**SP Medical Group**

Documentação

Sumário

[1. Resumo 3](#_Toc71092221)

[2. Descrição do projeto 3](#_Toc71092222)

[3. Modelagem de Dados 3](#_Toc71092223)

[Modelo Conceitual 3](#_Toc71092224)

[Modelo Lógico 3](#_Toc71092225)

[Modelo Físico 3](#_Toc71092226)

[Cronograma 3](#_Toc71092227)

[Trello 4](#_Toc71092228)

[4. Back-End 4](#_Toc71092229)

[Funcionalidades 5](#_Toc71092230)

[Sistema Web 5](#_Toc71092231)

[Perfis de usuário: 5](#_Toc71092232)

[Funcionalidades: 5](#_Toc71092233)

[Sistema Mobile 5](#_Toc71092234)

[Perfis de usuário: 5](#_Toc71092235)

[Funcionalidades: 5](#_Toc71092236)

# Resumo

Criamos um Projeto de organização de dados de uma Clínica, que pode organizar tanto os usuário, administradores, médicos , suas informações.

# Descrição do projeto

Uma nova clínica médica chamada SP Medical Group, empresa de pequeno porte que

atua no ramo da saúde, foi criada pelo médico Fernando Strada em 2020 na região da

Paulista em São Paulo. Fernando tem uma equipe de médicos que atuam em diversas

áreas (pediatria, odontologia, gastrenterologia etc.).

Sua empresa, por ser nova, iniciou a administração dos registros de forma simples,

utilizando softwares de planilhas eletrônicas e, com o sucesso da clínica, sua gestão

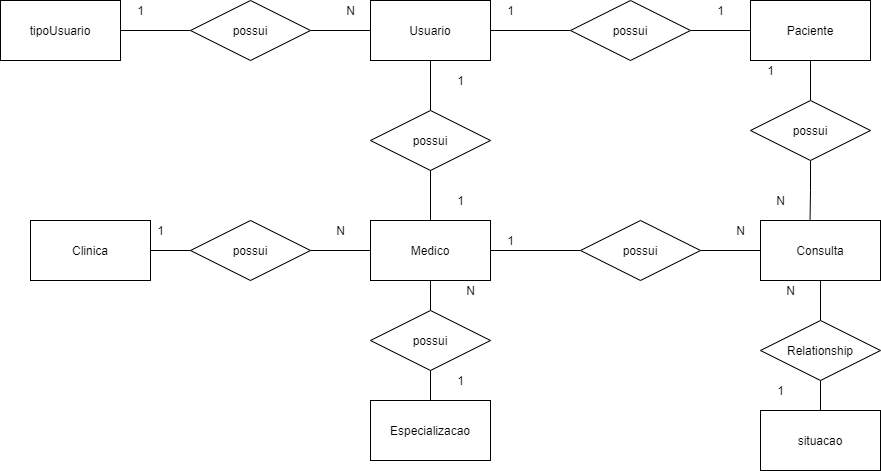
passou a se tornar complicada devido à alta demanda dos pacientes.

# Modelagem de Dados

A Modelagem de dados é a parte da ciência da computação que se encarrega de construir estruturas que permitam armazenar e recuperar informações em contextos específicos

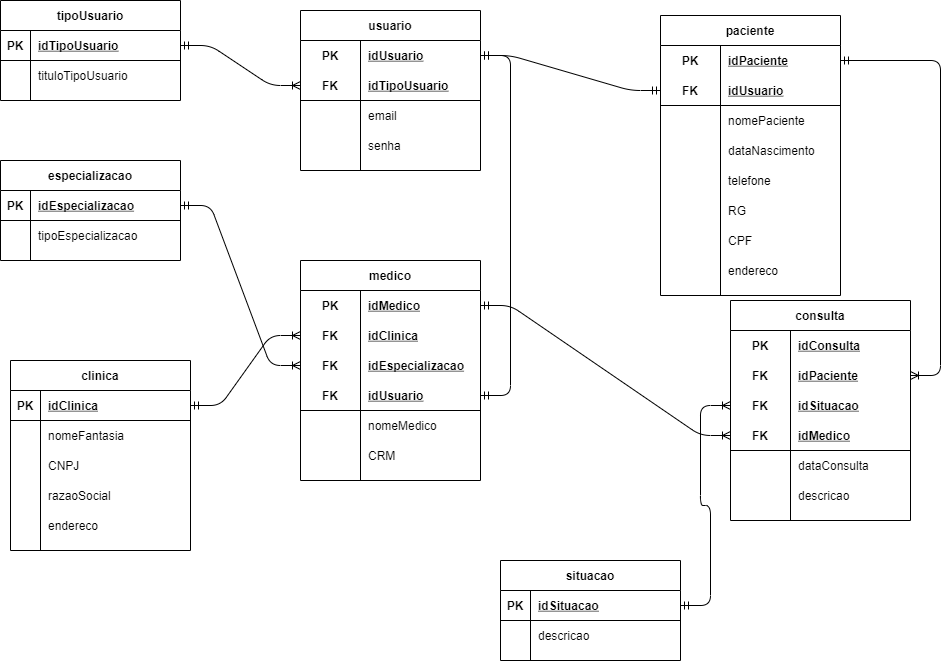
## Modelo Conceitual

O modelo conceitual é um modelo de dados de alto nível e não leva em conta banco de dados em si, mais sim a forma como as serão criadas para armazenar os dados.



