

FGF- Faculdade Integrada da Grande
Fortaleza
Sistemas para Internet

Calculadora de Operações Simples com Conversão Binária

Acadêmico: Tharles Michael Batista Amaro (201610750)

Acadêmico: Francisco Lucas Agostinho de sousa (201610916)

Acadêmico: José Enivaldo Costa Queiroz (201610936)

Professor orientador: Hitalo Joseferson Batista Nascimento

Abril
2018

FGF- Faculdade Integrada da Grande
Fortaleza
Sistemas para internet
Programa

Documentação

A documentação abordada é direcionada ao usuário, onde mesmo pode ter o conhecimento das funcionalidades do aplicativo da calculadora e compreenderá sobre a criação do aplicativo e as ferramentas utilizadas para o seu desenvolvimento.

Abril
2018

Índice

1	PREFÁCIO	1
1.1	Objetivo deste documento	1
1.2	Uso deste documento	1
2	INTRODUÇÃO	2
2.1	Propósito	2
2.2	Definições, Acrônimos e Abreviações	2
3	VISÃO GERAL DO SISTEMA	4
3.1	Características do sistema	4
3.2	Arquitetura do Sistema	4
4	Descrição de atividades	14
5	Análise dos Resultados	21
6	Trabalhos Futuros	22
	Bibliografia	23

1 PREFÁCIO

1.1 Objetivo deste documento

Esse trabalho aborda a criação do aplicativo Calculadora com conversão de sistemas numéricos como binário, decimal, hexadecimal, octal, operações e conversões de números reais. No intuito de ajudar o usuário a resolver problemas básicos de cálculos usando as quatro operações simples e as conversões. Para o desenvolvimento desse aplicativo foi utilizado ferramentas de desenvolvimento tais como Ionic 3 e Angular 4, que estabelecem o suporte nos principais sistemas operacionais móveis como IOS e Android.

1.2 Uso deste documento

Essa documentação tem como intuito mostrar o desenvolvimento do aplicativo, mostrando e explicando o código fonte da calculadora. Além de mostrar com imagens ilustrativas o passo a passo do funcionamento do aplicativo.

2 INTRODUÇÃO

2.1 Propósito

O aplicativo tem como base o uso da calculadora com as quatro operações básicas, tais como soma, subtração, multiplicação e divisão, usando o sistema numérico decimal e o sistema de números reais. Pela aba de conversão, o aplicativo ajuda o usuário a saber qual o valor de um determinada operação matemática em diferentes tipos de sistemas numéricos, tais como o sistema binário, decimal, octal e hexadecimal.

A construção do aplicativo foi realizado utilizando o framework Ionic 3, Node.js e Angular 4, pois os frameworks citados oferecem a criação de aplicações híbridas. O Ionic é um open source SDK que usa um conceito chamado native-feeling mobile apps, que significa o desenvolvimento de aplicativos móveis com tecnologias web como, HTML, CSS e JavaScript. Usando assim, o conhecimento sobre esses frameworks para poder desenvolver esse projeto, de forma que venha condizer com a realidade para solucionar os problemas matemáticos estabelecidos pelo usuário.

2.2 Definições, Acrônimos e Abreviações

- **Ionic 3:** É um framework para desenvolvimento de aplicações para dispositivos móveis que visa o desenvolvimento de apps híbridas e de rápido e fácil desenvolvimento. Utilizando de linguagens de programação WEB.
- **AngularJS:** É um framework JavaScript código aberto, mantido pelo Google, que auxilia na execução de single-page aplicativos.
- **Cordova:** É uma estrutura de desenvolvimento móvel de código aberto. Ele permite que você use tecnologias padrão da web - HTML5, CSS3 e JavaScript para desenvolvimento em várias plataformas.
- **HTML:** Linguagem de Marcação de Hipertexto. Consiste em uma linguagem de marcação utilizada para produção de páginas na web, que permite a criação de documentos que podem ser lidos em praticamente qualquer tipo de computador e transmitidos pela internet.

- **CSS:** O Cascading Style Sheets (CSS) é uma "folha de estilo" composta por "camadas" e utilizada para definir a apresentação (aparência) em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação (como XML, HTML e XHTML).

3 VISÃO GERAL DO SISTEMA

3.1 Características do sistema

Efetuar as quatro operações básicas da matemática e realizar conversão de sistemas numéricos binário, decimal, octal e hexadecimal. Cujo o código é de fácil interpretação e a usabilidade do aplicativo é intuitivo.

