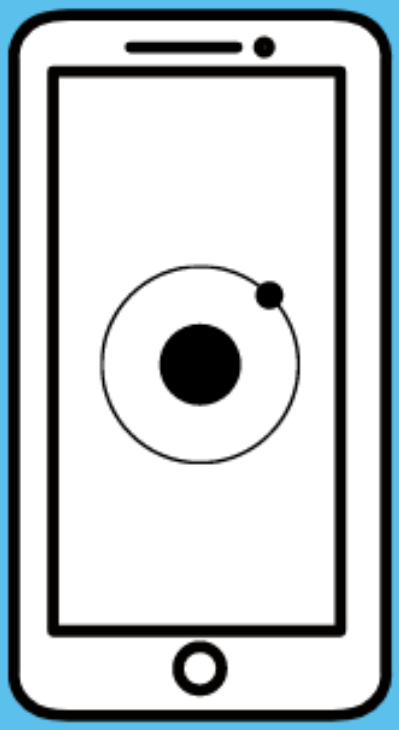
IONIC

DO HISTÓRICO À PRÁTICA EM UMA SIMPLES LEITURA



<DESENVOLVIDO POR>

ANA BEATRIZ F. DELFINO;
GUILHERME F. DA R. VIANNA;
JOÃO LUCAS F. MARQUES;
LARISSA DOS S. RUSSO;
MARIA LUISA C. SANTOS;
MEL J. LEMES;
NAUAN R. A. VIANA;
RAQUEL DA S. ARAÚJO.

</ DESENVOLVIDO POR>

CUBATÃO, 2021

SUMÁRIO

1. HISTÓRICO DO IONIC E SUAS PRINCIPAIS CARACTERÍSTICAS

O IONIC foi desenvolvido por Max Lynch, Ben Sperry e Adam Bradley da Dryfty Co. no ano de 2013. Ele possui o código-fonte de seu software aberto (open source) e sua tecnologia é utilizada para a construção de aplicativos híbridos — os quais funcionam em diferentes sistemas operacionais, como iOS, Android e Windows — para dispositivos móveis.

Além disso, o IONIC é um framework — um conjunto de bibliotecas utilizadas para criar uma base para aplicação — front-end, ou seja, é a parte de aplicação que interage com o usuário. Ele é, na verdade, um conjunto de outros componentes e frameworks. Seus componentes são:

- HTML: consiste em uma linguagem de marcação de documentos com hipertexto utilizada na construção de páginas da web;
- CSS: resume-se em um padrão de formatação para documentos HTML/XHTML. Com esse padrão, o CSS acaba por proporcionar maior controle sobre os atributos topográficos de um site;
- JavaScript e TypeScript: consistem em linguagens de programação que oferecem formas de tornar determinados processos de páginas web mais dinâmicos;
- Angular: é uma plataforma de aplicações web de código aberto, baseado em JavaScript e TypeScript, que facilita o desenvolvimento e os testes dos aplicativos;
- Node: consiste em uma plataforma usada para construir aplicações web escaláveis de alta performance usando JavaScript, que otimiza todo o processo de construção do aplicativo;
- Cordova: é uma estrutura livre e de código aberto para a construção de aplicativos nativos de plataforma usando HTML, CSS e JavaScript. Possibilita, desse modo, que um aplicativo híbrido possa ter acesso aos recursos do celular, mesmo que de forma mais limitada.

Utilizar-se do framework IONIC para a construção de aplicações híbridas traz diversas vantagens para o programador, dentre elas são:

- Aplicação híbrida gera ganho de tempo e velocidade, uma vez que o aplicativo opera em diferentes sistemas operacionais utilizando o mesmo código;
- os aplicativos híbridos são capazes de acessar a maioria dos recursos nativos do dispositivo (como a câmera e o giroscópio) utilizando *plugins* nativos;
- é desenvolvido em linguagens já muito utilizadas por programadores — comunidade de desenvolvedores é mais ampla;
- é uma alternativa menos custosa, ou seja, mais barata.

Porém, infelizmente também pode apresentar alguns tipos de desvantagens, tais como:

- Embora a aplicação híbrida ofereça velocidade, uma aplicação nativa a ultrapassa;
- nem todos os recursos dos dispositivos permitem 100% da utilização pelo IONIC, fazendo-se necessário maior desempenho dos recursos nativos do hardware;
- apresenta problemas de performance em alguns ambientes, principalmente em Androids mais antigos.

É importante saber avaliar quando o IONIC é a melhor alternativa para o desenvolvimento da aplicação, pois fazendo a avaliação correta, pode-se obter mais vantagens, extraindo assim o que há de melhor do framework, o que consequentemente facilita a construção do aplicativo desejado.

2. IDE's POSSÍVEIS E IDE ESCOLHIDA

IDE, Integrated Development Environment (Ambiente de Desenvolvimento Integrado), é um software que tem a função de, basicamente, ajudar os usuários a programar novas aplicações de forma mais rápida, combinando ferramentas comuns de desenvolvimento em uma única interface gráfica (GUI).

Os IDEs mais populares do framework Ionic, de acordo com o site oficial, são: O Visual Studio Code, Atom, WebStorm, ALM e o Angular IDE by Webclipse. O grupo optou por utilizar o Visual Studio Code por ser o mais recomendado pelos usuários e estarmos mais familiarizados com este IDE, pois daria mais trabalho utilizar um IDE nunca utilizado antes.

A principal característica do VSCode, IDE escolhido pelo grupo, é ser open source, ou seja, permite à comunidade técnica contribuir com seu desenvolvimento e facilitando a criação de extensões e novas funcionalidades. Além disso, há o IntelliSense, um recurso que fornece conclusões inteligentes diretamente no editor e, às vezes, dá exemplos de pequenos códigos para auxiliar o usuário. Outra característica importante é o código de depuração direto do editor, que ajuda a navegar pelo código para inspecionar o estado do aplicativo e mostrar seu fluxo de execução.

Portanto, o IDE da Microsoft, Visual Studio Code, mostra-se uma ferramenta com potencial e facilidades de uso para ajudar usuários a programar em novos frameworks.

3. PASSO A PASSO PARA INSTALAÇÃO DA IDE

3.1. Passo 1

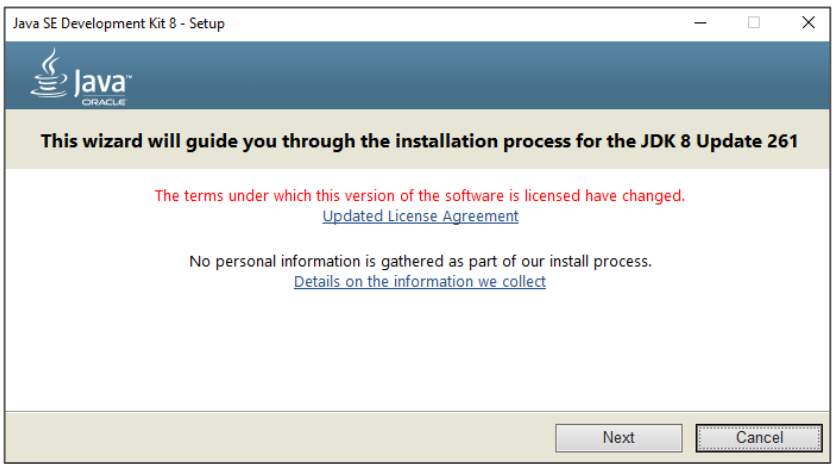
Instalar o JDK, a versão mais atualizada é a 8, até o momento. O download pode ser feito pelo seguinte link: https://www.oracle.com/br/java/technologies/javase/javase-jdk8-downloads.html

Você deve selecionar o seu software para fazer o download da aplicação correta.

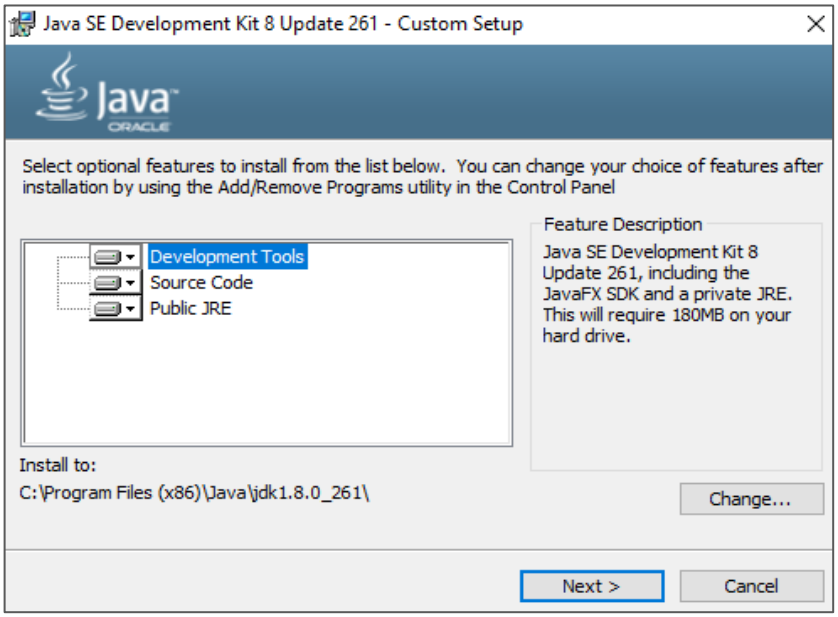
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	73.4 MB	jdk-8u261-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	70.3 MB	jdk-8u261-linux-arm64-vfp-hflt.tar.gz
Linux x86 RPM Package	121.92 MB	jdk-8u261-linux-i586.rpm
Linux x86 Compressed Archive	136.81 MB	jdk-8u261-linux-i586.tar.gz
Linux x64 RPM Package	121.53 MB	jdk-8u261-linux-x64.rpm
Linux x64 Compressed Archive	136.48 MB	jdk-8u261-linux-x64.tar.gz
macOS x64	203.94 MB	jdk-8u261-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.77 MB	jdk-8u261-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.72 MB	jdk-8u261-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.23 MB	jdk-8u261-solaris-x64.tar.Z
Solaris x64	92.47 MB	jdk-8u261-solaris-x64.tar.gz
Windows x86	154.52 MB	jdk-8u261-windows-i586.exe
Windows x64	166.28 MB	jdk-8u261-windows-x64.exe

Será necessário criar um login no Oracle, mas isso durará apenas 5 minutos.

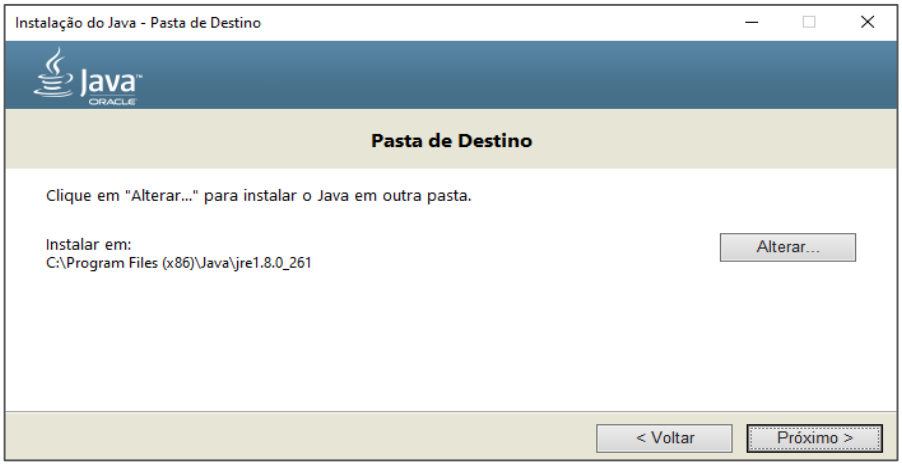
Após o término do download, execute o arquivo e aparecerá a seguinte imagem, clique em “Next”:



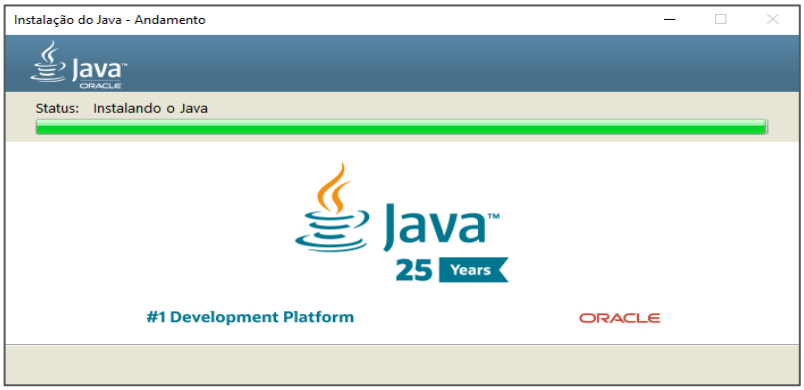
Aparecerá a seguinte tela:
Clique em “Next” mais uma vez para seguir em frente.



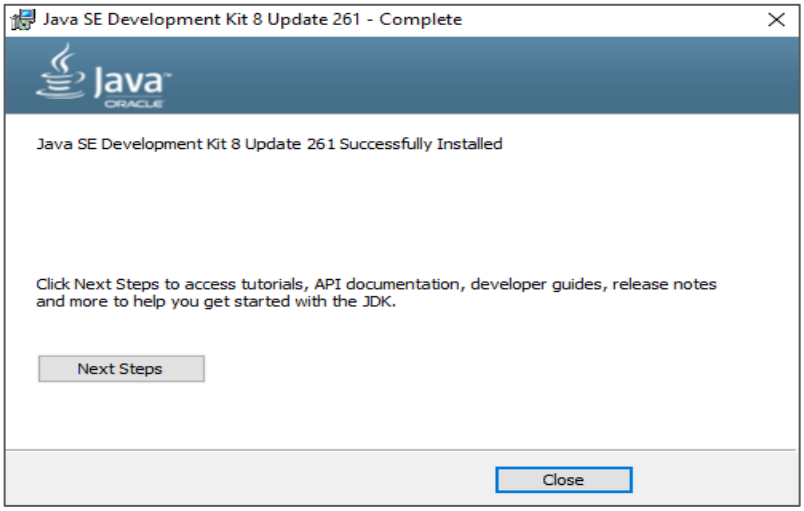
Clique em “próximo” para chegar ao processo final de instalação.



O Java estará sendo instalado:

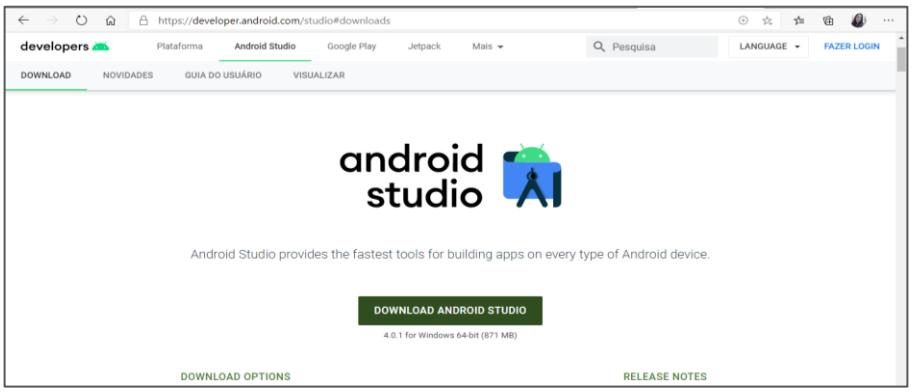


Pronto, o JDK está instalado em sua máquina.



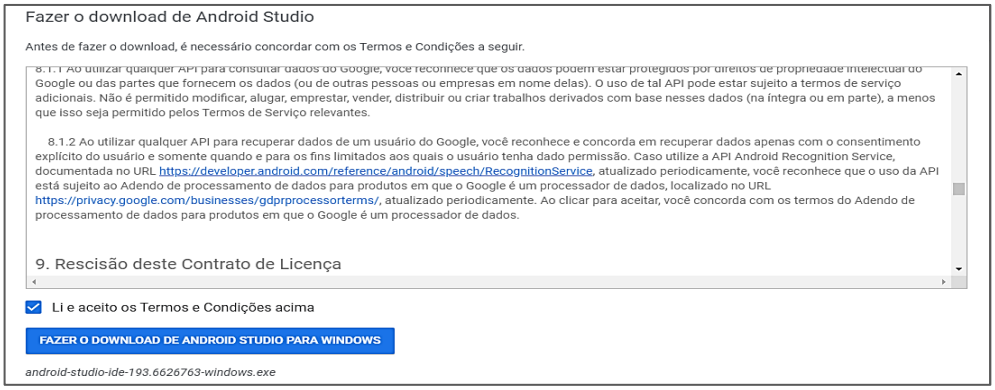
3.2. Passo 2

Download do Android SDK pelo link: https://developer.android.com/studio#downloads

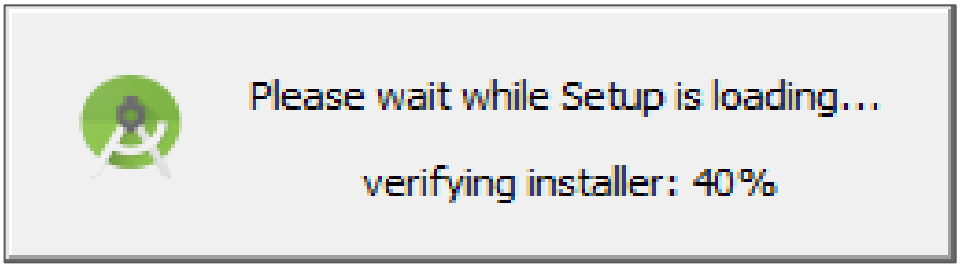


Antes de clicar para realizar o download certifique-se que este é o correto para o software de seu computador.

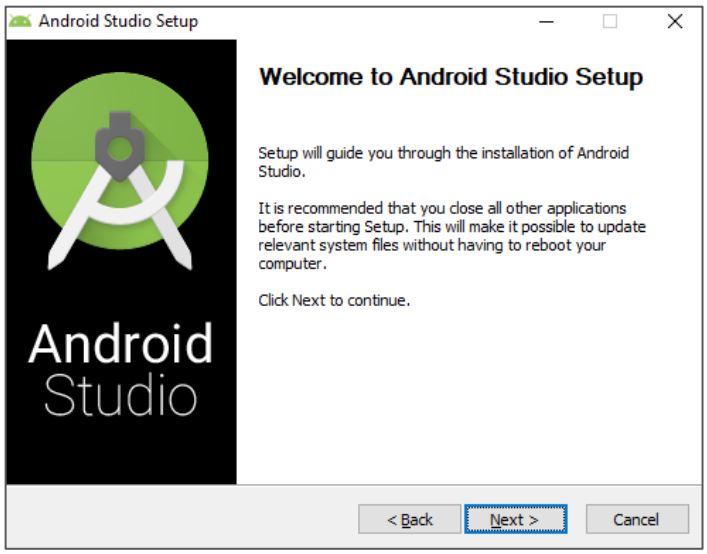
Ao clicar para realizar o download será necessário que aceite os termos e condições.



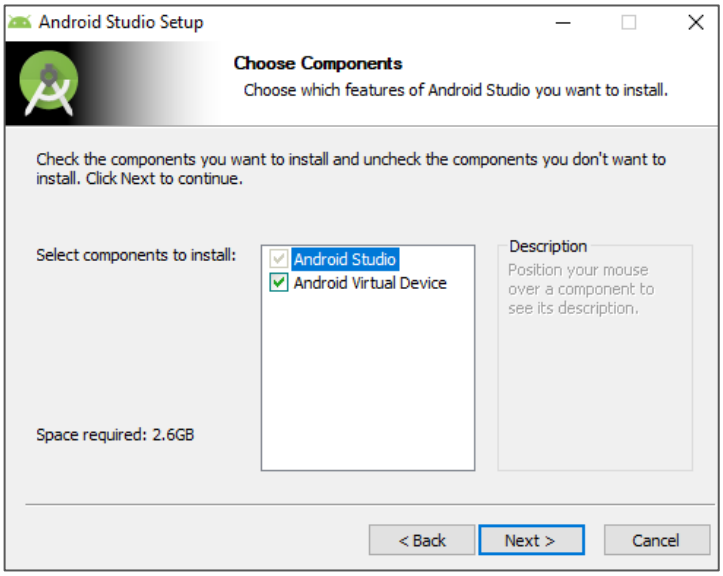
Após aceitar os termos, execute o arquivo baixado.