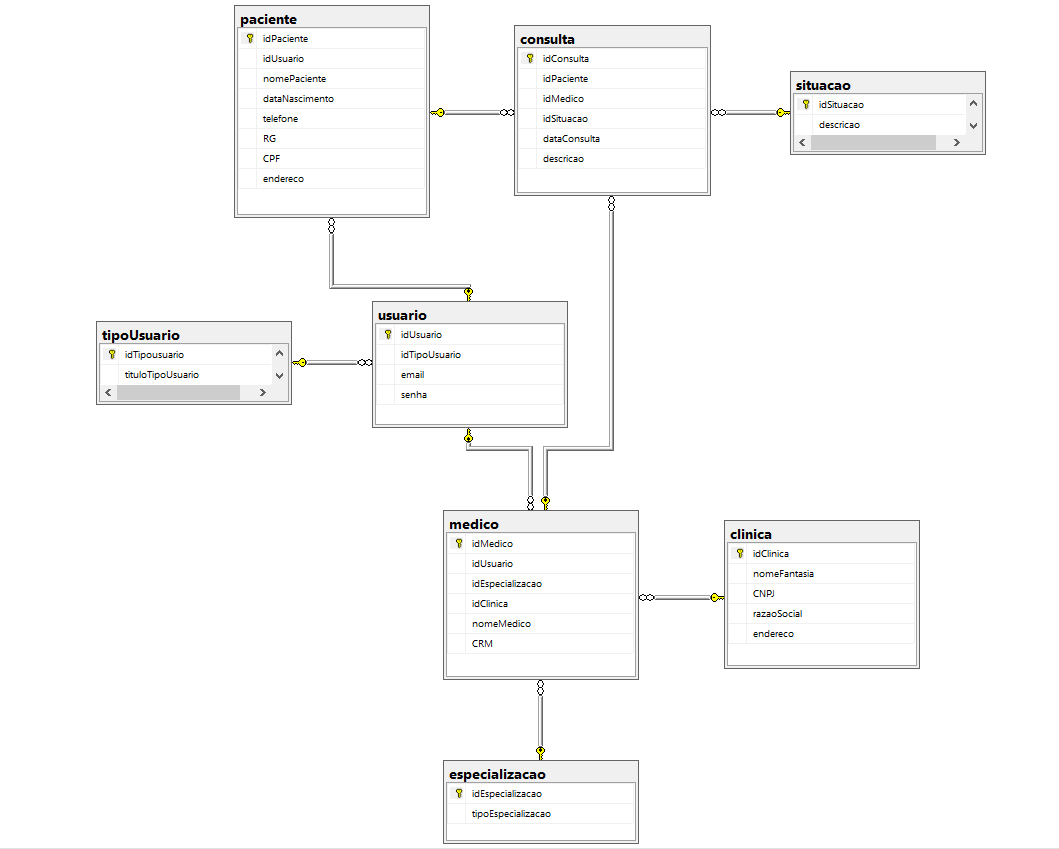
## Modelo Lógico

O modelo lógico já leva em conta algumas limitações e implementa recursos como adequação de padrão e nomenclatura, define as [chaves primárias e estrangeiras](https://www.luis.blog.br/chave primaria-chave-estrangeira-e-candidata.aspx), normalização, integridade referencial, entre outras.



## Modelo Físico

## No modelo físico fazemos a modelagem física do modelo de banco de dados. Neste caso leva-se em conta as limitações impostas pelo SGBD escolhido e deve ser criado sempre com base nos exemplos de modelagem de dados produzidos no item anterior, modelo lógico.



## Cronograma

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

### 

# Back-End

Utilizei a API para a montagem do back-end, utilizando Microsoft Visual Studio.

**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.

Além desses , utilizei outros tipos de recurso como:

- **Entity Framework**: o Entity Framework é um conjunto de tecnologias no ADO.NET que dão suporte ao desenvolvimento de aplicativos de software orientado a dados. Os arquitetos e desenvolvedores de aplicativos orientados a dados lutam com a necessidade de realizar dois objetivos muito diferentes. Precisam modelar as entidades, as relações e a lógica dos problemas de negócios que estão solucionando e também precisam trabalhar com os mecanismos de dados usados para armazenar e recuperar os dados.

-**Swagger** : Usado para adquirir uma documentação dos endpoints.

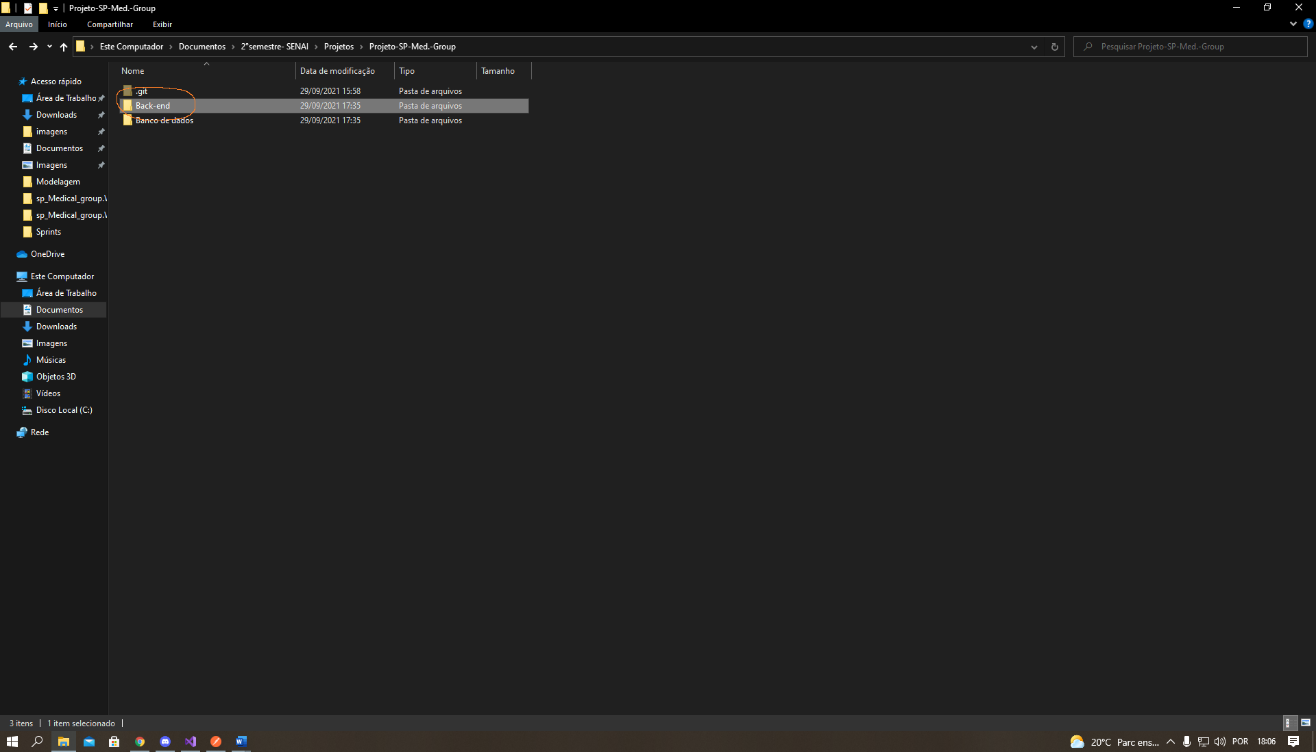
-**DataBase First**: É um tipo de estratégia para começar a sua API, mais para usar ela precisa ter um banco de dados já existente, para que ela consiga gerar classes (Domain, Context).

