**SP Medical Group**

Documentação

Sumário

[1. Resumo 3](#_Toc90481069)

[2. Descrição do projeto 3](#_Toc90481070)

[3. Modelagem de Dados 3](#_Toc90481071)

[Modelo Conceitual 3](#_Toc90481072)

[Modelo Lógico 4](#_Toc90481073)

[Modelo Físico 4](#_Toc90481074)

[Cronograma 5](#_Toc90481075)

[Trello 5](#_Toc90481076)

[4. Back-End 5](#_Toc90481077)

[Funcionalidades 6](#_Toc90481078)

[Perfis de usuário: 6](#_Toc90481079)

[Funcionalidades: 6](#_Toc90481080)

[5. Front-End 7](#_Toc90481081)

[6. Mobile 7](#_Toc90481082)

[7. NoSQL 7](#_Toc90481083)

# Resumo

Essa documentação está relacionada ao meu projeto de banco de dados SP Medical Group, com isso nesse documento será possível ver sobre esse projeto e assuntos relacionados.

# Descrição do projeto

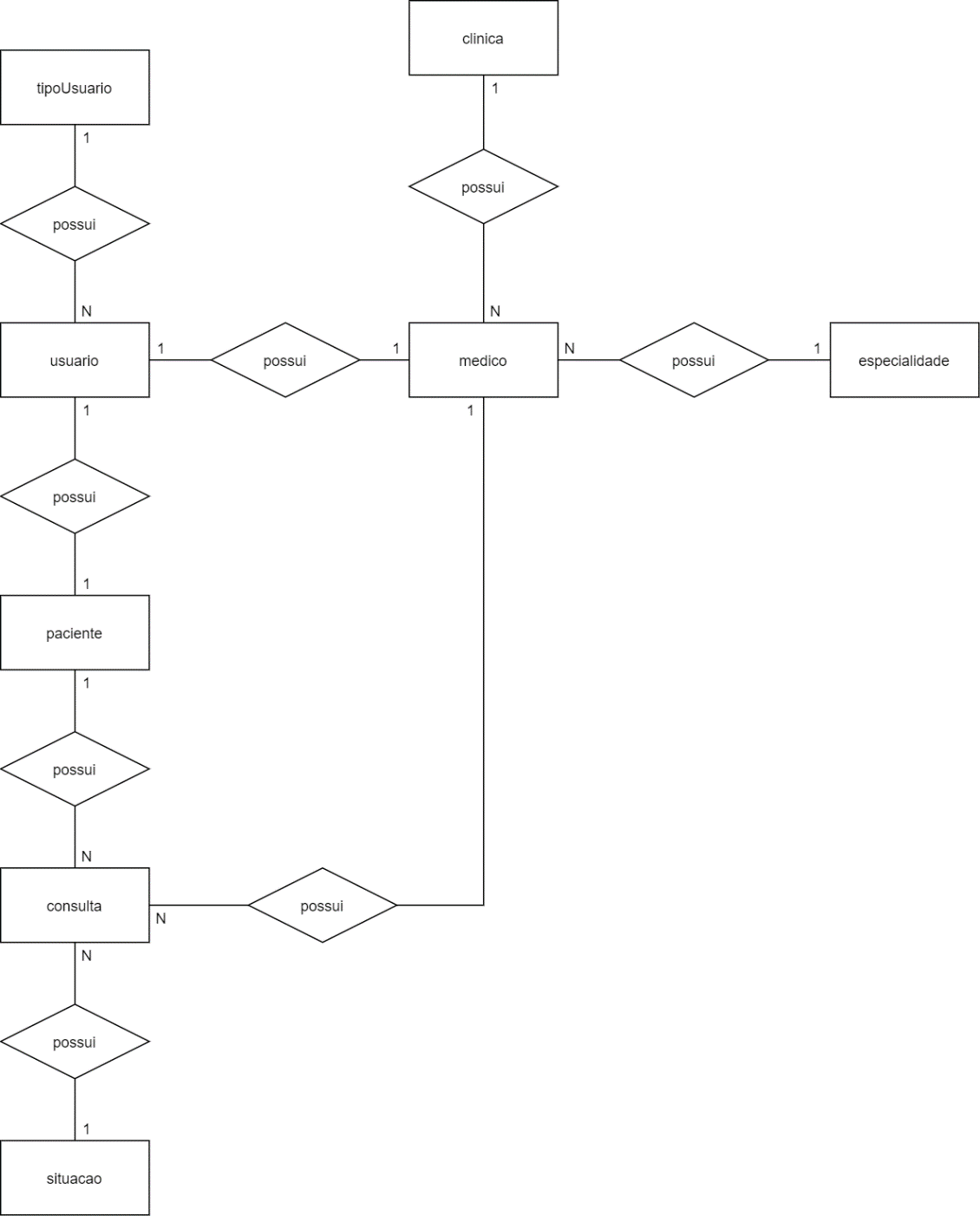
O projeto foi feito tendo como base uma clínica médica que precisava de um banco de dados para realizar suas consultas e facilitar ao ver as informações dos pacientes. E para realizar isso comecei um Trello para me organizar e depois fiz as modelagens do projeto, e com isso consegui fazer os três scripts (DDL, DML, DQL).