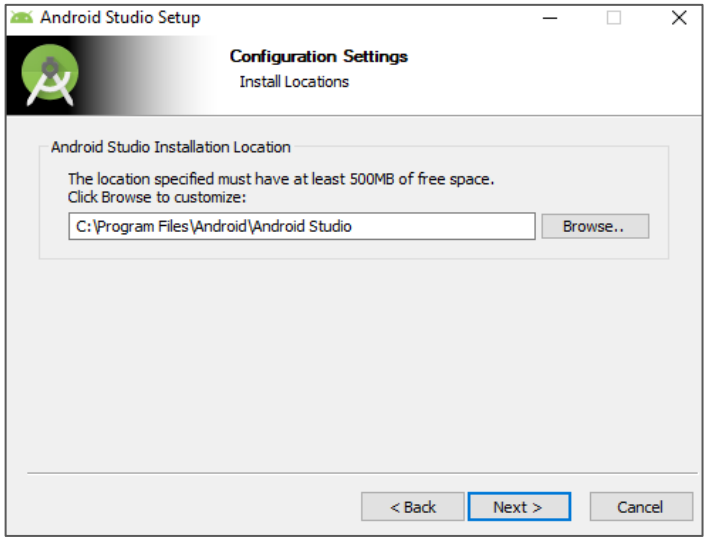
Em seguida, clique em “Next”.



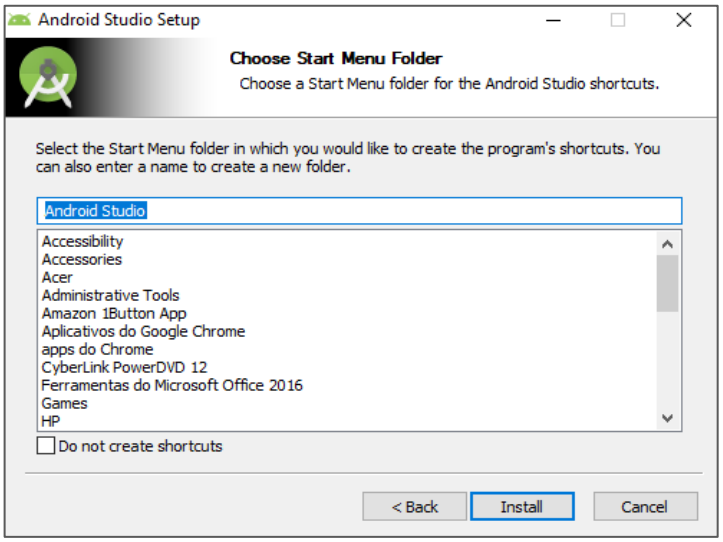
Clique em “Next” mais uma vez.



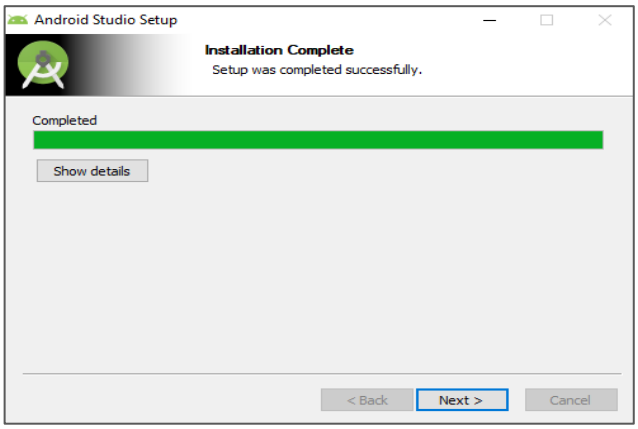
Mais uma vez prossiga clicando em “Next”



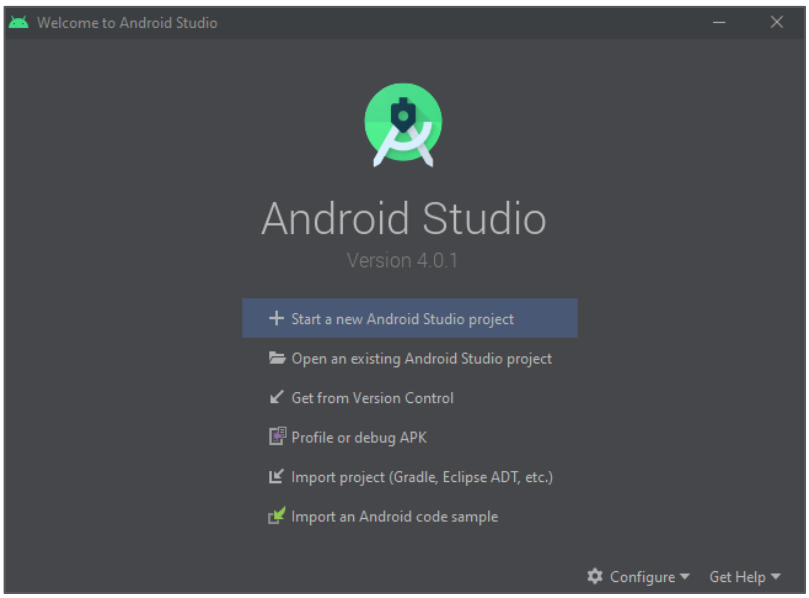
Por último, clique em “Install”



Ao final do download aperte “Next” para abrir a aplicação do Android Studio.

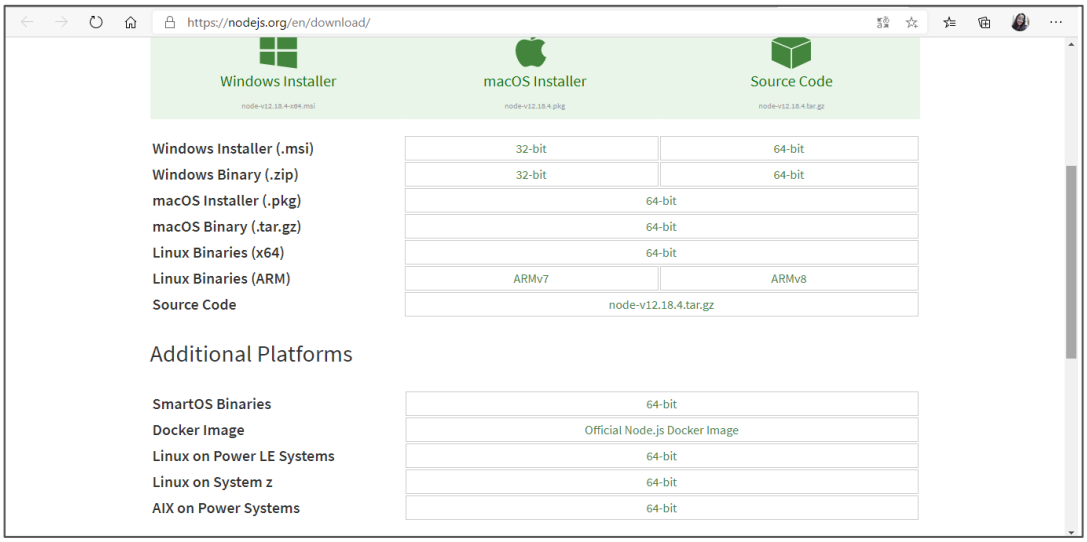


Essa é a tela principal do Android Studio, mas não o usaremos por enquanto.

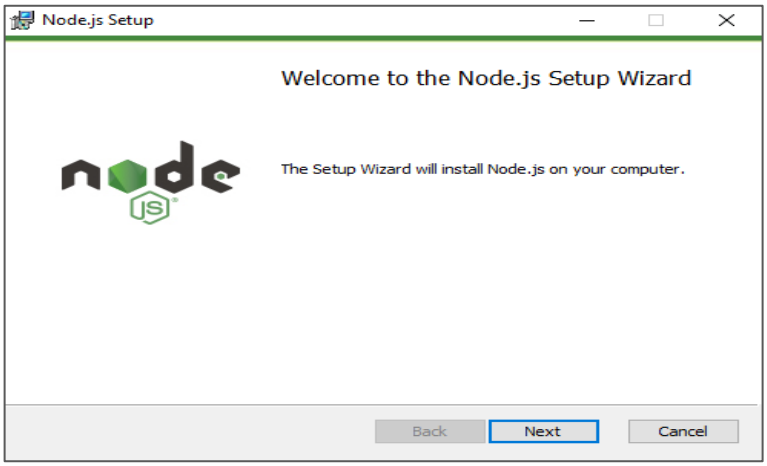


3.3. Passo 3

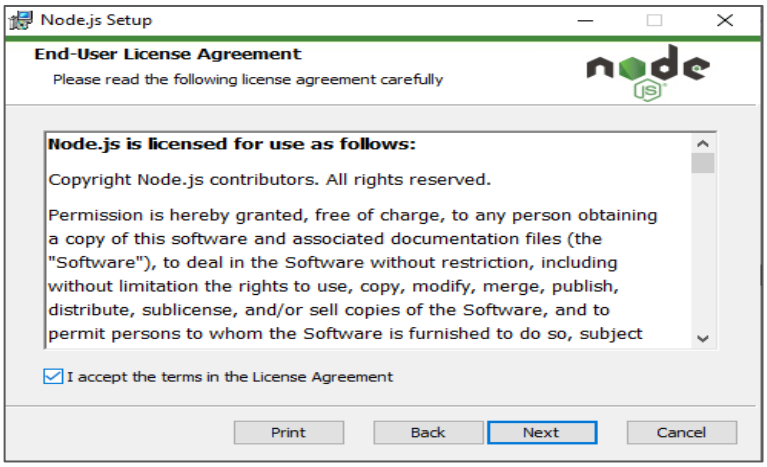
Download do NodeJs, que pode ser encontrado no seguinte link: https://nodejs.org/en/download/
Selecione seu software corretamente e faça o download.



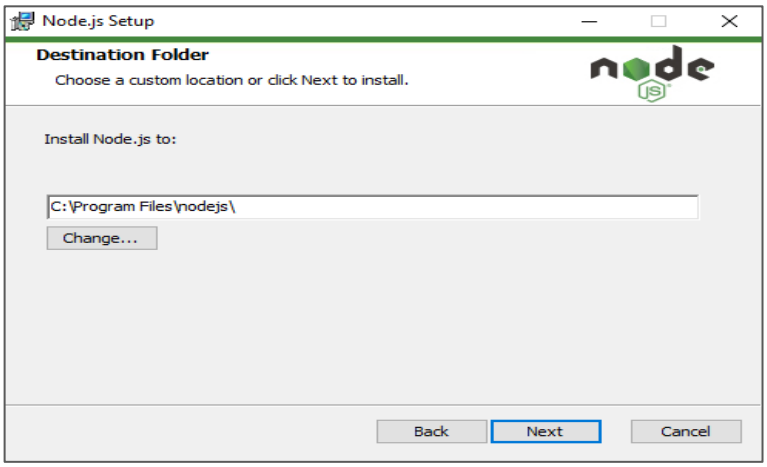
Após o download, execute o arquivo e logo em seguida, clique em “Next”.



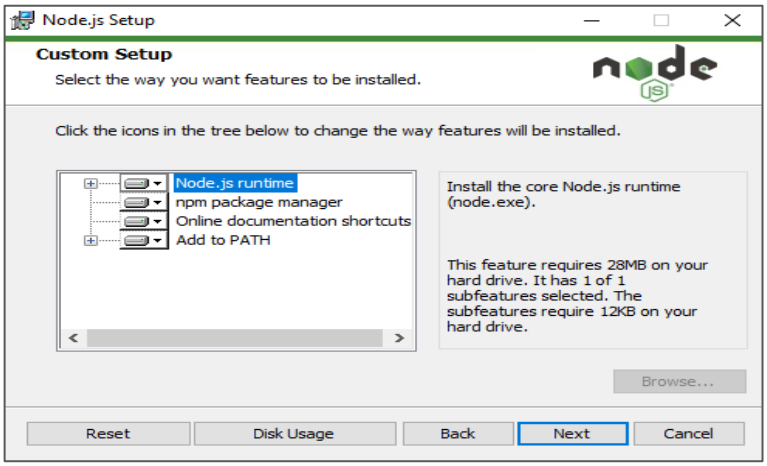
Aceite os termos e clique em “Next”.



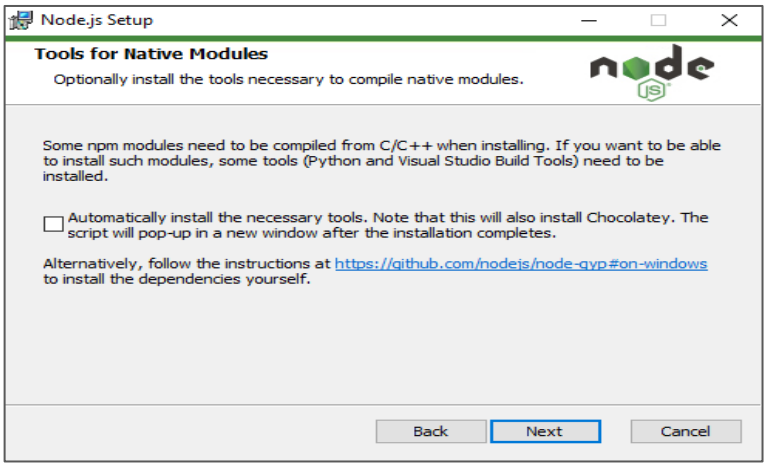
Clique em “Next” mais uma vez.



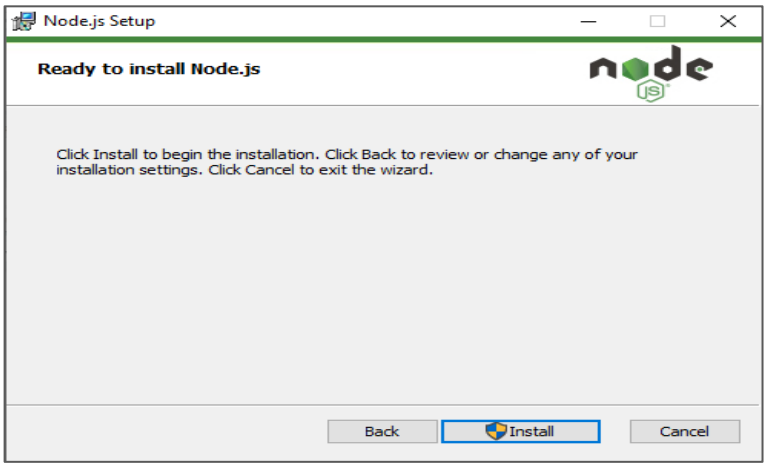
Mais uma vez em “Next”.



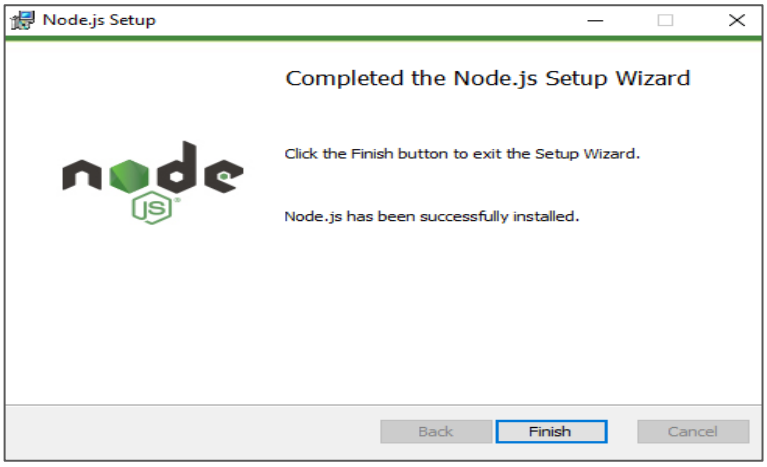
"Next" de novo



E, finalmente, clique em "Install"

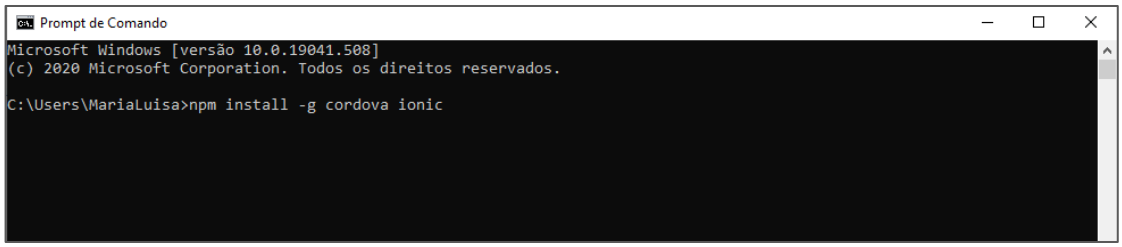


O Node.Js estará instalado em seu computador, clique em "Finish".



3.4. Passo 4

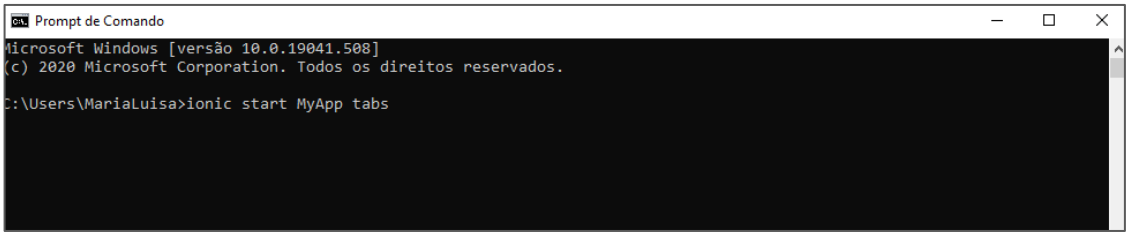
O próximo passo é instalar o Ionic e o Cordova em seu computador. Para isso, abra o seu Prompt de Comando e digite o seguinte comando: *npm install -g cordova ionic*



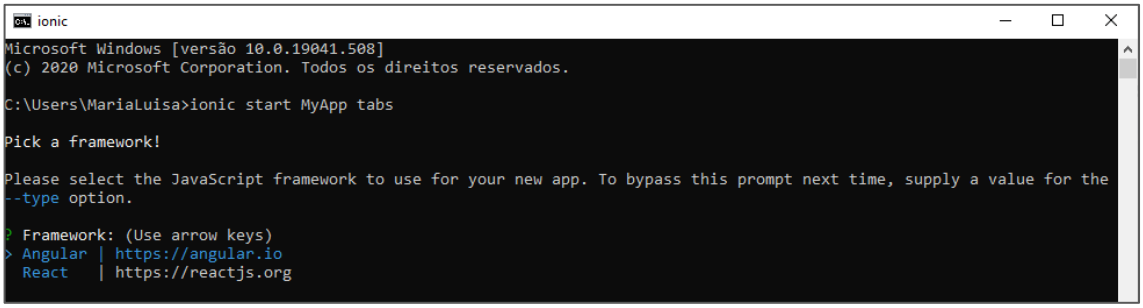
Ao final da instalação, você pode digitar *ionic* no próprio CMD para ver alguns comandos que podem ser utilizados com ele.

