**SP Medical Group**

Documentação

Sumário

[1. Resumo 3](#_Toc80311768)

[2. Descrição do projeto 3](#_Toc80311769)

[3. Banco de dados relacional 3](#_Toc80311770)

[4. Modelagem de dados 3](#_Toc80311771)

[Modelo Conceitual 3](#_Toc80311772)

[Modelo Lógico 3](#_Toc80311773)

[Modelo Físico 3](#_Toc80311774)

[Cronograma 3](#_Toc80311775)

[Trello 3](#_Toc80311776)

# Resumo

Com o crescimento da Sp Medical Group foi necessária a criação de uma aplicação para facilitar os trabalhos da clínica, principalmente com o relacionamento de diferentes médicos e pacientes a suas consultas e praticidade para administradores que são responsáveis por esses dados.

# Descrição do projeto

Com o uso de diferentes recursos e tecnologias para o manuseio e armazenamento de dados foi construída uma aplicação responsável por digitalizar o trabalho da clínica e abrir um espaço para novas clínicas parceiras ou filiadas que busquem o uso do sistema

# Banco de dados relacional

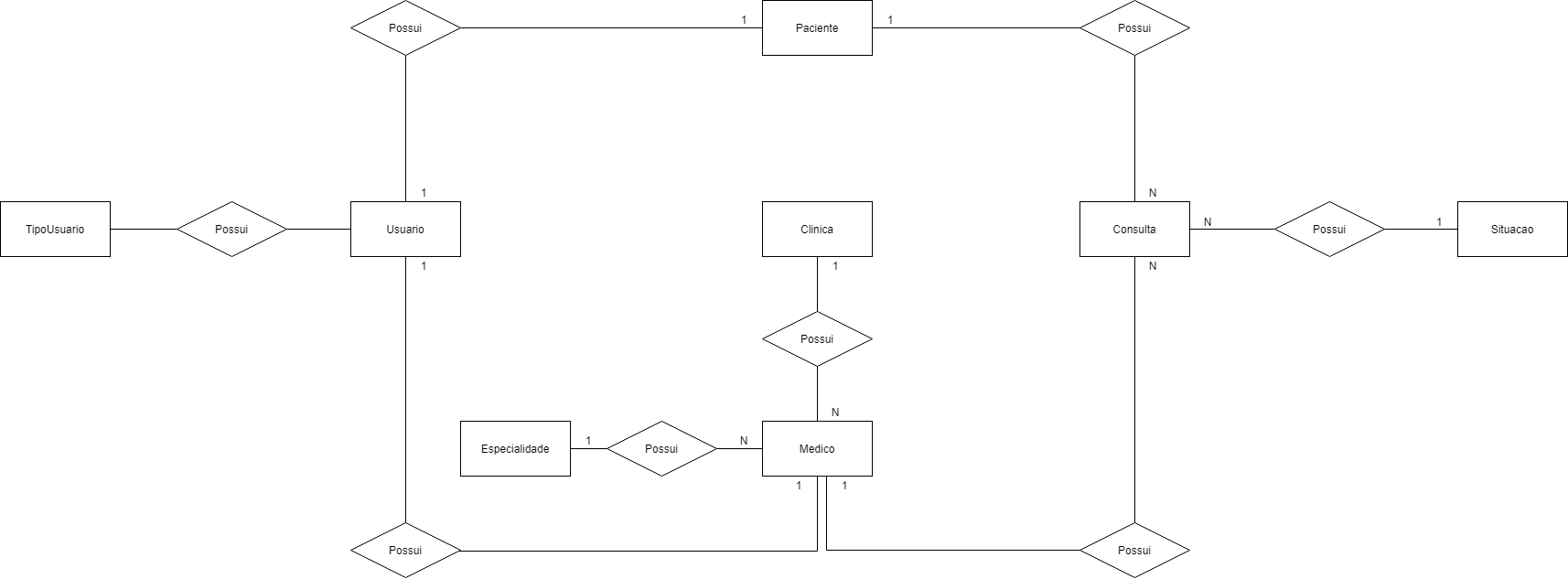
Com o uso da linguagem de SQL e o Software do Microsoft SQL Server, foi criado um banco de dados, por meio de scripts de definição, modificação e consulta, atendendo as necessidades do cliente, com o relacionamento dos usuários a seus tipos, consultas médicas e outros dados individuais, como documentos e especialidades no caso dos contribuintes da área da saúde.

Um banco de dados é uma estrutura que define um padrão para armazenamento e manutenção de diferentes bases de dados, atuando com a criação de entidades. Em bancos de dados relacionais, essas entidades têm seus relacionamentos definidos por cardinalidades, que definem diferentes tipos de relacionamento entre cada uma que se tornam físicos pelo uso de diferentes chaves, estrangeiras e primárias.

# Modelagem de dados

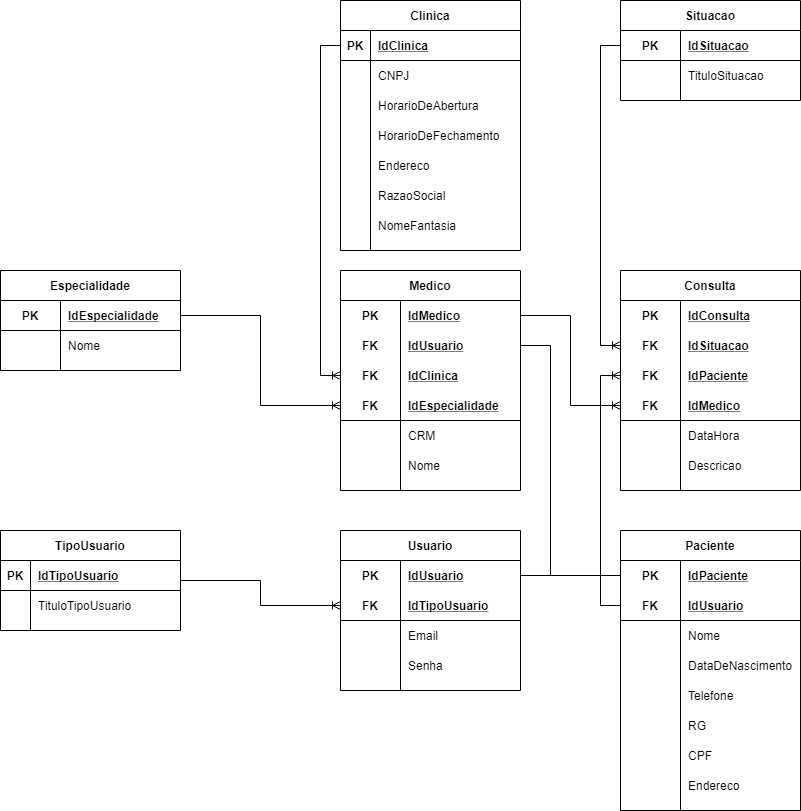
Modelagens são maneiras de visualizar e planejar o banco de dados por meio de diagramas ou tabelas, podendo ser utilizadas antes, durante ou depois da criação do BD.

## Modelo Conceitual



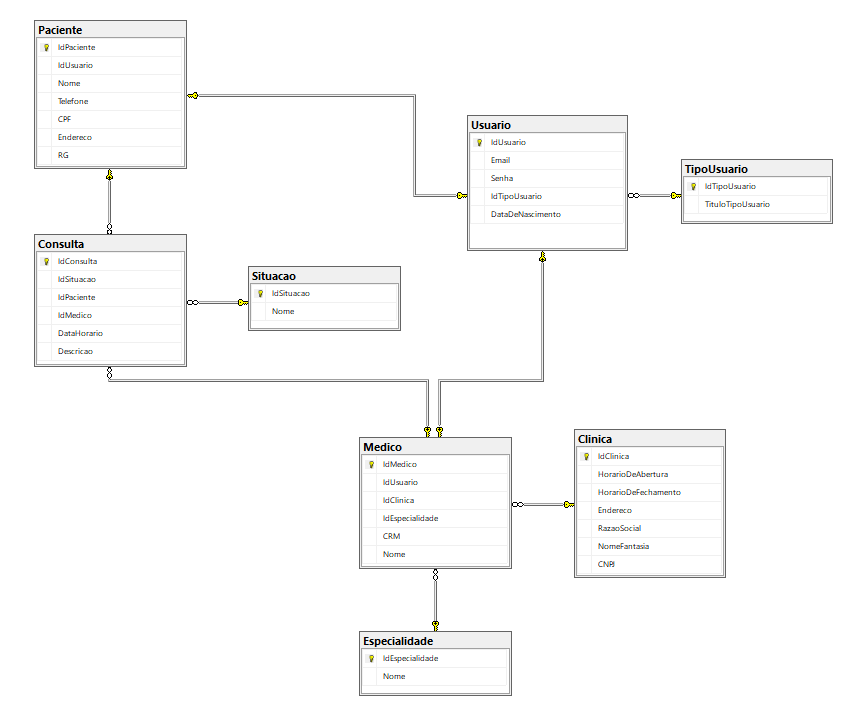
No modelo conceitual, foram definidas as relações entre as entidades de maneira geral. Relações de cardinalidade 1:1 (um para um) indicam uma relação individual entre cada registro das tabelas, como no Usuário que possui um registro diferente para cada paciente, médico ou administrador cadastrado. Relações de cardinalidade 1:N (um para muitos) indicam que um registro de uma determinada tabela pode se relacionar com diversos registros de outra, como no caso das especialidades que podem se relacionar com diferentes médicos. Relacionamentos de cardinalidade N:N (muitos para muitos) são utilizados quando diversos registros de uma tabela podem se relacionar com diversos registros de outra tabela. Em bancos de dados planejados de maneira correta, as relações N:N resultam na criação de uma tabela intermediária entre as entidades.

## Modelo Lógico



Modelos lógicos ainda definem seus relacionamentos por meio do conceito de cardinalidade, onde “N” é representado por finais ou inícios de linhas com uma espécie de subdivisão, e “1” é representado com finais ou inícios inalterados. A diferença entre a modelagem lógica e conceitual é a presença dos campos e chaves das entidades, associando os relacionamentos às chaves primárias (PK) e estrangeiras (FK), além de evidenciar onde as diferentes colunas de dados não-chaves ficarão.

## Modelo Físico



Por fim, modelos físicos apresentam a estrutura do BD em si, evidenciando as chaves e ligações na estrutura em que o banco de fato funciona após o uso dos scripts. Também pode ser idealizado de outras formas, como no excel com o uso de dados fictícios.

## Cronograma

|  |  |  |
| --- | --- | --- |
|  | Dia 1 | Dia 2 |
| Modelo Conceitual | X |  |
| Modelo Lógico | X |  |
| Modelo Físico | X |  |
| Script DDL | X |  |
| Script DML |  | X |
| Script DQL |  | X |

### Trello: Sprint 1 BD

<https://trello.com/b/BnvTmAN5/senaispmedgrouppedrolucas2dt>

5. Back-End

Para o backend foi feita uma API (Interface de programação de aplicações) utilizando a estrutura proposta pelo repository pattern, um padrão de construção da API que se baseia em 4 camadas:

1. Domains ou Domínios, representações das entidades do Banco de Dados;
2. Interfaces: responsáveis por definir as regras de negócio da aplicação;
3. Repositories ou Repositórios, responsáveis por executar as regras de negócio;
4. Controllers: Responsáveis por manusear as requisições do frontend

Além disso, foram seguidos a risca os padrões de API definidos pelo REST, como na nomeação de recursos (links e URLs) de maneira intuitiva e condizente com as classes criadas, uso de verbos corretos para diferentes requisições nos controllers (POST para cadastros e adições ao banco, DELETE para remoções, GET para leituras e etc) e retorno de status codes(códigos que definem o proceder da requisição, como 201(Created) para adições de objetos feitas com sucesso no banco, 404(Not Found) para valores não encontrados no Banco de acordo com a requisição e etc) corretos para diferentes situações.

Outros pontos adicionais foram o uso do JWT(Json web token) para uma comunicação segura com o frontend, com o uso de informações ocultadas. E o swagger para a criação da documentação mais geral e técnica da API.

## Funcionalidades

### Usuários

1. **Administrador**, responsável pelo manuseio de informações sensíveis.
2. **Médico**, profissional da saúde que atua em alguma das clínicas parceiras.
3. **Paciente**, cliente com cadastro na aplicação.

### Funcionalidades

1. **Administradores** são os únicos autorizados a cadastrar, atualizar e deletar qualquer tipo de informação, desde Clínicas, Especialidades e Consultas até novos tipos de usuário. Também podem mudar a situação de diferentes consultas para “cancelada”.
2. **Médicos** podem visualizar consultas relacionadas eles e alterar suas descrições. Também podem visualizar um paciente específico pelo seu Id (identificador numérico dentro do sistema).
3. **Pacientes** podem visualizar consultas relacionadas a eles e visualizar um médico específico pelo seu Id(Identificador numérico dentro do sistema).
4. A visualização das diferentes especialidades disponíveis, juntamente com seus médicos, e clínicas é **pública**, e pode ser vista até mesmo por usuários sem cadastro prévio no sistema.
5. Todos os usuários **logados** (autenticados pelo sistema) podem adicionar imagens de perfil, e usuários **cadastrados** podem fazer login para terem acesso às funcionalidades descritas acima, dependendo do tipo de usuário em que se encaixam.

### Azure

A API também está em um servidor da Azure. Pode ser acessada pelo seguinte link: <http://spmedgroup-api-pedro.azurewebsites.net/>

## Tópicos secundários: Backend

### Trello : Sprint 2 Backend

<https://trello.com/b/497slvZ7/spmedgroupapisenai>, quadro do trello responsável pela organização durante a criação da API.

### Github: Testes e considerações

Todo o código construído, desde os scripts do Banco de Dados até a API podem ser acessados pelo seguinte [repositório do github](https://github.com/PedroLCamara/Projeto_SpMedGroup_Senai). É recomendado o acesso ao mesmo, já que testes mais práticos da API não podem ser feitos pelo Azure, por conta da necessidade de utilização de diferentes tokens para o acesso de determinadas funcionalidades, como foi explicado acima.

Portanto, com o uso do software Postman, é possível importar o arquivo presente na pasta de mesmo nome no repositório para ter acesso as funcionalidades mencionadas, e testa-las seja inserindo novos dados ou acessando os já presentes na base de dados fornecida no início do projeto.