FIAP

Vitor Dias (RM: 565422)
Iago Liziero (RM: 564063)
Enrico Delesporte (RM: 565760)

HC Conecta
Hospital Das Clínicas Challenge

Objetivo e Escopo do Projeto

O projeto **HC Conecta API** tem como objetivo principal fornecer uma interface de programação de aplicações (API) robusta e eficiente para a gestão de informações de **Pacientes** e **Consultas** em um contexto de saúde, como um Hospital ou Clínica (HC). O escopo da API abrange as operações fundamentais de **CRUD (Create, Read, Update, Delete)** para ambas as entidades, permitindo que sistemas front-end ou outros serviços consumam e manipulem esses dados de forma padronizada e segura.

O projeto foi desenvolvido utilizando a tecnologia **Quarkus**, um *framework* Java otimizado para a criação de microsserviços e aplicações *cloud-native*, o que garante alta performance e baixo consumo de memória. A arquitetura da solução segue o padrão **RESTful**, expondo recursos via HTTP e utilizando **JSON** para a troca de dados.

Descrição Objetiva e Concisa da Solução Proposta

A solução é uma **API RESTful** desenvolvida em **Java** com o *framework* **Quarkus**, utilizando o padrão **JAX-RS** para a definição dos *endpoints*. Ela atua como a camada de serviço central para a gestão de dados de pacientes e agendamentos de consultas. A arquitetura é modular, separando a lógica de negócio (`BO - Business Object`) da camada de acesso a dados (`DAO - Data Access Object`), o que facilita a manutenção e a escalabilidade.

A API é projetada para ser o *backbone* de um sistema de gestão de saúde, oferecendo as seguintes capacidades:

1. **Gestão de Pacientes:** Registro, consulta, atualização e exclusão de informações cadastrais de pacientes.
2. **Gestão de Consultas:** Agendamento, consulta, atualização e cancelamento de consultas, incluindo a funcionalidade de listar todas as consultas de um paciente específico.

Breve Descrição das Principais Funcionalidades da Solução

A API HC Conecta oferece um conjunto de funcionalidades essenciais para a gestão de pacientes e consultas, organizadas em dois recursos principais: `/pacientes` e `/consultas`.

Recurso	Funcionalidade	Método HTTP	Endpoint	Descrição
Pacientes	Criar Paciente	`POST`	`/pacientes`	Registra um novo paciente no sistema.

Recurso	Funcionalidade	Método HTTP	Endpoint	Descrição
	Buscar Paciente por ID	`GET`	`/pacientes/{cd_paciente}`	Retorna os dados de um paciente específico.
	Atualizar Paciente	`PUT`	`/pacientes/atualiza-paciente/{cd_paciente}`	Modifica as informações cadastrais de um paciente.
	Excluir Paciente	`DELETE`	`/pacientes/{cd_paciente}`	Remove o registro de um paciente.
Consultas	Criar Consulta	`POST`	`/consultas`	Agenda uma nova consulta.
	Buscar Consulta por ID	`GET`	`/consultas/{cd_consulta}`	Retorna os detalhes de uma consulta específica.
	Buscar Consultas por Paciente	`GET`	`/consultas/consultas-paciente/{cd_paciente}`	Lista todas as consultas agendadas para um paciente.
	Atualizar Consulta (Completo)	`PUT`	`/consultas/{cd_consulta}`	Atualiza todos os dados de uma consulta.
	Atualizar Status da Consulta	`PUT`	`/consultas/atualiza-consulta/{cd_consulta}`	Atualiza apenas o status da consulta (e.g., Agendada, Cancelada).
	Excluir Consulta	`DELETE`	`/consultas/{cd_consulta}`	Cancela e remove o registro de uma consulta.