3.5. Passo 5

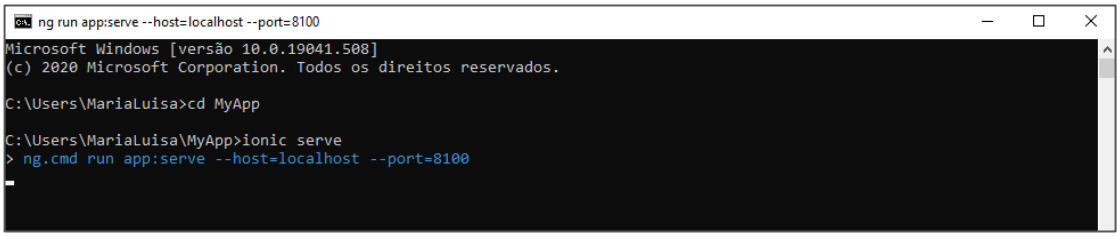
Criar um projeto ionic. Esta parte é simples. Você precisará utilizar o comando *ionic start [nome_da_aplicacao] [tipo_de_menu]*. Para o tipo de menu você pode escolher entre, sidemenu, tabs e blank.

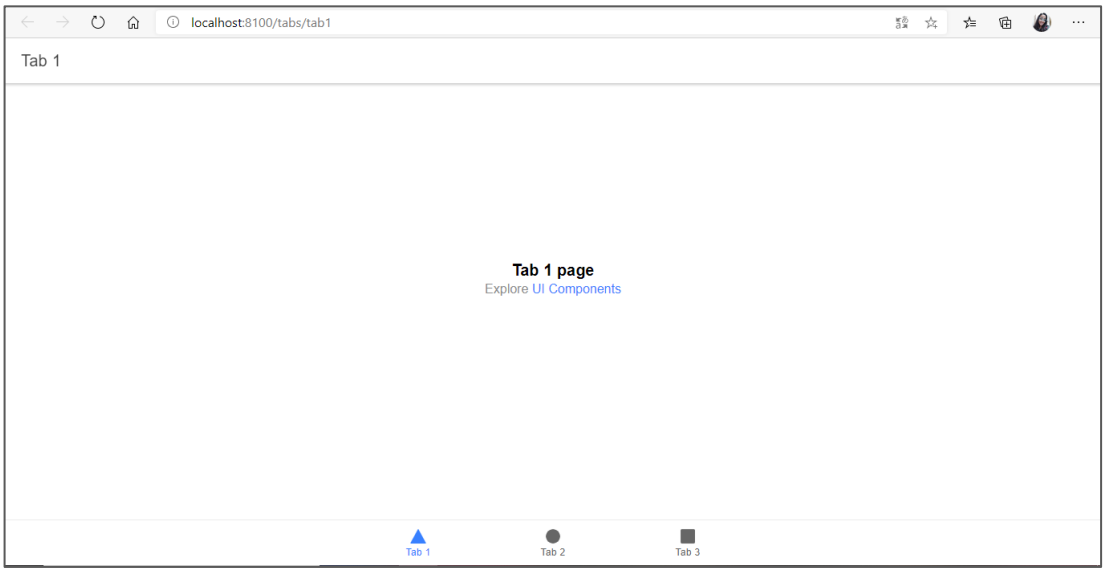


Após esse comando, será necessário escolher entre o Ionic Angular ou React, neste caso, escolhemos o Angular.



Pronto, seu projeto está criado, para visualizá-lo, vá para a pasta do projeto *cd [nome_do_projeto]* e realize o comando *ionic serve*. Este comando abrirá seu projeto em seu navegador padrão.

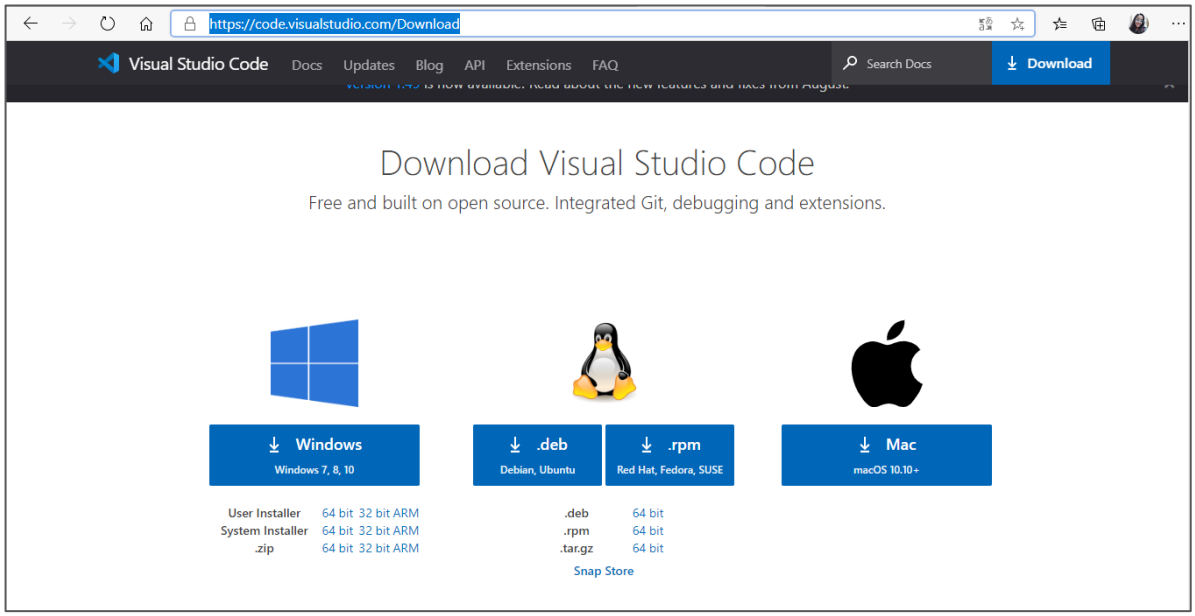




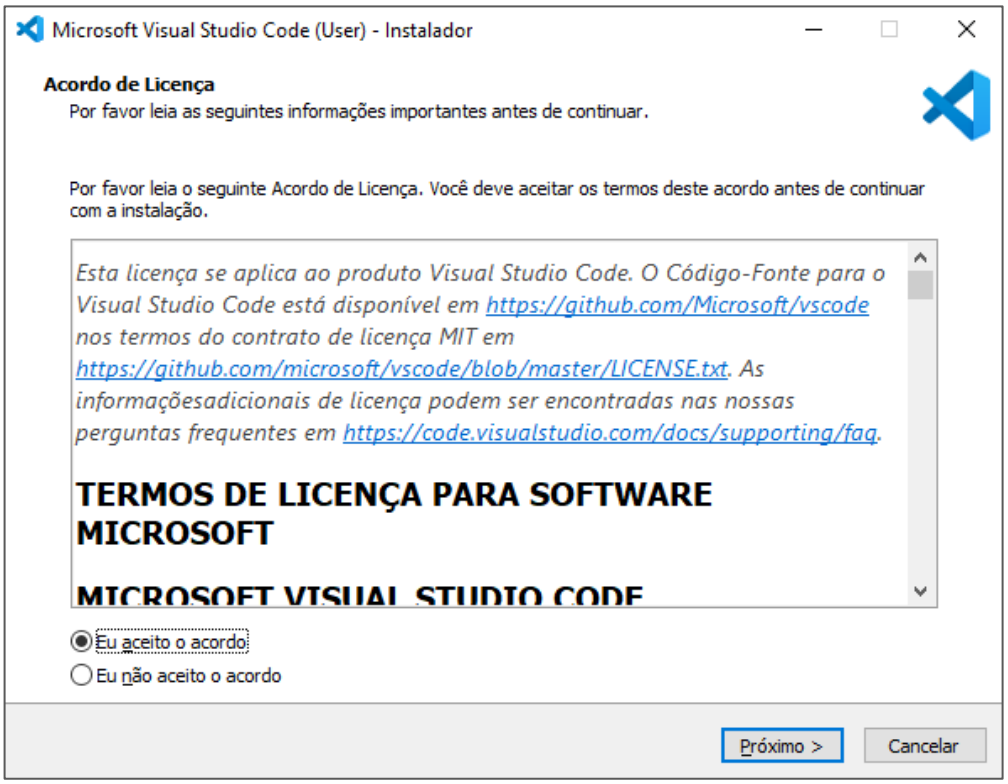
3.6. Passo 6

Para editar seu projeto será necessário baixar uma IDE. No nosso caso escolhemos o Visual Studio Code. Você pode utilizar o seguinte link: https://code.visualstudio.com/Download.

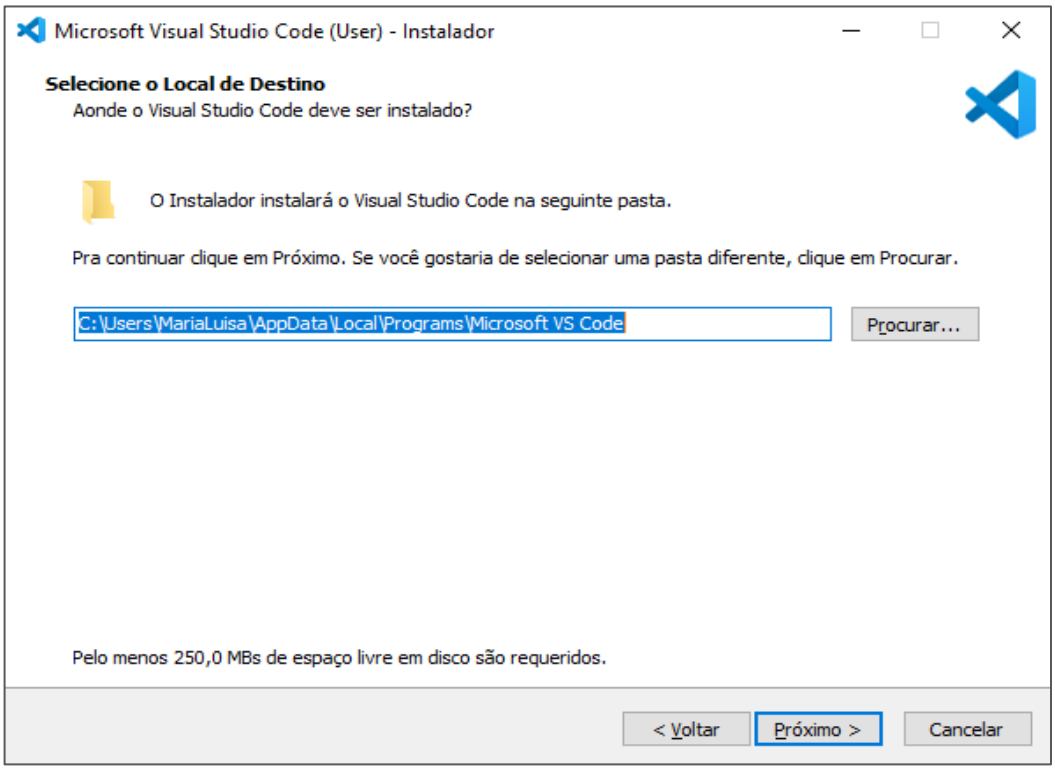
Escolha corretamente o software da sua máquina.



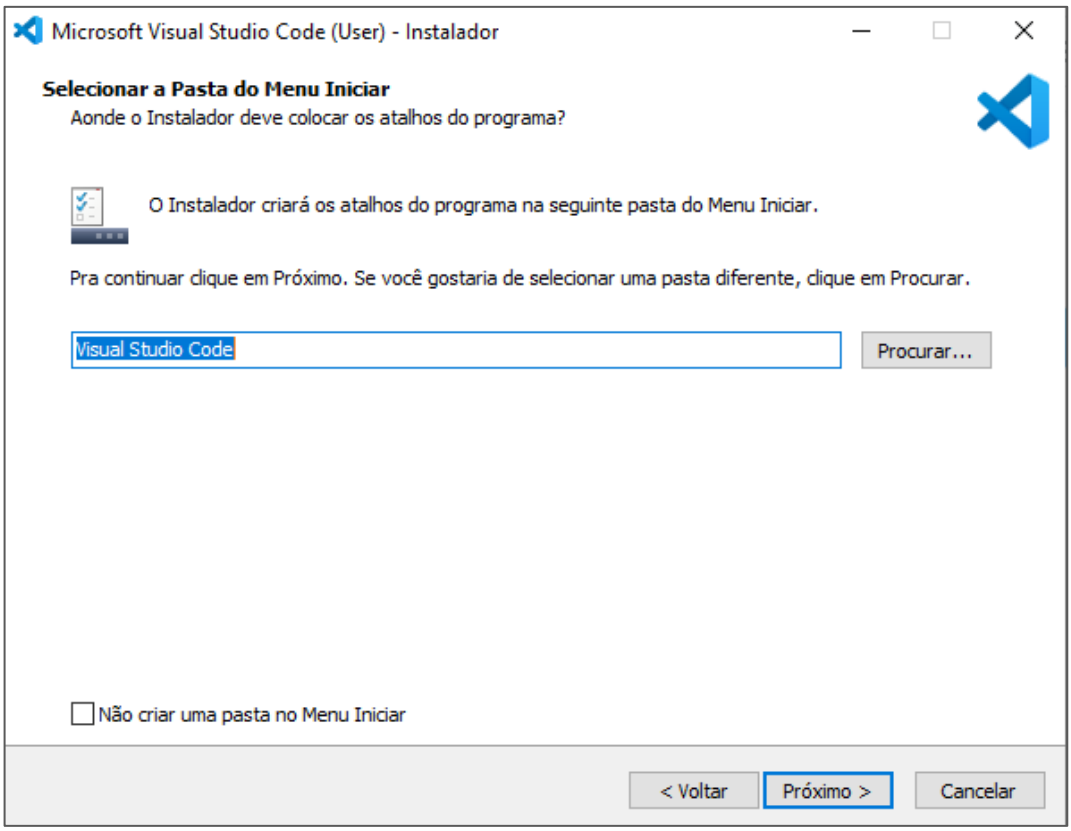
Após terminar o download, execute o arquivo e aceite os termos e condições. Ao aceitar, clique em “Próximo”.



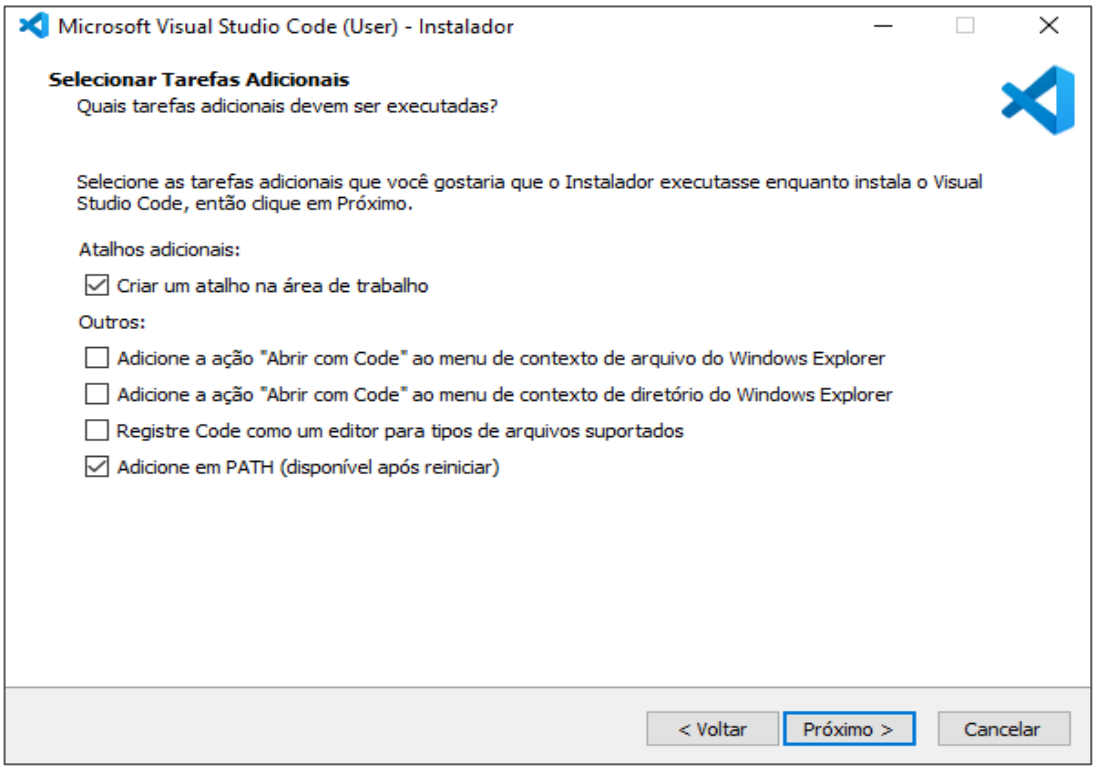
Clique em “Próximo”



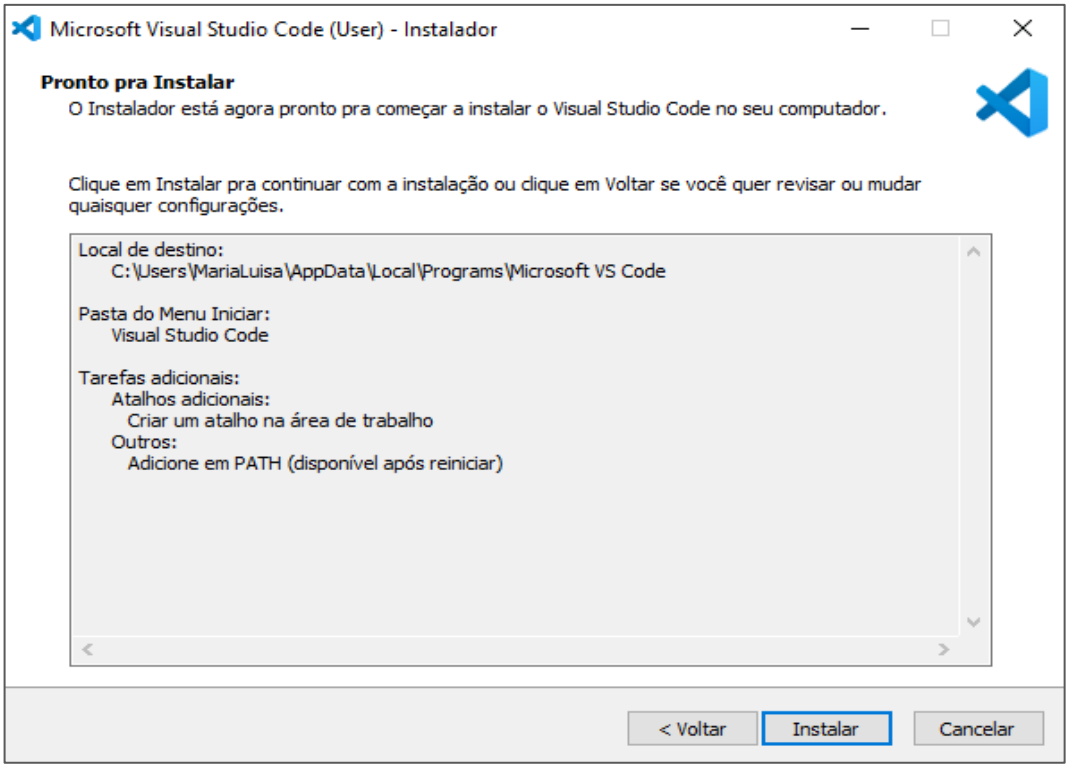
Clique em “Próximo” mais uma vez.