# Modelagem de Dados

A modelagem de dados é o planejamento explicando como será o projeto, tendo suas características, para assim evitar erros durante a fase dos scripts.

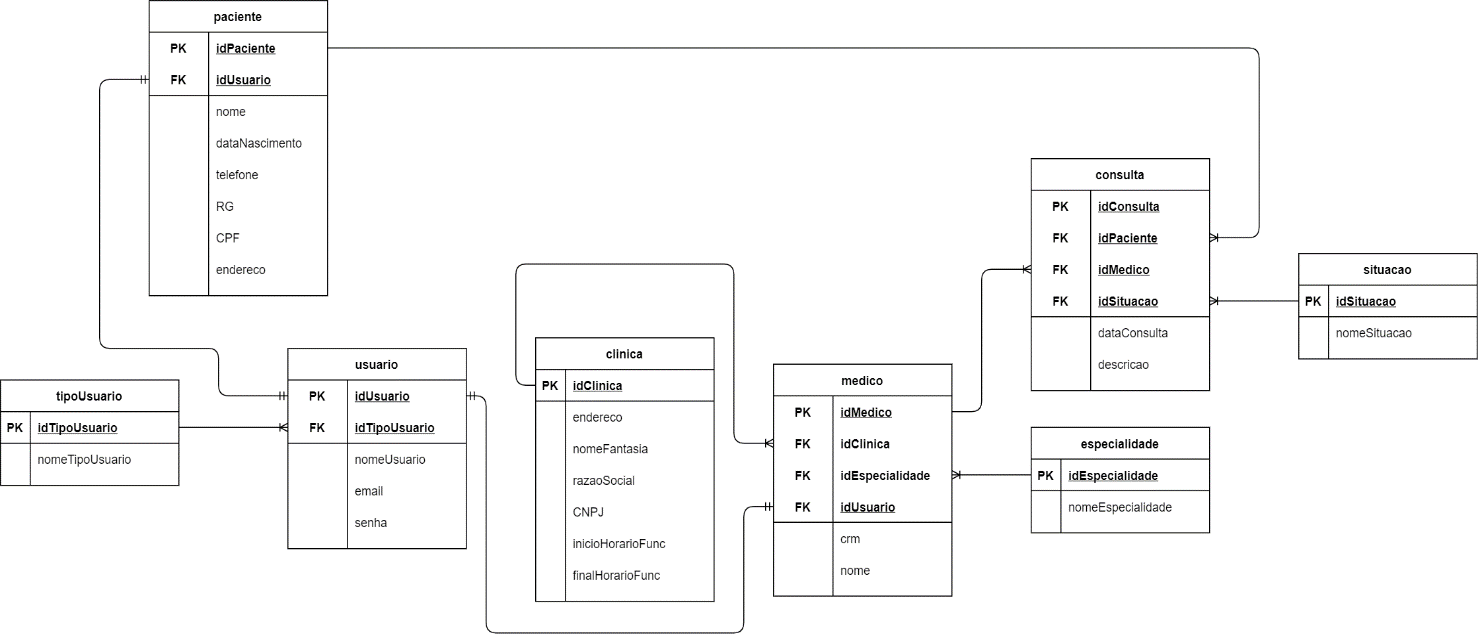
## Modelo Conceitual

Várias consultas são feitas por um médico, e também um paciente pode ter várias consultas, uma clínica possui vários médicos que também possuem somente uma especialidade, e as consultas tem uma situação.



