INSTITUTO FEDERAL DE SÃO PAULO - CAMPUS CUBATÃO

Carlos Miguel Costa Santiago da Conceição
Eduardo Freitas do Nascimento Filho
João Guilherme Buriola de Freitas
Leonardo Cintra Moreno Rodrigues
Lucas Morais Marques
Mariana Amarante e Almeida Batista
Vitor Vieira da Silva

TRABALHO DE CONCLUSÃO DE CURSO: IONIC

Cubatão / SP
2021

Índice

Histórico e principais características do Ionic

A programação, hoje, pode ser aplicada em diversos segmentos e com diferentes funcionalidades. Embora existam diferentes aplicações de softwares, alguns aspectos são similares e estruturais e, por isso, são implementados em aplicações diversas.
Aqui entram os frameworks, que tem como objetivo garantir que os recursos das aplicações sejam desenvolvidos com rapidez. Em resumo, frameworks são ferramentas que permitem o compartilhamento de trechos de código entre aplicações com funcionalidades semelhantes.
O frameworks que nosso grupo irá usar para desenvolver o trabalho de conclusão de curso é o Ionic, e falaremos mais sobre ele a seguir.

O Ionic foi desenvolvido por Max Lynch, Ben Sperry e Adam Bradley, e teve seu lançamento em 2013. No site deles está escrito que o Ionic auxilia equipes a desenvolver e distribuir belos aplicativos híbridos em várias plataformas, com foco na experiência do usuário ou na interação do mesmo com o aplicativo.

O Ionic é um framework open source usado para desenvolver aplicativos móveis multiplataforma, ou seja, aplicações mobile híbridas. Para que isso aconteça, são utilizadas tecnologias que encontramos normalmente na construção do front-end de soluções web: HTML, CSS e JavaScript.

A linguagem de programação básica do Ionic é o Typescript, o que aumenta mais o “poder” de desenvolvimento de aplicativos. Esta linguagem é um conjunto de JavaScript, portanto, todos os códigos de JavaScript podem ser usados tranquilamente no desenvolvimento de aplicações híbridas com o Ionic.

A estrutura do Ionic torna possível a criação de aplicativos com interface e experiência semelhante em plataformas iOS e Android. O typescript proporciona uma boa experiência de usuário com interatividade pois, com ele, é possível a execução de uma lógica complexa para processar os dados do usuário.

Como diferencial, o Ionic apresenta recursos que simplificam o desenvolvimento do aplicativo e o deixam com um aspecto mais profissional. Ele apresenta um design limpo, simples e funcional, com componentes padrões, tipografia, paradigmas interativos e diversos templates. Além disso, o Ionic ainda possui um cliente de linha de comando (CLI) para gerenciar todos os projetos criados. O CLI é uma ferramenta que possibilita a criação de aplicativos Ionic de forma mais rápida e disponibiliza comandos que facilitam o desenvolvimento usando o framework. Há também um servidor de desenvolvimento integrado, ferramentas de compilação e depuração, entre outras.

Para finalizar, o Ionic possui muitas vantagens, e algumas delas são:

- Estabilidade na criação de aplicações híbridas: embora use HTML, CSS e JavaScript, o Ionic entrega aplicativos estáveis e com desempenho semelhante ao de aplicativos nativos;

- Multiplataforma: é possível desenvolver um único código que pode ser usado em diferentes plataformas, otimizando o tempo de desenvolvimento de aplicações;
- Menor tempo de desenvolvimento: por ser multiplataforma, o tempo de desenvolvimento no Ionic é mais vantajoso do que em aplicações nativas, visto que os códigos criados serão reutilizados em qualquer sistema operacional móvel;
- Menor custo: como o tempo de desenvolvimento é menor, diminui-se também os custos da aplicação, fazendo com que seja mais rentável criar aplicativos multiplataforma;
- Prototipação: existe uma ferramenta de “arrasta e solta” chamada Ionic Creator que deixa mais fácil a criação de telas;
- Documentação: a documentação do Ionic é bem completa, além de possuir uma grande comunidade.

IONIC: Principais IDE's

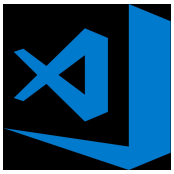
Sublime Text:

Lançado em 2008, o Sublime Text se destaca por sua facilidade de uso, aceitando múltiplas linguagens de programação. O Sublime é um editor de texto simples, porém apresenta bastantes recursos e funcionalidades a adicionar.

Android Studio:

O Android Studio teve seu lançamento em 2013. Ele é uma IDE para o sistema operacional Android e é baseado no IntelliJ IDEA. Ademais, oferece um completo editor com ferramentas extras para ajudar no desenvolvimento de apps, um emulador rápido com inúmeros recursos, entre outras funcionalidades.

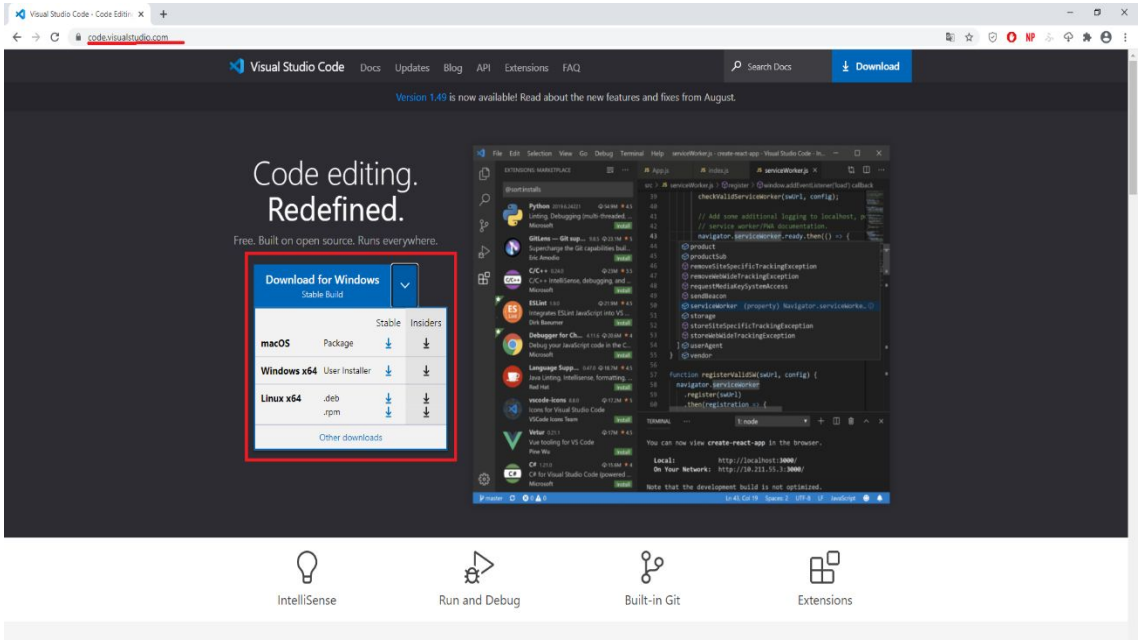
Visual Studio Code:



O Visual Studio Code é um editor de texto multiplataforma disponibilizado pela Microsoft, no ano de 2015, afim de desenvolver aplicações web com suporte a ASP.NET5 e Node.JS. E será com essa IDE que nós iremos trabalhar.

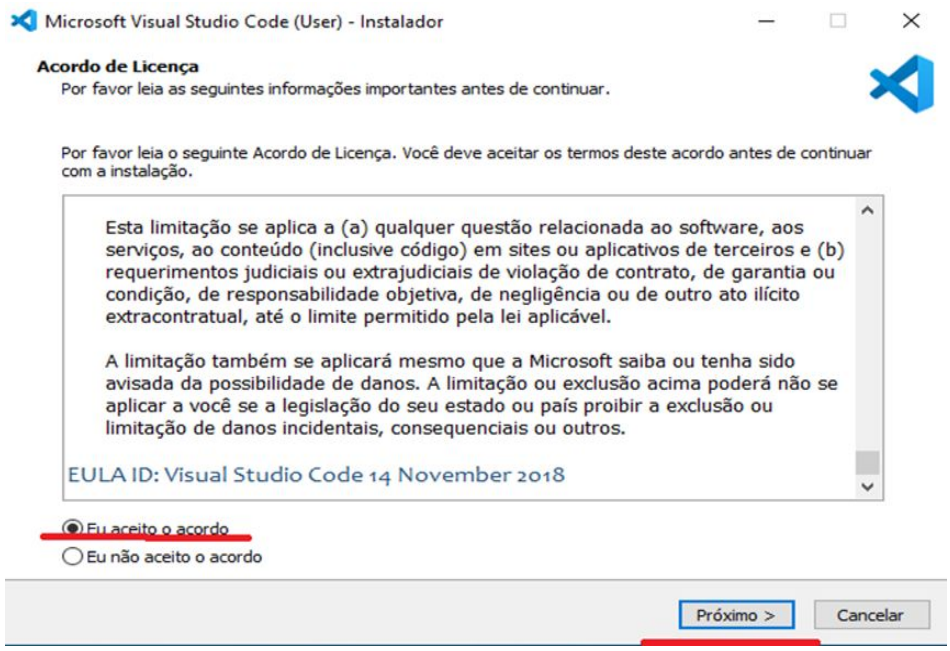
Instalando o Visual Studio

Para instalar o Visual Studio Code será preciso acessar o seu site e efetuar seu download. Depois de clicar em "Download for Windows", aparecerá as opções para baixar. Selecione a opção de acordo com o seu sistema operacional.

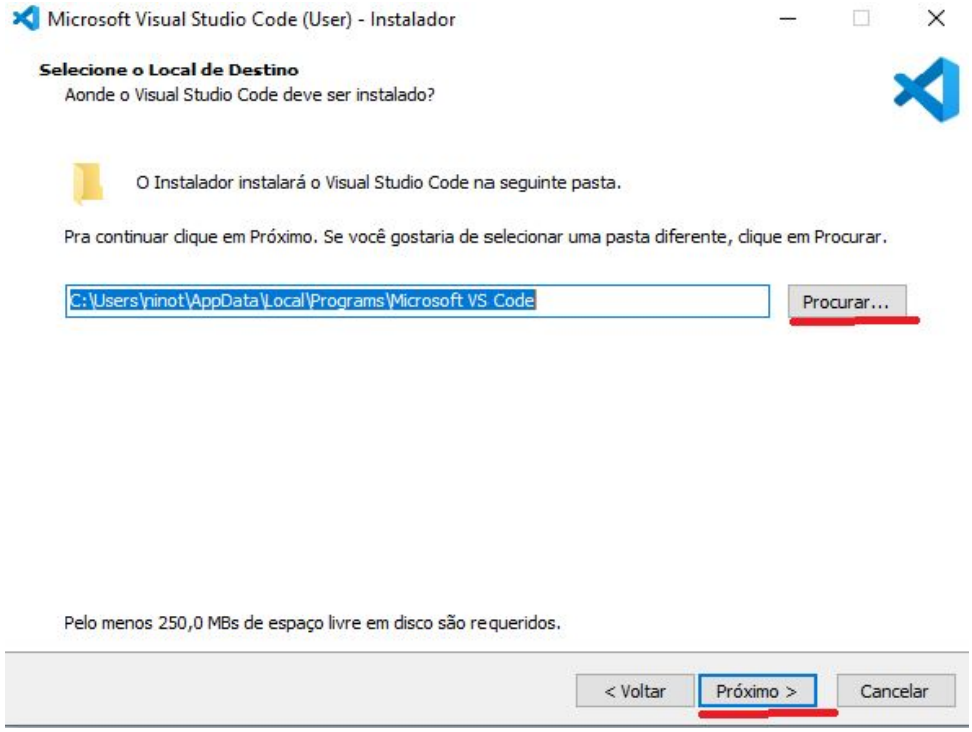


Após a conclusão do download, abra o arquivo executável.

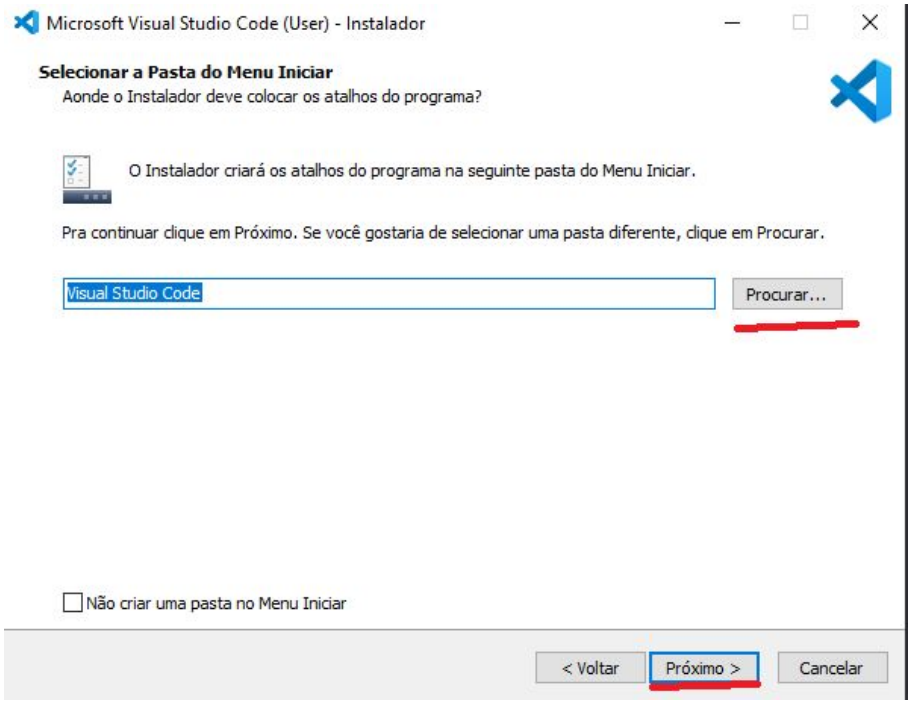
Em seguida, aparecerá a tela inicial de instalação do Visual Studio Code e, depois de ler os termos, aceite o acordo e prossiga com o processo.



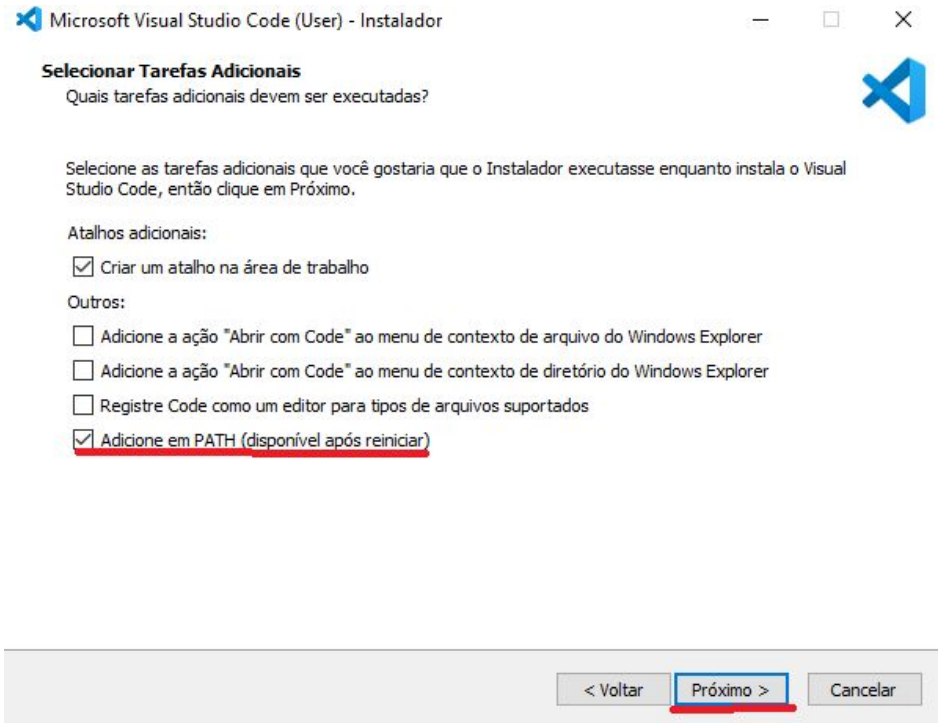
Escolha o diretório que deseja instalar Visual Studio Code e avance novamente.



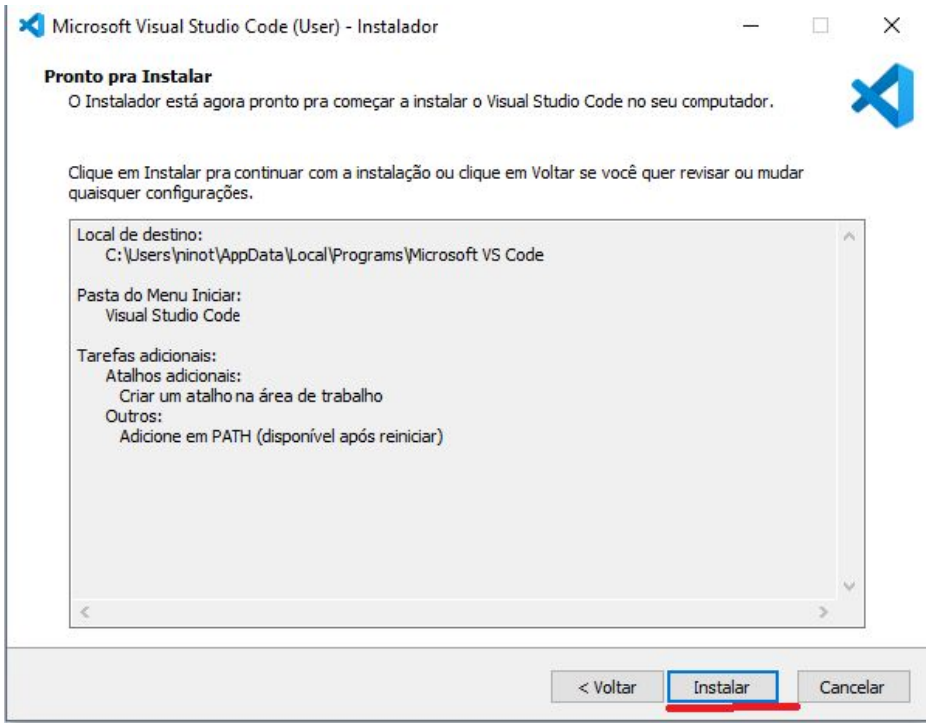
Determine a área em que deseja por o atalho do Visual Studio Code. Após isso, clique em “Próximo”.



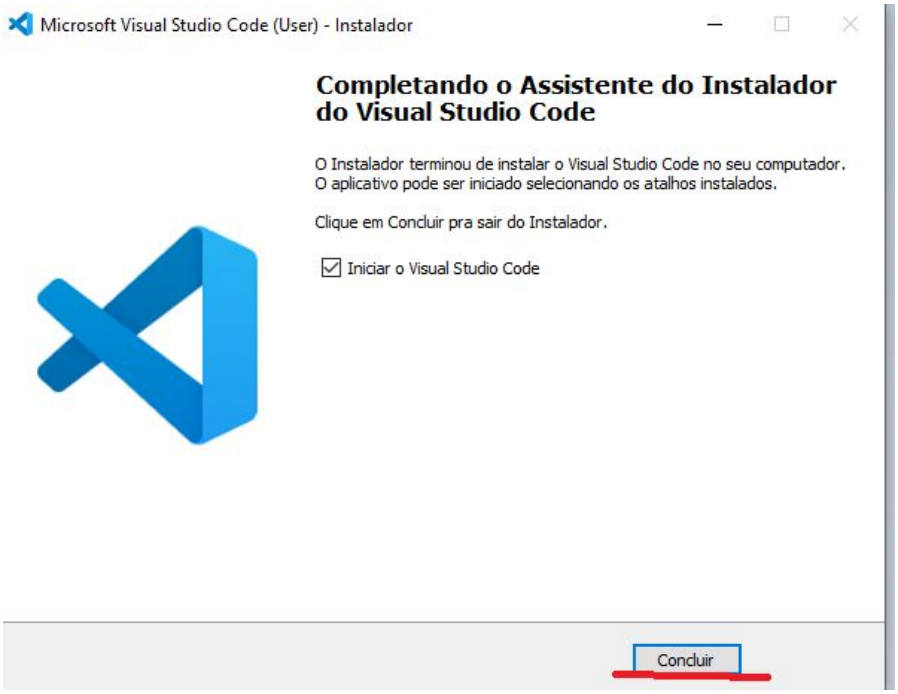
Na próxima tela, já estará marcada a opção “Adicione em Path”. A escolha das outras opções é opcional. Avance novamente.