-**JWT** : É um método que realiza uma autenticação e autorização, por meio de um token gerado.

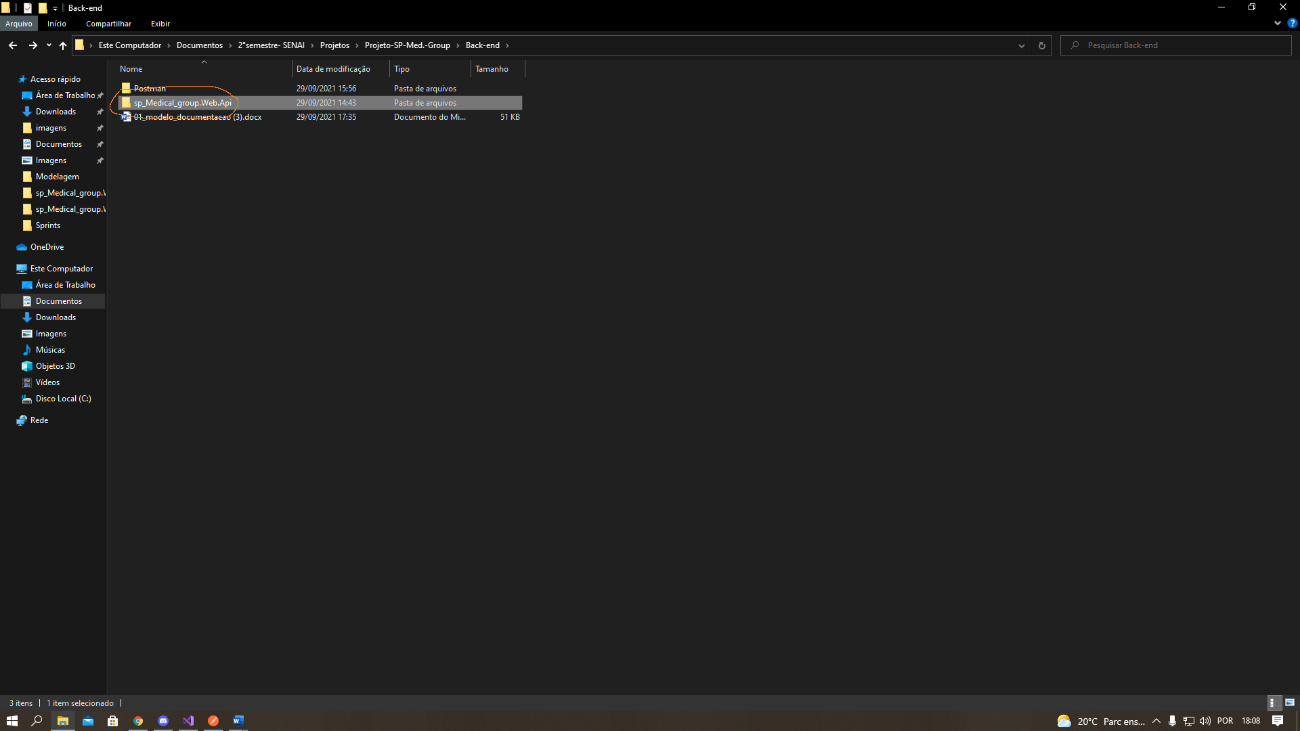
Passo a passo de como executar e testar a API:

-Clonar o repositório.

-Abrir a pasta back-end.



-Abrir a pasta sp\_Medical\_group.Web.Api.

