

Documentação

SenaiMEDGROUP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Alameda Barão de Limeira, 539 – São Paulo/SP |  |  |  | sp.senai.br |

Sumário

1. [Resumo 2](#_Toc533767843)

[Objetivos](#_Toc533767844)

1. [Descrição do projeto 2](#_Toc533767845)

[Resumo do projeto](#_Toc533767846)

1. [Modelagem de Software 3](#_Toc533767847)

[Modelo Lógico](#_Toc533767848)

[Modelo Físico](#_Toc533767849)

[Modelo Conceitual](#_Toc533767850)

[Cronograma](#_Toc533767851)

1. [Funcionalidades 5](#_Toc533767852)

[Web](#_Toc533767853)

[Mobile](#_Toc533767854)

1. [Protótipos 6](#_Toc533767855)

[Web](#_Toc533767856)

[Mobile](#_Toc533767857)

1. [Front-End 7](#_Toc533767858)
2. [Mobile 8](#_Toc533767859)
3. [Arquitetura do Projeto 9](#_Toc533767860)
4. [Referências 10](#_Toc533767861)

[Links](#_Toc533767862)

[Livros](#_Toc533767863)

Resumo

Objetivos

O seguinte documento tem como objetivo registrar a criação do projeto do Senai: “SenaiMEDGROUP”. Neste documento se apresenta as especificações técnicas, capacidades, etc do projeto acima mencionado.

Descrição do projeto

O projeto se refere a um sistema de dados para a área de saúde/clínicas/medicina para aprendizado de banco de dados, API, REACT, etc. Através dessas funcionalidades será possível cadastrar médicos, pacientes, clínicas, consultas, etc. Podendo ser acessado através da Web e programado usando as capacidades FRONT END, BACK END e MOBILE.

Resumo do projeto

Um projeto de sistema para cadastros de usuários, serviços, localidades disponíveis em vários ambientes, seja web, seja mobile.

Modelagem de Software

Modelo Lógico

Tabelas apresentadas com suas chaves primárias e estrangeiras

Chaves primárias e estrangeiras de acordo com as entidades/tabelas apresentadas.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Consulta | | | | | |  |  |
| ID | CNPJ | Endereco | Data\_Inicio | Razao\_Social | Nome\_Fantasia | Horario\_Funcionamento | Id\_Medico |
| 1 | 321321321 | Alameda Barão de Limeira, 539 - Santa Cecilia, São Paulo - SP, 01202-001 | 2019-02-11 | SP Medical Group | SP Medical Group | SP Medical Group | 7:00  As  19:00 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Usuários | | | | | | |  |  |
| ID | Nome | CPF | RG | Telefone\_Contato | Email | Senha | ID\_Tipo\_Usuario | Data\_Nascimento |
| 1 | Ricardo | 8640090230 | 8640090230 | (11) 986400230 | ricardo@email.com | 123 | 1 | 2019 |
| 2 | Roberto | 8640090230 | 8640090230 | (11) 986400230 | roberto@email.com | 123 | 1 | 2019 |
| 3 | Helena | 8640090230 | 8640090230 | (11) 986400230 | helena@email.com | 123 | 1 | 2019 |
| 4 | Ligia | 94839859000 | 435225435 | (11) 986400230 | ligia@email.com | 123 | 2 | 2019 |
| 5 | Alexandre | 73556944057 | 326543457 | (11) 986400230 | alexandre@email.com | 123 | 3 | 2019 |

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo\_Médicos | | | |
| ID | CRM | ID\_Area\_Clinica | Id\_Usuario |
| 1 | 54356-SP | 2 | 15 |
| 2 | 53452-SP | 17 | 16 |

|  |  |
| --- | --- |
| Tipos de Usuário | |
| ID | Tipo |
| 3 | Administrador |
| 1 | Médico |
| 2 | Paciente |

|  |  |
| --- | --- |
| Status\_Consulta | |
| ID | Status |
| 1 | Agendada |
| 3 | Cancelada |
| 2 | Realizada |

