FIAP – Faculdade de Informática e Administração Paulista

Sprint 3 – Domain Driven Design using Java
IMREA – Sistema de Teleconsultas

Aluno: Leonardo José Pereira
RM: 563065

Aluno: Ícaro Santos
RM: 563070

Aluno: Fabrício Santos
RM: 563072

São Paulo – 2025

Objetivo e Escopo do Projeto

O sistema **IMREA - Sistema de Teleconsultas** foi desenvolvido com o objetivo de modernizar o atendimento remoto do Instituto de Medicina Física e Reabilitação (IMREA).

O objetivo desta 4ª Sprint foi de atualizar um sistema de console para uma **API Restful** robusta e completa, utilizando o framework **Spring Boot**. Esta API serve como o back-end que proverá todos os dados e lógica de negócio necessários para uma aplicação front-end web ou mobile.

O escopo do sistema compreende o gerenciamento completo de cadastros de pacientes, profissionais (médicos), especialidades e, principalmente, o agendamento e cancelamento de teleconsultas, garantindo a integridade dos dados, verificação de conflitos de horário e tratamento de exceções

Descrição das Principais Funcionalidades

A API foi estruturada em uma arquitetura de 5 camadas (Model, Repository, Service, Controller, Exception) e implementa as seguintes funcionalidades de negócio:

- **Agendamento de Consultas (`POST /api/agendamentos`):**
  - Validação de antecedência mínima de 30 minutos.
  - Prevenção de agendamentos duplicados para o mesmo profissional no mesmo horário (checagem de conflito).
- **Cancelamento de Consultas (`PATCH /api/agendamentos/{id}/cancelar`):**
  - Alteração do status do agendamento para "CANCELADO".
- **Reagendamento de Consultas (`PUT /api/agendamentos/{id}/reagendar`):**
  - Alteração da data/hora de um agendamento, com as mesmas validações de antecedência e conflito.
- **Listagem de Recursos:**
  - Exposição de endpoints `GET` para listar todos os Pacientes e todos os Profissionais, permitindo ao front-end popular menus de seleção.
- **Tratamento de Exceções (`@ControllerAdvice`):**
  - Implementação de um handler global que captura erros (ex: "Profissional não encontrado", "Horário ocupado") e retorna um JSON padronizado com o status HTTP correto (ex: 404, 400).

Tabela de Endpoints (API Restful)

A tabela a seguir descreve os principais endpoints implementados para o funcionamento do front-end.

A tabela resume a **API de Agendamentos e Cadastros** com os seguintes endpoints:

