**SP Medical Group**

Clínica Possarle

Feito por Robert Sena

Sumário

[1. Resumo 2](#_Toc72860274)

[2. Descrição do projeto 2](#_Toc72860275)

[3. Modelagem de Dados 3](#_Toc72860276)

[Modelo Conceitual 4](#_Toc72860277)

[Modelo Lógico 4](#_Toc72860278)

[Modelo Físico 4](#_Toc72860279) e 5

[Cronograma 5](#_Toc72860280)

[Trello 5](#_Toc72860281)

[4. Back-End 6](#_Toc72860282)

[Perfis de usuário: 7](#_Toc72860283)

[Funcionalidades: 7](#_Toc72860284)

# Resumo

Aqui vemos um pouco do processo de criação do projeto SP Medical Group, os passos que seguimos, desde o começo até seu final, além dos dias que levamos para realizar o projeto.

# Descrição do projeto

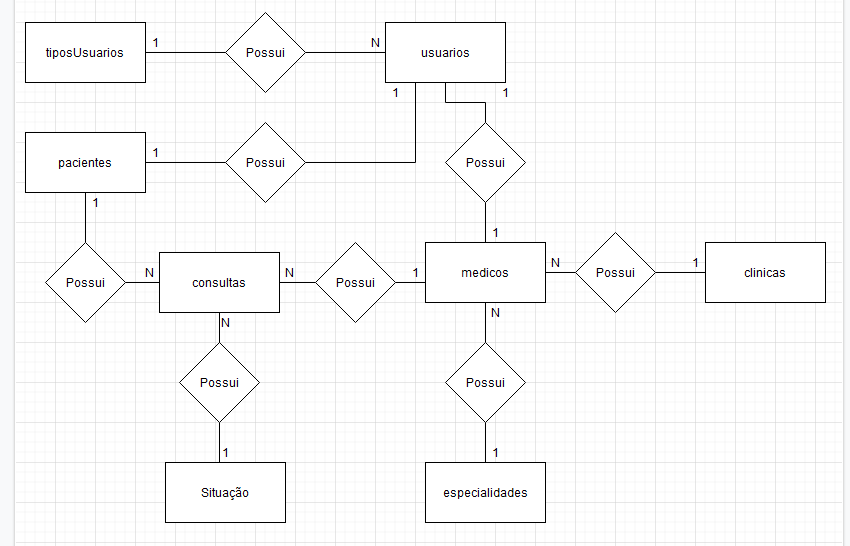
Fui para desenvolver um projeto web/mobile integrado onde seja possível realizar a gestão da clínica de forma automatizada e ter acesso fácil aos dados sobre as informações de seus pacientes. Neste período do nosso projeto, trabalhamos com o banco de dados relacional.

# Modelagem de Dados

Banco de dados relacional é o tipo de banco que é criado com uma maneira de representar os dados em uma tabela, sendo essas tabelas relacionadas, e um banco de dados não-relacional representam dados em umas coleções de documentos JSON, sendo essas coleções não relacionadas. Um banco de dados é de suma importância, porque economiza tempo no armazenamento de dados, é de fácil uso e é super organizada

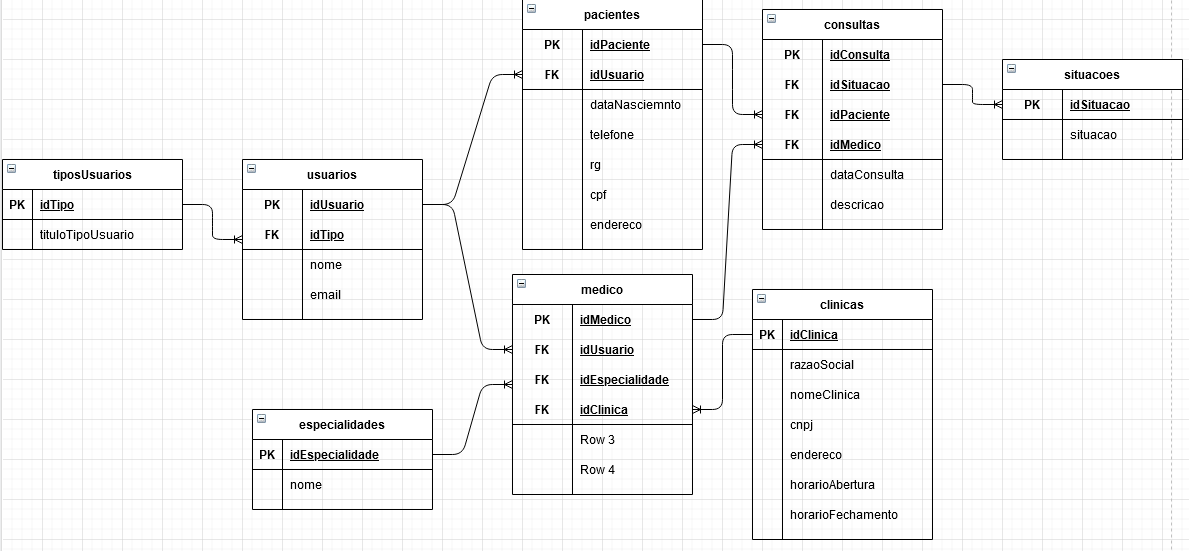
## Modelo Conceitual

Modelo feito para descobrir as entidades do projeto e suas cardinalidades



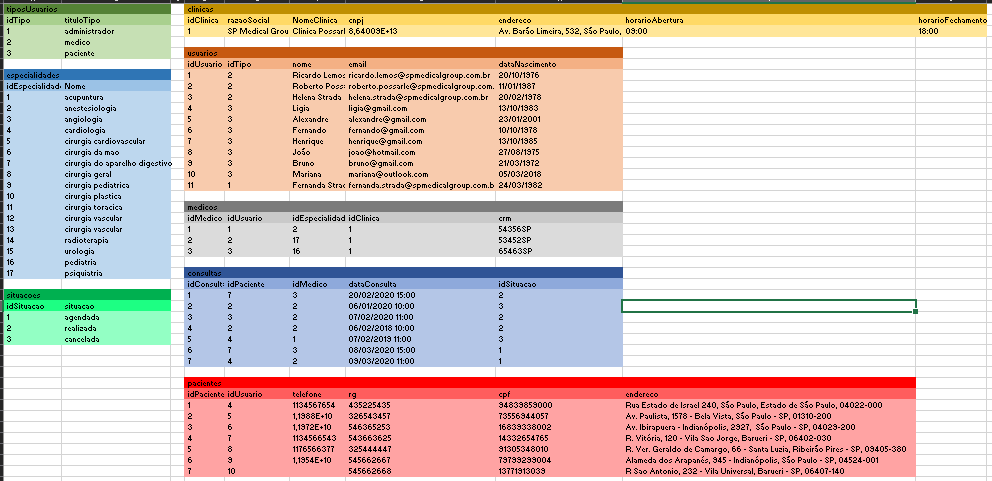
## Modelo Lógico

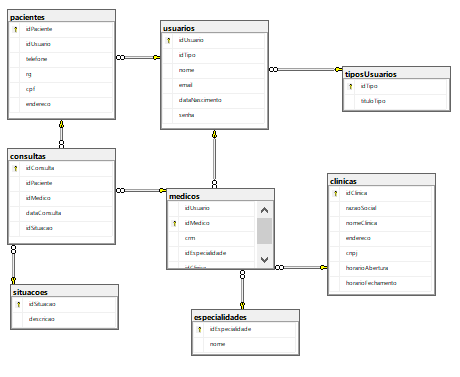
Modelo feito para saber quais as ligações entre as tabelas e suas chaves, além de inserir seus dados para que fique melhor na inserção de dados.



## Modelo Físico

Este modelo é criado com todas as colunas do nosso projeto preenchidas, o que facilita para o desenvolvedor saber o que deve e o que não deve ser inserido no banco de dados. e a imagem exportada (printscreen da planilha do excel, por exemplo, e/ou o diagrama exportado do SSMS)





## Cronograma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 5 |
| Modelo Conceitual | X |  |  |  |  |
| Modelo Lógico |  | X | X |  |  |
| Modelo Físico |  |  |  | X | X |

## Trello

https://trello.com/b/wOA2r1pE/projeto-sp-medical-group

# Back-End

O código do sistema foi desenvolvido através de uma API utilizando o Microsoft Visual Studio, uma IDE criada para programarmos

**API** é um conjunto de padrões e instruções estabelecidos para utilização do software, definindo as requisições e as respostas seguindo o protocolo HTTP, neste caso expresso no formato JSON, para que seja possível acessar o sistema em diversos dispositivos distintos sem a preocupação com a linguagem que será utilizada por estes.

Além disso, foi utilizado o estilo de arquitetura REST.

**API** – Application Programming Interface – Interface de Programação de Aplicativos.

**HTTP** – Hypertext Transfer Protocol – Protocolo de Transferência de Hipertexto.

**JSON** – JavaScript Object Notation – Notação de Objetos JavaScript.

**REST** – Representational State Transfer – Transferência de Estado Representacional. É o que permite a conexão entre aplicativos

**POSTMAN** – Aplicativo criado para testar uma API sem precisar do FRONT-END

**JWT** – JSON Web Token – Usado para a autenticação do sistema

Insira também um passo a passo de como executar a API criada e testar as requisições no Postman.

Para executar a API criada, deve-se ter, no mínimo, o .NET Runtime instalado no seu computador para rodar nossa aplicação, e caso não tenha acesse este link e baixe: <https://dotnet.microsoft.com/download>. Feito isso, entre na pasta do projeto e digite cmd no caminho da pasta e dê enter, que abrirá uma janela. Nela, digite assim: dotnet run. Feito isso nossa aplicação estará rodando

Para testar as requisições criadas e exportadas pelo Postman, basta abrir a pasta que foi exportada e, com a API rodando, ir testando cada um dos métodos criados. Não se esqueça das autorizações que colocamos

### Perfis de usuário:

1. **Administrador**: Colaborador da área administrativa da clínica;
2. **Médico**: Colaboradores que atuam na área da saúde;
3. **Paciente**: Clientes da clínica;

### Funcionalidades:

1. O **administrador** poderá cadastrar qualquer tipo de usuário;
2. O **administrador** poderá agendar uma consulta, informando o paciente, data do agendamento e qual médico irá atender a consulta (o médico terá sua especialidade);
3. O **administrador** poderá cancelar o agendamento;
4. O **administrador** deverá informar os dados da clínica (endereço, horário de funcionamento, CNPJ, nome fantasia e razão social);
5. Qualquer **médico** poderá ver os agendamentos (consultas) associados a ele;
6. O usuário **médico** poderá incluir a descrição da consulta vinculada ao paciente;
7. O usuário **paciente** poderá visualizar suas próprias consultas;