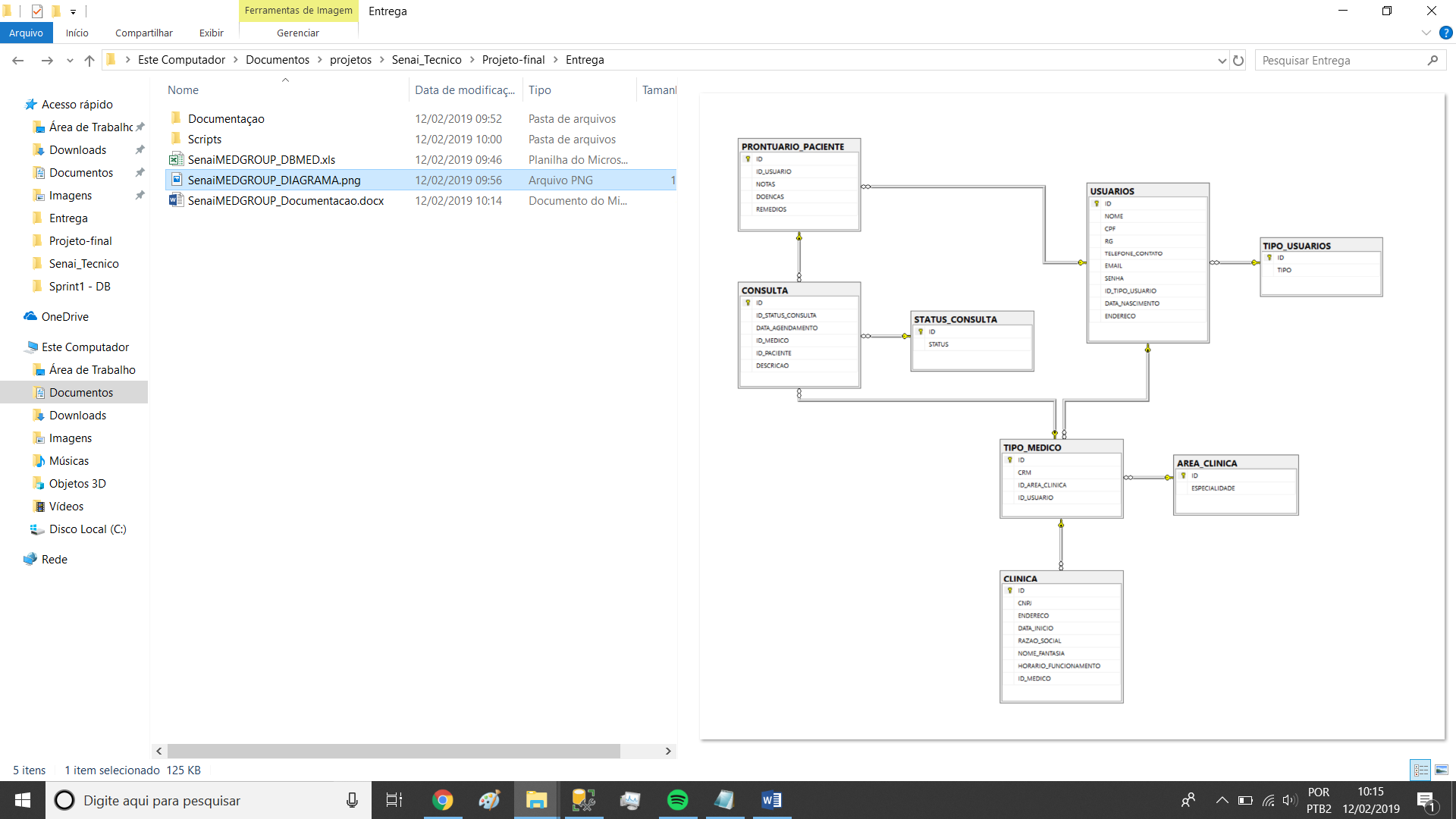
|  |  |
| --- | --- |
| Area\_Clinica | |
| ID | Nome |
| 1 | Psiquiatra |
| 2 | Cardiologista |
| 3 | Endocrinologista |
| 4 | Nutricionista |
| 5 | Pediatra |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prontuario\_Paciente | | | | | |
| ID | ID Usuário | Notas | Doencas | Remedios |
| 1 | 1 | 18 | EBOLA | CHAZINHO |
| 2 | 2 | 19 | ESTRESSE | CHAZINHO |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Consultas | | | | | |
| ID | ID Paciente | ID Médico | Data da consulta | Observações | ID Status |
| 1 | 1 | 2 | 12/04/2019 | Foi receitado medicamento x... | 2 |
| 2 | 2 | 1 | 05/06/2019 | Precisa ser medicado diariamente. | 1 |

