UniFil
EDUCAÇÃO A DISTÂNCIA

Unidade 4

Desenvolvendo Aplicativos Móveis com IONIC Framework e Angular

Introdução

Neste módulo veremos como desenvolver um aplicativo com IONIC em um projeto de calculadora, voltado para aplicativos móveis, além de uma abordagem voltada para PHP e criação de banco de dados.

Objetivos

- Desenvolver um aplicativo utilizando *Framework* IONIC – Projeto Calculadora;
- Introduzir o desenvolvimento em IONIC – Lista Memória e Lista Storage;
- Desenvolver um aplicativo IONIC com instalação de banco de dados e com a linguagem PHP;
- Aprofundar os conhecimentos na criação de um banco de dados;
- Conhecer mais sobre IONIC utilizando PHP via API.

Conteúdo programático

Aula 01 – Desenvolvimento de aplicativo IONIC (Projeto Calculadora)

Aula 02 – Desenvolvimento de aplicativo com IONIC PHP e criação de um banco de dados

Você poderá também assistir às videoaulas em seu celular! Basta apontar a câmera para os **QRCodes** distribuídos neste conteúdo.

Pode ser necessário instalar um aplicativo de leitura QRcode no celular

Aula 01 – Desenvolvimento de aplicativo IONIC (Projeto Calculadora)

Olá pessoal, nesta aula abordaremos o desenvolvimento de um projeto de calculadora utilizando o IONIC. Nós vamos nos basear no mesmo modelo criado e prototipado no Balsamiq Mockup, depois construindo um MVP e usando MIT APP Inventor.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 1

Na construção deste projeto, verificaremos diversos aspectos de uma aplicação construída em IONIC, na qual iremos interagir entre interface e código. O Primeiro passo, será utilizar um projeto Blank e em seguida vamos criar o nosso projeto.

Acessaremos a pasta raiz do nosso projeto:

```
adail@Adail:~$ cd tads/
adail@Adail:~/tads$ ls
aula2  aula4  aula5  aula6  aula7
adail@Adail:~/tads$
```

Antes de criarmos o nosso projeto, é necessário verificar se o IONIC está funcionando corretamente, através do comando abaixo.

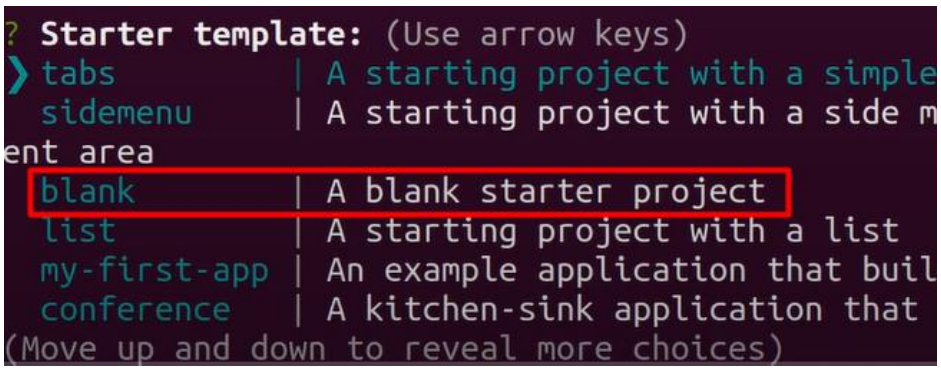
```
adail@Adail:~/tads$ ionic -v
6.16.3
```

Para iniciarmos a criação do nosso projeto, utilizaremos o comando abaixo para **start**.

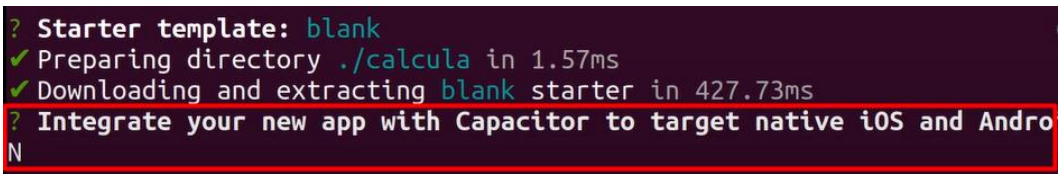
```
adail@Adail:~/tads$ ionic start calcula
```

Em seguida, é necessário escolher o modelo de tratamento do nosso projeto, sendo o **Angular**.

Escolher o *template* sendo **Blank.**



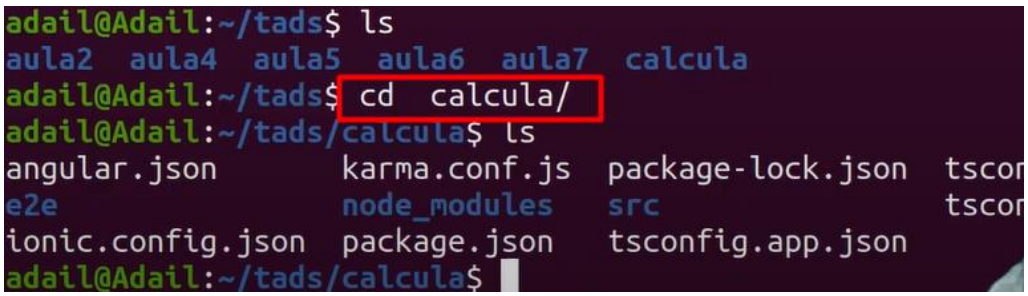
O sistema perguntará se vamos criar a integração em Android e neste momento selecionaremos **Não**.



Logo, o sistema terminará de realizar o *download*.

O sistema também perguntará se desejamos criar uma conta no IONIC e não será necessário. Ou seja, marcaremos como **Não**.

Assim que terminar, executar os comandos abaixo para entrar dentro da pasta do nosso projeto **Calcula**.

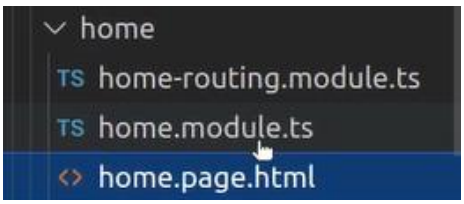


Neste momento, devemos iniciar o Visual Studio Code para trabalharmos com este projeto.

Executar o comando code. para abrir o Visual Studio Code.

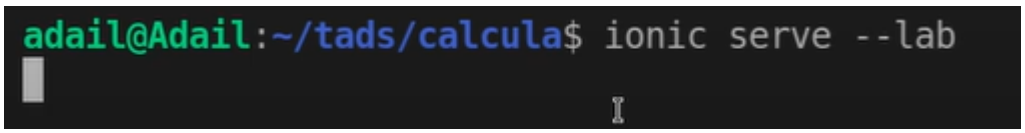
No caso do projeto Calculadora, não vamos criar mais páginas e sim alterar a estrutura desta única página.

Primeiramente, devemos navegar até a página *Home*.



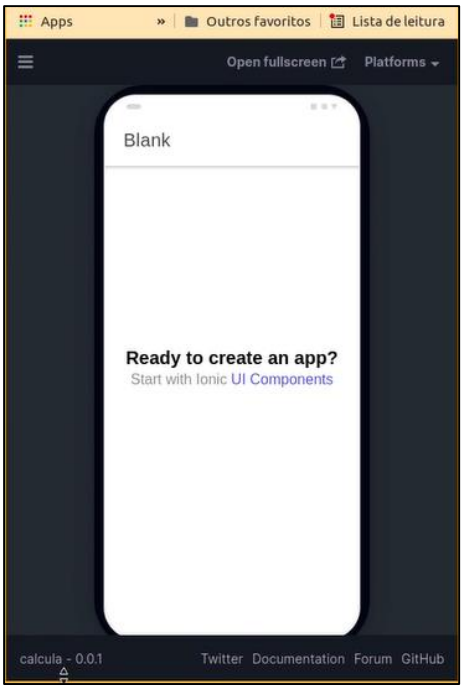
Para iniciar o projeto e visualizarmos de forma mais clara as alterações, clicar em terminal > *new terminal*.

Para iniciarmos nosso projeto, execute o comando ionic serve –lab.



Obs.: no primeiro momento que executamos nosso projeto, a ferramenta perguntará se desejamos realizar o *download* dos complementos necessários para executar a aplicação.

Nosso código em execução foi criado.

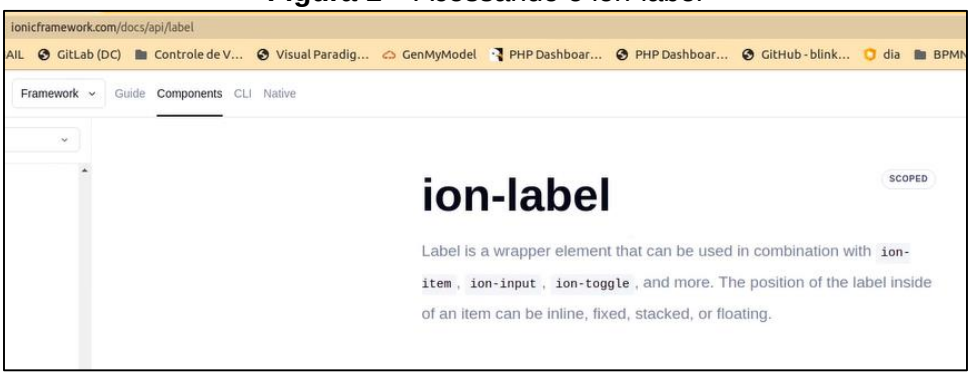


Basicamente, iremos remover (limpando) todo o conteúdo da página, restando apenas o código, como o formato abaixo:

```
1  <ion-content [fullscreen]="true">
2    <div id="container">
3      <h1>TESTE</h1>
4    </div>
5  </ion-content>
6
```

Em um primeiro momento, iremos buscar um componente adequado para a aplicação da calculadora, buscando nas documentações do IONIC (UI Componentes). No nosso cenário, utilizaremos o ION – LABEL.

