

Integrantes

- Olavo Porto Neves - RM563558
- Pedro Henrique Dias França - RM561940
- Luiz Gustavo Gonçalves - RM564495

Hera Corporation

Hera Platform (Sistema de Automação e Engajamento para
Teleconsultas)

Sumário

Objetivo e escopo do projeto.....	2
Descrição das Funcionalidades da Solução.....	3
Tabela de Endpoints.....	5
Protótipo.....	7
Modelo de Entidade-Relacionamento.....	11
Diagrama de Classes.....	12

Objetivo e escopo do projeto

A Hera Corporation desenvolveu uma solução inovadora de automação voltada para o controle da jornada de teleconsulta dos pacientes do IMREA.

O objetivo principal do sistema é otimizar a experiência do paciente e reduzir a carga operacional dos colaboradores do hospital, garantindo que todo o processo seja automatizado, seguro e eficiente.

Após a adesão ao plano oferecido pela Hera Corporation, o hospital passa a ter acesso a uma aplicação própria destinada aos funcionários responsáveis pela gestão dos dados dos pacientes. O fluxo de funcionamento pode ser descrito da seguinte forma:

1. O colaborador do hospital insere os dados do paciente em um formulário;
2. Essas informações são enviadas ao backend, que valida os dados recebidos e os armazena em um banco de dados;
3. Se os dados forem validados com sucesso, inicia-se o processo de automação inteligente, que monitora continuamente o banco em busca de atualizações relevantes.

Dentro desse escopo, a automação atua de forma estratégica:

- Envio de vídeos tutoriais para orientar os pacientes antes da teleconsulta;
- Envio de lembretes automáticos em intervalos definidos, reforçando prazos e horários importantes;
- Atualização de status da jornada do paciente, mantendo transparência para todos os envolvidos;
- Integração de gamificação, incentivando o engajamento e adesão ao tratamento;

Toda essa camada de automação é orquestrada através do n8n, integrado com os principais canais de comunicação utilizados pelos pacientes: WhatsApp, E-mail e SMS.

Em resumo, o sistema proporciona uma gestão unificada e automatizada da jornada de teleconsulta, trazendo benefícios tanto para os pacientes quanto para os profissionais de saúde, com foco em eficiência, engajamento e melhoria contínua da experiência de atendimento.

Descrição das Funcionalidades da Solução

O sistema desenvolvido pela Hera Corporation foi projetado para oferecer uma experiência completa de automação na jornada de teleconsulta. As funcionalidades podem ser separadas em dois níveis principais: funcionalidades dos administradores e funcionalidades da aplicação em si.

Funcionalidades dos Administradores

O Portal dos Administradores foi construído com o foco de oferecer a manutenção e gerenciamento eficiente das informações médicas, facilitando seu trabalho diário. Entre suas principais funcionalidades estão:

- **CRUD de Paciente:** permite o cadastro, consulta, atualização e exclusão de pacientes, garantindo que todos os dados pessoais e clínicos necessários estejam organizados.
- **CRUD de Médico:** possibilita o gerenciamento completo dos dados de profissionais de saúde, incluindo especialidade, disponibilidade e informações cadastrais.
- **CRUD de Consulta:** oferece controle total sobre os agendamentos, permitindo criar, alterar, consultar e remover consultas médicas de acordo com a necessidade.
- **CRUD de Usuário:** além de facilitar o login no portal, garantindo o acesso às ferramentas apenas se for autorizado, também permite criar, alterar, excluir e consultar os usuários de forma simples.

Essas funcionalidades garantem que os funcionários responsáveis (Administradores) tenham um ambiente seguro e confiável para armazenar e manipular os dados que servirão de base para as automações da jornada do paciente.