Destaque das Funcionalidades Implementadas

As funcionalidades implementadas demonstram uma solução completa de gestão de dados, com destaque para:

1. **CRUD Completo para Pacientes:** A API oferece todas as operações básicas de persistência para a entidade `Paciente`, garantindo a integridade e a gestão do cadastro de usuários.
2. **Gestão Detalhada de Consultas:** Além do CRUD básico, a API implementa funcionalidades específicas para o fluxo de agendamento:
    - **Listagem por Paciente:** O *endpoint* `/consultas/consultas-paciente/{cd_paciente}` é crucial para que o paciente ou o sistema possa visualizar seu histórico e agendamentos futuros.
    - **Atualização de Status Dedicada:** O *endpoint* `/consultas/atualiza-consulta/{cd_consulta}` sugere uma lógica de negócio separada para a mudança de status (e.g., de "Agendada" para "Realizada" ou "Cancelada"), o que é uma prática comum em sistemas transacionais.
3. **Tecnologia Moderna e Otimizada:** O uso do **Quarkus** indica uma preocupação com a performance e a eficiência de recursos, características essenciais para microsserviços em ambientes de nuvem.
4. **Estrutura de Código Limpa:** A separação em `Resource` (camada de apresentação/API), `BO` (lógica de negócio) e `DAO` (acesso a dados) adere a princípios de design de software que promovem a manutenibilidade e a testabilidade do código.
5. **Tratamento de CORS:** A presença do `CorsFilter.java` indica que a API está preparada para ser consumida por aplicações *front-end* hospedadas em domínios diferentes, resolvendo um problema comum de integração.
6. **Validação de Dados:** A utilização da anotação `@Valid` nos métodos `POST` e `PUT` sugere que a API realiza a validação dos dados de entrada (TOs - Transfer Objects) antes de processar a lógica de negócio, garantindo a qualidade dos dados persistidos.

ENDPOINTS DA API RESTFUL

Recurso: Pacientes

---

1. URI: /pacientes
   Verbo HTTP: POST
   Descrição: Cadastra um novo paciente.
   Status de Resposta Esperado: 201 Created (Sucesso), 400 Bad Request (Erro de validação ou BO)

2. URI: /pacientes/{cd_paciente}
Verbo HTTP: GET
Descrição: Busca um paciente pelo ID.
Status de Resposta Esperado: 200 OK (Encontrado), 404 Not Found (Não encontrado)

3. URI: /pacientes/atualiza-paciente/{cd_paciente}
Verbo HTTP: PUT
Descrição: Atualiza os dados de um paciente pelo ID.
Status de Resposta Esperado: 201 Created (Sucesso), 400 Bad Request (Erro de validação ou BO)

4. URI: /pacientes/{cd_paciente}
Verbo HTTP: DELETE
Descrição: Remove um paciente pelo ID.
Status de Resposta Esperado: 204 No Content (Sucesso), 404 Not Found (Não encontrado)

5. URI: /pacientes/login
Verbo HTTP: GET
Descrição: Realiza o login do paciente (autenticação por CPF e Senha).
Status de Resposta Esperado: 200 OK (Sucesso), 404 Not Found (Credenciais inválidas)

Recurso: Consultas
---
1. URI: /consultas
Verbo HTTP: POST
Descrição: Agenda uma nova consulta.
Status de Resposta Esperado: 201 Created (Sucesso), 400 Bad Request (Erro de validação ou BO)

2. URI: /consultas/consultas-paciente/{cd_paciente}
Verbo HTTP: GET
Descrição: Lista todas as consultas de um paciente específico.
Status de Resposta Esperado: 200 OK (Sucesso), 404 Not Found (Paciente sem consultas ou não encontrado)