3.2 Arquitetura do Sistema

```
calculator.html x
1 <ion-header>
2
3 <ion-navbar>
4   <ion-title>Calculadora</ion-title>
5 </ion-navbar>
6 </ion-header>
7
8 <ion-content padding>
9
10 <ion-list>
11
12   <ion-item>
13     <ion-input type="text" placeholder="0" [(ngModel)]="result" readonly</ion-input>
14   </ion-item>
15 </ion-list>
16
17 <div style="margin-top: 50px">
18
19   <div class="row">
20     <div class="col col-20 col-offset-10">
21       <button ion-button block (click)="addValue('1')">1</button>
22     </div>
23     <div class="col col-20">
24       <button ion-button block (click)="addValue('2')">2</button>
25     </div>
26     <div class="col col-20">
27       <button ion-button block (click)="addValue('3')">3</button>
28     </div>
29     <div class="col col-20">
30       <button ion-button block color="secondary" (click)="addValue('.')">-</button>
31     </div>
32   </div>
33
```

Figura 1: Código calculator.html

```
calculator.html x
34 </div>
35
36 <div class="row">
37   <div class="col col-20 col-offset-10">
38     <button ion-button block (click)="addValue('4')">4</button>
39   </div>
40   <div class="col col-20">
41     <button ion-button block (click)="addValue('5')">5</button>
42   </div>
43   <div class="col col-20">
44     <button ion-button block (click)="addValue('6')">6</button>
45   </div>
46   <div class="col col-20">
47     <button ion-button block color="secondary" (click)="addValue('+')">+</button>
48   </div>
49 </div>
50
51 <div class="row">
52   <div class="col col-20 col-offset-10">
53     <button ion-button block (click)="addValue('7')">7</button>
54   </div>
55   <div class="col col-20">
56     <button ion-button block (click)="addValue('8')">8</button>
57   </div>
58   <div class="col col-20">
59     <button ion-button block (click)="addValue('9')">9</button>
60   </div>
61   <div class="col col-20">
62     <button ion-button block color="secondary" (click)="addValue('/')">/</button>
63   </div>
64 </div>
65
66 <div class="row">
```

Figura 2: Código calculator.html

```

calculator.html ✕
66 <div class="row">
67   <div class="col col-20 col-offset-10">
68     <button ion-button block color="danger" (click)="addValue('CE')">CE</button>
69   </div>
70   <div class="col col-20">
71     <button ion-button block (click)="addValue('0')">0</button>
72   </div>
73   <div class="col col-20">
74     <button ion-button block color="secondary" (click)="addValue('.')">.</button>
75   </div>
76   <div class="col col-20">
77     <button ion-button block color="secondary" (click)="addValue('*')">*</button>
78   </div>
79 </div>
80
81 <button ion-button full color="danger" (click)="addValue('=')">=</button>
82
83 </div>
84
85 </ion-content>
86

```

Figura 3: Código calculator.html

No arquivo calculator.html as figuras 1, 2 e 3, exibe o código fonte da calculadora, onde se encontra a estrutura em HTML e CSS que na qual mostra como são feitos os teclados numéricos e os botões que correspondem as quatro operações básicas de matemática.

O código em questão utiliza *tags* e bibliotecas específicas do Ionic 3, cuja a ferramenta é um *framework* que se utiliza as linguagens de marcação como HTML e CSS para construção de *layouts* de aparência na criação dos teclados numéricos e nos botões que correspondem as quatro operações básicas da matemática.

Nota-se que no código, o Ionic usar o chamado native-feelling mobile apps, que quer dizer desenvolvimento de aplicativos móveis com tecnologias web como HTML, CSS e JavaScript.


```

1  import {Component} from '@angular/core';
2  import {IonicPage, NavController, NavParams, AlertController} from 'ionic-angular';
3
4  @IonicPage()
5  @Component({
6    selector: 'page-calculator',
7    templateUrl: 'calculator.html',
8  })
9  export class CalculatorPage {
10
11    result: string = '';
12
13
14    addValue(input): void {
15      switch (input) {
16        case 'CE':
17          this.result = '';
18          break;
19
20        case '=':
21          if (this.result == '') {
22            return;
23          }
24          try {
25            this.result = eval(this.result);
26          } catch (error) {
27

```

Figura 1: Código calculator.ts

Na figura 1, na linha 12 a variável "result" do tipo String. Recebe o valor concatenado da entrada do usuário e também guarda o valor dos cálculos. Na linha 21 Caso o usuário digite a entrada "CE" o valor da variável result será definido como "vazio". Na linha 26 Caso a entrada seja "=" a função irá realizar a operação e guardar o resultado na variável "result", as funções são utilizadas para a ação aos botões de Igual onde irá gerar o resultado de uma operação matemática e o botão CE que na sua funcionalidade resulta apagar o valor mostrado, deixando limpo o local onde se insere o valor.

```

34    this.showMessage();
35    this.result = '';
36  }
37  break;
38
39  case '.':
40
41    let emptyResult: boolean;
42    let lastCharacter: string;
43    let lastDigitIsOperator: boolean;
44
45    emptyResult = this.result == '';
46    lastCharacter = this.result.charAt(this.result.length - 1);
47
48    lastDigitIsOperator = lastCharacter == '+' || lastCharacter == '-' || lastCharacter == '*' || lastCharacter ==
49    '/';
50
51    this.result += emptyResult || lastDigitIsOperator ? '0.' : input;
52    break;
53
54  default:
55    this.result += input;
56    break;
57  }
58
59  showMessage(): void {