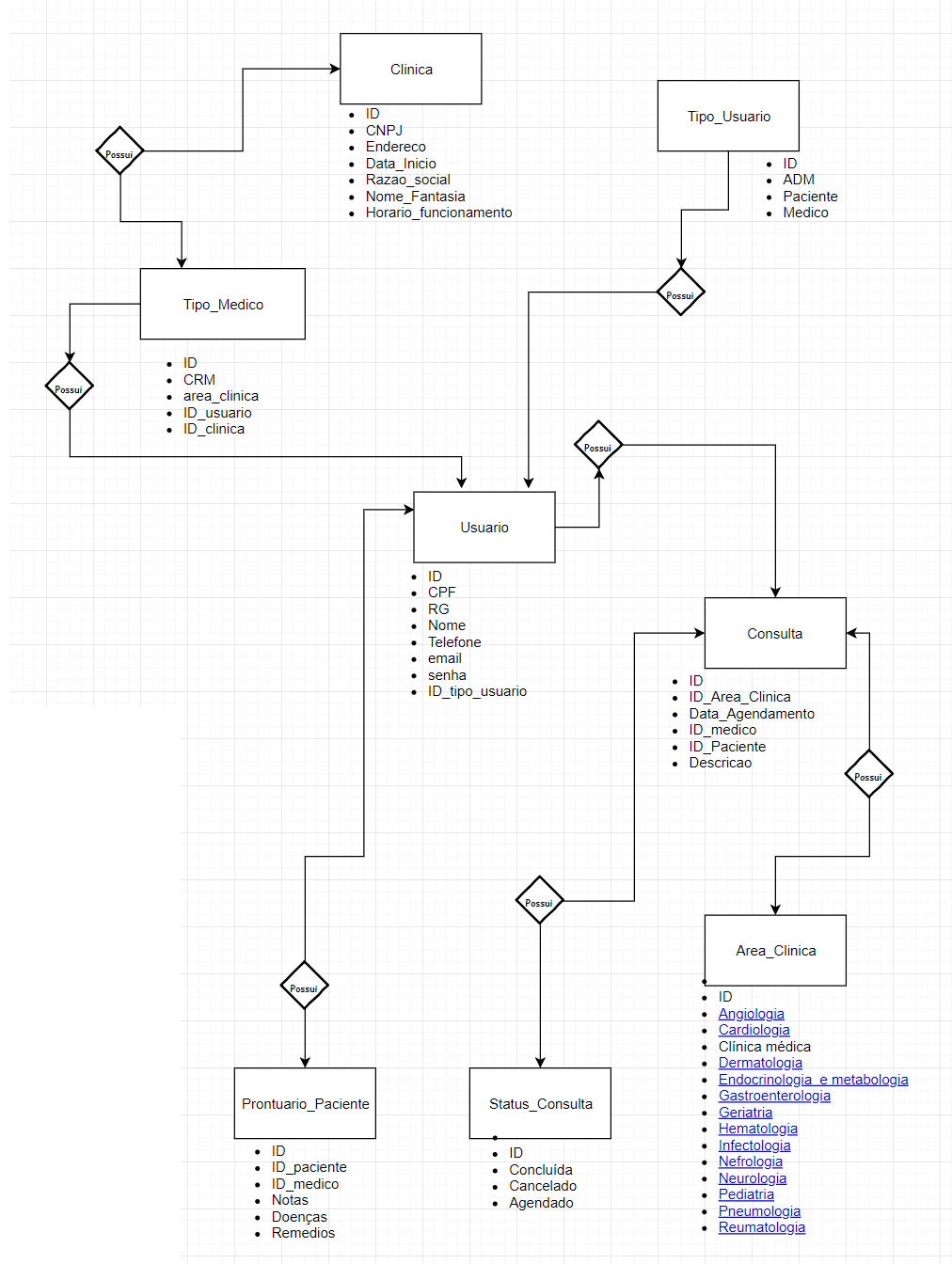
Modelo Físico

Diagrama do Banco de Dados expondo a forma com que os dados estão organizados



Modelo Conceitual

O banco de dados é composto por 8 tabelas, interligadas pela tabela de usuários que possui informações que serão utilizadas para dar base as tabelas Tipo\_Medico que especifica o usuário médico e Prontuario\_Paciente que especifica o usuário paciente. Assim as tabelas Clinica e Consulta podem ser inseridas se ligando com médico e pacientes respectivamente



Cronograma

|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 5 | Dia 6 | Dia 7 | Dia 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Modelo Lógico** | x |  |  |  |  |  |  |  |
| **Modelo Físico** | x |  |  |  |  |  |  |  |
| Modelo Conceitual | x |  |  |  |  |  |  |  |

Funcionalidades

Web

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 vinculado ao paciente (prontuário);

7. O paciente poderá visualizar suas próprias consultas;

Mobile

1. O paciente poderá visualizar suas próprias consultas;

2. O paciente poderá visualizar a rota entre sua localização e o endereço da clínica;

3. O médico poderá ver os agendamentos associados a ele;

4. Notificação: No momento que o administrador cadastrar um agendamento, o médico deverá receber uma notificação pelo aplicativo;

