite: 684]

  * **Resposta:** JSON com `timestamp`[cite: 691], `status`[cite: 691], `error: "Não Encontrado"`[cite: 691], `message` (da exceção)[cite: 691], `path`[cite: 691].

* ⚠️ **`DuplicatedResourceException`** ([caminho](br/com/fiap/mottu/exception/DuplicatedResourceException.java)) [cite: 676]

  * **Quando:** Tentativa de criar um recurso que viola uma restrição de unicidade (ex: Cliente com CPF já existente). [cite: 679]

  * **HTTP Status:** `409 Conflict`. [cite: 677]

  * **Resposta:** JSON com `timestamp` [cite: 693], `status` [cite: 693], `error: "Conflito de Dados"` [cite: 693], `message` [cite: 693], `path` [cite: 693].

* ⚠️ **`InvalidInputException`** ([caminho](br/com/fiap/mottu/exception/InvalidInputException.java)) [cite: 681]

  * **Quando:** Dados de entrada inválidos que não são cobertos pela validação padrão do Bean Validation (ex: CEP não encontrado no ViaCEP). [cite: 682]

  * **HTTP Status:** `400 Bad Request`. [cite: 681]

  * **Resposta:** JSON com `timestamp` [cite: 695], `status` [cite: 695], `error: "Requisição Inválida"` [cite: 695], `message` [cite: 695], `path` [cite: 695].

> 💡 **Validações de DTOs:** Erros de validação em DTOs (definidos por anotações como `@NotBlank`, `@Size`) são geralmente tratados pelo Spring antes de atingirem esses handlers, resultando também em respostas `400 Bad Request`.

🌎 Integrações com Serviços Externos

📮 `ViaCepService`

Para enriquecer os dados de endereço, o sistema se integra ao **ViaCEP**, um serviço web gratuito para consulta de CEPs no Brasil.

- **Localização:** `br.com.fiap.mottu.external.viacep.ViaCepService` [cite: 703]

- **Objetivo:** Obter dados completos de um endereço (logradouro, bairro, cidade, UF) a partir de um CEP.

- **Funcionamento:**

  1. Recebe um CEP como string. [cite: 706]
  2. Limpa o CEP, removendo caracteres não numéricos. [cite: 706]
  3. Valida se o CEP limpo possui 8 dígitos; caso contrário, retorna um erro (`IllegalArgumentException`). [cite: 707]
  4. Utiliza `WebClient` para fazer uma requisição GET assíncrona para a API do ViaCEP (`https://viacep.com.br/ws/{CEP}/json/`). [cite: 705, 708]
  5. Mapeia a resposta JSON para o DTO `ViaCepResponse`. [cite: 708]

6. Se o ViaCEP indicar que o CEP não foi encontrado (`"erro": true` na resposta [cite: 702]), retorna um `Mono.empty()`. [cite: 709]

- **Uso Principal:** Injetado no `EnderecoService` [cite: 1209] para ser utilizado durante a criação (`criarEndereco` [cite: 1215]) e atualização (`atualizarEndereco` [cite: 1224]) de entidades `Endereco`.

Este mecanismo garante que os endereços cadastrados no sistema sejam validados e padronizados conforme os dados oficiais dos Correios, sempre que possível.