

SUMÁRIO

SUMÁRIO	1
1 BANCO DE DADOS	3
1.1 DEPENDÊNCIAS	3
1.2 FUNCIONAMENTO	3
1.3 PRÉ REQUISITOS	4
1.4 INSTALAÇÃO E SETUP	4
1.5 CONFIGURAÇÃO DO DOCKER	6
1.6 COLOCANDO AS INFORMAÇÕES NO BANCO DE DADOS	7
2 BACKEND/WEBSERVICE	9
2.1 DEPENDÊNCIAS	9
2.2 INSTALAÇÃO E COMPILAÇÃO	9
3 FRONTEND	13
3.1 DEPENDÊNCIAS	13
3.2 INSTALAÇÃO/COMPILAÇÃO	13
3.3 USABILIDADE	14
3.3.1 Página inicial do site	15
3.3.2 Página de login e cadastro	15
3.3.3 Página de gráficos	16
3.3.4 Página de Dados de Equipamentos	20
3.3.5 Página de mapas	22
3.3.6 Equipamentos	23
3.3.7 Métricas	24
3.3.8 CNES	25
4 MOBILE	26
4.1 MOBILE ANDROID	26
4.1.1 Dependências	26
4.1.2 Instalação e Compilação	26
4.1.3 Usabilidade PinSIS mobile android	31
4.2 MOBILIO IOS	38
4.2.1 Dependências	38
4.2.2 Instalação/Compilação	39
4.2.3 Usabilidade PinSIS mobile iOS	46
4.2.4 Carregar/Recarregar portal PinSIS	47
4.3 USABILIDADE	49

4.3.1 Tela Inicial	50
4.3.2 Cadastro de usuário	53
4.3.3 Login de usuário	55
4.3.4 Logout de usuário	57
4.3.5 Cadastro de hospital	58
4.3.6 Listar hospital	61
4.3.7 Cadastro de equipamento	63
4.3.8 Listagem de equipamento	73
4.3.1 Gráficos de uso dos equipamentos	77
4.3.2 Métricas	80
5 COLOCANDO O SITE DISPONÍVEL AO PÚBLICO USANDO O NGINX	82
5.1 PRÉ REQUISITOS	82
5.2 INSTALAÇÃO	82
7 AGENTE DE COLETA	93
7.1 DEPENDÊNCIAS	93
7.2 INSTALAÇÃO/EXECUÇÃO	93

1 BANCO DE DADOS

1.1 DEPENDÊNCIAS

Esta seção tem como objetivo apresentar as dependências necessárias para utilizar o banco de dados do PlnSIS, com o passo-a-passo para a instalação de cada dependência. Para realizar este tutorial é necessário um ambiente rodando Debian GNU/Linux 10.

- **PostgreSQL**

1. Digite em seu terminal o seguinte comando:

```
sudo apt-get -y install postgresql
```

- **Pandas**

1. Digite em seu terminal o seguinte comando:

```
pip install pandas
```

- **Hashlib**

1. Digite em seu terminal o seguinte comando:

```
pip install hashlib
```

- **NodeJS v13.12.0**

1. Digite em seu terminal o seguinte comando:

```
sudo apt install nodejs npm
```

- **Docker**

1. Tutorial de instalação e funcionamento abaixo

1.2 FUNCIONAMENTO

Como piloto do PlnSIS utilizamos os dados provenientes do Hospital Erasto Gaertner. Os dados recebidos do hospital são compostos por dois bancos de dados: do TASY (software de gestão do hospital Erasto Gaertner, no qual podemos identificar o tipo de convênio) e da VARIAN (software de administração dos equipamentos monitorados). Para obtermos as informações necessárias para as análises de dados do PlnSIS, precisamos cruzar os dados destes dois bancos e realizar um tratamento de dados. O tratamento dos

dados se refere à adaptação dos dados para os propósitos necessários para o projeto. Nesta etapa, recebe-se o conjunto dos dados da entrada dos pacientes, para fins de verificação do convênio (SUS ou particular), e os dados dos tratamentos feitos pelo equipamento monitorado. Os dados dos tratamentos são anonimizados e a referência cruzada com os dados da entrada de pacientes é feita por um identificador numérico (cartão do SUS). Toda a transmissão é feita através do agente desenvolvido e, após processamento inicial, as informações relevantes ao projeto são armazenadas no banco de dados do PlnSIS para disponibilização no Portal Web.

As informações que são pertinentes ao projeto são, quanto ao paciente, um número de prontuário da consulta para os propósitos de prevenir duplicatas e o tipo de convênio para o quesito de contabilização dos tratamentos realizados. Quanto às informações providas pelo dispositivo médico, são relevantes aquelas referentes a sua utilização. Para o quesito de manutenção e monitoramento, é relevante também a identificação do equipamento em particular e sua instituição de origem.

Para cruzarmos as informações dos bancos de dados do TASY e VARIAN, utilizamos o identificador do paciente, utilizado para cruzar com as informações do TASY.

1.3 PRÉ REQUISITOS

Para a criação do banco de dados, será necessário o acesso a JSONs referentes ao uso de máquinas, aqui obtidos do sistema da VARIAN. Além disso, será necessário o dump do banco de dados referente aos agendamentos de consultas, nesse caso o banco de dados do TASY. Também será necessário credenciais de acesso ao banco de dados utilizado.

1.4 INSTALAÇÃO E SETUP

Antes de tudo, é necessário baixar o repositório que pode ser encontrado [aqui](#).

Então, deve-se instalar as bibliotecas utilizadas pelo node, para isso basta digitar no diretório do projeto:

npm install

Há alguns arquivos que precisam ser criados e alguns outros a serem alterados para que o script consiga rodar.

É preciso criar o diretório dos logs e que vai guardar os JSONs após serem processados, para isso apenas vá ao diretório do pinsis-private-data e digite:

mkdir -p log/json/

Agora, é necessário criar o arquivo de configuração do banco de dados, para isso basta copiar o arquivo de exemplo:

cp config.js.example config.js

Após isso, basta modificar as informações do config.js para as da database que será utilizada.

Ainda, falta alterar as informações no script populate-db.sh, basta apenas modificar a variável storage_dir para o caminho absoluto do diretório que guarda os JSONs ainda não processados, e modificar a variável private_data_dir para o caminho absoluto para o diretório pinsis-private-data.

Além disso, o dump do banco de dados administrativo precisa ter seu nome e localização indicados no script pinsis-private-data/src/scriptsBD/extract_info.sh, modificando a variável db_to_restore para o path absoluto para o dump.

Por fim, será necessário inicializar o banco com o schema das tabelas:

./setup_db.sh “nome da database”

Onde “nome da database” é o nome da database que será utilizada para o projeto

1.5 CONFIGURAÇÃO DO DOCKER

Devido uma limitação das novas versões do *Debian*, não é possível instalar o servidor Microsoft SQL server, necessário para o correto funcionamento do sistema. Para contornar este problema, foi adicionado o uso de um *Docker container*, que contém o servidor Microsoft SQL presente e possibilita o uso no sistema *Debian*. Considerando que os passos descritos na seção 1.1, referentes a instalação da *Docker Engine*, siga os seguintes passos para iniciar o *Docker Container*.

1 - Baixe a imagem do *container* que contém o Microsoft SQL

```
docker pull mcr.microsoft.com/mssql/server
```

2 - Após o container ter sido baixado localmente, inicie o container. Os parâmetros passados serão explicados posteriormente

```
docker run --name='mssql_server' -v '<PATH DUMPS>:/root:ro' -e 'ACCEPT_EULA=Y'  
-e 'SA_PASSWORD=<SA PASSWORD>' -p 1433:1433 -d  
mcr.microsoft.com/mssql/server:latest
```

O comando anterior irá iniciar um container rodando o Microsoft SQL, com nome *mssql_server*. Os parâmetros passados são:

- **-v** - Indica que o container deverá montar a pasta PATH DUMP no container na pasta */root*. Esta pasta contém os dumps que serão utilizados para iniciar o banco.
- **-e ACCEPT_EULA=Y** - Define a variável de ambiente que aceita os termos para usar o banco Microsoft SQL
- **-e SA_PASSWORD=Y** - Define a variável de ambiente que indica qual o password a ser usado para o usuário SA.

- **-p 1433:1433** - Indica que a porta 1433 do container deverá ser exposta no sistema na porta 1443, ou seja, o serviço que está rodando no container na porta 1433 (banco Microsoft) estará exposto na porta 1433 da máquina
- **-d** - Executa o container em *detached mode*, colocando ele para executar em *background*.
- **mcr.microsoft.com/mssql/server:latest** - Imagem que será usada para rodar o container, neste caso é a imagem que foi baixada anteriormente no passo 1

Caso seja necessário iniciar o container (após ter sido iniciado com *docker run*), basta executar:

docker start mssql_server

Caso seja necessário parar o container, basta executar:

docker stop mssql_server

Caso seja necessário excluir o container (removendo os dados do banco), basta executar

docker rm mssql_server

Para verificar se o container com o banco está rodando, basta executar `docker ps`. Após executar o comando, verifique que o container `mssql_server` aparece na lista de containers em execução.

1.6 COLOCANDO AS INFORMAÇÕES NO BANCO DE DADOS

Com tudo devidamente configurado, basta apenas rodar o comando no diretório do projeto:

./main.sh nome do hospital

Substituindo “nome do hospital” pelo nome do hospital de origem do JSON.

Caso tenha digitado o nome de um hospital que não exista no banco de dados, ele dará uma mensagem de erro e retornará os nomes dos hospitais que estão no BD.

Não havendo o hospital desejado, será necessário adicioná-lo pelo frontend, pela página de mapas -> cadastrar novo hospital.

Após tudo isso, rodado com o nome de um hospital válido, o script inserirá os JSONs no BD um a um e os jogará na pasta log/json, anonimizará as tabelas, inserirá as informações referentes aos agendamentos pegos do banco de dados administrativo e, por fim, também os anonimizará.

2 BACKEND/WEBSERVICE

Este README tem como objetivo apresentar os requisitos necessários para compilar/executar o backend/webservice do pinsis em um ambiente de testes e produção. Ele também cobre parte da usabilidade e funcionalidades, com exemplos

2.1 DEPENDÊNCIAS

Para ser colocar o serviço no ar do portal é preciso que sua máquina possua o [node](#) na versão v13.12.0.

Além disso, é preciso ter instalado em sua máquina o [PostgresSQL](#) e o [MongoDB](#)

Caso deseje rodar o serviço numa máquina de produção também será necessário realizar a instalação e configuração do [nginx](#).

Para compilar o webservice é necessária a instalação/uso do LXterminal e um ambiente rodando Debian GNU/Linux 10 (buster).

2.2 INSTALAÇÃO E COMPILAÇÃO

Esta seção irá cobrir os procedimentos necessários para poder compilar o backend bem como ter uma versão funcional do webservice rodando em um ambiente Debian. É necessário que você tenha na sua máquina local as dependências explicitadas na seção ‘Dependências’ para prosseguir com os próximos passos.

1. Entre nesse [link](#) e baixe o repositório do github, o repositório irá ser baixado de forma compactada (.zip);
2. Descompacte o arquivo;
3. Abra seu terminal (Ctrl + Alt + t) e vá até o diretório onde você baixou o projeto do webservice. Por padrão, o gitlab coloca os arquivos baixados no diretório de Downloads. Para acessar o diretório digite em seu terminal o comando:

```
cd Downloads/webservice-development
```

4. Digite o comando npm install;
5. Digite:

cp config.js.example config.js

6. Digite o comando abaixo para abrir o editor de texto e configurar o configuração do arquivo de banco de dados:

vim config.js

Um exemplo de como escrever as configurações baseado nas definições suas do banco de dados pode ser encontrada digitando:

cat config.js.example

```
danterat@sakgboy:~/webservice$ cat config.js.example
var config = module.exports = {};

config.db_config = {
    user: DB_USER,
    password: DB_PASS,
    database: DB_DBNAME,
    host: DB_HOST,
    port: DB_PORT
};
```

7. Digite npm start;
8. Criando o serviço:

Primeiramente, nós criaremos um serviço para rodar o backend de fundo de modo mais organizado, para isso crie um diretório:

mkdir services; cd services

Ali, crie um script em bash que inicie o backend e cole o seguinte código, substituindo o path para o backend (verificar se < /path/para/arquivo/com/senha/do/db não é necessário):

gedit service.sh

```
nodejs /path/para/webservice/src/server.js -p 3000 -n numero_de_workers -c  
/path/para/webservice/config.js >> /path/para/log&
```

Exemplo completo:

```
nodejs /home/pinsis/webservice/src/server.js -p 3000 -n 4 -c  
/home/pinsis/webservice/config.js >> /root/.log_webservice&
```

Então, torne esse arquivo executável:

```
chmod 755 service.sh
```

Teste o service.sh para ver se funciona:

```
./service.sh
```

Caso ocorra o erro de que o modulo do bcrypt foi compilado com uma versão diferente do node, rode as seguintes linhas:

```
cd /path/para/webservice  
npm uninstall bcrypt  
npm install bcrypt
```

Depois, crie um arquivo definindo o serviço e cole o seguinte código, substituindo o path pelo path desse diretório:

```
gedit pinsis.service
```

```
[Unit]
```

```
Description=pinsis
```

```
[Service]
```

```
Type=forking  
ExecStart=/bin/bash /path/desse/diretorio/service.sh
```

```
[Install]  
WantedBy=multi-user.target
```

Por fim, copie esse arquivo para o diretório do systemd. Caso esteja rodando como root:

```
cp pinsis.service /lib/systemd/system/
```

Caso não:

```
cp pinsis.service /usr/local/lib/systemd/system
```

Agora, o serviço do pinsis deverá estar criado, então digite para habilitar o serviço

```
sudo systemctl enable pinsis
```

E, finalmente, para iniciá-lo:

```
sudo systemctl start pinsis
```

E o back estará rodando de fundo

Caso queira checar o status do serviço basta digitar:

```
sudo systemctl status pinsis
```

3 FRONTEND

Este README tem como objetivo apresentar os requisitos necessários para compilar/executar o frontend/site do pinsis em um ambiente de testes e produção. Ela também cobre parte da usabilidade e funcionalidades das aplicações do site do PinSIS, com exemplos.

3.1 DEPENDÊNCIAS

O código da aplicação do frontend do PinSIS possui todas as suas dependências documentadas no arquivo pinsisApp/package.json que pode ser encontrada [aqui](#). As dependências são fundamentais para possibilitar a compilação do portal e para que o serviço de gráficos e mapas sejam interativos e funcionais. Essas dependências são instaladas de forma automática, como será descrito na seção de Instalação.

Para ser possível realizar a compilação do portal é preciso que sua máquina possua o [node](#) na versão v13.12.0.

Para compilar o portal é necessária a instalação do LXterminal e um ambiente rodando Debian GNU/Linux 10 (buster).

3.2 INSTALAÇÃO/COMPILAÇÃO

Esta seção irá cobrir os procedimentos necessários para poder compilar a aplicação bem como ter uma versão funcional da aplicação rodando localmente em um ambiente Debian. É necessário que você tenha na sua máquina local as dependências explicitadas na seção ‘Dependências’ para prosseguir com os próximos passos.

1. Entre nesse [link](#) e baixe o repositório do github, o repositório irá ser baixado de forma compactada (.zip);
2. Descompacte o arquivo;
3. Abra seu terminal (Ctrl + Alt + t) e vá até o diretório onde você baixou o projeto do pinsis-portal. Por padrão o gitlab coloca os arquivos baixados no

diretório de Downloads. Para acessar o diretório digite em seu terminal o comando:

cd Downloads/pipsis-portal-development/pipsisApp

4. Digite o comando **npm install**;
5. Para fazer o projeto funcionar em **ambiente local** digite o comando **npm start** no terminal e, em seu navegador acesse o link <http://localhost:4200/>. Ou, para realizar os testes do [mobile-cad](#) em um dispositivo android físico é necessário o uso do comando **npm start --host 0.0.0.0 --ssl true**. Deve aparecer em seu terminal uma mensagem similar a essa:

```
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2020-10-22T12:26:20.252Z
Hash: 6d698a4aecf593ce2c0c
Time: 18213ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 909 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 603 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 684 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 14.3 MB [initial] [rendered]

webpack: Compiled successfully.
```

6. Caso deseje fazer o código em ambiente de produção é preciso ter o nginx instalado e configurado em sua máquina e digitar o comando :

ng b --prod

3.3 USABILIDADE

A página inicial do site possui as funcionalidades de:

- Cadastrar/logar usuários;
- Acessar as páginas de gráficos dos equipamentos monitorados;
- Acessar a página de mapas para visualizar os hospitais monitorados e cadastrar novos hospitais;
- Acessar a página de equipamentos para visualizar o estado de um equipamento ou cadastrar um equipamento novo;
- Acessar a página de métricas para visualizar os testes de escalabilidade do projeto;
- Acessar a página do CNES, para visualizar gráficos relacionados ao Cadastro Nacional de Estabelecimento de Saúde.

3.3.1 Página inicial do site

Onde o usuário pode acessar todas as outras páginas referentes ao projeto, utilizando os botões para cada página, também é possível acessá-los através do menu na parte superior da página.



3.3.2 Página de login e cadastro

Disponibiliza meios para criar ou logar em contas que dão acesso às outras funcionalidades do site, em especial aquelas que envolvem cadastro de hospitais ou equipamentos

The image consists of two vertically stacked screenshots of a web application interface. Both screenshots have a light blue header bar. The top screenshot is titled 'Login' and contains two input fields labeled 'Usuário' and 'Senha', followed by a purple 'Entrar' button. The bottom screenshot is titled 'Cadastro' and contains three input fields labeled 'Usuário', 'Senha', and 'Confirmar senha', followed by a purple 'Cadastrar' button. The header bar includes the logo 'PlnSIS' (Pesquisa e Inovação em Sistemas de Informação para Saúde), a navigation menu with 'LOGIN', 'CADASTRO', 'PÁGINA INICIAL', 'SOBRE O PlnSIS', and 'CNES', and a search bar.

3.3.3 Página de gráficos

Essa página disponibiliza gráficos referentes às consultas realizadas nas unidades monitoradas, verificando a quantidade, o tipo, isto é, se é pelo SUS ou não, e se foi executada ou não.

O gráfico pode ser modificado para atender às necessidades do usuário utilizando o menu na barra lateral esquerda, bastando selecionar as opções desejadas e clicar em buscar.

Dentre as opções, temos a de mostrar o gráfico como uma distribuição no tempo em barras, linear, proporcional, com valores absolutos ou como comparação.

Também pode-se filtrar as consultas pelo seu status de execução e, ainda, selecionar o intervalo de tempo desejado e qual o tamanho do intervalo representado no gráfico.

Pode-se também modificar o gráfico para um que revela a quantidade de raios disparados pelo acelerador linear cadastrado, ou também um que mostra as datas de manutenção dessa máquina

Os gráficos de barras, linear, proporção e mostrando valores absolutos são por distribuição ao longo do tempo, portanto, as informações são mostradas considerando tamanho do intervalo selecionado em "Tamanho do intervalo", podendo ser diário ou mensal, além do status da consulta. Como mostra as imagens abaixo:

- Gráfico de distribuição dos atendimentos ao longo do tempo em barras com informações de consultas executadas e não executadas, com o intervalo mensal:



- Gráfico linear de atendimentos ao longo do tempo em valores absolutos com consultas executadas e não executadas, com o intervalo mensal:



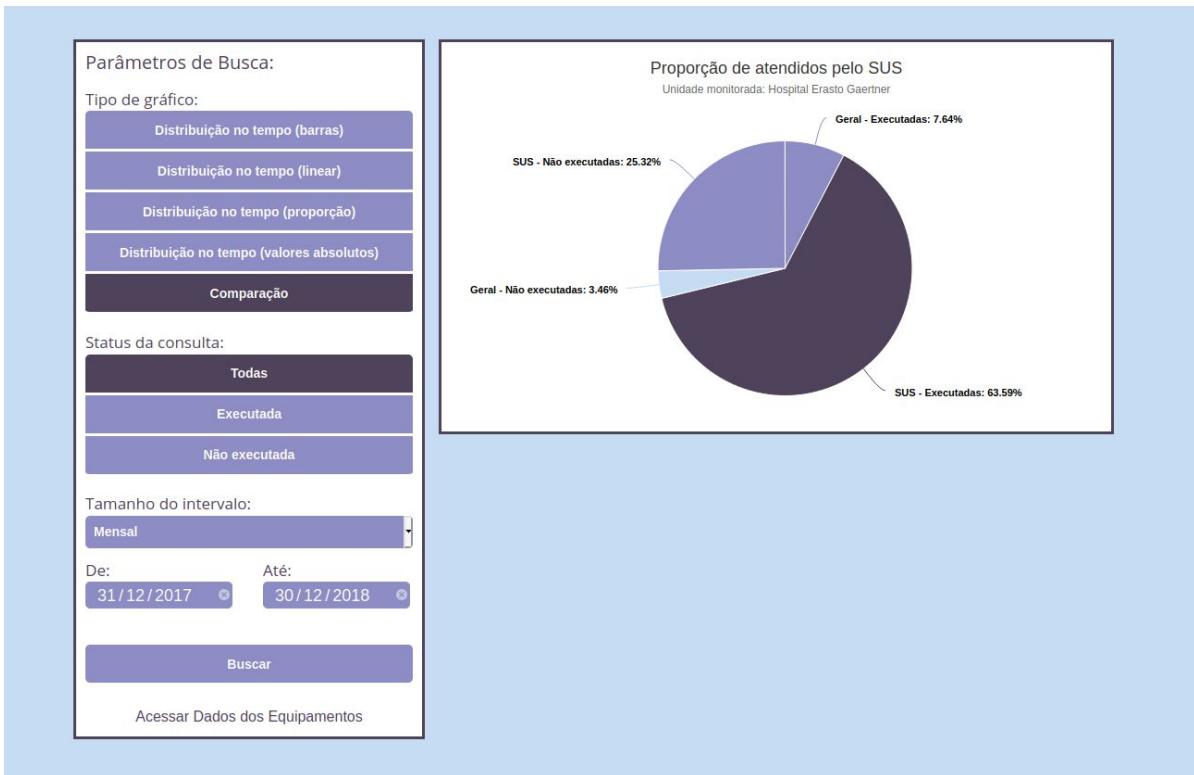
- Gráfico por proporção de atendimentos ao longo do tempo em valores percentuais com consultas executadas e não executadas , com o intervalo mensal:



- Gráfico ao longo do tempo em valores absolutos com consultas executadas e não executadas, com o intervalo mensal:



No gráfico de comparação, é mostrado um gráfico pizza, com a proporção de pacientes atendidos pelo SUS, com as informações de acordo com as opções selecionadas.