Figura 1 – Acessando o ion-label



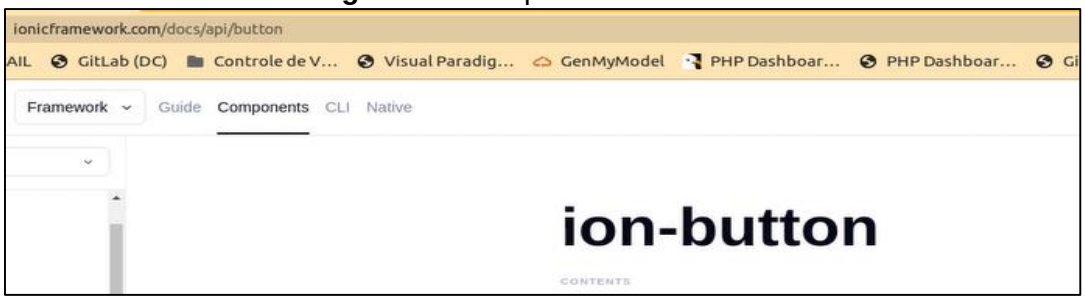
Fonte: disponível em: ionicframework.com/docs/api/label. Acesso em: 25 abr. 2021.

Copiaremos o código do ion – label alterando para 0 o conteúdo.

```
<ion-label>0</ion-label>
```

Voltaremos na documentação para verificar o componente ion-button.

Figura 2 – Componente ion-button



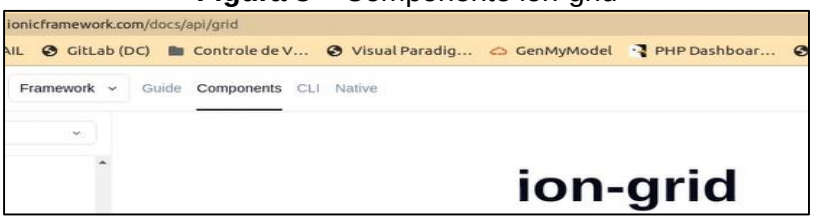
Fonte: disponível em: ionicframework.com/docs/api/button. Acesso em: 25 abr. 2021.

Copiaremos o código do ion – button para nossa aplicação.

```
<ion-button>Default</ion-button>
```

Para que consigamos organizar a aplicação da forma esperada, devemos utilizar um componente que nos auxilie neste processo. Neste caso, utilizaremos o componente ion – grid.

Figura 3 – Componente ion-grid



Fonte: disponível em: ionicframework.com/docs/api/button. Acesso em: 25 abr. 2021.

Vamos copiar uma estrutura de ion – grid para nossa aplicação.

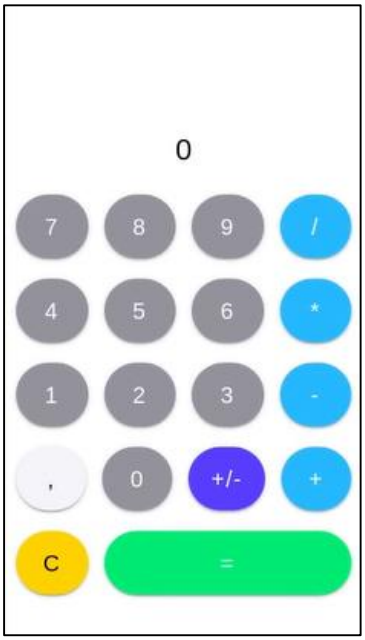
```
<div id="container">
  <ion-label>0</ion-label>
  <ion-button>Default</ion-button>
  <ion-grid>
    <ion-row>
      <ion-col>
        ion-col
      </ion-col>
      <ion-col>
        ion-col
      </ion-col>
      <ion-col>
        ion-col
      </ion-col>
      <ion-col>
        ion-col
      </ion-col>
    </ion-row>
  </ion-grid>
```

Organizaremos nosso código em formas de linhas e colunas para melhorar o aspecto destes botões. Atualizaremos o código ajustando o tamanho e as cores adequadas para que o usuário possa clicar. Em seguida, copiaremos o mesmo formato para os demais botões de nossa aplicação.

```
<ion-button size="large" color="medium" expand="block" shape="round">7<
```

Faremos uma cópia da primeira linha de botões para as demais linhas, alterando o conteúdo dos botões/elementos de acordo com a necessidade de nosso projeto.

Nosso protótipo inicial ficará conforme a imagem abaixo:



Videoaula 1

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 1.

Disponível em: https://www.youtube.com/watch?v=sFqExwXrx3E. Acesso em: 25 abr. 2021.

Videoaula 1

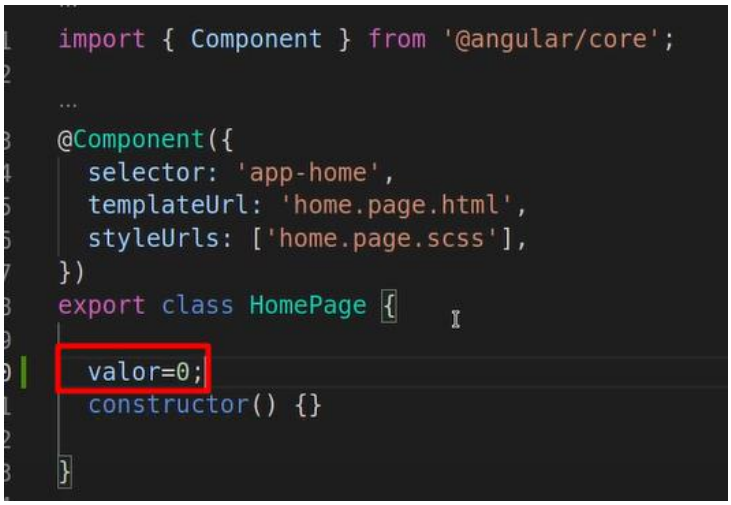
Utilize o QRcode para assistir!

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 1.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 2

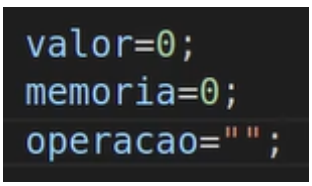
Daremos continuidade do nosso projeto de calculadora. Agora acessaremos a pasta home.page.ts que faz o comportamento de interface da calculadora.

O primeiro detalhe que vamos aprender a fazer é criar variáveis dentro do nosso código. A primeira variável que criaremos será para armazenar o valor que está sendo colocado na tela. Acessaremos a pasta home.page.ts e definiremos a variável **valor = 150**.

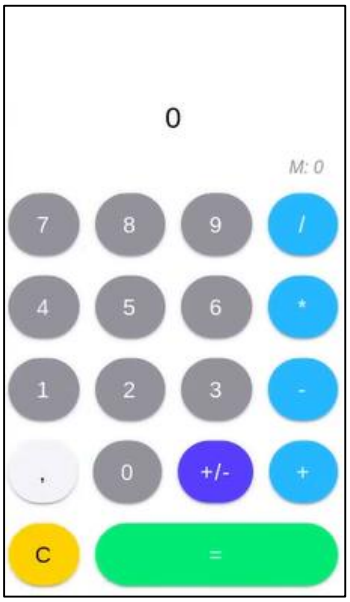


Dentro do diretório **home.page** iremos associar a variável criada com a interface da calculadora. O Código para realizarmos este vínculo está demonstrado abaixo:

Iremos retornar no diretório home.page para definição de outras vavíaveis necessárias para nossa aplicação, conforme abaixo:



Faremos algumas alterações nos códigos para melhor visualização da interface de nossa aplicação. Após alguns alinhamentos de recursos adicionais, nossa aplicação ficará desta forma:



Videoaula 2

Assista agora a videoaula na qual veremos mais sobre a criação do projeto de Calculadora – Parte 2. Disponível em: https://youtu.be/76ZtzA_RJI0. Acesso em: 25 abr. 2021.

Videoaula 2

Utilize o QRcode para assistir!

Assista agora a videoaula na qual veremos mais sobre a criação do projeto de Calculadora – Parte 2.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 3

Dando continuidade do nosso projeto de calculadora, vamos implementar o acionamento dos botões. A primeira ação que devemos fazer é criar uma função btNumerico dentro do diretório home.page.ts.

```
btNumerico(numero){
  this.valor=numero;
}
```

Dentro do nosso código html, acessaremos o botão 7 e incluiremos a função.

```
<ion-button (click)="btNumerico(7)"
```

Em sequência retornaremos para o diretório home.page.ts e incluiremos o tipo de algumas variáveis e ajustaremos à função.

```
valor:string="0";
memoria=0;
operacao="";

constructor() {}

btNumerico(numero){
  this.valor=this.valor+numero;
}
```

Criaremos a função **limpar**.

```
limpar(){
  this.valor="0";
}
```

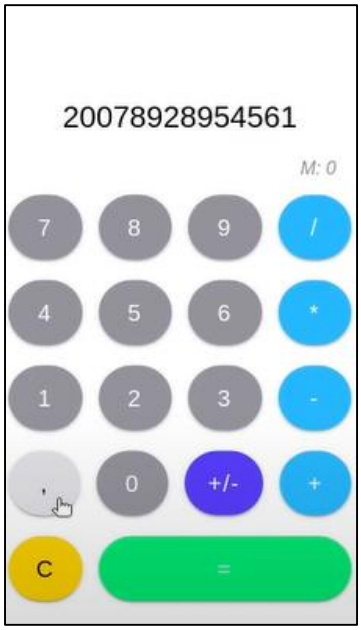
E associamos o botão **limpar** em nossa aplicação. Ajustaremos a função btNumerico para melhorar a concatenação.

```
btNumerico(numero){
  if (this.valor == "0") this.valor=numero;
  else this.valor=this.valor+numero;
}
```

O mesmo comportamento incluído no botão 7, será copiado para os demais botões de nossa aplicação.

```
<ion-button (click)="btNumerico('7')" size="large" color="medium" expand="bloc
```

Com a alteração no código, nossa aplicação, ficará com a seguinte interface modulada.



Videoaula 3

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 3.

Disponível em: https://youtu.be/J48pL3whVNA. Acesso em: 25 abr. 2021.

Videoaula 3

Utilize o QRcode para assistir!

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 3.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 4

Nesta parte da aula, iremos implementar o acionamento dos botões de vírgula e mais (+) e menos (-).

No diretório home.page.ts vamos criar mais uma função chamada btVirgula.

Dentro deste parâmetro, faremos uma condição comparando o valor informado na calculadora. Nosso código ficará como a imagem abaixo:

```
btVirgula(){
  if (this.valor.indexOf(".") == -1) {
    this.valor=this.valor+".";
  }
}
```

Para o botão de mais (+) e menos (-) faremos a criação de outra função, chamada btMaisMenos. Nosso código ficará como abaixo:

```
btMaisMenos(){
  if ((this.valor != "0") && (this.valor != "0.")) {
    if (this.valor.indexOf("-") == -1) {
      this.valor="-"+this.valor;
    } else {
      this.valor=this.valor.substr(1);
    }
  }
```

Obs.: ambas funções criadas, necessitam ser associadas aos botões na interface (Código HTML).

Videoaula 4

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 4. Disponível em: https://youtu.be/Ljo3Txz2zl4. Acesso em: 25 abr. 2021.

Videoaula 4

Utilize o QRcode para assistir!

Agora, assista à videoaula que abordará a criação do projeto de Calculadora – Parte 4.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 5

Para continuarmos o nosso projeto de calculadora, vamos implementar o acionamento dos botões de cálculo. Para isso, faremos a criação da função btCalculo no diretório home.page.ts. O valor informado, deve ser armazenado dentro da memória para que a operação selecionada possa ser realizada.

```
btCalculo(op){
  if (this.operacao != "") {
    this.calcula();
  }
  this.memoria=this.valor;
  this.operacao=op;
  this.valor="0";
}
```

Em seguida, faremos a criação da função calcula, que tem por objetivo verificar qual operação estamos realizando e efetuar o cálculo de fato.

```
calcula(){
  if (this.operacao != "") {
    var aux=0;
    if (this.operacao == "/") {
      if (this.valor != "0") {
        aux=eval(this.memoria+this.operacao+this.valor);
      } else {
        return;
      }
    } else {
      aux=eval(this.memoria+this.operacao+this.valor);
    }
    this.valor=aux.toString();
    this.memoria="0";
    this.operacao="";
  }      You, seconds ago • Uncommitted changes
}
```

Obs.: ambas funções criadas, necessitam ser associadas aos botões na interface (Código HTML).

Videoaula 5

Agora, assista esta videoaula que aborda a criação do projeto de Calculadora – Parte 5. Disponível em: https://youtu.be/LZVE3wdr6S0. Acesso em: 25 abr. 2021.

Videoaula 5

Utilize o QRcode para assistir!

Agora, assista esta videoaula que aborda a criação do projeto de Calculadora – Parte 5.

Desenvolvimento de aplicativo IONIC (Projeto Calculadora) – Parte 6

Nesta etapa, finalizaremos o nosso projeto de calculadora, faremos alguns ajustes e melhoraremos sua aparência. Para isso, existem alguns aspectos que podem melhorar o comportamento da nossa calculadora. Ex.: quando uma operação está em andamento, o

botão de resultado deve ficar desabilitado. O primeiro passo é verificar as documentações do ionic – button, nele temos o atributo *disabled*. Desta forma, vamos criar uma função desabilitaIgual.

```
desabilitaIgual(){
  if (this.operacao == "") return true;
  else return false;
}
```

Em seguida, incluiremos o resultado da operação na nossa interface.

Videoaula 6

Agora, assista à videoaula da parte 6 da criação do projeto de Calculadora. Disponível em: https://youtu.be/OTKGQuGJgek. Acesso em: 25 abr. 2021.

Videoaula 6

Utilize o QRcode para assistir!

Agora, assista à videoaula da parte 6 da criação do projeto de Calculadora.

Indicação de Leitura

CURSO DE HTML5 da *https://www.w3c.br/.* Os cursos de HTML5 do W3C Brasil visam apresentar as novidades que vêm sendo discutidas no WG HTML5 e que estão sendo implementadas na nova versão. Veja mais no *link* abaixo:

Disponível em: https://www.w3c.br/Cursos/CursoHTML5. Acesso em: 25 abr. 2021.

Curiosidades

O JavaScript foi criado na década de 90 por Brendan Eich, à serviço da Netscape. Essa década foi um período de revolução, pois os *browsers* ainda eram estáticos. Veja mais informações abaixo:

Disponível em: https://pt.wikipedia.org/wiki/JavaScript, https://pt.wikipedia.org/wiki/Hist%C3%B3ria_das_linguagens_de_programa%C3%A7%C3%A3o. Acesso em: 25 abr. 2021.

Disponível em: https://pt.wikipedia.org/wiki/ECMAScript. Acesso em: 25 abr. 2021.

Aula 02 – Desenvolvimento de aplicativo com IONIC PHP e criação de um banco de dados

Neste novo projeto, criaremos uma lista de compras, na qual vamos armazenar em memória.

IONIC (Lista Memória) – Parte 1

O primeiro passo é criarmos um projeto para colocar a nossa lista de compras para funcionar. Depois precisaremos acessar o terminal (terminal > *new terminal*) e em seguida criarmos o novo projeto de lista de compras, com o comando ionic star lista1.

```
adail@Adail:~/tads$ ionic start lista1
```

Novamente, nosso projeto será baseado em Angular e também usaremos o **blank** sendo o *template* do projeto.

```
? Framework: (Use arrow keys)
❯ Angular | https://angular.io
  React   | https://reactjs.org
  Vue     | https://vuejs.org
```

```
blank       | A blank starter project
list        | A starting project with a list
my-first-app | An example application that builds a camera with gallery
conference  | A kitchen-sink application that shows off all Ionic has to
```

O nosso projeto foi criado e novamente acessaremos a pasta.

```
adail@Adail:~/tads$ cd lista1
adail@Adail:~/tads/lista1$ ls -la
```

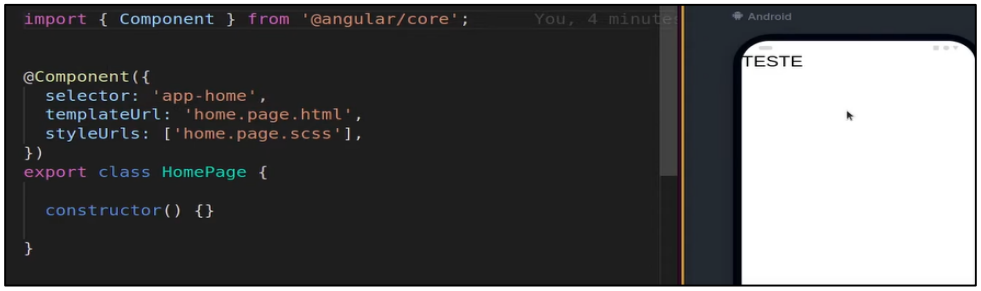
Em seguida, devemos iniciar o Visual Studio Code.

```
adail@Adail:~/tads/lista1$ code .
```

Logo adiante, iniciar o nosso projeto para compilação.

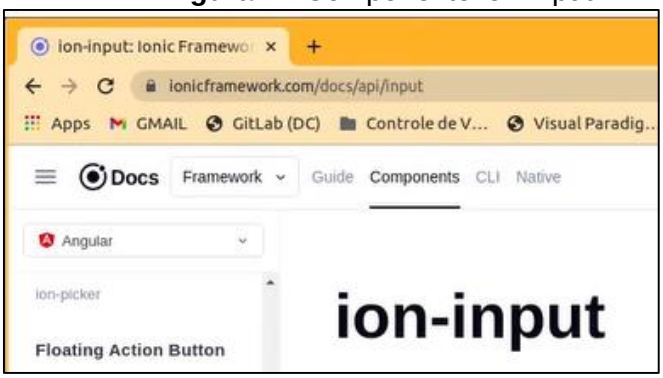
```
adail@Adail:~/tads/lista1$ ionic serve --lab
```

Do mesmo modo que nos projetos anteriores, limparemos o código, conforme a imagem abaixo:



Verificaremos na documentação do ionic o compomente ion-input.

Figura 1 - Componente ion-input



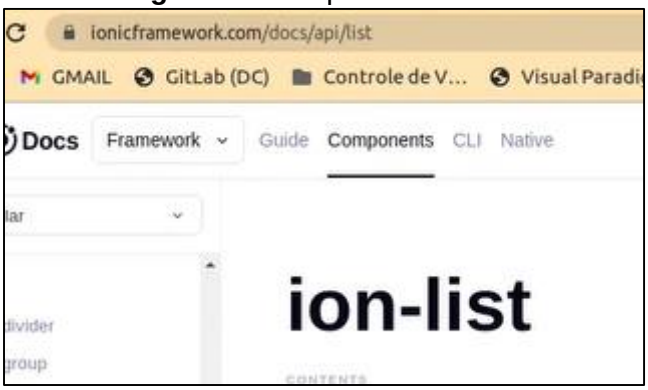
Fonte: disponível em: ionicframework.com/docs/api/input. Acesso em: 25 abr. 2021.

Vamos copiar o evento do *input* e copiar em nosso código html.

```
<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col>
        ion-col
      </ion-col>
      <ion-col>
        ion-col
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

Logo, vamos ajustar nosso código HTML para adequação da aplicação na página. Verificaremos na documentação do ionic o componente ion-list.

Figura 2 - Componente ion-list



Fonte: disponível em: ionicframework.com/docs/api/list. Acesso em: 25 abr. 2021.

Após alguns ajustes da página, inseriremos ícones e grades para adequação das informações em nosso projeto de lista de compras. Desta forma, estamos próximos do comportamento esperado (protótipo) da lista de compras.

Videoaula 1

Nesta videoaula abordaremos a criação do projeto de Lista de compras (Memóra) – Parte 1. Disponível em: https://youtu.be/5zkmtI63DoI. Acesso em: 26 abr. 2021.

Videoaula 1

Utilize o QRcode para assistir!

Nesta videoaula abordaremos a criação do projeto de Lista de compras (Memóra) – Parte 1.

IONIC (Lista Memória) – Parte 2

Dando continuidade ao nosso aplicativo de lista de compras, no qual aprenderemos como armazenar os dados informados na memória do dispositivo. O primeiro passo é fazermos com que a lista seja armazenada em um objeto de listas. Na pasta home.page.ts incluiremos o objeto minhaLista.

```
milhaLista= ["Batata","Limão","Cebola"];
```

Desta forma, dentro do Angular, existe um comando que pode ser vinculado uma condição IF.

```
    </ion-grid>      You, seconds ago • Uncommitted ch
    <ion-list>
      <ion-item *ngFor="let item of milhaLista" >
        <ion-label>{{item}}</ion-label>
        <ion-button color="danger" size="small">
          <ion-icon name="trash-outline"></ion-icon>
        </ion-button>
      </ion-item>
    </ion-list>
  </div>
</ion-content>
```

Em seguida, faremos uma nova função para incluir os itens em nossa lista. Ou seja, necessitamos realizar a criação desta função, para ligação entre a variável e botão de inclusão e remoção. O ponto de ligação se dá pela criação do código abaixo:

```
<ion-input [(ngModel)]="texto" placeholder="
```

O ponto de ligação entre o botão e a variável criada acontece conforme o código abaixo:

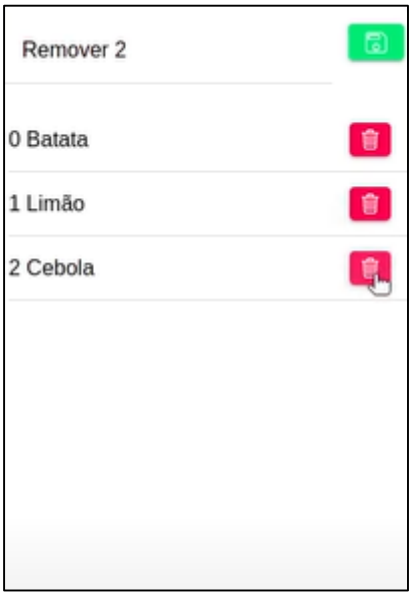
```
<ion-button (click)="adiciona()"  size="small" color="success">
```

A criação da função adiciona e remove resulta no código abaixo:

```
adiciona(){
  this.texto="Inclir";
}

remove(indice){
  this.texto="Remover "+indice;
}
```

Desta forma, nossa aplicação receberá um índice em torno de cada item inserido e removido.



Videoaula 2

Nesta videoaula veja a criação do projeto de Lista de compras (Memória) – Parte 2. Disponível em: https://youtu.be/j_jnwlZnwZM. Acesso em: 25 abr. 2021.

Videoaula 2

Utilize o QRcode para assistir!

Nesta videoaula veja a criação do projeto de Lista de compras (Memória) – Parte 2.

IONIC (Lista Memória) – Parte 3

Nesta etapa, daremos continuidade ao nosso aplicativo de lista de compras e com o armazenamento dos dados informados na memória do dispositivo. Desta forma, nossa aplicação necessita retornar o item que está sendo removido e adicioná-lo na tela.

Ainda na pasta home.page.ts, alteraremos a função adiciona com o ajuste de um push.

```
adiciona(){
  this.milhaLista.push(this.texto);
  this.texto="";
}
```

Ou seja, o primeiro passo de inclusão e tratamento do item, está concluído. O segundo passo, decorre no ajuste do botão remover a fim de retirarmos elementos de nossa lista.

```
remove(indice){
  this.milhaLista.splice(indice,1);
}
```

Logo, realizaremos uma tratativa em torno dos botões de teclado, pois a tratativa que realizamos até o momento se dava por clique do *mouse*.

```
<ion-content>
  <div id="cont">
    <ion-grid>
      <ion-row>
        <ion-col size="10">
          <ion-item>
            <ion-input
              #input
              [(ngModel)]="texto"
              (keyup.enter)="adiciona()"
              placeholder="Informe um item">
            </ion-input>
          </ion-item>
        </ion-col>
        <ion-col size="2">
          <ion-button (click)="adiciona()"  size="small" color="success">
```

Deste modo, é necessário referenciar os elementos e inserirmos o foco de determinado elemento.

```
@ViewChild('input') meuInput;

milhaLista= ["Batata","Limão","Cebola"];
texto="";

constructor() {}

adiciona(){
  this.milhaLista.push(this.texto);
  this.texto="";
  this.meuInput.setFocus();
}
```

Do mesmo modo, é necessário alterar o código na função remove.

```
remove(indice){
  this.milhaLista.splice(indice,1);
  this.meuInput.setFocus();
}
```

Desta forma, nossa aplicação ficará conforme a imagem abaixo:

Videoaula 3

Assista à videoaula com a parte 3 da criação do projeto de Lista de compras (Memória). Disponível em: https://youtu.be/EW00QJOLQeE. Acesso em: 25 abr. 2021.

Videoaula 3

Utilize o QRcode para assistir!

Assista à videoaula com a parte 3 da criação do projeto de Lista de compras (Memória).

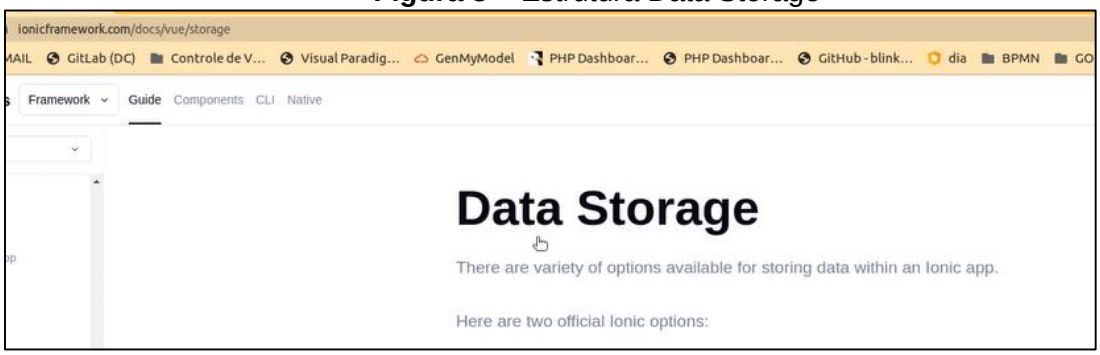
IONIC (Lista Storage) – Parte 1

Nesta etapa, faremos a criação de um método para armazenamento no próprio dispositivo. Realizaremos uma busca dentro da documentação do IONIC.

ionicframework.com/docs

Usaremos a estrutura do Data Storage.

Figura 3 – Estrutura Data Storage



Fonte: disponível em: ionicframework.com/docs/vue/storage. Acesso em: 25 abr. 2021.

O primeiro passo é instalarmos o recurso do Data Storage. Em seguida, dentro do nosso terminal, onde a aplicação estará executando, realizaremos o comando abaixo:

```
adail@Adail:~/tads/lista1$ npm install @ionic/storage-angular
```

Assim que terminar a instalação, utilizaremos a tratativa de definição de Angular da documentação, dentro do nosso código.

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

import { IonicStorageModule } from '@ionic/storage-angular';

You, seconds ago | 1 author (You)
@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [
    BrowserModule,
    IonicModule.forRoot(),
    IonicStorageModule.forRoot(),      You, seconds ago • Unco
    AppRoutingModule],
```

Ainda em nosso código, é necessário definir onde nossos dados serão armazenados no Storage.

```
IonicStorageModule.forRoot({
  name:"menuBanco",
  driverOrder:[ Drivers.IndexedDB, Drivers.LocalStorage]
}),
```

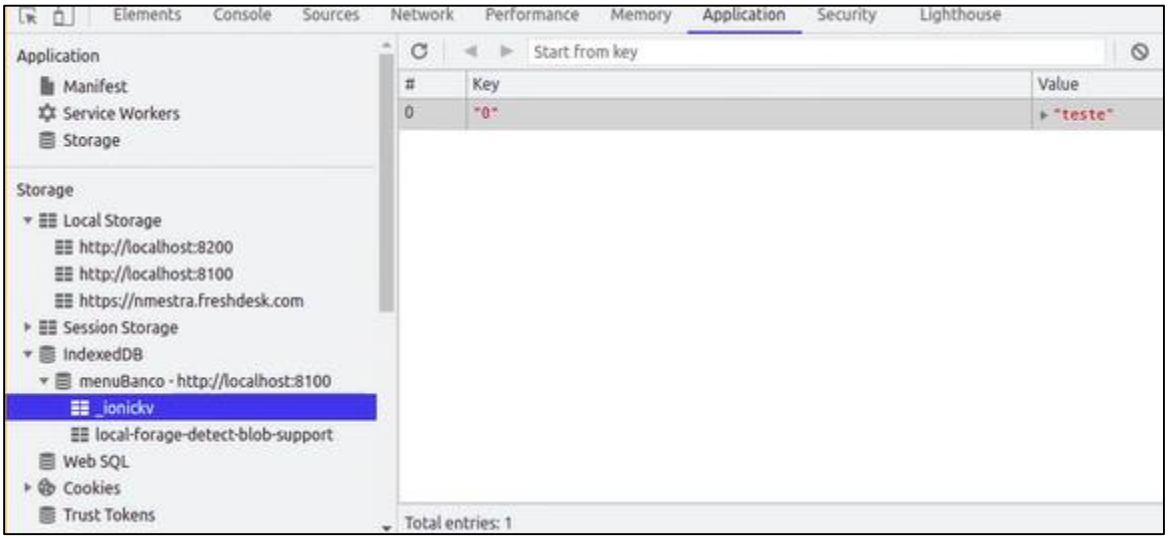
Logo, na estrutura do nosso código, é necessário ajustar com a criação de um construtor.

```
constructor(private storage:Storage) {
  this.storage.create();      You, seco
}
```

Por hora, vamos iniciar nossa aplicação novamente.

```
adail@Adail:~/tads/lista1$ ionic serve --lab
```

Logo, em nosso teste de aplicação, foi armazenado no banco de dados criado, o elemento inserido.



Videoaula 4

Nesta videoaula veremos a criação do projeto de Lista de compras (Storage) – Parte 1. Disponível em: https://youtu.be/q-PwQjz8psI. Acesso em: 26 abr. 2021.

Videoaula 4

Utilize o QRcode para assistir!

Nesta videoaula veremos a criação do projeto de Lista de compras (Storage) – Parte 1.

IONIC (Lista Storage) – Parte 2

Nesta etapa da criação, o armazenamento é feito por chave e valor. Desta forma, faremos um ajuste no código para que tenhamos um vetor de chave e valor.

```
milhaLista= [["1","Batata"],["2","Limão"],["3","Cebola"]];
texto="";
```

```
      </ion-col>
      <ion-col size="2">
        <ion-button (click)="adiciona()"  size="small" color="success">
          <ion-icon name="save-outline"></ion-icon>
        </ion-button>
      </ion-col>
    </ion-row>
  </ion-grid>
  <ion-list>
    <ion-item *ngFor="let item of milhaLista; let i=index" >
      <ion-label>{{item}}</ion-label>
      <ion-button (click)="remove(i)" color="danger" size="small">
        <ion-icon name="trash-outline"></ion-icon>
      </ion-button>
    </ion-item>
  </ion-list>
</div>
```

Em seguida, faremos a tratativa onde cada item inserido ou removido também seja alterado na memória.

Faremos algumas alterações nos códigos para melhor tratativa.

```
var a=0;
this.milhaLista.forEach(item => {
  if( parseInt(item[0]) > a) {
    a=parseInt(item[0]);
  }
})
// soma um no maior codigo
a=a+1;
// armazena no banco o item usando o condigo mais um
this.storage.set(a.toString(),this.texto);

this.milhaLista.push([a.toString(),this.texto]);
this.texto="";
this.meuInput.setFocus();

}
```

Logo, se faz necessário criar uma função atualizaLista.

```
atualizaLista(){
  this.milhaLista=[];
  this.storage.forEach( (value, key, index) => {
    this.milhaLista.push([key,value]);
  } )
}
```

Em sequência, incluiremos o método Async, para atualizar a lista assim que gravado na memória.

```
async adiciona(){
  /// procura o maior codigo
  var a=0;
  this.milhaLista.forEach(item => {
    if( parseInt(item[0]) > a) {
      a=parseInt(item[0]);
    }
  })
  // soma um no maior codigo
  a=a+1;
  // armazena no banco o item usando o condigo mais um
  await this.storage.set(a.toString(),this.texto);

  this.atualizaLista();

  this.texto="";
  this.meuInput.setFocus();
```

Para ajustar o método remove e em sequência atualizar a lista e no banco, ajustamos também o método remove.

```
async remove(indice){
  await this.storage.remove(indice);
  this.atualizaLista();
  this.meuInput.setFocus();
}
```

Por agora, podemos parar o servidor ou até mesmo fechar a aplicação, que os dados pré-informados serão armazenados na memória e no banco.

Este tipo de armazenamento decorre da utilização do método Index add.

Podemos ainda, utilizar alguns componentes do IONIC para melhorar a interface de Scroll ou inserir esta tratativa diretamente no código CSS.

```
ion-list {
  overflow: auto;
  max-height: 500px;
}
```

Videoaula 5

Agora, assista esta videoaula que aborda a criação do projeto de Lista de compras (Storage) – Parte 2 (Final). Disponível em: https://youtu.be/ronv-gMhuiA. Acesso em: 26 abr. 2021.

Videoaula 5

Utilize o QRcode para assistir!

Agora, assista esta videoaula que aborda a criação do projeto de Lista de compras (Storage) – Parte 2 (Final).

IONIC Instalação – Maria DB e PHP

Agora iremos começar o processo de evolução da nossa aplicação de Lista de Compras. Por enquanto, iremos evoluir com algumas tratativas e com a instalação de ferramentas adicionais.

Iremos utilizar o banco de dados relacional chamado **MariaDB**. A ferramenta a qual iremos instalar também irá realizar o acesso ao banco de dados, se chama DBEAVER. Esta ferramenta tem como característica atender a vários tipos de banco de dados.

Por fim, o terceiro elemento que é a linguagem utilizada para realizar o acesso ao banco de dados relacional é o PHP. Esta linguagem será importante para construção da API de serviços ao banco de dados.

Para instalação primeiramente é necessário a inicialização de um servidor *web*. Iremos acessar o endereço abaixo para realizar o *download*.

httpd.apache.org

Para facilitar a vida do desenvolvedor, utilizaremos a ferramenta XAMPP, que permitirá realizar a instalação com maior facilidade.

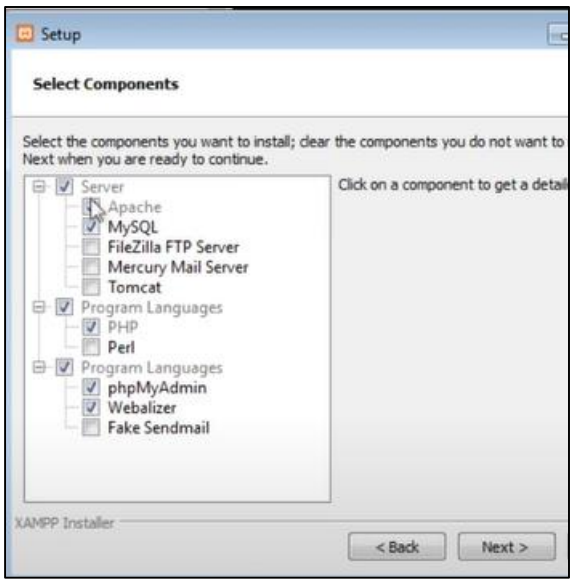
A maior parte das documentações do nosso banco de dados estão localizadas no endereço: https://mariadb.com/kb/pt-br/documentacao-mariadb/

Figura 4 – Documentação do MariaDB

Fonte: disponível em: https://mariadb.com/kb/pt-br/documentacao-mariadb/. Acesso em: 26 abr. 2021.

Em sequência, faremos a instalação do XAMPP.

Em um primeiro momento, iremos demarcar os que não vamos utilizar. A imagem abaixo representa como será o padrão da instalação desta ferramenta.

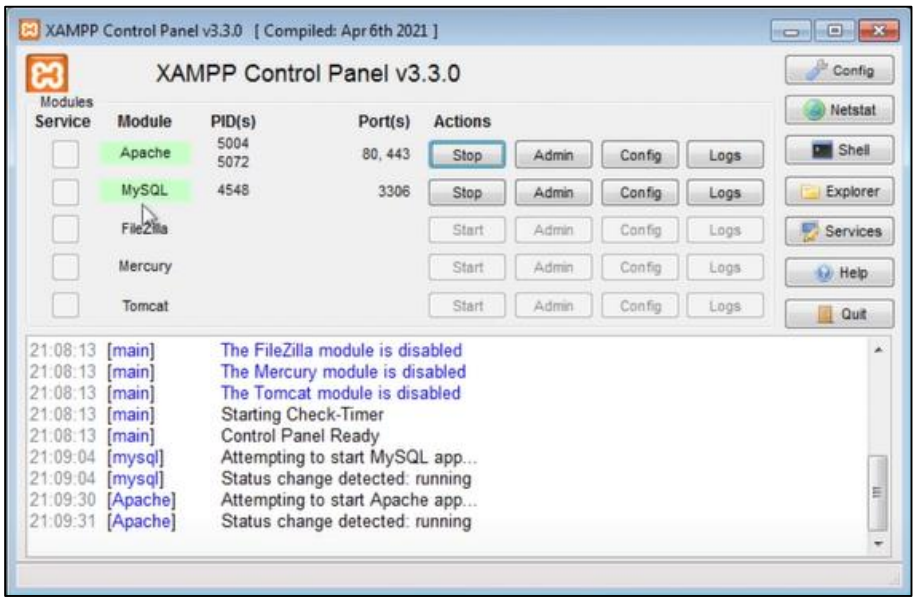


Assim que instalado, alguns serviços serão iniciados na máquina.

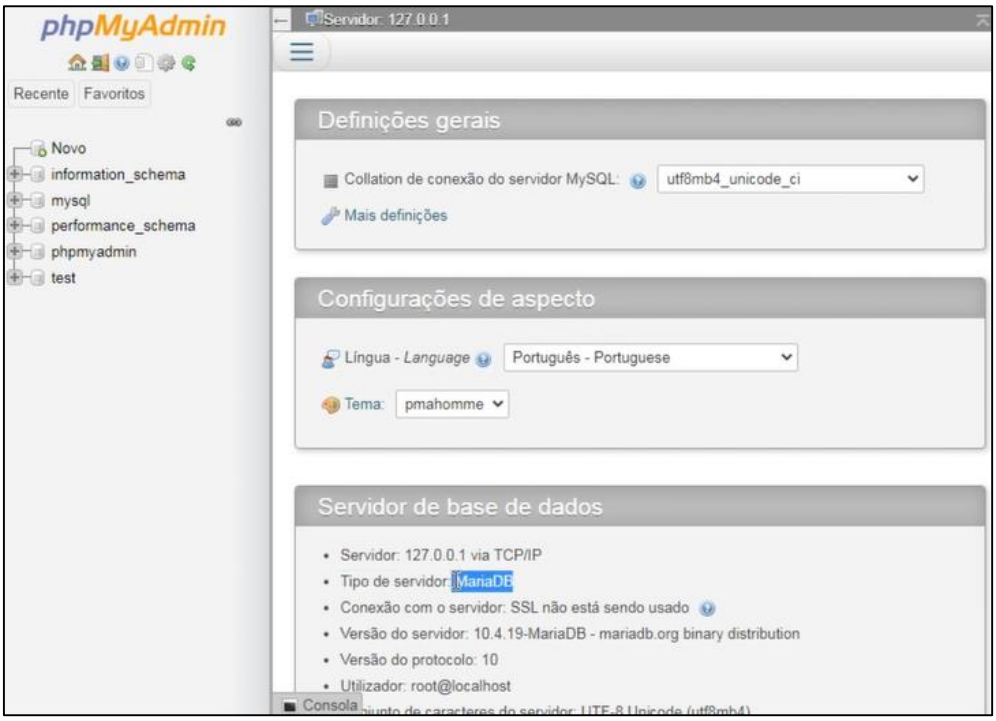
Enquanto a ferramenta é instalada, podemos acessar o endereço:

E fazer o *download* da ferramenta de interação.

Em seguida, assim que finalizar a instalação, devemos inicializar o Apache e MySQL.



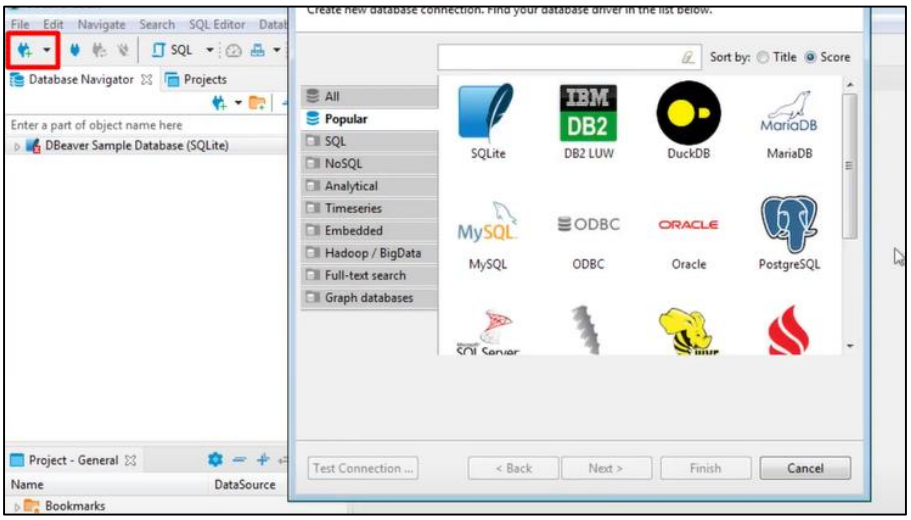
Logo, a aplicação fornece uma interface admin a qual visualizaremos que a instalação do banco MariaDB foi feita com sucesso.

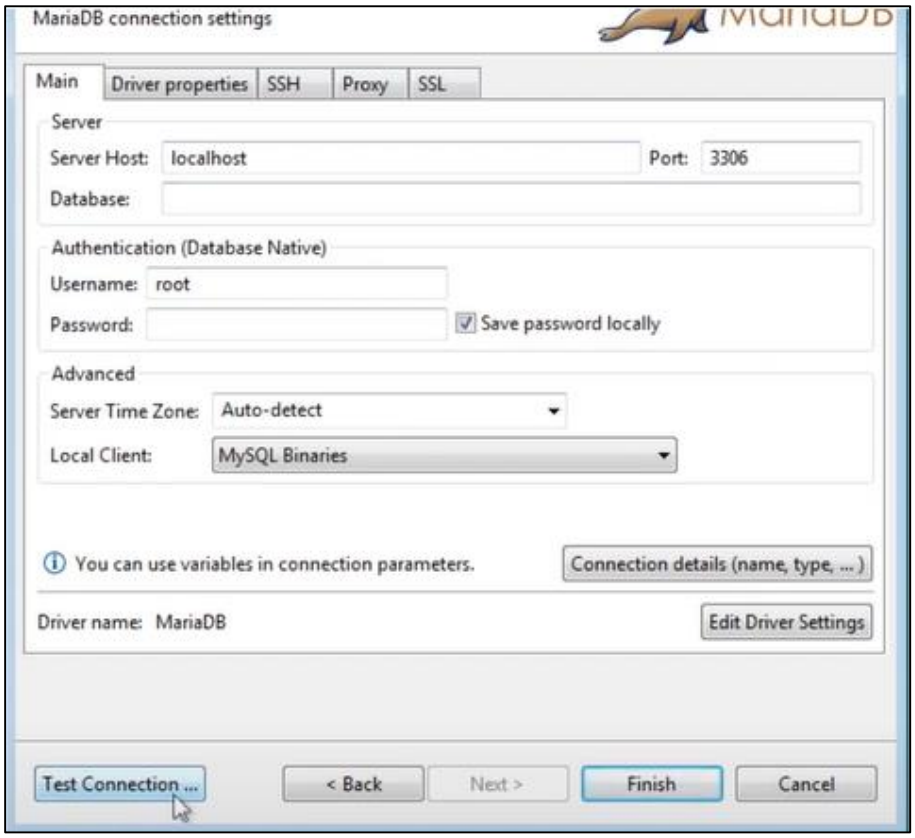


Em seguida, podemos inicializar a instalação do dbeaver.

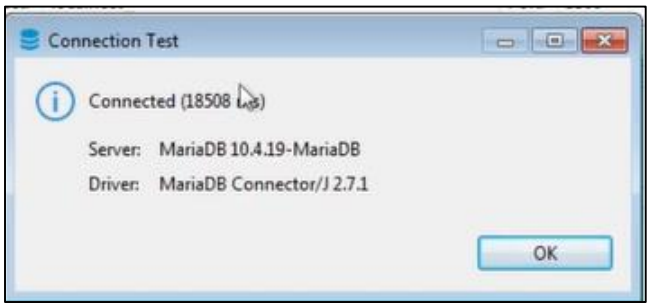
A instalação desta ferramenta se dá por forma padrão.

Assim que terminada a instalação, devemos abrir a ferramenta dbeaver para criarmos uma nova conexão com o banco de dados.



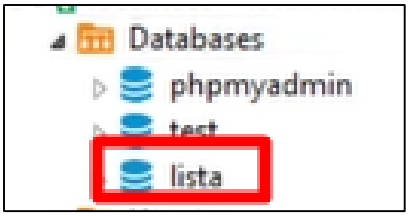


Por hora, será solicitado o *download* do driver e por fim podemos conectar no banco.



As ferramentas phpmyadmin e DBeaver permitem que criamos uma nova base de dados.

Vamos criar pela ferramenta DBeaver uma nova base de dados, chamada lista.



Videoaula 6

Agora, assista esta videoaula que aborda a instalação do banco de dados MariaDB e a ferramenta PHP. Disponível em: https://youtu.be/n2BDeO4XG9c. Acesso em: 26 abr. 2021.

Videoaula 6

Utilize o QRcode para assistir!

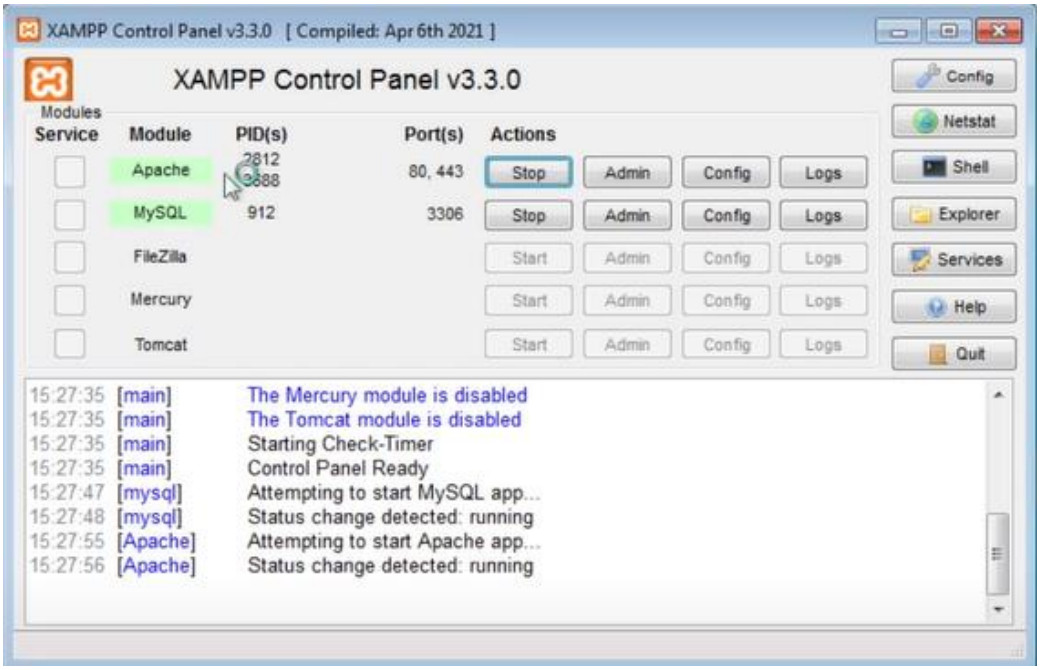
Agora, assista esta videoaula que aborda a instalação do banco de dados MariaDB e a ferramenta PHP.

IONIC Criação do Banco de Dados

Nesta parte da aula veremos como criar nossas tabelas para fazermos as tratativas do nosso aplicativo, **lista de compras**.

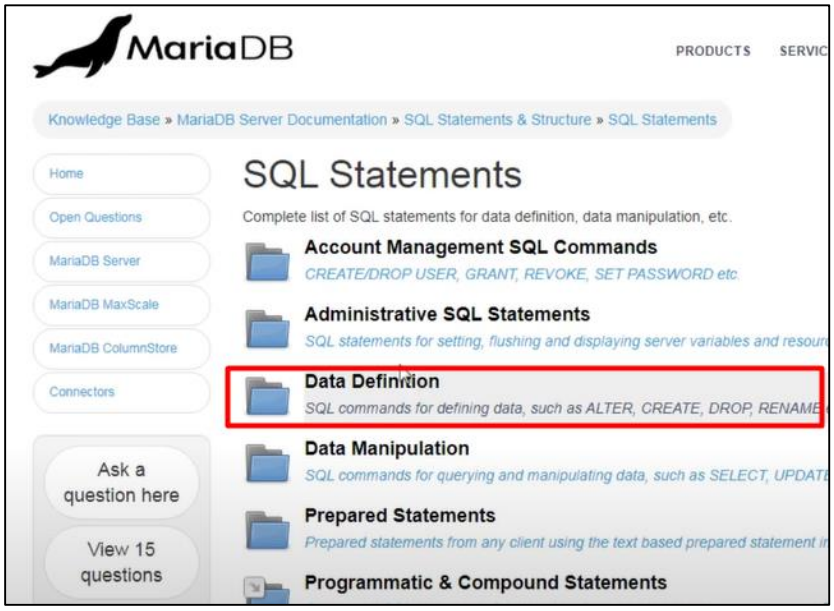
Para isso, vamos precisar aprender alguns comandos básicos da linguagem SQL.

Obs.: antes de iniciarmos é necessário ativar os serviços do XAMPP.



As documentações do banco de dados Maria DB estão disponíveis em:

Utilizaremos os comandos de Definição de dados.



Em seu *browser*, busque o endereço:

Em seguida, clicaremos em SQL.

Em um primeiro momento, utilizaremos o comando CREATE (verificar documentação).

Execute o comando abaixo e em seguida clique no botão executar.

```
1 CREATE DATABASE basenova;
```

Em seguida, podemos executar o comando abaixo para apagar a base de dados.

```
DROP DATABASE IF EXISTS basenova;
```

Até então, fizemos apenas alguns testes para demonstrar os comandos.

Por hora, vamos executar o comando para criar a base de dados.

```
create DATABASE lista;
```

Em seguida, vamos criar uma tabela, utilizando o comando create table.

```
CREATE TABLE lista.compras ( codigo bigint, descricao varchar(200) );
```

Na sequência, vamos inserir o registro em uma tabela, utilizando o comando INSERT (*Data Manipulation* – Documentação Maria DB). Exemplo:

```
INSERT INTO `compras` (`codigo`, `descricao`) VALUES ('1', 'Tomate');
```

Em sequência, faremos a inserção do código 2 para Cebola.

Abaixo está um exemplo de inserção (*Insert*) dos registros na tabela e também para consultar dados (*Select*).

```
SELECT * FROM lista.compras ;

insert into lista.compras (codigo,descricao)
values
(4,'Abacate'),
(5,'Batata'),
(6,'Laranja');
```

Caso precisemos alterar algum dado no banco de dados, utilizaremos o comando (*update*).

```
update lista.compras set descricao="Abacate Verde"
where
codigo = 4
```

Caso seja necessário deletar algum dado, utilizaremos o comando (*delete*).

```
delete from lista.compras
where codigo = 5;
```

Videoaula 7

Nesta videoaula abordaremos a criação do banco de dados MariaDB. Disponível em: https://youtu.be/90XpD3LxjHM. Acesso em: 16 abr. 2021.

Videoaula 7

Utilize o QRcode para assistir!

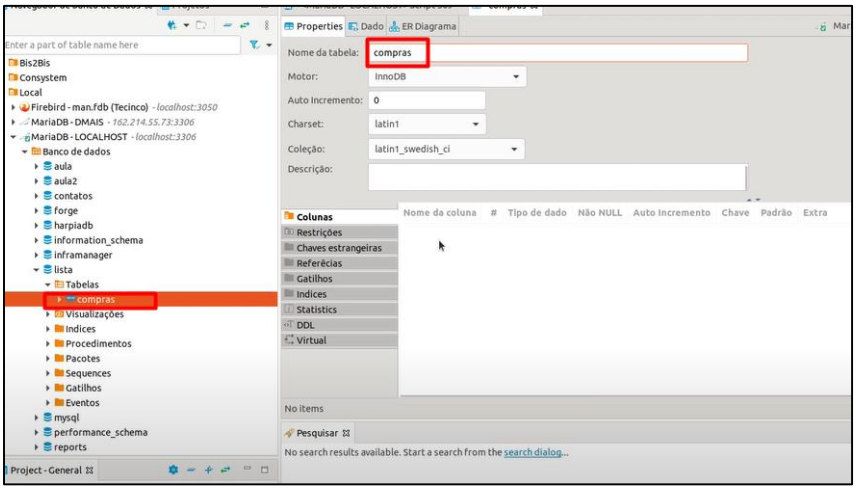
Nesta videoaula abordaremos a criação do banco de dados MariaDB.

IONIC Conectando no MariaDB

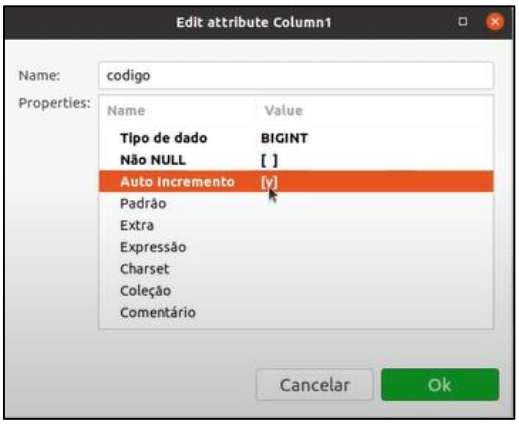
Nesta etapa da aula, daremos andamento no nosso projeto e para isso vamos construir uma API em PHP. Vamos abrir o executor de código SQL e executar o comando para criar uma database.

```
create database lista;
```

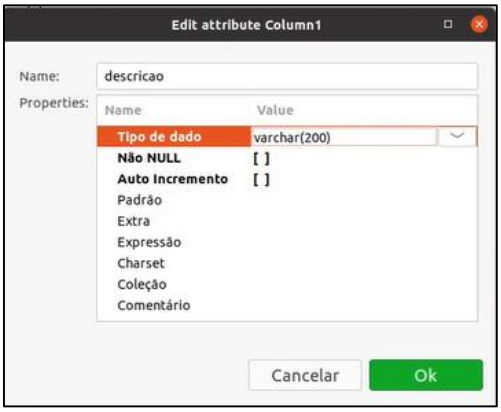
Em seguida, vamos criar uma tabela.



Logo após, vamos criar uma coluna chamada código. Tal recurso fornecido pelo MariaDB permite a inserção de um campo Auto Incremento.



E também a coluna descrição.



Basicamente, temos o seguinte código para ser executado.

```
create database lista;

CREATE TABLE lista.compras (
    codigo BIGINT auto_increment NOT NULL,
    descricao varchar(200) NULL,

);
```

Logo, é necessário verificar nas documentações do Maria DB, como se da para criar uma chave primária. Na documentação da linguagem, temos os comandos de *Data Definition* para *create* de uma tabela.

Em seguida, após a alteração o código será como a imagem abaixo:

```
create database lista;

CREATE TABLE lista.compras (
    codigo BIGINT primary key auto_increment NOT NULL,
    descricao varchar(200) NULL
);
```

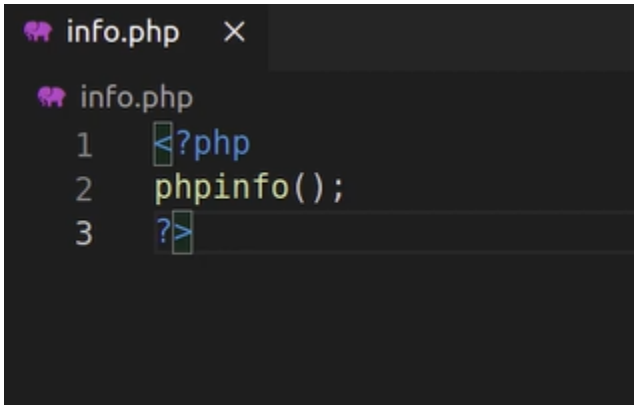
Dentro do Linux, vamos criar uma subpasta para nossa aplicação, com o comando:

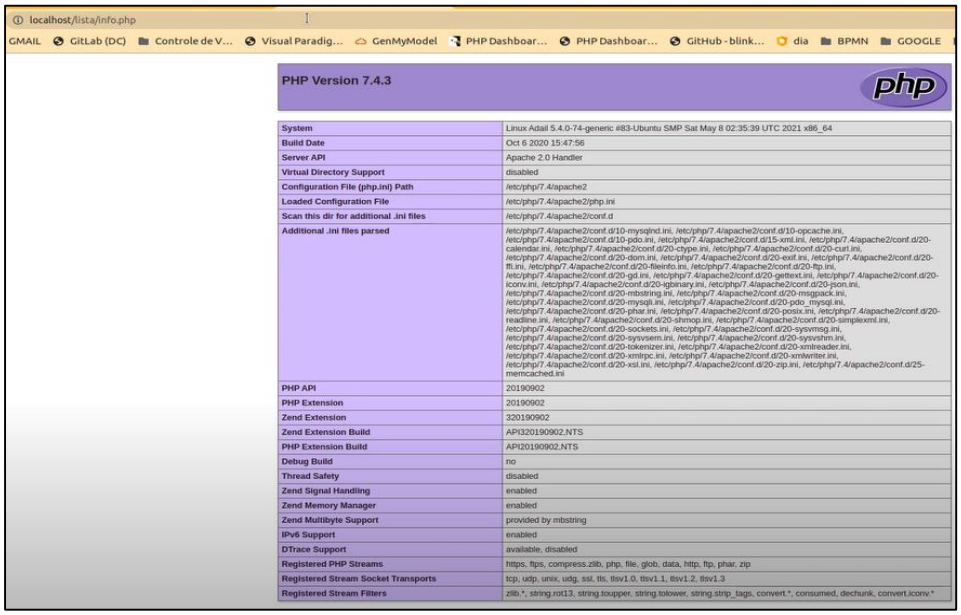
```
adail@Adail:/var/www/html$ mkdir lista
```

Em seguida, vamos abrir o Visual Studio Code dentro desta pasta.

```
adail@Adail:/var/www/html/lista$ code .
```

Vamos criar um arquivo PHP e em seguida acessá-lo.





Dentro do nosso código vamos realizar a inserção do código para inicializarmos a conexão com o banco.

```php
<?php
$servidor="localhost;port=3306";
$banco="lista";
$usuario="root";
$senha="root";
$strcon="mysql:host=".$servidor.";dbname=".$banco.";charset=utf8";
$con =new PDO($strcon,$usuario,$senha);

?>
```

Em seguida, faremos uma alteração em nosso código.

```php
$sql = "SELECT * FROM compras";

$dados=$con->query($sql);

foreach($dados as $seq => $valor ) {
    echo("<p>");
    print_r($valor[codigo] );
    echo(" - ");
    print_r($valor[descricao] );
    echo("</p>");
}
```

Videoaula 8

Agora, assista à videoaula que abordará sobre a conexão no MariaDB. Disponível em: https://youtu.be/XvdhIl-9wQk. Acesso em: 25 abr. 2021.

Videoaula 8

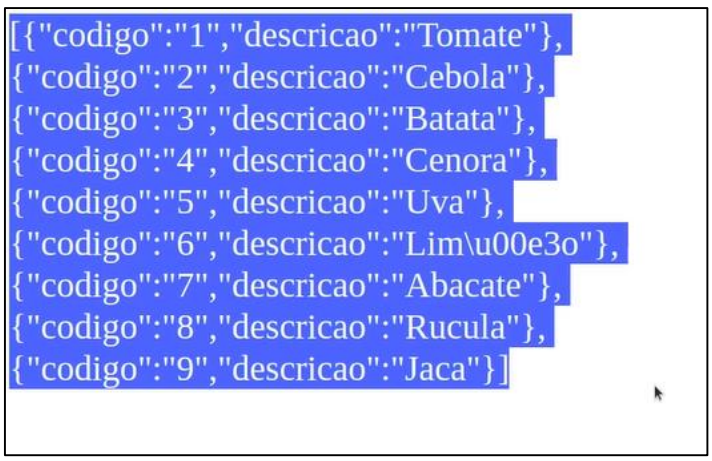
Utilize o QRcode para assistir!

Agora, assista à videoaula que abordará sobre a conexão no MariaDB.

IONIC PHP Leitura via API

Agora faremos a leitura de uma API para conectar na nossa lista de produtos. Primeiramente vamos acessar o endereço abaixo:

O resultado será emitido na tela:



Vamos realizar algumas alterações no código para ligação da API com a nossa aplicação. Em seguida, vamos abrir o Visual Studio Code dentro da pasta na qual a nossa aplicação esta alocada.

Assim que iniciado o nosso projeto, vamos retirar a conexão com o banco de dados local e utilizar a API.