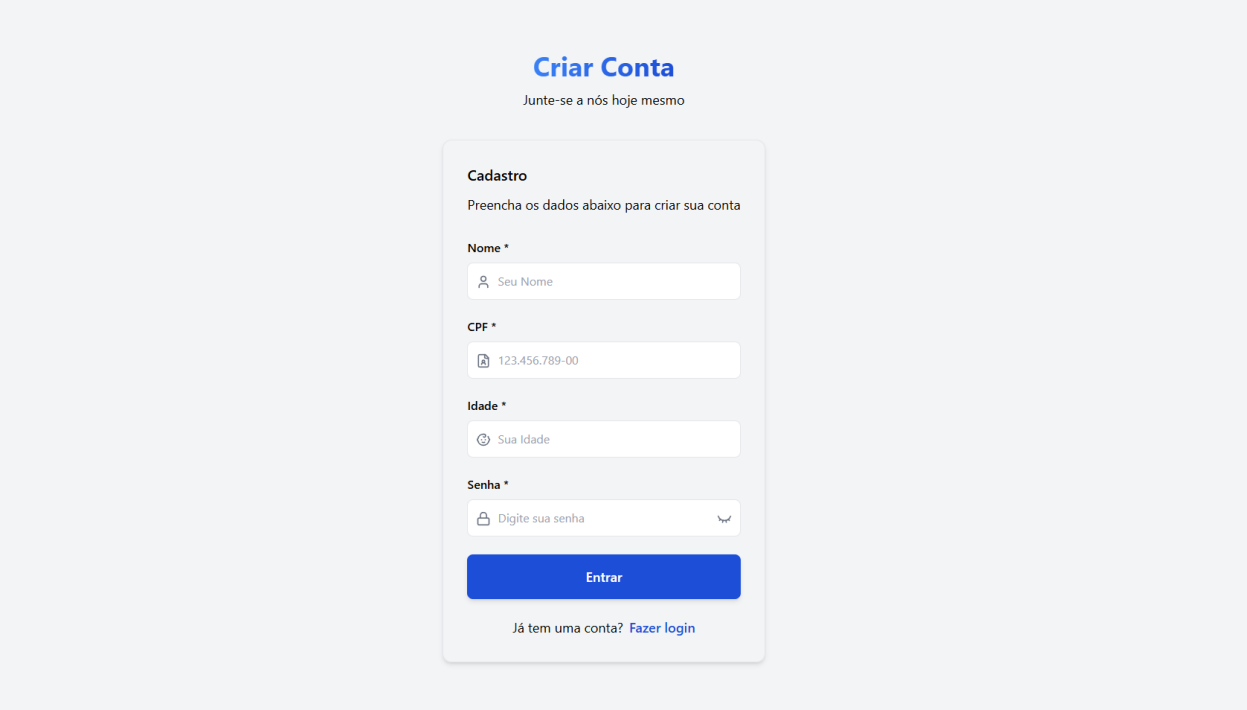
3. URI: /consultas/{cd_consulta}
   Verbo HTTP: GET
   Descrição: Busca uma consulta pelo ID.
   Status de Resposta Esperado: 200 OK (Encontrado), 404 Not Found (Não encontrado)

4. URI: /consultas/{cd_consulta}
   Verbo HTTP: PUT
   Descrição: Atualiza todos os dados de uma consulta pelo ID.
   Status de Resposta Esperado: 201 Created (Sucesso), 400 Bad Request (Erro de validação ou BO)

5. URI: /consultas/atualiza-consulta/{cd_consulta}
   Verbo HTTP: PUT
   Descrição: Atualiza o status de uma consulta pelo ID.
   Status de Resposta Esperado: 201 Created (Sucesso), 400 Bad Request (Erro de validação ou BO)