Mais uma vez em “Próximo”



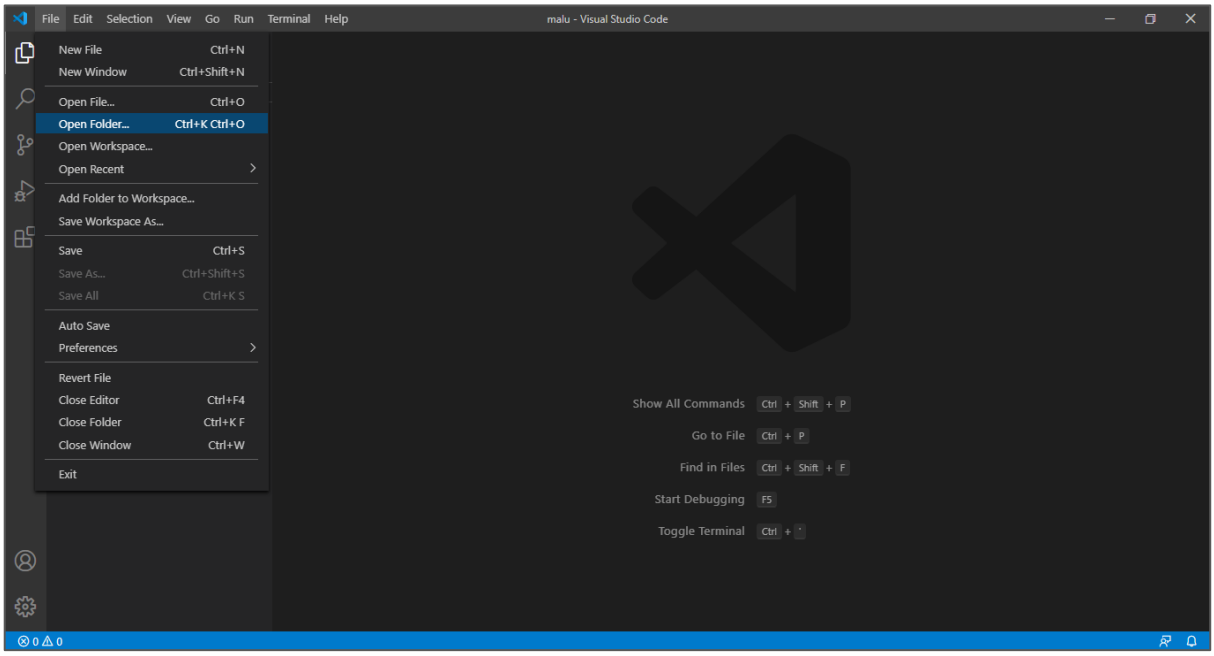
E, por último, clique em “Instalar”.



Ao final da instalação, o Visual Studio Code será aberto.

3.7. Passo 7

Abra o Visual Studio Code. Vá até “File” e clique em “Open Folder”. Procure o seu projeto para abri-lo no Visual Studio.

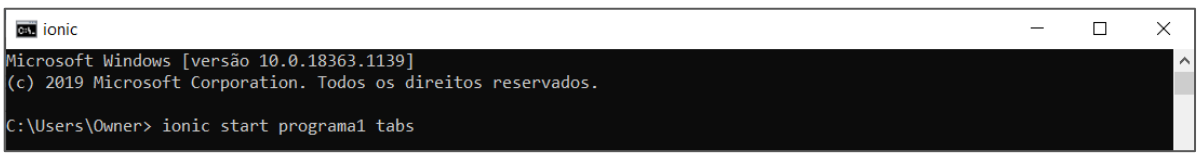


Todas as pastas e arquivos do seu projeto serão abertos no Visual Studio Code. Mais adiante ensinaremos como editar o seu projeto.

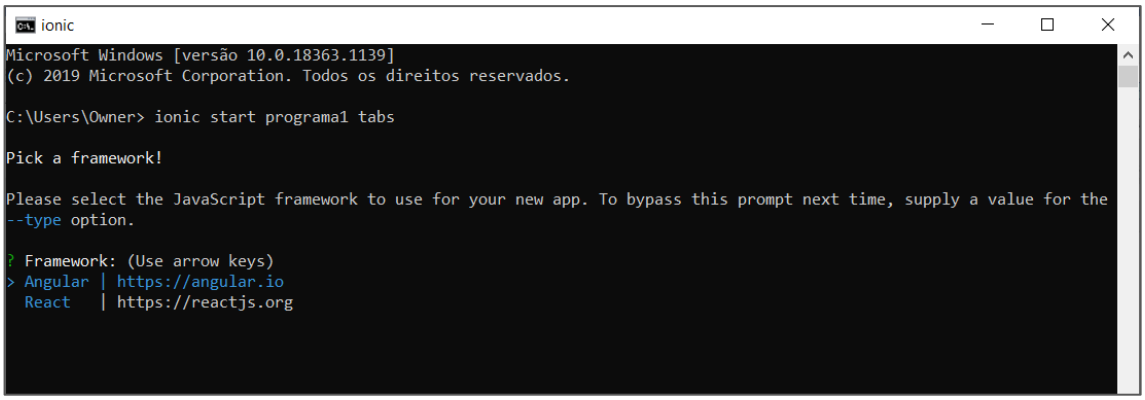
4. ÁREA DO TRIÂNGULO

4.1. Passo 1

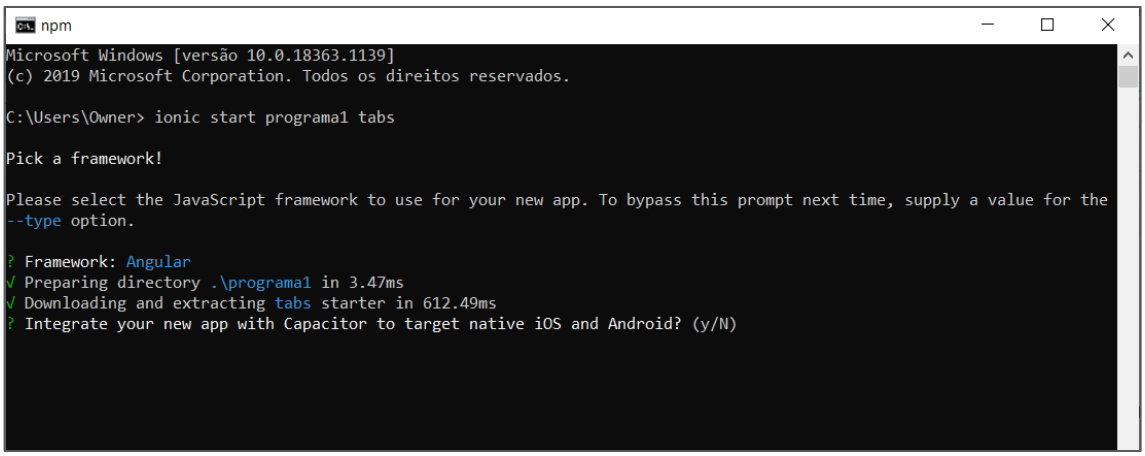
Crie um projeto ionic. No cmd, você precisará utilizar o comando ionic start [nome_da_aplicacao] [tipo_de_menu], tipos de menu podem ser e, sidemenu, tabs e blank, utilizaremos o tabs.



O cmd pedirá para escolher entre o angular e o react, escolheremos o angular.



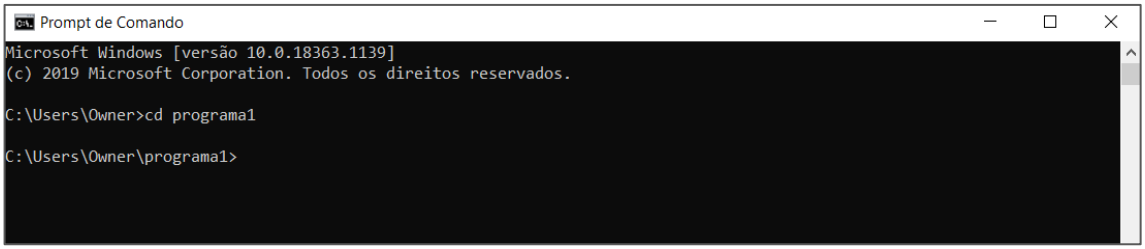
O cmd vai perguntar se quer integrar o app em iOS e Android. Responda sim (y).



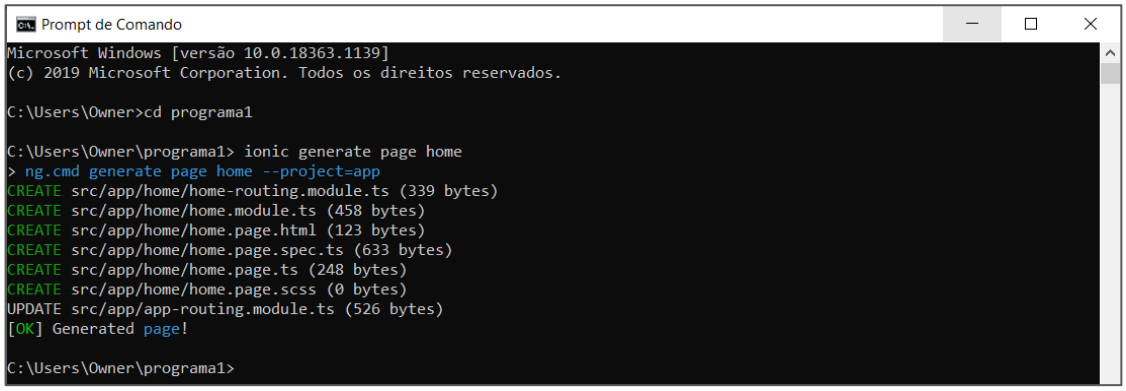
Pronto, o Projeto já existe!

Precisamos criar também uma página home, onde estará a interface do aplicativo, portanto:

Ainda no cmd, entre na pasta do nosso projeto (cd programa1).



Digite: ionic generate page home

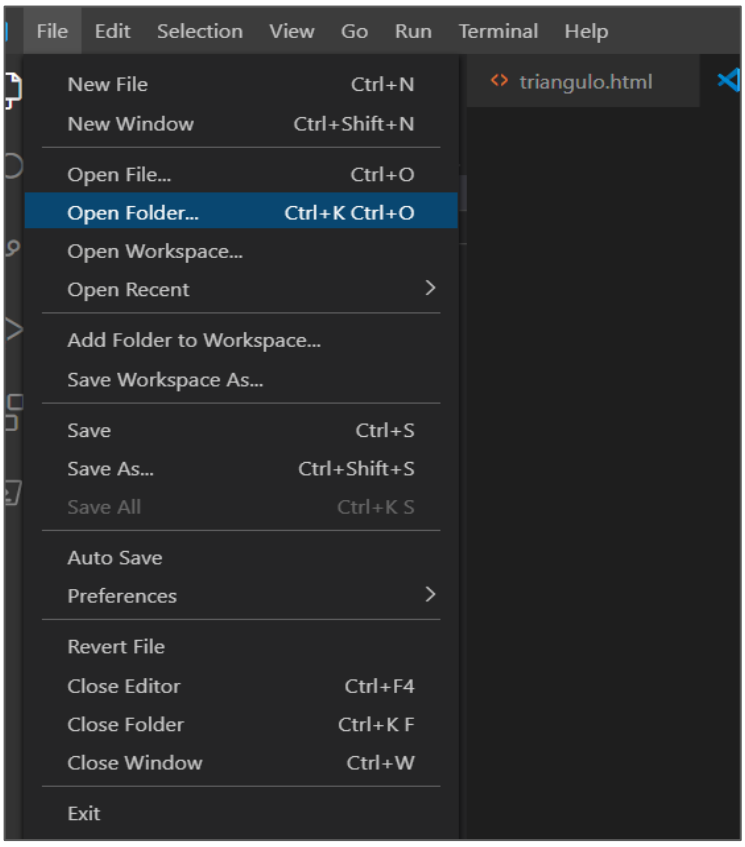


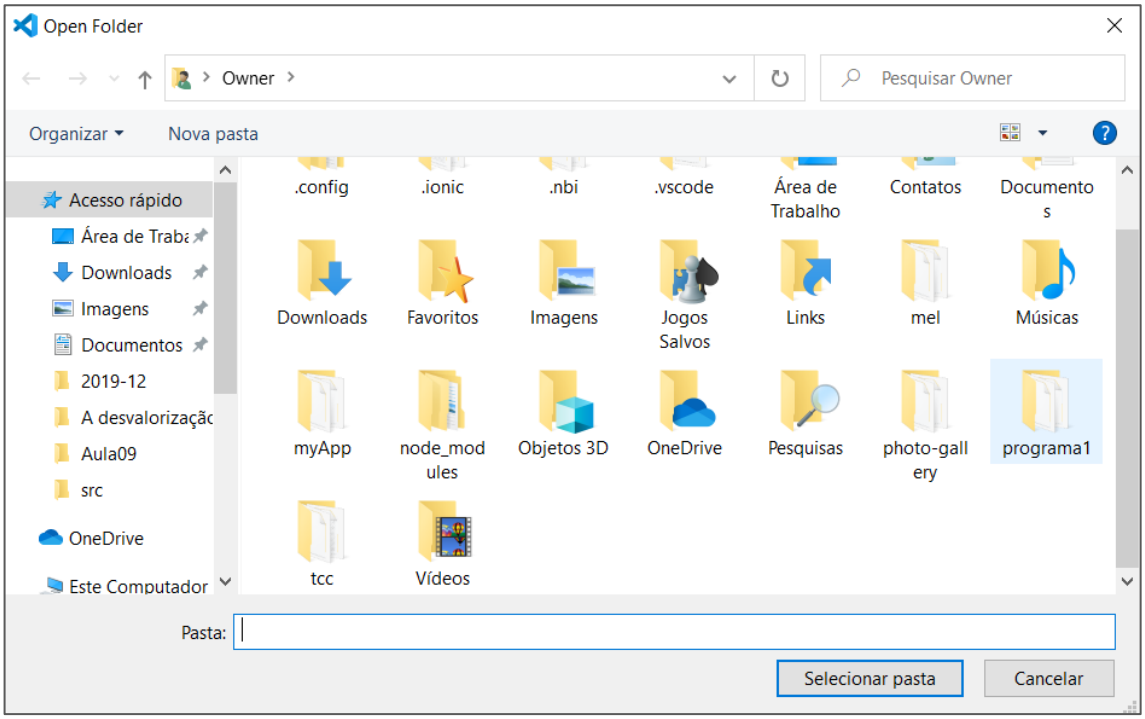
Pronto, a pasta está criada.

Agora, vamos abri-lo no Visual Studio.

4.2. Passo 2

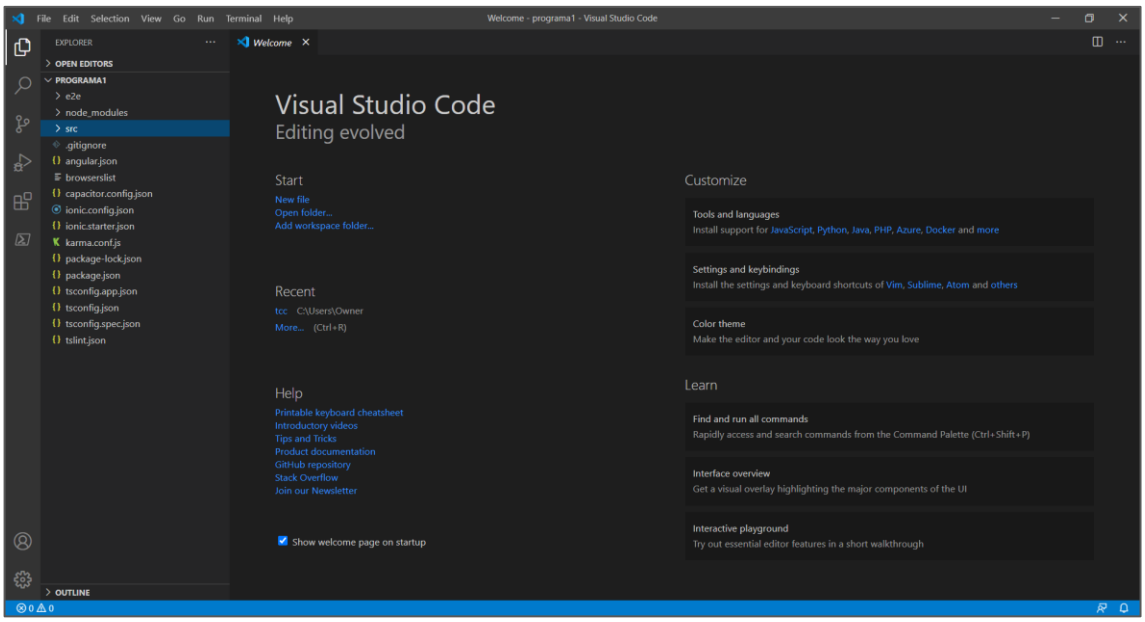
Abra o projeto no Visual Studio Code. Vá em file, no canto superior esquerdo da tela e clique em open folder.



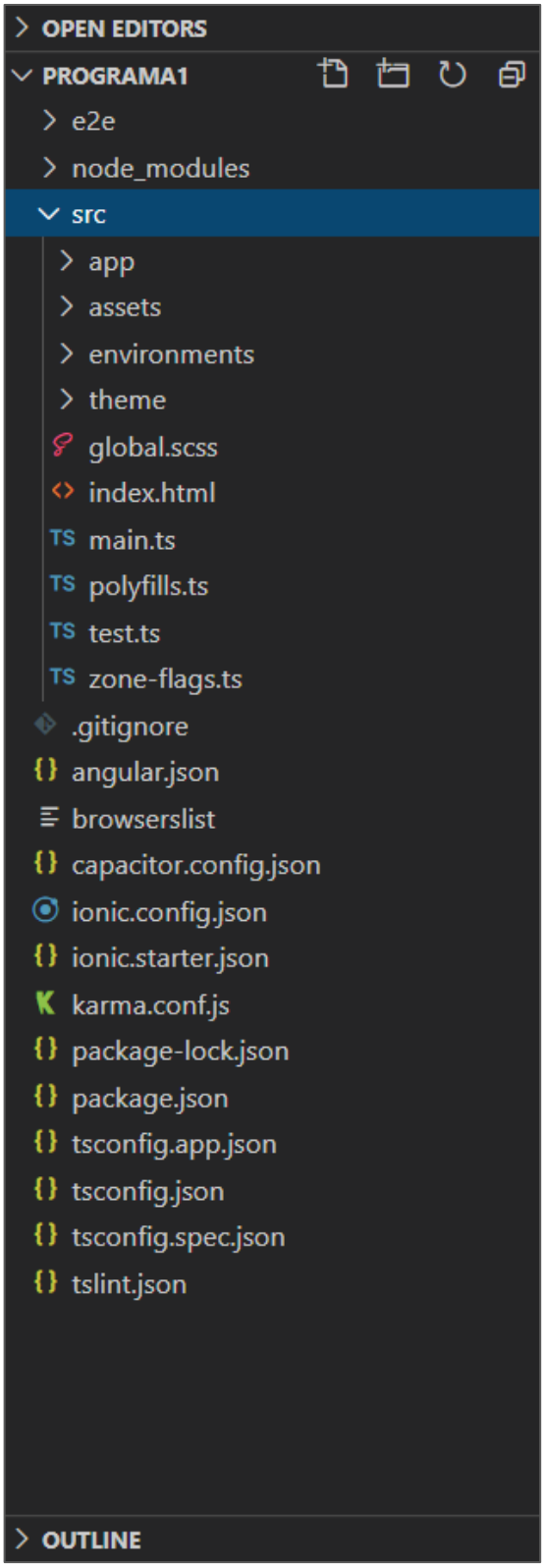


Clique na pasta e depois em “Selecionar pasta”.