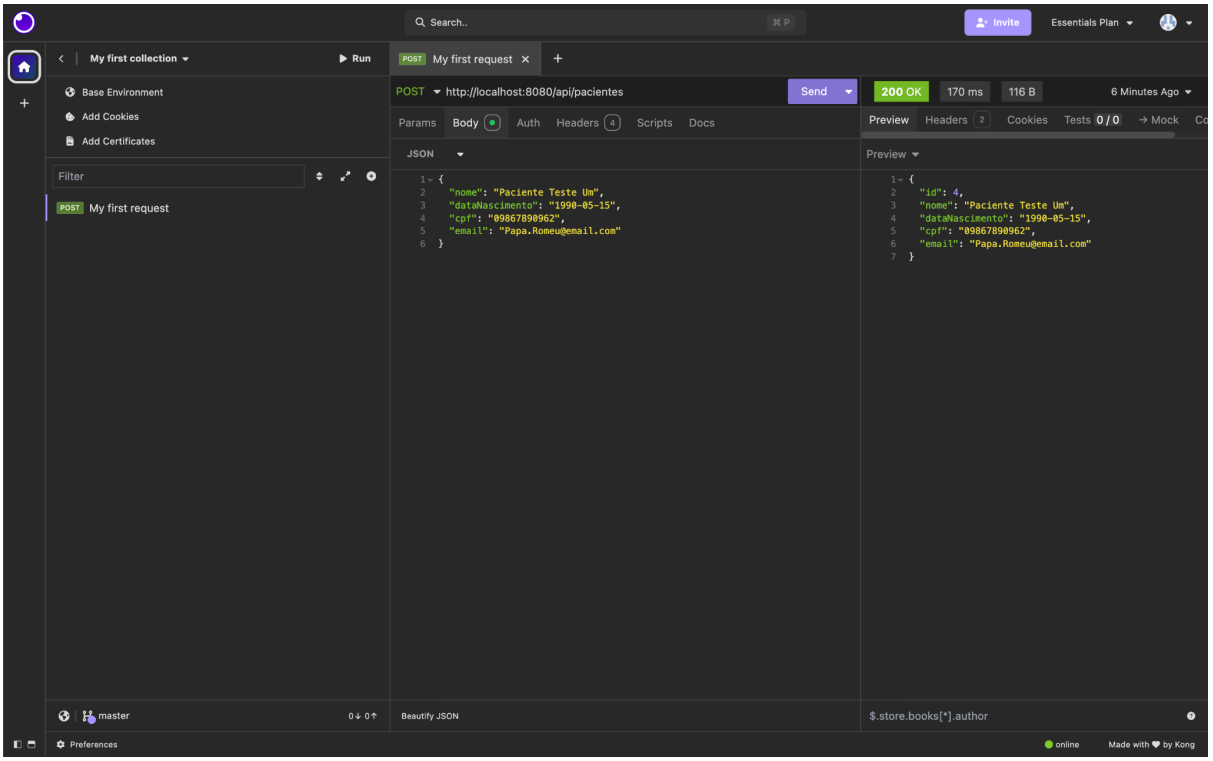
Verbo HTTP	URI	Descrição	Sucesso	Erro Cliente
POST	`/api/agendamentos`	Cria um agendamento.	201	400 (Conf…
GET	`/api/agendamentos/{id}`	Busca agendamento por ID.	200	404
PATCH	`/api/agendamentos/{id}/cancelar`	Cancela agendamento.	200	404
PUT	`/api/agendamentos/{id}/reagendar`	Reagenda data de agendamento.	200	400, 404
GET	`/api/pacientes`	Lista pacientes.	200	-
GET	`/api/pacientes/{id}`	Busca paciente por ID.	200	404
GET	`/api/profissionais`	Lista profissionais.	200	-
GET	`/api/profissionais/{id}`	Busca profissional por ID.	200	404

Protótipo – Prints das Telas Implementadas

A seguir, são apresentados os testes funcionais da API, demonstrando o fluxo completo de requisição e resposta para cada endpoint principal.