```

Figura 2: Código calculator.ts

Na figura 2, linha 40 a função utilizada para dar ação ao botão Ponto, tem como ação auxiliar o usuário na utilização de números decimais em uma

das operações matemáticas. Já na linha 50 A função "charArt" vai pegar um único caractere em uma posição específica que é passada por parâmetro "length - 1" que é aplicado a variável "result", que pega sempre o último caractere digitado. nas demais linhas darão ações aos botões de operações matemáticas, tais como soma, subtração, multiplicação e divisão.

```
TS calculator.ts
60     this.result += emptyResult || lastDigitIsOperator ? '0.' : input;
61     break;
62
63
64     default:
65     this.result += input;
66     break;
67   }
68 }
69
70
71 showMessage(): void {
72   this.alertCtrl.create({
73     title: 'Operação inválida.',
74     subTitle: 'Por favor, tente novamente.',
75     buttons: ['Ok']
76   }).present();
77 }
78
79 constructor(public navCtrl: NavController, public navParams: NavParams, private alertCtrl: AlertController) {
80 }
81
82 ionViewDidLoad() {
83   console.log('ionViewDidLoad CalculatorPage');
84 }
85
86 }
```

Figura 3: Código calculator.ts

Já na figura 3, linha 71 temos uma função utilizada para alertar o usuário, caso tente realizar uma operação inválida. Irá aparecer uma seguinte mensagem “Operação Inválida. Por favor, tente novamente” e abaixo aparecerá um botão de OK onde retornará a tela inicial da calculadora.

```
conversion.html
1 <ion-header>
2
3 <ion-navbar>
4   <ion-title>Conversão</ion-title>
5 </ion-navbar>
6
7 </ion-header>
8
9 <ion-content padding>
10
11 <ion-list>
12   <ion-item>
13     <ion-label>Escolha a base</ion-label>
14     <ion-select [(ngModel)]="currentBase" (ionChange)="buttonDisplay()">
15       <ion-option value="2">Binária</ion-option>
16       <ion-option value="8">Octal</ion-option>
17       <ion-option value="10">Decimal</ion-option>
18       <ion-option value="16">Hexadecimal</ion-option>
19     </ion-select>
20   </ion-item>
21 </ion-list>
22
23 <ion-list>
24
25   <ion-item>
26     <ion-input type="text" placeholder="0" [(ngModel)]="conversionInput" readonly></ion-input>
27   </ion-item>
28
29   <div style="margin-top: 20px">
30
31     <div class="row">
32       <div class="col col-20 col-offset-10">
33         <button ion-button block [ngClass]="{'not-visible': !showBtnToI(), 'is-visible': showBtnToI()}"
```

Figura 1: Código conversion.html

```

34 | | | | (click)="addValue('0')">0
35 | | | | </button>
36 | | | | </div>
37 | | | | <div class="col col-20">
38 | | | |   <button ion-button block [ngClass]="{'not-visible': !showBtn0To1(), 'is-visible': showBtn0To1()}"
39 | | | |     (click)="addValue('1')">1
40 | | | |   </button>
41 | | | | </div>
42 | | | | <div class="col col-20">
43 | | | |   <button ion-button block [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
44 | | | |     (click)="addValue('2')">2
45 | | | |   </button>
46 | | | | </div>
47 | | | | <div class="col col-20">
48 | | | |   <button ion-button block [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
49 | | | |     (click)="addValue('3')">3
50 | | | |   </button>
51 | | | | </div>
52 | | | | </div>
53 | | | | <div class="row">
54 | | | |   <div class="col col-20 col-offset-10">
55 | | | |     <button ion-button block
56 | | | |       [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
57 | | | |       (click)="addValue('4')">4
58 | | | |     </button>
59 | | | |   </div>
60 | | | |   <div class="col col-20">
61 | | | |     <button ion-button block
62 | | | |       [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
63 | | | |       (click)="addValue('5')">5
64 | | | |     </button>
65 | | | |   </div>
66 | | | | </div>

```

Figura 2: Código conversion.html

```

67 | | | | <div class="col col-20">
68 | | | |   <button ion-button block
69 | | | |     [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
70 | | | |     (click)="addValue('6')">6
71 | | | |   </button>
72 | | | | </div>
73 | | | | <div class="col col-20">
74 | | | |   <button ion-button block
75 | | | |     [ngClass]="{'not-visible': !showBtn2To7(), 'is-visible': showBtn2To7()}"
76 | | | |     (click)="addValue('7')">7
77 | | | |   </button>
78 | | | | </div>
79 | | | | </div>
80 | | | | <div class="row">
81 | | | |   <div class="col col-20 col-offset-10">
82 | | | |     <button ion-button block
83 | | | |       [ngClass]="{'not-visible': !showBtn8To9(), 'is-visible': showBtn8To9()}"
84 | | | |       (click)="addValue('8')">8
85 | | | |     </button>
86 | | | |   </div>
87 | | | |   <div class="col col-20">
88 | | | |     <button ion-button block
89 | | | |       [ngClass]="{'not-visible': !showBtn8To9(), 'is-visible': showBtn8To9()}"
90 | | | |       (click)="addValue('9')">9
91 | | | |     </button>
92 | | | |   </div>
93 | | | |   <div class="col col-20">
94 | | | |     <button ion-button block
95 | | | |       [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
96 | | | |       (click)="addValue('A')">A
97 | | | |     </button>
98 | | | |   </div>
99 | | | | </div>