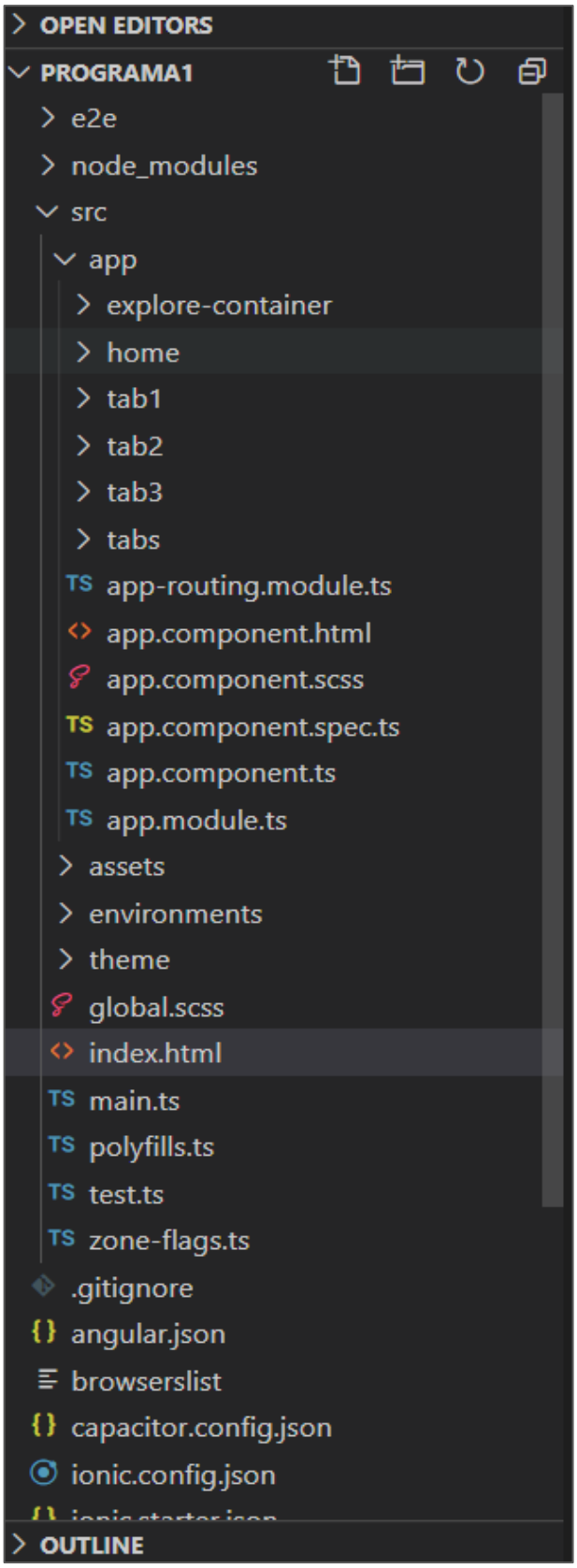
Essa é a página que vai aparecer:



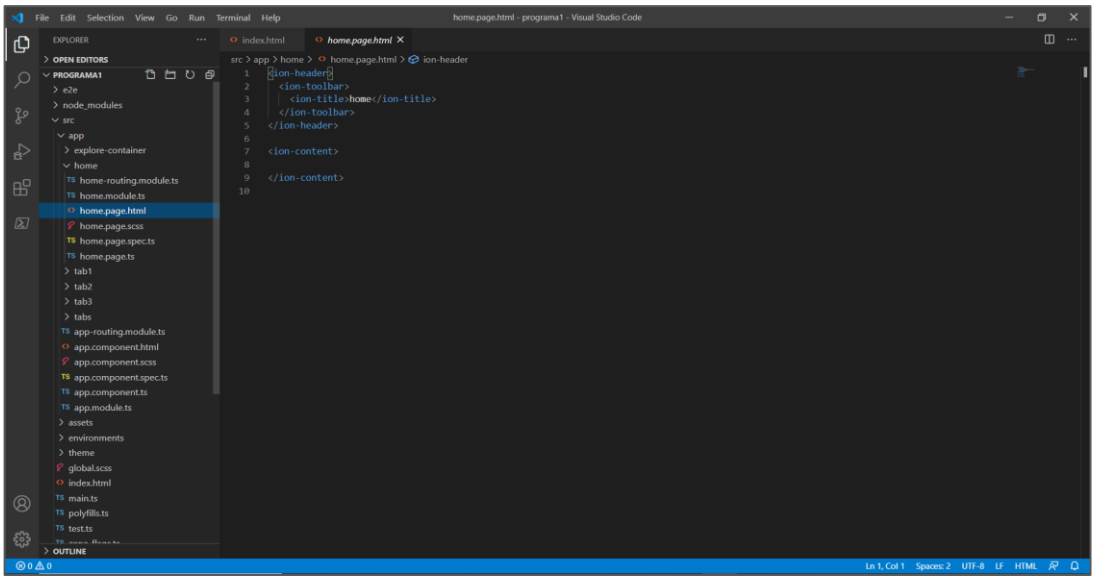
Clique em “scr”.



Vá em “app”.


OPEN EDITORS
PROGRAMA1
e2e
node_modules
src
app
explore-container
home
tab1
tab2
tab3
tabs
app-routing.module.ts
app.component.html
app.component.scss
app.component.spec.ts
app.component.ts
app.module.ts
assets
environments
theme
global.scss
index.html
main.ts
polyfills.ts
test.ts
zone-flags.ts
.gitignore
angular.json
browserslist
capacitor.config.json
ionic.config.json
OUTLINE

Abra “home” e vá em home.page.html



Essa é a página que aparecerá, delete todo o conteúdo dela;

Agora, criaremos a interface.

4.3. Passo 3

Crie a interface. A proposta é criar um programa que, quando inseridos os valores de base e altura, calcule a base do triângulo, portanto precisaremos dos seguintes componentes:

- Caixa de texto para digitação da base;
- caixa de texto para digitação da altura;
- botão calcular;
- espaço para exibição do resultado.

Portanto, começaremos com

```
<label> Base: </label>

<p> <input id="txt1" type="number"/> </p>

<p> <label>Altura: </label> </p>

<input id="txt2" type="number"/>

<label> </label>
```

indica ao usuário qual informação deve ser inserida na caixa de texto a seguir, o elemento label é o comando para inserir uma caixa de texto **não editável**.

```
<p>  <input id=" " type="number"/> </p>
```

Onde o usuário irá inserir o valor da base. Nesta linha temos os seguintes comandos:

- * * : para isolar um parágrafo, neste caso, a caixa de texto;
- *<input id=" " type="number"/>* : *input* é a caixa de texto **editável** do html, *id* é a nossa variável, que nomeamos de txt1 e txt2 e o *type* indica o que será inserido, neste caso, colocamos "number" para travar a digitação de qualquer caractere não numérico.

Para o botão, faremos:

```
<p>  <button id="calcular" onclick="calcularArea()">Calcular área</button> </p>
```

Essa linha indica a criação do botão "Calcular área", temos:

- *button:* indica a criação de um botão;
- *id:* para o nome dele utilizado para programar, no caso é "calcular"
- *onclick:* indica a função do botão,

OBS:. neste caso é "calcularArea()" que programaremos em seguida.

O nome que irá parecer para o usuário (Calcular área) será inserido no meio dos comandos que abrem e fecham o botão (<button ...> </button>), entre <button id="calcular" onclick="calcularArea()">**Calcular área**</button>

Para finalizar, criaremos o espaço para a exibição do resultado:

```
<label> Resultado: </label>
 <div id="resultado"></div>
```

Assim:

- *<label> </label>:* guiará o usuário para informação que virá a seguir;
- * :* será onde aparecerá o resultado; o comando *div* organiza o espaço, deixa ele exclusivo para a exibição do resultado que aparecera na variável "resultado", que o *id* indica.

Para questões de estética e organização, acrescentaremos:

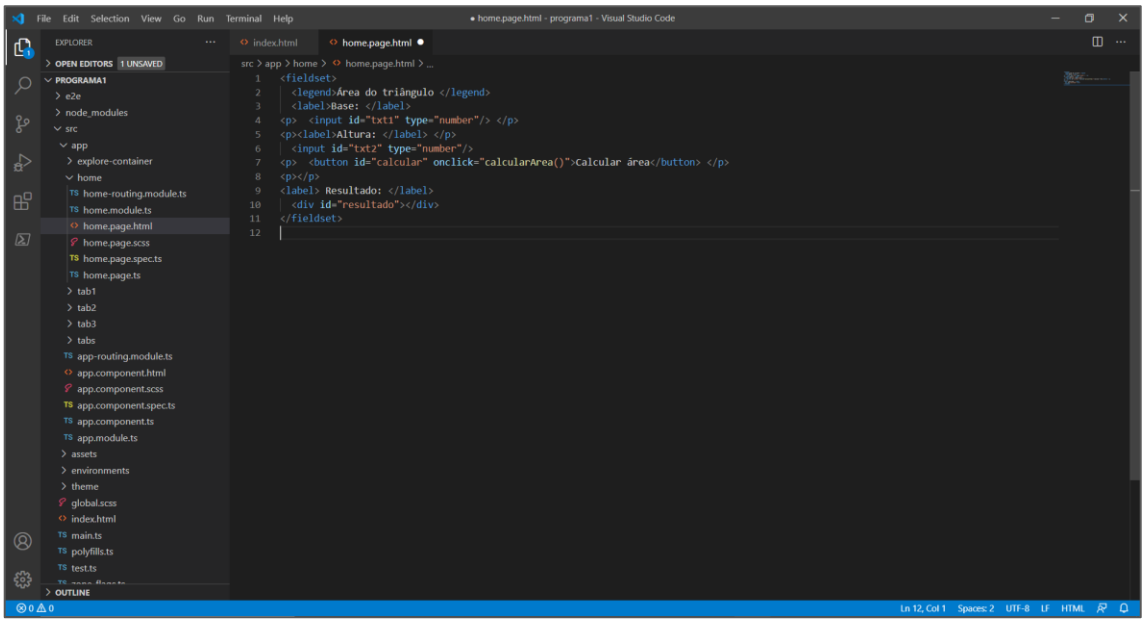
```
<fieldset>
<legend>Área do triângulo </legend>
...</fieldset>
```

Assim:

- *fieldset:* agrupa elementos, no nosso caso, queremos agrupar tudo, então todo nosso código para interface estará contido neste *fieldset*;
- *legend:* uma legenda para o *fieldset*, colocamos o nome de "Área do triângulo", pois é esse o propósito do nosso programa.

O código inteiro ficará assim:

```
<fieldset>
<legend>Área do triângulo </legend>
<label>Base: </label>
<p>  <input id="txt1" type="number"/> </p>
<p> <label>Altura: </label> </p>
<input id="txt2" type="number"/>
<p>  <button id="calcular" onclick="calcularArea()">Calcular área</button> </p>
<p></p>
<label> Resultado: </label>
<div id="resultado"></div>
</fieldset>
```



4.4. Passo 4

Função do Botão. Aqui iremos chamar função que determina o que o botão vai fazer quando acionado. O código ficará assim:

```
<script type="text/javascript">
  function calcularArea(){
  }
</script>
```

Descrições:

- *script:* utilizado para incluir códigos de javascript, como especificamos em *type*, com "text/javascript".
- *function calcularArea(){ }:* é a função que irmos chamar, que está programada na página index.html

O código final desta página ficará assim:

app.component.html | home.page.html | index.html

src > app > home > home.page.html > ...

```
1   <fieldset>
2     <legend>Área do triângulo </legend>
3     <label>Base: </label>
4   <p>  <input id="txt1" type="text"/> </p>
5   <p><label>Altura: </label> </p>
6     <input id="txt2" type="text"/>
7   <p>  <button id="calcular" onclick="calcularArea()">Calcular área</button> </p>
8   <p></p>
9   <label> Resultado: </label>
10    <div id="resultado"></div>
11  </fieldset>
12
13  <script type="text/javascript">
14    function calcularArea(){
15    }
16  </script>
17
18
19
```

5. ACESSANDO O GPS

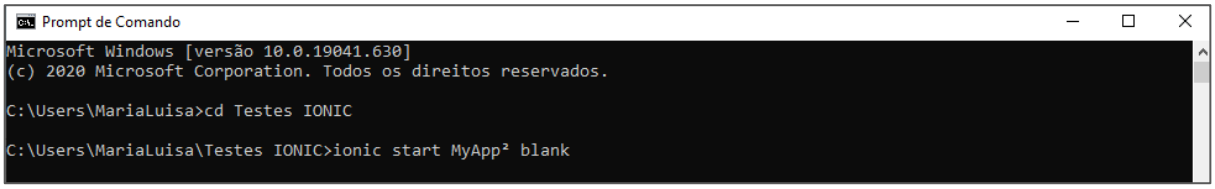
Neste capítulo, mostraremos o passo a passo da criação de um aplicativo que possui um botão que acesse o GPS do celular, descobrindo a latitude e longitude da nossa localização e exibindo essa informação na tela.

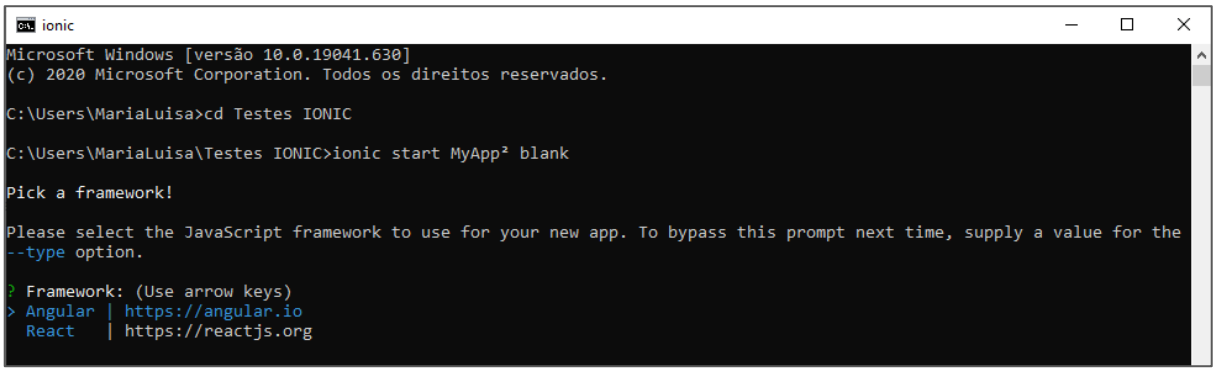
Além disso, o ponto também é plotado no mapa Google Maps ou equivalente. O objetivo deste projeto é mostrar como acessar recursos de hardware nativos do celular, no caso o GPS (Global Positioning System - Sistema de Posicionamento Global), e como integrar recursos de terceiros, no caso o Google Maps.

A seguir, encontra-se o passo-a-passo para a criação da aplicação.

5.1. Passo 1

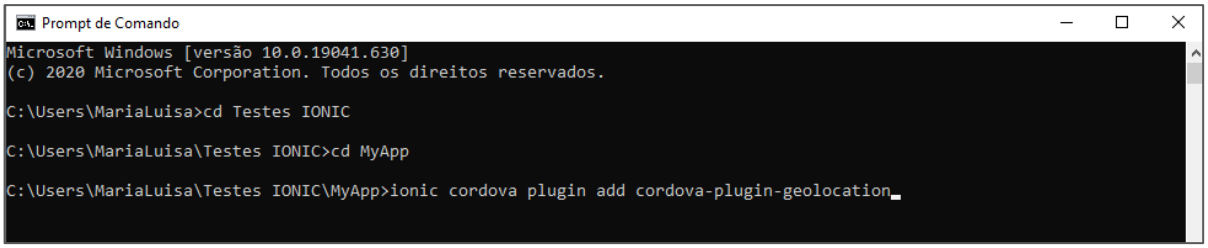
Crie uma aplicação em Ionic. O cmd irá pedir para o usuário escolher entre o Angular e o React, escolha o Angular.

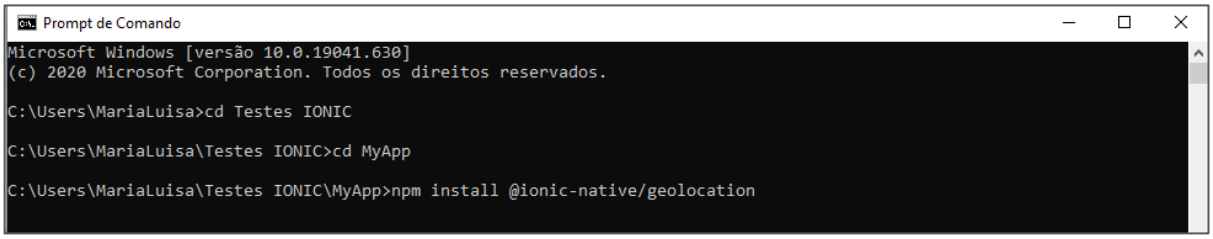




5.2. Passo 2

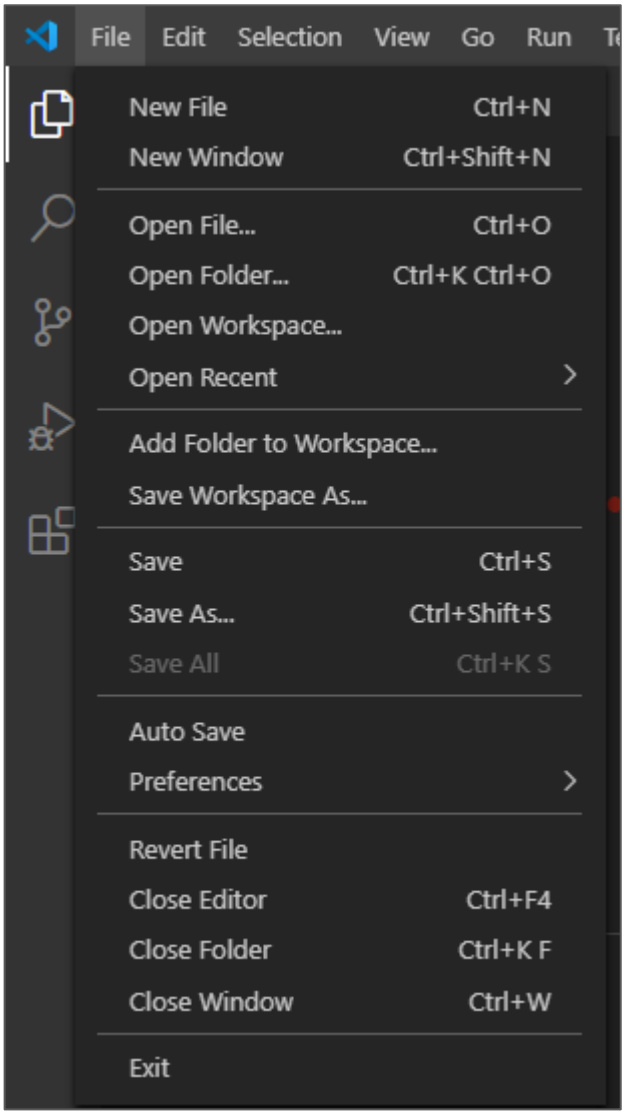
Entre na pasta do programa e, dentro dela, instale o plugin de geolocalização e a pasta com os seguintes comandos exemplificados nas imagens.





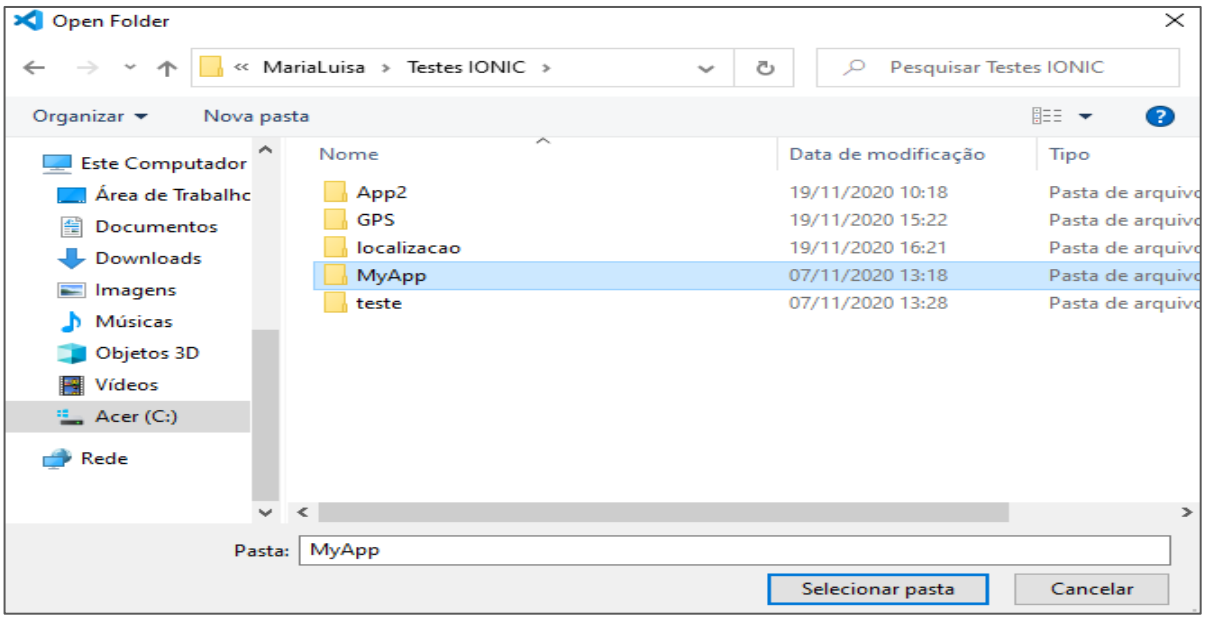
5.3. Passo 3

Abra o projeto no Visual Studio Code. Para isso, clique em “File” no canto superior esquerdo e, depois, clique em “Open Folder”.