Teste 1: Cadastro Paciente

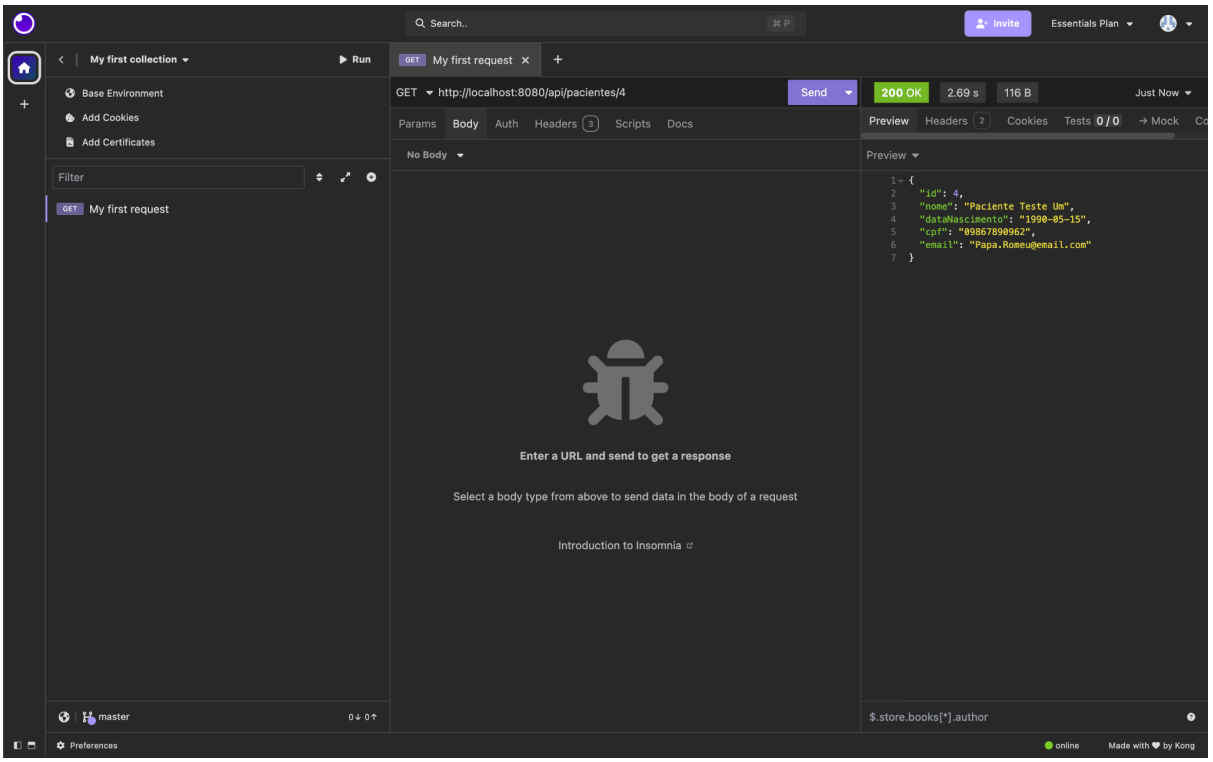
JSON com os dados do "Paciente Teste Um
O servidor recebeu esses dados, cadastrou o paciente com sucesso no banco de dados e respondeu com um status 200 OK.
Confirmação: O mais importante é que a resposta do servidor incluiu o "id": 4. Isso confirma que o paciente foi criado e o banco de dados atribuiu a ele o ID de número 4.



.