Agora, clique em “Instalar” e aguarde a conclusão.

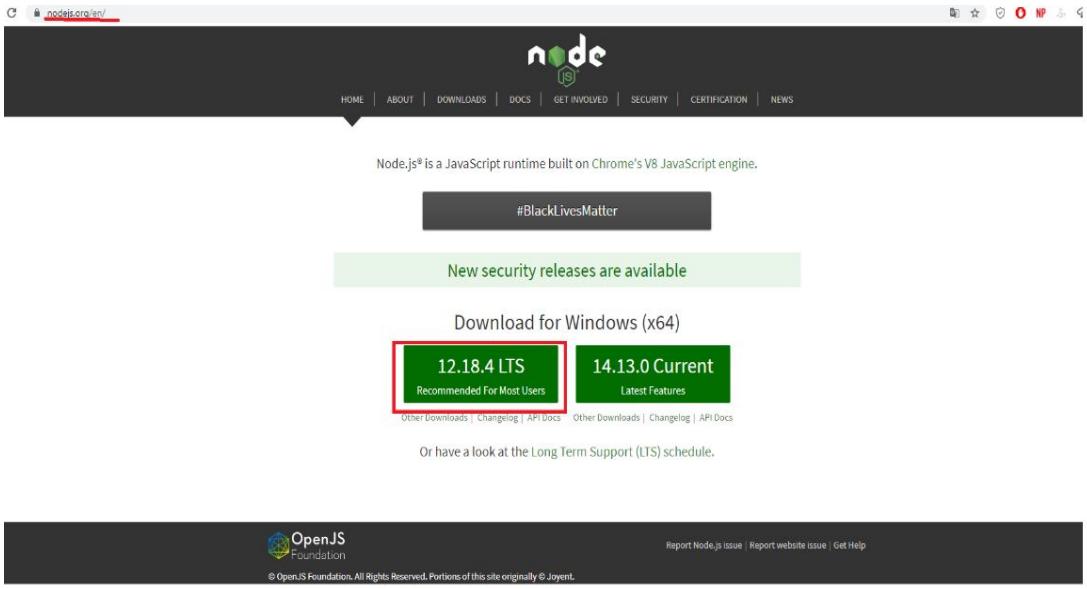


A instalação do Visual Studio Code foi realizada.



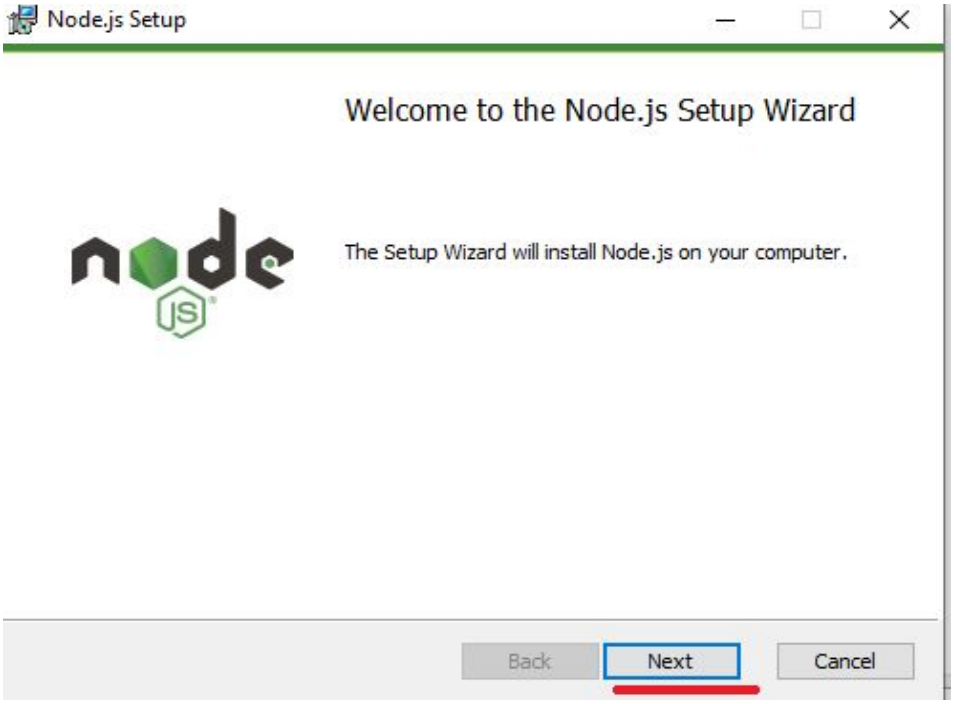
Instalando o IONIC

Antes de instalar o IONIC é preciso fazer o download do Node.JS. Então, acesse o site do Node e selecione a opção marcada em vermelho para efetuar o download.

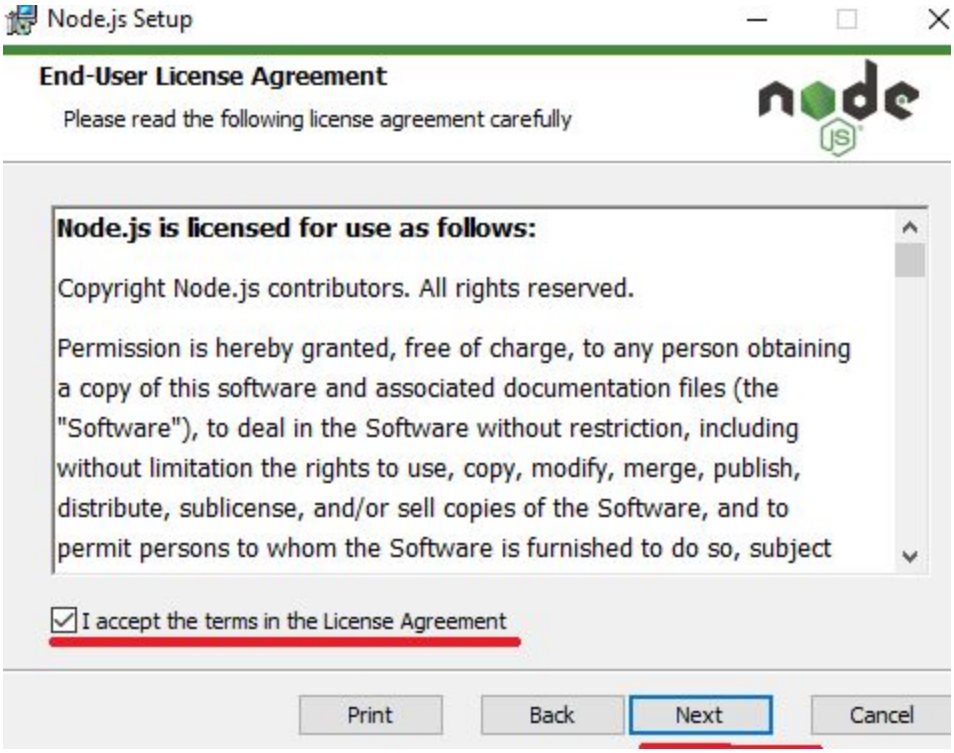


Após o término do download, abra o arquivo executável.

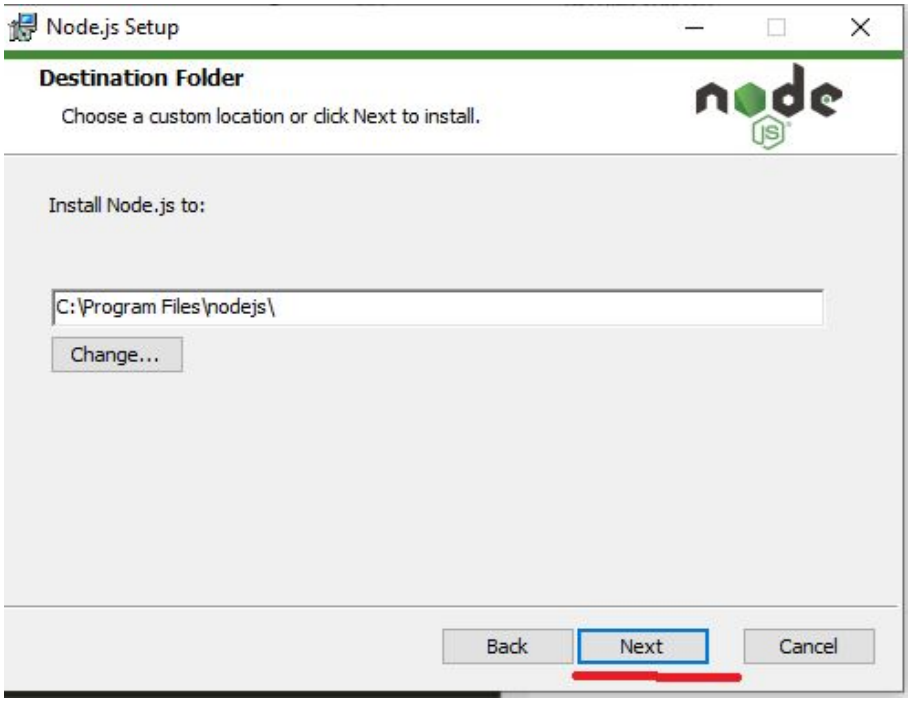
Selecione “Next” para começar a instalação do Node.JS.



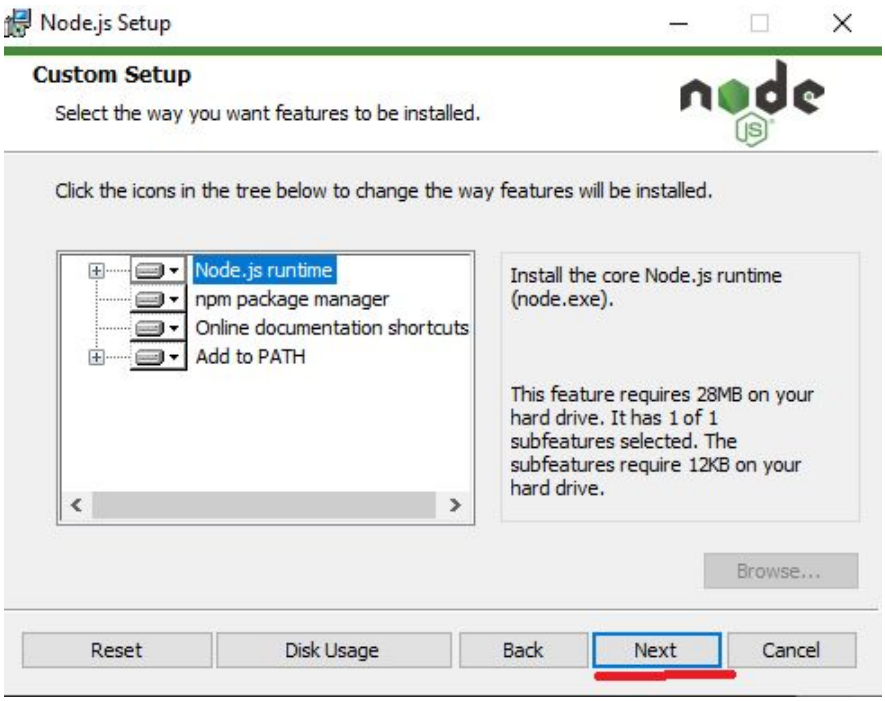
Leia os termos de licença, aceite e prossiga o processo.



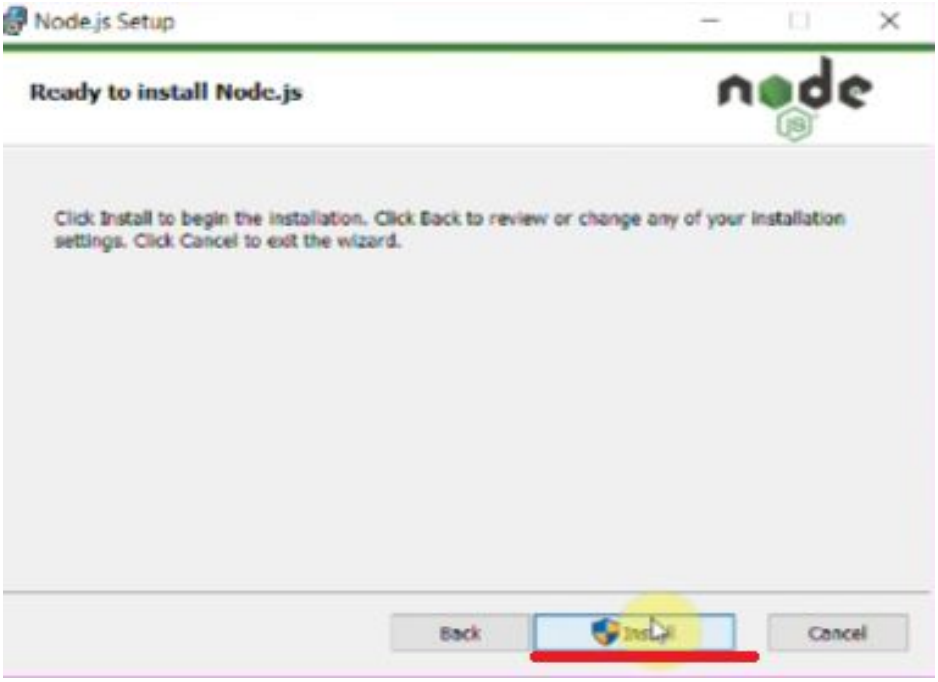
Agora, escolha o diretório onde o Node.JS será instalado e, após isso, prossiga.



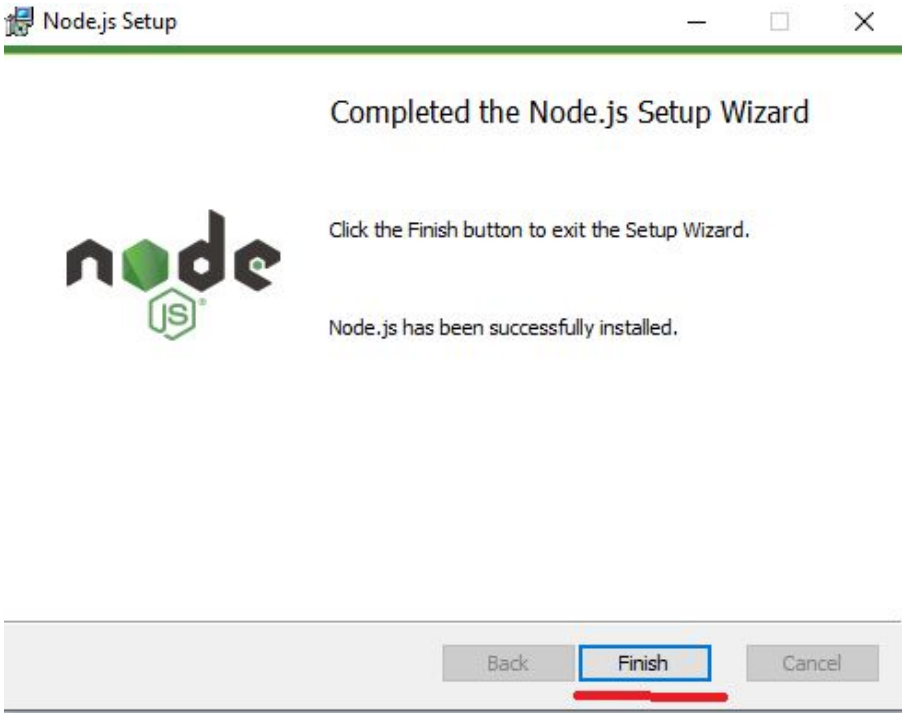
Após isso, aparecerá os componentes a serem instalados e, novamente, prossiga.



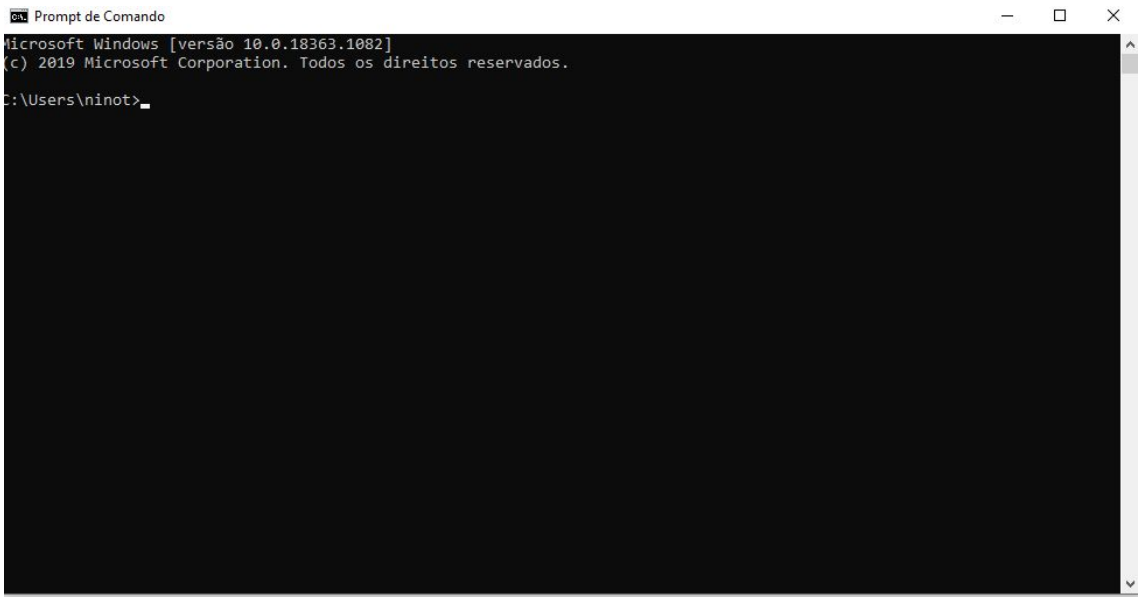
Clique em “Install” para instalar o Node.JS.



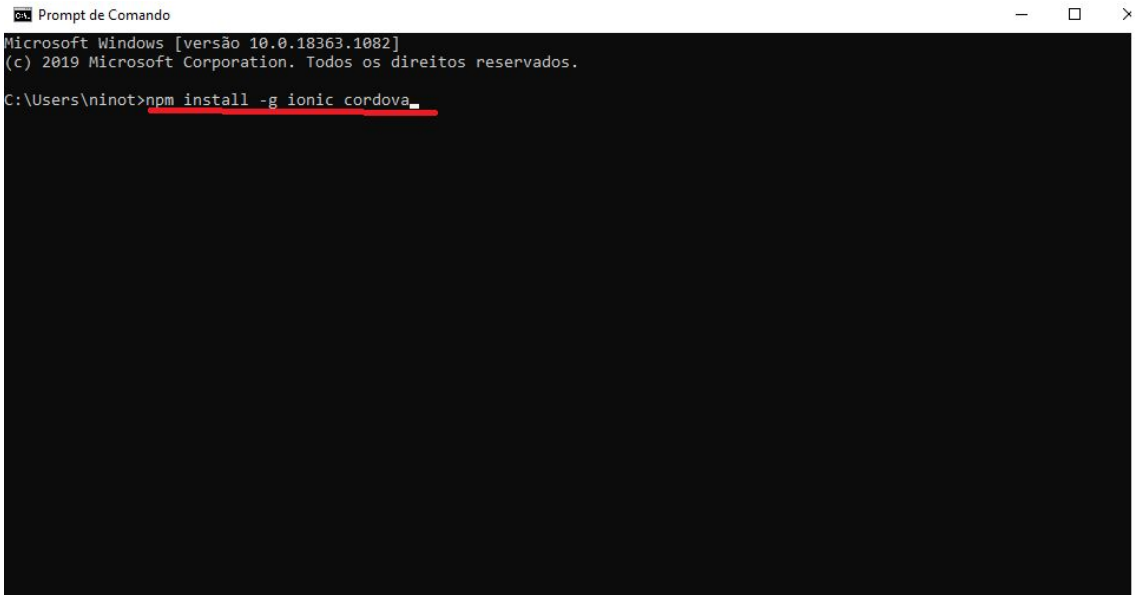
Após o término da instalação, clique em “Finish”.