```

Figura 3: Código conversion.html

```

100 | | | | <div class="col col-20">
101 | | | |   <button ion-button block
102 | | | |     [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
103 | | | |     (click)="addValue('B')">B
104 | | | |   </button>
105 | | | | </div>
106 | | | | </div>
107 | | | | <div class="row">
108 | | | |   <div class="col col-20 col-offset-10">
109 | | | |     <button ion-button block
110 | | | |       [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
111 | | | |       (click)="addValue('C')">C
112 | | | |     </button>
113 | | | |   </div>
114 | | | |   <div class="col col-20">
115 | | | |     <button ion-button block
116 | | | |       [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
117 | | | |       (click)="addValue('D')">D
118 | | | |     </button>
119 | | | |   </div>
120 | | | |   <div class="col col-20">
121 | | | |     <button ion-button block
122 | | | |       [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
123 | | | |       (click)="addValue('E')">E
124 | | | |     </button>
125 | | | |   </div>
126 | | | |   <div class="col col-20">
127 | | | |     <button ion-button block
128 | | | |       [ngClass]="{'not-visible': !showBtnAtoF(), 'is-visible': showBtnAtoF()}"
129 | | | |       (click)="addValue('F')">F
130 | | | |     </button>
131 | | | |   </div>
132 | | | | </div>

```

Figura 4: Código conversion.html

```

133 </div>
134
135
136
137 <div class="row">
138   <div class="col col-20 col-offset-10">
139     <button ion-button full color="secondary" (click)="conversion()">Converter</button>
140   </div>
141   <div class="col col-20 col-offset-10">
142     <button ion-button full color="danger" (click)="limpar()">Limpar</button>
143   </div>
144 </div>
145
146 </div>
147
148 <div [ngClass]="{'not-visible': !currentBaseNotNull(), 'is-visible': currentBaseNotNull()}">
149
150   <ion-item>
151     <ion-label floating:Binário:</ion-label>
152     <ion-input type="text" [(ngModel)]="resultBinary" readonly</ion-input>
153   </ion-item>
154   <ion-item>
155     <ion-label floating:Octal:</ion-label>
156     <ion-input type="text" [(ngModel)]="resultOctal" readonly</ion-input>
157   </ion-item>
158   <ion-item>
159     <ion-label floating:Decimal:</ion-label>
160     <ion-input type="text" [(ngModel)]="resultDecimal" readonly</ion-input>
161   </ion-item>
162   <ion-item>
163     <ion-label floating:Hexadecimal:</ion-label>
164     <ion-input type="text" [(ngModel)]="resultHexadecimal" readonly</ion-input>
165   </ion-item>
166 </div>

```

Figura 5: Código conversion.html

```

148 <div [ngClass]="{'not-visible': !currentBaseNotNull(), 'is-visible': currentBaseNotNull()}">
149
150   <ion-item>
151     <ion-label floating:Binário:</ion-label>
152     <ion-input type="text" [(ngModel)]="resultBinary" readonly</ion-input>
153   </ion-item>
154   <ion-item>
155     <ion-label floating:Octal:</ion-label>
156     <ion-input type="text" [(ngModel)]="resultOctal" readonly</ion-input>
157   </ion-item>
158   <ion-item>
159     <ion-label floating:Decimal:</ion-label>
160     <ion-input type="text" [(ngModel)]="resultDecimal" readonly</ion-input>
161   </ion-item>
162   <ion-item>
163     <ion-label floating:Hexadecimal:</ion-label>
164     <ion-input type="text" [(ngModel)]="resultHexadecimal" readonly</ion-input>
165   </ion-item>
166   <ion-item class="not-visible"></ion-item>
167 </div>
168
169 </ion-list>
170
171 </ion-content>
172
173

```

Figura 6: Código conversion.html

Na figura 1, 2, 3, 4, 5 e 6.html, temos o código fonte da conversão, onde se encontra a estrutura em HTML e CSS que na qual mostra como são feitos o teclado numérico, a janela de escolha de base e os botões de converter e limpar.

A estrutura da janela de conversão mostra o espaço onde será inserido o valor, a função que gera uma janela onde o usuário poderá escolher a base numérica para fazer a conversão.