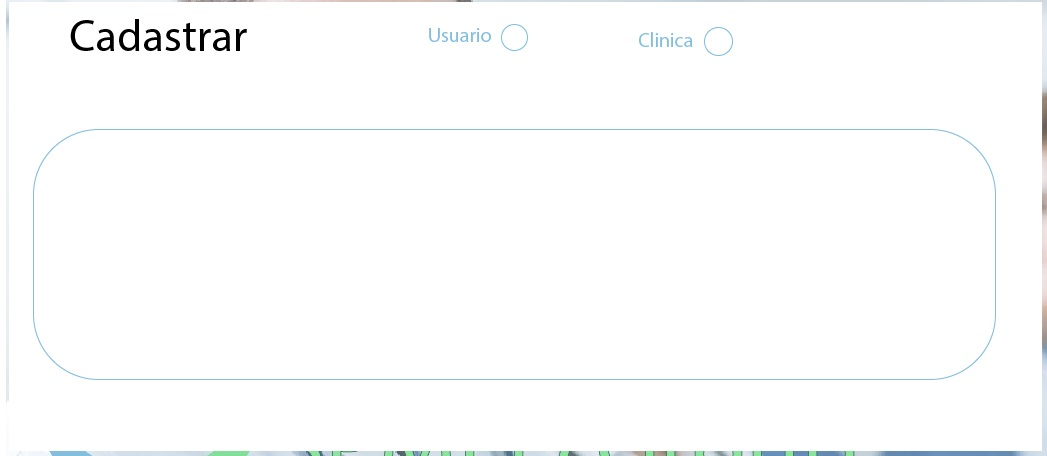
Protótipos

Web

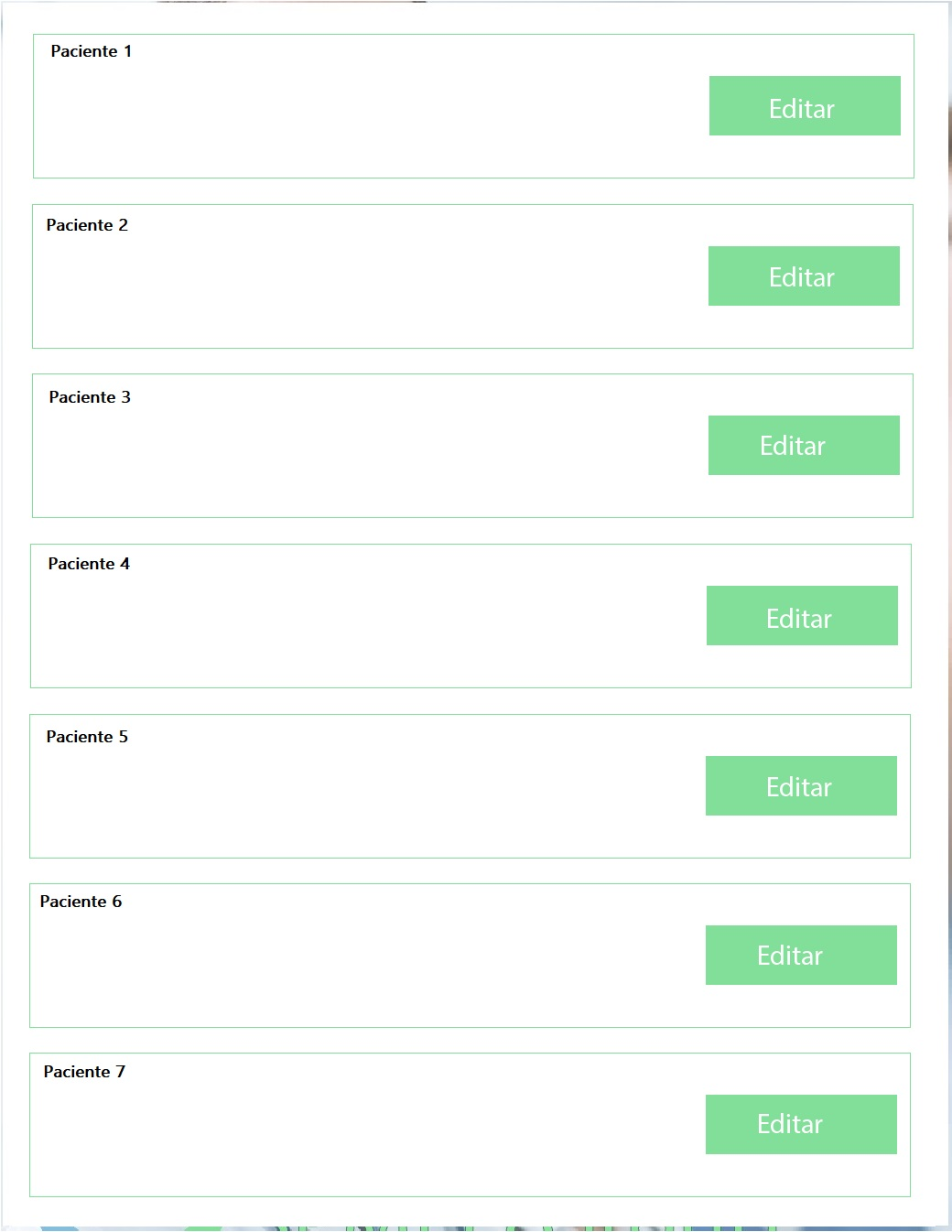
Homepage



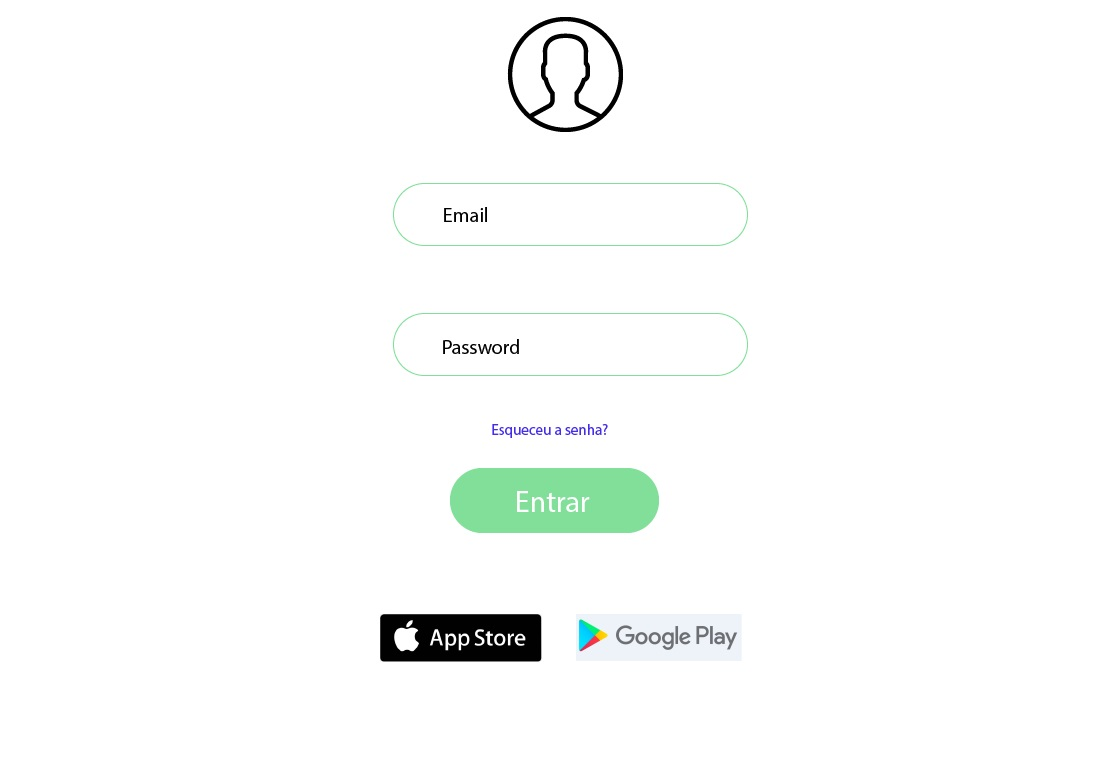
Página de Cadastro



Página de Listagem de Consultas

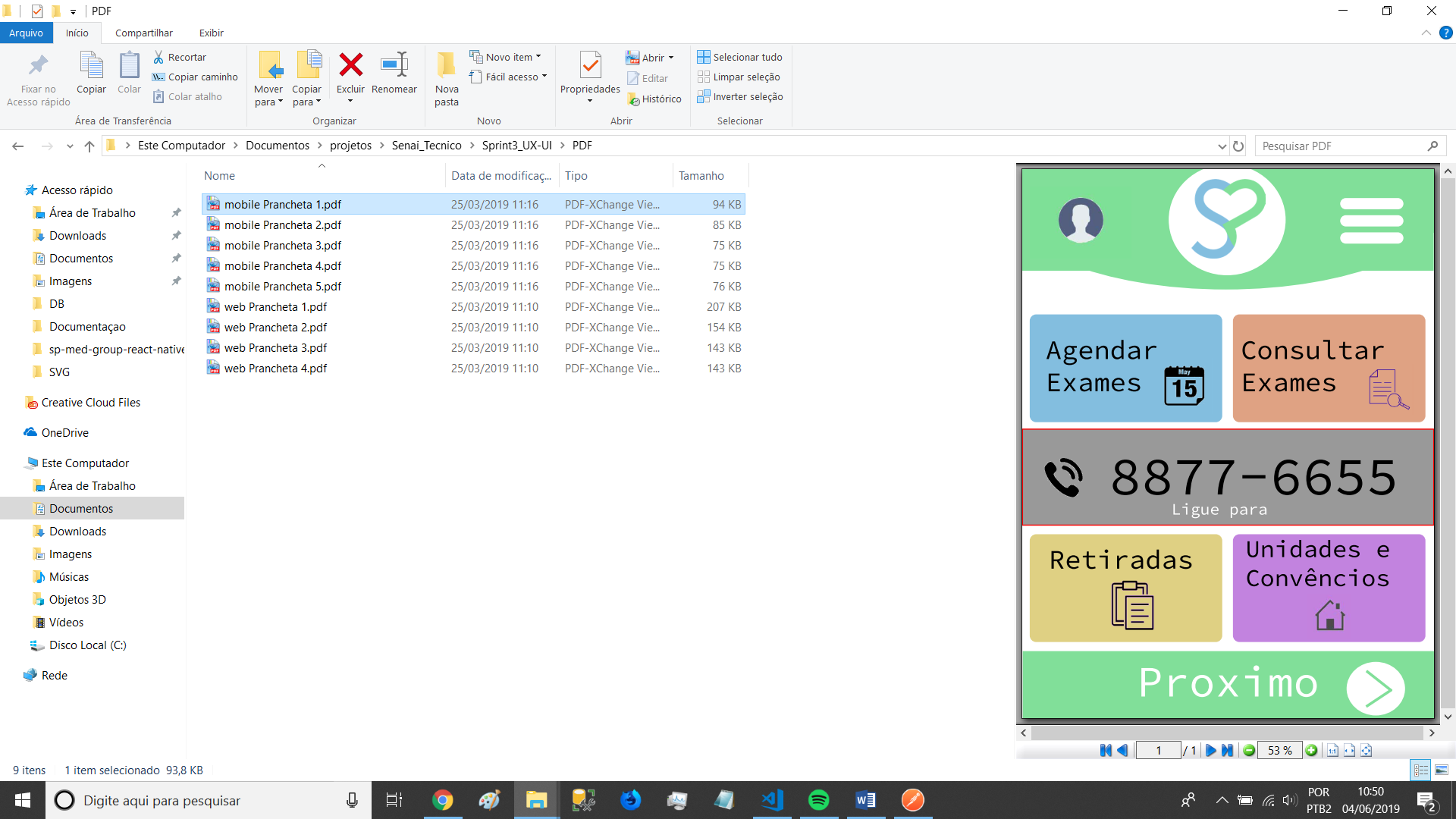


Página de Login



Mobile

Página principal



Back-End

**Tutorial**

**Utilizando os arquivos do Github**

Primeiramente é necessário baixar os arquivos no site: <https://github.com/ArturLOtt/Senai-Sprint2-Projeto-Final.git>

**Criação do banco de dados**

Requisito: Faça uso do Microsoft SQL Server Managements Studio ou outro programa de sua preferência que suporte a linguagem SQL

Na pasta Scripts, execute os três arquivos ali presentes, na seguinte ordem, para criar o banco de dados. Utilize a Tecla F5 ou clique na seta verde para executar os comandos.

* DDL – CRIACAO SPMEDGROUP
* DML – MANIPULACAO SPMEDGROUP
* DQL – QUERY SPMEDGROUP)

**Utilizando o banco de dados na aplicação**

Requisito: Faça uso do Visual Studio ou outro programa de sua preferência que suporte a linguagem C# e o Entity Framework

Abra o arquivo SPMEDGROUP\_MANHA.SLN no Visual Studio e prossiga com o próximo passo.

Utilizando o Gerenciador de Pacotes do Nuget, instale os seguintes arquivos:

* Microsoft.EntityFrameworkCore.Design;
* Microsoft.EntityFrameworkCore.SqlServer;
* Microsoft.EntityFrameworkCore.Tools;
* Sendgrid;
* Swashbuckle.AspNetCore;
* Swashbuckle.AspNetCore.Swagger;

Após isso, abra o “Console de Gerenciador de Pacotes”, dentro de “Ferramentas” no menu superior e execute o seguinte comando:

Scaffold-DbContext "Data Source=.\SqlExpress;Initial Catalog=SPMEDGROUP;User Id=sa;pwd=132;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Domains -ContextDir Contexts -Context SPMEDGROUPContext

Caso haja algum erro, verifique se o comando que esta tentando executar está igual ao comando apresentado. Caso o erro persistir adicione “-force” no final da linha desse comando e execute novamente, isso irá forçar o comando a ser executado.

Ao termino dessa operação você terá feito a ligação do banco de dados ao projeto.

**Fazendo requisições da sua aplicação**

Requisito: Faça uso do Postman ou outro programa de sua preferência em que seja possível fazer requisições com a internet.

Com o programa aberto, Clique em “Import” e selecione o arquivo “**SPMEDGROUP.postman\_collection.json**” para utilizar poder utilizar todas as funcionalidades.

Volte ao Visual Studio e coloque sua aplicação para funcionar. Para ver as requisições possíveis dessa API, utilize o seguinte link: http://localhost:5000/swagger

Para fazer uso de qualquer um dos métodos, abra o respectivo arquivo, preencha as informações necessárias e clique o grande botão azul SEND. Verifique o Status da requisição, se der 200 OK, ela foi executada com sucesso, caso contrário, há algum erro e é melhor revisar os passos anteriores.

Front-End

O projeto web foi realizado utilizando a tecnologia javascript através da biblioteca React

**Dependências:**

Os passos anteriores (Scripts e Backend)s devem ser efetuados para o correto funcionamento dessa aplicação.

**Tutorial**

Baixe o Node.js através deste link:

<https://nodejs.org/en/>

Sugere-se que se baixe a versão mais estável recomendada (a de menor numeração)

A seguir abra o prompt de comando e digite:

Npm install –g create-react-app

1. Instale o React  
   4.1. Baixe e instale o Node.js neste link -> [Node.js](https://nodejs.org/en/download/) 4.2. Abra o prompt de comando e digite : npm install -g create-react-app  
   4.3. Instale as dependencias do projeto : npm install -g react-router-dom  
   4.4. Baixe esse projeto  
   4.5. Vá para pasta C:\Users\**{NomeDoUsuario}**\AppData\Roaming\npm\node\_modules\create-react-app e copie a pasta **node\_modules** 4.6. Cole a pasta node\_modules dento da pasta **sp-medical-group**
2. Abra o prompt de comando no diretorio do projeto (mesma pasta) e execute o comando npm start
3. O projeto será aberto em uma aba do seu navegador padrão
4. Algumas informações extras -> [README.md](https://github.com/Chingling152/React-SpMedGroup/blob/master/sp-medical-group/README.md)

### 2. Sobre o projeto

Aqui terá informações sobre o projeto em geral.

#### 2.1. Arquitetura do projeto

Precisei criar esse arquivo para explicar onde está cada coisa do projeto (pois ele é muito grande e dividido em várias pastas).

##### 2.1.1. Paginas

Aqui ficam todas as páginas do projeto. Todas os arquivos de cada pasta são acessados através do arquivo **src/index.js**.

* **Administrador**  
  Aqui ficam todas as pastas de funcionalidades do Administrador. Essas páginas só podem ser acessadas caso o usuário seja um administrador.  
  **Home**: A página inicial do administrador, todas as outras páginas só podem ser acessadas através dela.  
  **Médicos/Pacientes/Especialidades/Instituições/Consultas**: São pastas onde o administrador poderá alterar ou cadastrar novos dados utilizando a API.
* **Usuário**  
  Aqui ficam todas as pastas de funcionalidades de um Usuário autenticado. Não deve ser confundida com a pasta Público. Essas páginas só são acessíveis para um usuário que tenha uma conta e esteja logado nela.  
  Todas as páginas podem ter diferenças (depende do nível de privilégio do usuário. Exemplo: a página de usuário de um médico terá mais funcionalidades do que a de um paciente) **Home**: A página inicial do usuário, todas as outras páginas só podem ser acessadas através dela. **Consultas**: Aqui ficará todas as informações de todas as consultas desse usuário. Médicos poderão alterar algumas informações delas enquanto pacientes poderão apenas visualiza-las.
* **Publico**  
  Aqui ficam todas as páginas que podem ser acessadas por qualquer pessoa independentemente de estar logada.  
  **Home**: A página inicial da aplicação, todos os usuários começam aqui caso não estejam logados. **Login**: Aqui é a página onde qualquer pessoa pode entrar em sua conta e ter acesso a todas as funções de um Administrador ou Usuário.  
  Falhas: Páginas que são acessadas apenas quando o usuário tenta acessar uma página que não existe ou que ele não tem permissão.

##### 2.1.2. Componentes

Nesta pasta ficarão partes que são reutilizadas em várias pastas (um bom exemplo é o Cabeçalho)

* **Feedback**  
  Pasta onde tem qualquer componente responsável por retornar uma mensagem ao usuário.
* **Partes**  
  São componentes comuns que compõem as paginas

##### 2.1.3. Recursos

Qualquer outro recurso que é usado pelas paginas do site (fotos/videos/icones/arquivos de estilização)

##### 2.1.4. Services

Todos scripts que são usados e compartilhados por varias paginas (e sim, poderia haver mais arquivos e eles poderiam estar mais organizados. Só que por agora está bom)

