Portal para todos

Tríade Acessível

Kauã Almeida Silveira - RM 552618

Matheus Alexandre Benites Scotti - RM 553073

Mariana Spinola - RM 553476

**Sumário**

[Objetivo e escopo do projeto 3](#_Toc167129750)

[Principais funcionalidades 4](#_Toc167129751)

[Tabela dos endpoints (API Restful) 5](#_Toc167129752)

[Corpo das Requisições 6](#_Toc167129753)

[Protótipos das telas 9](#_Toc167129754)

[Modelagem Banco de Dados 12](#_Toc167129755)

[Estrutura do projeto 13](#_Toc167129756)

[Procedimento para executar 14](#_Toc167129757)

[Link do repositório com do código Java do GitHub. 14](#_Toc167129758)

# Objetivo e escopo do projeto

Em relação à acessibilidade, nossa abordagem se concentra na criação de uma estrutura semântica durante o desenvolvimento do código. Estamos adotando uma linguagem mais direta e clara para usuários que podem ter dificuldades com expressões complexas ou gírias. Além disso, estamos implementando uma dinâmica de cores e uma navegação mais amigável para melhorar a experiência do usuário (UX). Uma das grandes inovações será a integração de um leitor de Libras, tornando o site acessível para pessoas surdas ou com deficiência auditiva.

Em relação à prospecção de novos clientes, estamos focando na interação do usuário para criar um algoritmo de recomendação personalizado. Isso permitirá que o site se adapte dinamicamente em pontos-chave, apresentando conteúdo relevante e direcionado para cada usuário. Essa abordagem autônoma garantirá que o usuário encontre exatamente o que procura, aumentando assim a eficácia da prospecção e melhorando a experiência do cliente.

Com essas melhorias, estamos confiantes de que o novo site da Salesforce se destacará não apenas pela sua renomada reputação, mas também pela sua acessibilidade e pela capacidade de atrair e engajar novos clientes de forma eficaz. Estamos entusiasmados para ver o impacto positivo que essas mudanças trarão para a empresa e seus usuários.

O projeto Salesforce é uma aplicação que gerencia serviços oferecidos pela empresa, permitindo que usuários se cadastrem, realizem o login, acessem serviços disponíveis, e consultem informações sobre os serviços.

Resumindo, a entidade Pessoa possui uma relação com Conta, Empresa e Endereço. A entidade serviço possui diversas instancias dos serviços para serem listados e exibidos no front-end, esse serviço ira pronto, com todas as caracteristicas e até mesmo imagem, para que o front-end somente renderize.

# Principais funcionalidades

Todas as funcionalidades são integradas com o front-end, logo na api é feita validação, tratamento de erro, autenticação, além de exibir todos os dados de uma forma que o front-end consiga consumir sem nenhum tipo de tratamento.

**Cadastro**

- Nessa funcinalidade o serviço AuthService.class insere no banco 4 entidades chaves para a aplicação, atraves do dto RegisterRequestDTO, aqui são extraidas informações p/ Endereço, Empresa, Pessoa e Conta;

- Vale lembrar que todos os campos a se inserir são validados, com o hibernate validator,além de verificar se o email já está em uso.

**Login**

- Nessa funcinalidade o o serviço AuthService.class, consulta no banco, com um SELECT, se a conta existe, passando por essa etapa é validada as informações passadas como email e senha e o login é efetuado retornando resposta 200. Caso contrario e retornado um erro dizendo que a conta não foi encontrada.

**FindAll dos Serviços**

- Essa funcionalidade retorna todos os serviços com seus respectivos atributos, inclusive a imagem para que o front-end tenha o trabalho de apenas renderizar o conteudo.

**FindAll das Contas**

- Retorna todas as contas

**FindAll dos Endereços**

- Retorna todos os endereços

**FindAll das Empresas**

- Retorna todas as empresa

**FindAll das Pessoas**

- Retorna todas as pessoas

# Tabela dos endpoints (API Restful)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HTTP | Proteção | Rota | Descrição | Observação |
| POST | Livre | /auth/register | Cadastro | /--/ |
| POST | Livre | /auth/login | Login | /--/ |
| GET | Livre | /servicos | Retorna todos os serviços | /--/ |
| GET | Precisa ter feito o login | /contas | Retorna todas as contas | Necessario colocar o token do login no Authorization do tipo Bearer Token |
| GET | Precisa ter feito o login | /empresas | Retorna todas as empresas | Necessario colocar o token do login no Authorization do tipo Bearer Token |
| GET | Precisa ter feito o login | /enderecos | Retorna todos os endereços | Necessario colocar o token do login no Authorization do tipo Bearer Token |
| GET | Precisa ter feito o login | /pessoas | Retorna todas as pessoas | Necessario colocar o token do login no Authorization do tipo Bearer Token |

# Corpo das Requisições

**Cadastro**

POST /auth/register

{

"endereco":{

"pais": "Brasil"

},

"empresa":{

"nome": "Drummond",

"numFuncionario": "10 a 150"

},

"pessoa": {

"nome": "pedro",

"sobrenome": "almeida",

"cargo": "ti"

},

"conta": {

"email": "pedro2@gmail.com",

"senha": "2011"

}

}

**Login**

POST /auth/login

{

"email": "pedro2@gmail.com",

"senha":"2011"

}

Para requisições protegidas devemos colocar o token gerado pelo login, em cada requisição, siga os exemplos abaixo.

A computer screen shot of a black screen

Description automatically generated

Copiamos o token gerado no login;

A screenshot of a computer

Description automatically generated

Na aba da requisição colocamos o Authorization como Bearer Token;

A screenshot of a computer

Description automatically generated

Em seguida só colar o token e chamar a requisição.

# Protótipos das telas

Home  
A person in a blue shirt

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedLogin

Cadastro

A screenshot of a computer

Description automatically generated

Servicos

A screenshot of a computer

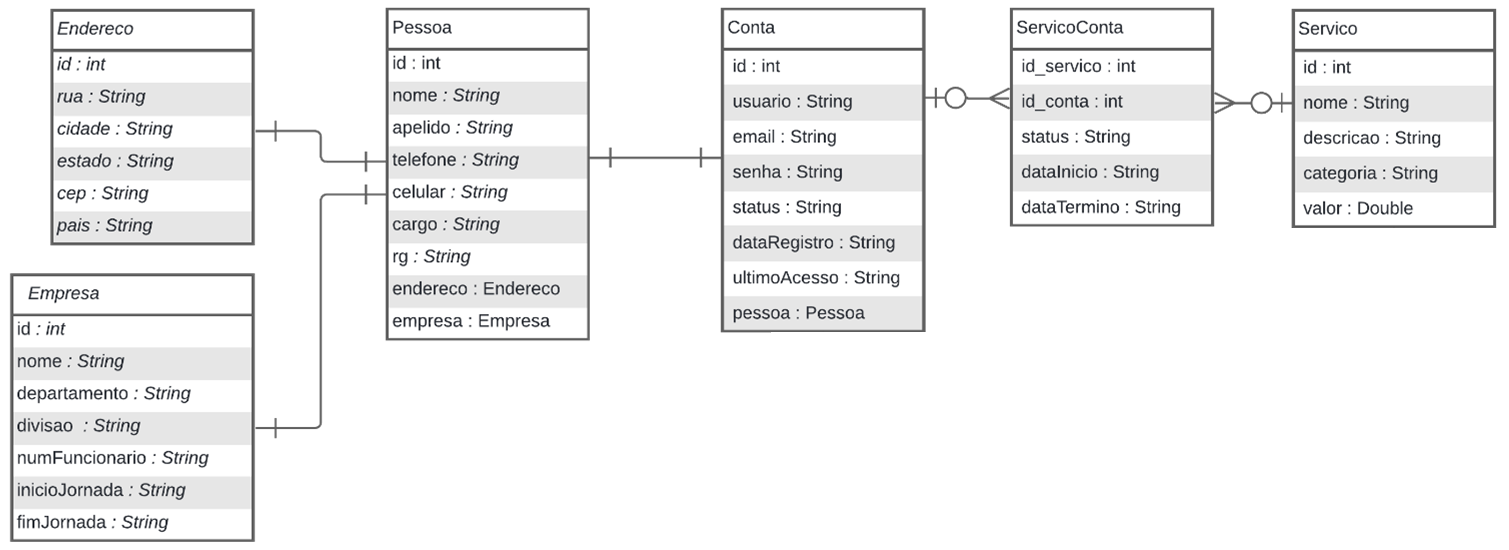
Description automatically generated

# Modelagem Banco de Dados

**A screenshot of a computer program

Description automatically generated**Tabelas e relacionamentos

Diagrama de classes



# Estrutura do projeto

Na estrutura do projeto existe alguns packges extremamente importantes, são eles: SalesForceApplication, controllers, dto, model, repositories, security e services

com.fiap.salesForce

No package de salesForce existe apenas uma classe resposavel pela execução do projeto a SalesForceApplication.

Controllers: Responsáveis por gerenciar as requisições HTTP recebidas pelo servidor, direcionando-as para os serviços apropriados e retornando as respostas adequadas aos clientes.

DTO (Data Transfer Objects): Objetos utilizados para transferir dados entre camadas da aplicação, facilitando a comunicação e garantindo a segurança e integridade dos dados transferidos.

Model: Contém as classes que representam as entidades do domínio do sistema, refletindo a estrutura dos dados armazenados no banco de dados.

Repositories: Interface com o banco de dados, fornecendo métodos para operações CRUD (Create, Read, Update, Delete) e consultas específicas nas entidades.

Security: Gerencia a autenticação e autorização, protegendo a aplicação contra acessos não autorizados e ataques, e implementando políticas de segurança.

Services: Implementam a lógica de negócio da aplicação, orquestrando as operações necessárias, chamando os repositórios e manipulando os DTOs conforme necessário.

# Procedimento para executar

Essa API esta online, logo não é necessario baixar e rodar na sua maquina, o host a baixo é o host da api, nele vc pode testar todas as rotas.

Host: <https://challengesalesforce-production.up.railway.app/>

Exemplo acessando rota: <https://challengesalesforce-production.up.railway.app/servicos>

Mas caso deseje executar a aplicação localmente, faça o download do projeto, execute a classe SalesForceApplication.class no package com.fiap.salesForce, e em seguida realize os testes da api no postman ou em outro software que desejar.

# Link do repositório com do código Java do GitHub.

Link: https://github.com/KauaAlmeidaSilveira/apiDDD\_ChallengeSalesForce