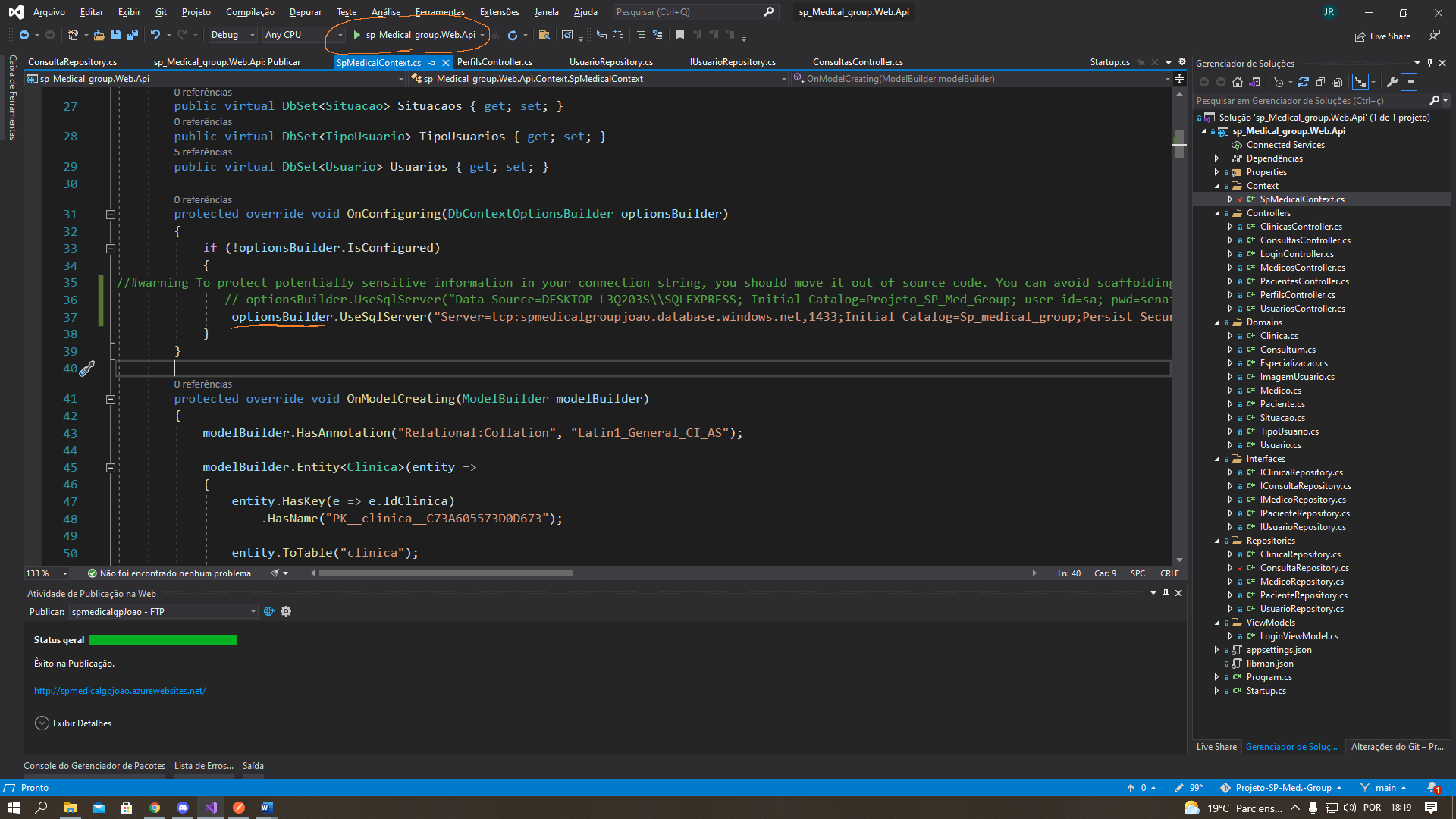


- Após abrir a pasta, abra o arquivo que esta especificando no final “.sln”.

-Após isso , espere o Visual Studio iniciar e carregar todos os arquivos.

-Ao abrir , vá ate a pasta context , abra o arquivo SpMedicalContext, e mude as informações que estiver “optionsBuilder”,e troque as informações que estiver nele, para as configurações da sua maquina.

-Após isso , rode a API:



-Após rodar a API, ela abrira o navegador padrão com a documentação em forma de Swagger.

-Abra o Postman, vá em “Import”,busque o link do Postman através dos seus arquivos, e teste com toda a felicidade do mundo.

## Funcionalidades

## Sistema Web

### Perfis de usuário:

### **1. Administrador:** Para o 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 (administrador,

## paciente ou médico);

## 2. O **administrador** poderá agendar uma consulta, onde será informado o paciente,

## data do agendamento e qual médico irá atender a consulta (o médico possuirá

## sua determinada especialidade);

## 3. O **administrador** poderá cancelar o agendamento;

## 4. O **administrador** deverá informar os dados da clínica (como endereço, horário

## de funcionamento, CNPJ, nome fantasia e razão social);

## 5. O **médico** poderá ver os agendamentos (consultas) associados a ele;

## 6. O **médico** poderá incluir a descrição da consulta que estará vinculada ao paciente

## (prontuário);

## 7. O **paciente** poderá visualizar suas próprias consultas;

## Sistema Mobile

### Perfis de usuário:

1. **Médico**: Colaboradores que atuam na área da saúde;

2. **Paciente**: Clientes da clínica;

### Funcionalidades:

1. O **paciente** poderá visualizar suas próprias consultas;

2. O **médico** poderá ver as consultas (os agendamentos) associados a ele;

# Front-end

Utilizei o React para a montagem do Front-end, utilizando Microsoft Visual Studio Code.

O React é uma biblioteca JavaScript de código aberto com foco em criar interfaces de usuário em páginas web. É mantido pelo Facebook, Instagram, outras empresas e uma comunidade de desenvolvedores individuais. É utilizado nos sites da Netflix, Imgur, Feedly, Airbnb, SeatGeek, HelloSign, Walmart e outros.

Alem disso usei outros tipos de linguagem para a construção do layout:

**HTML** - HTML abreviação para a expressão inglesa HyperText Markup Language, que significa: "Linguagem de Marcação de Hipertexto" é uma linguagem de marcação utilizada na construção de páginas na Web.