Funcionalidades da Aplicação

Além do portal dos administradores, a aplicação implementa processos automatizados voltados para a interação direta com o paciente, utilizando integrações com WhatsApp, e-mail e SMS. Entre elas destacam-se:

- **Envio automático do vídeo inicial:** logo no início da jornada, o paciente recebe um vídeo tutorial explicativo sobre o funcionamento da teleconsulta.
- **Gestão do status da jornada:** o sistema atualiza e informa automaticamente os diferentes estágios do paciente (pré-consulta, consulta confirmada, consulta realizada etc.).
- **Envio de lembretes programados:** mensagens automáticas são disparadas periodicamente para lembrar o paciente de compromissos importantes.

- Gamificação do processo: o sistema recompensa pacientes engajados, por exemplo, enviando mensagens de reconhecimento quando o paciente comparece às consultas.

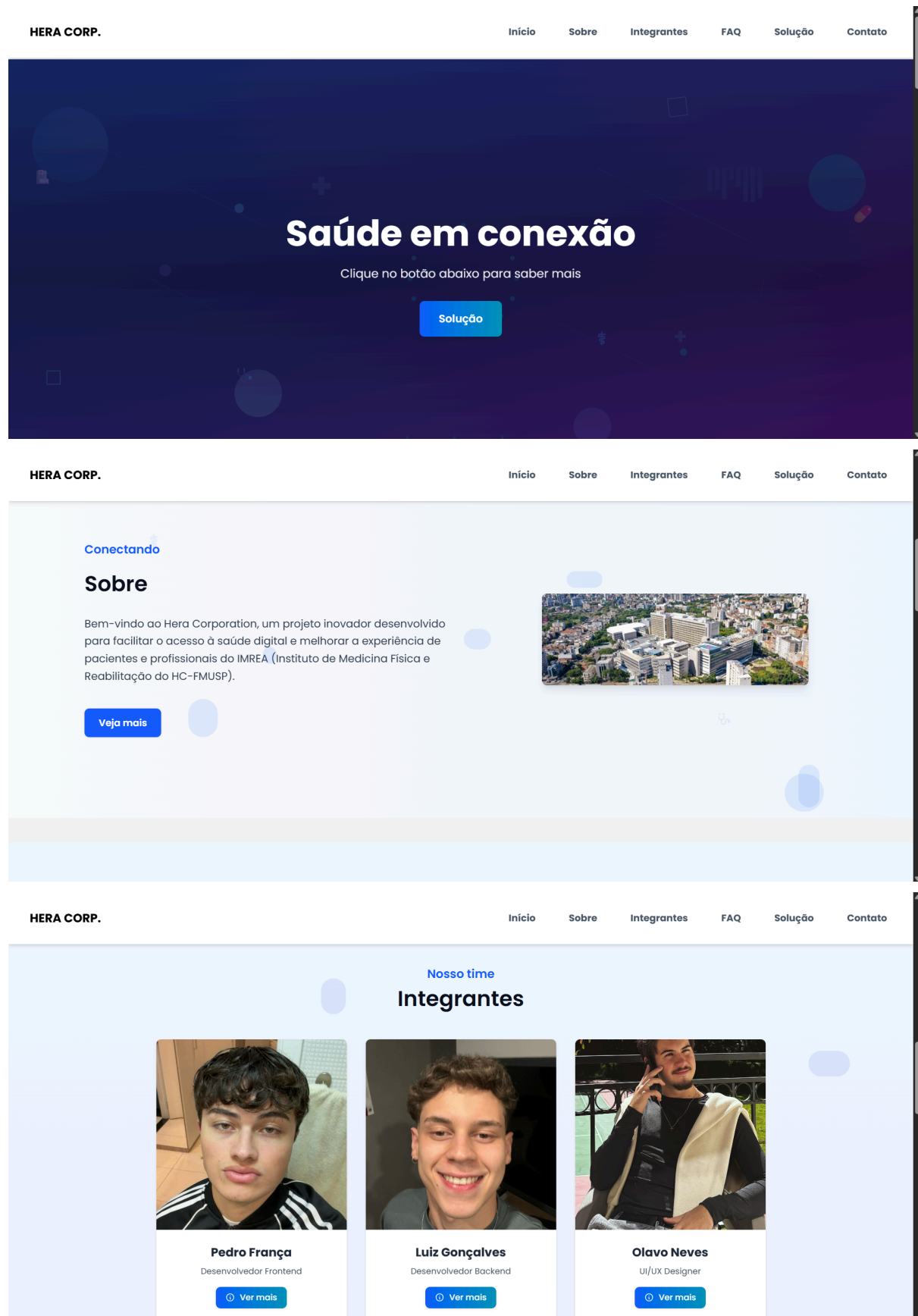
Essas funcionalidades proporcionam maior engajamento, acompanhamento contínuo e melhor experiência para o paciente, ao mesmo tempo que reduzem a necessidade de intervenção manual por parte da equipe do hospital.