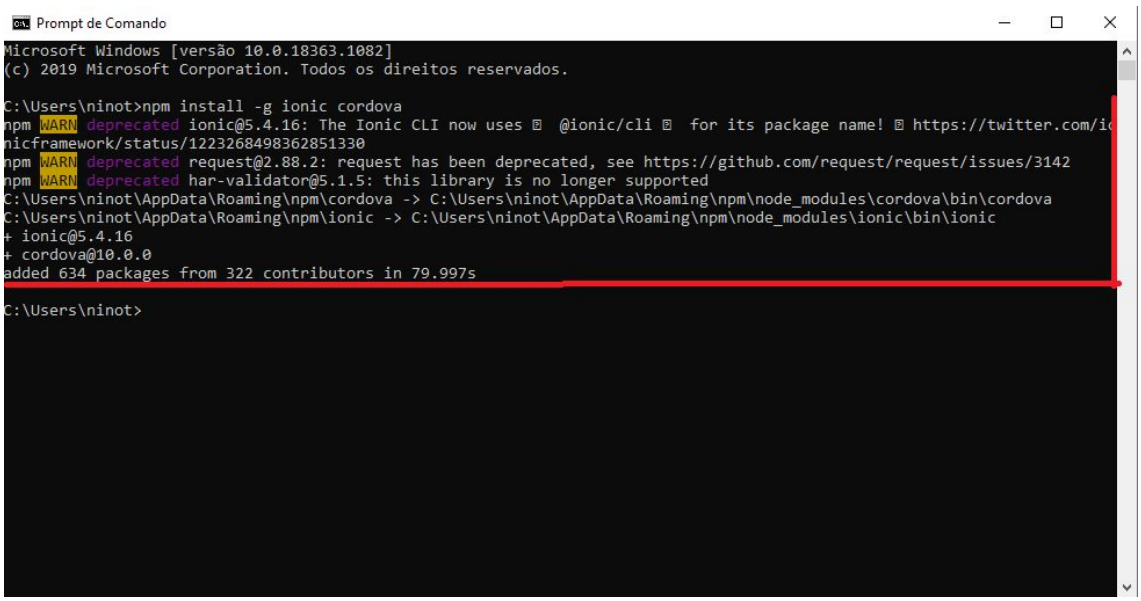
Agora, pressione as teclas “Windows + R” e no menu executar digite “cmd” para abrir o Prompt de Comando.



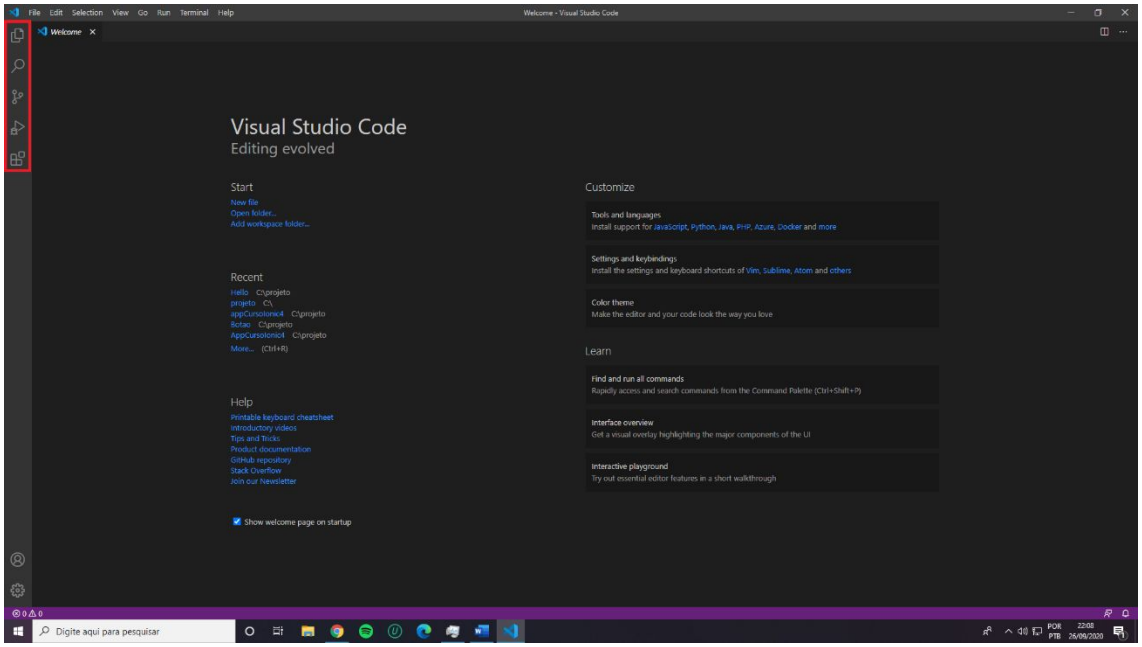
Digite “Install -g ionic cordova” e pressione a tecla “Enter” para instalar o Ionic em seu computador.



Instalação do Ionic concluída.


Prompt de Comando
Microsoft Windows [versão 10.0.18363.1082]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.
C:\Users\ninot>npm install -g ionic cordova
npm WARN deprecated ionic@5.4.16: The Ionic CLI now uses @ionic/cli for its package name! https://twitter.com/ionicframework/status/1223268498362851330
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\ninot\AppData\Roaming\npm\cordova -> C:\Users\ninot\AppData\Roaming\npm\node_modules\cordova\bin\cordova
C:\Users\ninot\AppData\Roaming\npm\ionic -> C:\Users\ninot\AppData\Roaming\npm\node_modules\ionic\bin\ionic
+ ionic@5.4.16
+ cordova@10.0.0
added 634 packages from 322 contributors in 79.997s
C:\Users\ninot>

Apresentação do Visual Studio Code



Ao executarmos o Visual Studio Code, seremos recebidos por uma tela, onde teremos algumas informações sobre o uso do programa. Situado na parte esquerda da tela, temos uma barra de ferramentas, onde respectivamente:

Explore: Exibe ou oculta as pastas e arquivos do aplicativo;

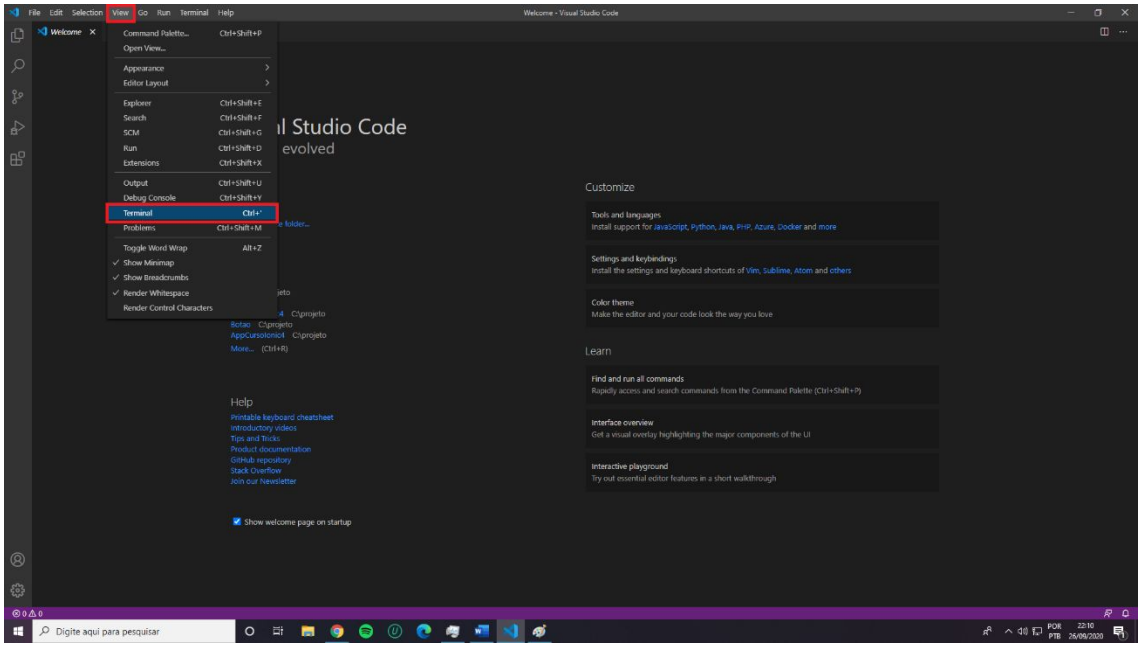
Search: Utilizado para realizar buscas de arquivos abertos;

Source Control: Ferramenta utilizada para facilitar a realização de commits para o repositório do projeto;

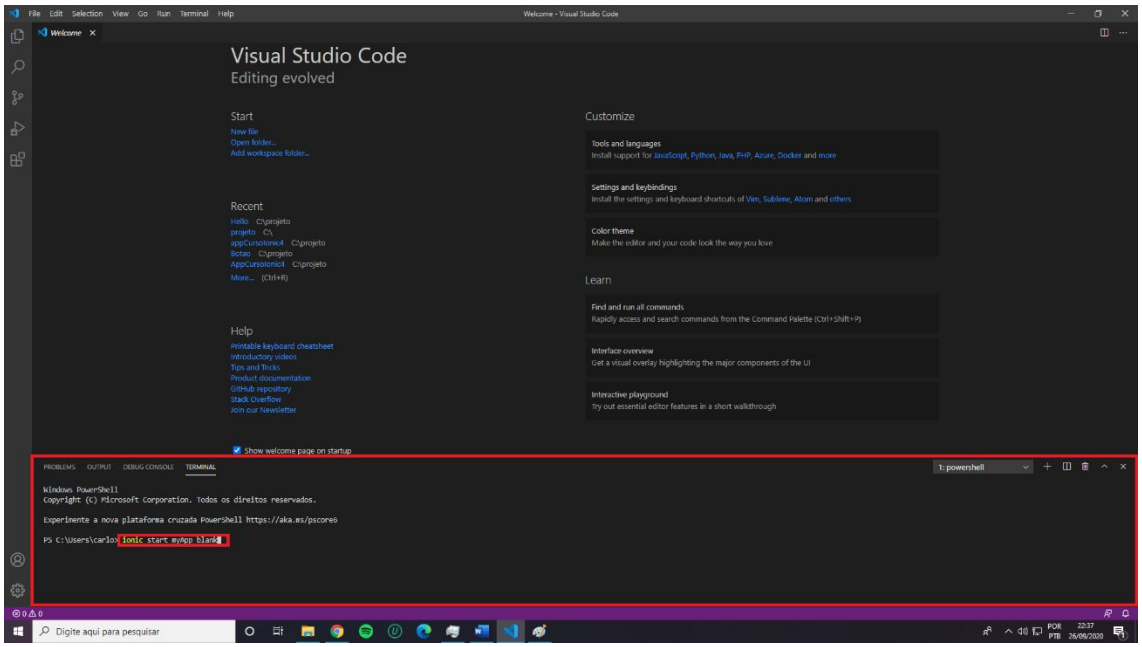
Debug: Ferramenta utilizada para analisar variáveis em tempo real de execução.

Extensions: Ferramenta utilizada para baixar extensões que futuramente podem auxiliar o programador no desenvolvimento do aplicativo.

Primeiro aplicativo

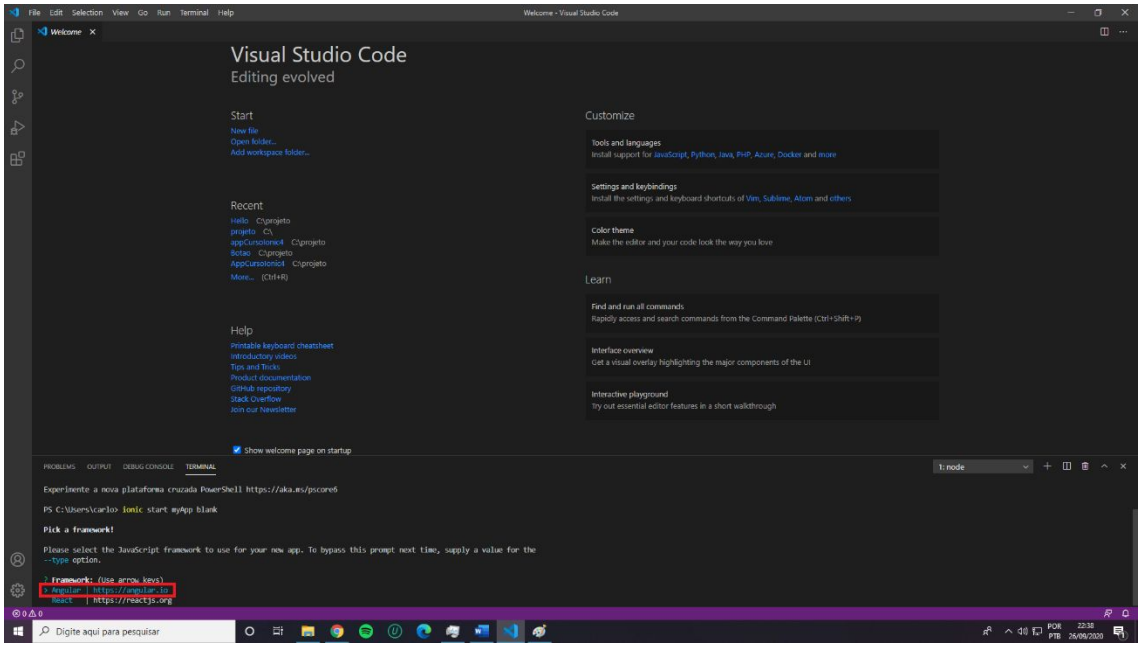


A maneira mais rápida de criar algum aplicativo em ionic é através do seguinte caminho: deve-se clicar no botão **View > Terminal**.

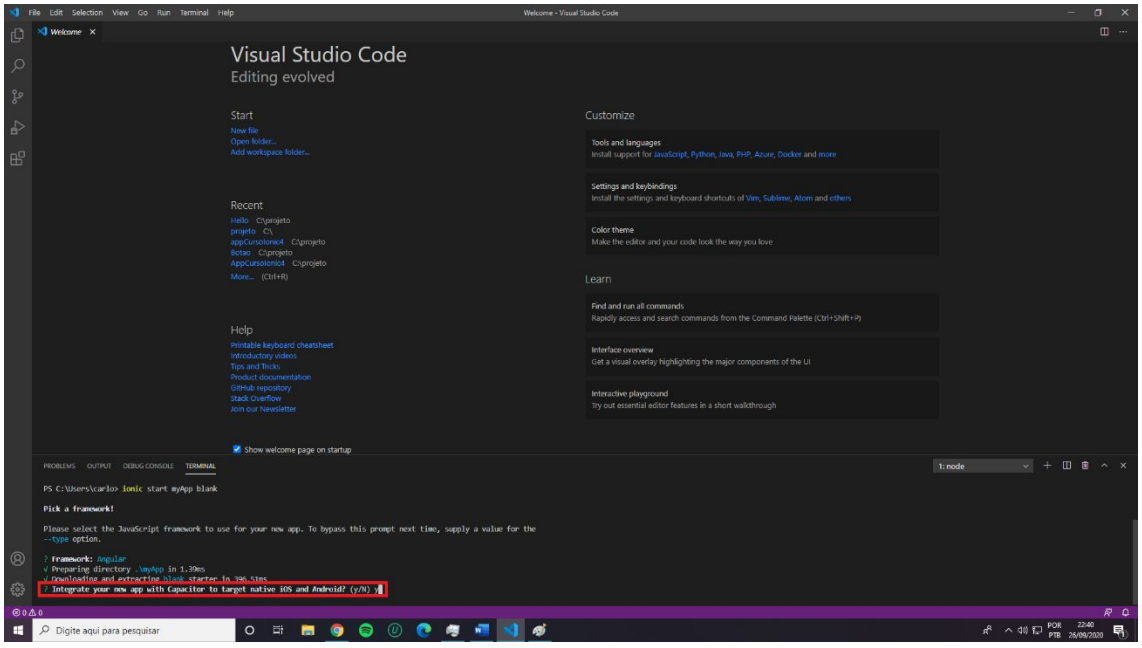


Realizando o procedimento anterior, é necessário apresentar no terminal o código de criação, um nome para o aplicativo e o tipo de design pré-fabricado (Blank, Tabs ou Side Menu), respectivamente. Nós utilizaremos o seguinte modelo:

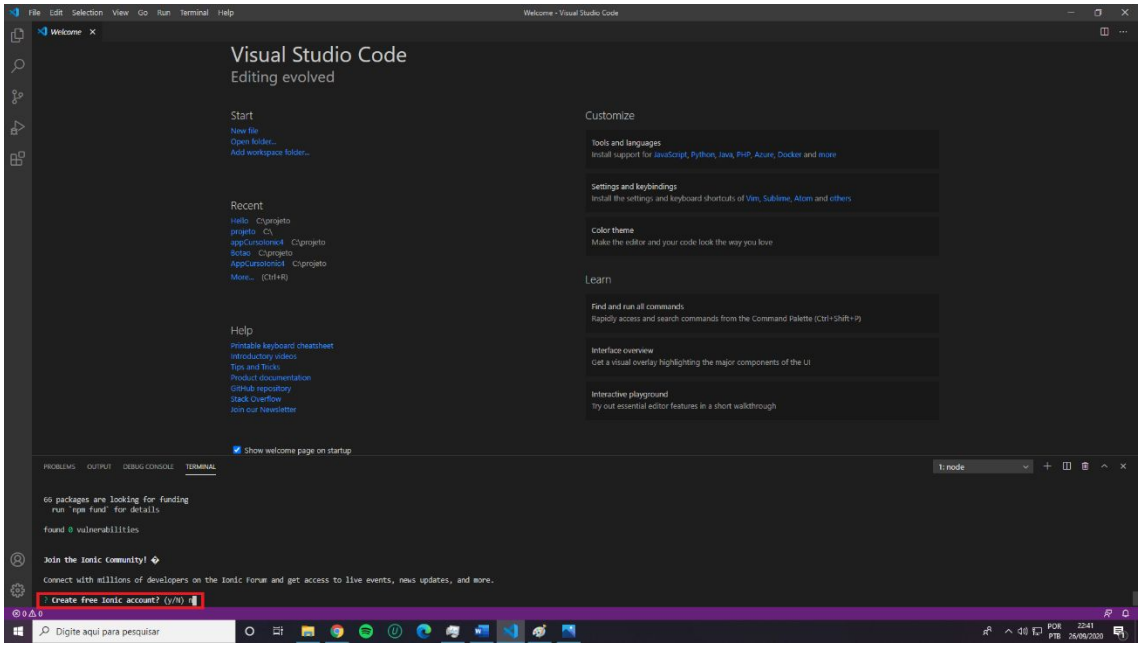
ionic start myApp blank



Em seguida, o terminal perguntará qual framework você deseja utilizar. Selecione “**Angular | https://angular.io**”.

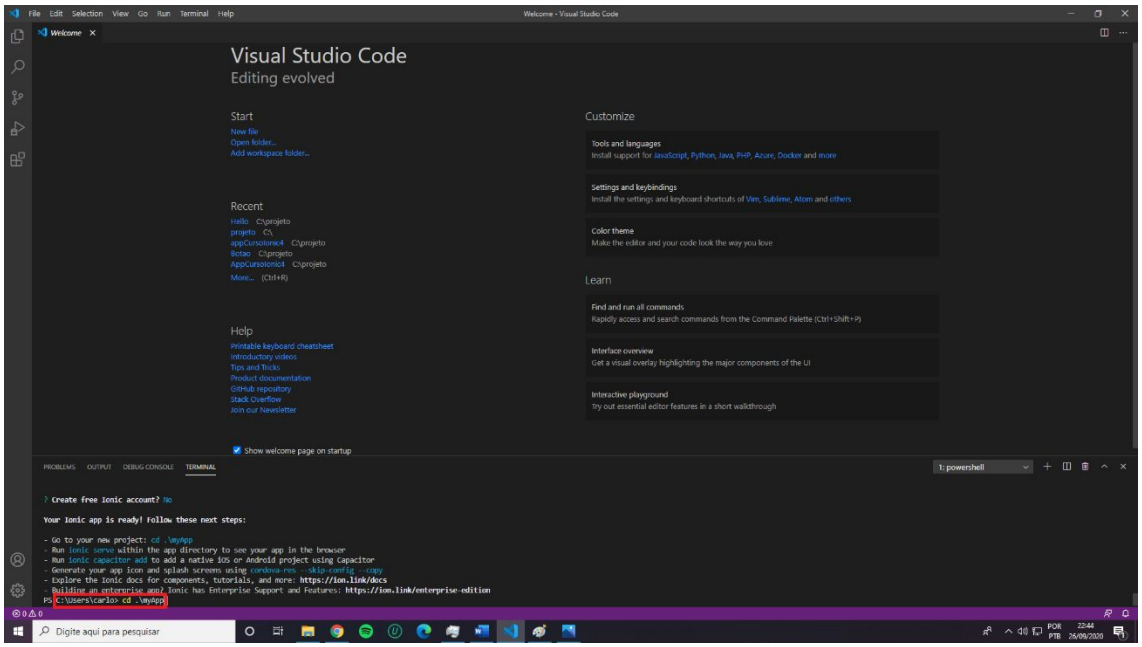


Após selecionar o framework, o terminal também perguntará se você possui interesse em integrar o seu aplicativo com funções específicas para Ios/Android. Responda com “yes”.