* **Autenticacao.TokenUsuario**  
  Busca o valor do token armazenado localmente e o retorna
* **Autenticacao.jwtParse**  
  Converte o token do metodo Autenticacao.TokenUsuario e envia para o metodo Autenticacao.TokenValido se ele for valido , o retorna em forma de objeto , se não , retorna null
* **Autenticacao.TokenValido**  
  Verifica se a data de espiração do token é menor do que a data atual. Se ela for , ele ainda é valido.
* **ApiService.chamada**  
  Uma promessa que pode fazer 4 ações diferentes. Essa promessa precisa ter um endpoint (parte do link da API) para saber onde deverá ser feita a requisição.  
  Os metodos dessa promessa serão o que será feito nessa requisição (Alterar valores , cadastrar , enviar , buscar ,etc).
  + **Login** : Metodo uti apenas para o usuario. Semelhante ao CADASTRAR , mas este metodo não precisa do token.
  + **Alterar** : Envia uma requisição na API para alterar algum valor (precisa de um corpo para ser enviado , ou seja um objeto que se encaixe no padrão da endpoint API)
  + **Cadastrar** : Envia uma requisição na API para enviar um valor (possivelmente inseri-lo no banco de dados) , assim como o metodo ALTERAR , precisa de um corpo que se encaixe com o valor aceito no endpoint da API
* **Cabecalho**  
  Recebe os valores do token (armazenado localmente), verifica se ele não é nulo e qual o nivel de privilegio dele e retorna um cabeçalho com as funcionalidades disponiveis para o Usuario
* **Enums.enumParse**  
  Converte uma enumeração do tipo EnSituacaoConsulta para um valor numerico sendo :
* 1 => Aguardando
* 2 => Concluida
* 3 => Cancelada

#### Cronograma do projeto

Aqui ficara o cronograma do projeto (Iniciado de 12/04 e finalizado dia 26/04).  
Para melhores informações tem o [cronograma.png](https://github.com/Chingling152/React-SpMedGroup) e [README.md](https://github.com/Chingling152/React-SpMedGroup) na pasta Informações do projeto.

### Utilizando o projeto

É um site simples onde tentei focar mais em UX do que no layout então não precisa de muitas informações de como usa-lo, só deixarei algumas informações.  
**Não é possivel (ainda) recuperar e alterar a sua senha**.  
**Somente um administrador pode cadastrar novos Usuarios** . Então você deve ter pelo menos um usuario cadastrado no banco de dados (com privilegios de administrador) para poder usar a aplicação.  
Ainda não existe filtragem de consultas.  
Para criar um **Paciente** ou **Medico** você precisa criar um **Usuario** antes com o nivel de privilegio desejado.  
O site ainda não tem todas as sua funcionalidades (Mas está completo por agora).

#### Iniciando projeto

Para iniciar o projeto

#### Permissões

* **Administradores** tem acesso a todas as paginas do site.
* **Administradores** podem cadastrar e alterar dados de Usuarios / Pacientes / Medicos / Consultas / Clinicas.
* **Administradores** não podem adcionar descrição às consultas.
* **Administradores** não podem ter consultas e então não tem acesso a pagina de visualizar suas proprias consultas.
* **Medicos** e **Pacientes** tem o mesmo nivel de acesso podendo acessar as mesma paginas só que com
* **Medicos** só podem alterar a descrição de suas consultas.

#### Validação

O site tem uma validação boa, mas não é o foco principal deixar o front-end com tanta segurança assim, então algumas vezes os dados vão até a API para serem validados ao inves de serem validados ali mesmo.  
Grande parte dos dados que são validados são coisas simples como valores requiridos ou algo fora do padrão (como por exemplo inserir uma senha com menos de 8 caracteres). Mas a maioria dos dados são validados na API (Veja : [Api : Validação e Autenticação](https://github.com/Chingling152/WebApi-SPMedGroup" \l "Valida%C3%A7%C3%A3o-e-Autoriza%C3%A7%C3%A3o))

### Prototipação

O site teve prototipação, mas depois que os prazos foram apertando e eu vi que isso estava tomando muito tempo meu , eu tive que pular direto pra a parte do codigo.  
O resultados tambem não foram iguais.

#### Wireframe

Todos os layouts de baixa fidelidade estão aqui (Web e mobile (mobile será movido para outro projeto)) [Neste Link](https://github.com/Chingling152/React-SpMedGroup/tree/master/Anexos/Layout%20de%20baixa%20fidelidade). A prototipação mobile não tem muito a ver com resposividade (afinal o site não ficou totalmente responsivo mesmo eu utilizando display:grid )

#### Design

O design de alta fidelidade não foi algo tão preciso quando os de baixa fidelidade.  
Talvez aqui eu deixe uma foto de comparação do projeto final e a Prototipação.

## Links

Mobile

## Instalação

Para que o projeto funcione você precisará ter :

* [Banco de dados](https://github.com/Chingling152/SQL-SPMedgroup)
* [API](https://github.com/Chingling152/WebApi-SPMedGroup) (E todos seus requisitos)
* [Node.js , JDK e Android SDK](https://gist.github.com/Chingling152/dc3e7dc6cd784636649a43e9fcaeb060)  
  Todos eles tem uma documentação propria, então é só seguir as instruções.
* O ultimo requisito é ter um celular android ou uma AVD com google play services instalado para que possa saber sua localização e da consulta.

### Preparação

Antes de iniciar o projeto você deve fazer tudo que está nos links acima. Então você deverá :

* Executar a API (verifique se ela esta funcionando ou não no endpoint /swagger/index.html testando o login)
* Caso tudo esteja funcionando, abra o cmd na pasta do projeto (aperte F4 , apague todo o caminho da pasta e digite **cmd**)
* Execute o primeiro comando : **npm install** (irá instalar tudo que o projeto precisa ([Veja:Bibliotecas](https://github.com/Chingling152/ReactNative-SpMedicalGroup" \l "Bibliotecas)))
* Você vai precisar ir no arquivo [APIService.js](https://github.com/Chingling152/ReactNative-SpMedicalGroup/blob/master/src/services/apiServices.js) e mudar a variavel **baseURL** para o ip da sua API.  
  Exemplo :

const baseURL = "http://{coloque seu ip aqui}:5000/api/v1/"//mude para o seu ip aqui

* Você tambem devera ir para a API e setar o mesmo ip para ela no arquivo [Properties/launchSettings](https://github.com/Chingling152/WebApi-SPMedGroup/blob/master/Senai.WebApi.SpMedGroup/Properties/launchSettings.json)
* O proximo comando sera : **react-native run-android**
* O projeto ira abrir na avd ou no seu celular (depende qual você esta usando)

### Bibliotecas

Aqui está todas as bibliotecas que serão instaladas ao ser digitado o comando **npm install** na pasta do projeto (alem do react-native)

* **react-navigation** - navegação entre paginas
* **react-native-gesture-handler** - lidar com scroll e gestos de zoom e rolagem
* **@react-native-community/async-storage** - armazenar dados do usuario localmente
* **jwt-decode** - validar dados do usuario
* **react-native-maps** - gerar mapas
* **react-native-maps-direction** - exibir rota entre 2 pontos
* **react-native-google-maps-directions** - Enviar dados para o google maps

Essas bibliotecas também deverão ser linkadas ao projeto utilizando o comando:

react-native link {nome da biblioteca}

### Google API

Caso a minha API do google não esteja funcionando (porque há um limite de requisições). Você devera criar uma nova, [neste link](https://console.developers.google.com/google).  
Lá você tera que gerar uma key para as seguintes APIs.

* Directions API
* Geocoding API
* Maps SDK for Android

## Telas

Apos iniciar o projeto a primeira tela sera a de login (você precisa pelo menos ter um valor cadastrado no banco de dados ([Veja: Banco de dados- Valores inicias](https://github.com/Chingling152/SQL-SPMedGroup/blob/master/Essenciais/5-Valores_iniciais.sql)) ou insira algum usando a API/Site)  
Ao logar você poderá apenas visualizar suas consultas, ou clicar em uma delas e ver todas as suas informações.  
Todas as telas só são acessiveis depois de o usuario estar logado.

Tela de login

Caso ocorra algum problema (com a API por exemplo) será retornado apenas um erro sem muitos detalhes.  
O mesmo acontece com erro de autenticação e validação.

Ao entrar na tela de visualizar consultas, você verá todas as consultas (caso você logue como administrador você podera ver todas , mas não podera altera-las porque esta função só é disponivel no [Site da SPMedicalGroup](https://github.com/Chingling152/React-SpMedGroup))

Tela de listagem de consultas

Na tela de visualizar terá apenas alguns detalhes a mais. Como por exemplo a descrição completa (que na listagem de todas as consultas apenas mostra no maximo 114 caracteres).

Tela de listagem de consulta

Você podera ver a sua localização atual e o caminho para a clinica onde a consulta foi marcada. Para isso voce deverá permitir que o aplicativo use a função GPS do seu celular e estar com ela ligada.

## Services

## Cronograma

O projeto foi iniciado no dia **13/05/2019** e finalizado no dia **23/05/2019**.  
Aqui ficara meu cronograma do projeto com todas minhas atividades.

* **Dia 0**  
  Eu ja tinha o layout pronto do mobile que foi feito na criação do site (então deixei isso como dia 0 porque não aconteceu no dia do projeto em si, mas foi usado nele (um pouco))
* **Dia 1**
  + Iniciei a arquitetura do projeto criando todas as paginas (apenas com um Text nela com o nome da pagina).
  + Instalei quase todas as bibliotecas necessarias (AsyncStorage foi instalada no **dia 3**).
  + Iniciei a pagina de login. Fiz os primeiros testes de login.
* **Dia 2**
  + Importei meu arquivo de services padrão (Conexão com API e Validação de token).
  + Recebi o token e lidei com os erros de email/senha.
  + Criei as rotas entre as telas e fiz a o usuario ser redirecionado para a pagina de listar consultas ao logar.
  + Iniciei a pagina de visualização de consultas.
  + Componentizei a consulta assim poderia buscar os valores.
* **Dia 3**
  + Iniciei a estilização do projeto.
  + Adicionei uma Splashscreen ao projeto e um icone.
  + Estilizei a pagina de Login e o StackNavigator.
  + Instalei a biblioteca para o AsyncStorage e iniciei o login automatico.
  + Iniciei a função de deslogar o usuario (ainda não estava 100% funcional).
* **Dia 4**
  + Finalizei a visualização de consultas.
  + Finalizei a função de logout.
  + Iniciei a visualização de apenas uma consulta.
  + Adicionei um indicador de carregamento nas telas de login e listagem de consulta.
* **Dia 5**
  + Finalizei a listagem de uma consulta.
  + Iniciei a documentação colocando informações ate aqui (fui incrementando esse cronograma a cada dia)
  + Adicionei feedback aos usuarios que não tem consultas
  + Adicionei uma pagina inicial para verificar dados do usuario e API
* **Dia 6**
  + Passei o dia inteiro retocando o projeto (testando vulnerabilidades).
  + Arrumei o design do aplicativo
  + Fixei alguns problemas de UX
* **Dia 7**
  + Iniciei a parte de localização do usuario.
  + Tentei resolver problemas com permissão do usuario para acessar o GPS.
  + Finalizei os requisitos minimos do aplicativo
* **Dia 8**
  + Resolvi problemas do GPS e permissão do usuario
  + Inicei o mapa
  + Modifiquei a status bar
* **Dia 9**
  + Adicionei o mapa com a localização da consulta e do Usuario
  + Criei a API no google maps
* **Dia 10**
  + Fiz a rota da localização do usuario até a da clinica Utilizando a API do google
  + Fiz o feedback caso a API não esteja funcionando ou o local da consulta não exista
  + Não adicionei a opção de redirecionar ao google maps p

Arquitetura do Projeto

Referências

Links

Livros