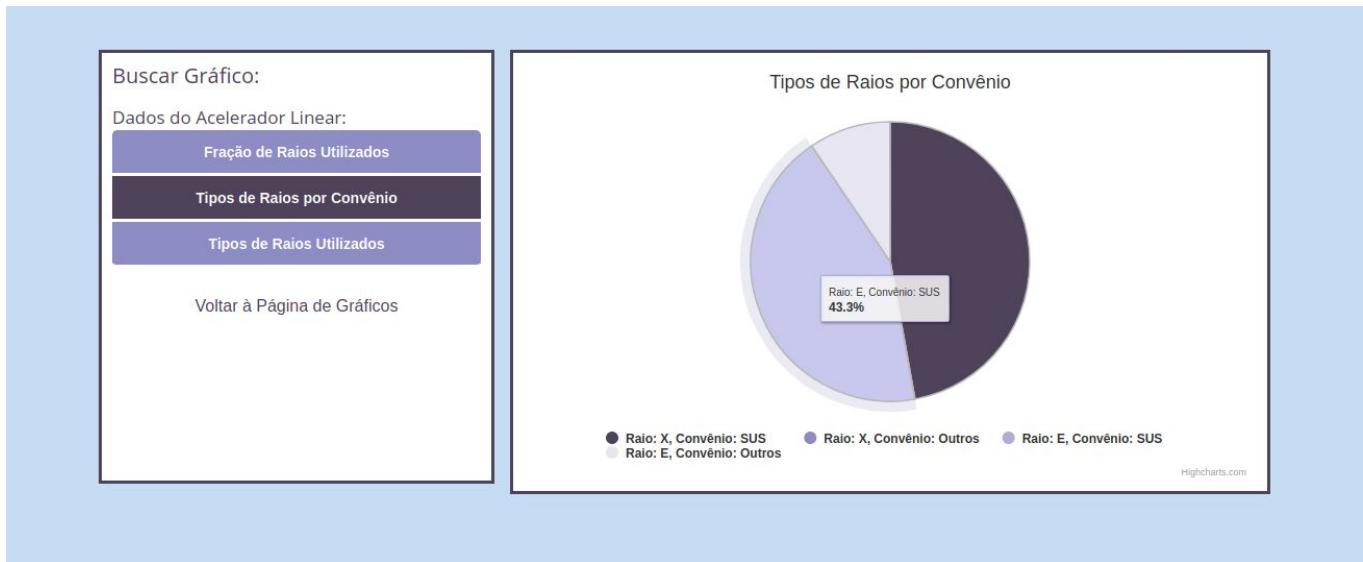
3.3.4 Página de Dados de Equipamentos

Dentro da página de gráficos, na parte inferior da caixa de Parâmetros de Busca há um link para acessar os dados dos equipamentos. Nele é possível buscar os gráficos relacionados com os raios dos equipamentos:

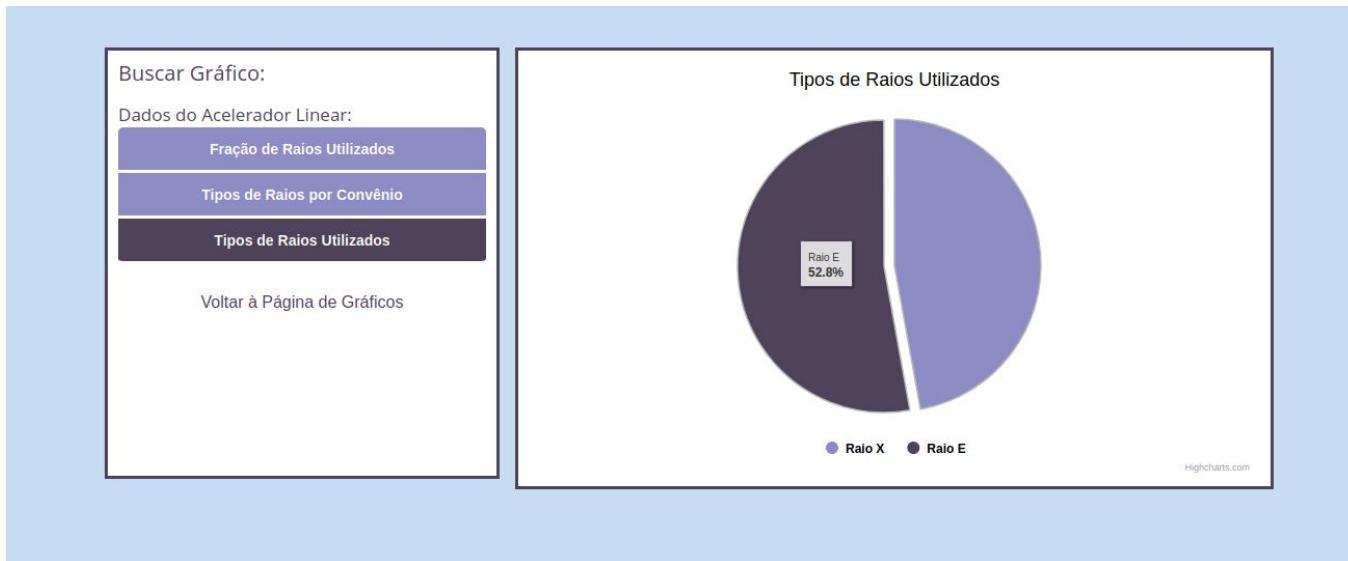
- O gráfico de Fração de Raios Utilizados mostra a média dos raios disparados separados por tipo de Raio:



- O gráfico de Tipos de Raio por Convênio mostra em um gráfico de pizza a porcentagem de cada raio utilizado pelo SUS e por outros convênios:



- O gráfico de Tipos de Raios Utilizados apresenta em forma de um gráfico de pizza a porcentagem da quantidade de cada tipo de raio utilizado:



3.3.5 Página de mapas

Essa página mostra no mapa a localização de todos os hospitais monitorados pelo Pinsis, clicando em um dos pins do mapa poderá ser visto o nome do hospital e a cidade em que se encontra. Além disso, os hospitais cadastrados são mostrados no menu à direita, podendo serem clicados para mostrarem uma imagem dele.

Caso se deseje cadastrar um novo hospital, basta clicar no botão à direita, preencher as informações pedidas e ele será prontamente adicionado ao mapa.

The screenshot shows the PlnSIS website interface. At the top left is the logo "PlnSIS". To its right is the text "Pesquisa e Inovação em Sistemas de Informação para Saúde". On the right side of the header are four buttons: "LOGIN", "CADASTRO", "PÁGINA INICIAL", "SOBRE O PlnSIS", and "CNES". Below the header is a map of South America with several blue location pins placed on Brazil. A sidebar on the right contains a button "Cadastrar novo hospital" and a list titled "Lista de Pontos Monitorados:" with items: teste, teste2, teste3, teste4, and teste5.

3.3.6 Equipamentos

Nesta página se encontram os registros de todas as máquinas cadastradas, nomeadas pelo seu número de série, e suas respectivas informações. Caso deseje encontrar uma máquina em específico basta procurar pelo seu número de série na barra de busca.

Se o usuário não estiver logado, ele poderá apenas visualizar as informações disponíveis. Se estiver, ele poderá editar qualquer informação que possa estar errada ou apagar alguma máquina que não está mais sendo usada. Pode, também, registrar uma nova utilizando o botão de registro, bastando apenas informar as informações necessárias.

Edição dos equipamentos:

Buscar número de série  

Cadastrar Equipamento

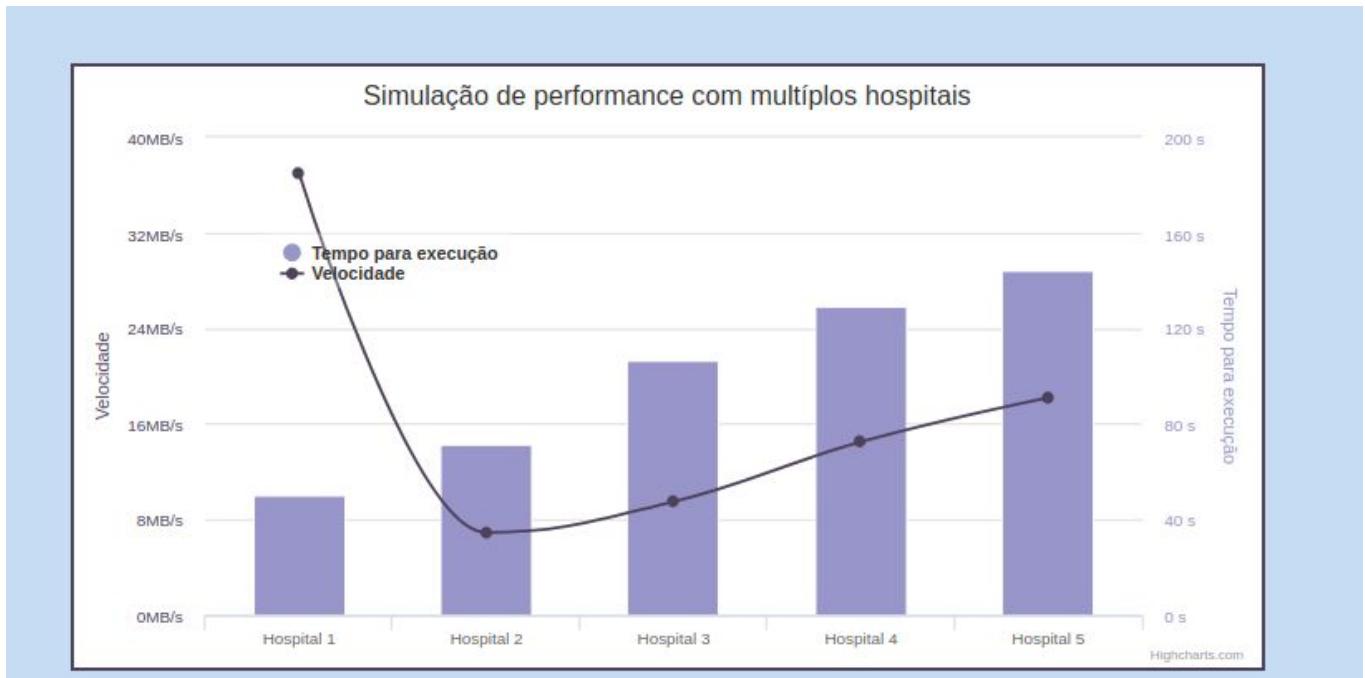


teste295029314

Mais Informações

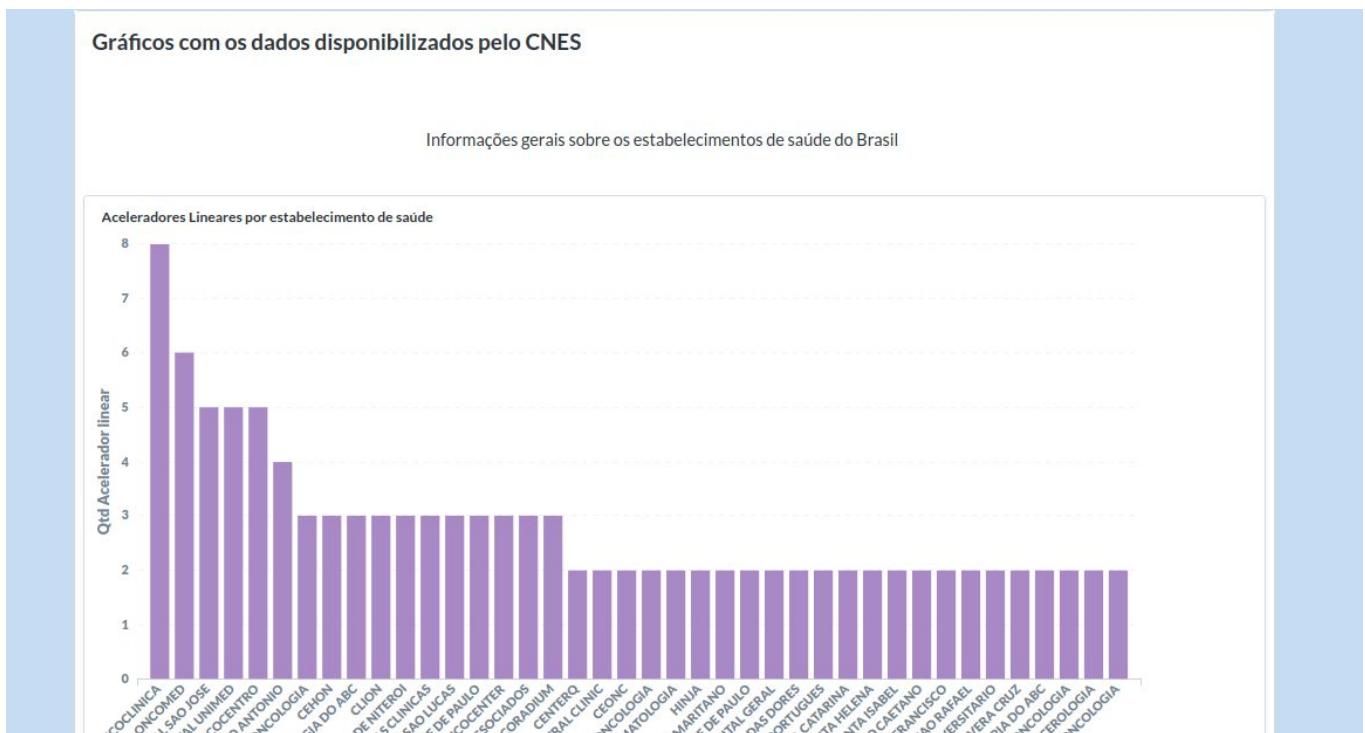
3.3.7 Métricas

Essa página possui um gráfico que demonstra o comportamento do sistema com múltiplos hospitais o alimentando com informações, o tempo que leva para completar as tarefas e a velocidade de transferência de dados



3.3.8 CNES

O CNES é uma plataforma que visa reunir todas as informações sobre o sistema de saúde brasileiro. Com a base de dados do CNES é possível encontrar informações sobre os estabelecimentos de saúde, equipamentos médicos, profissionais vinculados a um estabelecimento médico e informações ambulatoriais especializadas em diálise, oncologia e hemoterapia.



4 MOBILE

4.1 MOBILE ANDROID

4.1.1 Dependências

- Android Studio - Versão 20.04.1

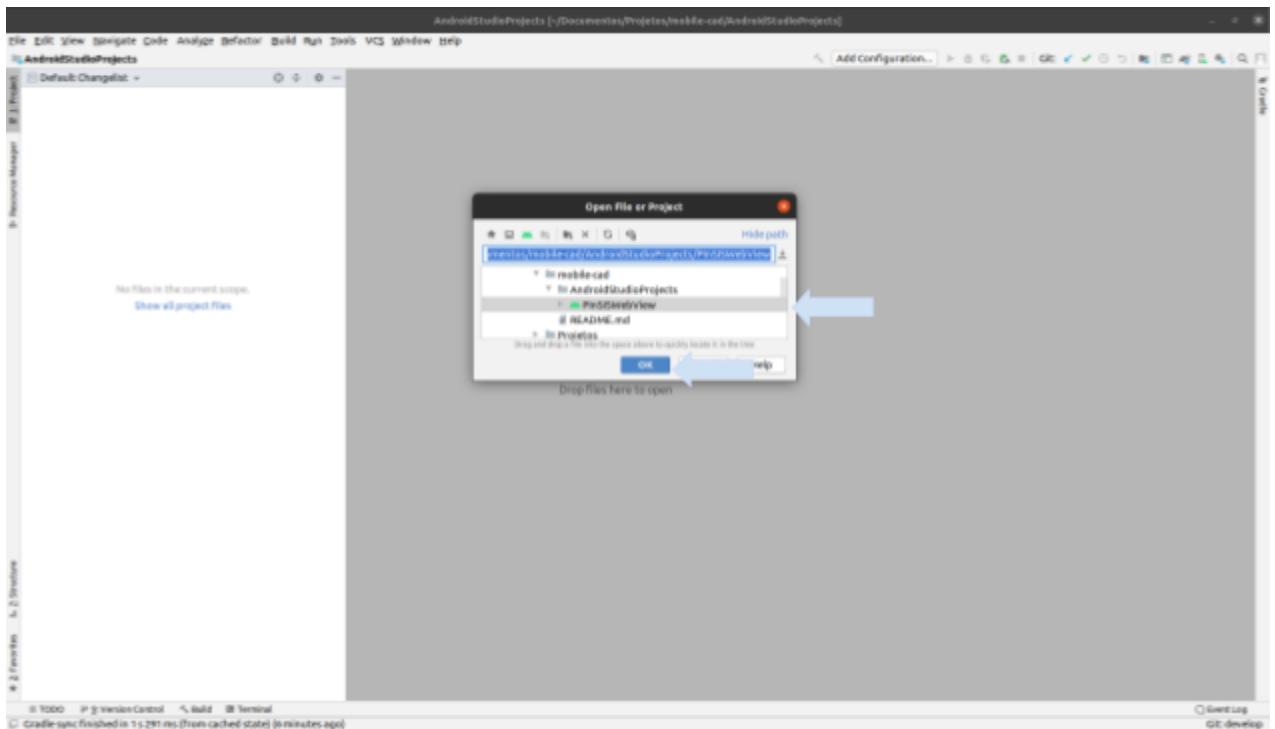
Para compilar a aplicação é preciso utilizar o aplicativo Android Studio, com um ambiente Debian. Os testes foram feitos com o Android Studio na versão 4.0.1 em um sistema Ubuntu 20.04.1 LTS, utilizando as versões do Android 8 e 9.

4.1.2 Instalação e Compilação

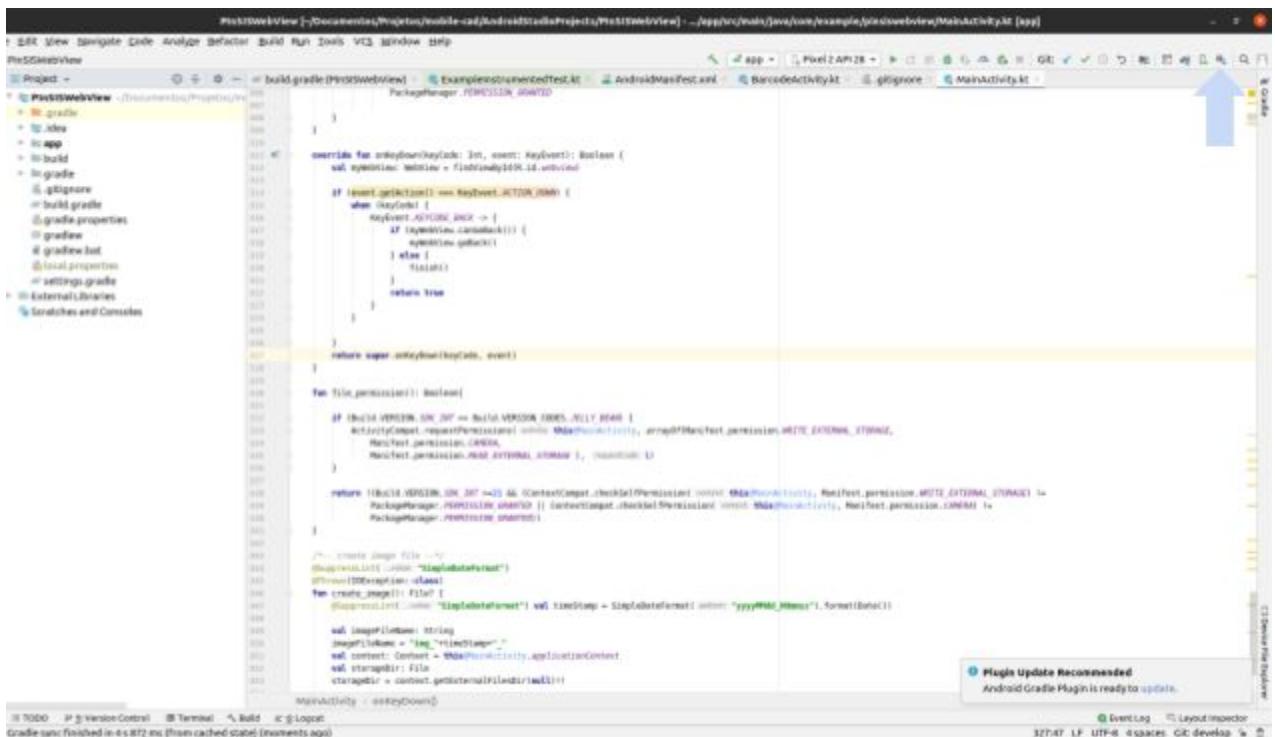
É necessário ter o projeto baixado, para isso é possível clonar ou realizar o download do repositório presente nesse [link](#).