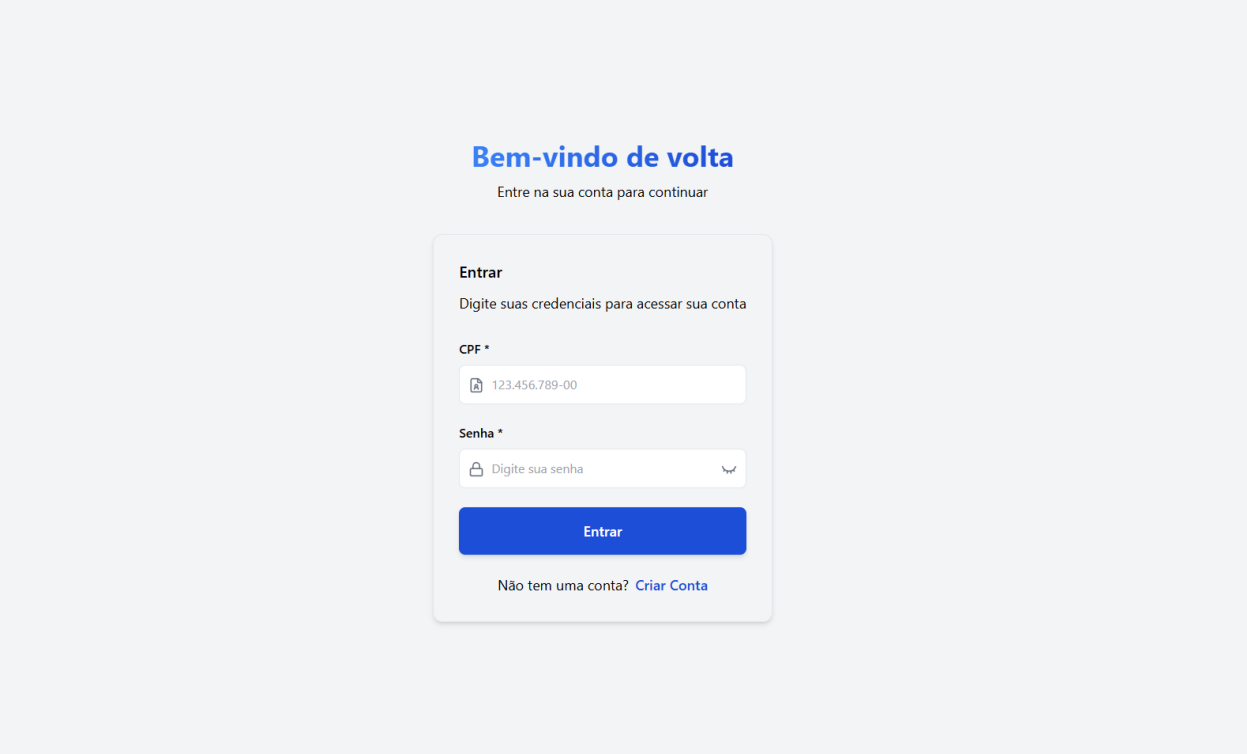
6. URI: /consultas/{cd_consulta}
   Verbo HTTP: DELETE
   Descrição: Cancela/Remove uma consulta pelo ID.
   Status de Resposta Esperado: 204 No Content (Sucesso), 404 Not Found (Não encontrado)

