**SP Medical Group**

Documentação

Sumário

[Sumário 2](#_Toc90360812)

[1. Resumo 3](#_Toc90360813)

[2. Descrição do projeto 3](#_Toc90360814)

[3. Banco de dados relacional 3](#_Toc90360815)

[4. Modelagem de dados 3](#_Toc90360816)

[Modelo Conceitual 3](#_Toc90360817)

[Modelo Lógico 4](#_Toc90360818)

[Modelo Físico 5](#_Toc90360819)

[Cronograma 6](#_Toc90360820)

[Trello 6](#_Toc90360821)

[5. Back-End 6](#_Toc90360822)

[Como testar a API: 7](#_Toc90360823)

[Funcionalidades 13](#_Toc90360824)

[6. Sistema Web 13](#_Toc90360825)

[Perfis de usuário: 13](#_Toc90360826)

[Funcionalidades: 13](#_Toc90360827)

[7. Sistema Mobile 14](#_Toc90360828)

[Perfis de usuário: 14](#_Toc90360829)

[Funcionalidades: 14](#_Toc90360830)

[8. NoSQL 14](#_Toc90360831)

# Resumo

Esse documento contém as informações referente a toda construção do projeto SP Medical Group, além de conceitos chave trazidos ao longo do semestre referentes a banco de dados relacionais, modelagens etc.

# Descrição do projeto

O projeto é para uma clínica médica a SP Medical Group, criada em 2020 que busca uma melhora na gestão, que antes era realizada com planilhas digitais, e agora deve ser realizada com a incrementação de uma aplicação web e mobile, eles necessitam de um sistema que seja possível ter fácil acesso aos dados, melhorando assim a gestão do negócio.

# Banco de dados relacional

São basicamente uma coleção de tabelas cada uma armazenando seus dados de maneira organizada com linhas e colunas, tendo uma importância fundamental no sistema para possíveis consultas utilizadas em campos como login etc.

Os Bancos Relacionais são os bancos de dados onde temos relações entre as tabelas, não necessariamente entre todas, mas algumas.

# Modelagem de dados

É definir como os dados se relacionaram, como vai funcionar o seu banco, suas regras de negócio etc.

Para isso são utilizados diversos diagramas sendo eles os modelos conceituais e logico por exemplo.

## Modelo Conceitual

O Modelo Conceitual é o diagrama inicial que nos mostra as relações entre as entidades, sem ter características e informações das colunas.