Para importar e compilar o projeto no Android Studio siga os passos a seguir:

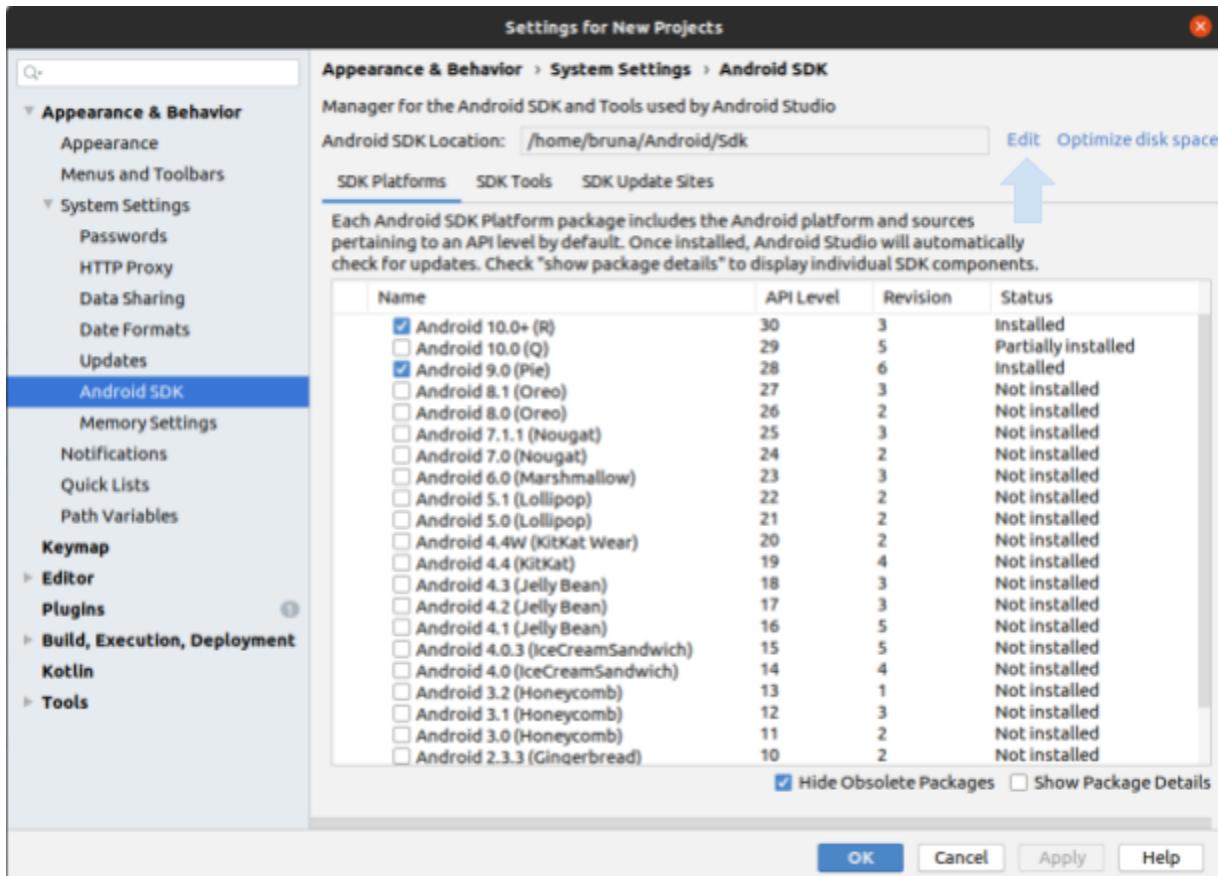
1. Clique em file na barra superior e abra a opção “Open”, selecione o projeto e clique em “OK”:



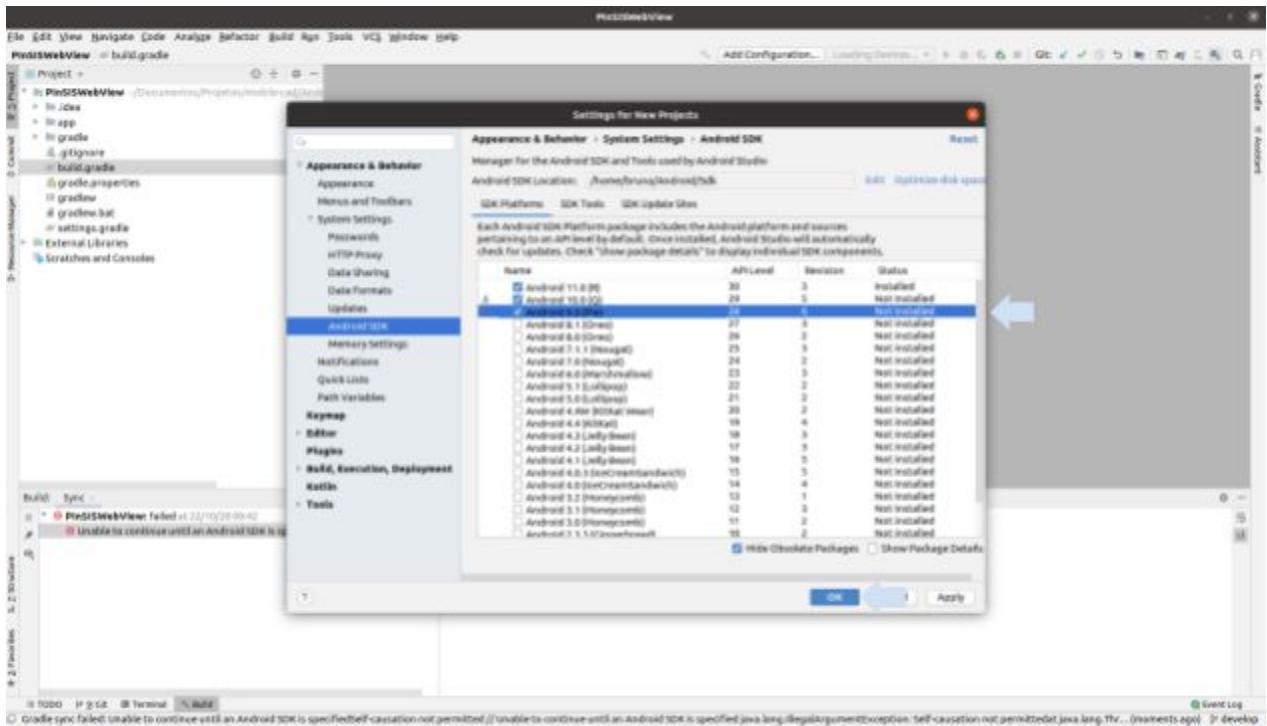
2. Abra o SDK Manager:



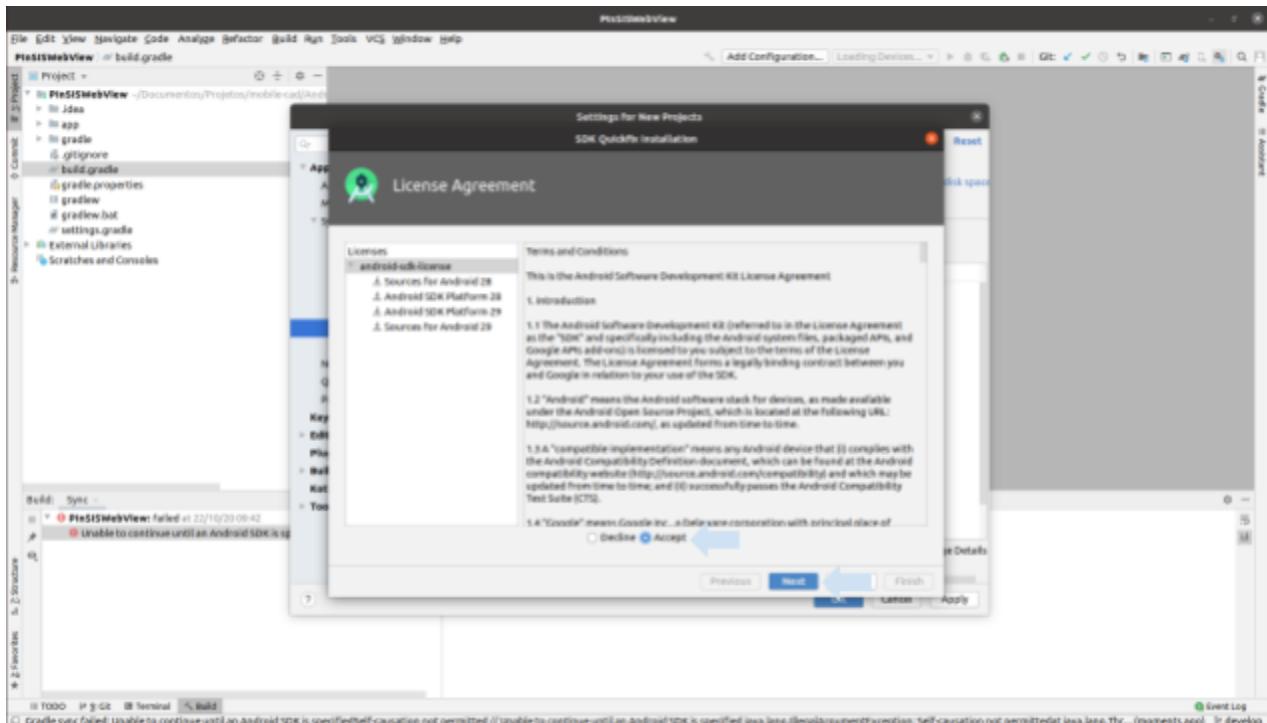
3. Clique em “Edit”:



4. Confira a localização para o SDK e clique no botão “next”;
5. Aguarde a instalação e finalize após o download ser realizado;
6. Selecione as versões do android desejadas e clique em OK:



7. Aceite a licença e clique em “Next”:



8. Aguarde o download dos componentes e clique em finalizar;

9. Clique no botão “Sync Project with Gradle Files” e aguarde o projeto ser construído:

MainActivity.java

```
package com.example.mainactivity; // Generated by Android Studio

import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    private WebView webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        webView = findViewById(R.id.webView);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("http://192.168.1.7:8080");
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            if (webView.canGoBack()) {
                webView.goBack();
            } else {
                finish();
            }
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
        if (requestCode == Build.VERSION.SDK_INT >= Build.VERSION_CODES.M ? ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED : ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
            if (grantResults.length > 0) {
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M ? ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED : ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
                        // Permission granted
                    } else {
                        // Permission denied
                    }
                }
            }
        }
    }

    private void createImageFile() {
        String timestamp = String.valueOf(System.currentTimeMillis());
        String date = String.valueOf(timestamp).substring(0, 10);
        String file = "Screenshot_" + date + ".png";
        File storageDir = new File(Environment.getExternalStorageDirectory(), "Screenshots");
        if (!storageDir.exists()) {
            storageDir.mkdirs();
        }
        File imageFile = new File(storageDir, file);
        try {
            FileOutputStream fos = new FileOutputStream(imageFile);
            fos.write(webView截断
```

10. A mensagem “BUILD SUCCESSFUL” indica que a construção foi feita corretamente:



Para inicializar o webview pelo local host, é necessário substituir a url "<https://pinsis.c3sl.ufpr.br/>" da função loadUrl pela url localhost:port no arquivo

MainActivity.kt, selecionado na imagem abaixo. Devido ao sistema de segurança presente no mobile seu funcionamento depende de estar rodando em “https”. Isso pode ser alcançado adicionando a flag “`--ssl true`” ao inicializar o `server`. Veja a seção 3 para maiores instruções.



```
135  
136  
137 myWebView.webViewClient = WebClient()  
myWebView.loadUrl(url: "https://pinsis.c3sl.ufpr.br/")
```

Logo em seguida, no mesmo arquivo, a função `onReceivedSslError` é necessária apenas para o desenvolvimento, para que o https seja aceito mesmo sem um certificado assinado. Usando a url “<https://pinsis.c3sl.ufpr.br/>” ou equivalente a função não é necessária pois contém o certificado assinado, e é preciso retirar a função por ser considerado uma vulnerabilidade de segurança.



```
139  
140  
141 //to accept https without sign certificate  
myWebView.webViewClient = object : WebClient(){  
    override fun onReceivedSslError(  
        view: WebView?,  
        handler: SslErrorHandler?,  
        error: SslError?  
    ) {  
        handler?.proceed()  
    }  
148  
149 }
```

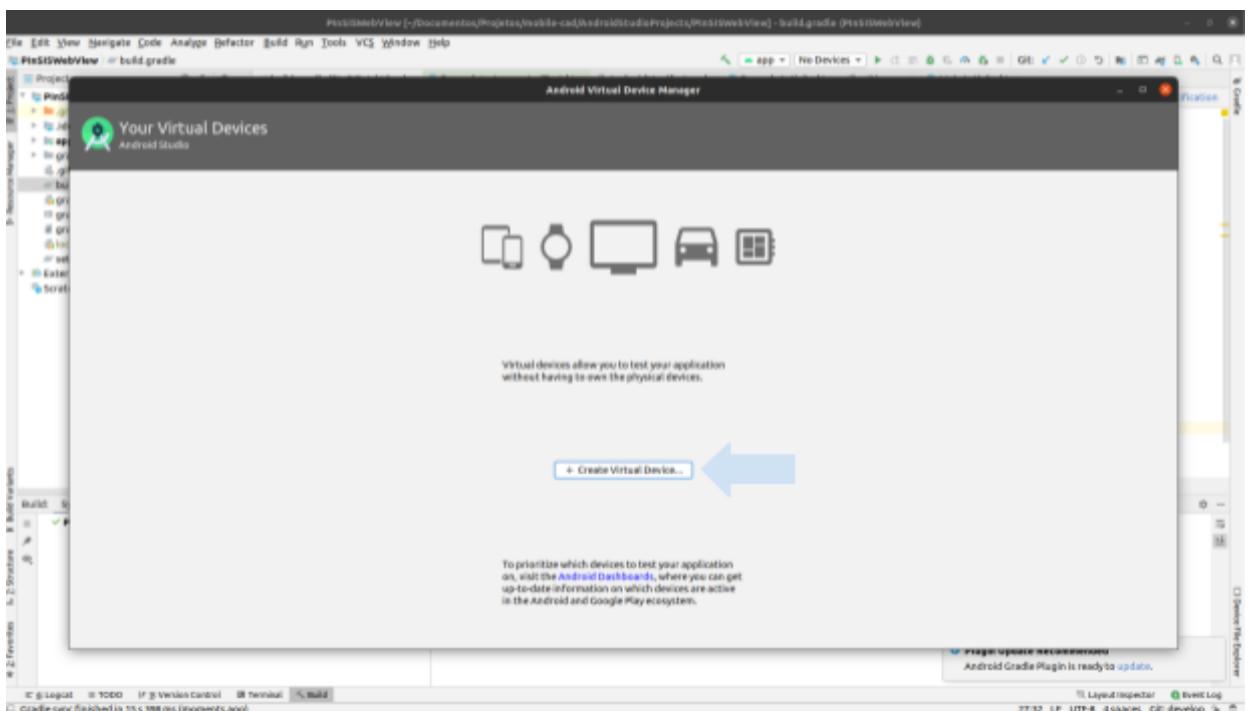
4.1.3 Usabilidade PinSIS mobile android

Para inicializar a emulação da aplicação, é explicado nos passos a seguir.