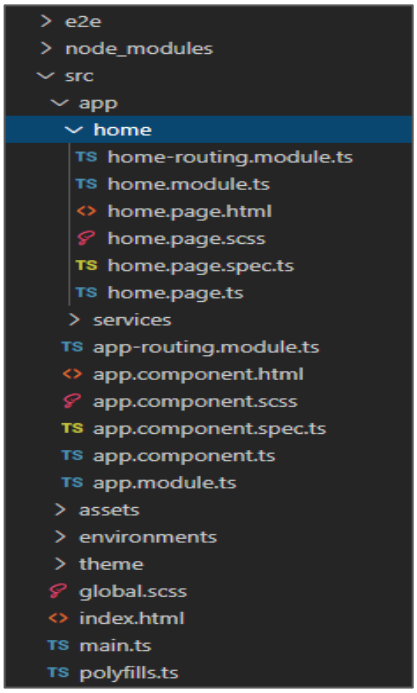
5.4. Passo 4

Selecione a pasta que você criou no cmd nos passos anteriores.



5.5. Passo 5

Na página inicial do Visual Studio Code, que aparece após a seleção da pasta, vá para o canto superior esquerdo da tela e clique em “scr”, depois em “app” e, enfim, em “home”. Visualize as pastas criadas.



5.6. Passo 6

Importe o Geolocation no home.module.ts.

- Clique em home.module.ts;
- Importe o Geolocation para que o projeto acesse as coordenadas: *import { Geolocation } from '@ionic-native/geolocation/ngx'*;
- Defina o Geolocation como um provedor: *providers: [Geolocation]*

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { HomePageRoutingModule } from './home-routing.module';
import { HttpClientModule } from '@angular/common/http';
import { Geolocation } from '@ionic-native/geolocation/ngx'; //Importa o Geolocation para que o projeto acesse as coord.


@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HttpClientModule,
    HomePageRoutingModule
  ],
  providers: [Geolocation] //Coloca o Geolocation como um provedor
})
export class HomePageModule { }
```

5.7. Passo 7

Programe a função do mapa no arquivo home.page.ts desta forma:

- Importe o Geolocation:

```
import {Geolocation} from '@ionic-native/geolocation/ngx';
```

- Importe o Leaflet, que é uma biblioteca JavaScript:

```
import * as Leaflet from 'leaflet';
```

- Defina as variáveis:

```
map: any; lon: number; lat: number;
```

- Chame o construtor, que leva como parâmetro o Geolocation:

```
constructor(private geolocation: Geolocation) { }
```

- Crie uma classe de modo que:

```
ngOnInit() {
```

- Traga a localização atual do usuário:

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

- Tome ciência das coordenadas:

```
this.map = Leaflet.map('mapId').setView([resp.coords.latitude, resp.coords.longitude], 15/*zoom*/);
```

- Coloque no mapa as coordenadas:

```
Leaflet.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
```

- Atribua um nome para o mapa: *attribution:*

```
'Mapa do Tcc';
```

- E determine o uso do “Leaflet” para construir o mapa a partir das coordenadas do usuário:

```
}).addTo(this.map);
```

- Insira o valor da latitude do usuário na variável “lat”:

```
this.lat = resp.coords.latitude;
```

- Insere o valor da longitude do usuário na variável “lon”:

```
this.lon = resp.coords.longitude;
```

● Defina o que a aplicação deve fazer - neste exemplo, usando “then” e “catch” - para tratar eventuais erros e exibir a mensagem para o usuário:

```
}).catch((error) => { console.log('Erro ao recuperar sua posição', error); })
```

```
import { Geolocation } from '@ionic-native/geolocation/ngx'; //Importa o Geolocation
import { Component, OnInit } from '@angular/core';
import * as Leaflet from 'leaflet'; //Importa o Leaflet, que é uma biblioteca JavaScript


@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})

export class HomePage {

  //Variaveis
  map: any;
  lon: number;
  lat: number;
```

```
  //Construtor que leva como parametro o Geolocation
  constructor(private geolocation: Geolocation) { }

  //Cria-se uma classe
  ngOnInit() {
    this.geolocation.getCurrentPosition() //Traz a localização atual do usuario
    .then((resp) => {
      this.map = Leaflet.map('mapId').setView([resp.coords.latitude, resp.coords.longitude], 15/*zoom*/);//nesta
      Leaflet.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', { //Coloca no mapa as coordenadas
        attribution: 'Mapa do Tcc', // Coloca nome no mapa
      }).addTo(this.map); // Usa-se o Leaflet para construir o mapa a partir das coordenadas do usuario

      this.lat = resp.coords.latitude; // Insere o valor da latidude do usuario na variavel lat
      this.lon = resp.coords.longitude; // Insere o valor da longitude do usuario na variavel lon

    }).catch((error) => {
      console.log('Erro ao recuperar sua posição', error); //then e catch tratam eventuais erros e exibem a mensa
    })
  }
}
```

5.8. Passo 8

Exiba o mapa por meio do arquivo home.page.html, mediante os seguintes comandos:

- Programe a exibição de um título para aplicação:

```
<ion-content>
<ion-label class="ion-text-center">
<div>
<h2> Latitude e Logintude: </h2>
```

- Programe a exibição da latitude e longitude na tela, provenientes da home.page.ts:

```
<h2> {{lat}}, {{lon}} </h2>
</div>
</ion-label>
```

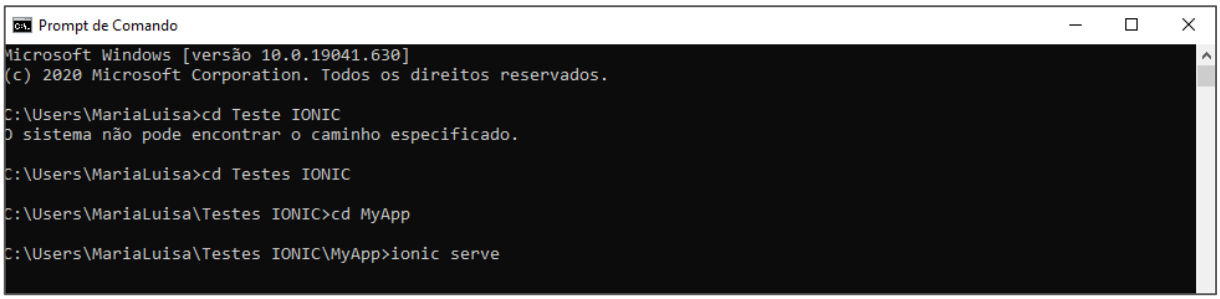
- Defina o tamanho do mapa na tela:

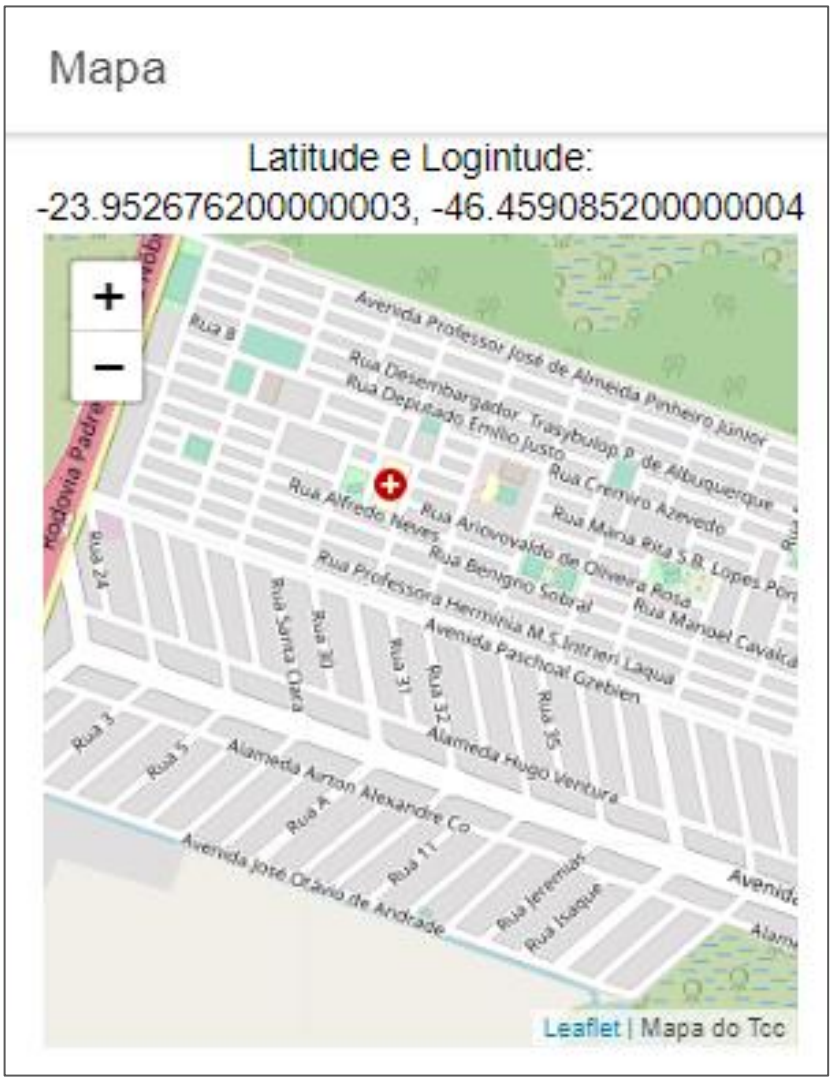
```
<div id="mapId" style="margin: auto; width: 90%;  height: 60%">
</ion-content>
```

```
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>Mapa</ion-title>
    </ion-toolbar>
  </ion-header>
  <ion-content>
    <ion-label class="ion-text-center">
      <div>
        <h2>Latitude e Logintude:</h2>
        <h2>{{lat}}, {{lon}}</h2> <!-- Traz para a tela a latitude e logintude da home.page.ts -->
      </div>
    </ion-label>

    <div id="mapId" style="margin: auto; width: 90%; height: 60%"></div> <!--dimensões do mapa-->
  </ion-content>
</ion-app>
```

5.9. Passo 9

Execute a aplicação e TCHARAM!!! A localização do usuário será mostrada no mapa da aplicação.





6. ACESSANDO OS CONTATOS

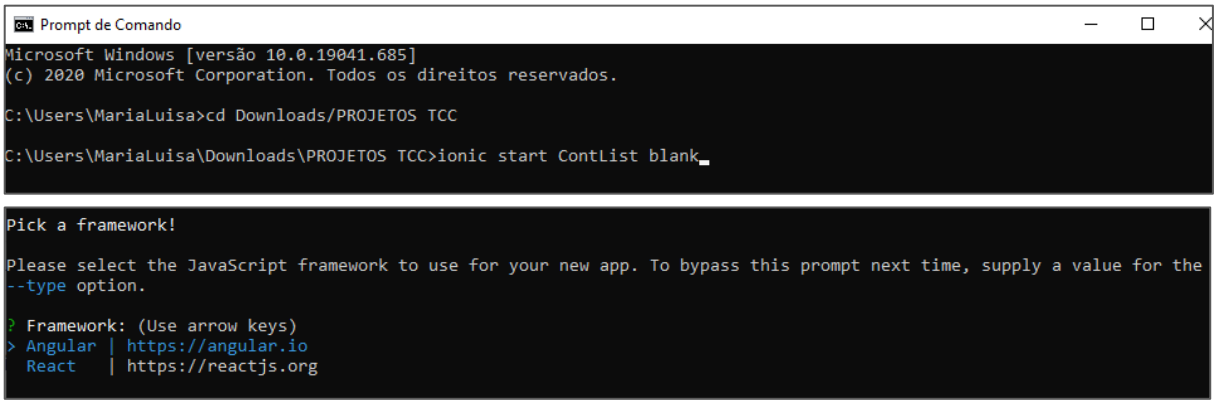
Aqui iremos criar um aplicativo onde deve-se digitar o nome de uma pessoa e pesquisar a existência desta na lista de contatos do celular do usuário, e, caso exista, exibir na tela seus dados (seu número de telefone).

O objetivo deste projeto é mostrar como acessar recursos de software nativos do celular, nativos do sistema Android, no caso, a lista de contatos.

A seguir, encontra-se o passo-a-passo para a criação da aplicação requerida.

6.1. Passo 1

Crie uma aplicação em Ionic. O cmd irá pedir para o usuário escolher entre o Angular e o React, escolha o Angular.



6.2. Passo 2

Entre na pasta do programa e, dentro dela, instale o plugin do Cordova.

```
C:\Users\MariaLuisa\Downloads\PROJETOS TCC\ContList>ionic cordova plugin add cordova-plugin-contacts
```

6.3. Passo 3

Ainda dentro da pasta do programa - no exemplo, “ContList” - faça a instalação do Native Contacts.

```
C:\Users\MariaLuisa\Downloads\PROJETOS TCC\ContList>npm install @ionic-native/contacts
```

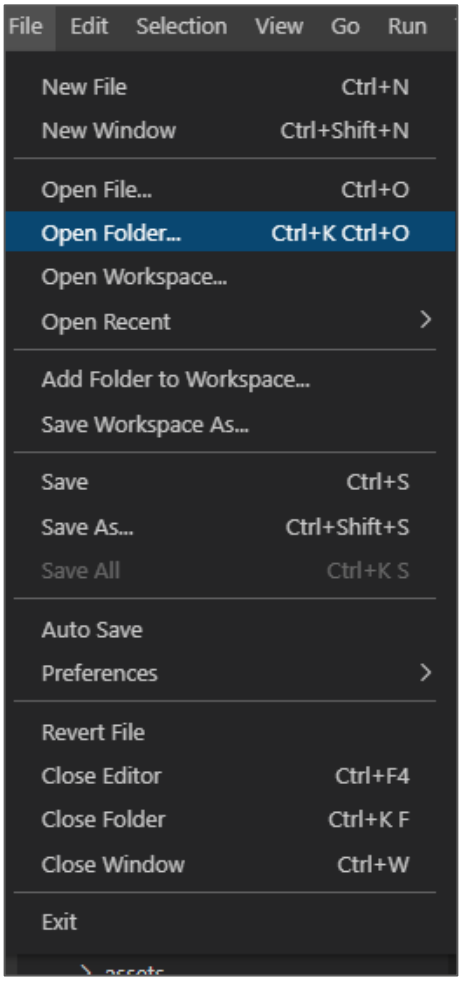
6.4. Passo 4

Faça a instalação dos recursos necessários para utilizar o aplicativo pelo sistema Android.

```
C:\Users\MariaLuisa\Downloads\PROJETOS TCC\ContList>ionic cordova platform add android
```

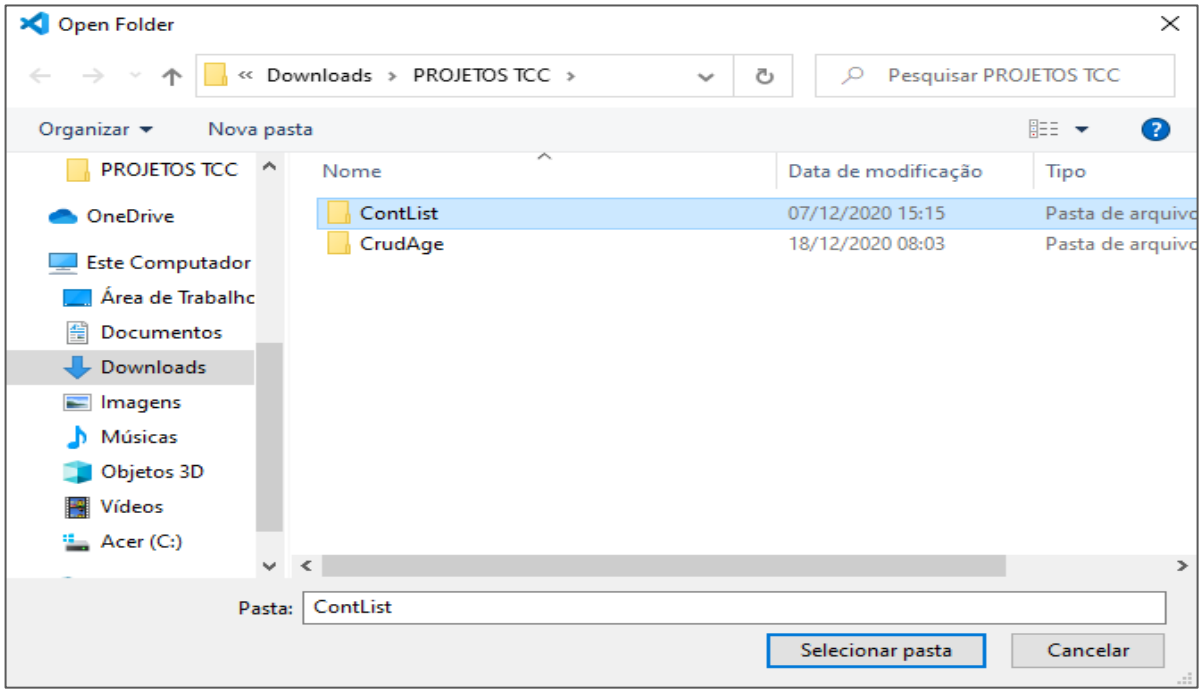
6.5. Passo 5

Abra o projeto no Visual Studio Code. Para isso, clique em “File” no canto superior esquerdo e, depois, clique em “Open Folder”.



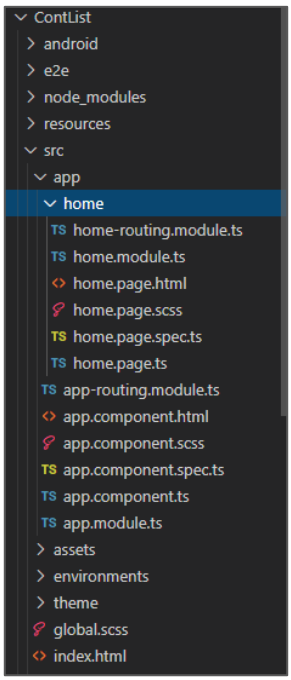
6.6. Passo 6

Selecione a pasta que você criou no cmd nos passos anteriores.



6.7. Passo 7

Na página inicial do Visual Studio Code, que aparece após a seleção da pasta, vá para o canto superior esquerdo da tela e clique em “scr”, depois em “app” e, enfim, em “home”.

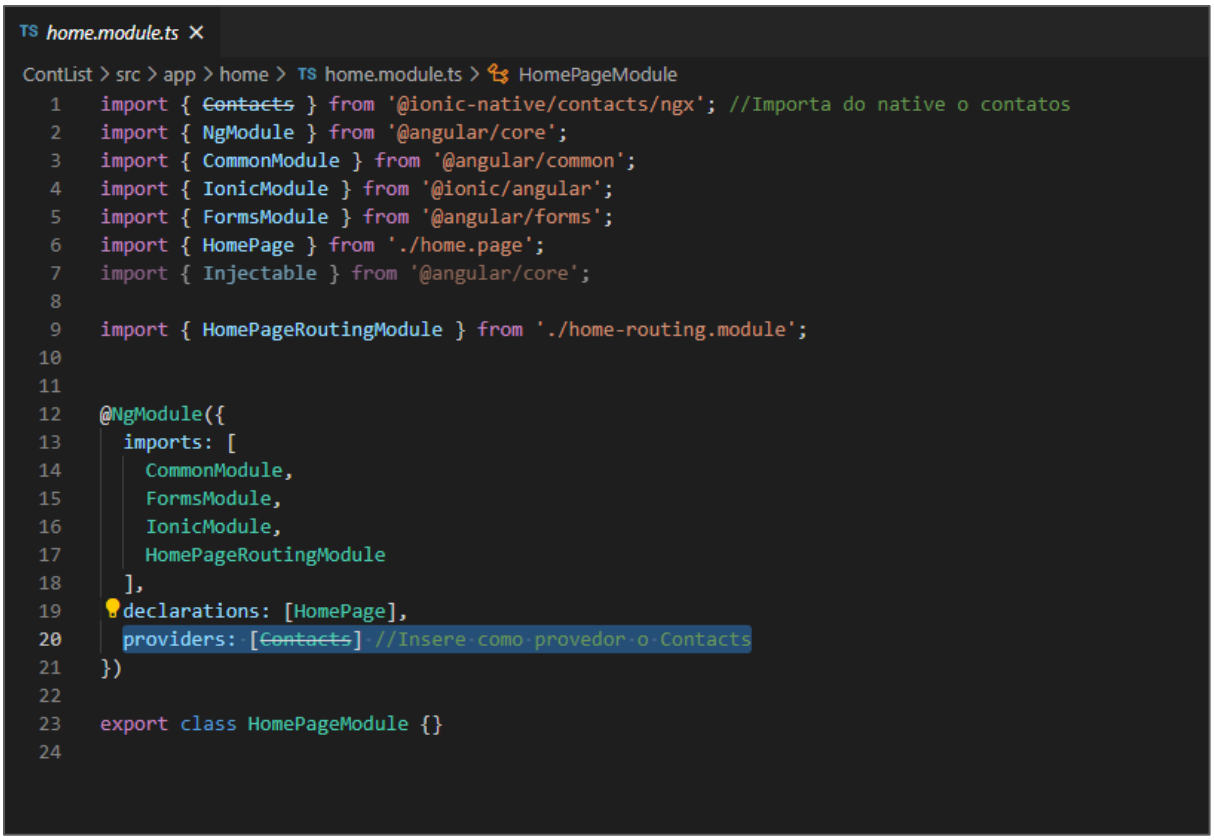


Visualize as pastas criadas e arquivos do projeto

6.8. Passo 8

No arquivo home.module.ts, importe Contacts do Native e depois insira-o como provedor.

- Clique em home.module.ts;
- Importe Contacts do Native:
  import { Contacts } from '@ionic-native/contacts/ngx';
- Defina o Contacts importado como provedor:
  providers: [Contacts]



6.9. Passo 9

No arquivo home.page.ts, faça as configurações para a pesquisa na lista de contatos da seguinte forma:

- Importe o Contacts do Native para permitir o acesso à lista de contatos do usuário:

```
import { Contact, ContactFieldType, Contacts, IContactFindOptions } from '@ionic-native/contacts/ngx'
```

- Faça a criação da variável “nome” para inserção do nome que aparecerá no display:

```
nome: ContactFieldType[] = ["displayName"];
contactsFound = [];
```

- Crie e chame o construtor que leva como parâmetro o Contacts, referenciando-o:

```
constructor(private Contacts: Contacts) { }
```

- Programe a exibição dos nomes que contêm a letras na ordem em que são digitadas, como uma espécie de varredura:

```
loadContats(q) {
const option: IContactFindOptions = { filter: q  }
```

- Programe a busca do nome digitado na lista de contatos do usuário:

```
this.Contacts.find(this.nome, option).then(conts => {
this.contactsFound = conts
});
```

- Exiba os contatos na tela, de acordo com as letras digitadas pelo usuário na busca:

```
onKeyUp(ev) {
his.loadContats(ev.target.value);
}
```

home.page.ts

ContList > src > app > home > home.page.ts > HomePage

```
import { Contact, ContactFieldType, Contacts, IContactFindOptions } from '@ionic-native/contacts/ngx';
//Importa do native o contatos para termos acesso a lista de contato do usuario
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})


export class HomePage {

  nome: ContactFieldType[] = ["displayName"]; //Cria a variavel nome e insere nela o Nome que aparece no display
  contactsFound = [];

  constructor(private Contacts: Contacts) { } //Cria um construtor e referencia o Contact

 loadContats(q) {
   const option: IContactFindOptions = {
     filter: q //A cada letras digitada, faz uma varredura na lista e exibe todos os nomes existentes com aquela letra (Na ordem que inserida)
   }
   this.Contacts.find(this.nome, option).then(conts => {
     this.contactsFound = conts
   }); //Utiliza a ferramenta de procura do Contacts para procurar na lista de contatos do usuario
 }

 onKeyUp(ev) {
   this.loadContats(ev.target.value); //Carrega os contatos na tela com base nas letras que vão sendo digitadas
 }
}
```

6.10. Passo 10

No arquivo home.page.html, insira uma barra de pesquisa e faça a programação para a exibição do número e do nome do contato pesquisado.

- Crie a barra de pesquisa e referência o evento:

```
<ion-searchbar (keyup)="onKeyUp($event)">
```

- Selecione, na lista de contatos, o contato encontrado:

```
<ion-item *ngFor="let contact of contactsFound">
```

- Exiba o contato encontrado:

```
<h2>{{contact?.displayName}}</h2>
```

- Selecione o número de telefone do contato encontrado:

```
<p *ngFor="let num of contact?.phoneNumbers">
```

- Programe para exibir o número do contato ao lado do nome.

```
{{num.value}}
```

home.page.html

ContList > src > app > home > home.page.html > ion-content

```
<ion-header>
  <ion-toolbar>
    <ion-searchbar (keyup)="onKeyUp($event)"></ion-searchbar> <!-- Cria a Barra de Pesquisa e referencia o evento -->
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <ion-list>
    <ion-item *ngFor="let contact of contactsFound"> <!-- *NgFor é uma diretiva que selecionando nos contatos o contato encontrado -->
      <h2>{{contact?.displayName}}</h2>  <!-- Mostra na tela o contato achado -->
      <p *ngFor="let num of contact?.phoneNumbers"> <!--  *NgFor é uma diretiva que está selecionando o numero do contato encontrado -->
        {{num.value}} <!-- Mostra ao lado do nome o número do contato -->
      </p>
    </ion-item>
  </ion-list>
</ion-content>
```

6.11. Passo 11

Para executar a aplicação, é necessário instalá-la no celular ou utilizar um modulador. O processo de verificação direto no celular é mais rápido e pode ser feito da seguinte maneira:

1. Ative as ferramentas de programador do celular, para que o usuário se torne um;
2. Ative a depuração por USB;

3. Conecte o celular no computador;
4. Vá até a página *chrome://inspect/#devices* e certifique-se que seu aparelho está aparecendo.
5. Vá até a pasta do seu projeto no cmd e rode o seguinte comando, para fazer com que a execução do aplicativo se inicie no celular:

6. A aplicação deve ser iniciada dentro de alguns minutos no celular.

```
C:\Users\MariaLuisa\Downloads\PROJETOS TCC\ContList>ionic cordova run android
```

6.12. Passo 12

Execute a aplicação conforme seu objetivo de busca e TCHARAM!!! A pesquisa do contato salvo no celular irá se parecer com o exemplo abaixo:

7. CRUD

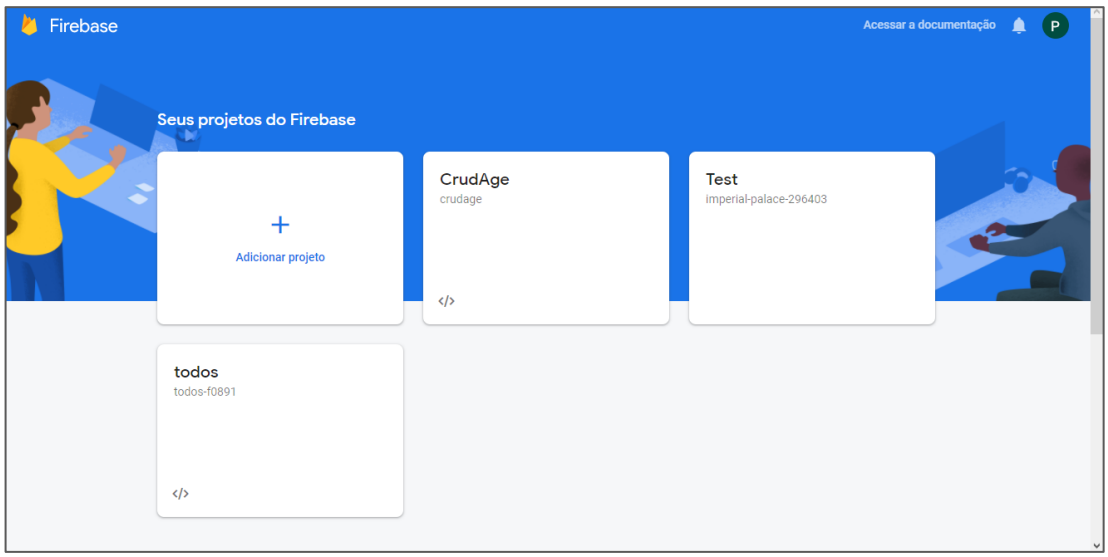
A nossa última proposta para essa apostila é a criação de um aplicativo onde o usuário possa inserir, consultar, atualizar e deletar um contato de sua lista. Fora isso, outro ponto é o armazenamento dos dados no recurso firebase da empresa Google.

O objetivo deste projeto é mostrar como acessar recursos de software nativos do celular, nativos do sistema Android, e criar a ligação entre o aplicativo e o Firebase.

A seguir, encontra-se o passo-a-passo para a criação da aplicação requerida.

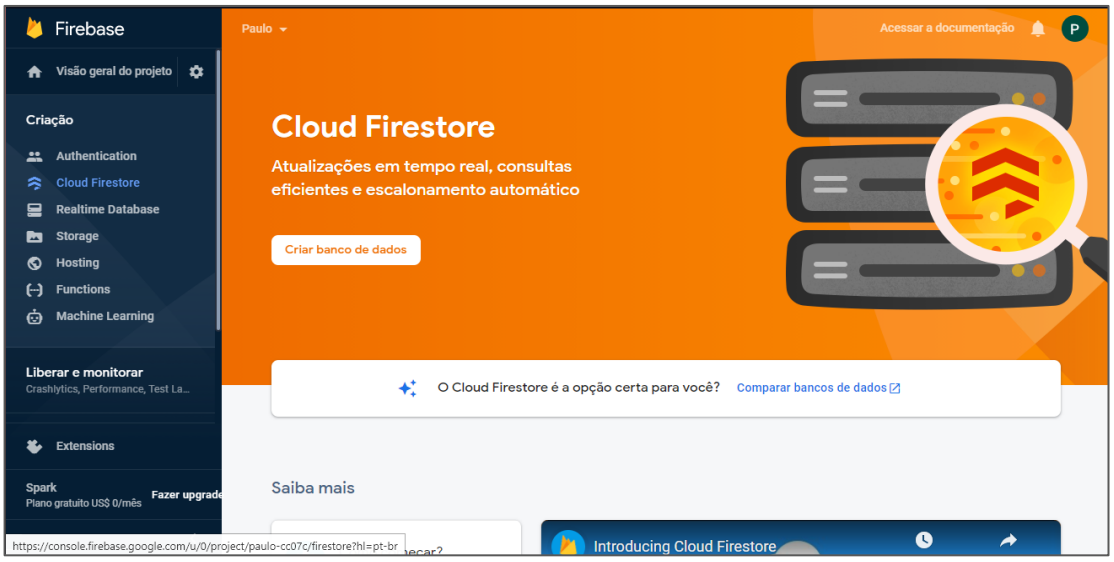
7.1. Passo 1

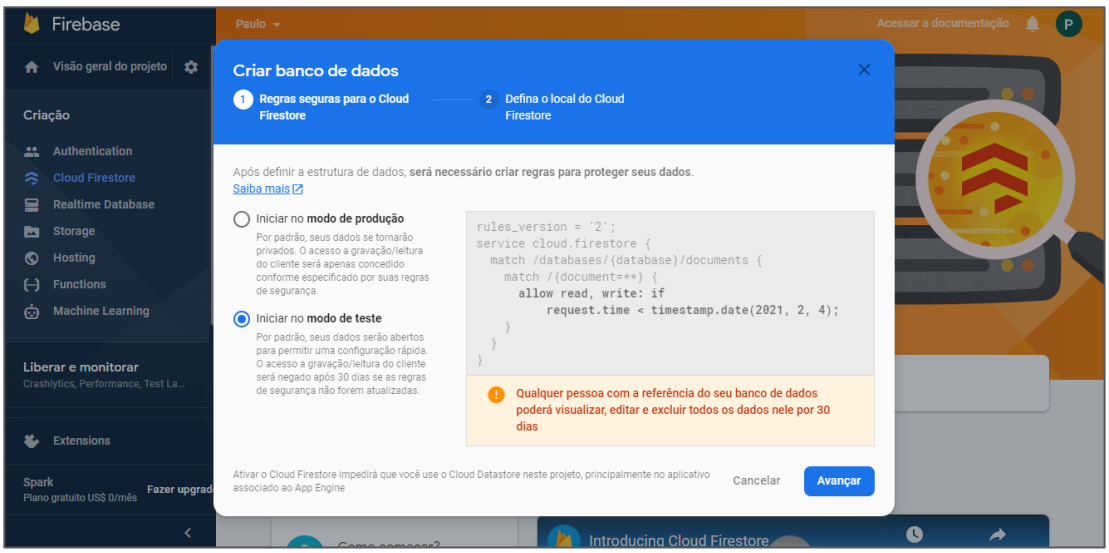
Se redirecionar ao site do FireBase para criar um novo projeto.



7.2. Passo 2

Na aba de Cloud Firestore, criar uma nova base de dados, utilizando a opção de modo de prova.

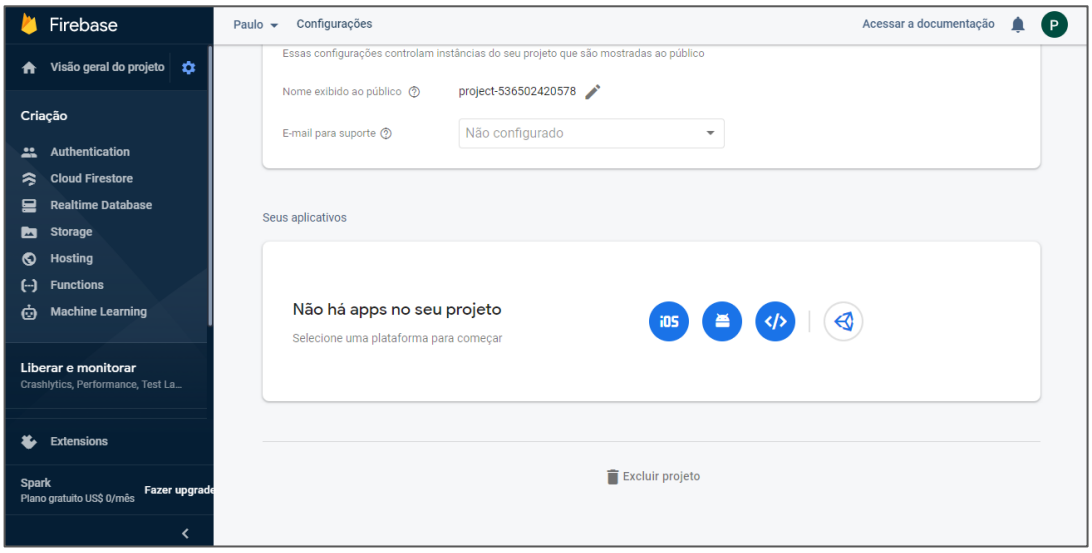




7.3. Passo 3

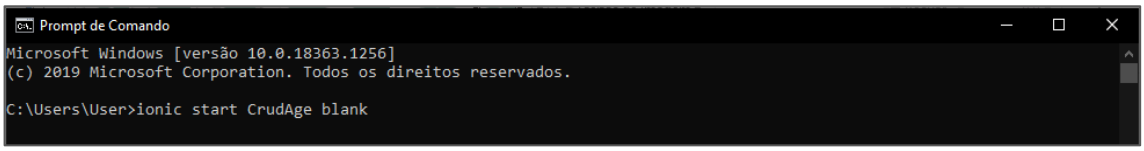
Com a base de dados pronta, ir até a engrenagem e verificar as configurações do projeto.

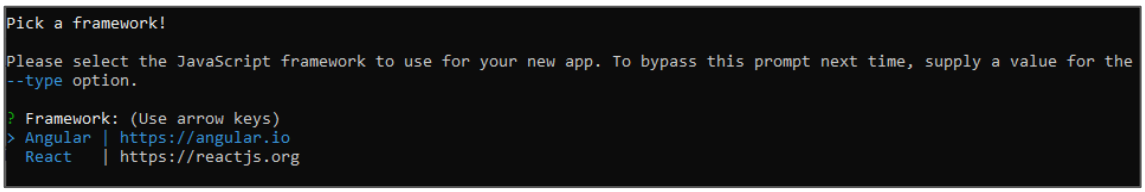
Desça um pouco a página e clique no ícone “</>”, lá você terá as informações necessárias para o projeto.



7.4. Passo 4

Feito isso, vamos para a criação do projeto. O cmd irá pedir para o usuário escolher entre o Angular e o React, escolha o Angular.

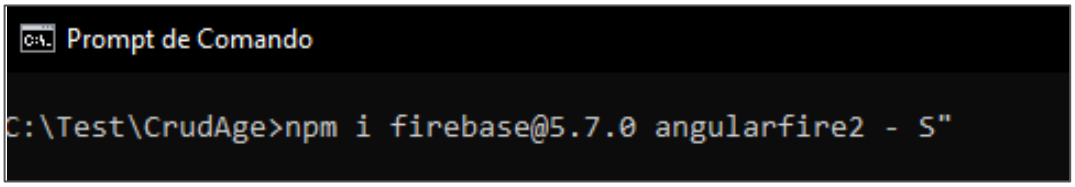




7.5. Passo 5

Ao terminar o carregamento, terá uma opção de instalação que pode ser ignorada. Se dirija a pasta do projeto com o comando “cd CrudAge”.

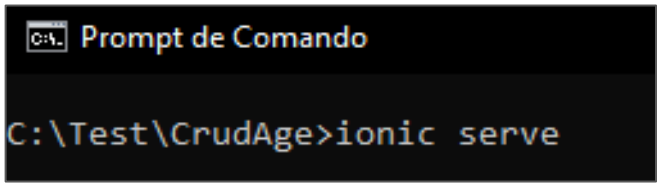
Para dar continuidade ao projeto serão necessários alguns downloads. Para isso execute o comando: “npm i firebase@5.7.0 angularfire2 – S”



7.6. Passo 6

Com as dependências instaladas, vamos para o Visual Code. Para isso basta digitar code em sua barra de pesquisa que o visual se abrirá.

Ainda no Ionic defina o servidor local através do seguinte código: ionic serve. Com isso teremos uma aba do navegador com a aplicação.



No visual code terá alguns fichários, prossiga na seguinte ordem: src>enviroments>enviroment.ts

Ainda se lembra das informações que pegamos no passo 4? Elas seão utilizadas dentro desse do arquivo enviroment. Após configurada teremos algo assim:

```
export const environment = {
  production: false,
  firebaseConfig : { //Configura o Firebase (O codigo já vem pronto no proprio firebase google, só copiamos
    apiKey: "AIzaSyAhOU5iLMGpx5EUgPUjatQ3cwBdpOkjtog",
    authDomain: "crudage.firebaseapp.com",
    databaseURL: "https://crudage-default-rtdb.firebaseio.com",
    projectId: "crudage",
    storageBucket: "crudage.appspot.com",
    messagingSenderId: "286405968244",
    appId: "1:286405968244:web:c4f27057f62275046cec21",
    measurementId: "G-8W5KQVF2DX"
  }
};
```

7.7. Passo 7

Agora abriremos um outro ficheiro, o app.module.ts. Nele você fará algumas importações, as importações necessárias serão a do AngularFireModule para baixo:

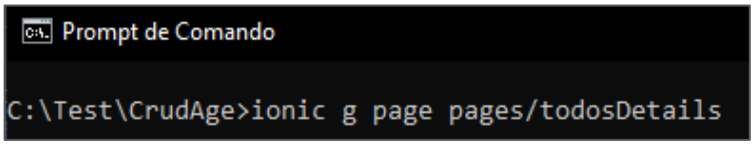
```
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';

import { AngularFireModule } from 'angularfire2';
import { environment } from '../environments/environment';
import { AngularFirestoreModule } from 'angularfire2/firestore';
//Importa todas as bibliotecas necessarias
```

7.8. Passo 8

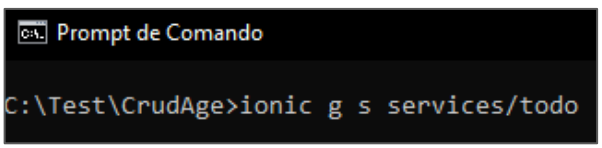
Com as importações feitas teremos que instanciar as mesmas. Descendo um pouco a aba que você está (app.module.ts) você verá que tem um trecho chamado de imports, nele adicione o AngularFireModule.IniatializeApp(enviroment.firebaseConfig) e também o AngularFIrestoreModule:

```
imports: [
  BrowserModule, IonicModule.forRoot(), AppRoutingModule,
  AngularFireModule.initializeApp(environment.firebaseConfig),
  AngularFirestoreModule,
  AngularFireDatabaseModule
],
```

Serão necessárias mais coisas para desenvolver a aplicação, com isso retorne ao console e digite o comando “ionic g page pages/todosDetails”.



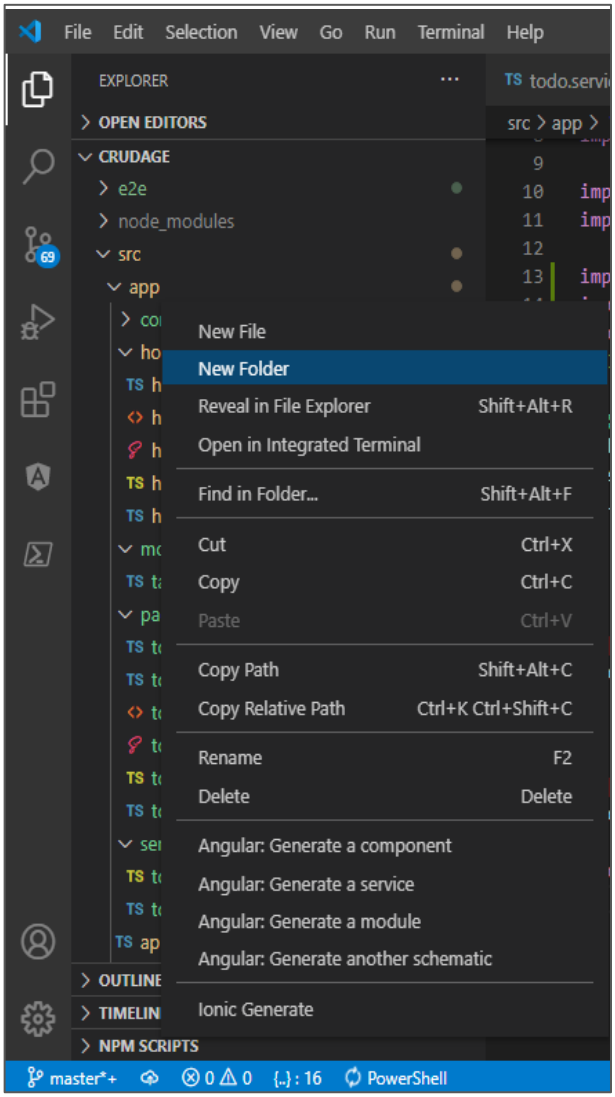
Na sequência digite “ionic g s services/todo”.



7.9. Passo 9

Agora digite novamente o ionic serve para reconectar.

Para o momento você terá que criar manualmente a interface. Para tal volte aos fichários do Visual Code abra o src e aperte com o botão direito sobre o app e com isso crie um New Folder.



Nomeie esse novo folder de models.

Dentro do mesmo crie um novo arquivo chamado de task.interface.ts. Após, devemos exportar na interface o id, o task e o priority.

```
export interface TaskI {
    id?:string; //O ? significa que o Id é opcional.
    task: string;
    priority: number;
}
```

7.10. Passo 10

Agora iremos trocar de fichário novamente – src>app>services> app-routing.module.ts. Fazendo as devidas modificações e adições o código dessa aba ficará assim:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomePageModule' },
  { path: 'details/:id', loadChildren: './pages/todo-details/todo-details.module#TodoDetailsPageModule' },
  { path: 'details', loadChildren: './pages/todo-details/todo-details.module#TodoDetailsPageModule' },
  {
    path: 'contact-edit',
    loadChildren: () => import('./contact-edit/contact-edit.module').then( m => m.ContactEditPageModule)
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Agora deixaremos isso de lado e iremos para os fichários, faça o seguinte caminho sr>app>services(expanda o services)>todo.service.ts.

Adicione algumas importações:

```
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreCollection } from 'angularfire2/firestore'; //Importa a biblioteca do Firebase para utilizarmos
import { Observable } from 'rxjs';  //Importa a biblioteca do RXjS para utilizarmos no programa
import { Action } from 'rxjs/internal/scheduler/Action';
import { map } from 'rxjs/operators';
import { TaskI } from '../models/task.interface';  //Importa a biblioteca da Interface para utilizarmos no programa
```

7.11. Passo 11

Descendo um pouco a página você adicionará algumas coisas dentro da classe TodoService:

```
export class TodoService {

  private todosCollection: AngularFirestoreCollection<TaskI>;
  private todos: Observable<TaskI[]>;

  constructor(db:AngularFirestore) {
    this.todosCollection = db.collection<TaskI>('todos'); //Conexão com a interface
    this.todos = this.todosCollection.snapshotChanges().pipe( //Nesta linha começa a conexão em tempo real com o Firebase, com o metodo +
      //snapshotChanges e o pipe que está combinar várias funções em uma única função.
      map(actions => { //O map é usado para modificar o dado no Observable atual
        return actions.map(a => {
          const data = a.payload.doc.data(); //Recuperamos todos os campos do documento como um objeto.
          const id = a.payload.doc.id; //Obtemos o ID do documento
          return {id, ...data}; // Retornamos o id e o data com o operador de propagação
        });
      })
    );

  }
```

Agora teremos que criar os métodos, get, update, add e remove no processo:

```
getTodos(){
  return this.todos; //Traz os dados do Firebase para o Programa
}

getTodo(id: string){
  return this.todosCollection.doc<TaskI>(id).valueChanges(); //Traz os dados do Firebase para o Programa
}

updateTodo(todo:TaskI, id: string){
  return this.todosCollection.doc(id).update(todo); //Faz o Update no Firebase
}

addTodo(todo: TaskI){
  return this.todosCollection.add(todo); //Adiciona no Firebase
}

removeTodo(id: string){
  return this.todosCollection.doc(id).delete(); //Deleta no firebase
}
```

Trocaremos mais de uma vez. Dessa vez vamos para src>app>home> home.page.html

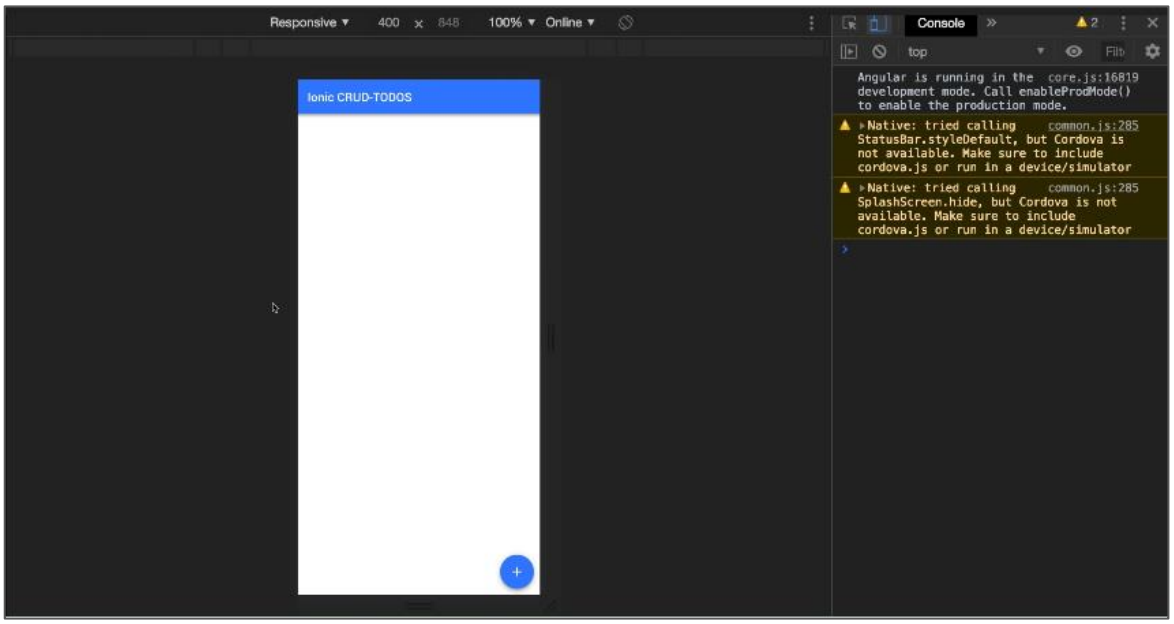
Agora teremos que alterar o home.page.html da seguinte forma, para construirmos a interface gráfica do projeto:

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>
      Crud
    </ion-title>
  </ion-toolbar>
</ion-header>
<ion-content>
  <ion-list>
    <ng-container *ngIf="!todos || todos.length == 0">
      <div *ngFor="let n of [0,1,2]" padding>
        <ion-skeleton-text></ion-skeleton-text>
        <p>
          <ion-skeleton-text class="text-skeleton"></ion-skeleton-text>
        </p>
      </div>
    </ng-container>
    <ion-item-sliding *ngFor="let todo of todos">
      <ion-item lines="inset" button [routerLink]="['/details', todo.id]">
        <ion-label>
          {{ todo.task }}
        </ion-label>
        <ion-note slot="end" color="primary">{{ todo.priority }}</ion-note>
      </ion-item>
```

```
      <ion-item-options side="end">
        <ion-item-option (click)="onRemove()" color="secondary">
          Check
          <ion-icon name="checkmark" slot="end"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
<ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button routerLink="/details" routerDirection="forward">
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
</ion-content>
```

7.12. Passo 12

Com isso tudo feito, retornaremos a página web onde temos o projeto e veremos ele da seguinte forma:



Voltamos para o Visual Code, no ficheiro [home.page.ts]
Nessa nova aba faremos mais importações:

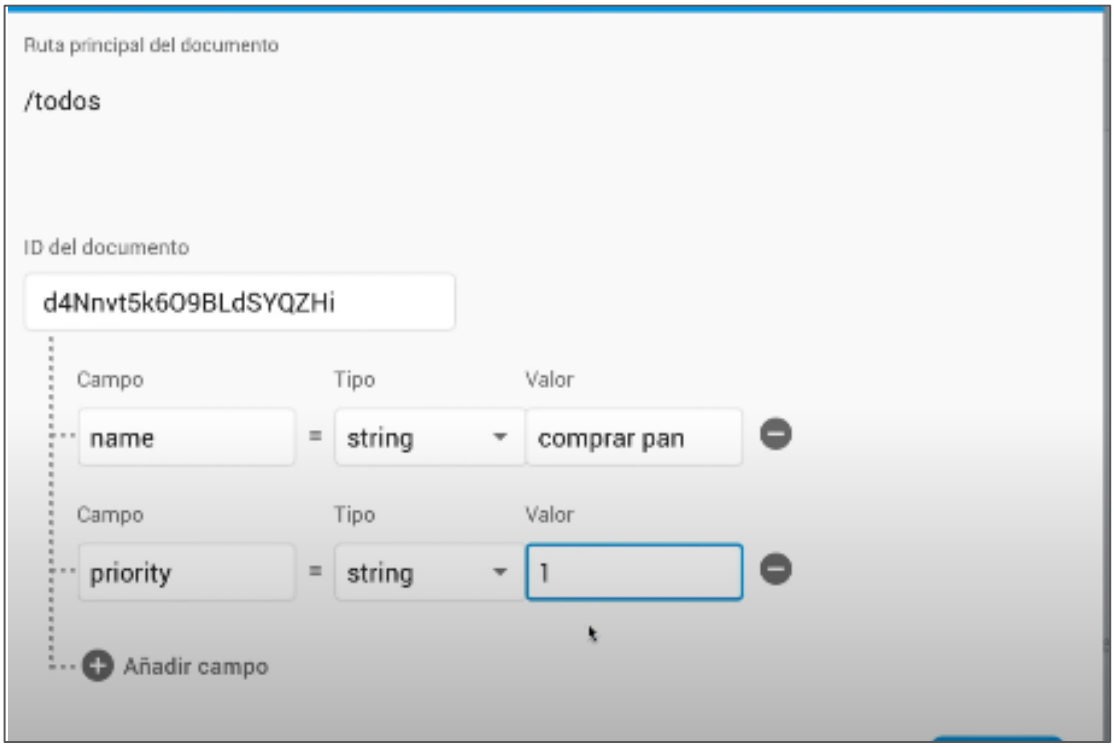
```
import { Component, OnInit } from '@angular/core';
import { TaskI } from '../models/task.interface';
import { TodoService } from '../services/todo.service';
```

A parte da classe terá algumas alterações para que o processe:

```
export class HomePage implements OnInit{
  todos: TaskI[];

  constructor(private todoService: TodoService){}
  //Coloca como parametro do construtor o serviço para realizar o CRUD

  ngOnInit(){
    this.todoService.getTodos().subscribe((todos) =>{ //Chama o serviço
      console.log('Todos', todos);
      this.todos = todos;
    })
  }
```

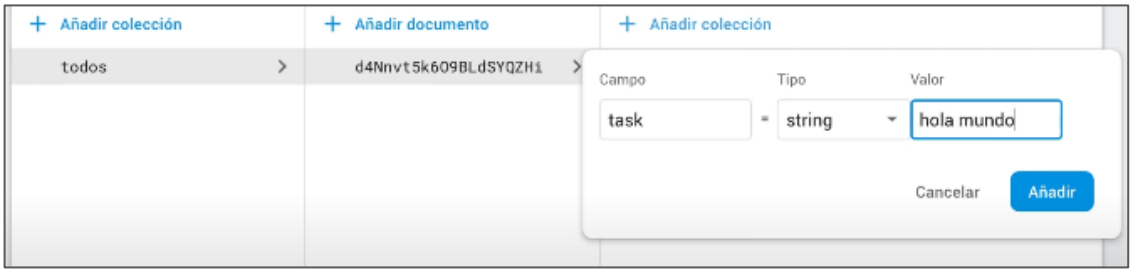
7.13. Passo 13

Agora teremos que retornar ao site firebase para testar se o servidor consegue receber as tarefas. Adiciona uma nova coleção, e preencha:

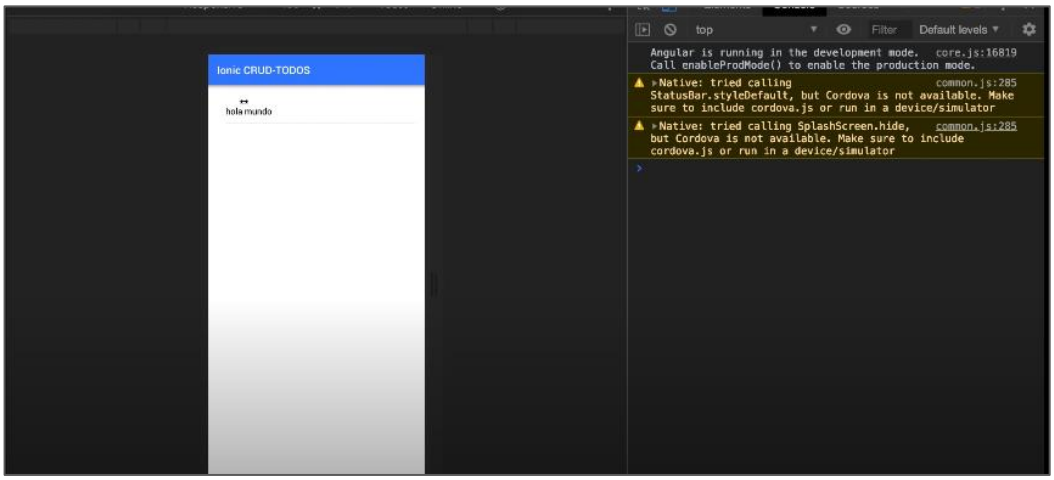


Com isso preenchido aqui você consegue observar na área de teste e no console que tem uma segunda aba Tarefas, nela teremos o que foi adicionado no Firebase.

Agora para testar que a tarefa está no lugar correto, você precisará retornar ao Firebase e adicionar um campo na sua mini base de dados



Se a Task aparecer, deste modo da imagem a seguir, significa que está tudo caminhando para termos um projeto bom.



7.14. Passo 14

Novamente voltando aos fichários e iremos para o todo-datails.page.ts. Vamos adicionar mais importações aqui.

```
import { Component, OnInit } from '@angular/core';
import { TaskI} from '../../models/task.interface';
import { TodoService } from '../../services/todo.service';
import { ActivatedRoute} from '@angular/router';
import { NavController, LoadingController } from '@ionic/angular';
```

Dando continuidade as modificações:

```
export class TodoDetailsPage implements OnInit {

  todo: TaskI = { //Inicializão dos campos
    task: '',
    priority: null
  };

  todoId= null; //Id só vai introduzido no proprio firebase

  constructor(private route: ActivatedRoute, private nav: NavController, private todoService: TodoService, private loadingController: LoadingController) { }

  ngOnInit() {
```

```
ngOnInit() {
  this.todoId = this.route.snapshot.params['id'];
  if (this.todoId){
    this.loadTodo();
  }
}
```

```
async loadTodo(){ //Cria o 1º Load para a inicialização do Firebase
  const loading = await this.loadingController.create({
    message: 'Aguarde....'
  });
  await loading.present();

  this.todoService.getTodo(this.todoId).subscribe(todo => {
    loading.dismiss();;
    this.todo = todo;
  });
}
```

Ainda no todo-details vamos criar outros métodos, para salvar e para excluir.

```
async saveTodo() { //Cria o loanding para salvar no Firebase
  const loading = await this.loadingController.create({
    message: 'Salvando....'
  });
  await loading.present(); //Manda para loading.present até que seja adicionado no Firebase

  if (this.todoId) { //Caso já exista uma ID, como utilizaremos o mesmo botão para salvar e fazer o update, o botão troca de função
    this.todoService.updateTodo(this.todo, this.todoId).then(() => {  //Passa como parametro todo e o Id para que possa ocorrer o Update
      loading.dismiss(); //Tela de loading até se concluir a operação
      this.nav.navigateForward('/'); //Redireciona o usuario para a Homepage
    });
  } else {
    this.todoService.addTodo(this.todo).then(() => { //Passa como parametro todo para que possa ocorrer a adição
      loading.dismiss();//Tela de loading até se concluir a operação
      this.nav.navigateForward('/'); //Redireciona o usuario para a Homepage
    });
  }
}
async onRemoveTodo(idTodo:string) {
  this.todoService.removeTodo(idTodo); //Passa o ID para o Firebase com a instrução de remover tudo que está relacionado com ele.
}
```

7.15. Passo 15

Após isso, retornaremos ao server e teremos o seguinte projeto concluído: