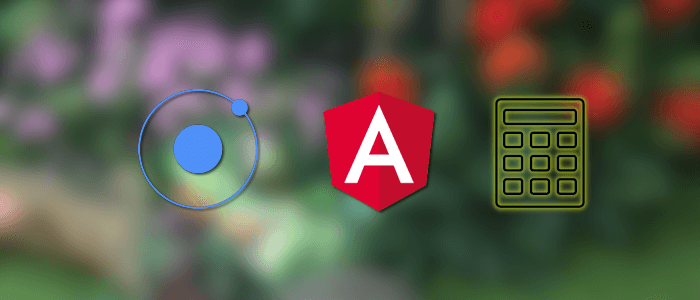
Tutorial Ionic 5: Crear una Calculadora con Ionic (Angular)



1. Introducción

Esta es una guía rápida de Ionic 5 para la creación de una Calculadora que podremos utilizar tanto en un dispositivo Android como en uno iOS.

En este tutorial rápido se utilizará un rema oscuro para la calculadora, similar a la apariencia de la calculadora de iOS. La característica de esta calculadora es que tiene los cálculos de una calculadora sencilla, es decir, las operaciones que pueden realizarse con operaciones algebráicas simples.

# Objetivos

* Crear una nueva aplicación con Ionic 5.
* Crear las funciones características para la Calculadora.
* Uso de bucles switch e if.
* Implimentar elementos UI/UX de la Calduladora.
* Ejecutar y Testear la apliación en un dispositivo.

Para poder realizar y seguir el Tutorial es necesario tener instaladas las siguientes apliaciones:

1. [Node.js](https://nodejs.org/).
2. [Ionic 5](https://ionicframework.com/).
3. [Ionic CLI](https://ionicframework.com/docs/cli).
4. [Angular](https://angular.io/).
5. [Cordova](https://cordova.apache.org/).
6. Terminal o Node Command Line
7. Editor de textos o IDE (se utilizará VSCode).

# Create a New Ionic 5 App

Se utilizará Ionic CLI para crear una aplicación Ionic 5, por lo que es necesario tener instalada o actualizada Ionic CLI a la última versió. Antes de eso es necesario asegurarse de que se tiene instalao Node.js y npm. Para comprobarlo escribiremos en un termina o en una línea de comandos los siguientes comandos:

node -v

v14.18.1

npm -v

8.1.3

Estas con las versions actualizadas cuendo se redacto el presente Tutorial (Noviembre 2021). Para instalar o actualizar Ionic CLI escribiremos el siguiente comando:

sudo[[1]](#footnote-1) npm install -g @ionic/cli

sudo es opcional si [Node.js/npm](http://node.js/NPM) require permisos de administrador. Tras esto se tiene instalada la última versión de Ionic/CLI; para comprobar la versión instalada escribiremos:

ionic -v

6.18.1

El siguiente paso será la creación de una aplicación con Ionic 5 escribiendo el siguiente comando:

ionic start calculator blank –type=angular

Cuando se nos pregunte si queremos iniciar sesión en nuestra cuenta de Ionic diremos que no pusando la tecla n. Lo siguiente será acceder a la carpeta recién creada, donde hemos creado nuestro proyecto, y ejecutar la aplicación para ver su aspecto inicial.

cd ./calculator

ionic serve -l

Al añadir el parámetro –l después de serve hacemos que la aplicación se nos muestre como si estuviera ejecutándose en un dispositivo Android y en uno iOS. Si queremos ver la apariencia en un naegador web no escribiremos este parámetro.

La primera vez que ejecutamos una aplicación con el parámetr –l se os pedirá que instalemos @ionic/lab, sólo deberemos pulsar la tecla y para aceptar. Si no se abre el navegador de forma automática con la aplicación, deberemos abrirlo nosotros e ir a la dirección localhost=8100[[2]](#footnote-2).

Ya estamos en disposición de empezar a desarrollar la aplicación.

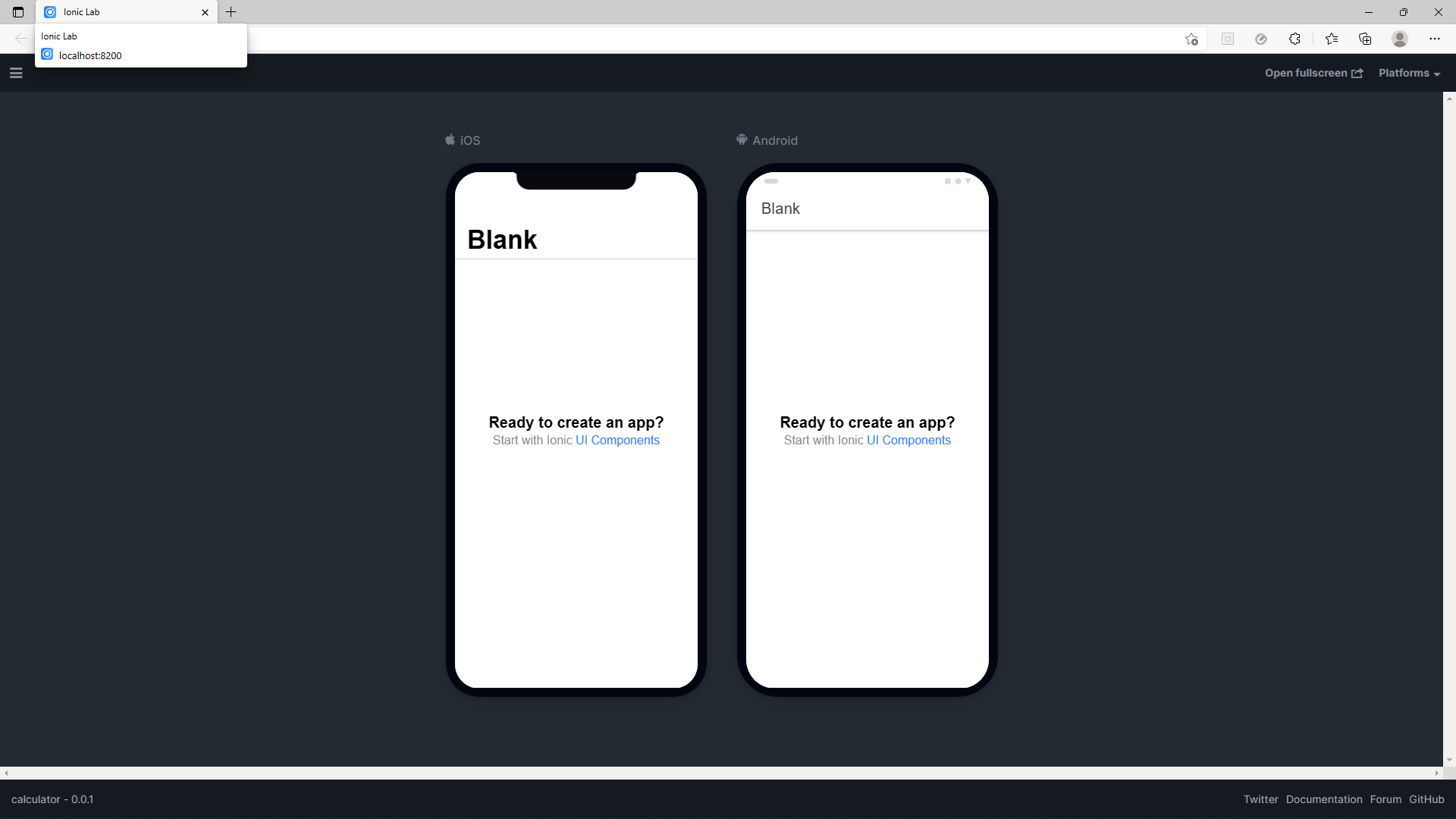


Imagen 1. Aspecto inicial de laaplicación.

# 3. Crear las funciones necesarias para la Calculadora

Vamos a utilizar una forma un poco fea para crear la funcionalidad de la calculadora, si quiere verse una forma más pulida consultad el Tutorial para la aplicación de sonidos. Iniciamos el IDES o el Editor de Textos seleccionado para abrir el proyecto de Ionic 5. En nuestro caso abriremos VSCode escribiendo en el terminal, una vez situados en la carpeta del Proyecto, el siguiente comando para ejecutar la aplicación y abrir el proyecto:

code .

## Las variables

El siguiente paso será editar el archivo [**home.page.ts**](http://home.page.ts) que está en la ruta. **src/app/home/home.page.ts** y añadiremos las siguientes variables justo debajo de la línea que pone: **export class HomePage {**.

display='0';

  firstval:number=null;

  operator:any=null;

  newcursor=false;

  iscomma=false;

  isc=false;

La variable **display** muestra el valor inicial que mostrará la calculadora al arrancarse; **firstval** es una variable numérica o el operador izquierdo; **operator** es una variable para mantener la operación en la calculadora: /, X, -, +; **newcursor** es una variable que comprueba si el operando está a la izuiqerda o a la derecha del operador; **isc** es una variable que comprueba si el botón de limpieza es *C* o *AC*; e **iscomma** es una variable que comprueba si el número contiene una coma o no pulsando el botón “*,*”.

Debe realizarse un comentario sobre el código mostrado anteriormente. Si nos fijamos bien, hay variables donde se está especificando el tipo de variable y en otras no y ambas formas son correctos ¿Por qué? El motivo es que si inicializamos la variable, le asignamos un valor, TypeScript reconocerá y asignará automáticamente un tipo a la variable y por lo tanto no deberemos asignarlo nosotros. Su esto es cierto, ¿por qué asignamos un tipo e inicializamos la variable firstval? El motivo es el valor asignado. A esta variable se ha inicializado en null y este valor puede asignarse a muchos tipos de variable; por lo tanto, en este caso si que debemos asignar un tipo a la variable ya que TypeScript no puede hacerlo por nosotros.

El siguiente paso es añadir una función para manejar la pulsación de los diferentes botones de la aplicación.

## Función click()

click(val:any){

    switch(val){

      case 'ac':

        this.display='0';

        this.firstval=null;

        this.operator=null;

        this.newcursor=false;

        break;

      case 'c':

        this.display='0';

        this.isc=false;

        break;

      case '+/-':

        if(Math.sign(parseInt(this.display,0))===1){

          const sign=-Math.abs(parseInt(this.display,0));

          this.display=sign.toString();

        }

        else if(Math.sign(parseInt(this.display,0))===-1){

          const sign=Math.abs(parseInt(this.display,0));

          this.display=sign.toString();

        }

        else

        {

          this.display=this.display;

        }

        break;

      case '%':

        this.addpercent();

        break;

      case ':':

        this.addoperator(':');

        break;

      case 'X':

        this.addoperator('X');

        break;

      case '-':

        this.addoperator('-');

        break;

        case '+':

          this.addoperator('+');

          break;

        case '=':

          if(this.firstval !==null && this.operator !== null){

            this.calclast();

          }

          this.operator=null;

          break;

        case '0':

          this.addnumber('0');

          break;

        case '1':

          this.addnumber('1');

          break;

        case '2':

            this.addnumber('2');

            break;

          case '3':

            this.addnumber('3');

            break;

          case '4':

            this.addnumber('4');

            break;

          case '5':

            this.addnumber('5');

            break;

          case '6':

            this.addnumber('6');

            break;

          case '7':

            this.addnumber('7');

            break;

          case '8':

            this.addnumber('8');

            break;

          case '9':

            this.addnumber('9');

            break;

          case ',':

            this.addcomma();

            break;

    }

  }

Como puede verse, hay algunas funciones para realiar diferentes acciones para cada botón. Para operadores, utilizamos la función **operator**, para los números utilizamos la función **addnumber**, para la coma utilizamos la función **addcomma**, para el cálculo del porcentaje utilizamos la función **addpercent**. También hay una función para el cambio de signo *+/-* y el igual utiliza una función de cálculo. También la acción para cambiar el signo del número mediante *+/-* y la acción para limpiar variables que se utilizan las funciones *AC* y *C*. Lo siguiente será añadir una función para añadir una coma al número en pantalla.

## Función addcomma()

addcomma(){

    if(this.iscomma===false){

      this.iscomma=true;

    }

    else

    {

      this.iscomma=false;

    }

  }

## Función addnumber()

  addnumber(nbr:string){

    if(nbr==='0'){

      if(this.newcursor===true){

        this.display=nbr;

        this.newcursor=false;

      }

      else if(this.display!=='0'){

        if(this.iscomma===true){

          this.display=`${this.display.toString()}.${nbr}`;

        }

        else{

          this.display=this.display.toString() + nbr;

        }

      }

    }

    else if(this.display==='0'){

      if(this.iscomma===true){

        this.display=`${this.display.toString()}.${nbr}`;

      }

    }

    else{

      if(this.newcursor===true){

        this.display=nbr;

        this.newcursor=false;

      }

      else if(this.display==='0'){

        if(this.iscomma===true){

          if(this.display.toString().indexOf('.') > -1){

            this.display=this.display.toString() + nbr;

          }

          else{

            this.display=`${this.display.toString()}.${nbr}`;

          }

        }

        else {

          this.display=nbr;

        }

      }

      else{

        if(this.iscomma===true){

          if(this.display.toString().indexOf('.') > -1){

            this.display=this.display.toString() + nbr;

          }

          else {

            this.display=`${this.display.toString()}.${nbr}`;

          }

        }

        else {

          this.display=this.display.toString() + nbr;

        }

      }

    }

    this.iscomma=true;

  }

## Función addpercent()

La siguiente función que vamos a añadir es **addpercent()**, para calcular el porcentaje de un número.

addpercent(){

    this.iscomma=false;

    const dispval=parseInt(this.display, 0)/100;

    this.display=dispval.toString();

  }

## Addoperator()

A continuación, añadimos la función **addoperator()**.

addoperator(op:string){

    if(this.newcursor===false){

      if(this.firstval===null){

        if(this.iscomma===true){

          this.firstval=parseFloat(this.display);

        }

        else{

          this.firstval=parseInt(this.display, 0);

        }

      }

      if(this.firstval!==null && this.operator!==null){

        this.calclast();

      }

    }

    this.iscomma=false;

    this.operator=op;

    this.newcursor=true;

  }

## Función calclast()

Ahora, añadimos la función **calclast()** para realizar los cálculos.

calclast(){

    switch(this.operator){

      case ':':

        if(this.iscomma===true){

          this.firstval=(this.firstval/parseFloat(this.display));

        }

        else{

          this.firstval=(this.firstval/parseInt(this.display, 0));

        }

        break;

      case 'X':

        if(this.iscomma===true){

          this.firstval=(this.firstval\*parseFloat(this.display));

        }

        else{

          this.firstval=(this.firstval\*parseInt(this.display,0));

        }

        break;

      case '-':

        if(this.iscomma===true){

          this.firstval=(this.firstval-parseFloat(this.display));

        }

        else{

          this.firstval=(this.firstval-parseInt(this.display,0));

        }

        break;

      case '+':

        if(this.iscomma===true){

          this.firstval=(this.firstval+parseFloat(this.display));

        }

        else{

          this.firstval=(this.firstval+parseInt(this.display,0));

        }

        break;

    }

    this.display=this.firstval.toString();

  }

# Implementando el UI/UX de la aplicación

Ahora, es hora de implementar las funciones ya vistas en la vista o el UI/UX de la aplicación Calculadora. Para ello debemos abrir en un editor de textos o un IDE el fichero [**home.page.html**](http://home.page.html)que encontraremos en la ruta:**src/app/home/home.page.html**. Lo primero que haremos será borrar todo el contenido del fichero y escribiremos lo siguiente:

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

<div class="calc-main">

<div class="calc-display">

<ion-input type="number" [(ngModel)]="display" class="ion-text-end"></ion-input>

</div>

<ion-grid>

<ion-row>

<ion-col>

<ion-button shape="round" color="light" \*ngIf="!isc" (click)="click('ac')">AC</ion-button>

<ion-button shape="round" color="light" \*ngIf="isc" (click)="click('c')">C</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="light" (click)="click('+/-')">+/-</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="light" (click)="click('%')">%</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="warning" (click)="click(':')">:</ion-button>

</ion-col>

</ion-row>

<ion-row>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('7')">7</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('8')">8</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('9')">9</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="warning" (click)="click('X')">X</ion-button>

</ion-col>

</ion-row>

<ion-row>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('4')">4</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('5')">5</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('6')">6</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="warning" (click)="click('-')">-</ion-button>

</ion-col>

</ion-row>

<ion-row>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('1')">1</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('2')">2</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click('3')">3</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="warning" (click)="click('+')">+</ion-button>

</ion-col>

</ion-row>

<ion-row>

<ion-col size="6">

<ion-button shape="round" color="dark" (click)="click('0')" class="double-width">0</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="dark" (click)="click(',')">,</ion-button>

</ion-col>

<ion-col>

<ion-button shape="round" color="warning" (click)="click('=')">=</ion-button>

</ion-col>

</ion-row>

</ion-grid>

</div>

</ion-content>

Ahora, nos pondremos con la apariencia UI/UX abriendo y editando el fichero [home.page.css](http://home.page.css) que si la siguiente: *src/app/home/home.page.scss*, y reemplazaremos todo código CSS por el siguiente:

.calc-main {

background-color: #000000;

height: 100%;

}

.calc-display {

background-color: #000000;

padding: 60px 15px 20px;

}

ion-input {

color: #ffffff;

font-size: 48px;

}

ion-col {

padding: 0;

margin: 0;

text-align: center;

}

ion-button {

height: 50px;

width: 50px;

border-radius: 50;

font-size: 20px;

}

.double-width {

width: 86%;

}

# Ejecutar y probar la aplicación en un dispositivo

En esta sección vamos a ver como ejecutar la aplicación en un dispositivo iOS y en uno Android; pere ver los pasos necesarios que debemos dar para obtener una apliación insalable y por tanto publicable en los diferentes markets en caso de que la firmemos.

## Ejecutando la aplicación en un dispositivo iOS

Para ejecutar la aplicación en un dispositivo iOS por primera vez, deberemos añadir la plataforma iOS utilizando Cordova mediante el siguiente commando:

ionic cordova platform add ios

Para modificar la el id del paquete de la aplicación en el archivo config.xml se debe brir y modificar la dirección que viene dentro de la etiqueta widget, la cual debe quedar de la siguiente maera:

<widget id="com.djamware.calculator" .../>

Para que el cambio surta efecto debemos eliminar el paquete iOS y volver a añadirlo. Para ello utilizaremos los siguientes comandos:

ionic cordova platform rm ios

ionic cordova platform add ios

Ahora ya solo nos queda la creación de la aplicación para un sistema iOS[[3]](#footnote-3).

ionic cordova build ios

El siguiente paso es abrit **MyApp.xcworkspace** que encontraremos en la dirección **platform/io/**. Para abrirlo se recomienda utilizar XCode. Una vez abierto iremos a la **configuración** (build settins). Modificaremos el equipo de desarrollo en nuestra cuenta de desarrollo.Debemos asegurarnos que el dispositivo iOS está conectado al equipo y pulsaremos en el bitón ejecutar (play) para instalar y ejecutar la aplicación aplicación en el dispositivo conectado.

Open the "platforms/ios/MyApp.xcworkspace" using XCode then go to the build settings. Change the Development Team to your Apple Development team account. Make sure your iPhone or iOS device connected then click the play button to run this Ionic 5 app to the iOS device. And here the final calculator app looks like.

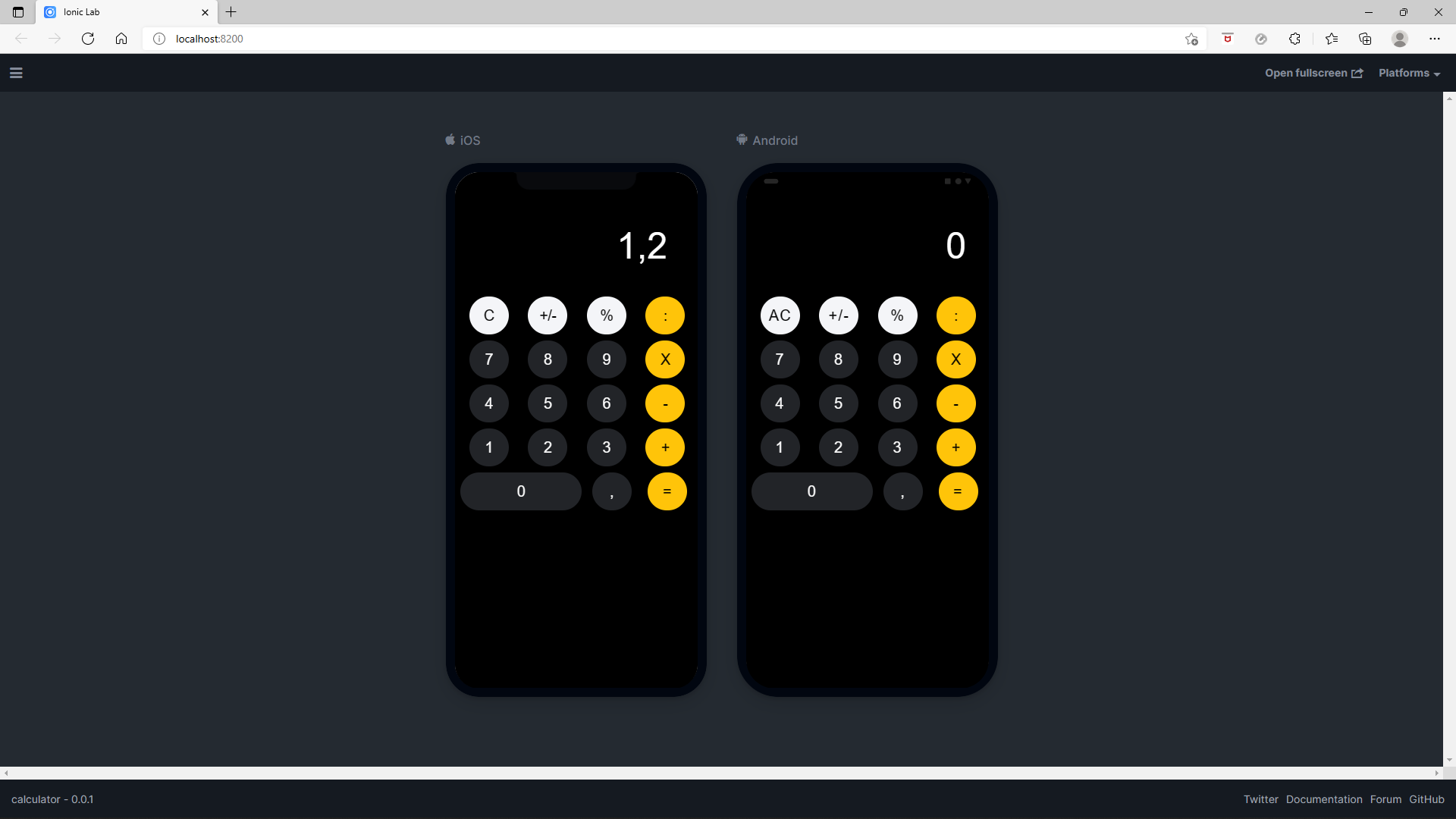


Imagen 2. Vista de la aplicación en in iPhone.

## Ejecutandola aplicación en un Android.

Para ejecutar la aplicación en un dispositivo Android deberemos añadir la plataforma utilizando el siguiente comando:

ionic cordova platform add android

Después conectamos un dispositivo Android al ordenador mediante USB y ejecutamos el siguiente comando para poder instalar y ejecutar la aplicación en un dispositvo Android.

ionic cordova run android

Tras la ejecución del commando aparecerá la aplicación en la pantalla de nuestro dispositivo Android.

A continuación, se muestra la aplicación ejecutada en un dispositivo Android.

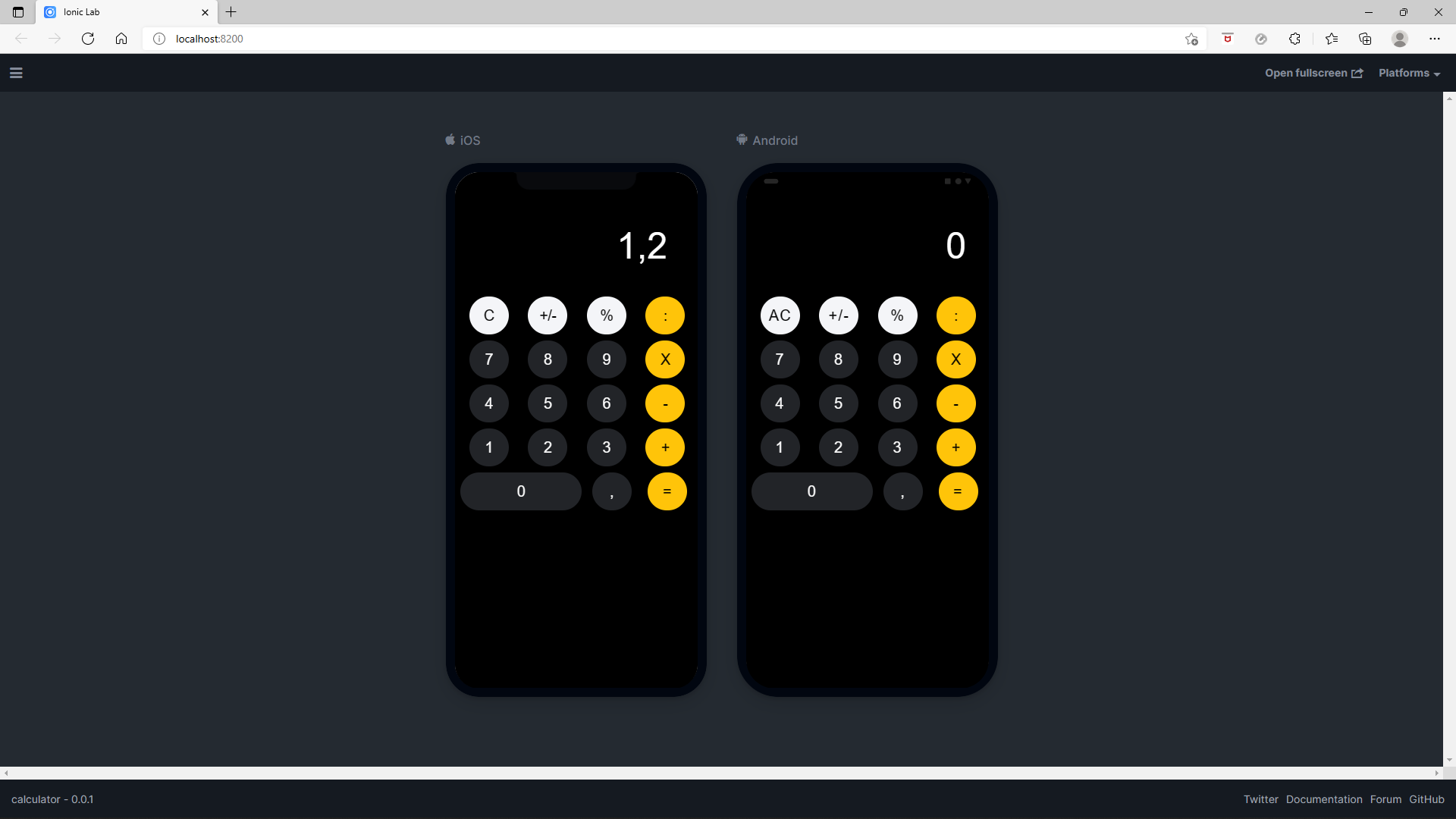


Imagen 3. Aplicación en un dispositivo Android.

1. Si se está ustilizando Windows no es debe escribirse sudo. [↑](#footnote-ref-1)
2. 8100 suele ser la dirección que toma Ionic por defecto, pero puede ser otra. Para saberlo deberemos mirar que dirección local nos indica el terminal después de ejecutar el comando. [↑](#footnote-ref-2)
3. Para crear un paquete que pueda utilizarse en iOS debe estase utilizando un sistemaMacOSX. [↑](#footnote-ref-3)