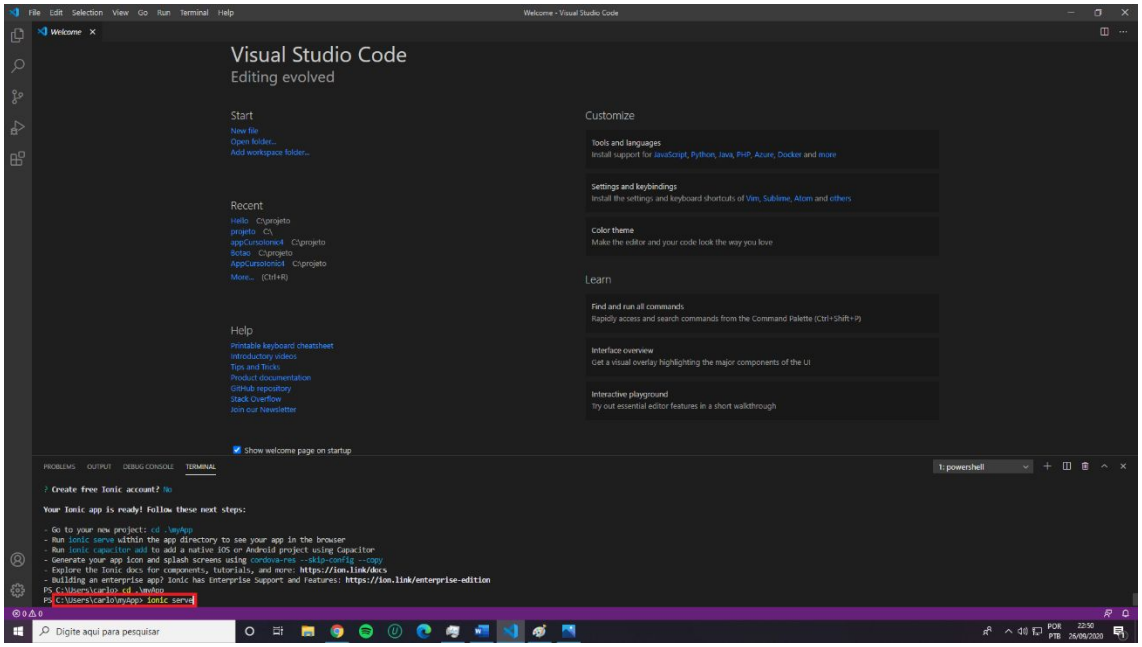
Por último, o terminal perguntará se você possui interesse em criar uma conta ionic. Para o que faremos em seguida, não é necessário fazer algum login. Responda com “no”.

Executando o aplicativo



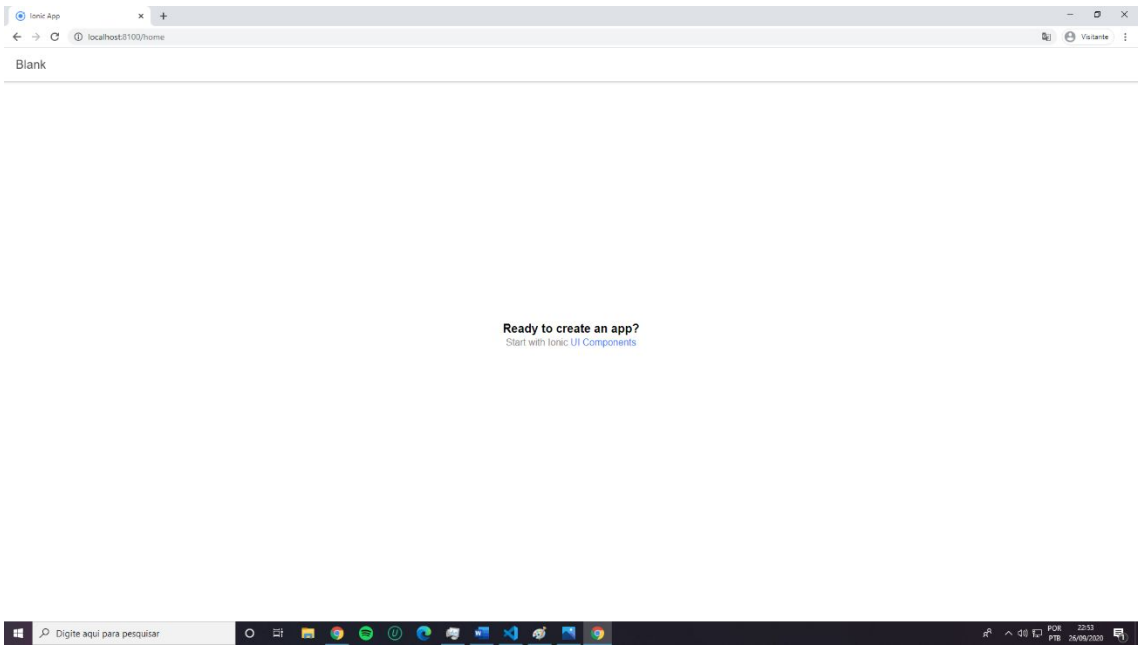
Antes de executarmos o programa, encontra-se necessário entrar na pasta do aplicativo. Atualmente estamos na pasta “C:\Users\carlo”. Digite o seguinte código para entrar na pasta do seu aplicativo:

cd .\myApp

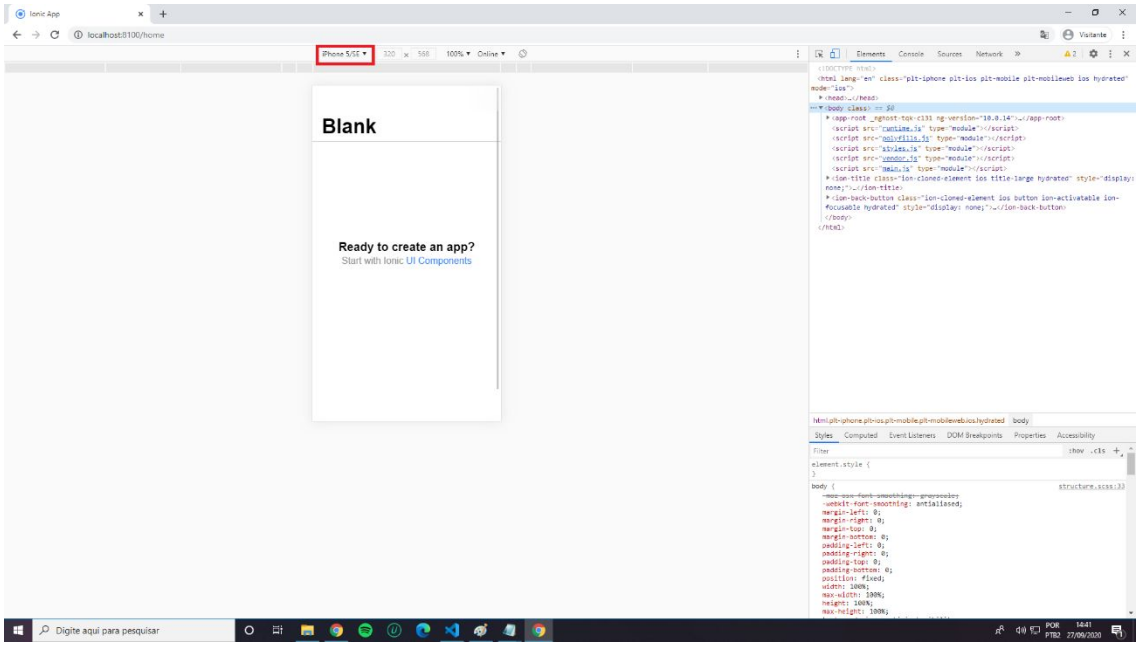


Por último, após entrar na pasta do seu aplicativo, você poderá executar o seu programa tranquilamente. Digite o seguinte código:

Ionic serve



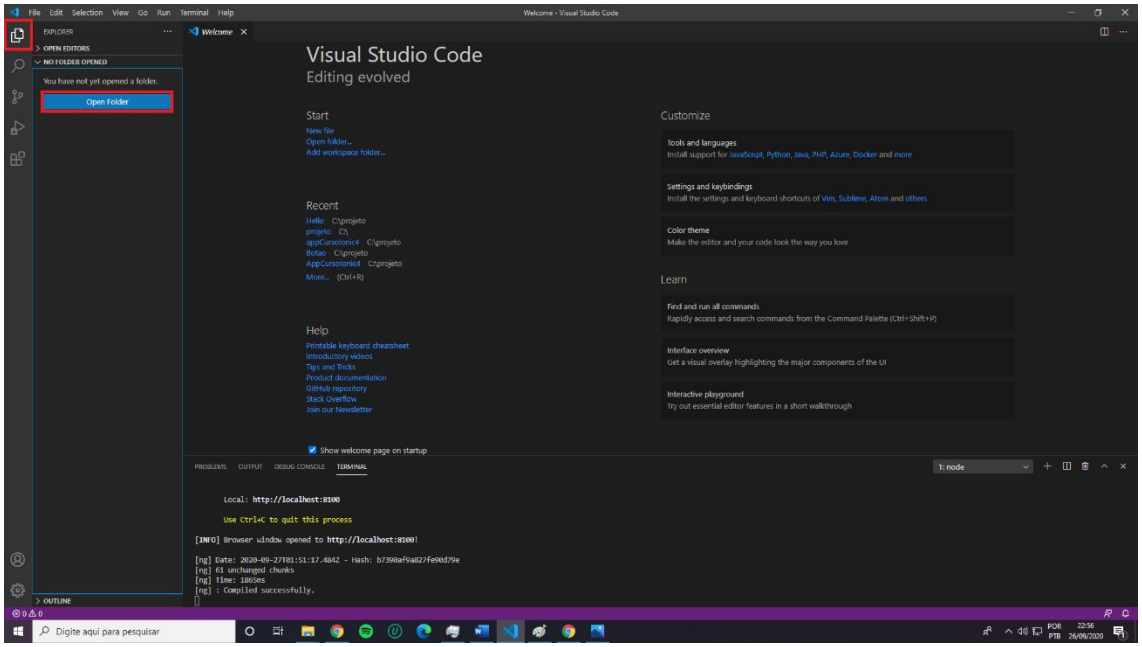
Compilando o aplicativo, ele será aberto em seu navegador. Para uma melhor visualização é recomendado deixar o navegador com esse formato, pois algumas ferramentas de design funcionam somente na tela mobile. Para realizar esse procedimento, apenas pressione a tecla F12 do seu teclado.



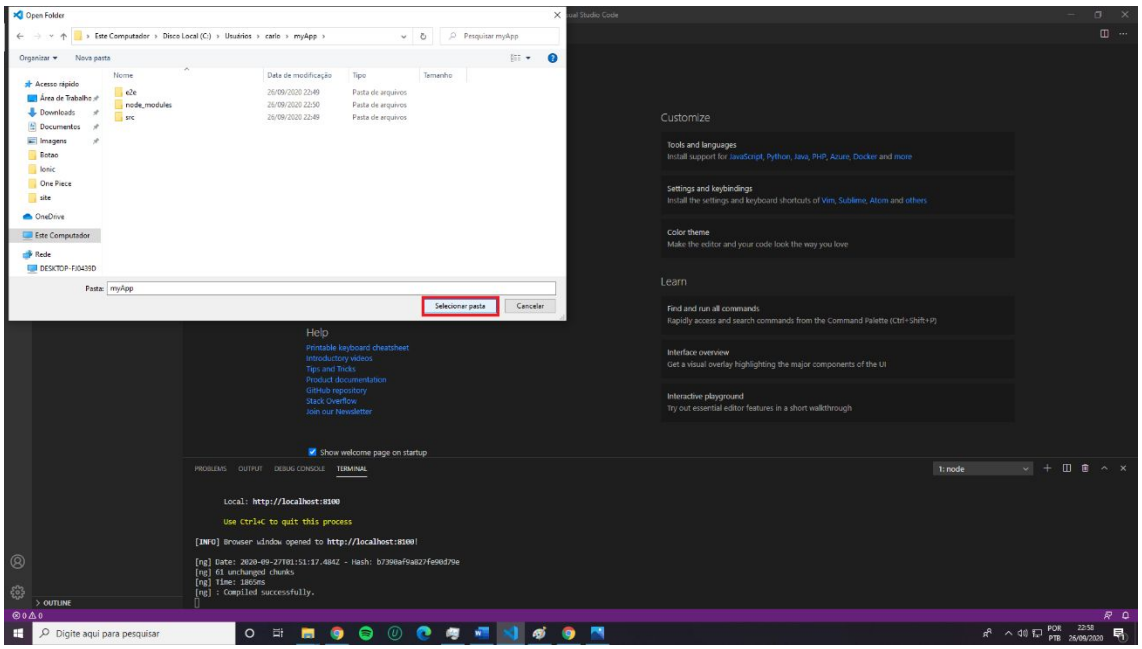
Após o procedimento, o design do aplicativo será adaptado para o formato mostrado na imagem acima. É possível alterar a tela para qualquer outra resolução que desejar. Apenas

selecione onde está marcado em vermelho e escolha o dispositivo a ser simulado em seu navegador.

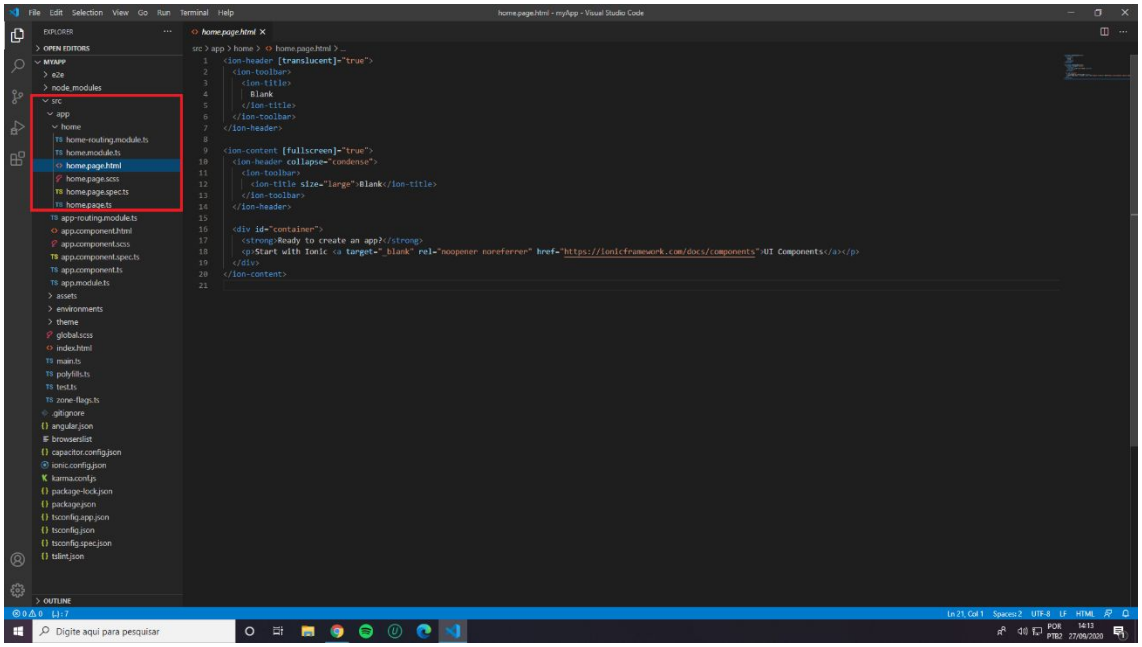
Programando o aplicativo



Para iniciar a programação do seu aplicativo, é necessário abrir a pasta com os arquivos. Existem diversas formas de localizar a pasta. Utilizaremos a seguinte forma: **Clique na ferramenta Explorer > Open folder** e então, localize o diretório da pasta.

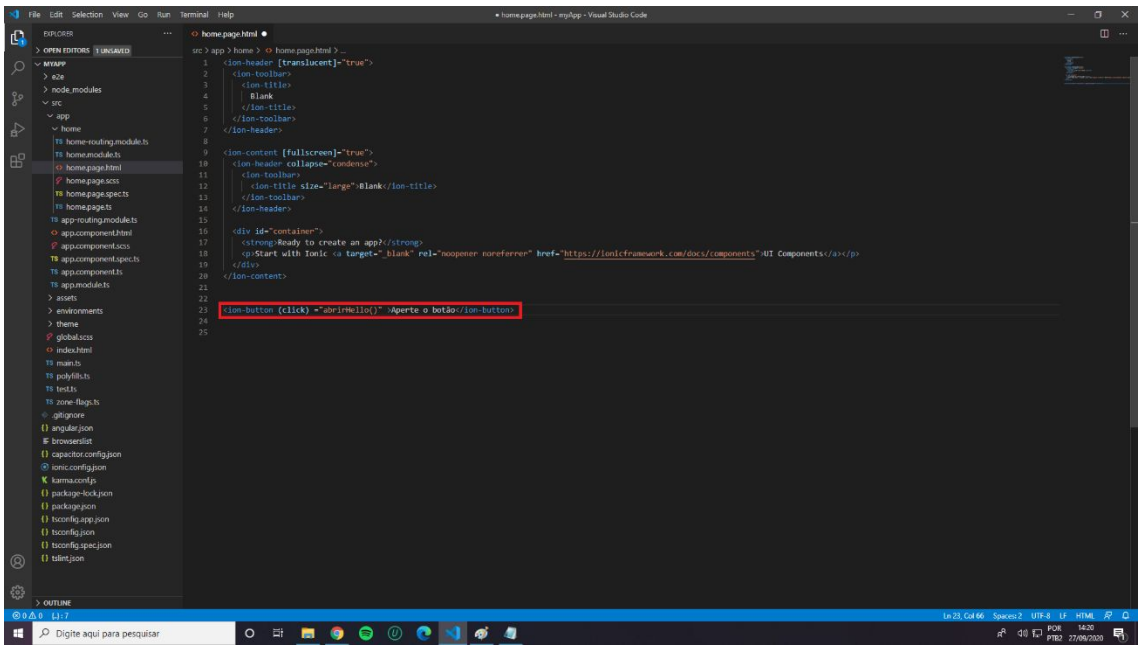


Após localizar a pasta, pressione em “selecionar pasta”.



Inicialmente, precisamos encontrar o código fonte do nosso aplicativo. Para isso acesse o caminho: **src > app > home.**

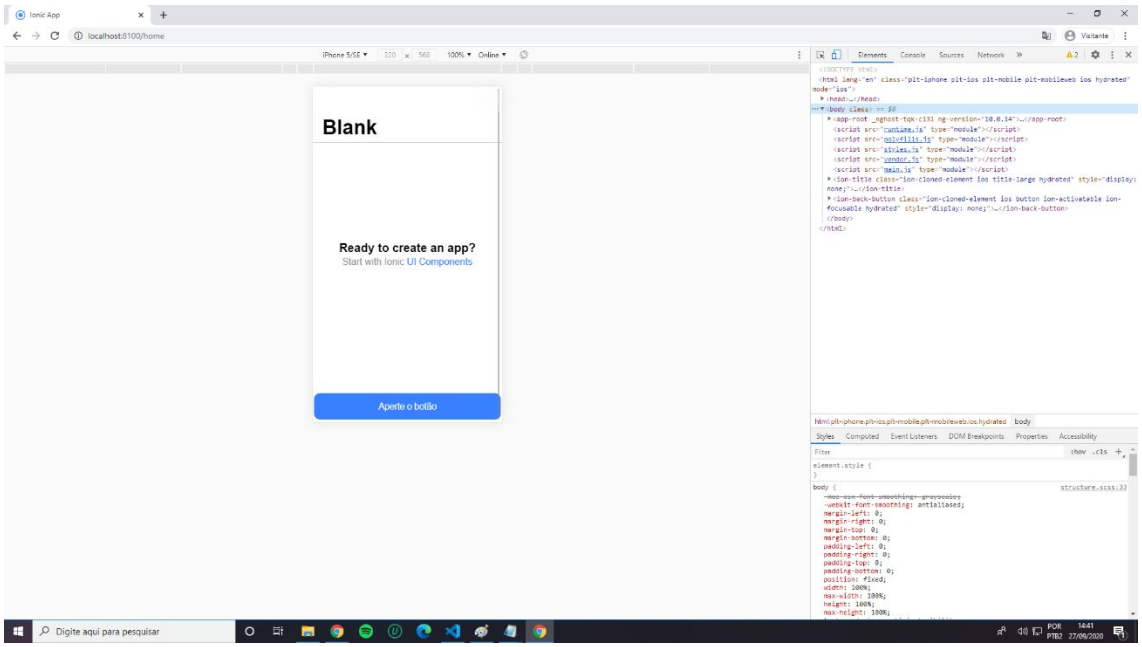
Para programarmos o nosso famoso “Hello world” utilizaremos apenas dois arquivos desta pasta. O arquivo “home.page.html é onde vamos adicionar o botão e outras ferramentas de design do aplicativo.