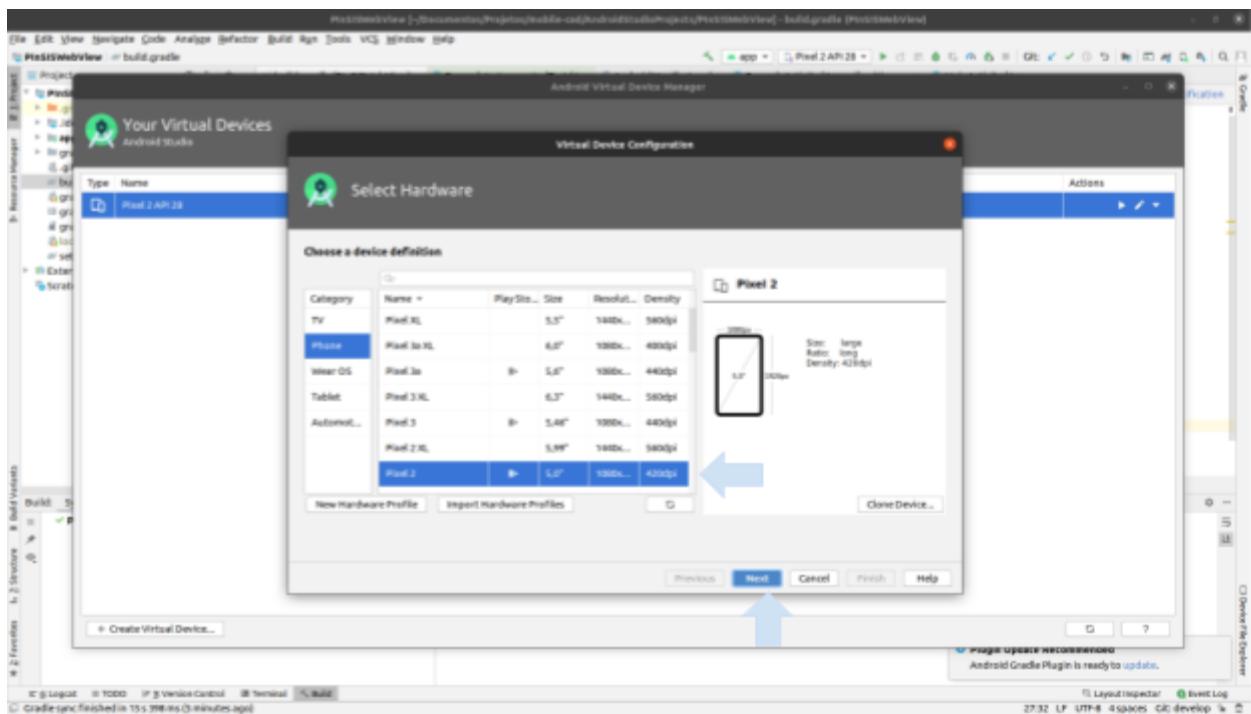
1. Abra o AVD Manager:

```
buildscript {  
    ext.kotlin_version = "1.3.40"  
    repositories {  
        google()  
        jcenter()  
    }  
}  
  
dependencies {  
    classpath "com.android.tools.build:gradle:3.3.4"  
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.3.40"  
    // NOTE: Do not place your application dependencies here; they belong  
    // in the individual module build.gradle files  
}  
  
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```

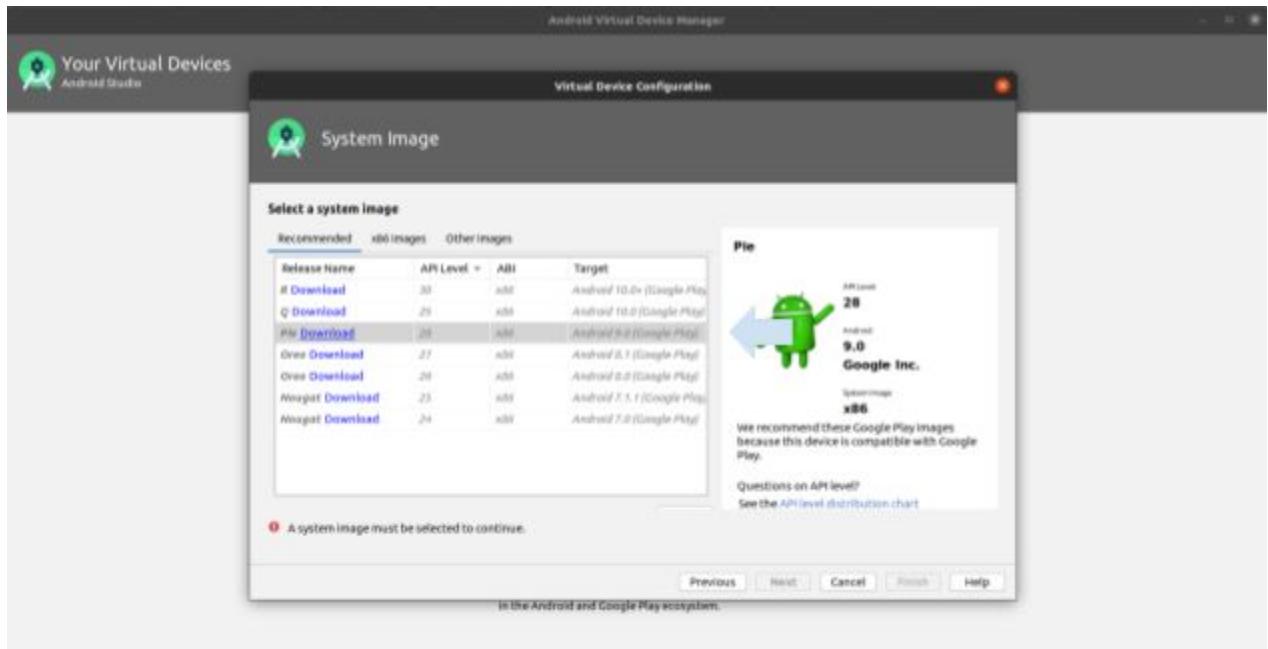
2. Clique no botão “Create Virtual Device”



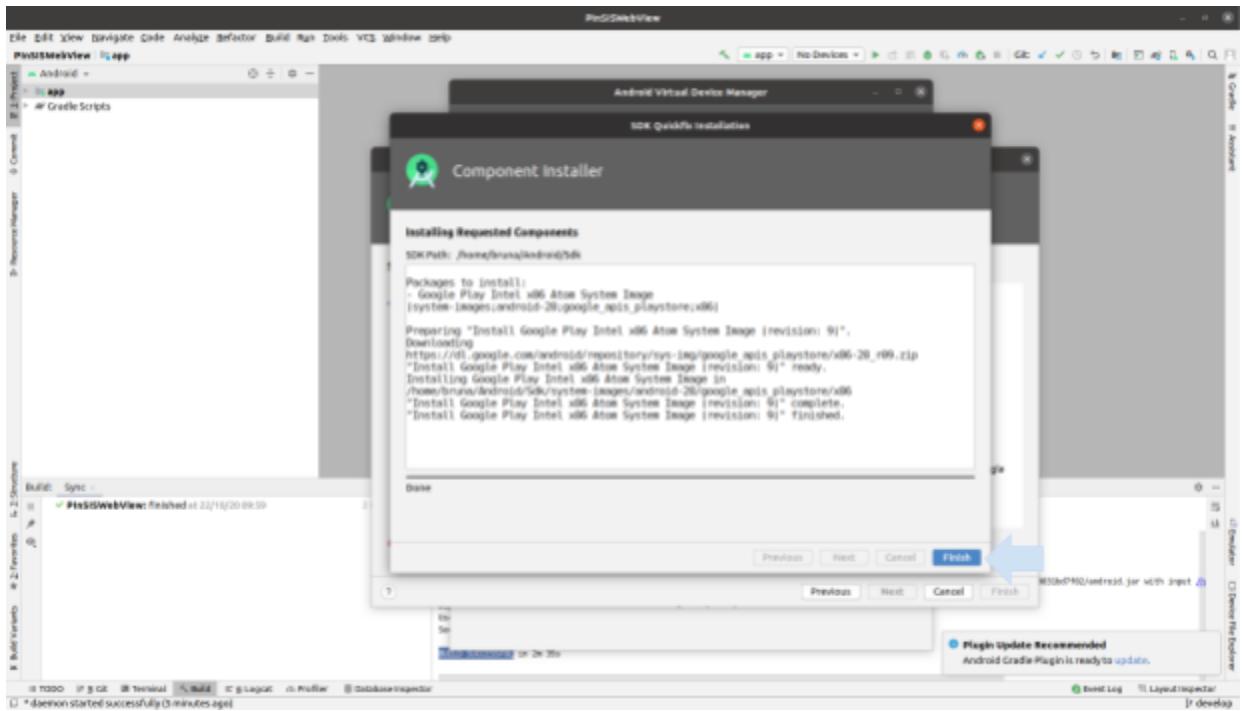
3. Selecione o hardware e clique em “next”:



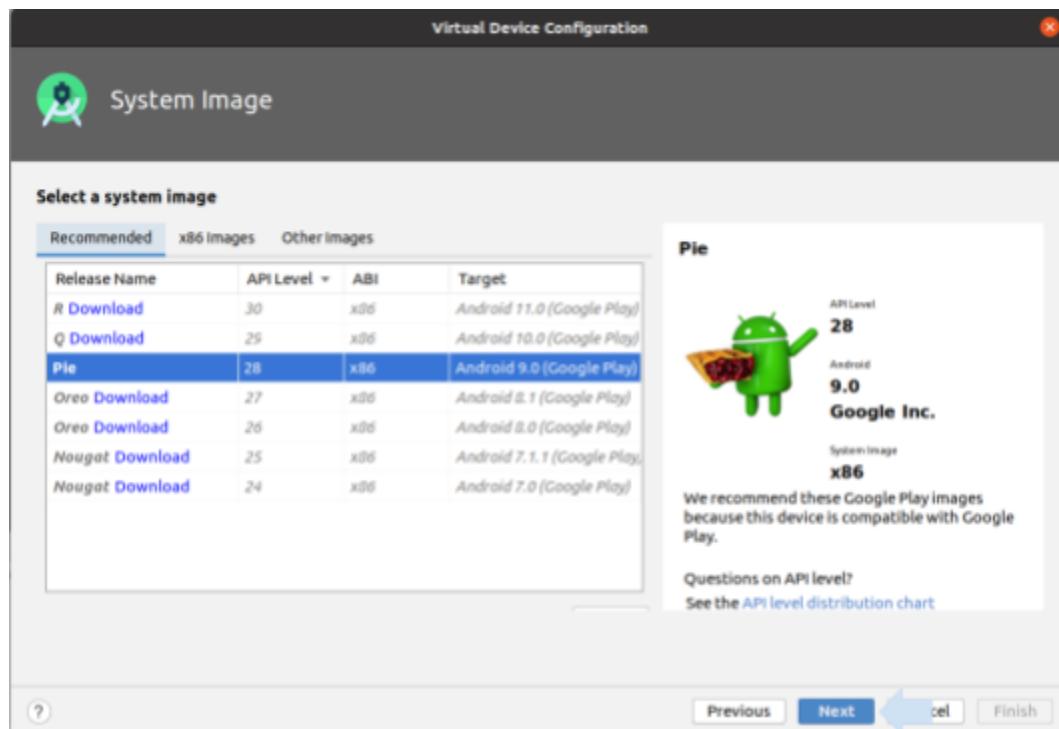
4. Selecione a versão do android (os testes foram realizados nas versões 8 e 9) e clique em “download” ao lado da versão do Android:



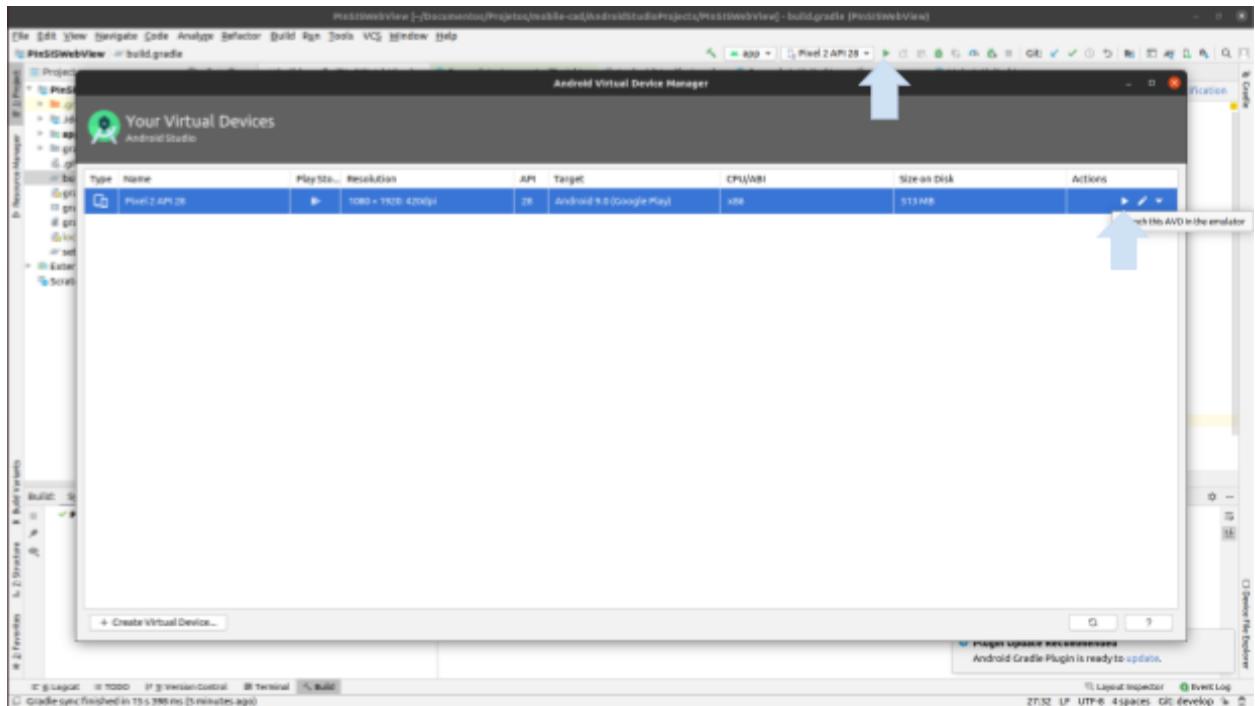
5. Aceite a licença e aguarde a instalação dos componentes, finalizando em seguida:



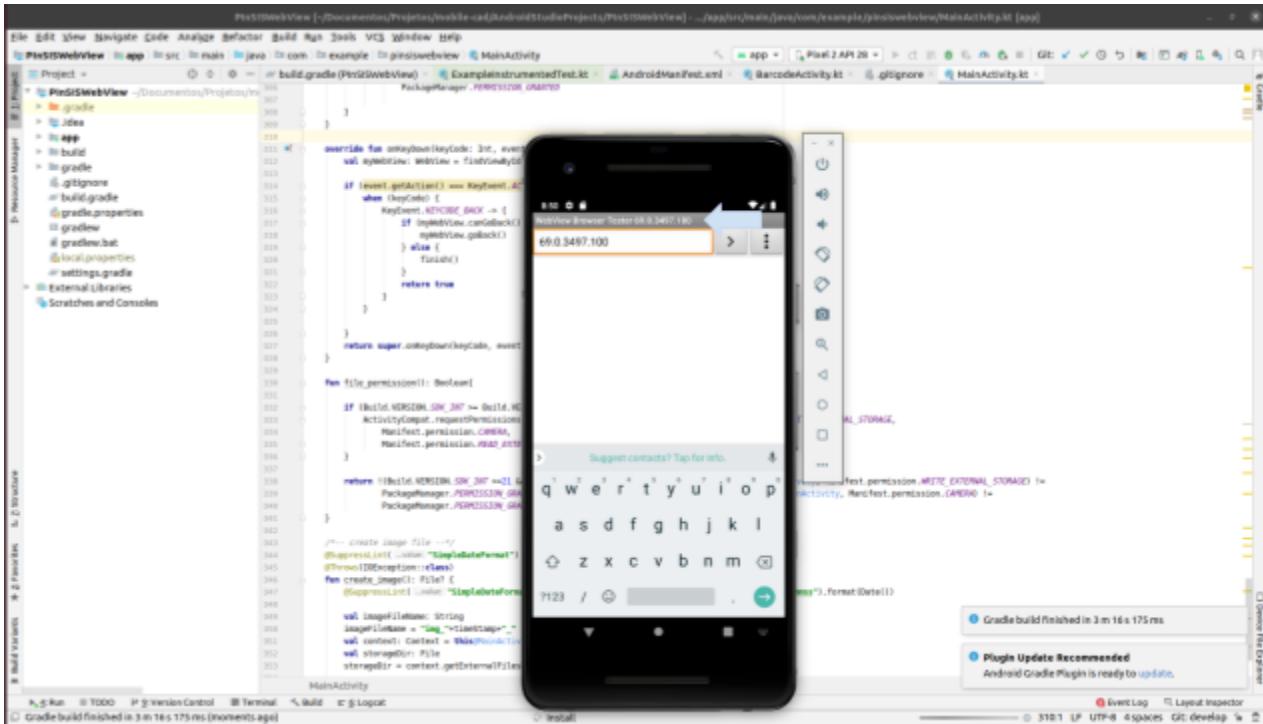
6. “next” em seguida:



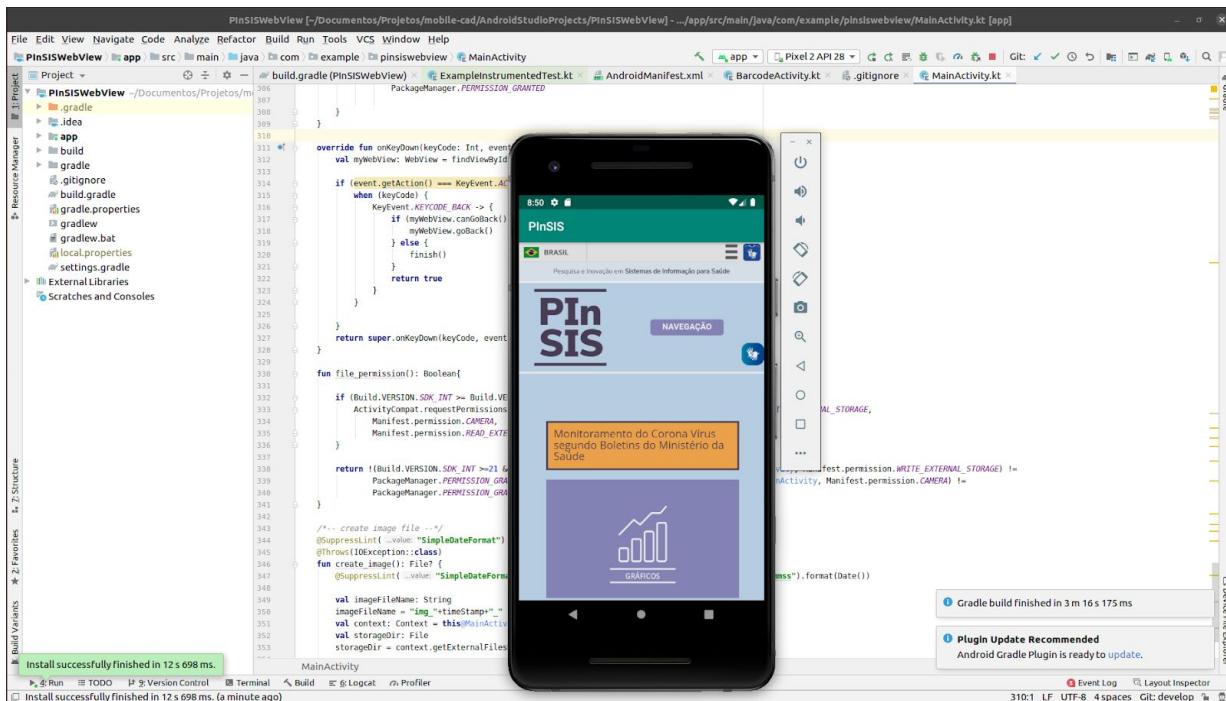
7. Após a instalação ser finalizada, é possível iniciar a emulação abrindo o AVD Manager e clicando no play no dispositivo desejado, ou selecioná-lo na barra superior do Android Studio e clicar em play ao lado das opções:



8. O emulador irá realizar a inicialização, quando terminar, abra o aplicativo webView e insira os números presentes na parte superior da aplicação na barra de search:

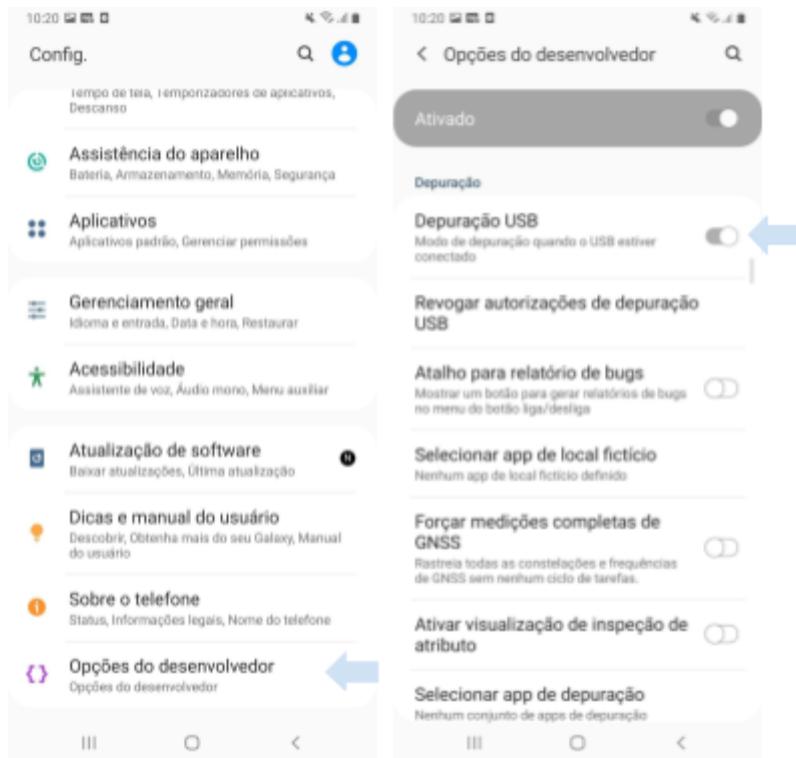


9. O projeto será compilado e a aplicação está pronto para o uso:

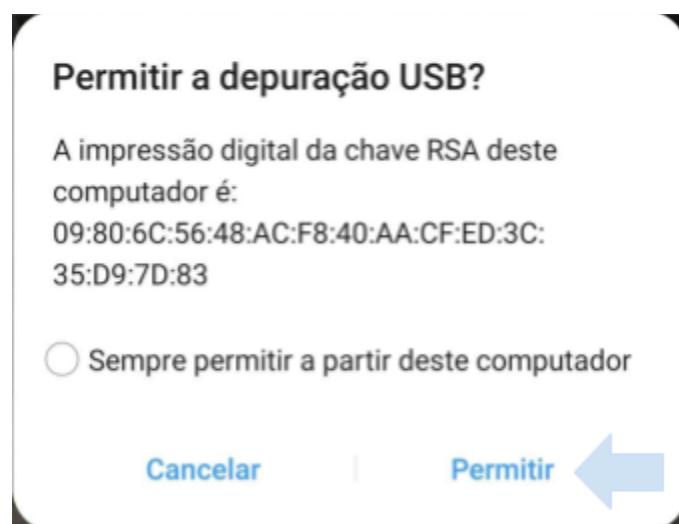


Para realizar a inicialização do webview por um dispositivo android físico:

1. No dispositivo mobile android, entre em configurações e na Opções de Desenvolvedor e habilite a Depuração USB:



2. Conecte o dispositivo por um cabo USB com o AndroidStudio aberto;
3. Permita a depuração USB:



4. Selecione o dispositivo no AndroidStudio e clique em play ao lado para iniciar aplicação:

```
buildscript {
    ext.kotlin_version = '1.3.41'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.3.0'
        classpath 'com.google.gms.google-services'
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

4.2 MOBILIZAR

Essa sessão tem como objetivo apresentar os requisitos necessários para compilar/executar a aplicação mobile-ios em um ambiente de testes.

4.2.1 Dependências

O código da aplicação iOS do PinSIS tem somente uma dependência, que já está importada no código do repositório. A dependência é o projeto CodeScanner (<https://github.com/twostraws/CodeScanner>) que possibilita o escaneamento de QR codes. Essa dependência possui uma licença MIT e portanto pode ser usada para fins comerciais sem problemas.

Para compilar a aplicação, é necessária a instalação do aplicativo XCode e um ambiente rodando sistema Mac OS X. Os testes foram feitos com a aplicação compilada utilizando a versão **12.0.1** em um sistema Mac OS X rodando na versão **10.15.6 macOS Catalina**.

Sumarizando a lista de dependências:

- CodeScanner (<https://github.com/twostraws/CodeScanner>) - Contido no repositório do código da aplicação mobile-ios
- XCode - Versão 12.0.1
- Mac OS X - Versão macOS Catalina 10.15.6

4.2.2 Instalação/Compilação

Esta seção irá cobrir os procedimentos necessários para poder compilar a aplicação bem como ter uma versão funcional da aplicação rodando em uma instância emulada de um dispositivo móvel iOS. É necessário que você tenha na sua máquina local as dependências explicitadas na seção ‘Dependências’ para prosseguir com os próximos passos.

A instalação do sistema Mac OS X está fora do escopo desta documentação. Para a instalação do XCode, basta acessar a App Store, no sistema Mac OS X e procurar por ‘XCode’.

Para importar o projeto da aplicação mobile para o XCode você primeiro precisa ter o código da aplicação localmente em sua máquina. Considerando que o código estará em um repositório Git, basta clonar o repositório em sua máquina. Ao final desta etapa, para importar o projeto no XCode basta seguir os seguintes passos:

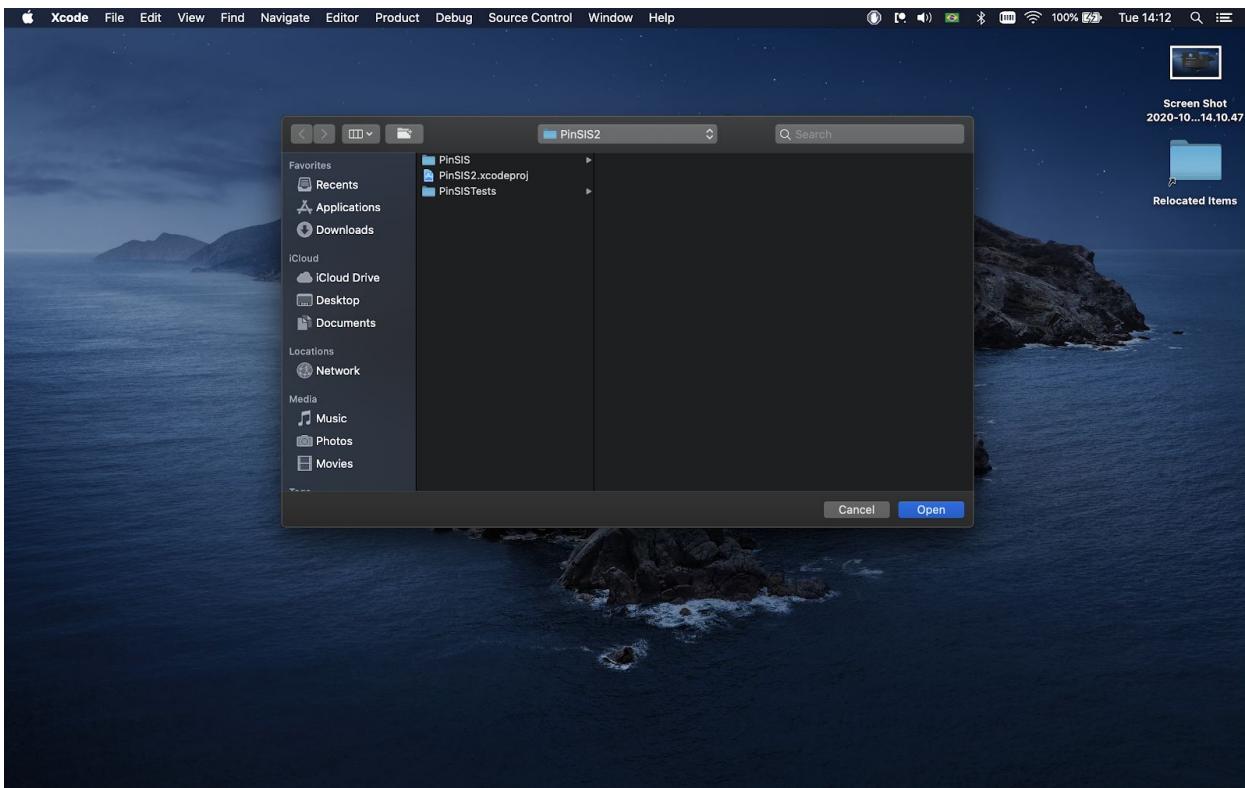
1 - Abra o XCode



2 - Selecione a opção 'Open a Project or File'



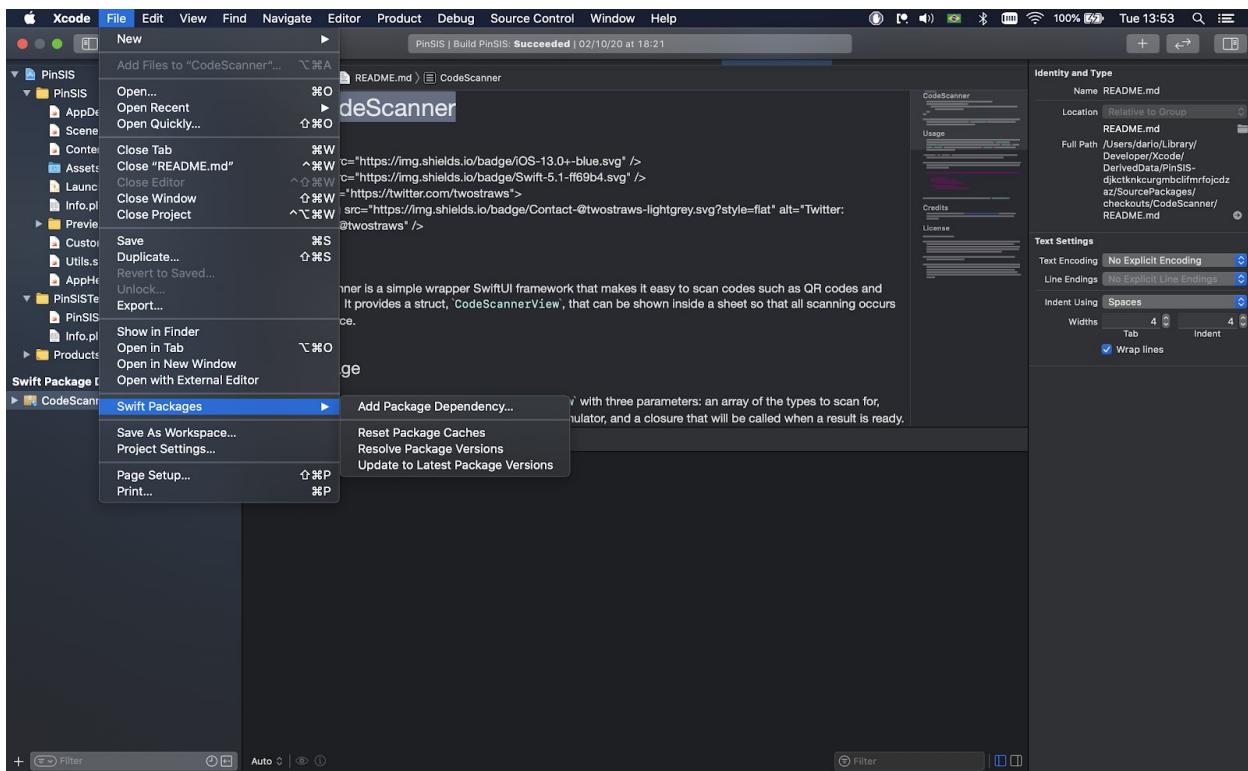
3 - Selecione a pasta que contém o código que foi clonado do repositório Git e de 'Open'