Implementação com o Front-End



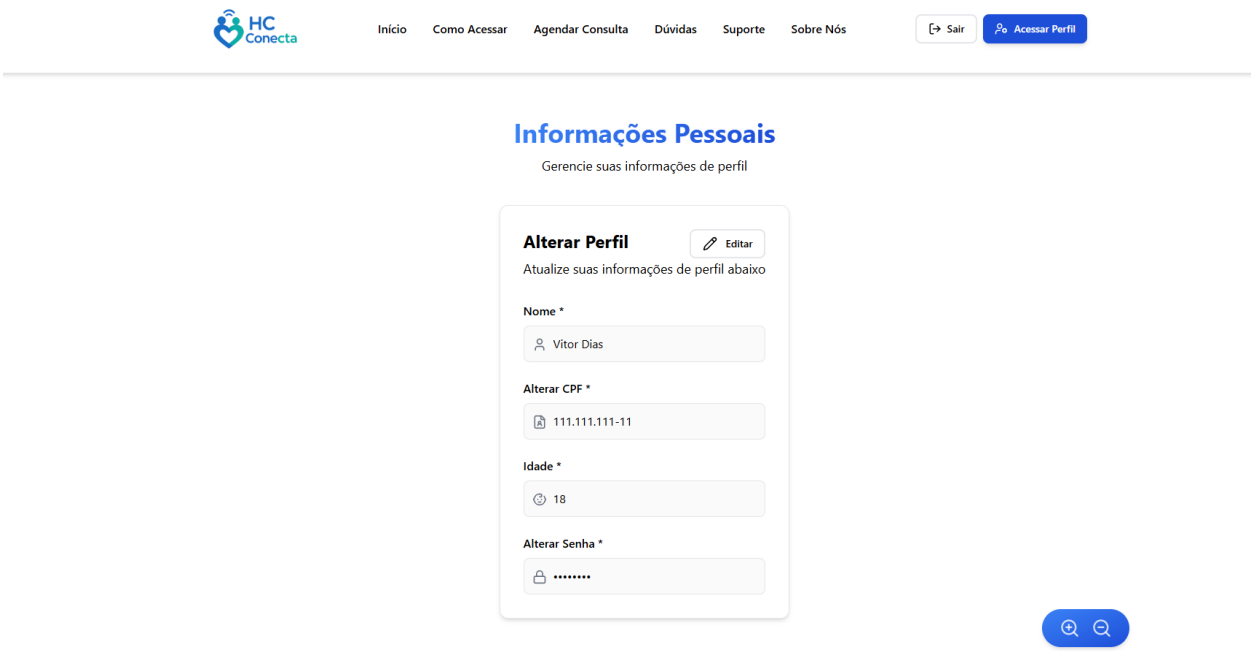
Cadastro

O usuário preenche nome, CPF, idade e senha.
O front envia as informações para a API, que cria uma nova conta no banco de dados após validação.
Ao concluir, o utilizador é redirecionado para a tela de login.



Login

O utilizador insere o CPF e a senha.
O sistema envia os dados ao back-end.
Com o login bem-sucedido, o utilizador é redirecionado para a área de perfil.



Perfil

Após o login, o sistema mostra os dados pessoais do utilizador.
Ele pode editar nome, idade e senha, e as alterações são enviadas ao back-end, que atualiza o registo no banco.

Modelo Entidade Relacionamento (MER)

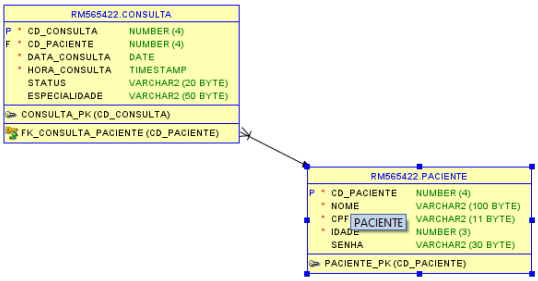
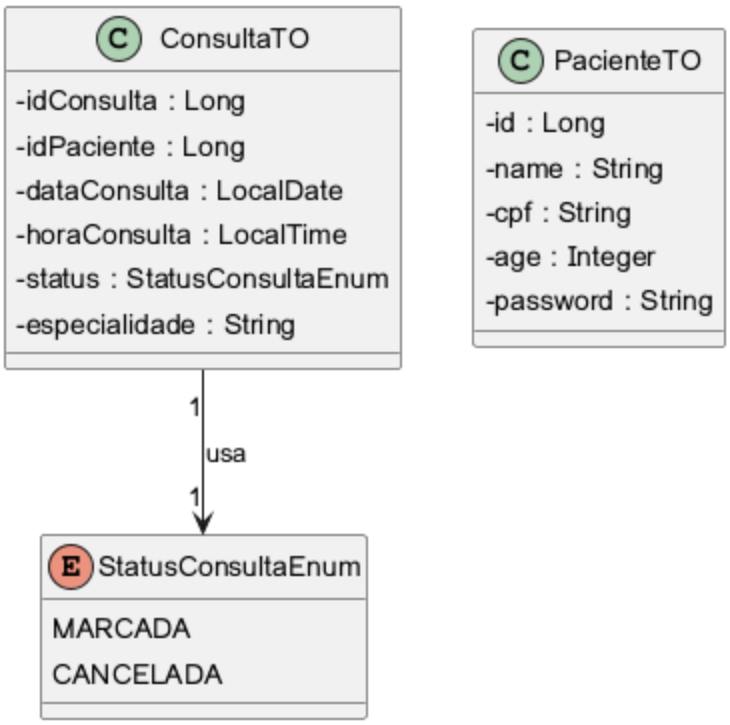


Diagrama de Classes


ConsultaTO
-idConsulta : Long
-idPaciente : Long
-dataConsulta : LocalDate
-horaConsulta : LocalTime
-status : StatusConsultaEnum
-especialidade : String
PacienteTO
-id : Long
-name : String
-cpf : String
-age : Integer
-password : String
1
usa
1
StatusConsultaEnum
MARCADA
CANCELADA