**SP Medical Group**

Documentação

Sumário

[1. Resumo 3](#_Toc83892249)

[2. Descrição do projeto 3](#_Toc83892250)

[3. Banco de dados relacional 3](#_Toc83892251)

[4. Modelagem de dados 3](#_Toc83892252)

[Modelo Conceitual 4](#_Toc83892253)

[Modelo Lógico 4](#_Toc83892254)

[Modelo Físico 5](#_Toc83892255)

[5. Back-End 6](#_Toc83892256)

[Funcionalidades 7](#_Toc83892257)

[Perfis de usuário: 7](#_Toc83892258)

[Funcionalidades: 7](#_Toc83892259)

[Autenticação / Autorização: 8](#_Toc83892260)

[6. Cronograma 8](#_Toc83892261)

[Trello 9](#_Toc83892262)

[gitHub 9](#_Toc83892263)

# Resumo

Este documento tem por objetivo demonstrar os resultados da aplicação prática de conhecimentos desenvolvidos na sprint 1 do curso técnico de Desenvolvimento de Sistemas. Tal conteúdo diz respeito a utilização de banco de dados relacionais, por meio da ferramenta SQL Server.

# Descrição do projeto

O projeto proposto chama-se SP Medical Group. Trata-se da criação de uma estrutura completa de scripts que contemplam os comandos pertinentes à DDL – Linguagem de Definição de Dados; DML – Linguagem de Manipulação de Dados e DQL – Linguagem de Consulta de Dados, todas elas utilizando a linguagem SQL – Linguagem de Consulta Estruturada. Inicialmente, o planejamento da organização e relacionamento dos dados em suas respectivas tabelas é feito através das modelagens conceitual, lógica e física, que serão abordadas mais profundamente adiante.

O objetivo principal destas tarefas é proporcionar a criação de um banco de dados capaz de armazenar, consultar e alterar registros pertencentes a uma clinica médica: SP Medical Group. Neste banco de dados, são manipuladas informações pertinentes aos pacientes da clínica, aos médicos que nela trabalham, aos dados destas pessoas e das consultas realizadas entre elas. Também são registradas informações sobre os usuários de um sistema informatizado que será continuamente desenvolvido ao longo do semestre letivo, onde a sprint “Banco de Dados é apenas a primeira”, dentre outras.

# Banco de dados relacional

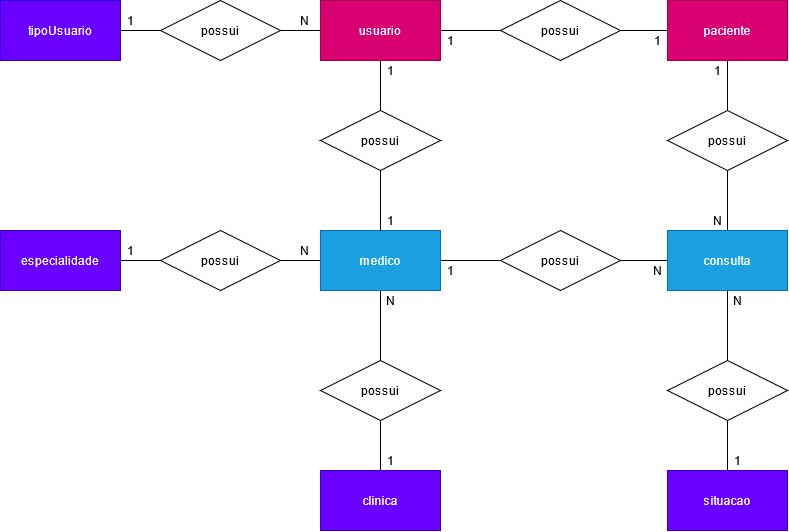
Podemos definir e entender um Bancos de dados como um conjunto de tabelas onde os dados são armazenados. Por dados, podemos entender a menor parte das informações, ou seja, suas unidades básicas. Quando agrupamos dados de tal forma que eles passam a representar um significado dentro de um determinado contexto, temos então informações úteis sobre pessoas, objetos, ações e diversos outros elementos do mundo real. No caso dos bancos de dados relacionais, há um vínculo entre as tabelas que contêm os dados. Ou seja, elas estão interligadas por dependências umas das outras (o que difere o sistema dos Bancos de Dados não-relacionais - NoSQL, onde não há tal necessidade), de maneira que formam estruturas bastante organizadas onde a correlação dos dados compõe e representam as informações, podendo serem armazenadas, acessadas e alteradas de maneira segura e eficiente.