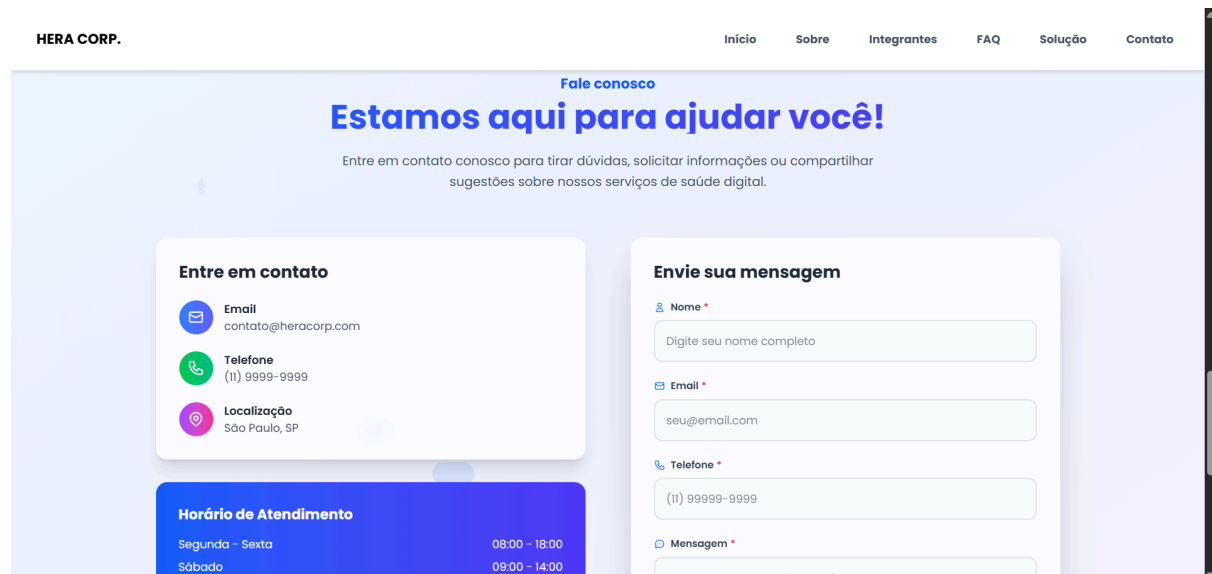
Tabela de Endpoints

URIs / Endpoints	Método HTTP	Descrição	Códigos de Resposta
https://hera-api.onrender.com/hera-api/usuarios	POST	Cria um novo usuário.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/usuarios/{id}	PUT	Atualiza os dados de um usuário existente.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/usuarios/{id}	DELETE	Exclui um usuário existente.	204 (Sem conteúdo), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/usuarios/{id}	GET	Retorna um usuário específico pelo ID.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/usuarios	GET	Retorna a lista de todos os usuários cadastrados.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/login	POST	Verifica se usuário existe no banco.	200 (OK), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/pacientes	POST	Cria um novo paciente.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/pacientes/{id}	PUT	Atualiza os dados de um paciente existente.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/pacientes/{id}	DELETE	Exclui um paciente existente.	204 (Sem conteúdo), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/pacientes/{id}	GET	Retorna um paciente específico pelo ID.	200 (OK), 404 (Não encontrado), 500 (Erro interno)