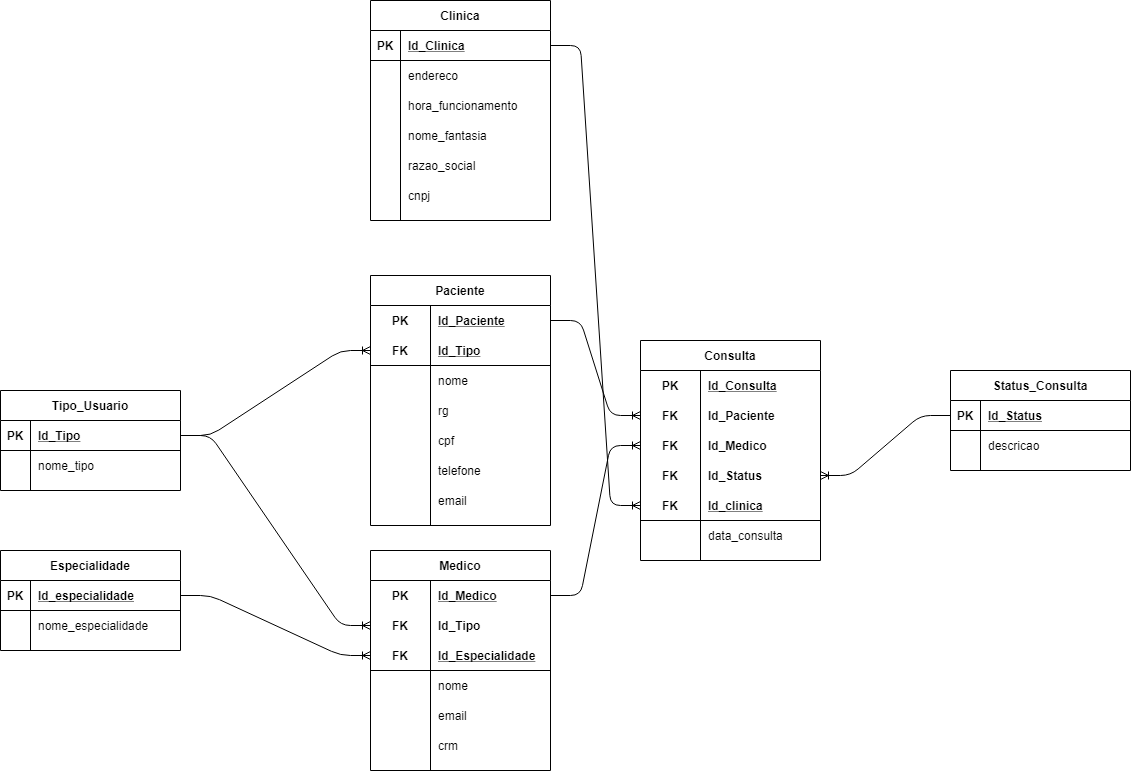
Diagrama

Descrição gerada automaticamente

Nesse caso aqui temos o modelo das possíveis entidades e suas relações do projeto SP Medical Group, trazendo assim possíveis questionamentos de organização para o sistema.

## Modelo Lógico

O Modelo Lógico vai pegar o Conceitual e adequar ao tipo de banco que será usado, colocando as colunas que vamos usar, fazendo as relações com as presenças de PK’s e FK’s



Como vemos aqui aquele modelo conceitual agora ganha a presença colunas com as informações que devem entrar no banco e montamos de maneira um tanto quanto mais fiel a que acontecera no modelo final.

## Modelo Físico

Ele vai nos mostrar o Banco de Dados de maneira física com alguns dados de exemplo para testar os tipos de relação e tudo mais.

Diagrama

Descrição gerada automaticamente

Aqui vemos o Modelo Fisico deste projeto ele foi gerado pelo SSMS e mostra de maneira ainda mais fiel o banco, ele traz o banco como algo existente mostrando assim suas relações e se parecendo até com o modelo lógico em sua base.

## Cronograma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 5 |
| Criação do Trello | X |  |  |  |  |
| Criação do Repositório do Github | X |  |  |  |  |
| Modelo Conceitual | X |  |  |  |  |
| Modelo Lógico | X |  |  |  |  |
| Modelo Físico | X | X |  |  |  |
| Script DDL |  | X |  |  |  |
| Importação dos dados para o banco |  | X |  |  |  |
| Script DQL |  | X |  |  |  |
| Documentação | X | X |  |  |  |

### Trello

<https://trello.com/b/62c6Ms6a/sp-med-group>

# Back-End

O sistema foi desenvolvido no Microsoft Visual Studio utilizando a tecnologia de **API**, onde nela foram definidos os métodos e as funções que a aplicação pode executar, que foram pedidas previamente no **StoryTeling** do projeto.

Além disso foram configuradas as requisições do projeto para enfim retornar no formato **JSON** as informações especificadas, seguindo os protocolos **HTTP,** de tal maneira estas respostas podem ser utilizadas por aparelhos móveis e aparelhos de desktop (computadores, notebooks e afins).

Dentro desta API utilizamos diversos **Pacotes do Nuget**, que trazem novas possibilidades de código a serem construídas, assim simplificando e deixando o código mais limpo.

Em relação a estes parâmetros de simplificação de código utilizamos uma **ORM** um tipo de **Framework** que neste caso específico foi o **Entity Framework**, que encurtou a construção dos métodos para pequenas linhas simples.

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

**StoryTeling** – Escopo do projeto aonde vem todos as coisas que devemos alcançar na sua construção.

**JSON** – Notação de Objeto JavaScript – Resposta das Requisições

**HTTP** – Protocolo de Transferencia de HiperTexto

**Pacotes do Nuget** – Pacote de Extensões do Visual Studio

**ORM** – Mapeamento de um Objeto Relacional – Facilita o contato entra a Aplicação e o Banco de Dados.

**Frameworks** – São conjuntos de códigos prontos que facilitam a construção dos métodos.

**Entity Framework –** Tipo de Framework utilizado no projeto.

## **Como testar a API:**

Forma, Retângulo

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente1° Passo: Deve se verificar se todos os programas estão instalados sendo eles o Postman e o Visual Studio

2° Passo: Agora com todos os programas instalados, abra a pasta do projeto (pasta destaca na primeira imagem) entre na pasta de Back-End e vá até a solução do projeto (o arquivo cujo final é .sln) e o inicialize.

Tela de computador com ícones coloridos

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

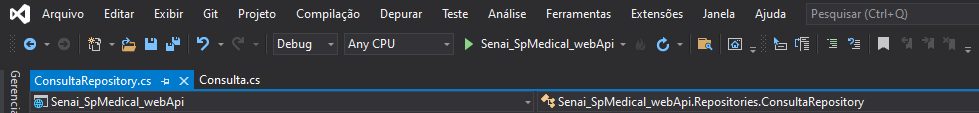
3° Passo: Agora com o arquivo aberto vá até IIS Express clique na seta ao lado e mude para Senai\_SpMedical\_webApi.

Tela de celular

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente



4º Passo: Agora com tudo selecionado clique sobre a seta verde e a aplicação começara a rodar, com isso abrira uma janela no navegador com a explicação dos métodos (Primeiro Print Subsequente) e uma janela do Prompt de Comando (Segundo Print Subsequente).

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

5° Passo: Agora com a API em Execução abra o Postman e importe a coleção das requisições.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

6° Passo: Agora você devera ou arrastar e soltar o arquivo da pasta Postman do projeto até o quadrado de importação ou selecionar Upload Files e buscar ele na pasta.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

7° Passo: Agora com a coleção importada você poderá testar método a método, alguns métodos necessitam de login então você deverá colocar o e-mail já cadastrado no banco no método de login e gerar um novo token e colocá-lo nas áreas de autenticação com a opção de Bearer Token selecionada.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Uma imagem contendo Texto

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Os métodos podem retornar diferente StatusCode e abaixo terá uma tabela com a funcionalidade de cada um.

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Os Status Codes de 200 até 204 significam que está tudo funcionando corretamente.

Os Status Codes de 400 até 409 significa que ocorreram erros de execução, dentre eles o 401 e o 403 respectivamente são caso você não esteja logado, e o outro mostra que você não tem permissão para acessar aquele método.

E Status Codes de level 500 normalmente são erros internos dentro da construção da própria API, que devem ser corrigidos para seu funcionamento.

## **Funcionalidades**

A seguir estão as funcionalidades presentes no projeto tanto no Sistema Web quanto no Mobile:

# Sistema Web

### **Perfis de usuário:**

1. **Administrador**: Área administrativa da Clínica;
2. **Médico**: Funcionário do Hospital que atua na área da Saúde;
3. **Paciente**: Cliente 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 **administrador** poderá cadastrar, deletar e atualizar novas clínicas;
6. O **administrador** poderá cadastrar novos usuários, além de deletá-los e atualizá-los;
7. O **administrador** poderá cadastrar novos pacientes e novos médicos no sistema;
8. O **administrador** poderá cadastrar, deletar e atualizar novas especialidades;
9. O **médico** poderá ver os agendamentos (consultas) associados a ele;
10. O **médico** poderá incluir a descrição da consulta que estará vinculada ao paciente (prontuário);
11. O **paciente** poderá visualizar suas próprias consultas;
12. **Todos os Usuários** poderão cadastrar fotos;

# Sistema Mobile

### **Perfis de usuário:**

1. **Administrador**: Área administrativa do Hospital;
2. **Médico**: Médicos do Hospital;
3. **Paciente**: Um paciente que está nos dados do Hospital.

### **Funcionalidades:**

1. O **paciente** poderá visualizar suas próprias consultas;
2. O **médico** poderá ver as consultas (os agendamentos) associados a ele;

# NoSQL