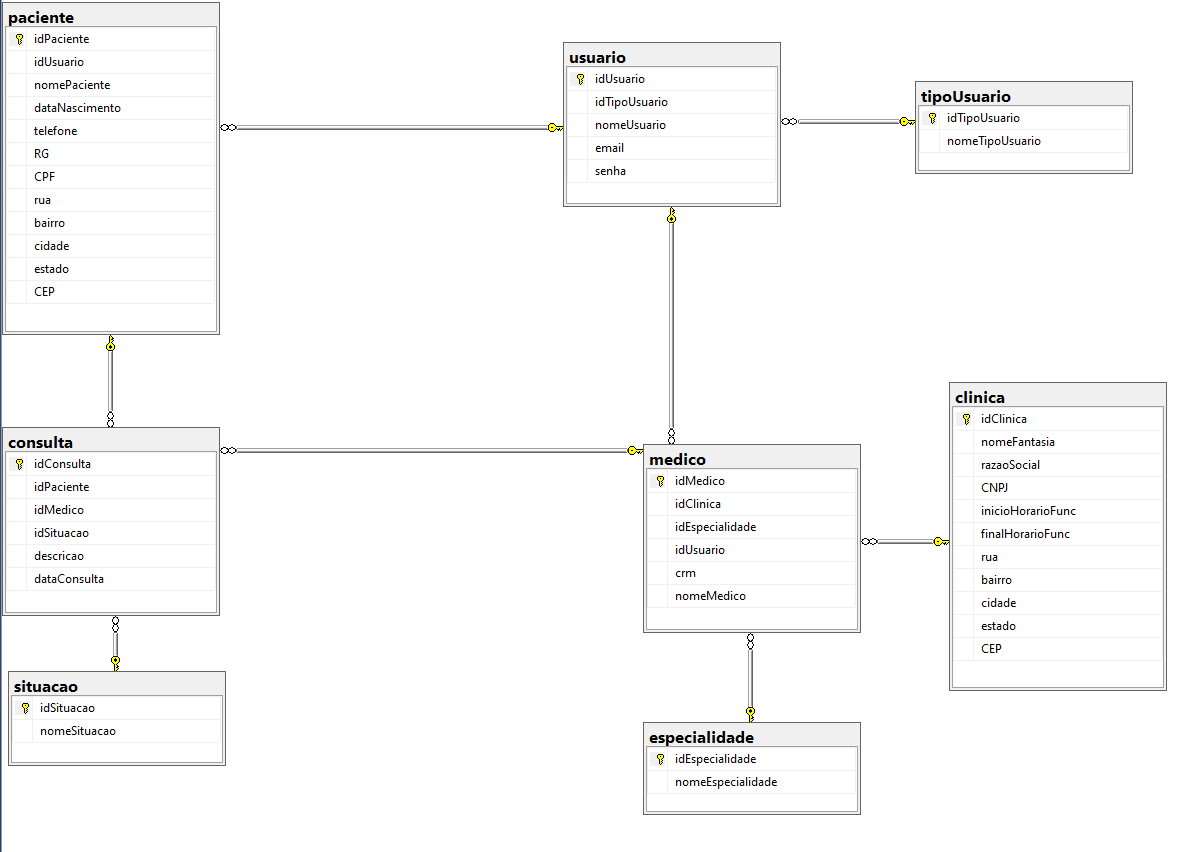
## Modelo Lógico

Nesse modelo já tem mais detalhes sobre o que cada tabela terá e como será melhor a suas relações. Tendo as chaves primárias, estrangeiras e os atributos que serão colunas dessas tabelas.