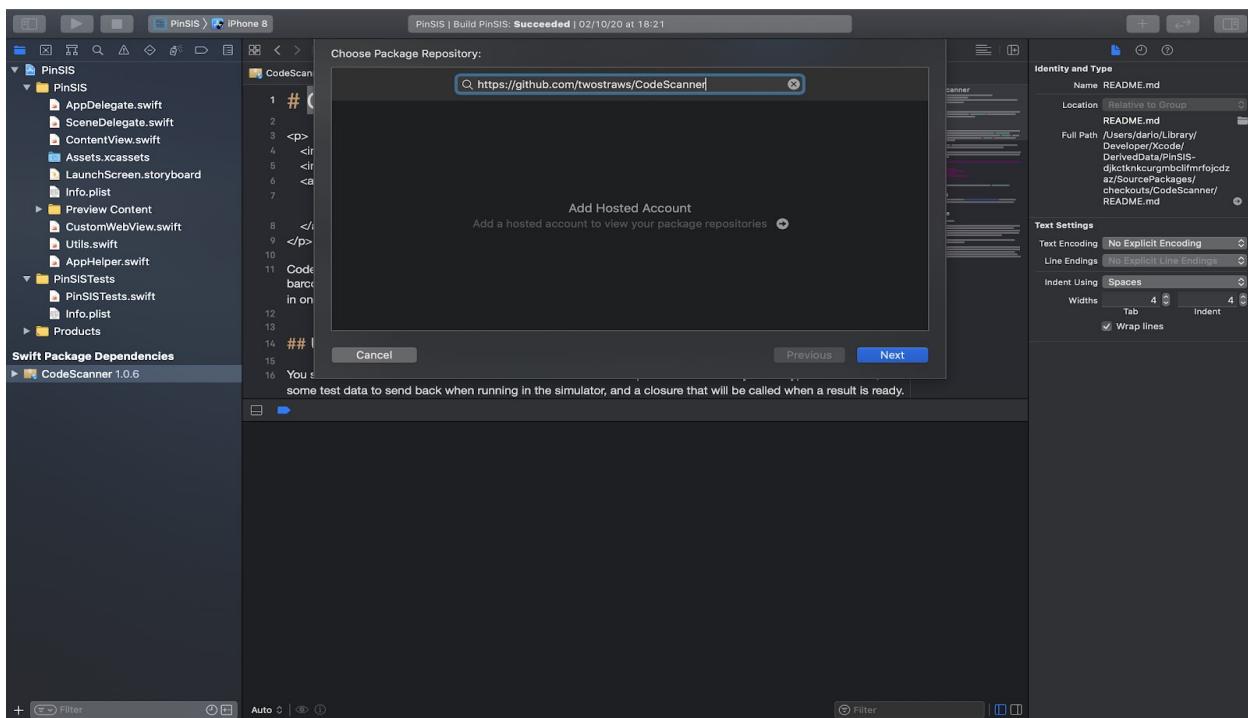
Ao final destes passos você terá importado o projeto com sucesso.

A dependência CodeScanner deve estar incluída dentro do repositório, mas caso não esteja, basta adicioná-la usando os seguintes passos:

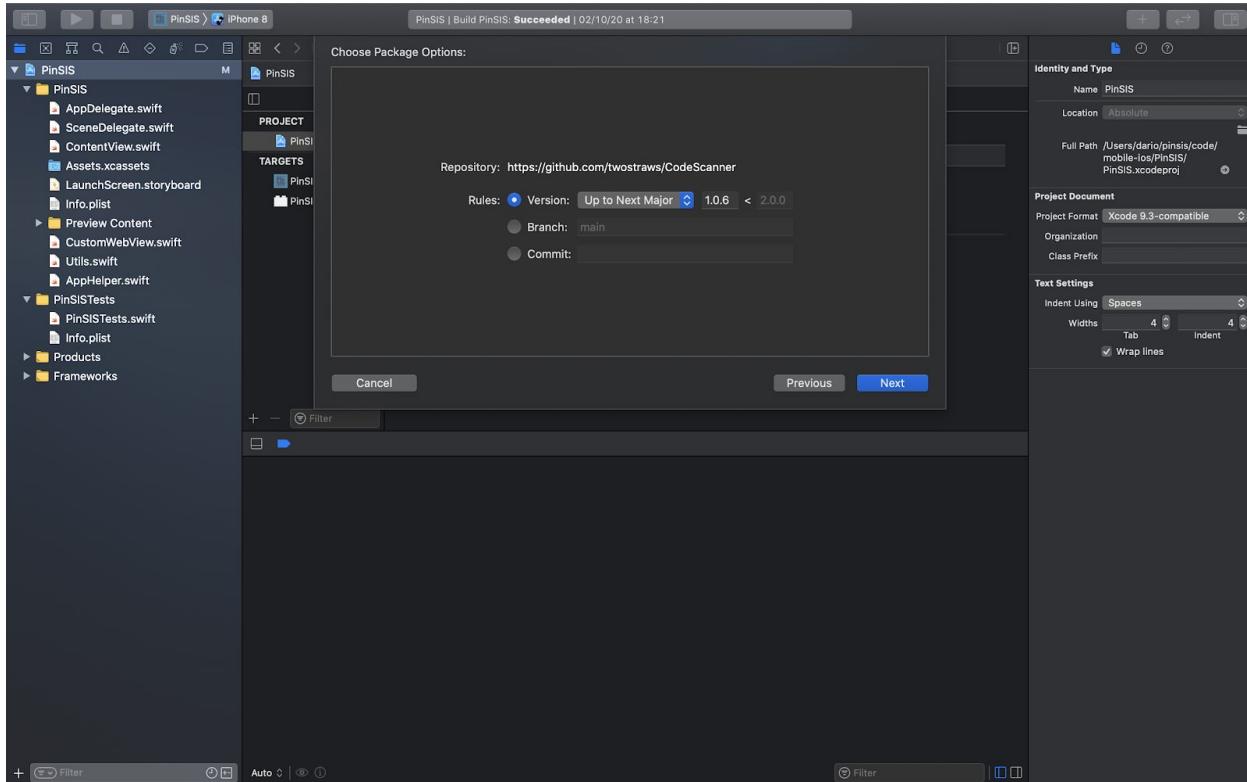
- 1 - Copie o link do repositório no Github - <https://github.com/twostraws/CodeScanner>
- 2 - Abra o projeto PinSIS no XCode
- 3 - Clique na barra superior em 'File' e vá até 'Swift Packages -> Add Package Dependency'



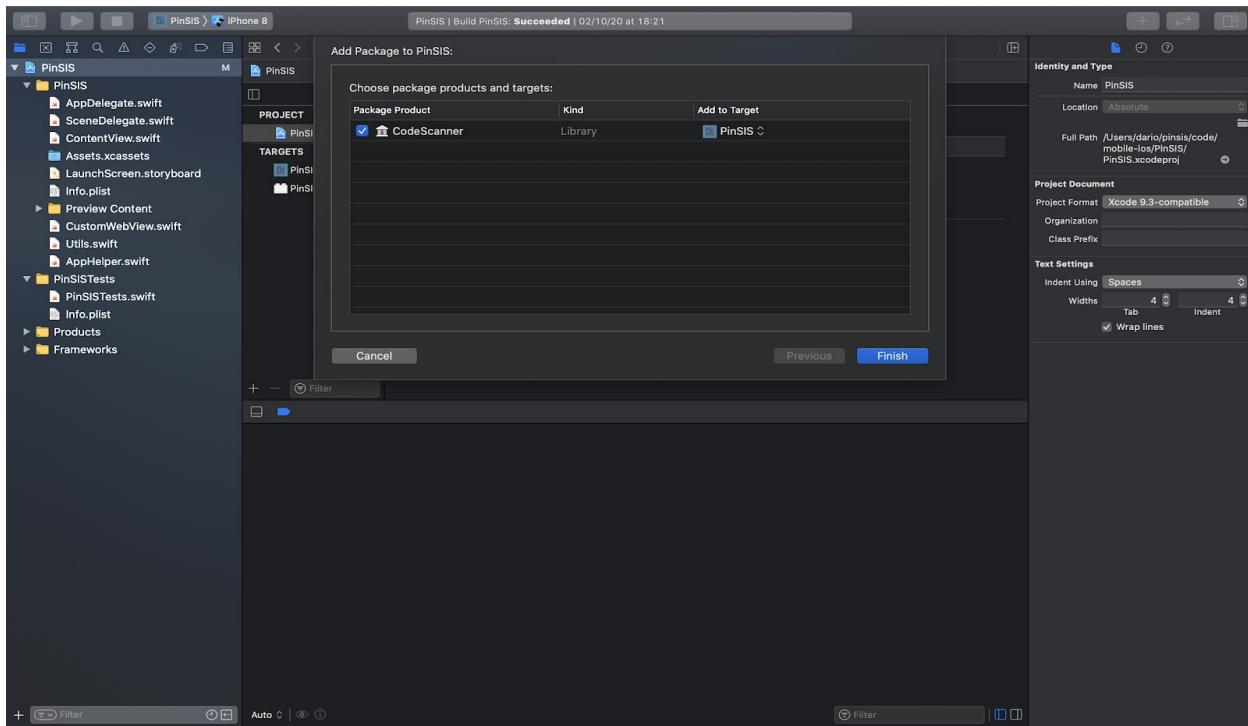
4 - Cole o link do repositório Github que foi copiado no passo 1 e de 'Next'



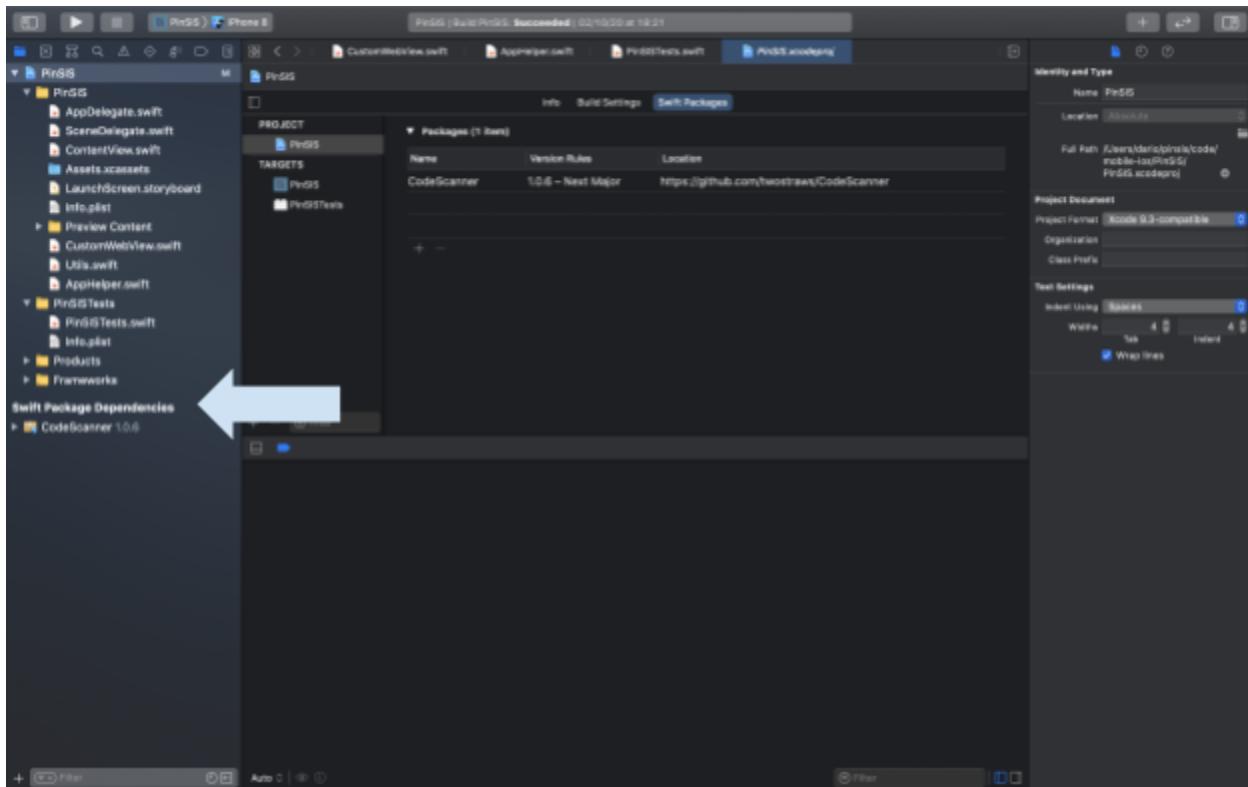
5 - De 'Next' novamente. Ele vai selecionar como dependência a última versão disponível que é a correta.



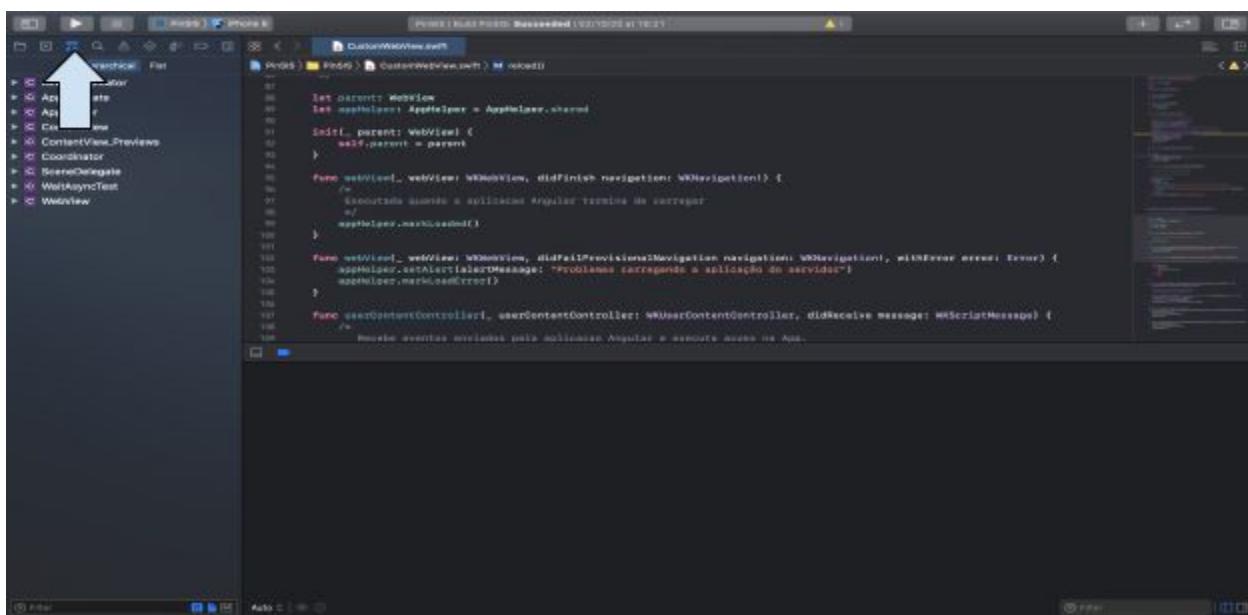
6 - De 'Finish' para adicionar a dependência ao projeto



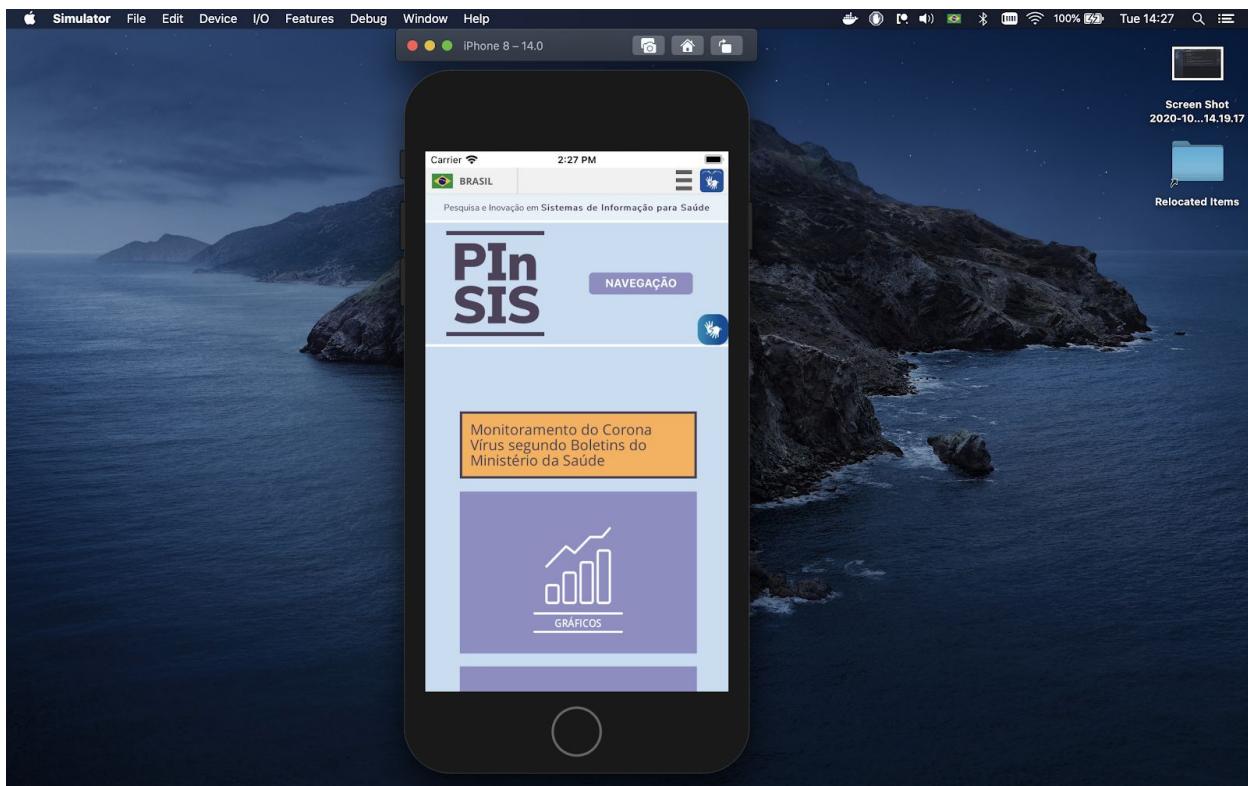
Ao final, a dependência deve ficar visível na parte dos arquivos embaixo, sobre o nome 'Swift Package Dependencies'.



Com todas as dependências instaladas e o código importado, podemos prosseguir para a etapa de compilação. Existem dois modos de compilar o código, o primeiro é através das teclas ‘Command + R’ e a outra é através da interface gráfica, clicando no botão ‘Play’.



Clicar ou no botão Play ou executar a sequência de teclas, o projeto será compilado pelo XCode e uma execução do aplicativo será iniciada no emulador. Ao final, você terá o programa rodando na instância emulada, pronto para uso.



4.2.3 Usabilidade PinSIS mobile iOS

Este documento apresenta como utilizar o aplicativo mobile do PinSIS para sistema iOS. Todos os cenários apresentados aqui foram feitos utilizando um device **iPhone 8** emulado rodando iOS versão **14.0**. O emulador do dispositivo foi provido pelo aplicativo XCode (emulador nativo).

Nas seções seguintes serão apresentadas as possíveis formas de usar o aplicativo mobile de iOS do PinSIS

Os testes foram feitos utilizando uma versão local do servidor do PinSIS, pois era preciso colocar dados não verdadeiros (de teste).

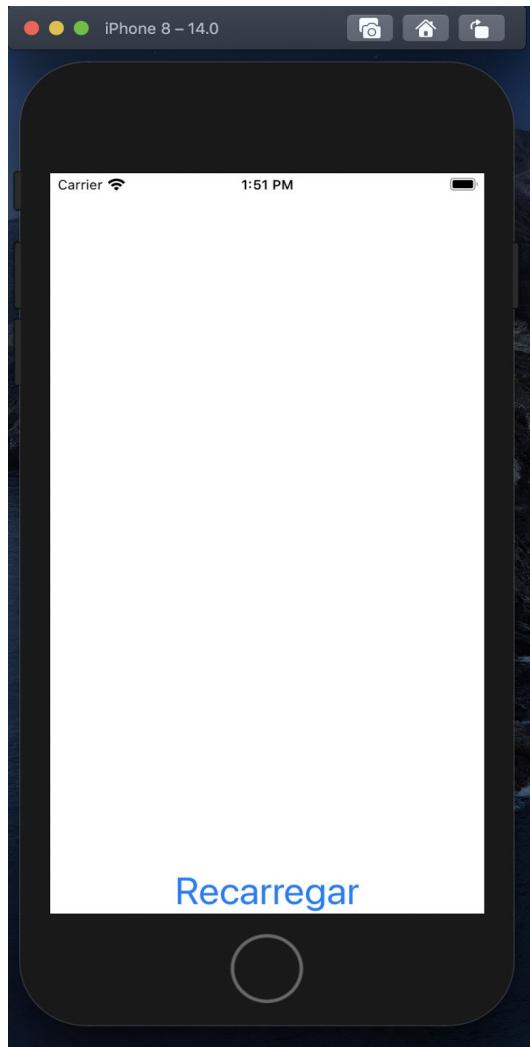
Outros detalhes de usabilidade são cobertos na seção 4.3

4.2.4 Carregar/Recarregar portal PinSIS

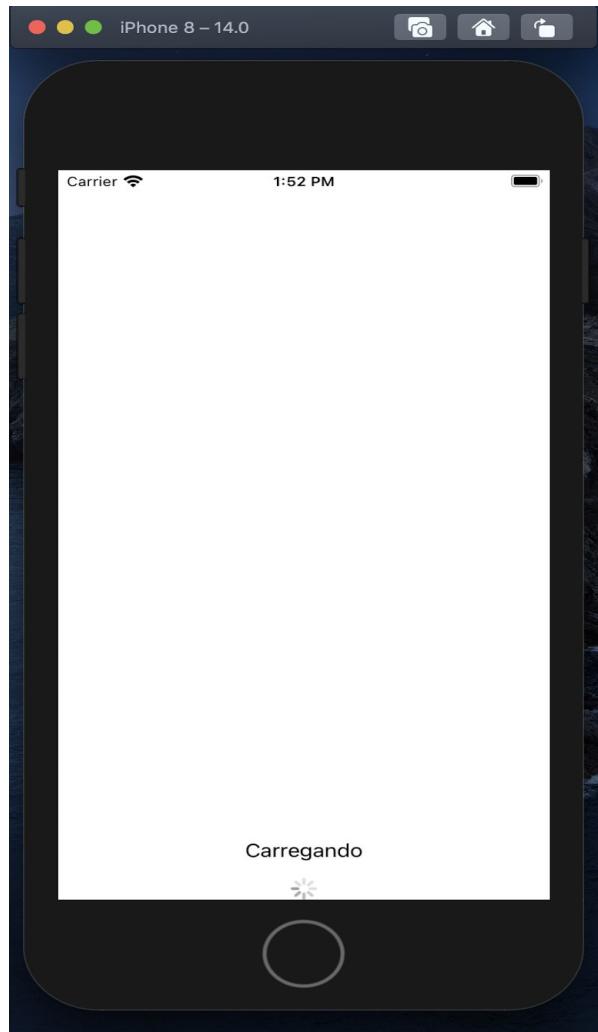
Ao acessar o aplicativo do PinSIS, a aplicação irá conectar-se ao servidor que está rodando o portal do PinSIS. Caso o servidor não esteja rodando, uma mensagem de erro será exibida na tela



E o botão ‘Recarregar’ ficará disponível para clicar. Ao clicar neste botão, a aplicação irá tentar novamente acessar o portal do PinSIS.



Sempre que a aplicação está tentando acessar o portal, a mensagem 'Carregando' será exibida na base da tela



Ao final do carregamento com sucesso, a tela inicial do portal para a versão mobile será exibida.

4.3 USABILIDADE

Essa seção contém detalhes sobre a usabilidade de cada funcionalidade presente na aplicação.

4.3.1 Tela Inicial

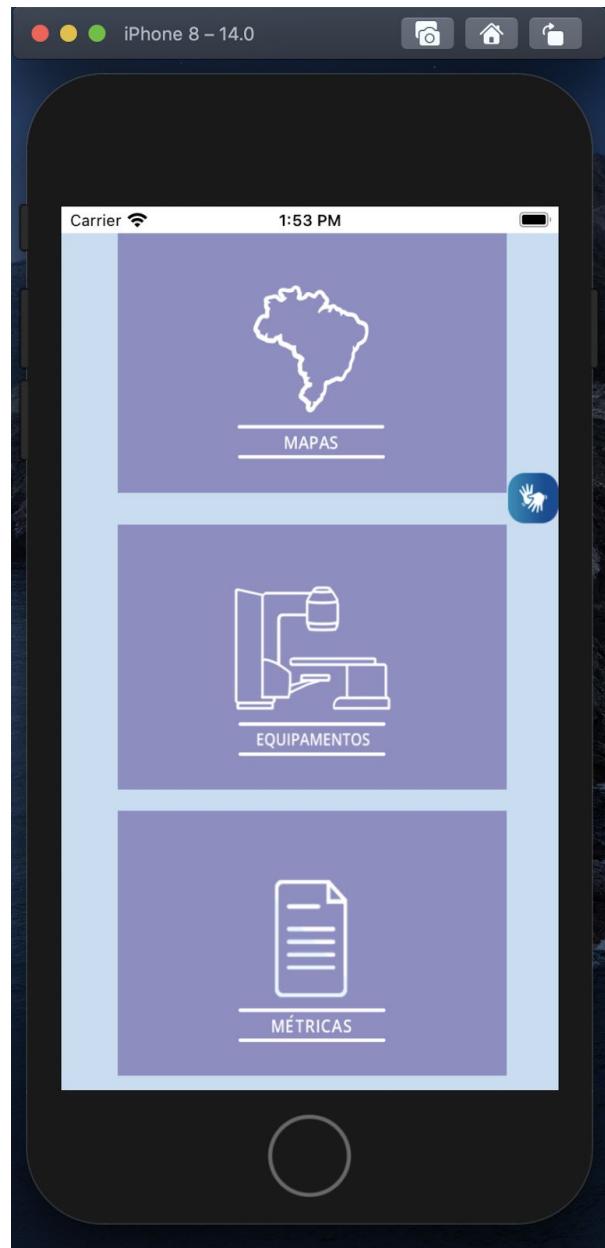
Após carregar as informações do PinSIS do portal, a aplicação irá exibir a tela inicial (muito semelhante a versão Web, porém com uma disposição melhor para rodar em dispositivos móveis).



Note que existe um botão chamado ‘NAVEGAÇÃO’. Este botão tem basicamente as mesmas funcionalidades da versão web, porém condensadas em um menu para providenciar uma melhor experiência em dispositivos móveis.



É possível dar scroll pela tela para ver outras funcionalidades principais do portal também como ‘Mapas’ que mostra um mapa com a localidade dos hospitais cadastrados, ‘Equipamentos’ que mostra a lista de equipamentos cadastrados, ‘Gráficos’ que mostra informações gráficas dos dados de usabilidade dos equipamentos e ‘Métricas’ que mostra uma simulação gráfica com o uso de múltiplos hospitais.

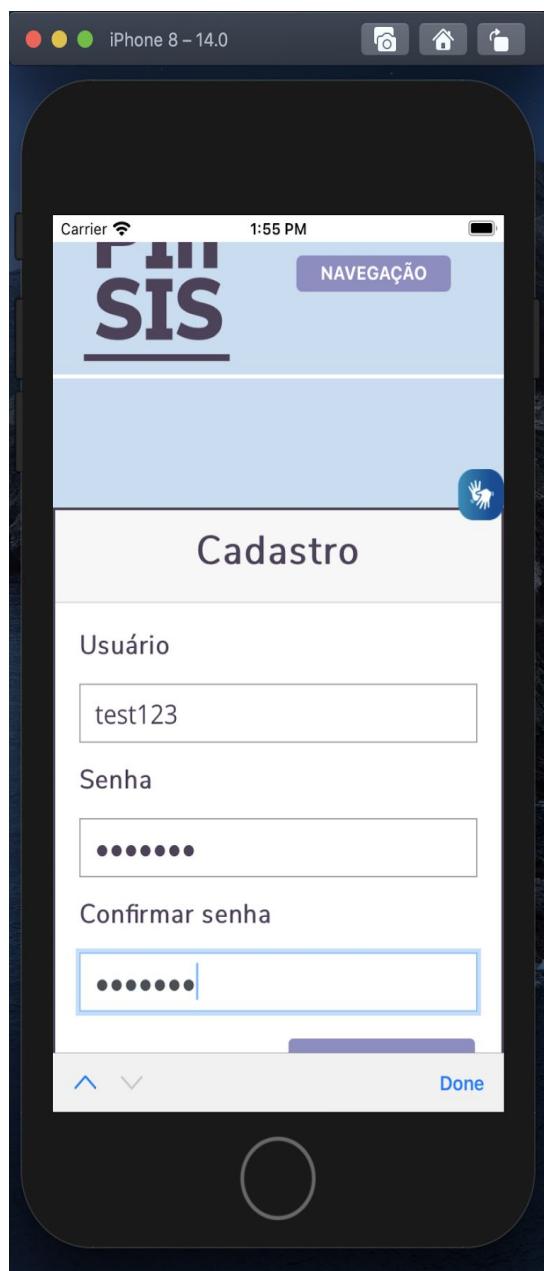


4.3.2 Cadastro de usuário

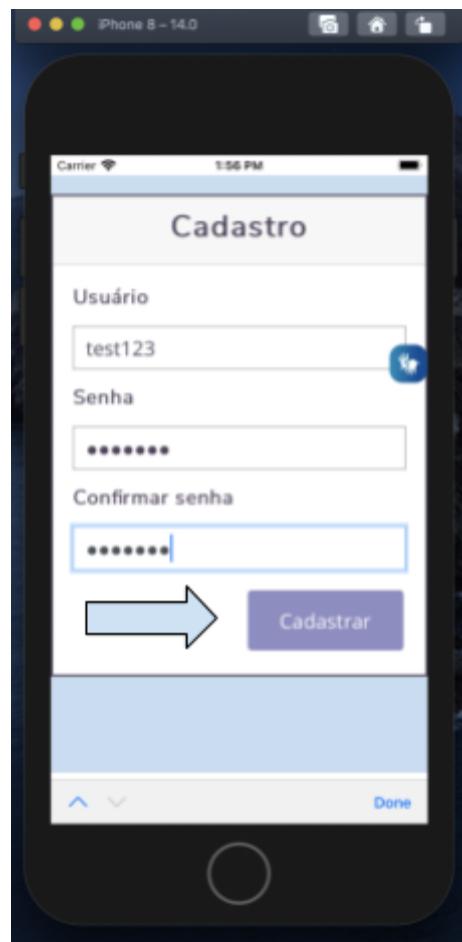
Para utilizar algumas funcionalidades do sistema como cadastro de equipamentos ou hospitais é preciso estar logado no Portal. Para cadastrar um novo usuário, basta clicar na opção ‘Cadastro’ no menu ‘Navegação’



Ao clicar na opção uma nova tela onde os dados do novo usuário deverão ser preenchidos irá abrir.



Ao final do preenchimento dos detalhes, basta clicar no botão cadastrar e o usuário será cadastrado no sistema



4.3.3 Login de usuário

Com um usuário cadastrado no sistema, é necessário fazer o login para utilizar as funcionalidades mencionadas anteriormente (cadastro de equipamento e hospital). Para realizar o login, basta clicar em 'Login' em 'Navegação'



Digitar os dados do usuário e clicar em Entrar



4.3.4 Logout de usuário

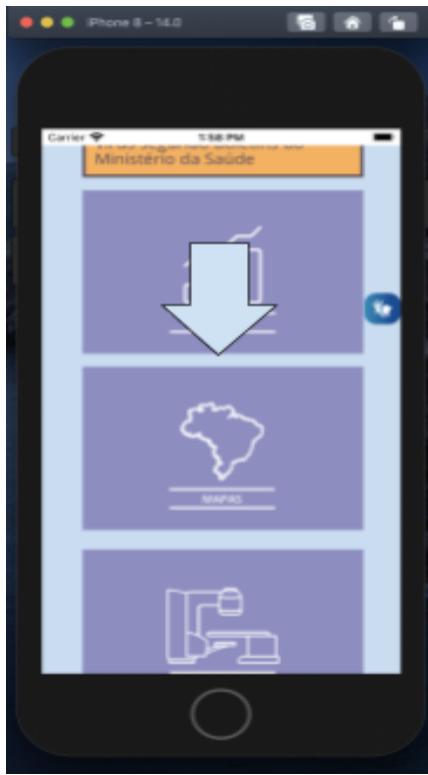
Com o usuário logado, o menu de navegação agora não oferece mais a opção de login, mas ao invés disso, a opção de logout aparece.



Ao clicar nesta opção o usuário é automaticamente deslogado do portal e é necessário efetuar o login novamente.

4.3.5 Cadastro de hospital

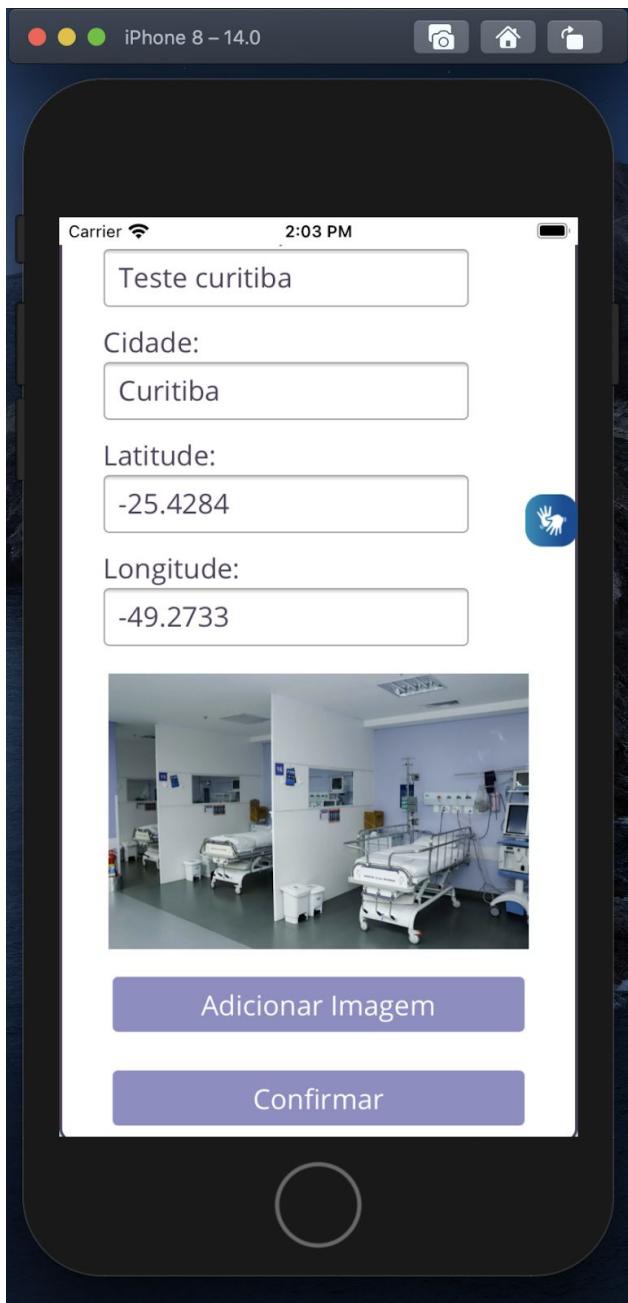
Para obter informações sobre um determinado hospital é necessário primeiro que ele esteja cadastrado no sistema. Para cadastrar um hospital é necessário primeiro selecionar a opção 'Mapas'



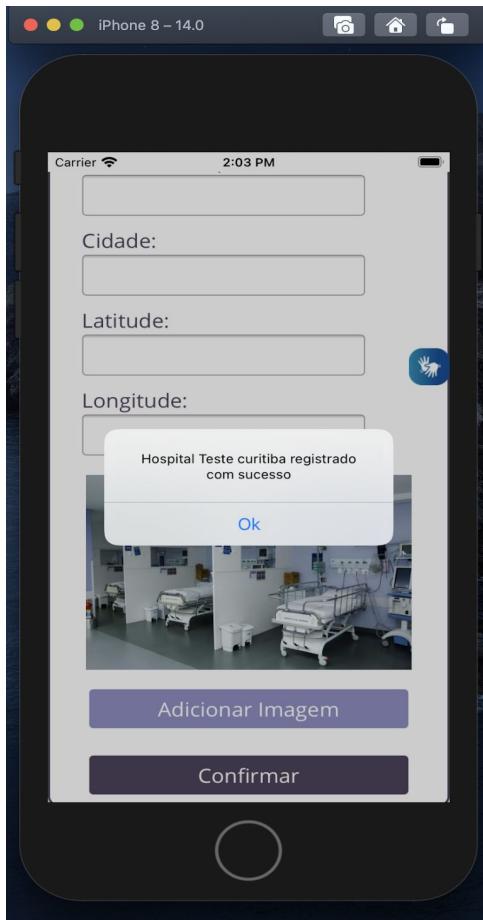
Ao selecionar esta opção você verá um mapa com todos os hospitais cadastrados. Abaixo do mapa existe a opção ‘Cadastrar novo hospital’.



Ao clicar nela você será levado a outra tela onde deverá inserir os dados do novo hospital a ser cadastrado



Ao clicar no botão 'Adicionar imagem' você poderá inserir uma imagem para representar. Após preencher todos os campos disponibilizados, basta clicar em 'Confirmar' que o novo hospital será cadastrado no sistema



4.3.6 Listar hospital

Como exibido na seção anterior (Cadastro de hospital) ao clicar na opção geral 'Mapas' é possível ver todos os hospitais cadastrados no sistema, de acordo com a geolocalização dos mesmos.



Na tela anterior é possível ver o hospital teste que foi cadastrado na seção ‘Cadastro de hospital’. Ao clicar no hospital é possível ver os detalhes inseridos na etapa de cadastro (nome e imagem).



4.3.7 Cadastro de equipamento

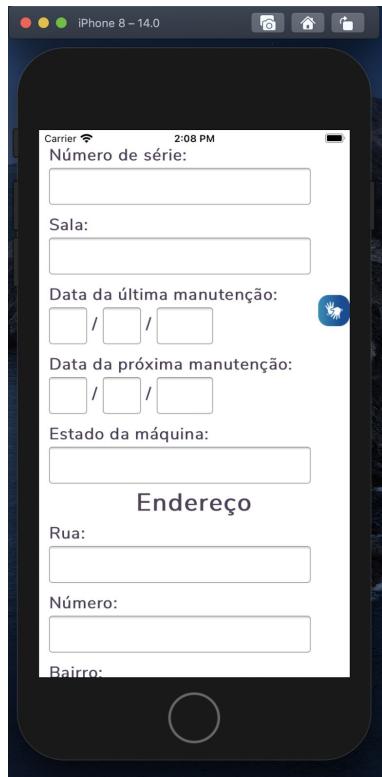
Para cadastrar um equipamento no sistema, é necessário primeiro acessar a opção geral ‘Equipamentos’



Ao clicar nela, você verá que existe o botão 'Cadastrar Equipamento'



Ao clicar no botão você será redirecionado para a tela onde deverão ser inseridos os dados do equipamento.



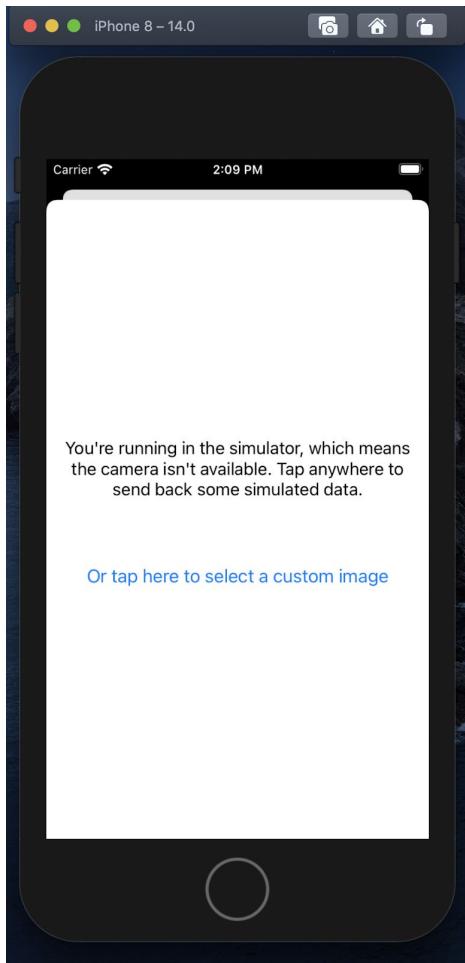
Você pode inserir os dados manualmente ou se preferir utilizar os botões que ajudam a preencher a geolocalização e o número serial da máquina (a partir de um QR code). Os botões estão localizados na parte inferior da tela



Ao clicar no botão ‘Adicionar Imagem’ é possível adicionar uma imagem relacionada ao equipamento que está sendo adicionado. Clicando nela você deverá escolher uma imagem do seu dispositivo e após selecioná-la ela estará visível na tela dos dados do equipamento.



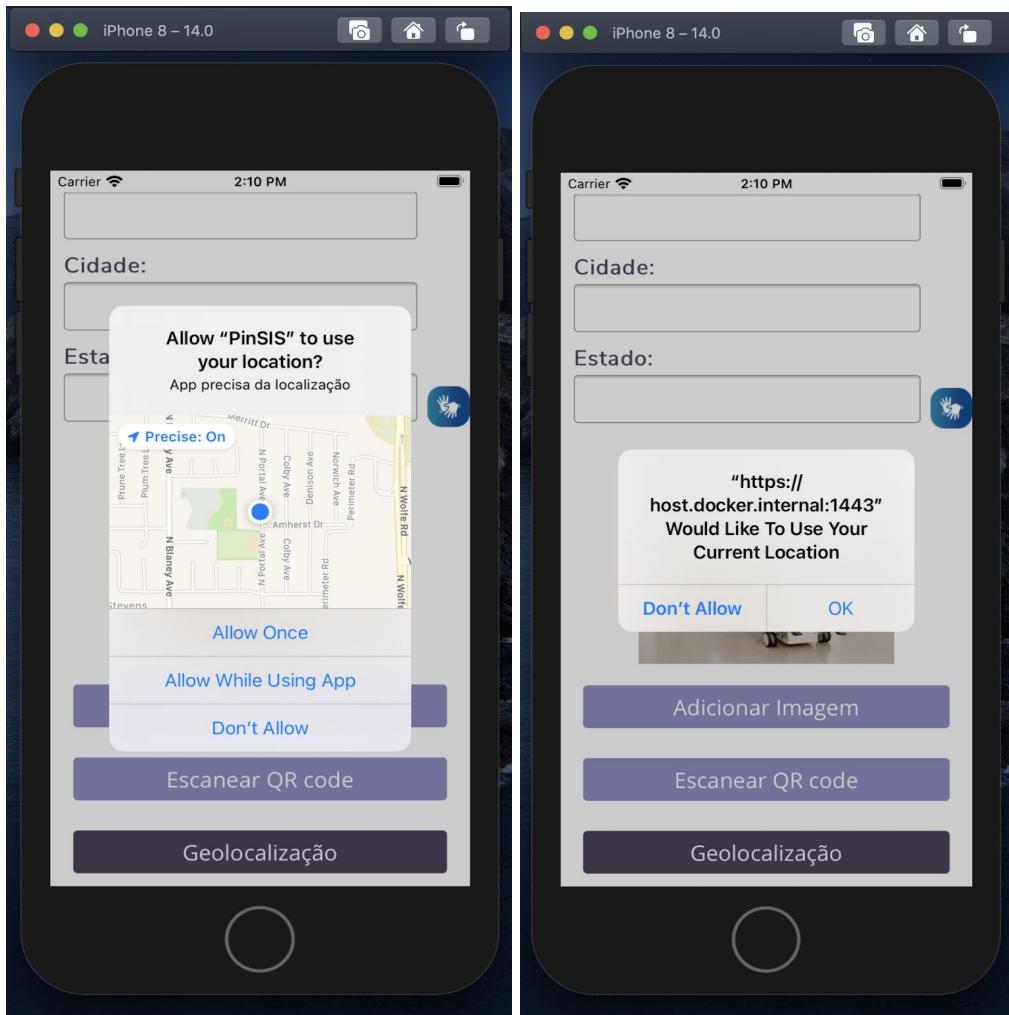
Ao clicar em ‘Escanear QR Code’ a câmera será aberta e você poderá escanear o QR Code com o número de série do equipamento. Como este tutorial foi feito utilizando um dispositivo emulado, não é possível abrir a câmera. Felizmente, a biblioteca utilizada para realizar o escaneamento permite a inserção de dados de teste, quando o dispositivo sendo utilizado é emulado.



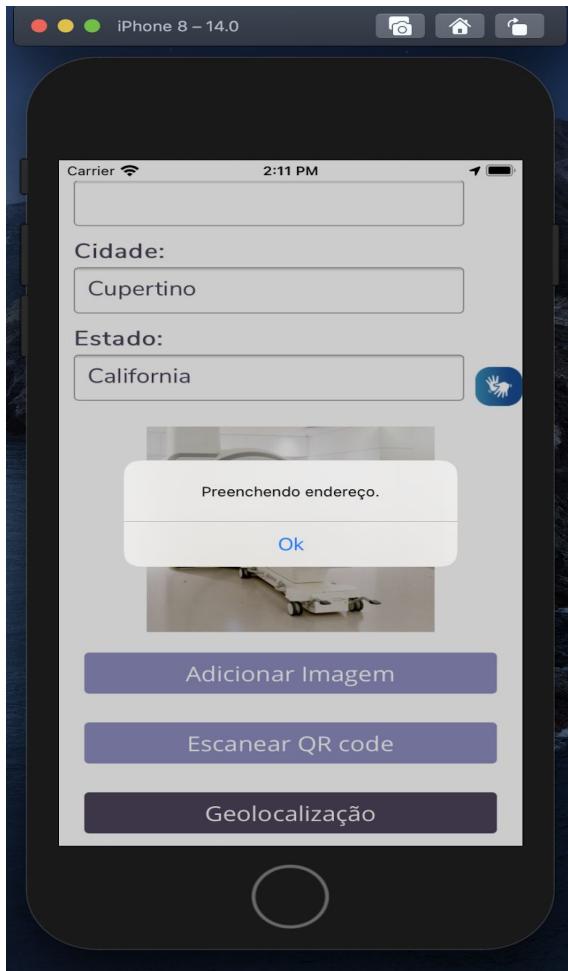
Ao final do escaneamento, a aplicação mostrará uma mensagem com o número serial escaneado



A última opção que pode ser utilizada para preencher automaticamente os dados é a geolocalização. Para obtê-la, basta clicar no botão 'Geolocalização'. Você precisará autorizar explicitamente a aplicação para que os dados possam ser coletados



Após autorizar, a aplicação mostrará uma mensagem indicando que os dados estão sendo preenchidos



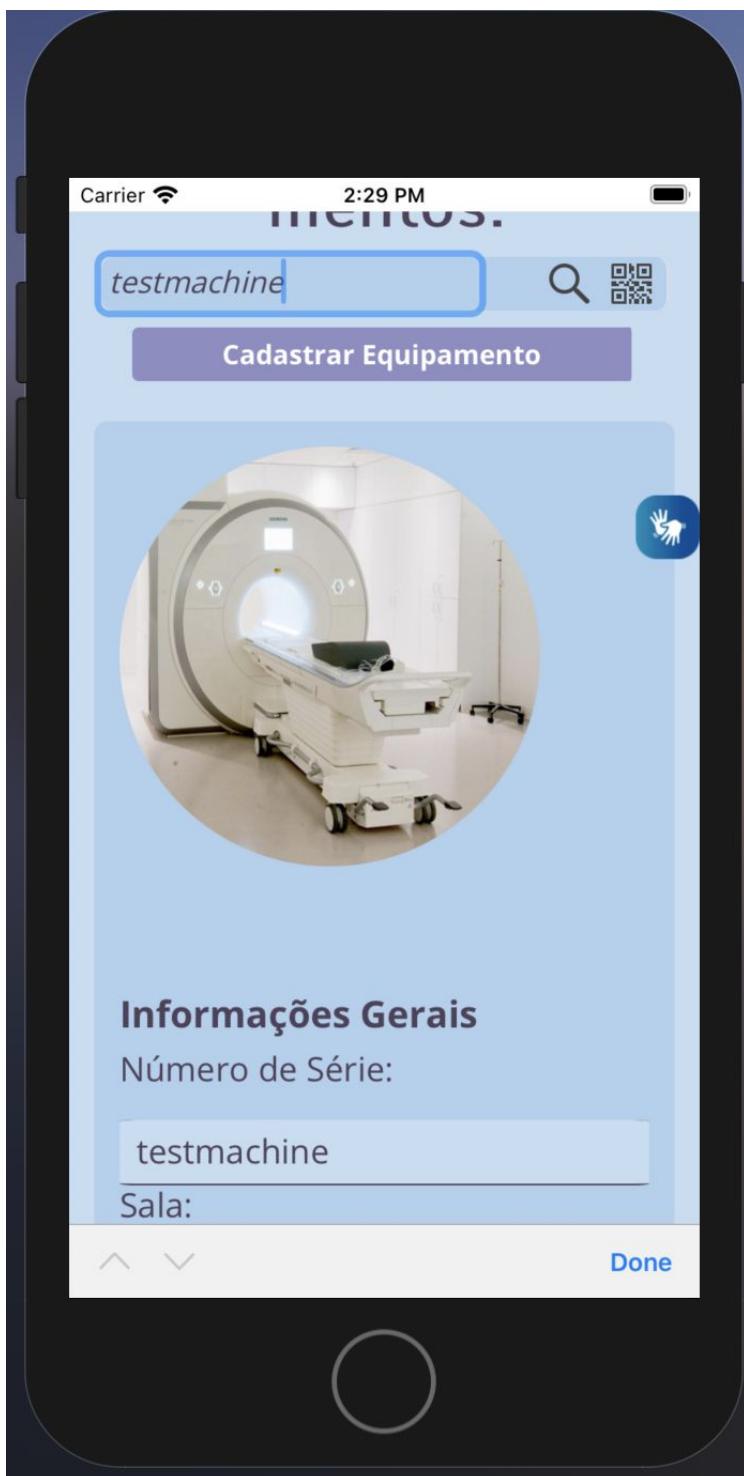
Ao final, verifique os dados faltantes para serem preenchidos, preencha eles e clique em 'Confirmar'. Ao término do cadastro, uma mensagem será exibida na tela indicando que o equipamento foi cadastrado



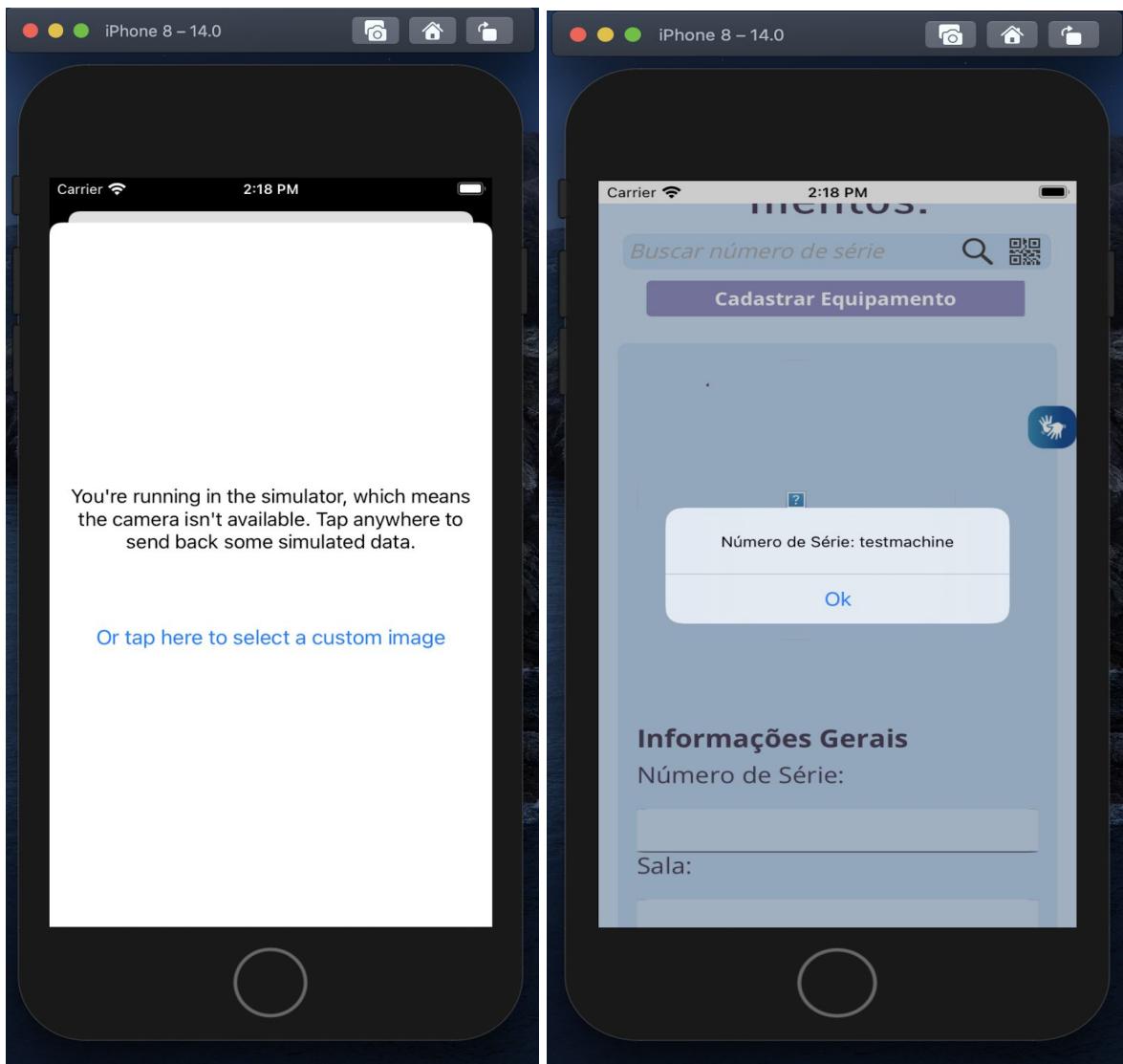
4.3.8 Listagem de equipamento

Para listar todos os equipamentos cadastrados, basta entrar na opção 'Equipamentos'. Ao entrar nesta tela é possível procurar um determinado equipamento. Para isto você pode procurar pelo número serial do equipamento ou utilizando o QR Code com o número serial do equipamento.

Para procurar pelo número serial basta colocar o valor na barra acima de ‘Cadastrar Equipamento’ e clicar no ícone da lupa a direita. Se o equipamento existir, ele será exibido na tela, como no exemplo abaixo.



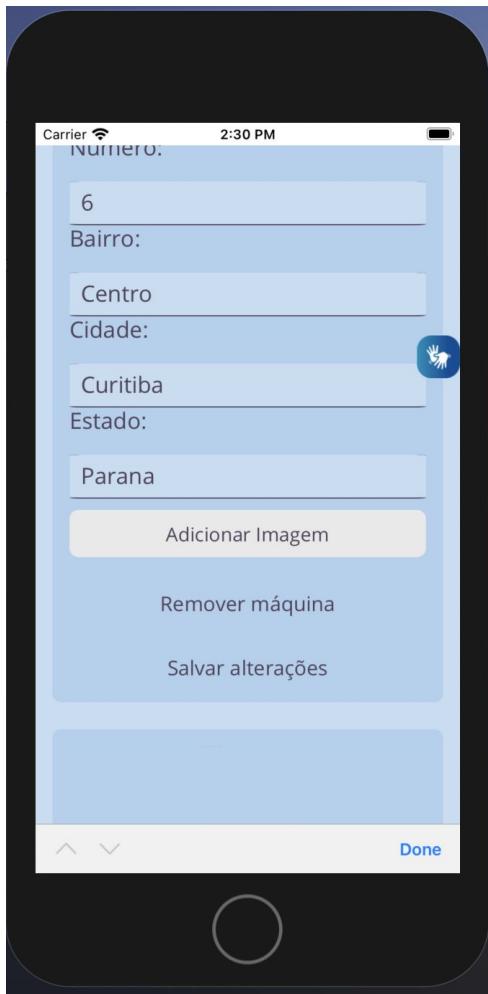
Caso deseje procurar utilizando o QR Code com o número serial, basta clicar no ícone do QR Code (a direita do ícone da lupa). O aplicativo irá abrir a câmera para poder escanear o código. Como dito anteriormente, devido os testes serem feitos utilizando um dispositivo emulado, a câmera não tem funcionalidade e portanto são utilizados dados de teste que só existem no dispositivo emulado. Ao fim uma mensagem é exibida com o dado que foi escaneado



Ao fim disto, serão exibidos os detalhes do equipamento cadastrado no sistema com o número serial que foi escaneado



Abaixo dos detalhes do equipamento existem algumas opções, caso seja necessário modificar alguma informação do equipamento ou removê-lo completamente do sistema.

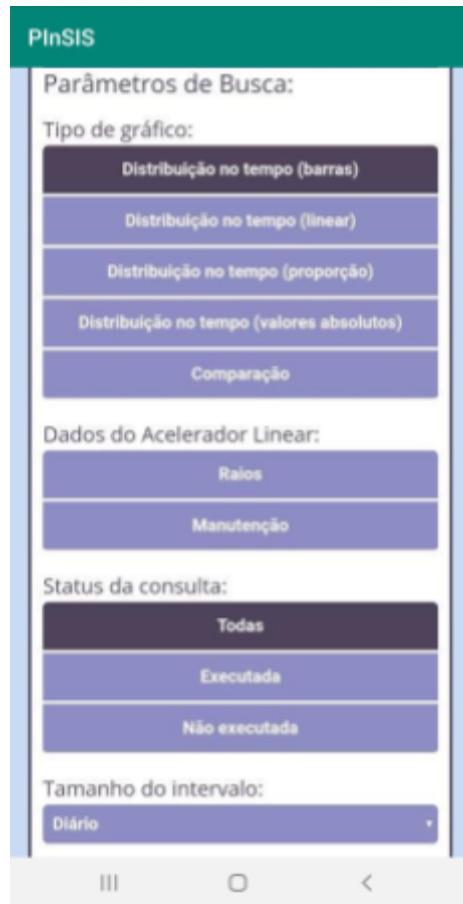


4.3.1 Gráficos de uso dos equipamentos

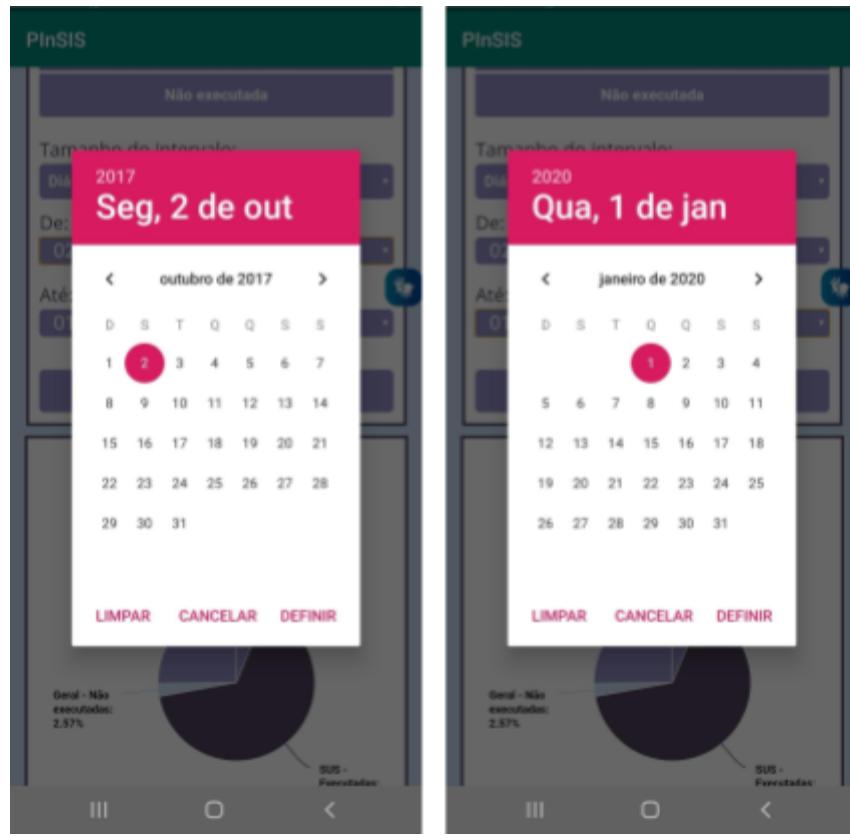
As informações coletadas estão apresentadas na página de gráficos, acessada clicando no botão "Gráficos" do menu principal.



Os parâmetros de busca estão separados em distribuição no tempo com os tipos de gráficos em barras, linear, proporção e com valores absolutos e de comparação. "Dados do Acelerador Linear" possui as opções raios, referentes a "Manutenção" que são dados referentes a manutenção dos aceleradores lineares. Em "Status da consulta" tem as opções de selecionar as consultas que foram executadas, não executadas e as duas (todas). Em "Tamanho do intervalo" é possível escolher entre as opções diário ou mensal.



E, por último, é necessário selecionar o intervalo de tempo referente às consultas através do calendário, como é mostrado na imagem abaixo:



Ao clicar em "buscar" os gráficos são apresentados de acordo com as opções selecionadas.

4.3.2 Métricas

É possível acessar a página de métricas clicando no botão “Métricas” no menu principal. As informações são referentes à simulação de performance do monitoramento em múltiplos hospitais, com o tempo e a velocidade de execução.



5 COLOCANDO O SITE DISPONÍVEL AO PÚBLICO USANDO O NGINX

5.1 PRÉ REQUISITOS

Será necessário uma máquina que esteja disponível 24h/dia, da qual você tenha acesso a privilégios de super user

Também será necessário ter um domínio(ou seja, uma url) para ser usada para hospedar o site

5.2 INSTALAÇÃO

1. Instalando o nginx

Antes de tudo será necessário instalar o nginx na sua máquina, para isso basta abrir o terminal e digitar as seguintes linhas de comando(o \$ apenas indica que são linhas de comando, não devendo ser digitado):

sudo apt update

sudo apt install nginx

Será necessário inserir as credenciais de super user.

2. Configurando firewall

Antes de tentar rodar o nginx, será necessário configurar o firewall para que ele de acesso ao serviço. Para verificar quais aplicativos o firewall já está configurado para lidar, digite:

sudo ufw app list

Você deverá ver listados 3 aplicativos do nginx, o Nginx HTTP para tráfico de rede não encriptado na porta 80, o Nginx HTTPS para tráfego de rede encriptado na porta 443 e por fim o Nginx Full que lida com ambos.

A princípio abriremos somente a HTTP já q não configuramos o SSL ainda.

sudo ufw allow 'Nginx HTTP'

Você pode verificar se funcionou utilizando a linha:

sudo ufw status

3. Testando o nginx

Após a instalação do nginx, o nginx já deve estar rodando. Para verificar se está tudo certo, digite:

systemctl status nginx

```
● nginx.service - A high performance web server and a reverse proxy server
  Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2020-10-22 09:28:00 -03; 4min 52s ago
    Docs: man:nginx(8)
   Main PID: 1094 (nginx)
      Tasks: 3 (limit: 614)
     Memory: 6.4M
        CPU: 0.000 CPU(s) since start
       CGroup: /system.slice/nginx.service
               ├─1094 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
               ├─1095 nginx: worker process
               └─1096 nginx: worker process
```

Caso deseje testar, basta digitar http://endereço_do_servidor na barra do navegador de sua escolha que esteja rodando na mesma máquina do nginx. Em caso de sucesso você verá uma tela dessa forma:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Caso não saiba seu endereço de ip, basta digitar

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/V.*$//'
```

Esse comando retornará várias linhas, tente com cada uma pra ver se funcionam

4. Ligando o nginx com o frontend

Nesse ponto, o nginx já deve estar funcionando localmente. Agora vamos fazer com que a página leve ao site do Pinsis. Siga os passos de instalação do Frontend na máquina que servirá de servidor ou apenas pegue o diretório compilado (explicação mais a frente)

Caso tenha instalado o projeto com sucesso, vá até o diretório onde o instalou e acesse o pinsis-portal/pinsisApp via linha de comando, lá será necessário alterar a variável q define a URL, para isso digite:

```
gedit src/environments/environment.prod.ts
```

Ali, altere o campo webServiceUrl de “<https://pinsis.c3sl.ufpr.br>” para a URL que será utilizada, lembre-se de colocar apenas http caso ainda não tenha tornado a conexão segura e https caso tenha.

Após isso, digite:

```
sudo npm install -g @angular/cli
```

ng build --prod

Esse comando compilará o projeto e gerará um diretório chamado “public/” contendo o site.

Caso não queira instalar o projeto nessa máquina, você pode apenas pegar esse diretório public compilado em outro lugar.

Antes de mais nada, tenha certeza de dar permissão de execução e leitura nesse diretório aos outros usuários, você pode fazer isso com a seguinte linha de comando:

chmod -R 755 public/

Possuindo o site compilado e configurado, falta ligá-lo ao nginx. Para isso precisaremos criar um arquivo de configuração do site

Para abrir o arquivo para edição usaremos a seguinte linha de comando:

sudo gedit /etc/nginx/sites-available/pinsis

No caso optei pelo gedit como editor de texto, mas você pode mudar para algum de sua preferência, como vim, atom ou nano, bastando apenas substituir nessa linha

Você deverá fazer com que o arquivo seja similar a esse, substituindo os campos /caminho/para/o/diretorio/public pelo caminho do diretório na sua máquina. Isso pode ser feito navegando para o diretório public e digitando:

pwd

Também deve ser alterado o seu_dominio para o domínio do site que você possuir, algo como pinsis.com.

```
server {  
    listen 80;  
    listen [::]:80;  
  
    root /caminho/para/o/diretorio/public;  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name seu_dominio www.seu_dominio;  
  
    location / {  
        try_files $uri $uri/ /index.html =404;  
    }  
}
```

Agora, para habilitar o site, é preciso criar um link simbólico para esse arquivo de configuração no diretório sites-enabled do nginx, para isso:

```
sudo ln -s /etc/nginx/sites-available/pinsis /etc/nginx/sites-enabled/
```

Agora, para testar se não há nenhum erro de sintaxe no arquivo:

```
sudo nginx -t
```

Caso haja algum erro, o corrija, se não basta reiniciar o nginx para fazer o site funcionar

```
sudo systemctl restart nginx
```

Feito isso, acesse o domínio que você colocou no arquivo de configuração e você deverá ver o site no ar.

5. Colocando o backend no ar

Inicie o serviço criado na seção do Webservice com:

```
sudo systemctl start pinsis
```

Para fazer com que requisições em http://seu_dominio/api/ sejam redirecionadas para o backend, basta adicionar as seguintes linhas no arquivo de configuração do nginx:

```
location /api/ {  
    proxy_pass http://localhost:3000;  
}
```

6. Habilitando HTTPS no seu servidor web

Esse passo é opcional, porém é extremamente recomendado visto que deixará o site significativamente mais seguro. Primeiro nós geraremos um CSR (Certificate Signing Request). Para isso, criaremos um diretório para guardá-lo e colocaremos permissões apenas para o super user:

```
sudo mkdir /etc/ssl/private  
sudo chmod 700 /etc/ssl/private
```

Agora, faremos uma requisição do CRS para o servidor do Nginx. Use a seguinte linha de comando para gerar o CRS, e preencha todas as informações pedidas:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

Então configuraremos o servidor para utilizar os certificados SSL. Para isso edite o script de configuração do SSL

```
sudo gedit /etc/nginx/conf.d/ssl.conf
```

Copie e cole o seguinte script

```
server {  
    listen 443 http2 ssl;  
    listen [::]:443 http2 ssl;  
  
    UbuntuPIT http://127.0.0.1/;  
  
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;  
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;  
    ssl_dhparam /etc/ssl/certs/dhparam.pem;  
}  
  
root /usr/share/nginx/html;  
  
location / {  
}  
  
error_page 404 /404.html;  
location = /404.html {  
}  
  
error_page 500 502 503 504 /50x.html;  
location = /50x.html {  
}  
}
```

Agora, redirecionando seu servidor Nginx de HTTP para HTTPS. Abra o script de configuração de redirecionamento SSI:

```
sudo gedit /etc/nginx/default.d/ssl-redirect.conf
```

E adicione a seguinte linha:

```
return 301 https://\$host\$request\_uri/;
```

Após isso, lembre-se de retornar no “pinsis-portal/pinsisApp/src/environments/environment.prod.ts”, trocar o http por https e recompilar o portal (ng build --prod)

Então basta reiniciar o nginx e testar:

```
sudo systemctl restart nginx
```

A URL agora deve iniciar com https se tudo tiver ocorrido bem.

6 CNES E METABASE

6.1 DEPENDENCIAS

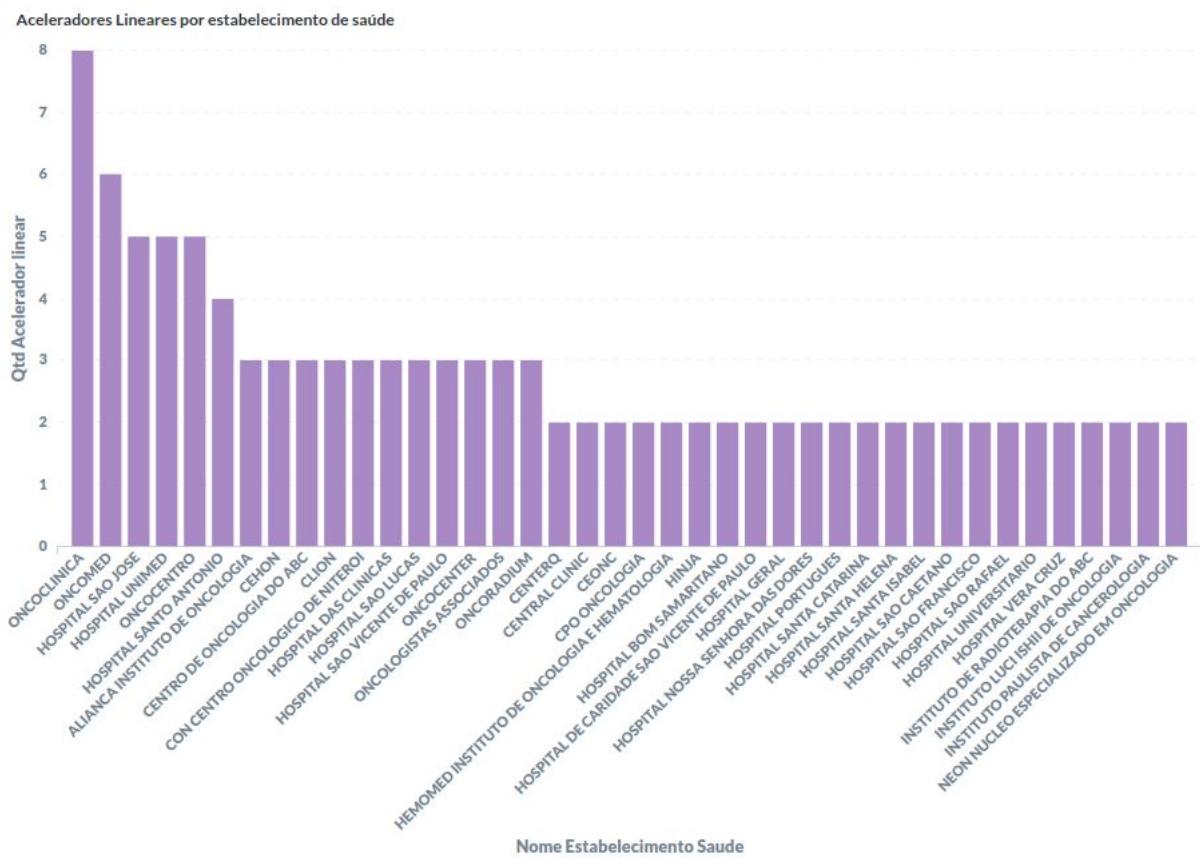
Essa etapa do projeto não apresenta nenhuma dependência, apenas é necessário que o pinsis-portal esteja instalado e rodando em sua máquina.

6.2 INSTALAÇÃO/COMPILAÇÃO

Ver a seção 3.2 do manual de instalação e compilação do frontend.

6.3 USABILIDADE

O CNES é uma plataforma que visa reunir todas as informações sobre o sistema de saúde brasileiro. Com a base de dados do CNES é possível encontrar informações sobre os estabelecimentos de saúde, equipamentos médicos, profissionais vinculados a um estabelecimento médico, etc. Com base nisso, é possível visualizar na página pinsis.c3sl.ufpr.br/graficos-cnes os seguintes gráficos:



Quantidade de leitos disponíveis convênio e sus em todas as instituições

Nome Estabelecimento Saude	Qt Exist	Qt Sus
INSTITUTO DO CANCER DO ESTADO DE SAO PAULO	219	219
INSTITUTO DO CANCER DO ESTADO DE SAO PAULO	150	150
A C CAMARGO CANCER CENTER	112	66
HOSPITAL DE CANCER DE PERNAMBUCO	108	108
A C CAMARGO CANCER CENTER	108	60
HOSPITAL AMARAL CARVALHO JAU	107	87
HOSPITAL DE CANCER	103	82
HOSPITAL HAROLDO JUACABA	94	54
HOSPITAL DO CANCER DE MURIAE	93	58
HOSPITAL DR LUIZ ANTONIO	89	89
FUNDACAO PIO XII BARRETOS	88	88
IRMANDADE DA SANTA CASA DE MISERICORDIA DE PORTO ALEGRE	88	45
MS INCA HOSPITAL DO CANCER I	88	88

Linhas 1-13 de 1116 ▲ ▶

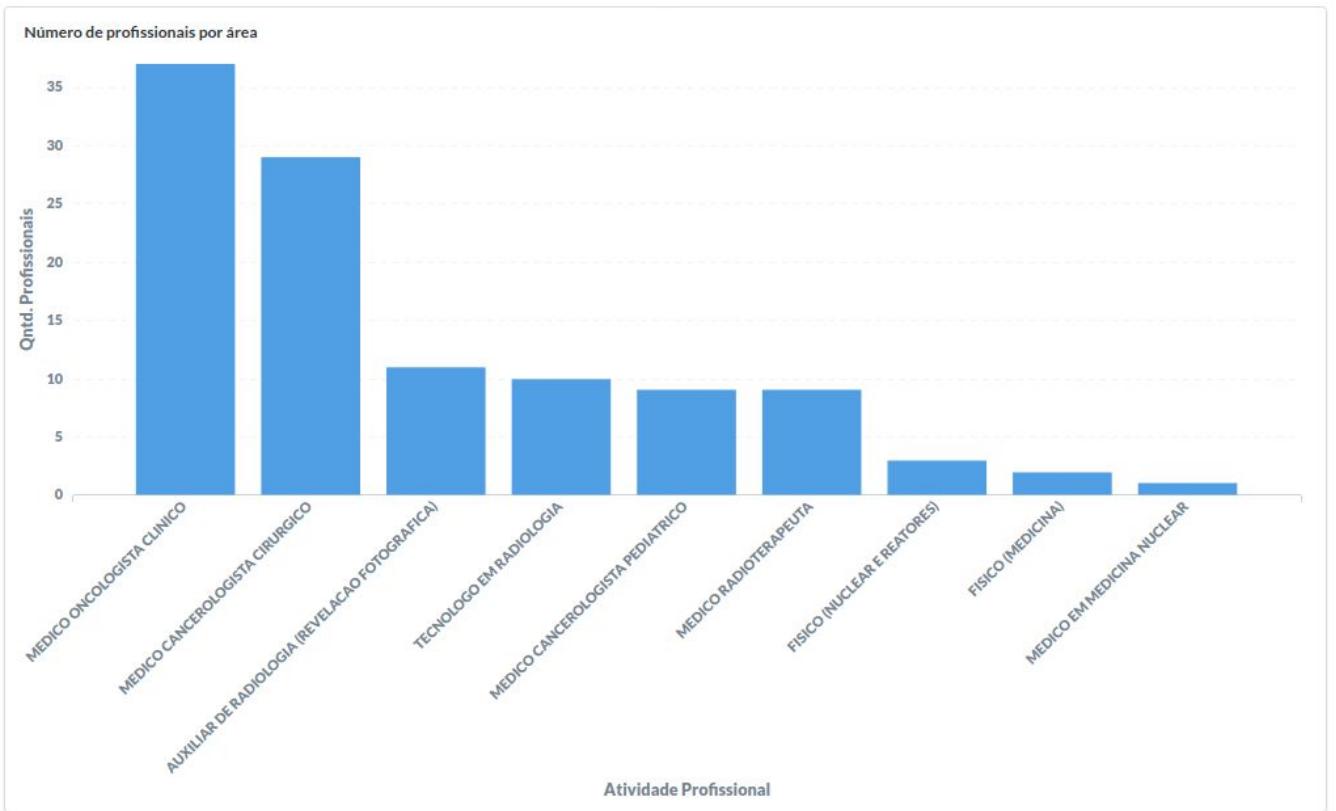
Informações Hospital Erasto Gaertner

111

Quantidade de leitos totais

79

Quantidade de leitos disponíveis para o SUS



7 AGENTE DE COLETA

Este README tem como objetivo apresentar os requisitos necessários para instalar/executar o componente pinsis-agent, que realiza a coleta dos dados médicos utilizados no sistema PinSIS.

7.1 DEPENDÊNCIAS

O componente pinsis-agent precisa ser executado em um ambiente Debian Linux, pois necessita de alguns comandos que estão presentes por padrão neste ambiente. Ele pode ser instalado em outros ambientes, porém não será discutido neste documento os passos necessários para tal.

A instalação do componente, para criação deste documento, foi feita em um ambiente **Ubuntu 16.04**, mas pode ser feita em outras distribuições Linux baseadas em Debian.

Além dos comandos presentes no Linux, o componente também necessita de alguns pacotes adicionais. Por padrão, a instalação do componente já faz a instalação destes pacotes, mas caso seja necessário instalar manualmente, os pacotes necessários são *samba*, *supervisor* e *default-jdk*.

O componente Java, responsável por fazer um *dump* de um determinado banco de dados, também tem algumas dependências, que são o pacote JSON in java (*org.json*), *ojdbc7* e *sqljdbc4*. Não é necessário instalar estas dependências pois elas já estão incluídas no repositório da aplicação e serão utilizadas quando a aplicação for compilada.

Sumarizando a lista de dependências:

- samba
- cron
- supervisor
- default-jdk
- Distribuição Linux Debian

7.2 INSTALAÇÃO/EXECUÇÃO

Esta seção irá cobrir os procedimentos necessários para poder instalar o componente em uma máquina Debian.. Para prosseguir, é necessário ter uma máquina rodando ambiente Debian bem como as dependências explicitadas na seção ‘Dependências’.

A instalação do sistema Debian está fora do escopo desta documentação.

Para instalar o componente, é necessário primeiro ter o código do componente *pipsis-agent* localmente na máquina. Considerando que o código estará em um repositório Git, basta clonar o repositório na máquina. Ao final desta etapa, é necessário configurar o componente de acordo com a sua necessidade.

O componente pode sincronizar dados de duas formas:

- Obtendo os dados a partir de um banco de dados;
- Obtendo os dados compartilhados através de um servidor Samba.

Para configurar o componente, basta modificar o arquivo em *Make.d/make.conf* e adicionar os detalhes necessários para o ambiente onde o componente irá executar.

Caso você utilize a sincronização com *dump* de banco de dados, não se esqueça de adicionar a configuração com os dados do banco que será utilizado. Para isto, basta copiar o arquivo *Databasedump/cred.properties.example* para a pasta *Databasedump/properties* antes de iniciar a instalação. Modifique o arquivo com as configurações necessárias para acessar o banco de dados e obter os dados que devem ser copiados para o PinSIS.

Após esta configuração, esteja certo de estar no diretório que contém os arquivos do *pipsis-agent*. Nos testes feitos para a produção desta documentação, o código estava localizado em *~/test/pipsis-agent-master*.

Para preparar os componentes do *pipsis-agent* execute o comando **make** como usuário root. O comando irá configurar todos os scripts que são executados pelo componente, bem como compilar o aplicativo Java, responsável por fazer a cópia dos dados do banco de dados.

```

test@test-VirtualBox:~/test/pinsis-agent-master
Setting up gnome-shell (3.34.3-1ubuntu1-19.10.1) ...
test@test-VirtualBox:~/test/pinsis-agent-master$ sudo make build
make[1]: Entering directory '/home/test/test/pinsis-agent-master/Databasedump'
sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gtr1.2-geodeglib-1.0 gtr1.2-gst-plugins-base-1.0 gtr1.2-gudev-1.0 gtr1.2-udisks-2.0 grillo-plugins-0.3-base gstreamer1.0-gtk3 guile-2.2-libs libboost-date-time1.67.0
  libboost-filesystem1.67.0 libboost-iostreams1.67.0 libboost-locale1.67.0 libcurl4-openssl-dev libcurl4-openssl4 libcurl4
  libdazzle-1.0-0 libe-book-0.1-1 libebook-1.0 libepubgen-0.1-1 libetonyek-0.1-1 libevent-2.1-6 libfreerdp-client2-2 libfreerdp2-2 libgcrypt2 libgsm-1.0-0 libgpgmeppg libgpgmepp
  libgpod4 libgrilo-0.3-0 liblangtag-common liblangtag1 liblrc-client0 libluis-3.0 libmnl-tpn-pc17 libmspub-0.1-1 libodfgen-0.1-1 liborcus-0.14-0 libqbqwingzv5 libraw19
  librevenge-0.6-0 librsync libsgtutils2-2 libsparsesparseconfig9 libvncclient libwinpr2-2 libximseci-nss lp-solve media-player-info python3-bcrypt python3-fasteners
  python3-future python3-lz4 python3-lockfile python3-mako python3-markupsafe python3-monotonic python3-paramiko syslinux syslinux-common syslinux-legacy usb-creator-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ca-certificates-java default-jdk-headless default-jre-headless fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev
  libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmp-dev libxt-dev openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless
  x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-demo openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java default-jdk-headless default-jre-headless fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev
  libpthread-stubs0-dev libsm-dev libx11-dev libxcb1-dev libxdmp-dev libxt-dev openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless
  x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 25 newly installed, 0 to remove and 0 not upgraded.
Need to get 260 MB of archives.
After this operation, 415 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 java-common all 0.72 [6.816 B]
Get:2 http://br.archive.ubuntu.com/ubuntu eoan-updates/main amd64 openjdk-11-jre-headless amd64 11.0.7+10-2ubuntu2-19.10 [37,4 MB]
Get:3 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 default-jre-headless amd64 2:1.11-72 [3.192 B]
Get:4 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 ca-certificates-java all 20190405ubuntu1 [12,2 kB]
Get:5 http://br.archive.ubuntu.com/ubuntu eoan-updates/main amd64 openjdk-11-jre amd64 11.0.7+10-2ubuntu2-19.10 [34,8 kB]
Get:6 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 default-jre amd64 2:1.11-72 [1.084 B]
Get:7 http://br.archive.ubuntu.com/ubuntu eoan-updates/main amd64 openjdk-11-jdk-headless amd64 11.0.7+10-2ubuntu2-19.10 [217 kB]
Get:8 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 libatk-wrapper-jni amd64 2:1.11-72 [1.148 B]
Get:9 http://br.archive.ubuntu.com/ubuntu eoan-updates/main amd64 openjdk-11-jdk amd64 11.0.7+10-2ubuntu2-19.10 [2.237 kB]
Get:10 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 fonts-dejavu-extra all 2.37-1 [1.953 kB]
Get:11 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 libatk-wrapper-jni amd64 2:1.11-72 [1.096 kB]
Get:12 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 libatk-wrapper-java all 0.35.0-3 [53,1 kB]
Get:13 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 libatk-wrapper-java-jni amd64 0.35.0-3 [45,2 kB]
Get:14 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 xorg-sgml-doctools all 1:1.11-1 [12,9 kB]
Get:15 http://br.archive.ubuntu.com/ubuntu eoan/main amd64 x11proto-dev all 2018.4-4 [251 kB]

```

Ao final desta etapa você verá uma tela parecida com esta, que indica que os componentes foram configurados corretamente.

```

javac DatabaseDump.java
cp DatabaseDump.class /tmp/test/Databasedump
java -Djava.security.egd=file:///dev/urandom -Duser.timezone=GMT-3 -Duser.language=pt -Duser.country=BR -cp "/tmp/test/Databasedump:/tmp/test/Databasedump.jar/*" DatabaseDump /tmp/test/Databasedump/properties/*
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
Trying to connect to the Database
Successfully connected to the Database
Closing connection to the Database
make[1]: Leaving directory '/home/test/test/pinsis-agent-master/Databasedump'
test@test-VirtualBox:~/test/pinsis-agent-master$ 

```

Após esta etapa, um dump inicial dos dados do banco será feito (caso a opção para obter dados de um banco de dados esteja configurada). Você pode encontrar este dump na pasta *Today*, criada no diretório definido como *BASE_PATH* no arquivo *Make.d/make.conf*. Nos testes feitos para a criação desta documentação, a pasta utilizada foi */tmp/test*. Portanto, no diretório */tmp/test/Today*, após a execução do comando *make*, podemos encontrar o dump inicial do banco.

```

test@test-VirtualBox:~/test/pinsis-agent-master$ ls /tmp/test/Today/
test_20201023_145517.json
test@test-VirtualBox:~/test/pinsis-agent-master$ 

```

Após a configuração dos componentes, execute o comando ***make install*** como usuário root, que irá instalar todos os recursos necessários do *pensis-agent* bem como configurá-lo para rodar periodicamente todos os dias a meia noite.

Ao final da instalação você verá uma tela parecida com a seguinte:

```
[+] cp -r DatabaseDump /tmp/test
[+] mkdir -p /tmp/test/Today /tmp/test/storage
mkdir: cannot create directory '/tmp/test/Today': File exists
+ chmod a+r /tmp/test/storage
+ cp data_sync.sh base_data_sync
+ '[' TRUE == TRUE ']'
+ sed -i s/SAMBA/g data_sync
+ '[' TRUE == TRUE ']'
+ sed -i s/JAVA/g data_sync
+ sed -i s/USER/root/g data_sync
+ sed -i s@QBPATH@/tmp/test@g data_sync
+ chmod +x data_sync
+ cp data_sync /tmp/test
+ crontab -l
+ echo '0 0 * * * /tmp/test/data_sync'
./Makefile.samba config.sh
+ mkdir /tmp/test/samba
+ apt -y install samba
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading state information... Done
samba is already the newest version (2:4.10.7+dfsg-0ubuntu2.6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
+ echo
[ptninst]
path = /tmp/test/samba
read only = no
guest ok = yes

+ service smbd restart
./Makefile.reverse_tunnel config.sh
+ apt -y install supervisor
Reading package lists... Done
Building dependency tree
Reading state information... Done
supervisor is already the newest version (3.3.5-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
+ cp ./sshd_config /etc/supervisor/conf.d
+ cp ssh-tunnel /usr/local/bin/
chown root:root /tmp/test
test@test-VirtualBox:~/test/pinsts-agent-master$
```

Feito isto, o agente estará instalado e configurado. Todo dia a meia noite ele irá executar. Para confirmar que ele está configurado corretamente, basta verificar que está instalado no *crontab*.

```
test@test-VirtualBox:~/test/pinsis-agent-master$ sudo crontab -l
* * * * * /tmp/test/data_sync
test@test-VirtualBox:~/test/pinsis-agent-master$
```