# Modelagem de dados

A modelagem de dados representa a fase inicial do desenvolvimento de um banco de dados. Trata-se do planejamento de suas estruturas, da definição de entidades (tabelas contendo dados de um elemento do mundo real), atributos (cada célula correspondente as tabelas) e seus relacionamentos, suas conexões e interligações. É uma fase crucial para o desenvolvimento dos bancos de dados relacionais, pois dela depende toda a eficiência que teremos ao manipular as informações. Baixo veremos os três tipos de modelagem que compõe o desenvolvimento de um banco de dados.

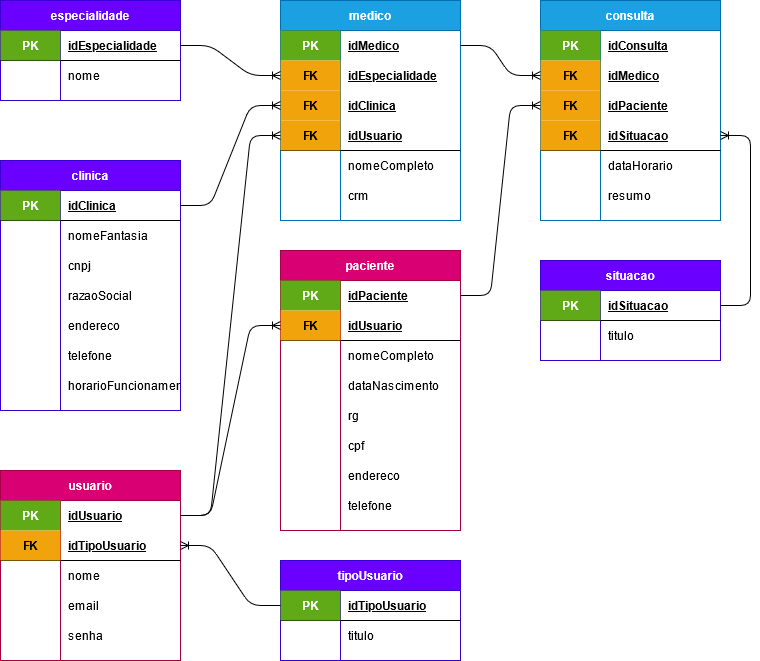
## Modelo Conceitual

No modelo conceitual, definimos as entidades que irão compor o banco de dados. Cada entidade irá se transformar em uma tabela. Neste modelo também definimos as relações de cardinalidade entre as entidades. Elas representam a quantidade de relacionamentos que cada registro pode assumir com outras tabelas (0, 1 ou N - muitos), representando assim parte da conexão entre os dados. No caso do projeto SP Medical Group, o modelo conceitual foi configurado de acordo com a figura abaixo:



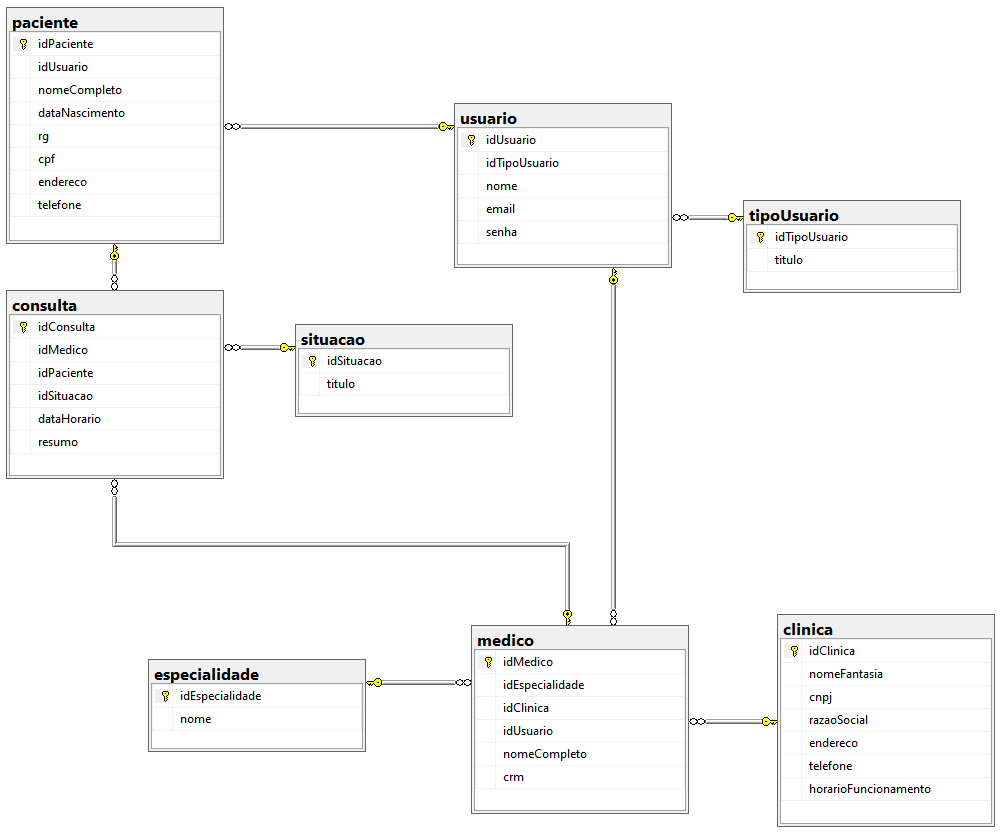
## Modelo Lógico

A partir do modelo conceitual, damos sequência ao planejamento com o desenvolvimento do modelo lógico, onde incluímos mais detalhes das entidades. São inseridas as colunas das tabelas que correspondem a dados relevantes identificados no escopo do trabalho. Também são definidas as relações entre as chaves primárias (PK) e chaves estrangeiras (FK), elementos que funcionam como identificadores de cada tabela, permitindo seu relacionamento e mantendo a unicidade dos dados registrados em cada uma delas:



## Modelo Físico

Por fim, dentre as etapas de modelagem, temos o modelo físico. Já tendo definido toda a estrutura lógica que representa os relacionamentos entre as tabelas, é no modelo físico que inserimos os dados a serem importados para o banco de dados. É uma etapa importante, pois permite visualizar a base de todos como um todo, ajudando a definir características como os tipos de dados que cada campo irá representar nas tabelas, algo que será feito já no SGBD – Sistema Gerenciador de Banco de Dados, por meio da interface gráfica SSML – SQL Server Management Studio. A imagem abaixo demonstra a estrutura relacional extraída diretamente do SSML, correspondendo ao modelo físico cuja representação consta em arquivo .xlsx, anexo.



# Back-End

Uma vez tendo sido desenvolvido o Banco de Dados onde as informações da aplicação serão armazenadas, chegamos ao processo de elaboração do sistema principal onde encontram-se as regras de negócio da aplicação. Ele corresponde a uma API - *Application Programming Interface* (Interface de Programação de Aplicativos), um conjunto de padrões e instruções estabelecidos para a interação de softwares, definindo as requisições e as respostas pelas quais se comunicam através do protocolo HTTP, mais especificamente aplicado segundo o padrão REST - *Representational State Transfer*. Neste caso, a entrada e saída de dados é expressa por meio do formato JSON (*JavaScript Objet Notation,* Notação de Objeto JavaScript), permitindo que seja possível acessar o sistema em diversos dispositivos distintos sem a preocupação com a linguagem utilizada por cada um deles. Para o desenvolvimento da aplicação, foi utilizada a IDE (Integrated Development Enviroment, Ambiente de Desenvolvimento Integrado) Microsoft Visual Studio, na qual o framework .NET pode ser integrado a diversas ferramentas, caso da persistência de dados realizada no Banco de Dados SQL Server, abordado anteriormente.

Para testar a aplicação, é possível baixar todo o código disponível nos links ao final deste documento. Uma vez clonado o repositório do GitHub, basta que seja acessado o arquivo de solução no formato .sln que encontra-se na pasta “senai.spmedgroup.webApi”, utilizando o Visual Studio. Ao rodar a aplicação, já integrada ao banco de dados, temos a possibilidade de executar sua utilização localmente no computador, acessando os *endpoints* disponíveis através da documentação Swagger, uma ferramenta que automaticamente é disponibilizada ao usuário no momento em que o sistema inicia sua operação. Os endpoints nada mais são do que as rotas, as URLs que o usuário acessa para se comunicar com a API. Cada endpoint tem sua sintaxe detalhada pelo Swagger, de modo a esclarecer ao usuário as funcionalidades disponíveis na API. Para inserir e visualizar os dados, deve ser utilizado o software Postman, uma interface que simula o ambiente FrontEnd pelo qual o usuário final teria acesso a API, caso tivéssemos essa aplicação disponivel. As principais funcionalidade desenvolvidas serão conceituadas a seguir.

Outra forma de acesso é através do serviço de hospedagem da Azure, plataforma em nuvem fornecida pela Microsoft para que toda a API e o banco de dados sejam armazenados em servidores on-line e permaneçam disponíveis publicamente. No entanto, por limitações quanto a custos de hospedagem, essa funcionalidade teve apenas o caráter didático no projeto SPMedicalGroup, que permanece disponível no GitHub, mas está inativo na nuvem.

## Funcionalidades

Sendo o SPMedicalGroup um sistema para gerenciamento de consultas, podemos pensar nas funcionalidades com uma interação entre os usuários e suas ações. Ou seja, os usuários se subdividem em 3 categorias: administrador; médico e paciente. Cada uma delas possui diferentes permissões de acesso aos serviços da API, que basicamente envolvem a administração e agendamento de consultas entre os profissionais médicos e os pacientes, integrando ainda todos eles as clinicas onde serão realizadas as consultas. Tanto os usuários como as clinicas e as consultas possuem informações especificas, dados de identificação que são indispensáveis, como nomes, endereços, datas, e tudo isso é organizado e gerenciados pelo sistema, seguindo algumas premissas principais, que são:

### Perfis de usuário:

1. **Administrador**: para o colaborador da área administrativa da clínica;
2. **Médico**: colaboradores que atuam na área da saúde;
3. **Pacientes**: 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 / Autorização:

Para acessar os recursos disponibilizados pela API, todos os usuários devem possuir cadastro no sistema e realizar login através de seu e-mail e senha pessoais. A autenticação é feita utilizando o modelo JWT (JSON Web Token), permitindo que padrões de segurança sejam incorporados aos procedimentos e definindo os níveis de acesso de cada pessoa as informações e serviços.

# Cronograma

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 1 | Dia 2 | Dia 3 | Dia 4 |
| Modelo Conceitual | X |  |  |  |  |  |  |  |
| Modelo Lógico | X | X |  |  |  |  |  |  |
| Modelo Físico |  | X |  |  |  |  |  |  |
| DDL |  | X |  |  |  |  |  |  |
| DML |  | X | X |  |  |  |  |  |
| DQL |  |  | X |  |  |  |  |  |
| Documentação |  |  |  | X |  |  |  |  |
| Solução VS |  |  |  |  | X |  |  |  |
| Pacotes NuGet |  |  |  |  | X |  |  |  |
| StartUp.cs |  |  |  |  | X |  |  |  |
| Scaffold(DBFirst) |  |  |  |  | X |  |  |  |
| Domains |  |  |  |  |  | X |  |  |
| Interfaces |  |  |  |  |  | X |  |  |
| Repositories |  |  |  |  |  | X |  |  |
| Controllers |  |  |  |  |  | X | X |  |
| Autenticação |  |  |  |  |  |  | X |  |
| Postman |  |  |  |  |  |  | X |  |
| Swagger |  |  |  |  |  |  |  | X |
| Hospedagem |  |  |  |  |  |  |  | X |
| Documentação |  |  |  |  |  |  |  | X |

### Trello

<https://trello.com/b/Znamzy7U/sp-med-group>

<https://trello.com/b/GRzLLJMl/spmedgroupsprint2backend>

### gitHub

<https://github.com/samuel-gHub/SENAI_2/tree/main/SP%20MEDICAL%20GROUP>