Após a escolha do sistema numérico, a uma função que acionada ira o aplicativo irá gera um teclado respectivo os números do sistema escolhido. Caso o usuário que tenta novamente, basta utilizar o botão limpar, onde deixa limpo o espaço para ser inserido um novo valor.

```

1  import {Component} from '@angular/core';
2  import {IonicPage, NavController, NavParams} from 'ionic-angular';
3
4  @IonicPage()
5  @Component({
6    selector: 'page-conversion',
7    templateUrl: 'conversion.html'
8  })
9  export class ConversionPage {
10
11    private isBinary: boolean;
12    private isOctal: boolean;
13    private isDecimal: boolean;
14    private isHexadecimal: boolean;
15    public currentBase: string;
16    public conversionInput: string;
17    public resultBinary: string;
18    public resultOctal: string;
19    public resultDecimal: string;
20    public resultHexadecimal: string;
21
22    buttonDisplay(): void {
23
24      this.conversionInput = '';
25      this.resultBinary = '';
26      this.resultOctal = '';
27      this.resultDecimal = '';
28      this.resultHexadecimal = '';
29
30      this.isBinary = this.currentBase == "2";
31      this.isOctal = this.currentBase == "8";
32      this.isDecimal = this.currentBase == "10";
33
34

```

Figura 1: Código conversion.ts

Na figura 1 nas linhas 23 à 34 teremos a limpeza dos dados das variáveis de resultado e dados de entrada da conversão em seguida verifica qual a base numérica escolhida(2,8,10,16) e atribui o valor true ou false para as variáveis correspondentes a base numérica escolhida.

```

34      this.isHexadecimal = this.currentBase == "16";
35    }
36
37    currentBaseNotNull(): boolean {
38      return !(this.currentBase == "" || this.currentBase == null);
39    }
40
41
42    showBtn0To1(): boolean {
43      return this.isBinary || this.isOctal || this.isDecimal || this.isHexadecimal;
44    }
45
46    showBtn2To7(): boolean {
47      return !this.isBinary && (this.isOctal || this.isDecimal || this.isHexadecimal);
48    }
49
50    showBtn8To9(): boolean {
51      return !this.isBinary && !this.isOctal && (this.isDecimal || this.isHexadecimal);
52    }
53
54    showBtnAtoF(): boolean {
55      return !this.isBinary && !this.isOctal && !this.isDecimal && this.isHexadecimal;
56    }
57
58    addValue(input): void {
59      this.conversionInput += input;
60    }
61
62    conversion(): void {
63
64
65

```

Figura 2: Código conversion.ts

na figura 2 nas linhas 38 à 40 termos a função que verifica se a base escolhida ainda não foi selecionada logo depois nas linhas 43 à 57 as funções "showBtn" fazem algumas validações e retorna true ou false. Isso no HTML irá ser usado para definir se o botão deve ser mostrado ou não em relação a base numérica escolhida.

```

TS conversion.ts x
67 let currentBaseValueNotNull: boolean;
68 let value: string;
69 let decimal;
70
71 currentBaseValueNotNull = this.currentBase == "2" || this.currentBase == "8" || this.currentBase == "10" ||
72 this.currentBase == "16";
73 value = this.conversionInput;
74
75 if (currentBaseValueNotNull) {
76   if (value == '' || value == null) {
77     this.resultBinary = '';
78     this.resultDecimal = '';
79     this.resultOctal = '';
80     this.resultHexadecimal = '';
81   } else {
82     switch (this.currentBase) {
83       case "2":
84         decimal = parseInt(value, 2);
85         break;
86       case "10":
87         decimal = parseInt(value);
88         break;
89       case "8":
90         decimal = parseInt(value, 8);
91         break;
92       case "16":
93         decimal = parseInt(value, 16);
94         break;
95     }
96   }
97 }
98

```

Figura 3: Código conversion.ts

Na figura 3 nas linhas 65 à 105 teremos função que irá realizar as conversões de base numéricas.

```

TS conversion.ts x
99 this.resultBinary = decimal.toString(2);
100 this.resultDecimal = decimal;
101 this.resultOctal = decimal.toString(8);
102 this.resultHexadecimal = decimal.toString(16);
103 }
104 }
105 }
106
107 |
108 clear(): void {
109   this.conversionInput = "";
110   this.resultBinary = "";
111   this.resultOctal = "";
112   this.resultDecimal = "";
113   this.resultHexadecimal = "";
114 }
115
116 constructor(public navCtrl: NavController, public navParams: NavParams) {
117 }
118
119 ionViewDidLoad() {
120   console.log('ionViewDidLoad ConversionPage');
121 }
122 }
123

```

Figura 4: Código conversion.ts

Na figura 4 linhas 108 à 114 teremos uma função que irá limpar a tela(valores do input e resultados).

```

TS conversion.module.ts ✕
1  import { NgModule } from '@angular/core';
2  import { IonicPageModule } from 'ionic-angular';
3  import { ConversionPage } from '../conversion';
4
5  @NgModule({
6    declarations: [
7      ConversionPage,
8    ],
9    imports: [
10     IonicPageModule.forChild(ConversionPage),
11   ],
12 })
13 export class ConversionPageModule {}
14

```

Figura 5: Código conversion.module.ts

Na figura 5 conversion.module.ts temos o código fonte do modulo de criação de pagina.

```

conversion.scss ✕
1  page-conversion {
2    .not-visible { display: none; }
3    .is-visible { display: inline-block; }
4  }
5

```

Figura 6: Código conversion.scss

Na figura 6, o código usado é o CSS.