## Modelo Físico

O modelo físico é aquele que já tem todos os dados, ou boa parte em sua tabela mostrando onde cada dado deve ficar no banco de dados.



## Cronograma

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 5 | Dia 6 | Dia 7 | Dia 8 | Dia 9 |
| Modelo Conceitual | X |  |  |  |  |  |  |  |  |
| Modelo Lógico | X |  |  |  |  |  |  |  |  |
| Modelo Físico | X |  |  |  |  |  |  |  |  |
| DDL |  | X |  |  |  |  |  |  |  |
| DML |  |  | X |  |  |  |  |  |  |
| DQL |  |  |  | X | X |  |  |  |  |
| API |  |  |  |  |  | X | X | X |  |
| Postman |  |  |  |  |  |  |  |  | X |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Dia 10 | Dia 11 | Dia 12 | Dia 13 | Dia 14 | Dia 15 | Dia 16 | Dia 17 | Dia 18 |
| Front-end | X | X | X | X |  |  |  |  |  |
| Mobile |  |  |  |  | X | X | X | X |  |
| NoSQL |  |  |  |  |  |  |  |  | X |

### Trello

<https://trello.com/b/lgvtJRD5/sp-medical-group>

# Back-End

O código do sistema foi desenvolvido através de uma API utilizando o Microsoft Visual Studio, também foi utilizado Microsoft SQL Server Management Studio para o banco de dados, e o Postman para fazer o teste de requisições.

**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 – Interface de Programação de Aplicativos.

Primeiramente é preciso que entre no meu repositório do Github, faça um git clone em seu computador para que esteja na sua máquina e depois disso encontre o arquivo da solução do projeto e execute-o, logo após isso abra o seu Microsoft SQL Server Management Studio (SSMS) adicione todas as tabelas e as informações, depois entre no Visual Studio encontre a string de conexão com o banco de dados e mude de acordo com o SSMS de sua máquina, e agora já podemos abrir com Postman a coleção que está lá, para fazer a requisição é preciso estar logado, ou seja, terá que fazer uma requisição em login para conseguir acessar e usar, depois de logado poderá fazer os testes, podendo passar um JSON para cadastrar logado como ADM e etc.

## Funcionalidades

### Perfis de usuário:

1. **Administrador**: Área administrativa da clínica;
2. **Médico**: Profissional que irá atender os pacientes;
3. **Paciente**: O indivíduo que terá a consulta.

### Funcionalidades:

1. O **administrador** poderá cadastrar qualquer tipo de usuário;
2. O **administrador** poderá agendar uma consulta, onde será informado o paciente, data do agendamento e qual médico irá atender a consulta;
3. O **administrador** poderá cancelar o agendamento;
4. O **administrador** deverá informar os dados da clínica;
5. Qualquer **usuário** autenticado poderá ver todos os eventos cadastrados;
6. O **médico** poderá ver os agendamentos (consultas) associados a ele;
7. O **médico** poderá incluir a descrição da consulta que estará vinculada ao paciente;
8. O **paciente** poderá visualizar suas próprias consultas;

# Front-End

O código do Frontend foi desenvolvido usando a ferramenta Microsoft Visual Studio Code, sendo assim criado os arquivos JSX usando a biblioteca React JS que foi desenvolvida pela empresa Meta (Facebook).

Para isso foi criado o Html e Css que está de acordo com o layout de cada página, para que depois fosse colocado no arquivo JSX e também com funções para que funcionasse corretamente todas as páginas usando a biblioteca React JS.

**HTML:**

**CSS:**

**JSX:**

**React JS:**

# Mobile

O código do Mobile foi desenvolvido usando Android Studio, que na qual foi utilizado React-native que é também uma biblioteca, sendo assim criado a estrutura com componentes.

**React-native:**

# NoSQL

Foi utilizado MongoDB que é uma ferramenta para desenvolvimento de banco de dados não relacional, para que assim tivesse um mapa de onde os pacientes moram.

**MongoDB:**