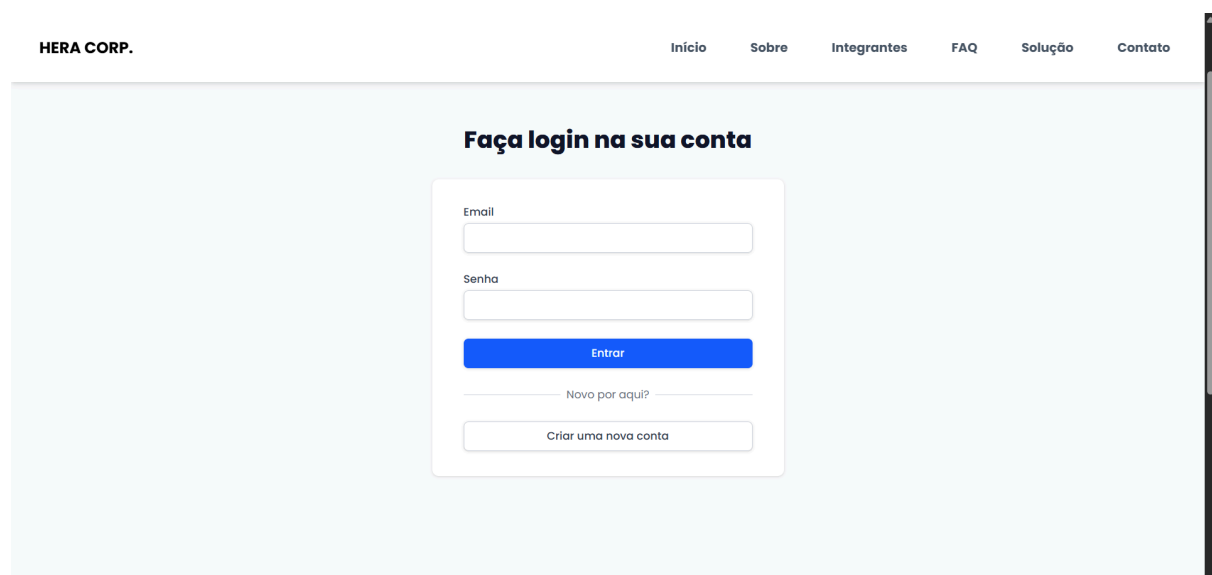
https://hera-api.onrender.com/hera-api/pacientes	GET	Retorna a lista de todos os pacientes cadastrados.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/medicos	POST	Cria um novo médico.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/medicos/{id}	PUT	Atualiza os dados de um médico existente.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/medicos/{id}	DELETE	Exclui um médico existente.	204 (Sem conteúdo), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/medicos/{id}	GET	Retorna um médico específico pelo ID.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/medicos	GET	Retorna a lista de todos os médicos cadastrados.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/consultas	POST	Cria uma nova consulta.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/consultas/{id}	PUT	Atualiza os dados de uma consulta existente.	201 (Criado), 400 (Requisição inválida), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/consultas/{id}	DELETE	Exclui uma consulta existente.	204 (Sem conteúdo), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/consultas/{id}	GET	Retorna uma consulta específica pelo ID.	200 (OK), 404 (Não encontrado), 500 (Erro interno)
https://hera-api.onrender.com/hera-api/consultas	GET	Retorna a lista de todas as consultas cadastradas.	200 (OK), 404 (Não encontrado), 500 (Erro interno)

Protótipo do Sistema





Essas são as páginas iniciais, ou seja, apenas a parte de apresentação sobre a Hera Corporation em si, além de pequenas informações sobre a solução. A solução real com as novas páginas vem abaixo (que pode ser acessada pelo botão Solução):



Página para efetuar o login. Aqui a gente vai verificar não só se o usuário existe, como as suas regras, ou seja, aqui que conseguimos saber se ele vai acessar o portal como administrador ou como usuário padrão. Necessário, pois se ele acessar como administrador, as funcionalidades que aparecem para ele são diferentes do usuário padrão (seria a ideia final).

HERA CORP.

Início

Sobre

Integrantes

FAQ

Solução

Contato

Crie uma nova conta

Nome completo

Email

Senha

Cadastrar

Já tem uma conta?

Fazer login

Página para cadastro do usuário, caso ele ainda não tenha efetuado esse cadastro. Necessário para funcionamento correto do login.

HERA CORP.

Início

Sobre

Integrantes

FAQ

Solução

Contato

Bom dia, Pedro!

Você tem consultas para marcar hoje. Vamos lá?

Consulta para o Marcos / Fisioterapia

Progresso da Jornada

2/4 concluídas

☒ Agendar consulta para o Marcos Aurélio

☒ Mandar guia para Fernanda Campos

☐ Mandar mensagem para Felipe Trindade

☐ Reagendar consulta do Lucas Alves

Conquistas

Conquistado

21/out/2024

Explorador Fiduciário

Conquista desbloqueada

Bloqueado

Paciente

Operação

Cadastrar

Confirmar

Médico

Operação

Cadastrar

Confirmar

Consulta

Operação

Cadastrar

Confirmar

HERA CORP.

Início

Sobre

Integrantes

FAQ

Solução

Contato

Cadastro de Paciente

Preencha os dados do paciente

1

Dados Pessoais

Nome Completo *

Digite o nome completo

CPF *

000.000.000-00

Data de Nascimento *

dd/mm/aaaa

Sexo *

Selecione

2

Contato

DDD *

00

Número *

00000-0000

Tipo de Telefone *

Celular

E-mail *

email@exemplo.com

9

Caso usuário logado seja criado como Administrador(regra de negócio), ele tem esse portal liberado para poder trabalhar. Então ele terá suas tasks diárias, e as opções de trabalhar com a Gestão (CRUDs), de Paciente, Médico e Consulta.

Facilitando tanto o trabalho dele, quanto o nosso, que já pegamos os dados enviados, direto no nosso banco, e depois jogamos para nosso fluxo de automação trabalhar.

Modelo de Entidade-Relacionamento

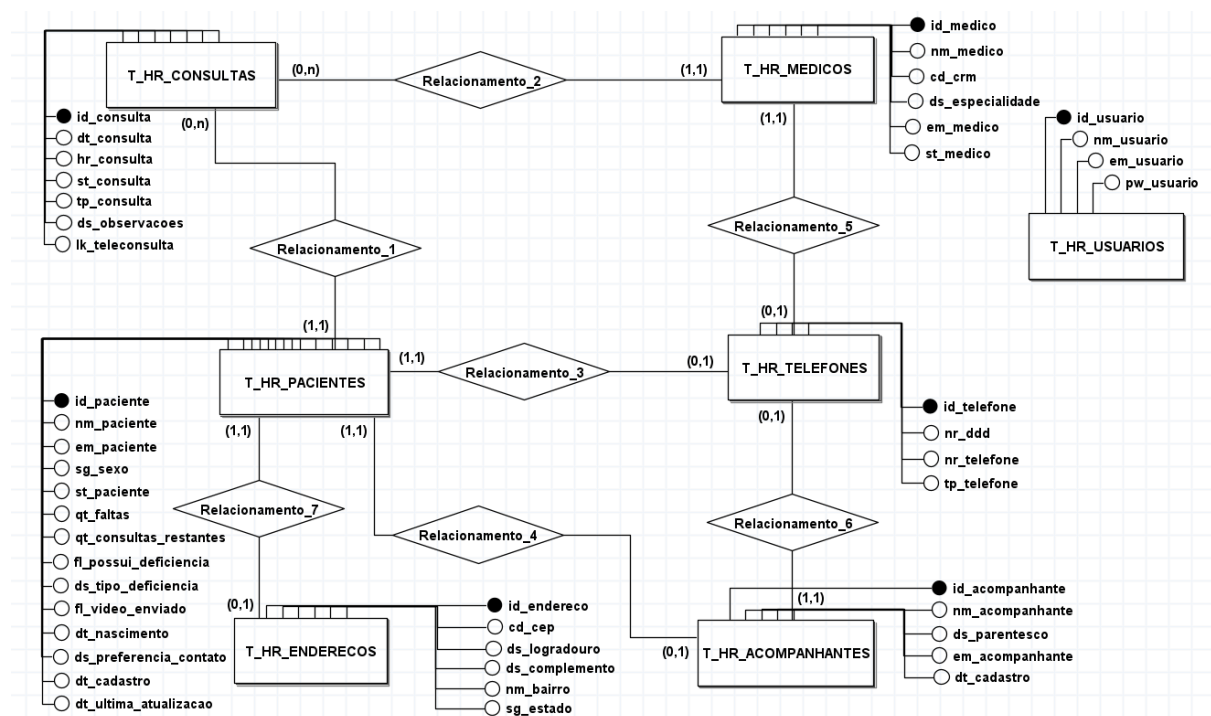


Diagrama de Classes

CorsFilter
+ filter(ContainerRequestContext request, ContainerResponseContext response): void

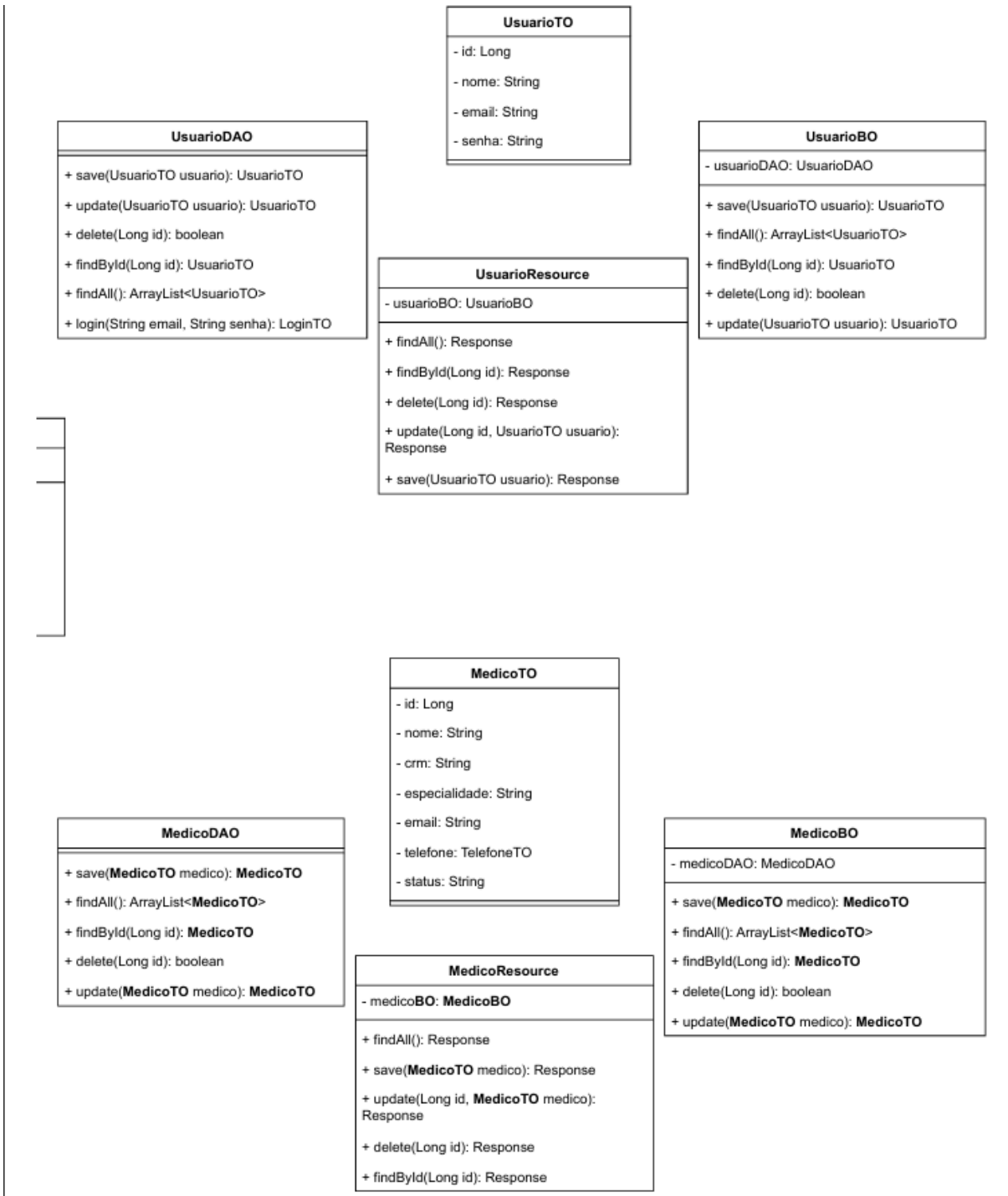
ConnectionFactory
- connection: Connection
+ getConnection(): Connection
+ closeConnection(): void

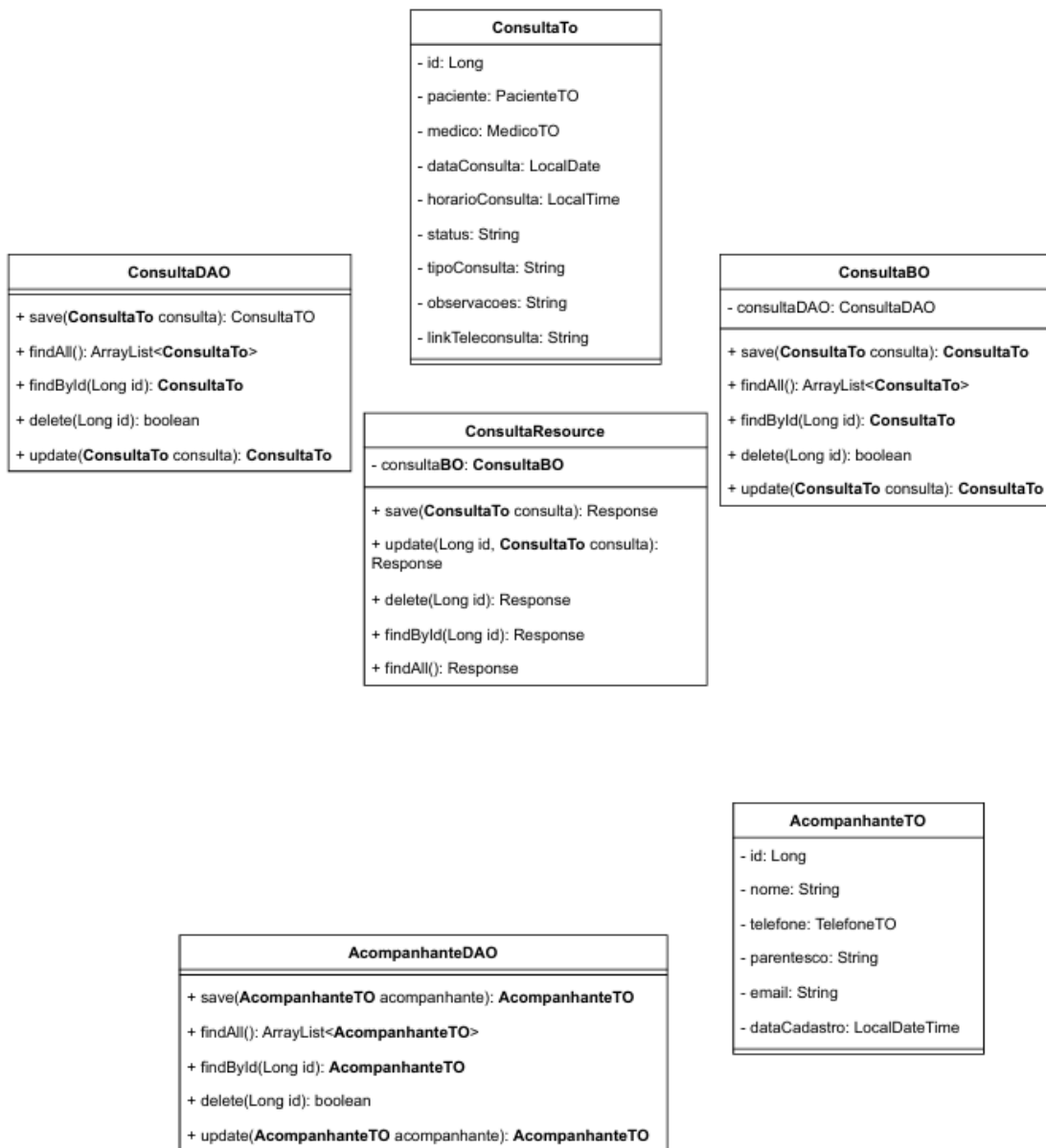
PacienteDAO
+ save(PacienteTO paciente): PacienteTO
+ findAll(): ArrayList<PacienteTO>
+ findById(Long id): PacienteTO
+ delete(Long id): boolean
+ update(PacienteTO paciente): PacienteTO

PacienteTO
- id: Long
- nome: String
- email: String
- sexo: String
- telefone: TelefoneTO
- status: String
- consultasRestantes: int
- faltas: int
- possuiDeficiencia: boolean
- tipoDeficiencia: String
- videoEnviado: boolean
- dataNascimento: LocalDate
- endereco: EnderecoTO
- preferenciaContato: String
- dataCadastro: LocalDateTime
- ultimaAtualizacao: LocalDateTime
- acompanhante: AcompanhanteTO

PacienteBO
- pacienteDAO: PacienteDAO
+ save(PacienteTO paciente): PacienteTO
+ findAll(): ArrayList<PacienteTO>
+ findById(Long id): PacienteTO
+ delete(Long id): boolean
+ update(PacienteTO paciente): PacienteTO

PacienteResource
- pacienteBO: PacienteBO
+ findAll(): Response
+ findById(Long id): Response
+ save(PacienteTO paciente): Response
+ update(Long id, PacienteTO paciente): Response
+ delete(Long id): Response





TelefoneDAO
+ save(TelefoneTO telefone): TelefoneTO + findAll(): ArrayList< TelefoneTO > + findById(Long id): TelefoneTO + delete(Long id): boolean + update(TelefoneTO telefone): TelefoneTO

TelefoneTO
- id: Long - ddd: String - numero: String - tipoDeTelefone: String

EnderecoDAO
+ save(EnderecoTO endereco): EnderecoTO + findAll(): ArrayList< EnderecoTO > + findById(Long id): EnderecoTO + delete(Long id): boolean + update(EnderecoTO endereco): EnderecoTO

EnderecoTO
- id: Long - cep: String - logradouro: String - complemento: String - bairro: String - estado: String

LoginTO
- id: Long - email: String - senha: String

LoginBO
- usuarioDAO: UsuarioDAO
+ login(String email, String senha): LoginTO

LoginResource
- loginBO: LoginBO
+ login(LoginTO login): Response