Primeiramente, devemos incluir o comando *import,* abaixo. No qual estamos incluindo a possibilidade de trabalhar com HTTP dentro da aplicação.

```
import { HttpClientModule } from '@angular/common/http'
```

No método construtor, vamos realizar as alterações provindas para a utilização de HTTP.

```
constructor(private http:HttpClient) {
  // private storage:Storage,
  this.storage.create( );
}
```

```
a=a+1;

// armazena no banco o item usando o condigo mais
// await this.storage.set(a.toString(),this.texto)

this.atualizaLista();

this.texto="";
this.meuInput.setFocus();
```

```
atualizaLista(){
  this.milhaLista=[];
  /*
  this.storage.forEach( (value, key, index) => {
    this.milhaLista.push([key,value]);
  } )
  */
}      You, seconds ago • Uncommitted changes
```

```
async remove(indice){
  //await this.storage.remove(indice);
  this.atualizaLista();
  this.meuInput.setFocus();
}
```

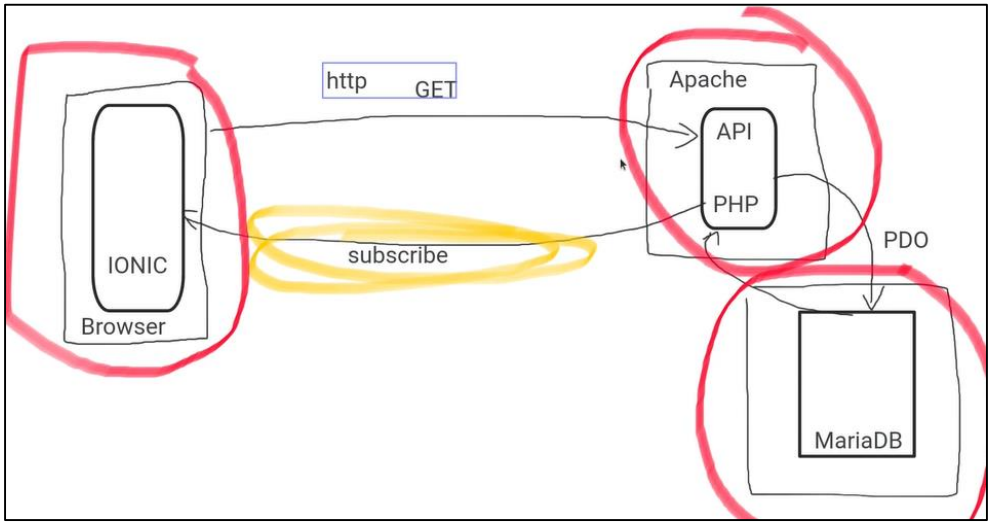
Na sequência, vamos incluir uma tratativa de chamada.

```
atualizaLista(){
  this.milhaLista=[];
  this.http.get("http://localhost/lista/consulta.php")
  /*
  this.storage.forEach( (value, key, index) => {
    this.milhaLista.push([key,value]);
  } )
  */
}
```

```
atualizaLista(){
  this.milhaLista=[];
  this.http.get<any[]>("http://localhost/lista/consulta.php")
           .subscribe( valor => {
             valor.forEach( dados => {
              this.milhaLista.push([dados.codigo,dados.descricao]);
             })
           })
  /*
  this.storage.forEach( (value, key, index) => {      You, seconds ago
    this.milhaLista.push([key,value]);
  } )
  */
}

async remove(indice){
  //await this.storage.remove(indice);
  this.atualizaLista();
  this.meuInput.setFocus();
```

De modo simplificado, abaixo está um fluxo no qual estamos realizando a nossa aplicação.



Videoaula 9

Agora, assista esta videoaula que aborda a conexão IONIC PHP via API. Disponível em: https://youtu.be/6Ndvgg5rvAE. Acesso em: 26 abr. 2021.

Videoaula 9

Utilize o QRcode para assistir!

Agora, assista esta videoaula que aborda a conexão IONIC PHP via API.

IONIC PHP Gravação e Remoção

Vamos agora continuar a construção da nossa aplicação, inserindo o tratamento para que a aplicação funcione de forma *on-line*, acessando através de uma API o nosso banco de dados MariaDB.

Em nosso código, vamos incluir uma tratativa para visualizar melhor o conteúdo.

```php
<?php
$servidor="localhost;port=3306";
$banco="lista";
$usuario="root";
$senha="root";
$strcon="mysql:host=".$servidor.";dbname=".$banco.";charset=ut
$con =new PDO($strcon,$usuario,$senha);

$sql = "SELECT * FROM compras";

$dados=$con->query($sql);

$obj = $dados->fetchAll(PDO::FETCH_ASSOC);
$resultado = json_encode($obj);

header("Content-Type: application/json");
header("Access-Control-Allow-Origin:*");

print_r($resultado);

?>
```

Abaixo, apresentamos apenas um detalhamento a respeito dos métodos de requisição.

POST

O método `POST` é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.

PUT

O método `PUT` substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.

DELETE

O método `DELETE` remove um recurso específico.

Realizaremos alguns ajustes em nossos métodos.

```php
<?php
$servidor="localhost;port=3306";
$banco="lista";
$usuario="root";
$senha="root";
$strcon="mysql:host=".$servidor.";dbname=".$banco.";charset=utf8";
header("Content-Type: application/json");
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST");

if ($_SERVER['REQUEST_METHOD']  == "POST" ) {

    $aux = file_get_contents('php://input');
    $valor = json_decode($aux);

    if (isset($valor['descricao']) === true) {
        $con =new PDO($strcon,$usuario,$senha);

        $sql = "INSERT INTO compras (descricao) VALUES (?)";
        $comando = $con->prepare($sql);

        $comando->execute([$valor['descricao']]);
```

Faremos ajustes em códigos, ajustando inserção e remoção, e também *select*.

```php
if ($_SERVER['REQUEST_METHOD']  == "POST" ) {

    $aux = file_get_contents('php://input');
    $valor = json_decode($aux);

    if (isset())
    $valor['descricao']

    $con =new PDO($strcon,$usuario,$senha);

    $sql = "SELECT * FROM compras";

    $dados=$con->query($sql);

    $obj = $dados->fetchAll(PDO::FETCH_ASSOC);
    $resultado = json_encode($obj);
```

```php
try {
    if ($_SERVER['REQUEST_METHOD']  == "DELETE" ) {

        $aux = file_get_contents('php://input');
        $valor = json_decode($aux,true);
        if (isset($valor['descricao']) === true) {
            $con =new PDO($strcon,$usuario,$senha);

            $sql = "DELETE FROM compras where codigo=?";
            $comando = $con->prepare($sql);

            $comando->execute([$valor['descricao']]);
            $codigo=$con->lastInsertId();

            http_response_code(201);
```

```php
    if (isset($valor['descricao']) === true) {
        $con =new PDO($strcon,$usuario,$senha);

        $sql = "INSERT INTO compras (descricao) VALUES (?)";
        $comando = $con->prepare($sql);

        $comando->execute([$valor['descricao']]);
        $codigo=$con->lastInsertId();

        http_response_code(201);
        echo('{"codigo":'.$codigo.'}');
    }

} else {
    http_response_code(404);
    echo('{"erro":"Nao foi chamado com POST"}');
```

Por fim, faremos o ajuste para requisição.

```php
        if (isset($valor['descricao']) === true) {
            $con =new PDO($strcon,$usuario,$senha);

            $sql = "INSERT INTO compras (descricao) VALUES (?)";
            $comando = $con->prepare($sql);

            $comando->execute([$valor['descricao']]);
            $codigo=$con->lastInsertId();

            http_response_code(201);
            echo('{"codigo":'.$codigo.'}');
        }

    } else {
        if ($_SERVER['REQUEST_METHOD'] == "OPTIONS") {
            http_response_code(200);
        } else {
            http_response_code(406);
            echo('{ "message2": "'.$_SERVER['REQUEST_METHOD'].'"}');
        }
    }
} catch (PDOException $e) {
    http_response_code(500);
```

Deste modo, nossa aplicação ficará como a imagem abaixo:

Videoaula 10

Agora, assista à videoaula que abordará a conexão do IONIC - Gravação e Remoção. Disponível em: https://youtu.be/z0FNSvRjB2A. Acesso em: 26 abr. 2021.

Videoaula 10

Utilize o QRcode para assistir!

Agora, assista à videoaula que abordará a conexão do IONIC - Gravação e Remoção.

Encerramento da Unidade

Considerando os pontos abordados nesta unidade, desde o desenvolvimento de aplicativos móveis utilizando o *Framework* Angular, onde desenvolvemos uma calculadora, até o aprofundamento com o desenvolvimento de IONIC, PHP, criando banco de dados local e também a tratativa para integração e conexão de uma API, destaco a importância e relevância para:

1. A criação de um banco de dados local, que pode ser utilizado como ambiente de testes para um desenvolvedor;
2. Consideramos que o aluno pode aprofundar o estudo nas linguagens HTML, PHP e CSS, assim como em integrações e conexões com API.

Agora que você completou esta unidade, você deve ser capaz de:

- Discutir a respeito de desenvolvimento de aplicativos móveis, utilizando IONIC, assim como conhecimento nas bibliotecas envolvendo componentes e diversos complementos para melhoria de interface das aplicações;
- Discutir a respeito de desenvolvimento de aplicativos móveis com PHP, utilizando um banco de dados local;
- Discutir a respeito de desenvolvimento de aplicativos móveis com PHP utilizando a conexão e integração de uma API.

Referências

FÉLIX, Rafael; SILVA, Everaldo Leme. **Arquitetura para computação móvel.** 2. ed. São Paulo: Pearson Education do Brasil, 2019.

FLATSCHART, Fábio. **HTML 5**: Embarque Imediato. Rio de Janeiro: Editora Brasport, 2011.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações móveis**: arquitetura, projeto e desenvolvimento. São Paulo: Pearson Makron Books, 2005.

MARINHO, Antonio Lopes (org.). **Desenvolvimento de aplicações para Internet**. São Paulo: Pearson Education do Brasil, 2016.

SEGURADO, Valquiria Santos (org.). **Projeto de interface com o usuário**. 1. ed. São Paulo: Pearson Education do Brasil, 2015.

SILVA, Diego (org). Desenvolvimento para dispositivos móveis. São Paulo: Pearson Education do Brasil, 2016.

SOMMERVILLE, Ian. **Engenharia de software**. Tradução Luiz Claudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson, 2018. 768 p.

Publicações da SBC (http://www.sbc.org.br/index.php?Itemid=196)

Portal Periódicos. CAPES (http://www.periodicos.capes.gov.br)

Google Acadêmico (http://scholar.google.com.br)

SciELO (http://www.scielo.br)

arXiv.og (http://arxiv.org)

ACM Digital Library (http://dl.acm.org)

IEEE Xplore (http://ieeexplore.ieee.org/Xplore/home.jsp)

ScienceDirect (http://www.sciencedirect.com)


UniFil
EDUCAÇÃO A DISTÂNCIA
UNIFIL.BR