No código as funções seguem um padrão de escolha, onde a escolha da base numérica afetara tanto na criação do teclado em HTML como na hora que for gera o resultado final. Pois, o código segue a lógica de sequência onde o primeiro sistema numérico é o binário, o segundo é o octal, terceiro o decimal e o quarto é hexadecimal.

Já sobre o resultado gerado, após o usuário ter escolhido o sistema numérico, abaixo do botão converter a o complemento da janela conversão onde o usuário poderá ver o resultado gerado pela conversão. Onde o resultado será mostrado em todos os sistemas numéricos na mesma sequência lógica da janela de escolha de base.

Já na figura 6, o código usado é o CSS onde irá oculta o teclado quando não estiver sendo usado e ira mostra o teclado respectivamente ao sistema numérico escolhido.

4 Descrição de atividades

As funcionalidades do aplicativo serão mostradas através das ilustrações e explicações do aplicativo.

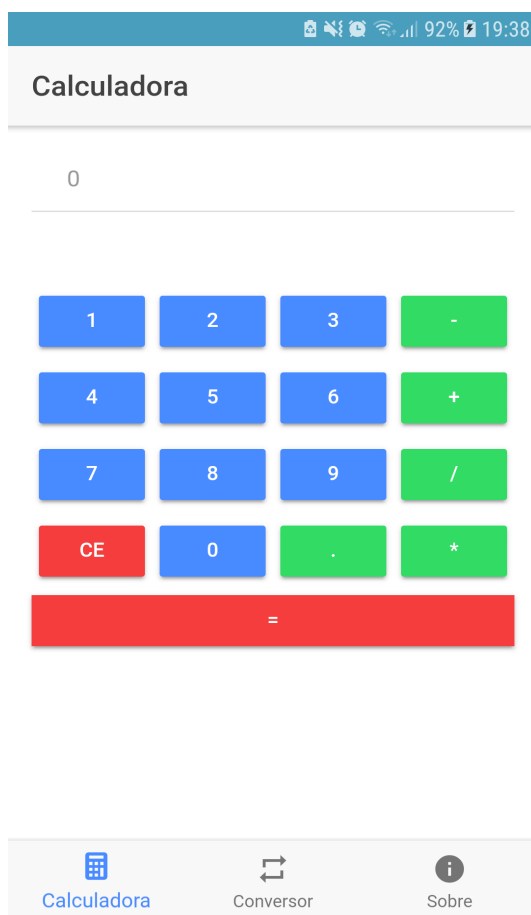


Figura 1: Tela Inicial do Aplicativo

Na figura 1, a tela inicial contém o teclado numérico e os ícones, que representam as operações simples de matemáticas, onde o usuário poderá resolver operações simples tais como soma, subtração, multiplicação e divisão.

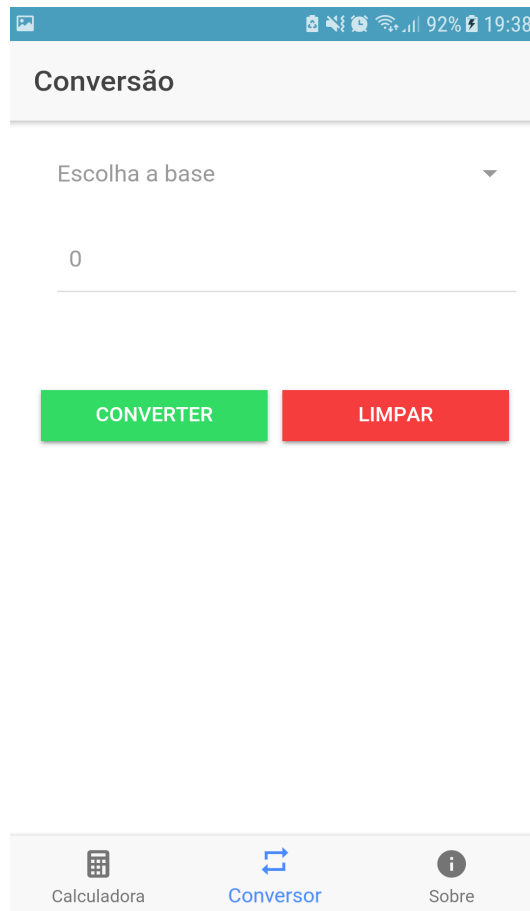


Figura 2: Tela de Conversão

Na figura 2, a tela Conversão tem o espaço em branco onde o usuário irá digitar o número. Na parte de escolha da base o usuário terá que ir na seta à direita para poder ver uma janela que mostrará as opções de sistemas numéricos.

Embaixo se encontra o botão de converter onde o usuário após digitado o número ele irá clicar em converter e assim mostrará o resultado. No lado esquerdo o usuário irá encontrar o botão limpa, onde ele poderá usar para apagar o valor caso tenha errado ou queira testar um novo valor.

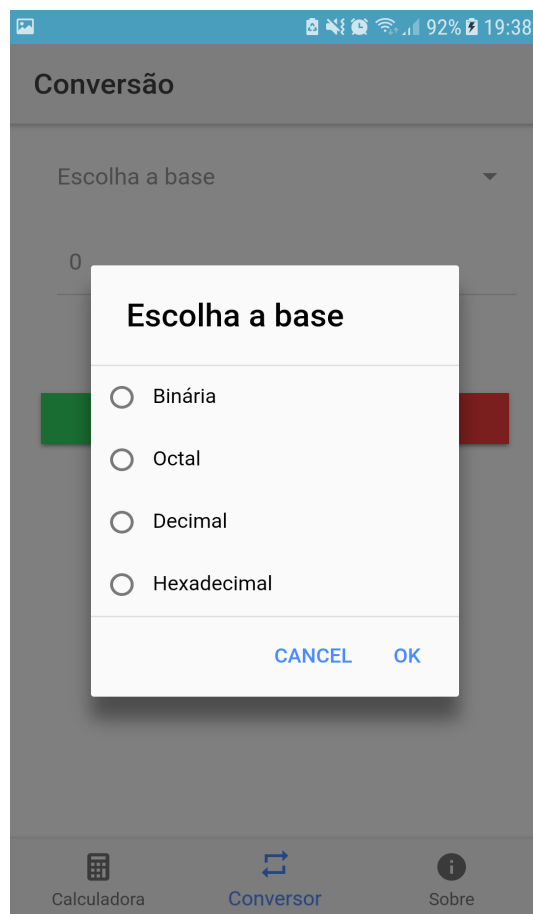


Figura 3: Tela de Escolha de base

Na figura 3, a tela de escolha de base na qual o usuário poderá selecionar a base numérica onde queira usar para converter o valor desejado, intercalando entre o sistema binário, octal, decimal e hexadecimal.

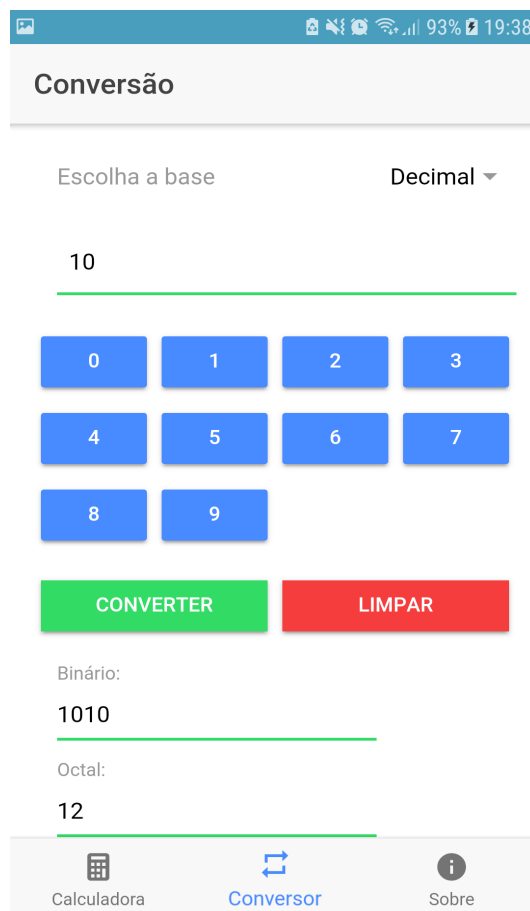


Figura 4: Tela de Resultado de conversão parte 1

Na figura 4, a tela de conversão temos um exemplo onde o usuário seleciona o sistema decimal e utilizar o teclado gerado pelo aplicativo, assim digitando o valor e selecionado o botão converter, mostrando assim o resultado nos demais sistemas numéricos.