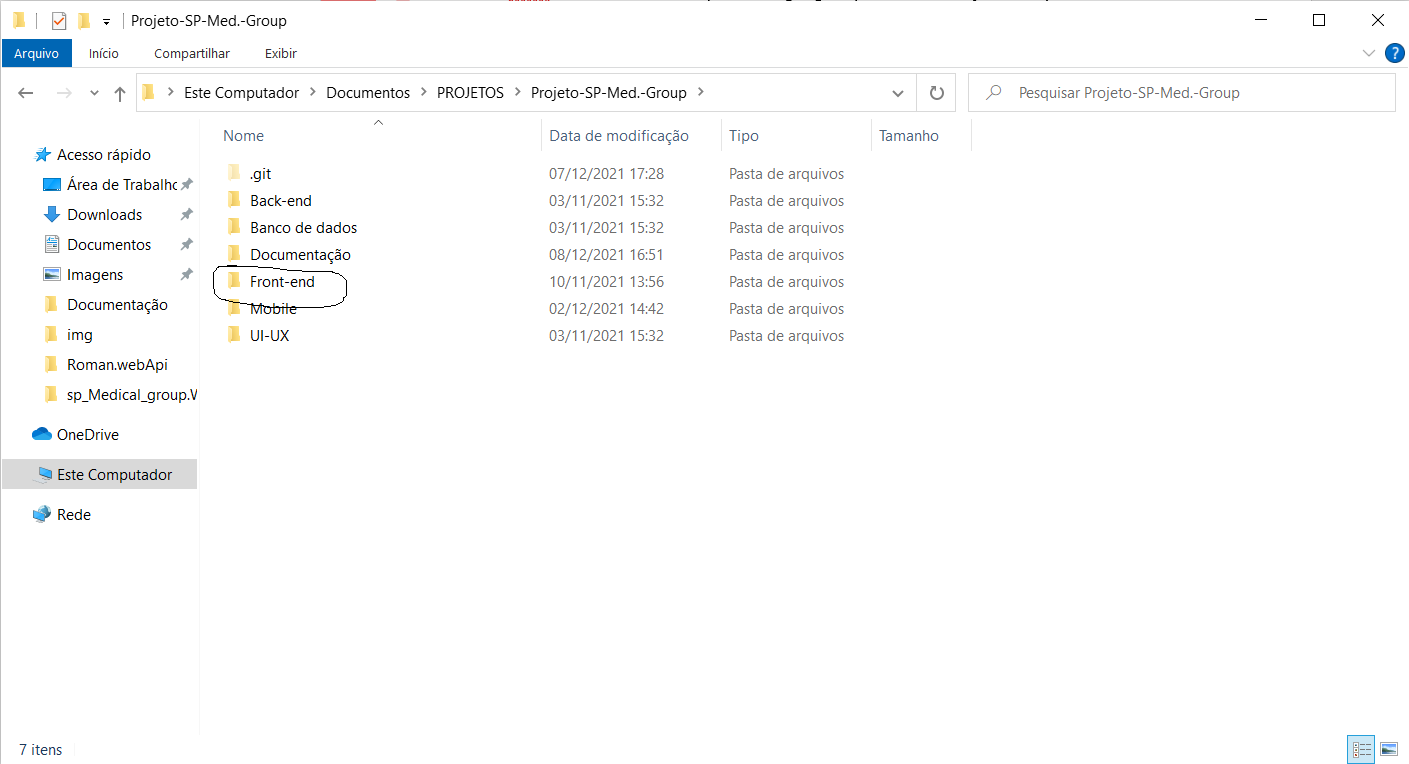
**CSS** - Cascading Style Sheets é um mecanismo para adicionar estilo a um documento web.

**JAVASCRIPT-** JavaScript é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web.

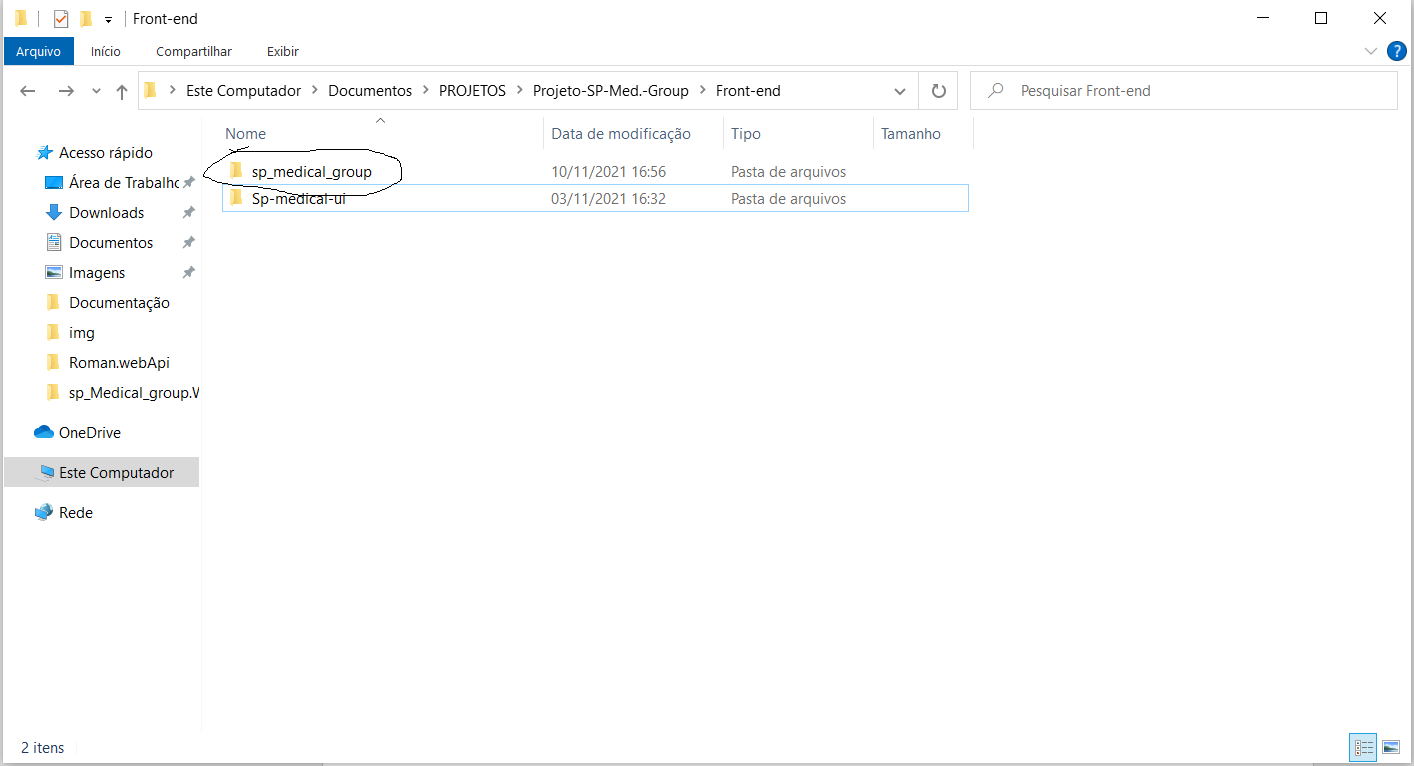
Passo a passo de como executar e testar o Front-end:

- Clona o repositório.

- Clique em Front-end.

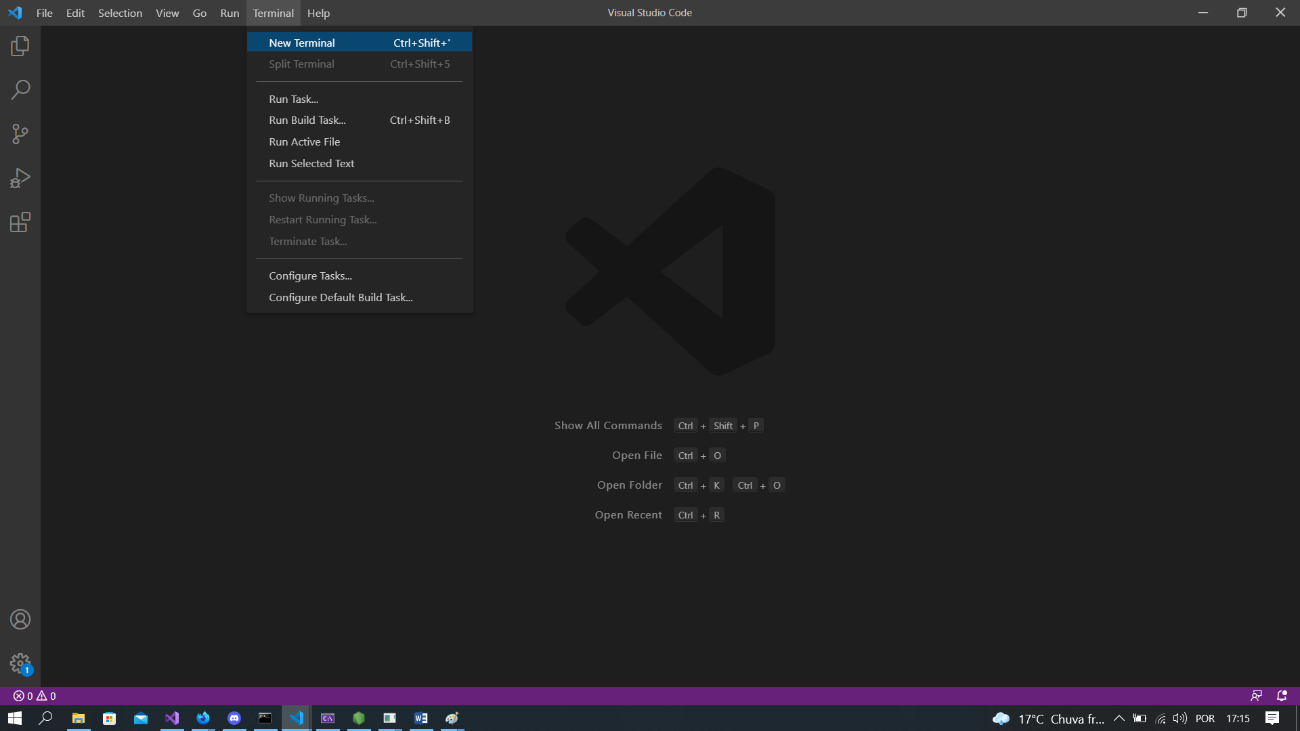


-Clique em sp\_medical\_group.



-Clique com o botão direito do mouse e abra com o Visual Studio Code.

-Ao abrir o Visual Studio Code abra um novo terminal



- Ao abrir um novo terminal, para rodar o projeto digite no terminal npm start.

- Após isso seu projeto abrira em uma nova janela do seu navegador, cujo a URL sera http//localhost:5000

-Não esqueça de rodar a API para que as funcionalidades possam funcionar.

-Com isso o seu projeto já estará em andamento.

### Perfis de usuário:

1. **Administrador**: Para o 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 (administrador,

paciente ou médico);

2. O **administrador** poderá agendar uma consulta, onde será informado o paciente,

data do agendamento e qual médico irá atender a consulta (o médico possuirá

sua determinada especialidade);

3. O **administrador** poderá cancelar o agendamento;

4. O **administrador** deverá informar os dados da clínica (como endereço, horário

de funcionamento, CNPJ, nome fantasia e razão social);

5. O **médico** poderá ver os agendamentos (consultas) associados a ele;

6. O **médico** poderá incluir a descrição da consulta que estará vinculada ao paciente

(prontuário);

7. O **paciente** poderá visualizar suas próprias consultas;

### Autenticação

Somente usuários autenticados deverão acessar as páginas restritas de sua aplicação.

### Autorização

Somente usuários autorizados poderão visualizar as suas próprias informações.

Exemplo: Um paciente poderá visualizar somente informações referentes a ele mesmo.

### Telas

### Administrador / Médico / Paciente

1. Login;

### Administrador

1. Cadastrar nova consulta;

2. Listagem de consultas;

### Médico

1. Listagem de consultas;

2. Inserir uma nova descrição na consulta;

### Paciente

1. Listagem de consultas;

# Mobile

Utilizei o React-Native para a montagem do App Android, utilizando Microsoft Visual Studio Code e Android Studio.

React Native é uma biblioteca Javascript criada pelo Facebook. É usada para desenvolver aplicativos para os sistemas Android e iOS de forma nativa.

Além disso usei alguns pacotes que o React-native oferece para termos uma usabilidade mais tranquila, como:

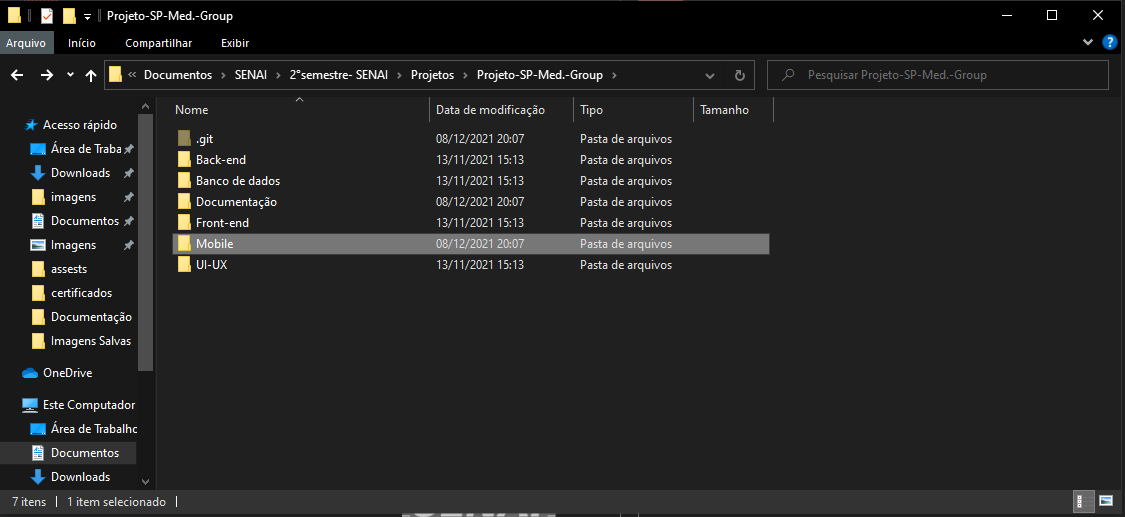
**React-navigation** - React Navigation é uma biblioteca popular para roteamento e navegação em uma aplicação React Native.

**React-native-async-storage-** **AsyncStorage** é uma API nativa do React Native, utilizada para armazenar dados persistentes no dispositivo. É uma forma de salvar dados no formato chave e valor . Os dados salvos com a API AsyncStorage, são asyncronos, com isso retornam um Promisse, e em caso de erro retorna um Error**.**

Passo a passo de como executar e testar o App Android:

- Clone o repositório.

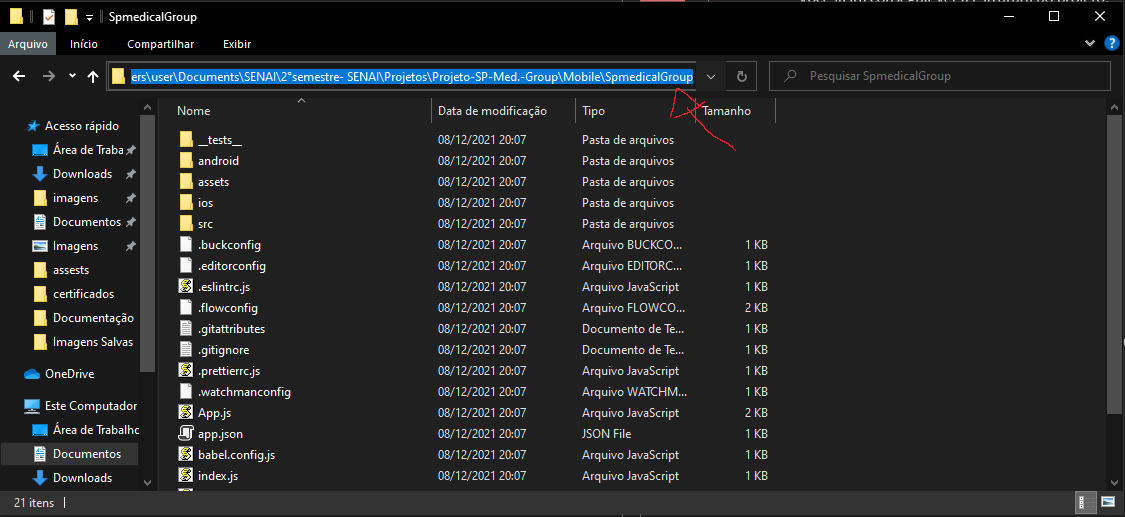
- Clique em Mobile na pasta.



- Abra a pasta SpmedicalGroup.

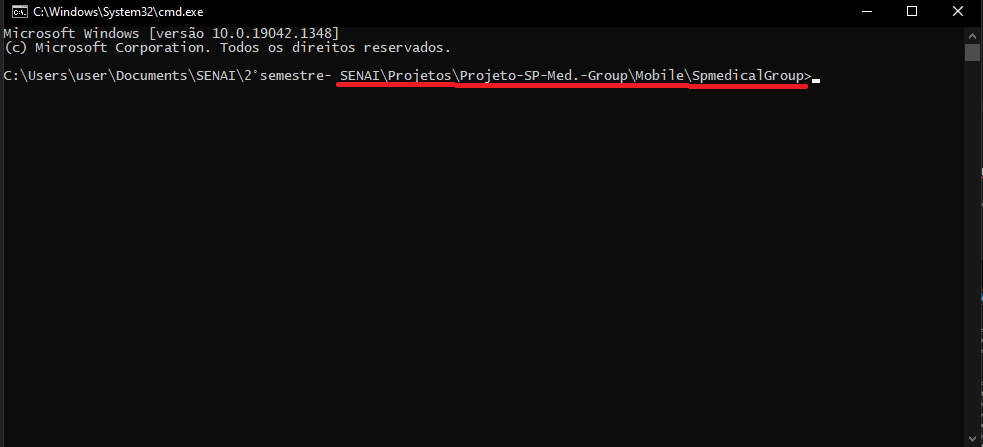
- Você já ira conseguir ver a estrutura do projeto.

- Clique na parte de cima ( no caminho da pasta ).



- Digite cmd e de enter logo em seguida.

- Abrira essa página.



- Confira se o caminho da pasta esta certo.

- Após isso digite npx react-native run-android e logo após de enter.

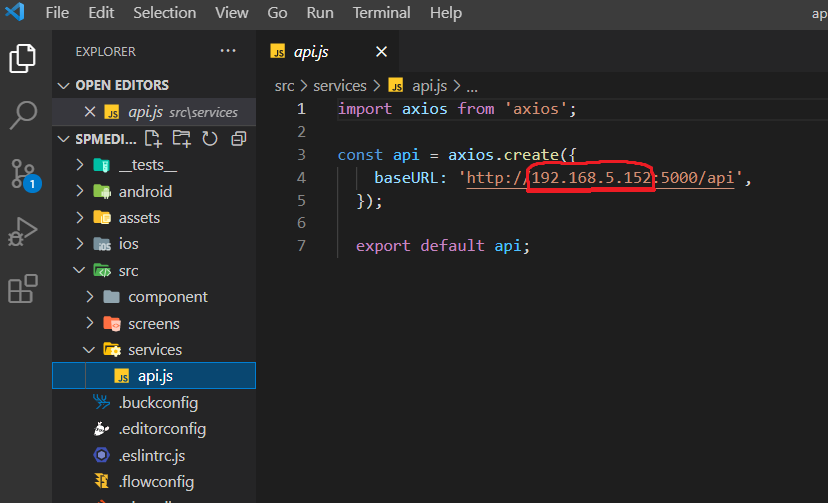
- Espere até o Android Studio abrir, e abrir o App.



- Não esqueça de rodar a Api e conferi o seu Banco de Dados

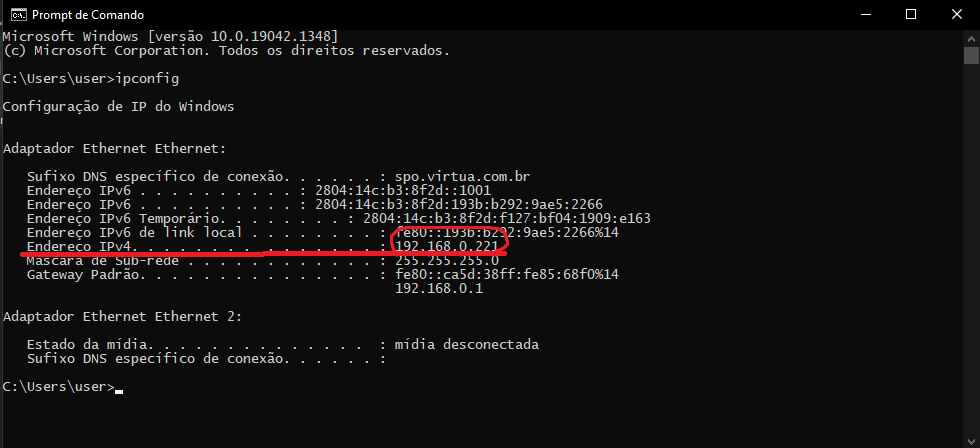
-Para que a Api rode no Android, vá na pasta src/sevices/api.js

- E verifique se o ip é o da sua maquina



-Para você saber seu ip, aperte a tecla Windows e digite Prompt de Comando.

- Digite “ipconfig” e de enter.



- Esse circulado em vermelho é o ip da maquina , substitua esse numero lá na pasta api.js, no lugar do numero que esta circulado em vermelho.

- A partir disso o App já poderá ser testado com todas as funcionalidades

### Perfis de usuário:

1. **Médico**: Colaboradores que atuam na área da saúde;

2. **Paciente**: Clientes da clínica;

### Funcionalidade:

1. O **paciente** poderá visualizar suas próprias consultas;

2. O **médico** poderá ver os agendamentos associados a ele;

Persistência de dados

Os dados referentes ao usuário logado na aplicação mobile deverão ser salvos

localmente.

## Telas

### Médico / Paciente

1. Login

### Médico

1. Visualizar as suas consultas

### Paciente

1. Visualizar as suas consultas