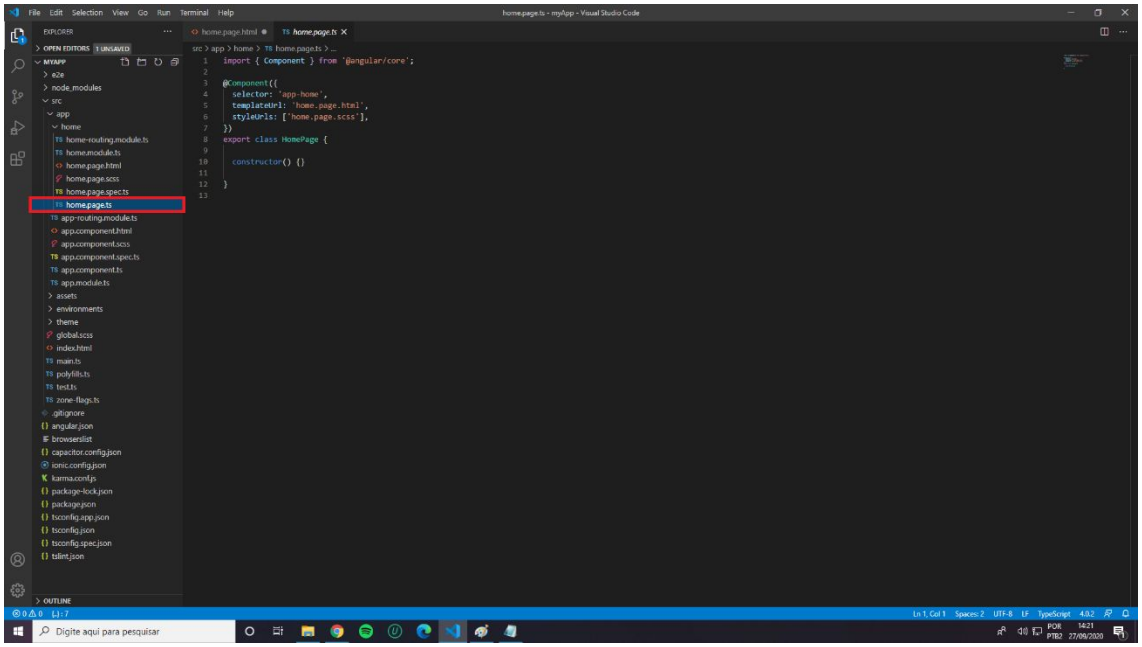
Agora vamos programar o botão que irá abrir a nossa janela escrito “Hello world”. A tag utilizada para a versão atual do ionic (6.11.8) deve ser escrita da seguinte maneira:

<ion-buttom> Nome do botão </ion-buttom>

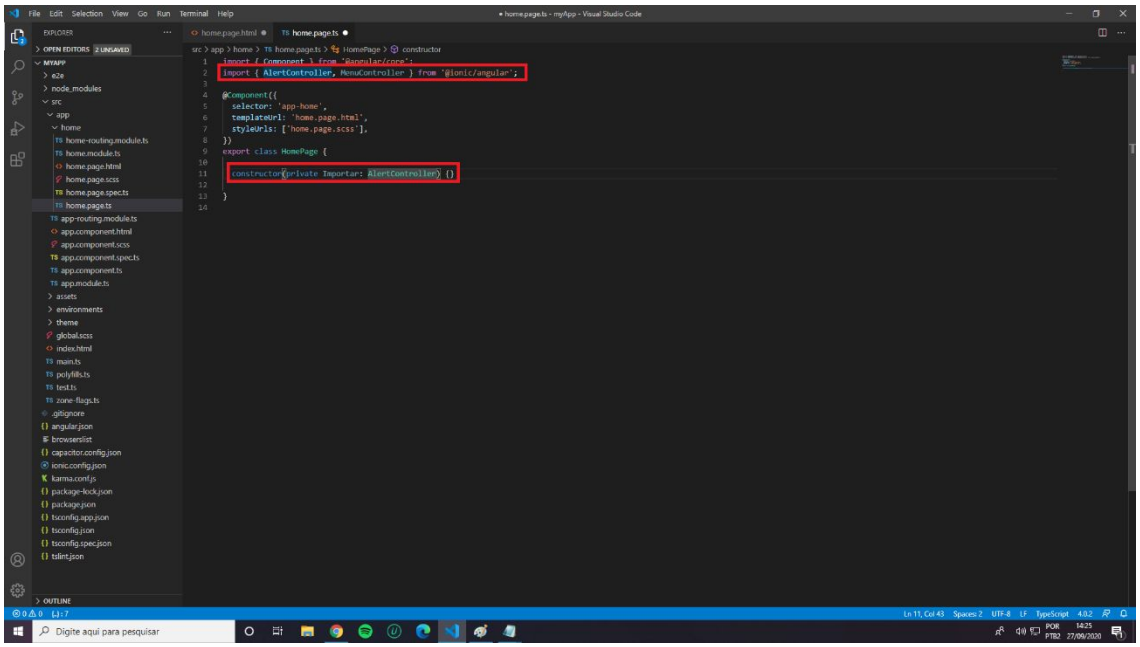
Como queremos programar um botão que abrirá uma janela. Precisamos declarar que ao acontecer um evento “click” do botão, abrirá uma janela. Para isso acontecer, é preciso criar uma função, assim como mostrado na imagem acima.



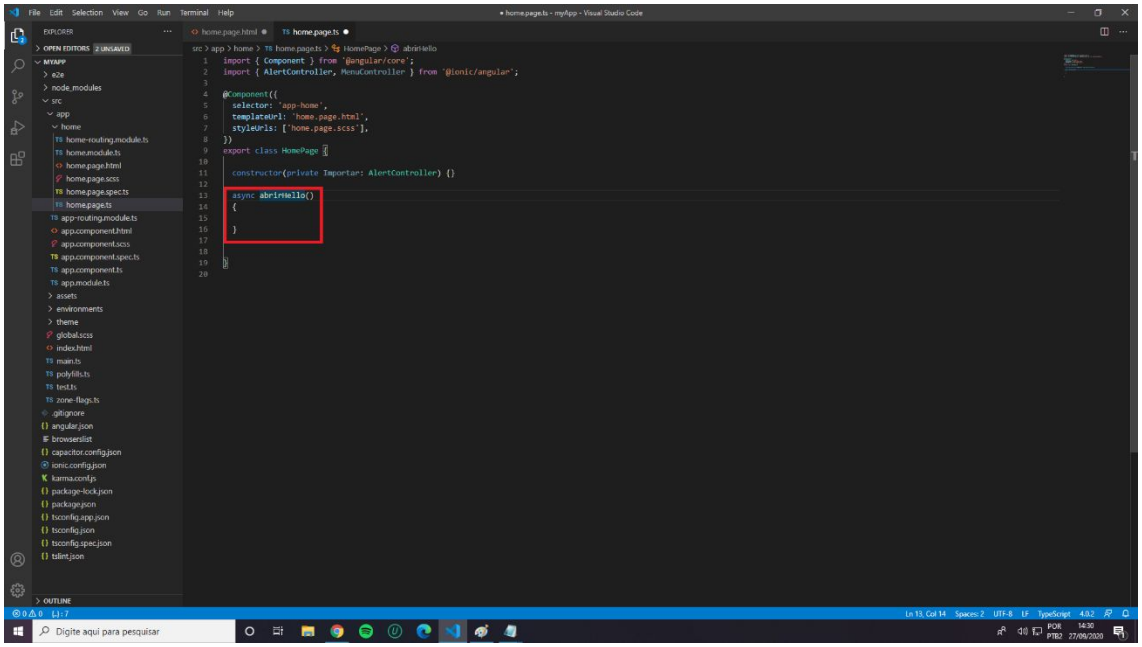
Após salvar as mudanças no aplicativo, ao observarmos o aplicativo no navegador, ele já apresenta o botão na parte inferior da tela. Como ainda programamos uma função ao botão, temporariamente ele não terá ações.



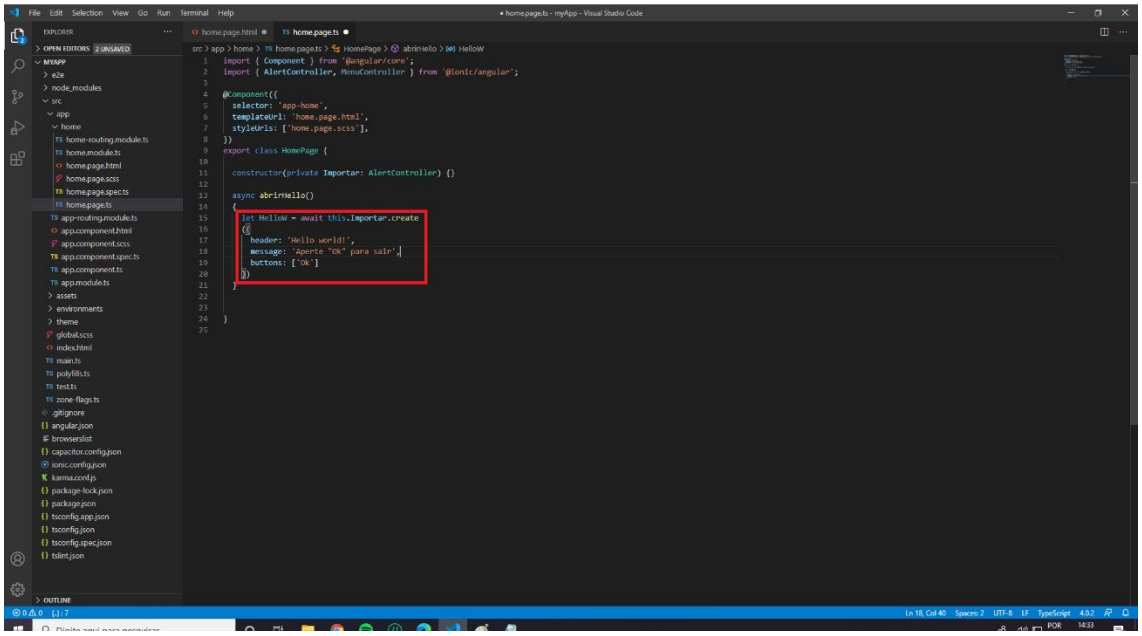
Retornando ao aplicativo, acesse o arquivo “home.page.ts”. Essa biblioteca servirá para programarmos o evento “click” do botão.



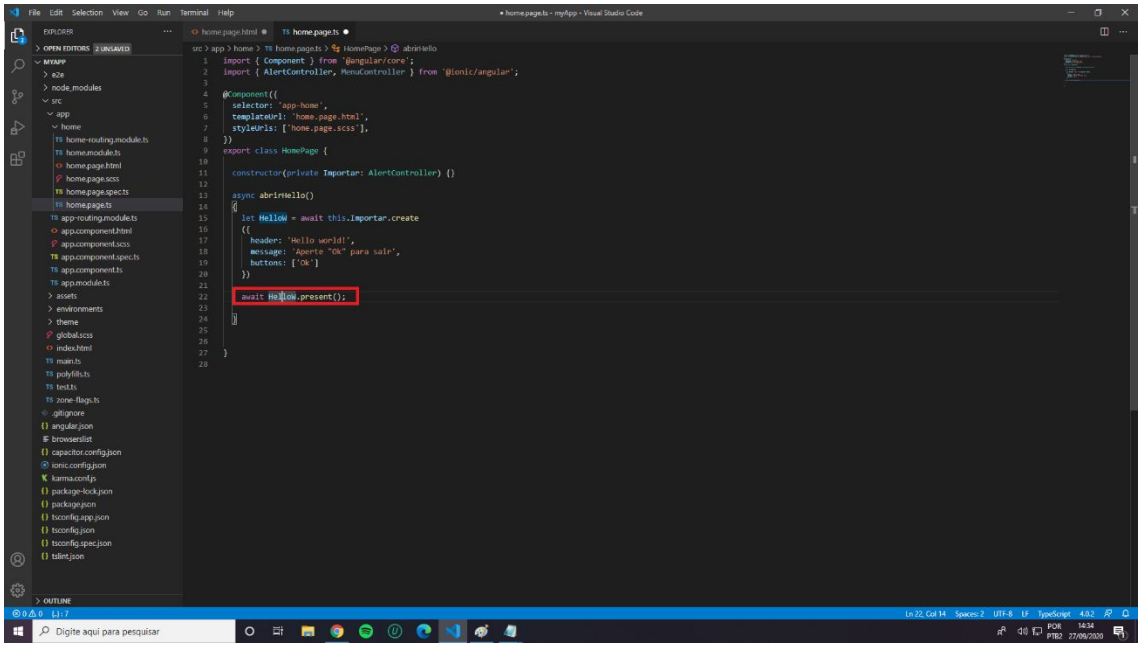
Para que ocorra o evento do botão, é necessário importarmos as funções da ferramenta “alert” do framework ionic. Devido a isso, criei uma variável chamada “importar”, que chama as funções “alertController”. Além disso, para que haja uma maior agilidade, o próprio software de programação declara automaticamente o código de importação.



Em seguida, criaremos uma função com o mesmo nome que atribuímos para o nosso botão “abrirHello()”.

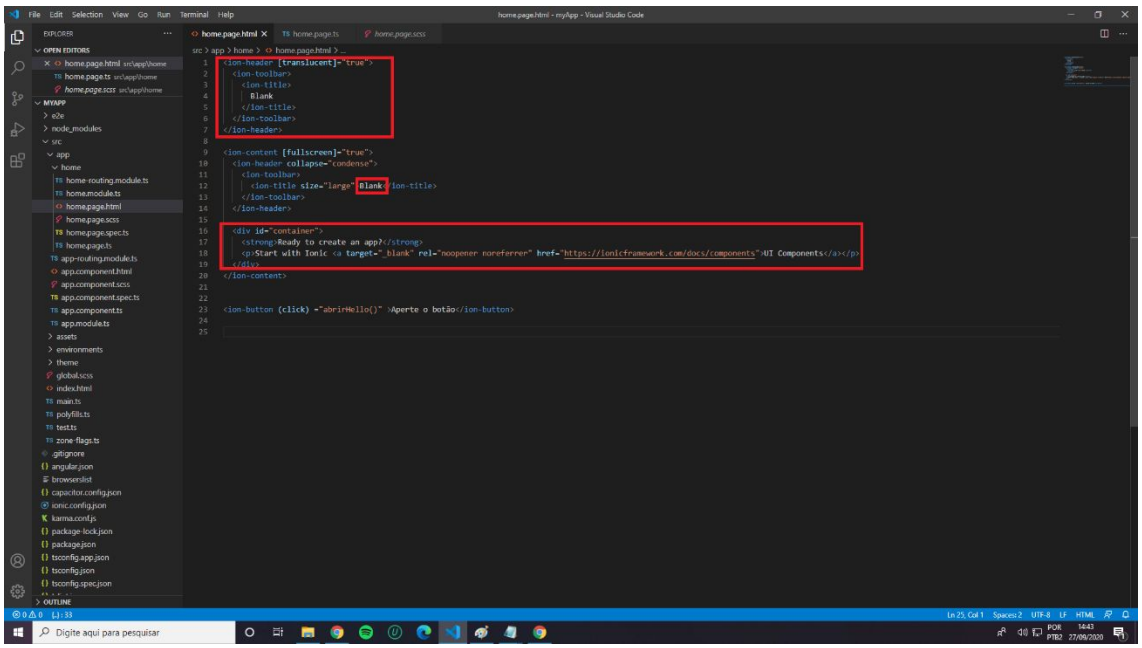


Dentro da função, informa o acontecimento do evento “click” do botão. Além disso, com as ferramentas “header e mensagem” respectivamente, declaro a mensagem e também o texto que informa como sair da janela criada.

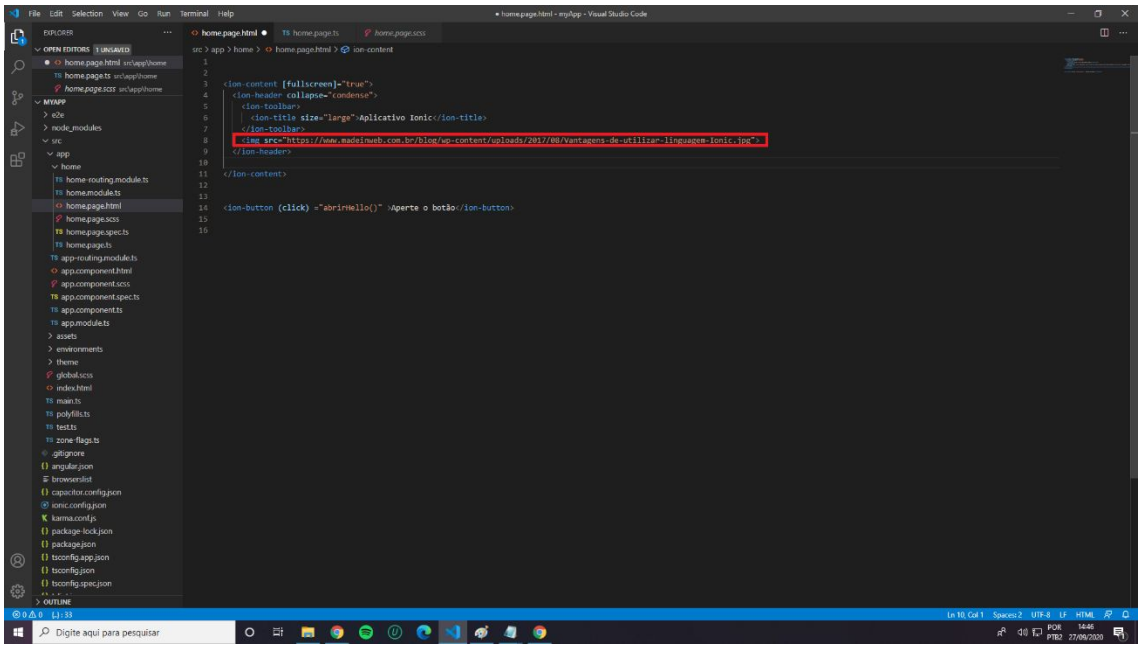


Por último, temos o código que retorna os dados para o usuário, assim concretizando o evento do botão.

Design do aplicativo

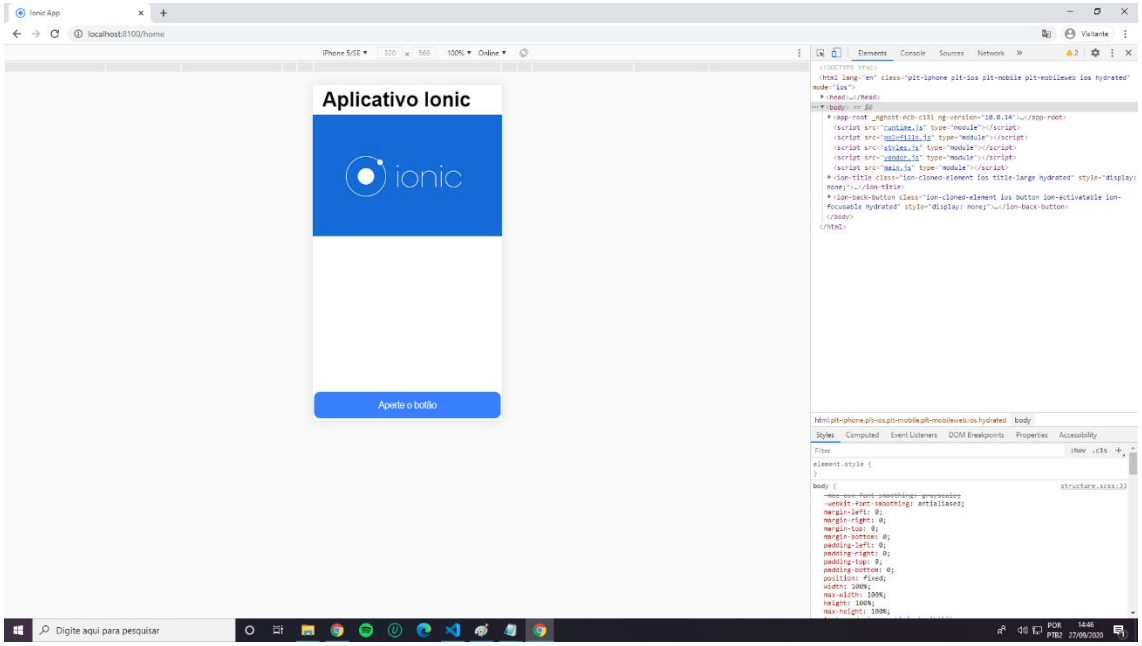


Para uma melhor apresentação estética do programa, apagaremos os três códigos grifados em vermelho. Como queremos alterar o título da página inicial, vamos substituir o “Blank” por “Aplicativo Ionic”

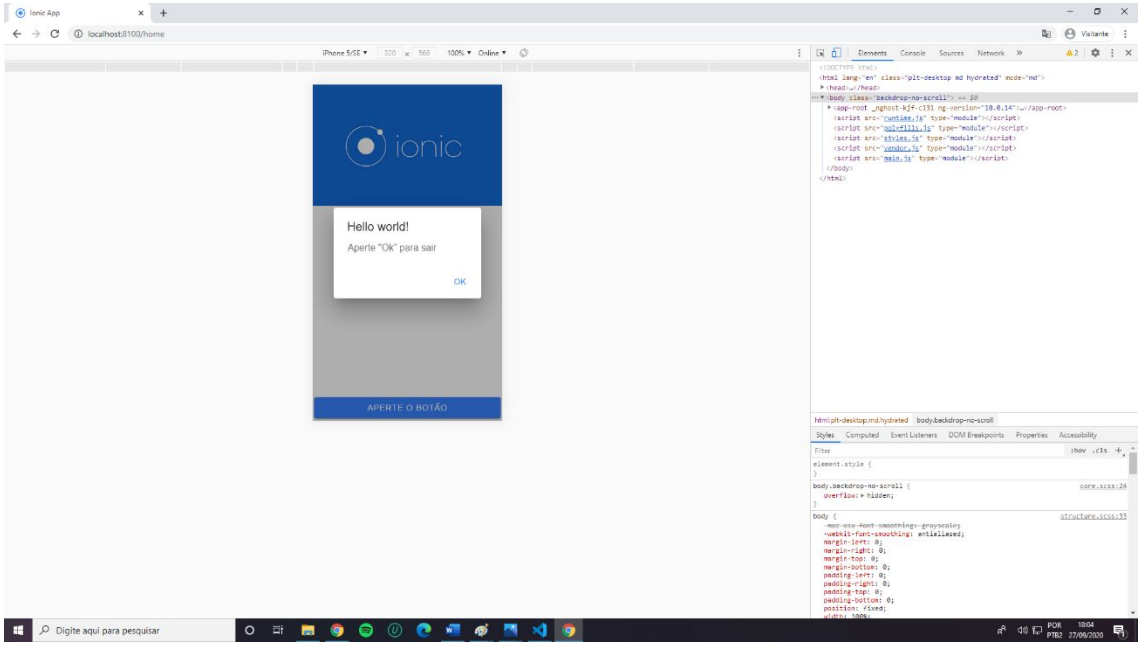


Após realizar os procedimentos, indico colocar alguma foto no aplicativo. Utilize a seguinte tag para isso:

<img src =”Link da imagem”>



Concluindo todas as alterações no Visual Code, temos essa aparência final.



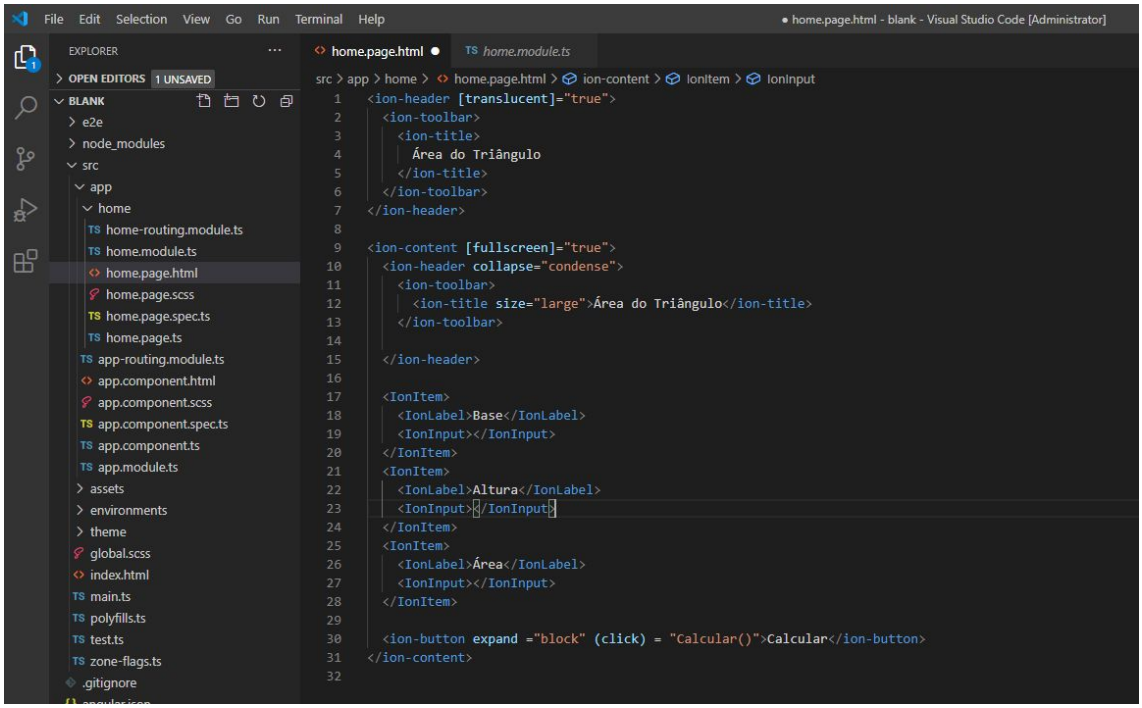
Quando o evento botão é acionado, ele apresenta está tela, pedindo para que o usuário aperte “ok” para sair da janela atual.

APP01

Para poder usar o Ionic deve baixar o Node.js, npm e o Visual Studio para poder editar o código. Para criar e executar seu projeto deve ir à pasta onde está ou deseja deixar o projeto utilizando o console de comando e digitar ionic start (para criar) e selecione que deseja usar o framework Angular e o template “Blank” e depois espere acabar de baixar tudo. Após terminar de criar, ainda na pasta do projeto, digite ionic serve para executar o projeto, após alguns segundos abrirá uma aba no seu navegador que estará no seu aplicativo.

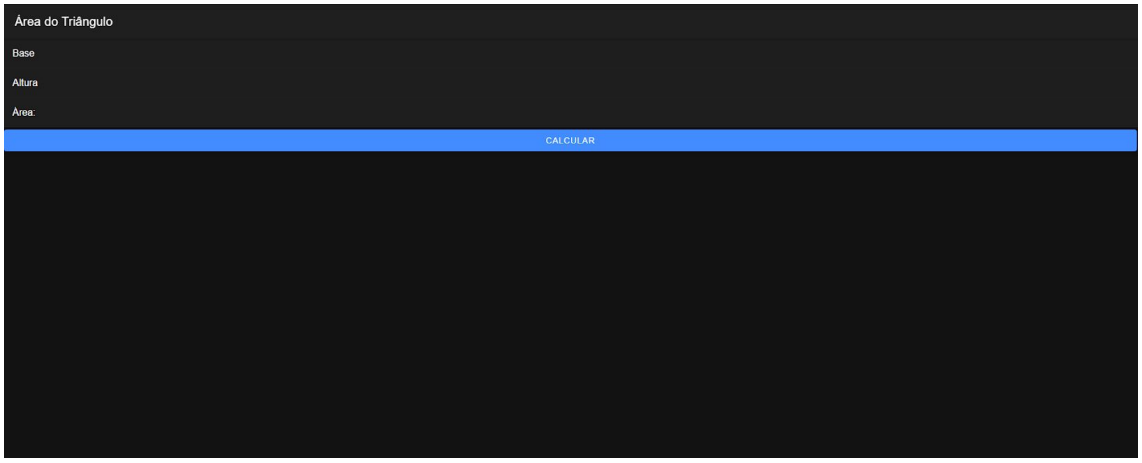
Após criar o aplicativo chamado “Blank”, utilizando o modelo pronto blank, dentro da pasta ...\blank\src\app\home, há o código fonte do aplicativo, home.page.html.

Dentro desse código é adicionado três componentes de Input, disponível em https://ionicframework.com/docs/components, dois para entrada dos valores da base e a altura, e um para exibir o valor da área do triângulo.

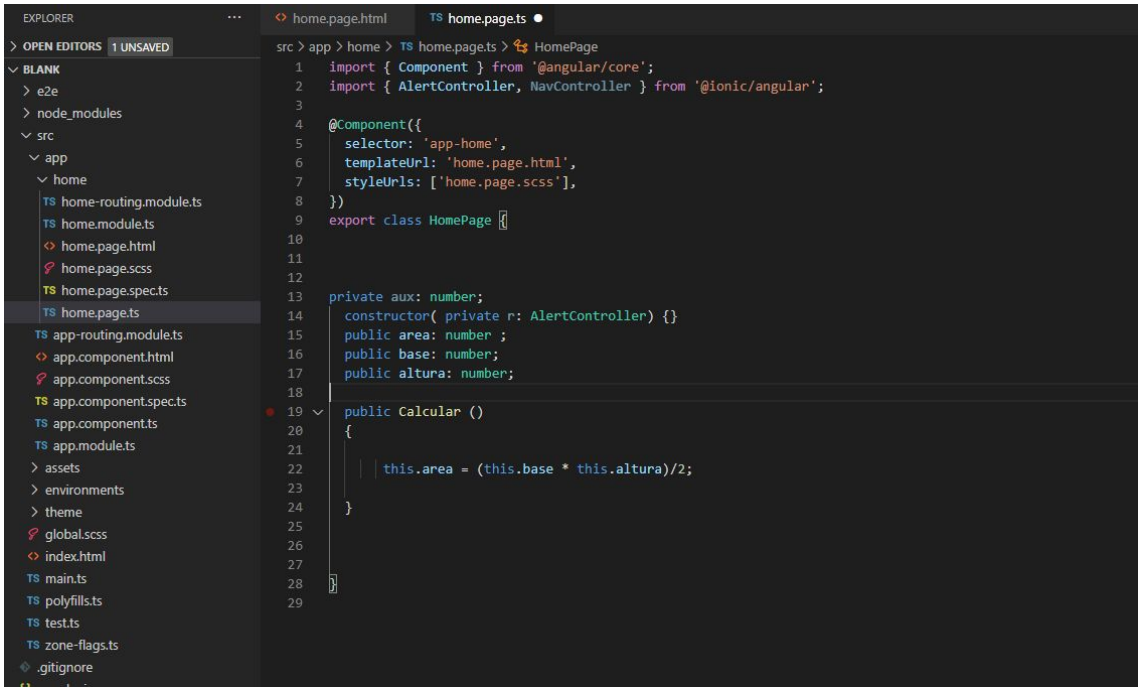


E um ion button para realizar o cálculo com seu click.

A página deverá ficar assim:

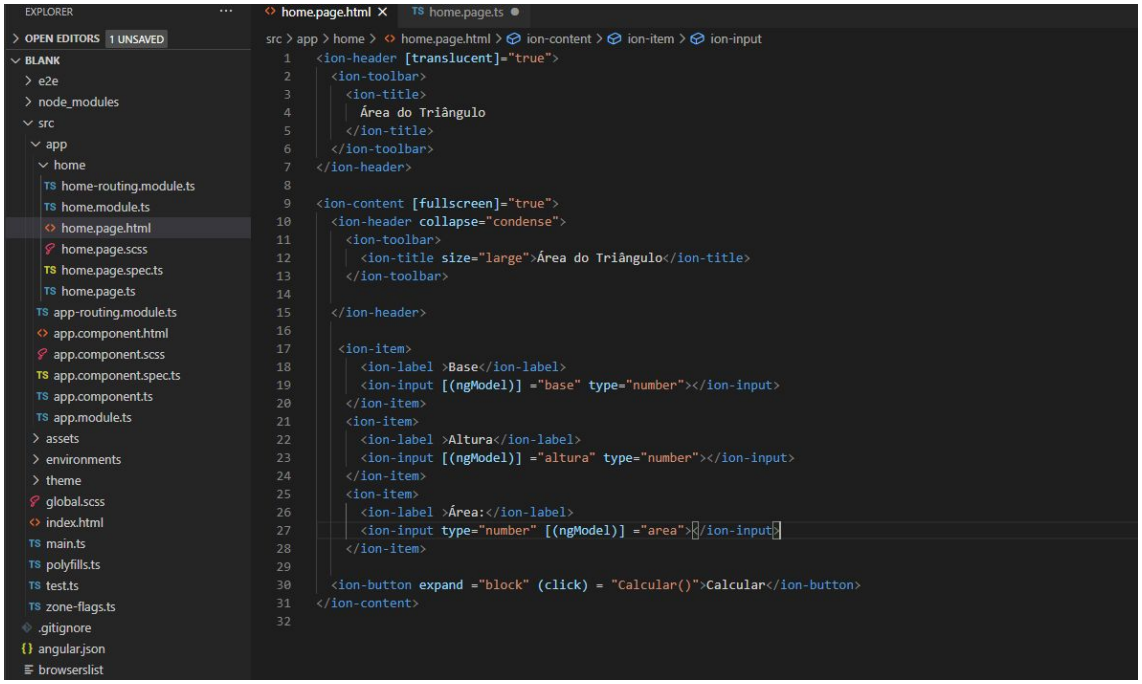


Dentro da classe home.page.ts, será criado o método de calcular do botão, utilizando a fórmula da base vezes a altura divido por dois, e as suas devidas variáveis públicas do tipo number: base, altura e área.

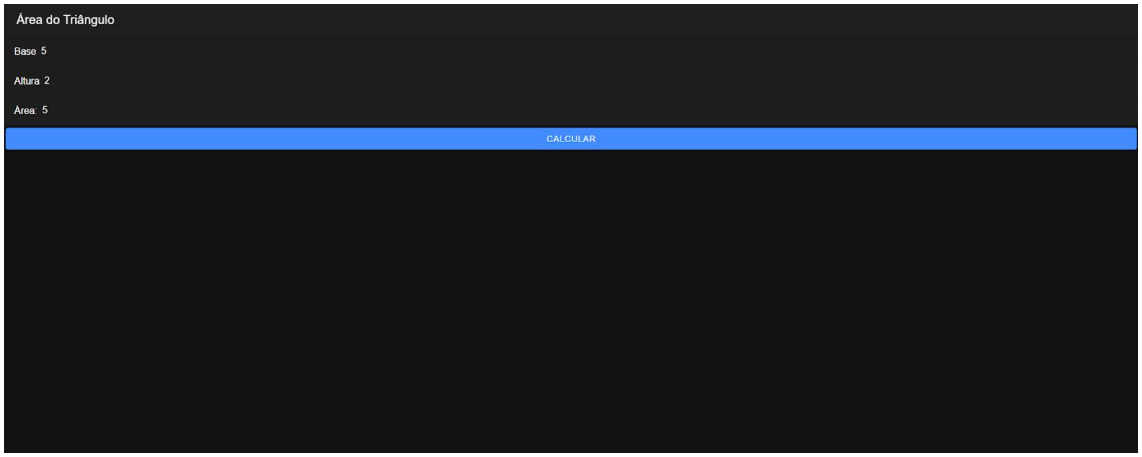


Voltando ao código fonte html, cada texto de cada um dos ion inputs, é definido como uma variável do tipo number(base, altura, área), permitindo que apenas números possam ser digitados no campo automaticamente.

E por fim, dentro do botão Calcular, é criado a ação do click do botão e inserido o método calcular nele.



E então é só executar e conferir os resultados.



APP02

Com uma nova aplicação criada, feita para o nosso novo programa, devemos instalar os plugins de Geolocalização e para localizar o endereço.

Geolocalização:

$ ionic cordova plugin add cordova-plugin-geolocation

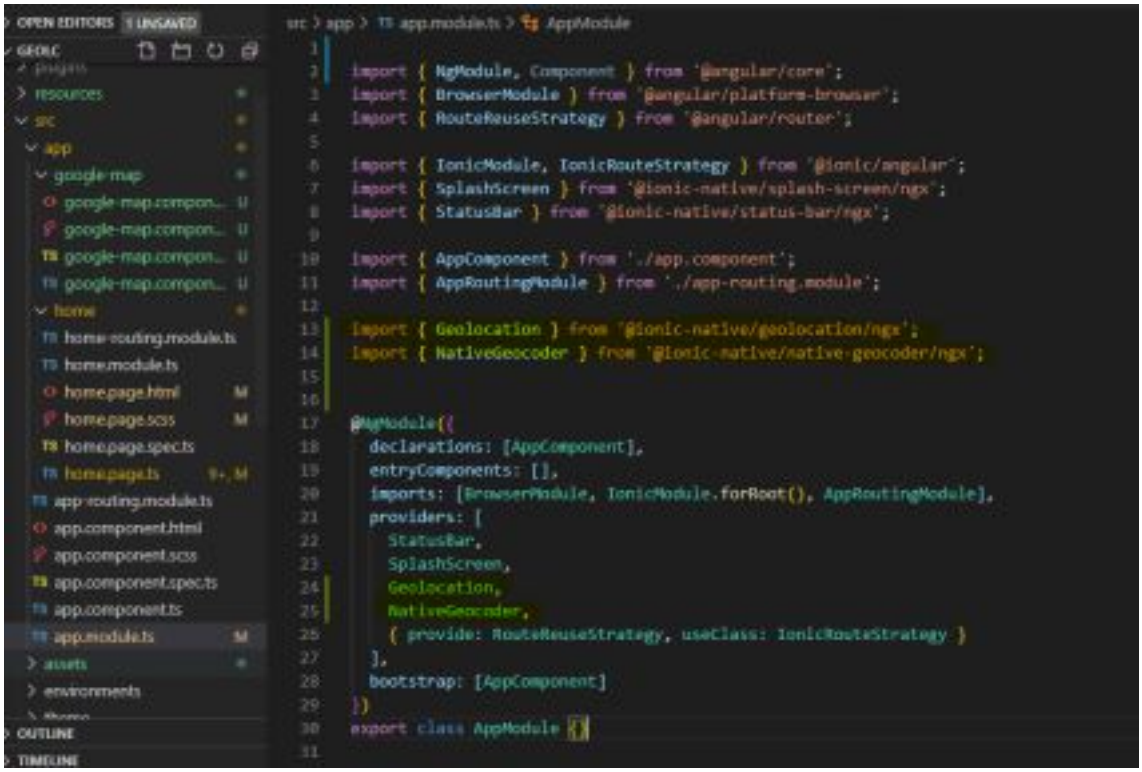
$ npm install @ionic-native/geolocation

Geocode

$ ionic cordova plugin add cordova-plugin-nativegeocoder

$ npm install @ionic-native/native-geocoder

Depois de instalá-los, devemos importar os plugins nos módulos da aplicação:

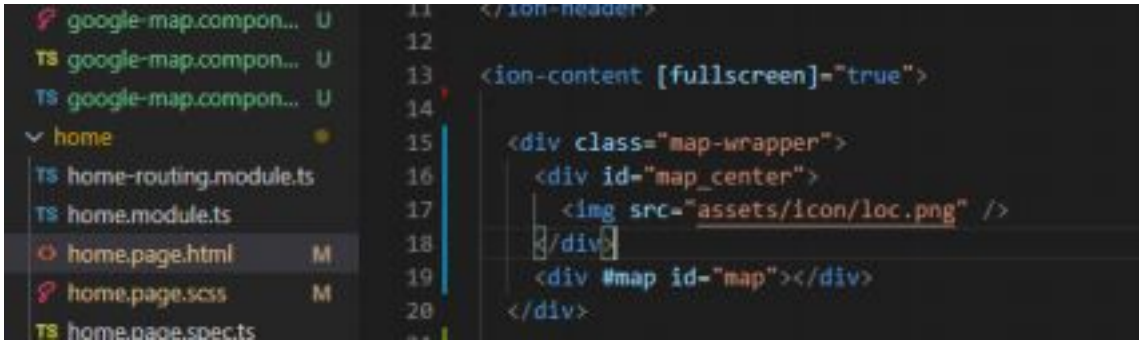


E é necessário ir até a página do google *https://developers.google.com/maps/documentation/javascript/get-api-key*, para conseguir uma chave de acesso, necessária para utilizar o serviço do google map. Após realizar os passos descritos no site, devemos pegar a chave e inseri-la no seguinte script no index.html.

```
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
<script defer
  src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY">
</script>
</head>
```

Insira sua chave de acesso em “YOUR_KEY”.

Para mostrar o mapa e a localização, devemos um div no home.page.html para abrir o google map.



E também, será feito um grid para inserir as informações de latitude e longitude, e também um button para marcar sua localização.

```
  <ion-toolbar color="warning">
    <ion-button (click)="loadMap()" shape="round" color="dark">
      <ion-icon slot="start" name="locate"></ion-icon>
      Localizar
    </ion-button>
    <ion-title slot="end">Google Map</ion-title>
  </ion-toolbar>

</ion-header>

<ion-content [fullscreen]="true">


  <ion-grid>
    <ion-row>
      <ion-col size="3">
        <b>Latitude</b>
      </ion-col>
      <ion-col>
        {{latitude}}
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size="3">
        <b>Longitude</b>
      </ion-col>
      <ion-col>
        {{longitude}}
      </ion-col>
    </ion-row>

  </ion-grid>
```

Para fazer funcionar a geolocalização, e também o mapa, devemos modificar o componente file do seguinte modo(home.page.ts):

Importando o plugin –

```
app > home > TS home.page.ts > HomePage > ngOnInit
  import { Component, ViewChild, ElementRef } from '@angular/core';

  import { Geolocation } from '@ionic-native/geolocation/ngx';
  import { NativeGeocoder, NativeGeocoderResult, NativeGeocoderOptions } from '@ionic-native/native-geocoder/ngx';

  declare var google;
```

E declarando as variáveis e criando um método para deixar o mapa pré carregado.

```
export class HomePage {

  @ViewChild('map', { static: false }) mapElement: ElementRef;
  map: any;
  latitude: number;
  longitude: number;

  constructor(
    private geolocation: Geolocation,
    private nativeGeocoder: NativeGeocoder) {
  }


  ngOnInit() {
    this.loadMap();
  }
```

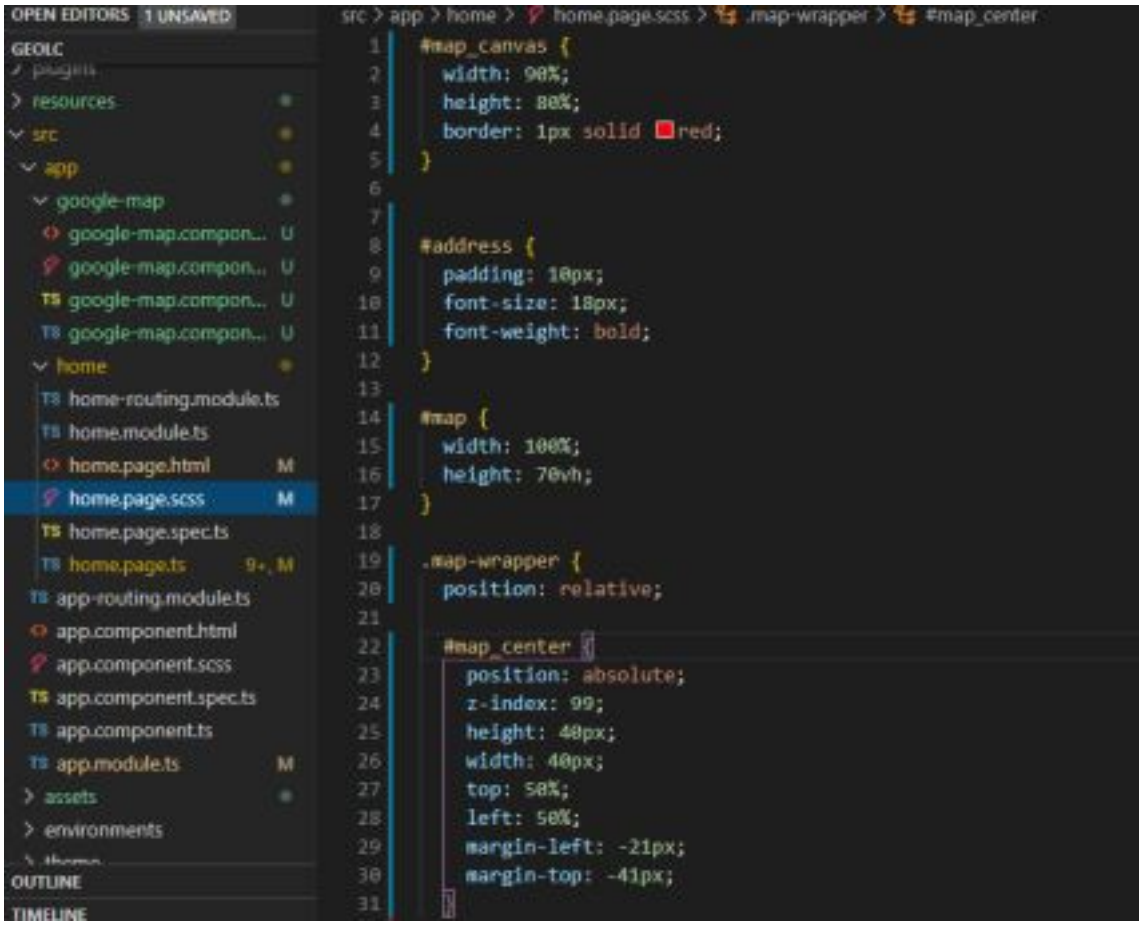
E o seguinte método para pegar a latitude e longitude a partir do plugin de geolocalização:

```
loadMap() {
  this.geolocation.getCurrentPosition().then((resp) => {

    this.latitude = resp.coords.latitude;
    this.longitude = resp.coords.longitude;

    let latLng = new google.maps.LatLng(resp.coords.latitude, resp.coords.longitude);
    let mapOptions = {
      center: latLng,
      zoom: 15,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    }

    this.getAddressFromCoords(resp.coords.latitude, resp.coords.longitude);

    this.map = new google.maps.Map(this.mapElement.nativeElement, mapOptions);

    this.map.addListener('dragend', () => {

      this.latitude = this.map.center.lat();
      this.longitude = this.map.center.lng();

      this.getAddressFromCoords(this.map.center.lat(), this.map.center.lng())
    });

  }).catch((error) => {
    console.log('Erro na localização', error);
  });
}
```

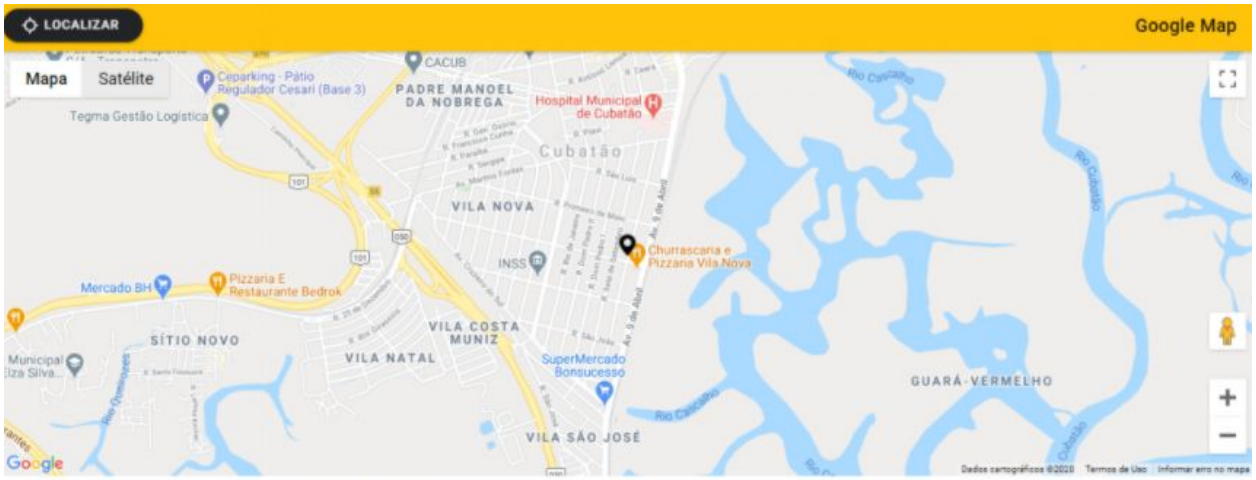
E finalmente, para encontrar sua localização a partir dos dados de latitude e longitude, devemos criar o seguinte método:

```
  getAddressFromCoords(lattitude, longitude) {
    console.log("getAddressFromCoords " + lattitude + " " + longitude);
    let options: NativeGeocoderOptions = {
      useLocale: true,
      maxResults: 5
    };

    this.nativeGeocoder.reverseGeocode(lattitude, longitude, options)
  }
}
```

Para o design de nosso programa, foi utilizado o seguinte css:



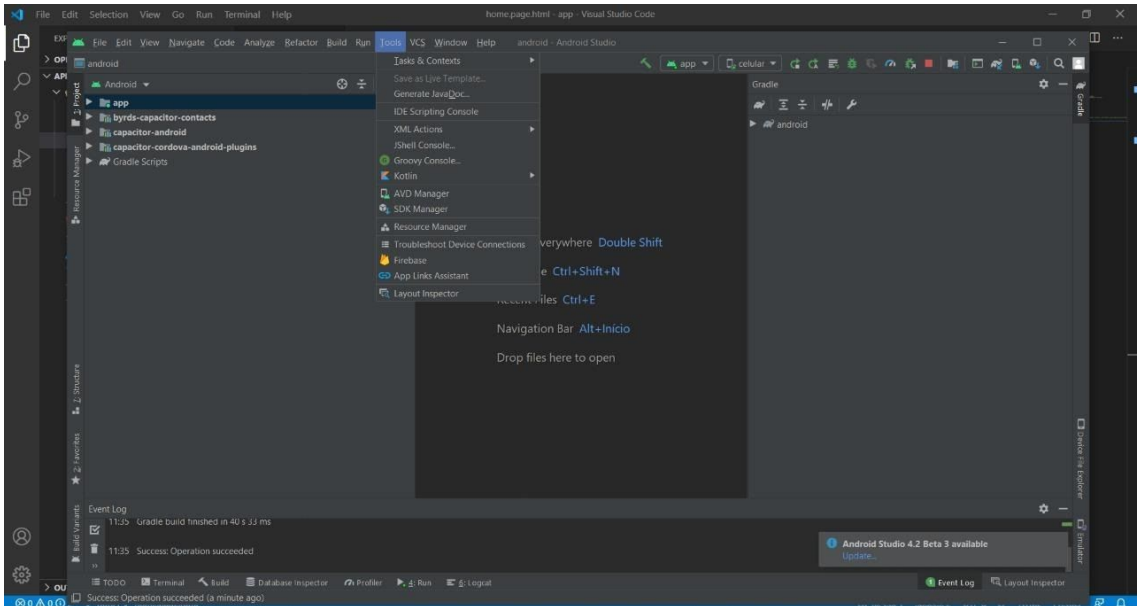
E então este será o nosso resultado:



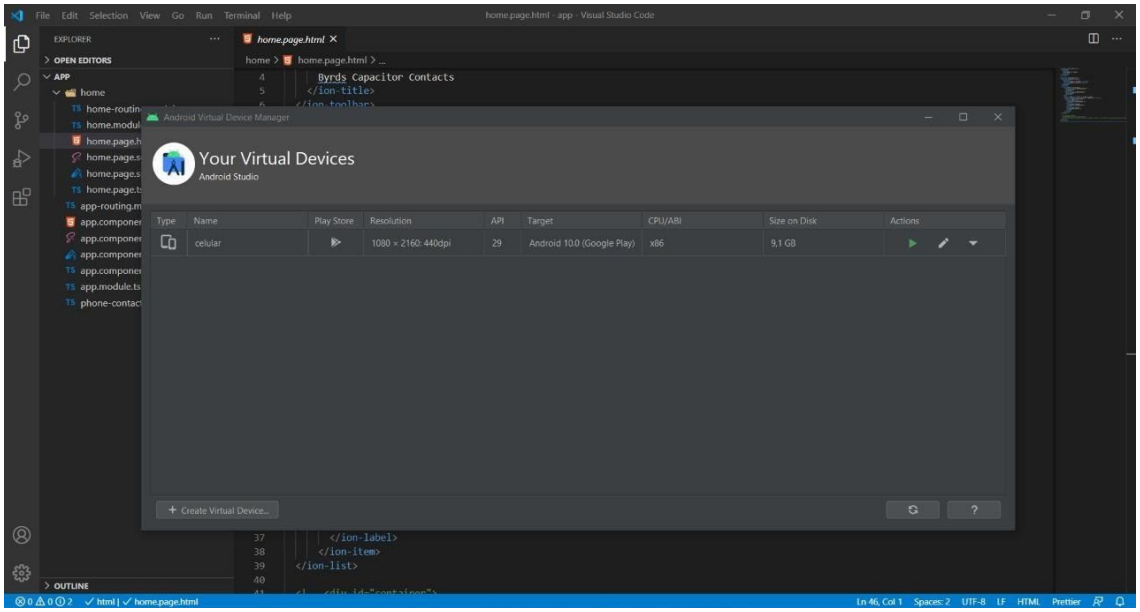
Latitude	-23.8963952
Longitude	-46.4221515

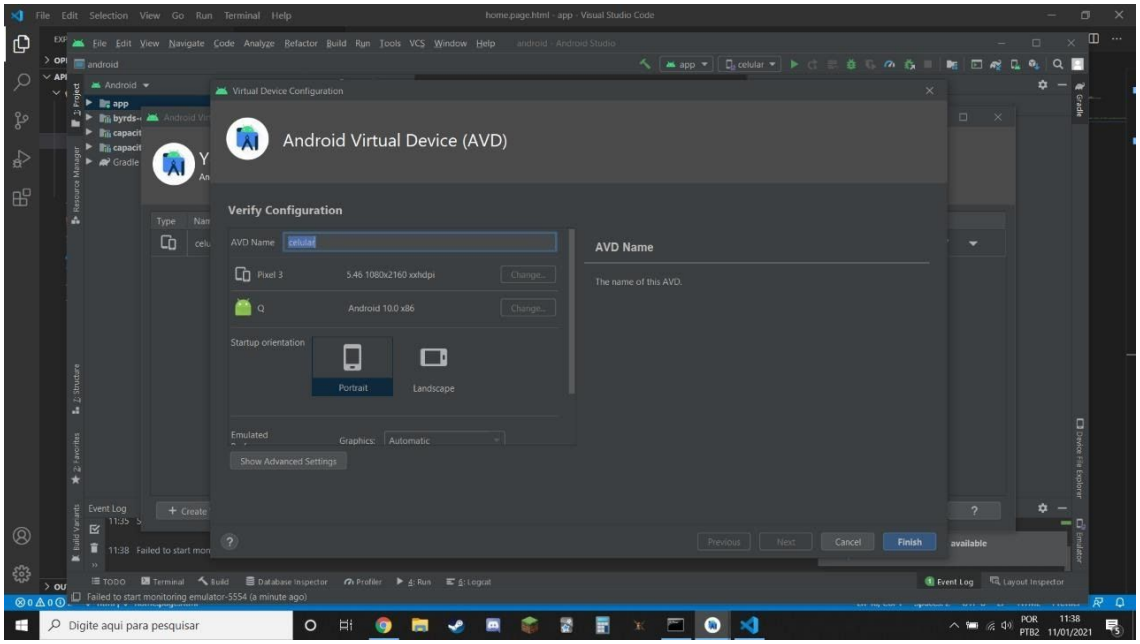
APP03

Para esta aplicação, vamos utilizar o emulador de android do Android Studio. Após criar uma nova aplicação no Android Studio, vá em Tools e AVD Manager para criar uma emulação do celular:



Crie um celular neste estilo:





E então, após gerar o emulador, vamos criar a pasta do projeto no Visual Studio.

Após sua criação, dentro do código fonte, devemos um searchbar para filtrar os contatos:

```
<ion-toolbar>
  <ion-searchbar placeholder="Pesquisar contatos"
                 [(ngModel)] = "queryText"
                 (ionInput) = "filterContato($event)"
                  clearInput>
  </ion-searchbar>
  </ion-toolbar>
```

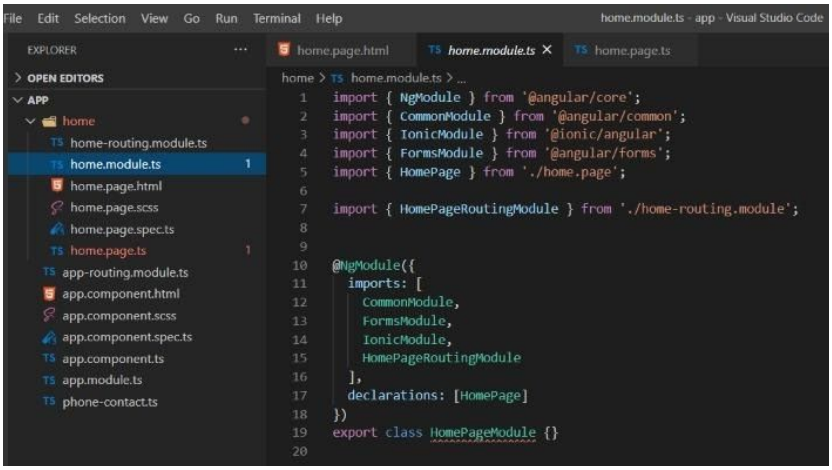
E criar 2 ion-button para pedir permissão para acessar os contatos do celular, e outro para listar os contatos:

```
<ion-list>
    <div class="ion-text-start">
      <ion-button (click)="getPermissions()" >
        Permitir Acesso
      </ion-button>
      <ion-button (click)="getContacts()">
        Pegar Contatos
      </ion-button>
    </div>
```

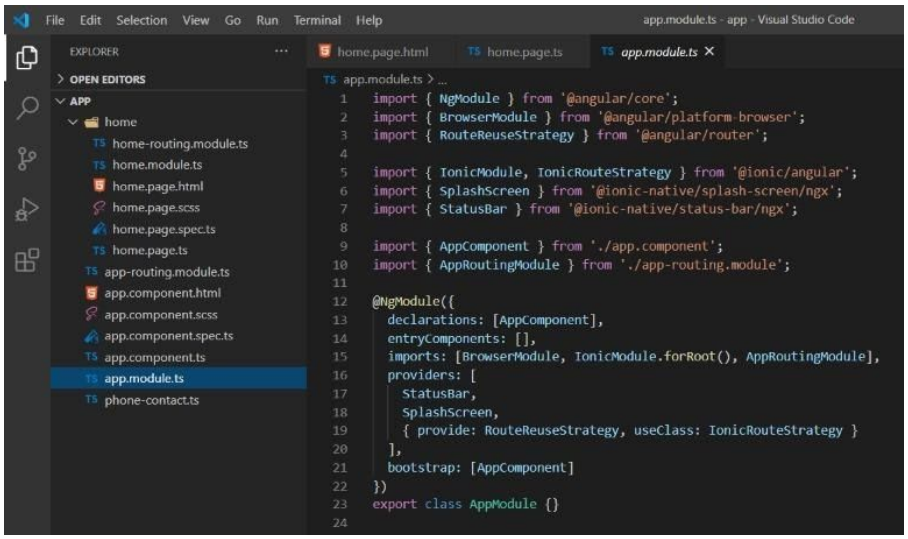
E então, dentro de um ion-item, colocamos o objeto let para que os contatos do array sejam listados um por um, e os dados exibidos em ion-text:

```
<ion-item *ngFor="let contact of contacts | async" >
  <ion-icon color="primary" slot="start" name="person"></ion-icon>
  <ion-label class="ion-text-wrap">
    <ion-text color="dark">
      <h3>{{contact.displayName}}</h3>
    </ion-text>
    <ion-text color="medium">
      <p>{{contact.phoneNumbers}}</p>
    </ion-text>
    <ion-text color="medium">
      <p>{{contact.emails}}</p>
    </ion-text>
  </ion-label>
</ion-item>
```

E no home.module.ts adicione essas importações:



No app.module.ts adicione estas importações:



E dentro da classe no home.page.ts, utilizando um array para armazenar os dados solicitados, é possível fazer a filtração a partir da digitação pelo usuário e retornando os valores identificados:

```
export class HomePage {
  contacts: Array<{displayName: any, phoneNumbers: any, emails: any}>;
  queryText: String;
  constructor() {}
```

```
filterContato(cont: any){
  let val = cont.target.value;
  if (val && val.trim() != ''){
  this.contacts = _.values(this.contacts);
  this.contacts = this.contacts.filter((contato) => { return (cont.displayName.toLowerCase().indexOf(val.toLowerCasa()) > -1);
  })
  }
  else {
  this.contacts = this.contacts;
  }
  }
```

É criado os métodos para pedir permissão e listar os dados registrados nos contatos do celular:

```
  async getPermissions(): Promise<void> {
    CapContacts.getPermissions();
  }

  async getContacts(): Promise<void> {
    CapContacts.getContacts().then(result => {
      console.log('result is:' , result);
      const phoneContacts: [PhoneContact] = result.contacts;
      this.contacts = of(phoneContacts);

    });
  }
}
```

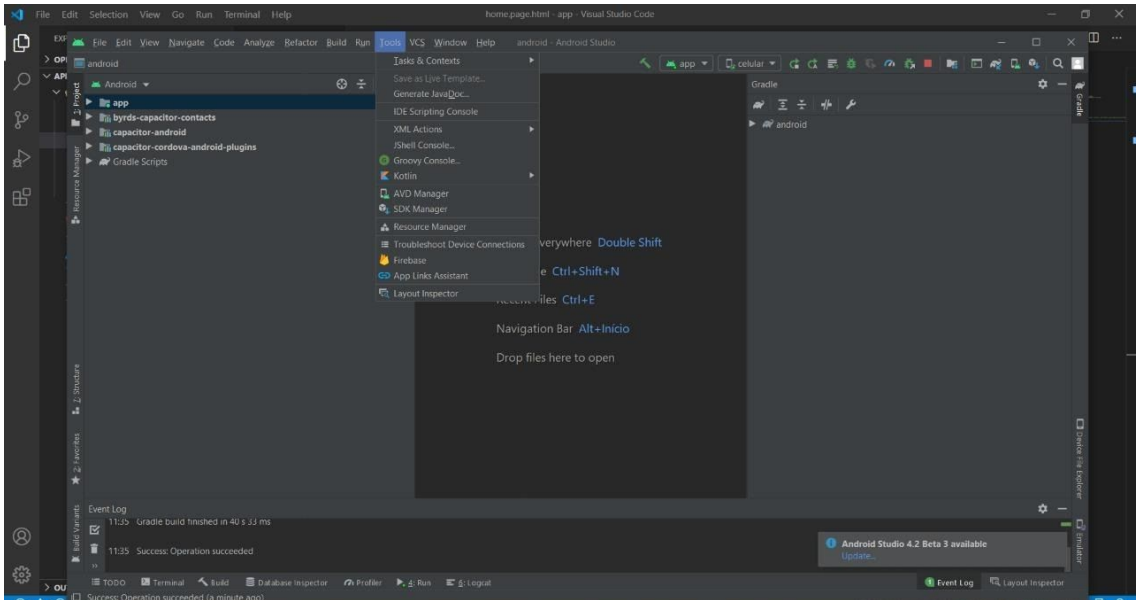
Após isso, para vincular e rodar o simulador com o projeto, devemos rodar os seguintes comando no terminal na pasta do projeto:

1. npm install
2. ionic build
3. npx cap sync

E no final:

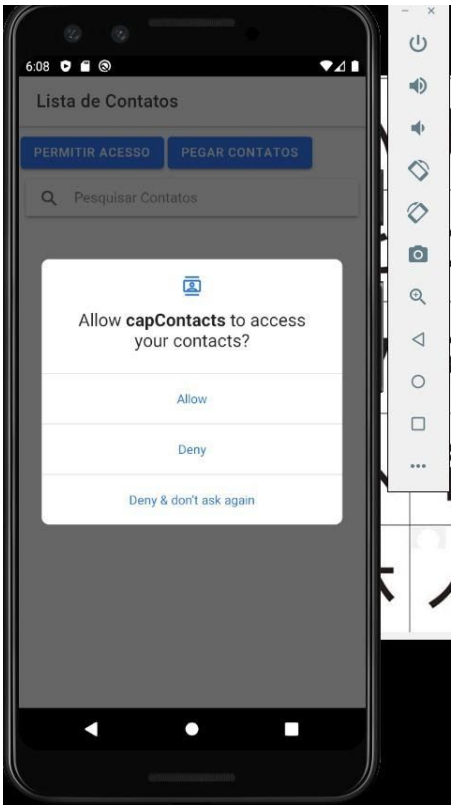
npx cap open android

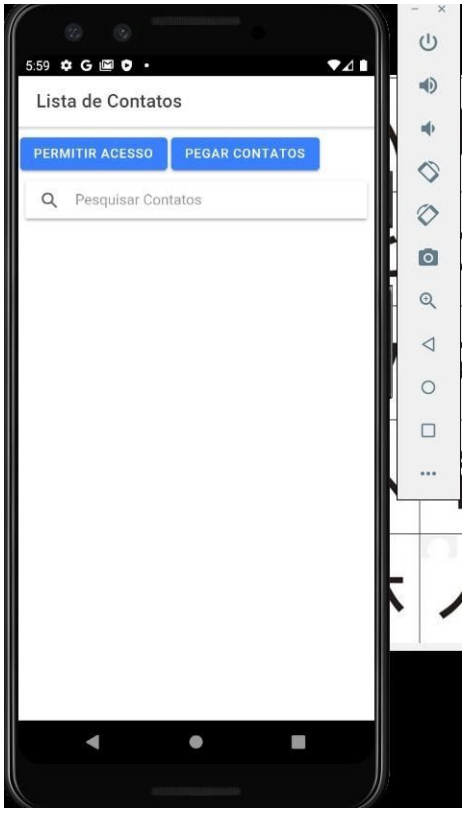
Abrindo esta pagina do android studio:



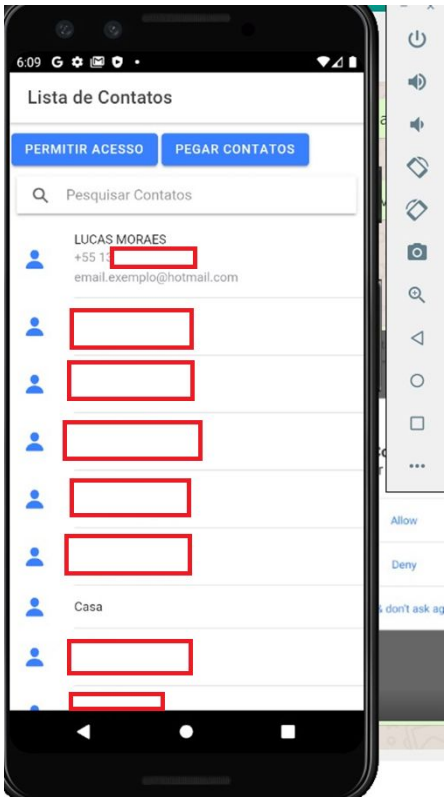
E então para abrir dentro do emulador, configure para executar no celular:

Abrindo a aplicação dentro do emulador:



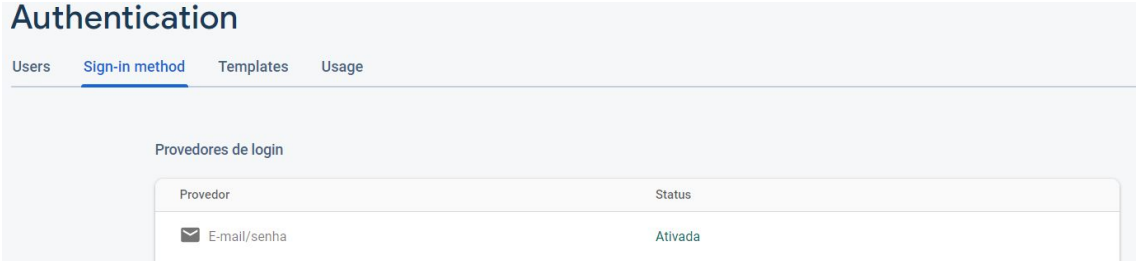


E clicando no botão Pegar Contatos, os dados são exibidos:



APP04

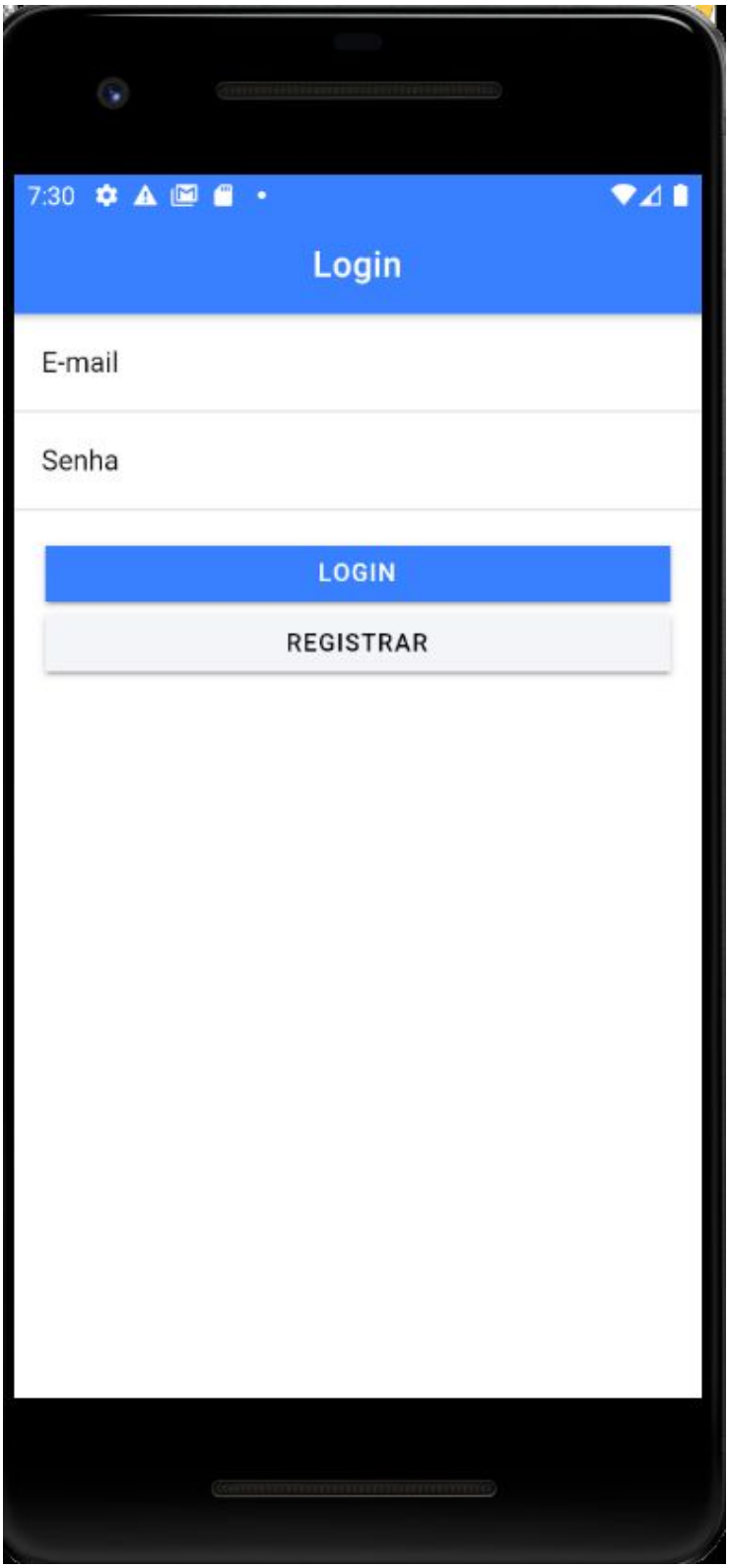
Antes de começar o projeto deve entrar no site do Firebase e criar um banco de dados antes e ative a autenticação por login.



Após ativar, vá no arquivo environment.ts e troque os dados placeholders com o do seu projeto, como a URL do banco de dados e a API.

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title class="ion-text-center">Login</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-item>
    <ion-label position="floating">E-mail</ion-label>
    <ion-input type="email" [(ngModel)]="user.email" required></ion-input>
  </ion-item>

  <ion-item>
    <ion-label position="floating">Senha</ion-label>
    <ion-input type="password" [(ngModel)]="user.password" required></ion-input>
  </ion-item>

  <ion-button
    class="ion-padding"
    color="primary"
    expand="full"
    (click)="login(user)"
    >Login</ion-button
  >

  <ion-button
    class="ion-padding"
    color="light"
    expand="full"
    [routerLink]="['/register']"
    >Registrar</ion-button
  >
</ion-content>
```

Agora começando no projeto, primeiro temos que criar uma página de inicial que será de login que irá criar ou logar usuários no banco de dados Firebase, com os mesmos campos de dados.



Deve-se criar um try catch para fazer a operação de login ou cadastro que consultará ou adicionará dados no Firebase.

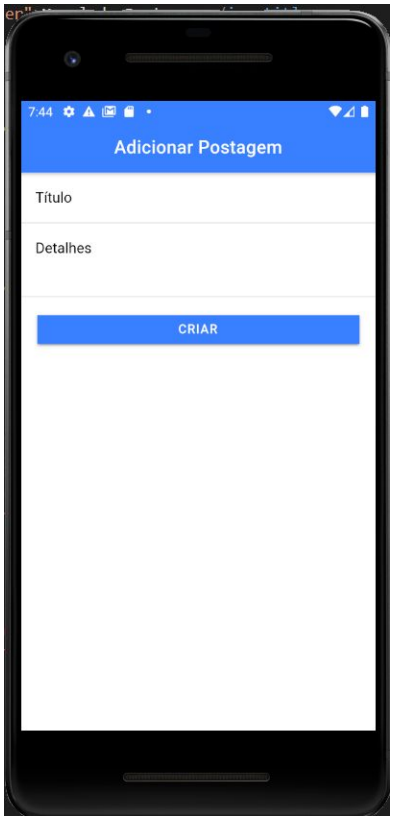
```
try {
  // login user with email and password
  await this.afAuth.auth
    .signInWithEmailAndPassword(user.email, user.password)
    .then(data => {
      console.log(data);

      // redirect to home page
      this.navCtrl.navigateRoot("home");
    })
    .catch();
} catch (e) {
  this.showToast(e);
}
```

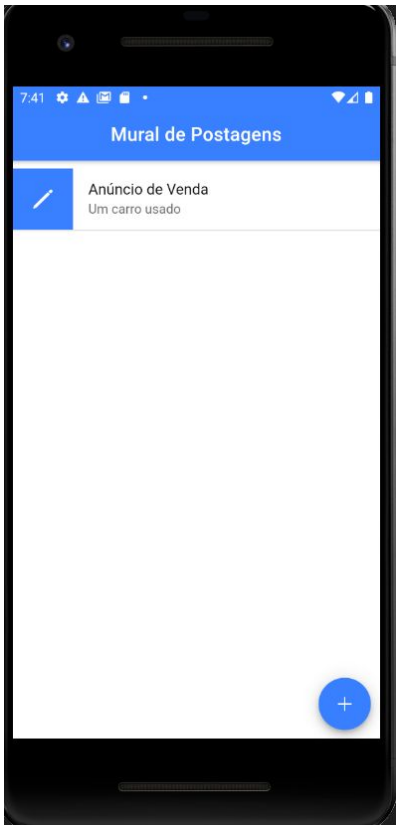
```
try {
  // register user with email and password
  await this.afAuth.auth
    .createUserWithEmailAndPassword(user.email, user.password)
    .then(data => {
      console.log(data);

      // redirect to home page
      this.navCtrl.navigateRoot("home");
    })
    .catch();
} catch (e) {
  this.showToast(e);
}
```

Após selecionar o botão de login, será redirecionado para a página principal para testar o CRUD com postagens. Para fazer a operação de update colocaremos um botão para editar quando deslizar para direita e outro para deletar na esquerda. E por fim um botão fixo na tela para adicionar outra postagem.

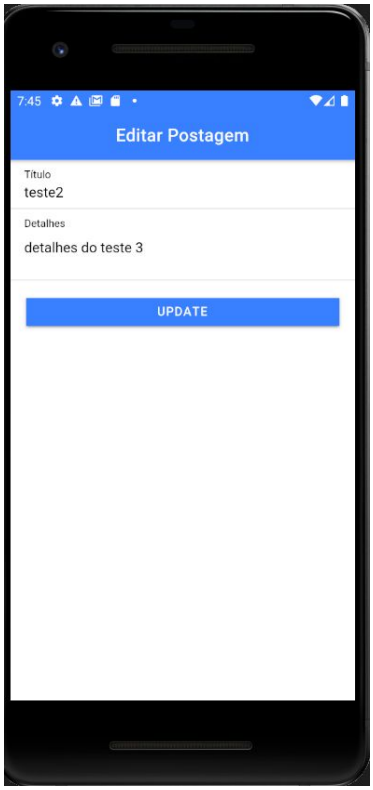
```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title class="ion-text-center">Mural de Postagens</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button [routerLink]="['/add-post']">
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>

  <ion-list lines="none">
    <ion-item-sliding *ngFor="let post of posts">
      <ion-item class="no-padding">
        <ion-label>
          <h2>{{ post.title }}</h2>
          <p>{{ post.details }}</p>
        </ion-label>
      </ion-item>

      <ion-item-options side="start">
        <ion-item-option [routerLink]="['/edit-post/', post.id]">
          <ion-icon slot="icon-only" name="pencil-outline"></ion-icon>
        </ion-item-option>
      </ion-item-options>

      <ion-item-options side="end">
        <ion-item-option color="danger" (click)="deletePost(post.id)">
          <ion-icon slot="icon-only" name="trash"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
</ion-content>

```

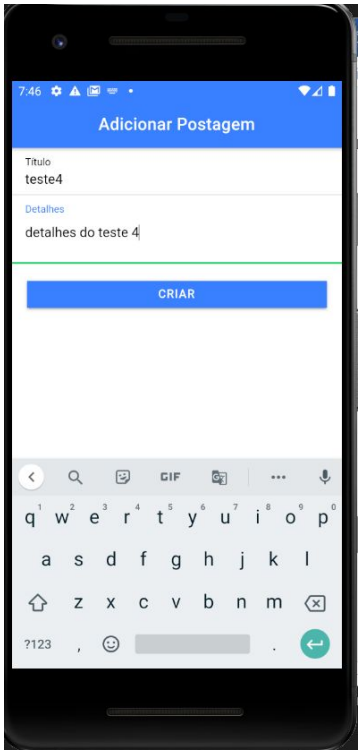
O botão de editar e adicionar irão redirecionar para uma página própria para fazerem suas próprias operações.


7:44
Adicionar Postagem
Título
Detalhes
CRIAR


7:41
Mural de Postagens
Anúncio de Venda
Um carro usado


7:41
Mural de Postagens
o de Venda
o usado


7:45
Editar Postagem
Título
teste2
Detalhes
detalhes do teste 3
UPDATE



O botão de Update e Adicionar ficaram com respectivamente:

```
try {
  await this.firestore.doc("posts/" + this.id).update(post);
} catch (e) {
  this.showToast(e);
}
```

```
try {
  await this.firestore.collection("posts").add(post);
} catch (e) {
  this.showToast(e);
}
```

Após testar, o banco de dados guardará os logins e as postagens com sucesso