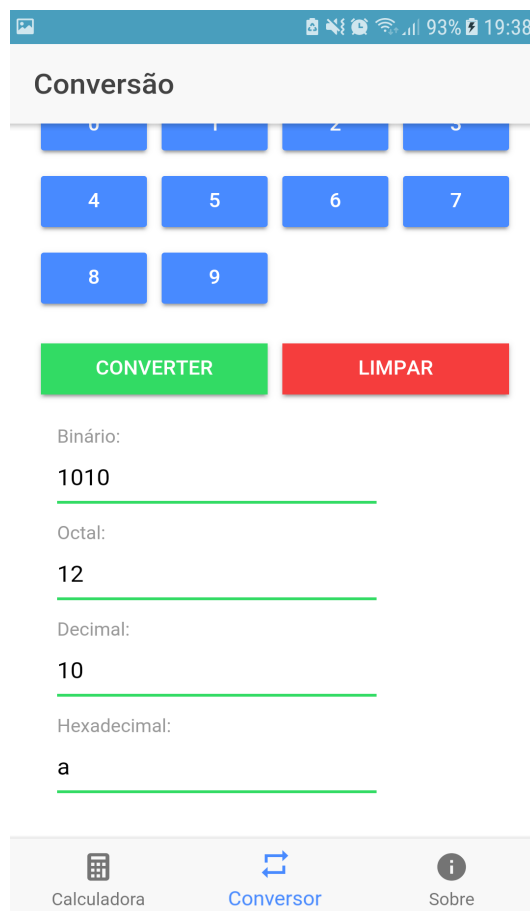


Figura 5: Tela de Resultado de conversão parte 2

Na figura 5, tela de conversão temos a continuação do resultado da primeira imagem, na qual o usuário irá visualizar o resultador nos demais sistemas numéricos.

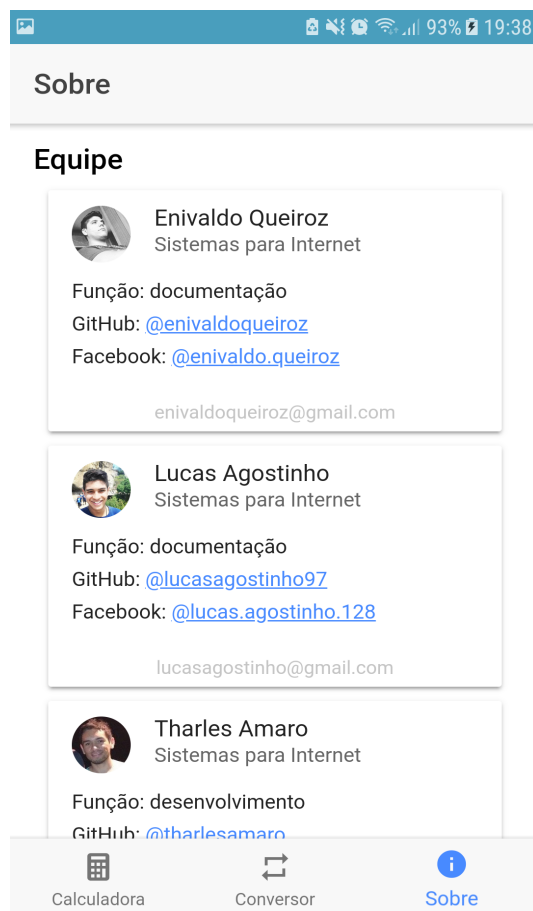


Figura 6: Tela de Sobre os desenvolvedores do Aplicativo

Na figura 6, aba Sobre, ha informações dos desenvolvedores. Contendo a funções de cada desenvolvedores, o Github e a rede social.



Figura 7: Tela de Sobre os desenvolvedores do Aplicativo

Na figura 7, Continuando na aba Sobre, contem informações sobre o desenvolvimento do aplicativo e quais ferramentas foram utilizadas e um link para o Github contendo o código fonte do aplicativo.

5 Análise dos Resultados

O resultado do aplicativo desenvolvido atendeu todas as necessidades exigidas pelo o edital. No qual os requisitos solicitados foram atendidos e amplamente testados, baseando nas ideias sugeridas pela equipe e pelo o orientador. Nelas implementamos o uso do Inoic 3 junto com o Node.js e Angular 4, para a construção estrutural, lógica e funcional do aplicativo.

Tendo assim como resultado um aplicativo simples, mas com característica exclusiva. Onde na aba de conversões o usuário poderá após digitado o valor, ver o resultado em todos os sistemas numéricos suportado pelo aplicativo.

6 Trabalhos Futuros

Com o conhecimento obtido nesse projeto, esperamos que a equipe desenvolvedora continue a construção e o melhoramento do aplicativo de calculadora. podendo assim oferece mais serviços com o intuito de poder ajuda o usuário final.

Bibliografia

IONIC FRAMEWORK . Cria aplicativos nativos incríveis em uma base código, para qualquer plataforma Web. Disponível em: <https://ionicframework.com/> Acesso em: 26 de março de 2018.

AngularJS. permite estender o vocabulário HTML para seu aplicativo. Disponível em: <https://angularjs.org/> Acesso em: 26 de março de 2018.

Nodejs. é um tempo de execução de JavaScript, usando um modelo de E / S sem bloqueio orientado a eventos que o torna leve e eficiente. Disponível em: <https://nodejs.org/en/> Acesso em: 26 de março de 2018.

Cordova. é uma estrutura de desenvolvimento móvel de código aberto. Disponível em: <http://cordova.apache.org/> Acesso em: 26 de março de 2018.