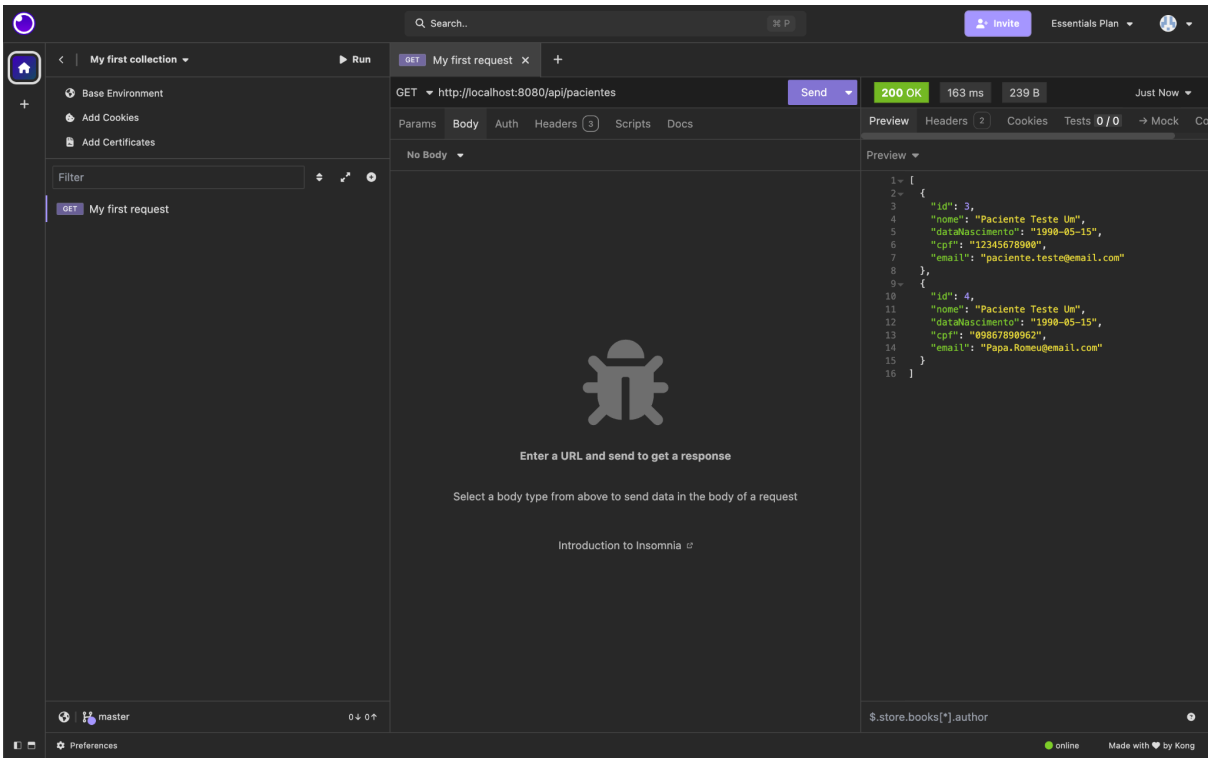
Teste 2: Consulta Paciente por ID

Enviamos uma requisição GET para a URL do paciente, usando o ID 4 que foi criado no teste anterior (.../api/pacientes/4). O objetivo é buscar os dados específicos desse paciente.
Confirmação: O servidor respondeu com status 200 OK e retornou o JSON completo do "Paciente Teste Um", provando que ele está corretamente armazenado no banco.



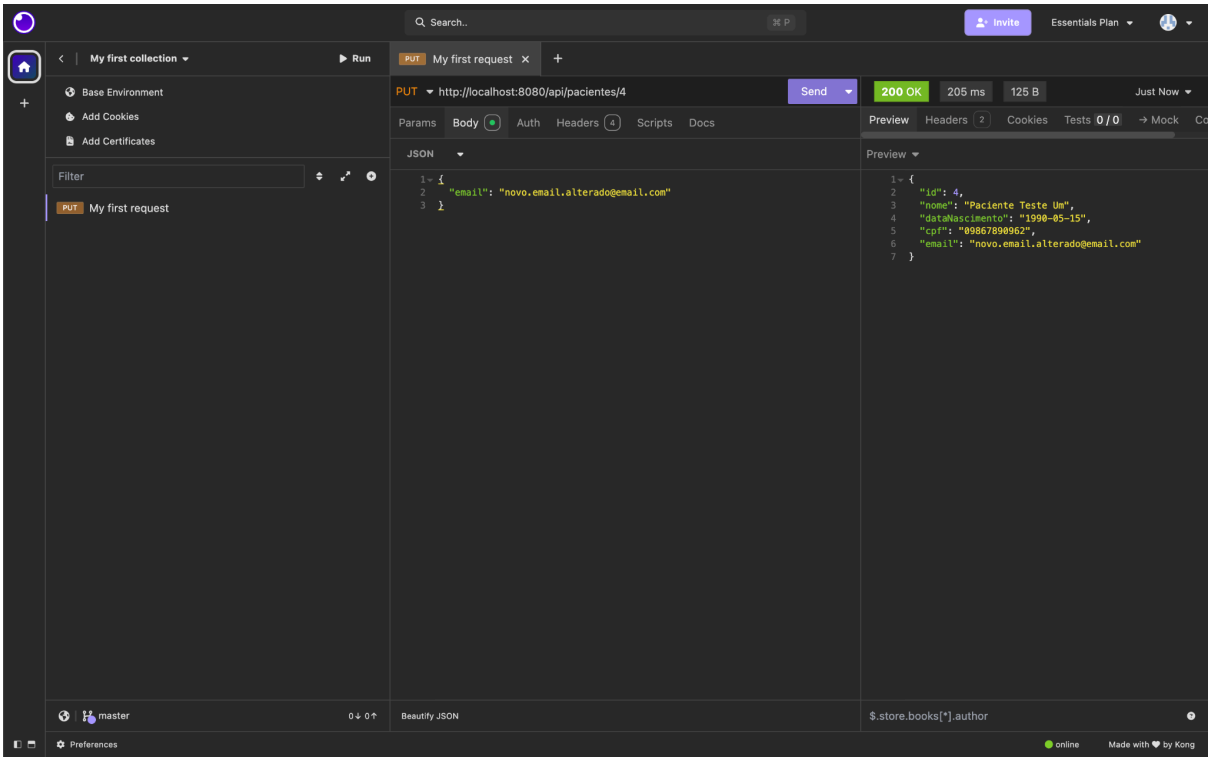
Teste 3: Listar Todos os Pacientes

Enviamos uma requisição GET para a URL base (.../api/pacientes) para solicitar uma lista de todos os pacientes cadastrados no sistema.
Confirmação: O servidor respondeu com status 200 OK e retornou uma lista (array []) contendo todos os pacientes. É possível ver o "Paciente Teste Um" (ID 4) no meio da lista.



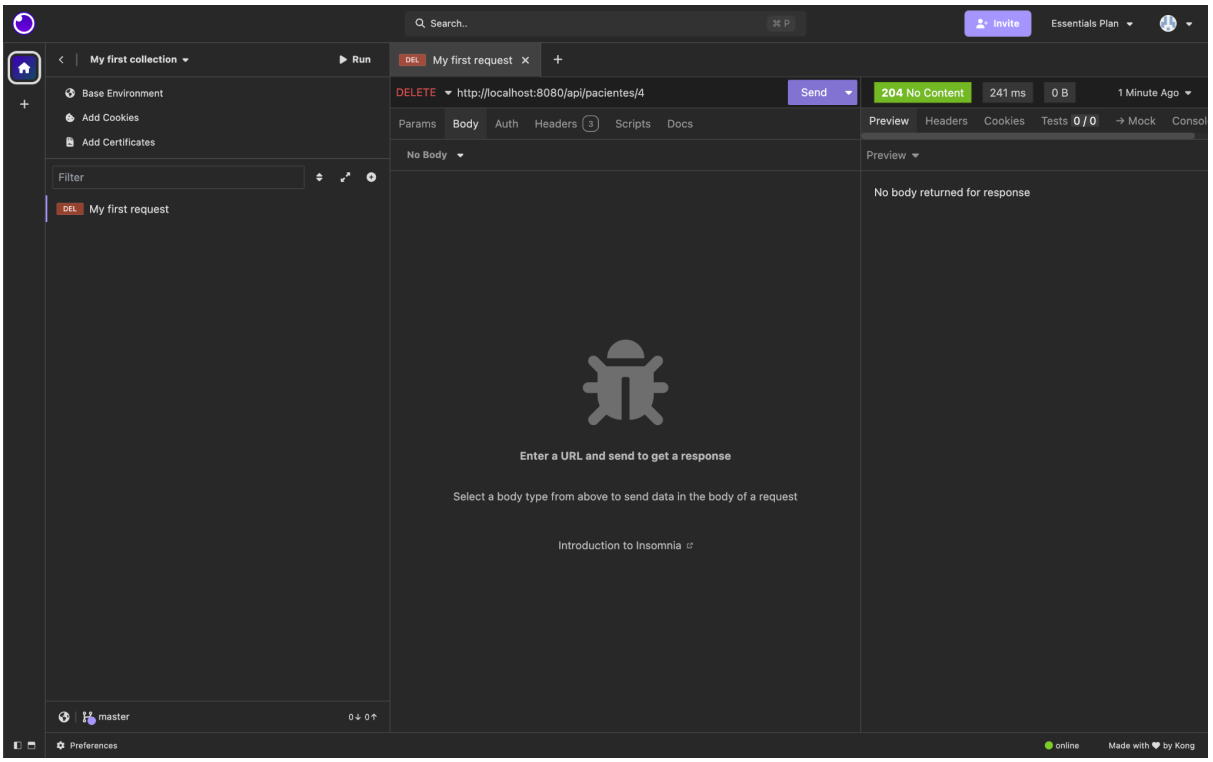
Teste 4: Atualizar Paciente

Enviamos uma requisição PUT para a URL do paciente (.../api/pacientes/4). No Body da requisição, enviamos um JSON com os campos que queríamos alterar (por exemplo, mudando o email).
Confirmação: O servidor processou a atualização e respondeu com 200 OK, retornando o objeto do paciente já com os dados atualizados (ex: com o novo email), confirmando que a alteração foi salva.



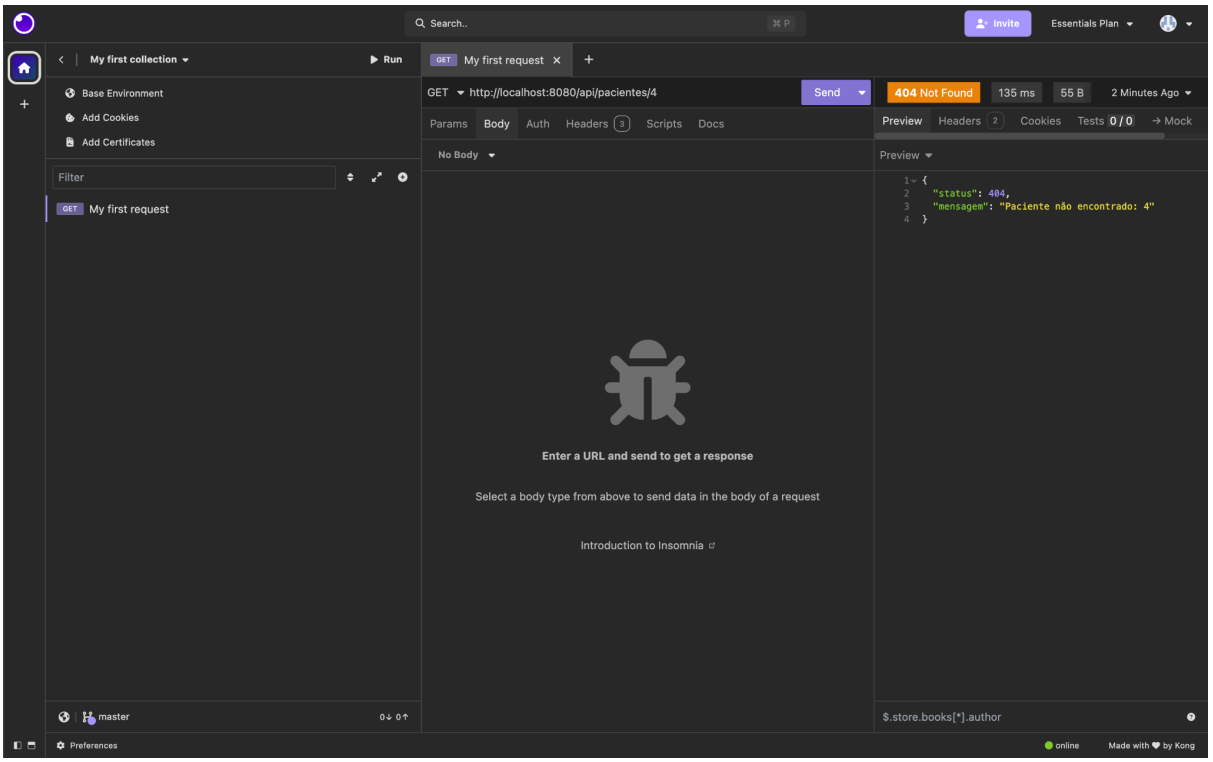
Teste 5: Deletar Paciente

Enviamos uma requisição DELETE para a URL do paciente (.../api/pacientes/4) para removê-lo do banco de dados.
Confirmação: O servidor processou a deleção com sucesso e respondeu com o status 204 No Content. Este status (sem conteúdo) é a resposta padrão de sucesso para uma operação de DELETE.



Teste 6: Validar Deleção

Para provar que o paciente foi mesmo deletado, enviamos novamente uma requisição GET para a URL do paciente que acabamos de apagar (.../api/pacientes/4).
Confirmação: O servidor respondeu com o status 404 Not Found (Não Encontrado). Isso confirma que o paciente com ID 4 não existe mais, validando o sucesso do teste de deleção.



Modelo de Entidade-Relacionamento (MER)

O modelo de banco de dados Oracle foi implementado com base no script As tabelas principais e seus relacionamentos são:

1. T_IMREA_PACIENTE

Armazena os dados cadastrais dos pacientes.

- **ID_PACIENTE (PK**

  (NOME_PACIENTE, CPF, DATA_NASCIMENTO, TIPO_DEFICIENCIA, CONTATO, ENDERECO, CONVENIO)

2. T_IMREA_ESPECIALIDADE

Tabela de domínio para as especialidades médicas.

- **ID_ESPECIALIDADE (PK) (** NOME_ESPECIALIDADE, DESCRICAO)

3. T_IMREA_PROFISSIONAL

Armazena os dados dos médicos e outros profissionais de saúde.

- **ID_PROFISSIONAL (PK)**

  (NOME_PROFISSIONAL, **ID_ESPECIALIDADE (FK)**: Referencia `T_IMREA_ESPECIALIDADE,` REGISTRO_PROF, CONTATO)

4. T_IMREA_ATENDIMENTO

Registra um atendimento (consulta ou sessão) realizado.

- **ID_ATENDIMENTO (PK)**
- **ID_PACIENTE (FK)**: Referencia `T_IMREA_PACIENTE`
- **ID_PROFISSIONAL (FK)**: Referencia `T_IMREA_PROFISSIONAL(` DATA_ATENDIMENTO, TIPO_ATENDIMENTO, OBSERVACOES)

5. T_IMREA_TRATAMENTO

Descreve um plano de tratamento contínuo para um paciente.

- **ID_TRATAMENTO (PK)**
- **ID_PACIENTE (FK)**: Referencia `T_IMREA_PACIENTE`
- **ID_PROFISSIONAL (FK)**: Referencia `T_IMREA_PROFISSIONAL`
- **ID_ESPECIALIDADE (FK)**: Referencia `T_IMREA_ESPECIALIDADE` , DATA_INICIO, DATA_FIM, DESCRICAO

6. T_IMREA_PROCEDIMENTO

Tabela de domínio para os procedimentos oferecidos.

- **ID_PROCEDIMENTO (PK)(**NOME_PROCEDIMENTO, DESCRICAO, CUSTO

7. T_IMREA_TRATAMENTO_PROCEDIMENTO

Tabela associativa (N:N) que vincula quais procedimentos fazem parte de um tratamento.

- **ID_TRATAMENTO (PK, FK)**: Referencia `T_IMREA_TRATAMENTO`
- **ID_PROCEDIMENTO (PK, FK)**: Referencia `T_IMREA_PROCEDIMENTO`

8. T_IMREA_AGENDAMENTO

Tabela principal da API, registra os agendamentos futuros.

- **ID_AGENDAMENTO (PK)**
- **ID_PACIENTE (FK)**: Referencia `T_IMREA_PACIENTE`
- **ID_PROFISSIONAL (FK)**: Referencia `T_IMREA_PROFISSIONAL`
- DATA_AGENDAMENTO, STATUS

9. T_IMREA_PRONTUARIO

Armazena as anotações e prescrições de um paciente, vinculadas a um atendimento.

- **ID_PRONTUARIO (PK)**
- **ID_PACIENTE (FK)**: Referencia `T_IMREA_PACIENTE`
- **ID_ATENDIMENTO (FK)**: Referencia `T_IMREA_ATENDIMENTO`
- ANOTACOES, PRESCRICAO)

Diagrama de Classes Atualizado

A arquitetura do projeto foi migrada para Spring Boot, seguindo os padrões de Domain Driven Design e separação de camadas.A arquitetura de classes segue o fluxo de responsabilidade:

1. **Controller (`AgendamentoController`, `PacienteController`)**
    - Recebe as requisições HTTP (`@RestController`).
    - **Depende de:** `Service`.
2. **Service (`AgendamentoService`, `PacienteService`)**
    - Contém a lógica de negócio (regras, validações) (`@Service`).
    - **Depende de:** `Repository`.
3. **Repository (`AgendamentoRepository`, `PacienteRepository`)**
    - Interface que estende `JpaRepository` para acesso aos dados (`@Repository`).
    - **Depende de:** `Model`.
4. **Model (`Agendamento`, `Paciente`)**
    - Classes que mapeiam as tabelas do banco (`@Entity`).
5. **Exception (`GlobalExceptionHandler`)**
    - Classe que captura exceções de todas as camadas (`@ControllerAdvice`).

Modelo de Entidade-Relacionamento (MER)

O modelo de banco de dados Oracle foi implementado com base no script `IMREA_CHALLENGE_ENTREGA4.sql`. As tabelas principais e seus relacionamentos são