

Manual TÉCNICO

GESTIÓN PRODUCTIVA 3.1



12 de febrero de 2019

Universidad técnica particular de loja

San Cayetano Alto - Loja

Aprendizaje basado en proyectos:

Aplicación móvil desarrollada en el Framework Ionic

# Información general del proyecto:

Desarrollo de una aplicación móvil mediante el Framework de Ionic, la app consiste en la enseñanza de matemáticas básica para niños en un rango de edad de 5 a 12 años las operaciones básicas que se puede realizar en la aplicación son:

* Suma
* Resta
* Multiplicación
* Mixta (combinación de las tres operaciones anteriores)

Al momento de iniciar con la aplicación, está le sugiere que ingrese el nombre mediante una alerta y por medio de un audio, la aplicación también cuenta con una sección de categorías en donde el estudiante puede escoger la categoría a conveniencia de igual forma, presenta una sección para seleccionar la dificultad que tendrán las operaciones matemáticas. Existe cuatro tipos de niveles que son Fácil, Medio, Profesional y Experto. Después de haber pasado por estas dos secciones se presenta una pantalla con el nombre del jugador, un temporizador, la puntuación y los intentos que tiene el estudiante, para esto se ha establecido un límite de 5 operaciones matemáticas, además el estudiante debe escoger la opción correcta acorde a la operación matemática planteada por la aplicación, en caso que el estudiante escoja la respuesta incorrecta la aplicación le presentara una retroalimentación de la operación matemática con su respectiva solución. Después de haber terminado con todos los intentos el estudiante podrá visualizar su resultado obtenido acorde al número de aciertos y errores, en ese momento el estudiante tiene la opción de volver a jugar nuevamente o simplemente salir del juego e intentarlo en otro momento.

# Objetivos:

* Motivar a los estudiantes de nivel básico de estudio al uso de aplicaciones móviles para fortalecer sus habilidades matemáticas.
* Que los docentes de las escuelas de educación básica hagan uso de aplicaciones móviles orientadas a la educación y fomente el uso de las TIC’s a sus estudiantes desde temprana edad.

# Requisitos o tareas:

Para el desarrollo de una aplicación hibrida es muy importante que el estudiante de Gestión Productiva tenga conocimientos de las siguientes tecnologías:

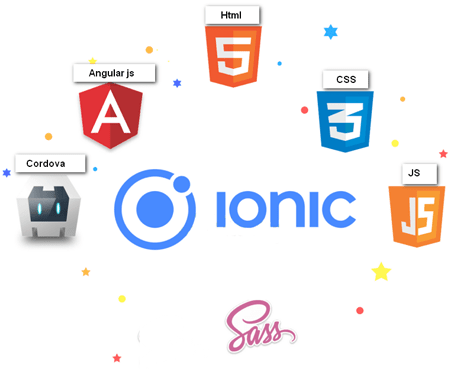


Figura 1. Tecnologías para desarrollar una aplicación en IONIC

* HTML 5
* CSS3
* JavaScript
* IONIC (versión actual).
* Angular (versión actual)
* Cordova
* Además de esto el estudiante debe poseer habilidades de programación.

**Nota:** Las aplicaciones móviles híbridas son una combinación de tecnologías web como HTML, CSS y JavaScript, que no son ni aplicaciones móviles verdaderamente nativas, porque consisten en un WebView ejecutado dentro de un contenedor nativo, ni tampoco están basadas en Web, porque se empaquetan como aplicaciones para distribución y tienen acceso a las APIs nativas del dispositivo.

# Metodología de desarrollo:

# Antes de realizar una aplicación en Ionic es necesario leer su documentación, para estar al tanto de los cambios realizados por cada versión de Ionic, más información en el siguiente enlace (https://ionicframework.com/).

# Instalación de Android SDK (servirá al momento de generar nuestra apk)

# Instalación del framework Ionic.

# Descargar [Node.js](https://nodejs.org/en/) recomendado la versión LTS.

# Instalar Node

# Instalar Ionic con el siguiente comando npm install -g ionic mediante la Shell de Windows o símbolo de sistema.

# Crear tu primer proyecto con el siguiente comando (ionic start myApp tabs), la palabra myApp debe ser reemplazada con el nombre que le quieras asignar a tu proyecto.

# Luego de crear tu proyecto se debe abrir en un editor de código de preferencia, en este caso se utilizó Visual Studio Code, debido a que cuenta con un Shell similar a la de Windows, en donde se puede ejecutar y compilar el proyecto.

# Para ejecutar el proyecto se utiliza los comandos ionic server como se observa en la Figura 2, comando que permite visualizar como quedara la aplicación para una página web, ionic serve –lab permite visualizar la aplicación en las diferentes plataformas de Android, IOS y Windows Phone como se observa en la Figura 3, ionic hace uso del livereload que supervisa los cambios en el sistema de archivos. Tan pronto como guarda un archivo, se procesa según sea necesario y el navegador se actualiza. Aún más fresco, cuando cambias un archivo CSS o una imagen, el navegador se actualiza instantáneamente sin volver a cargar la página.

# NOTA: si al ejecutar ionic serve –lab no se visualiza la emulación del juego, basta con escribir /ionic-lab seguido de <http://localhost:8100> lo cual nos quedaría de la siguiente manera <http://localhost:8100/ionic-lab>

# 

Figura 2. Visualización Web

# 

Figura 3. Visualización Multiplataforma

Después de haber realizado todos los pasos anteriores es momento de proceder con el maquetado, estilos y funcionalidad del juego.

Primeramente siempre se debe tener presente las versiones de las tecnologías utilizadas para el desarrollo de una aplicación hibrida, para conocer el versiona miento se ejecuta el siguiente comando desde la dirección de tu proyecto **ionic info** en mi caso yo he trabajado con las siguientes versiones:

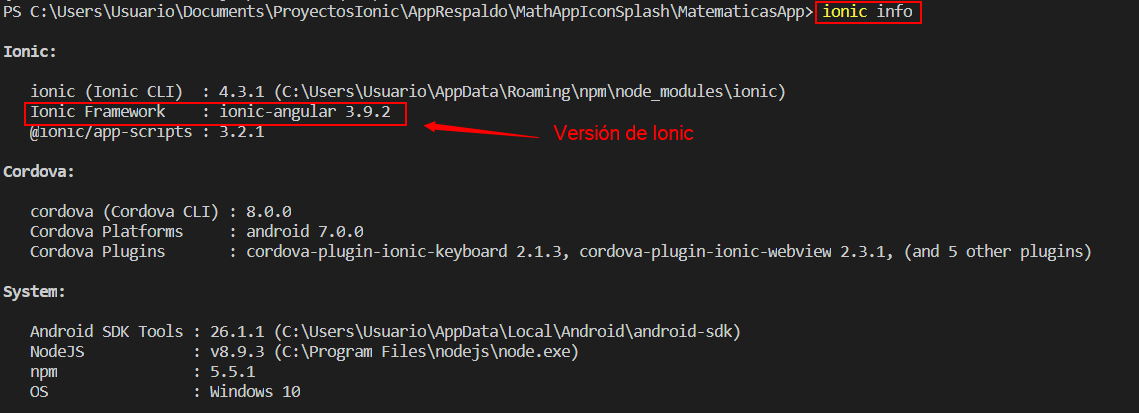


Figura 4. Versiones de las tecnologías aplicadas al desarrollo.

Esto es muy importante al momento de desarrollar un proyecto en ionic ya que cada versión sus módulos varían en ciertos aspectos como su nombre y dirección y trabajan con una versión superior de Angular. Pero eso no es problema porque el equipo de Ionic también pensó en nosotros de tal manera que si quieres continuar con el desarrollo de esta aplicación y vas a trabajar con una versión superior a esta te recomiendo revisar el siguiente enlace que trata de como migrar una aplicación de una versión anterior a otra posterior.

<https://moduscreate.com/blog/upgrading-an-ionic-3-application-to-ionic-4/>

# Estructura del proyecto

En la versión de Ionic 3 nos presenta la siguiente jerarquización de carpetas:

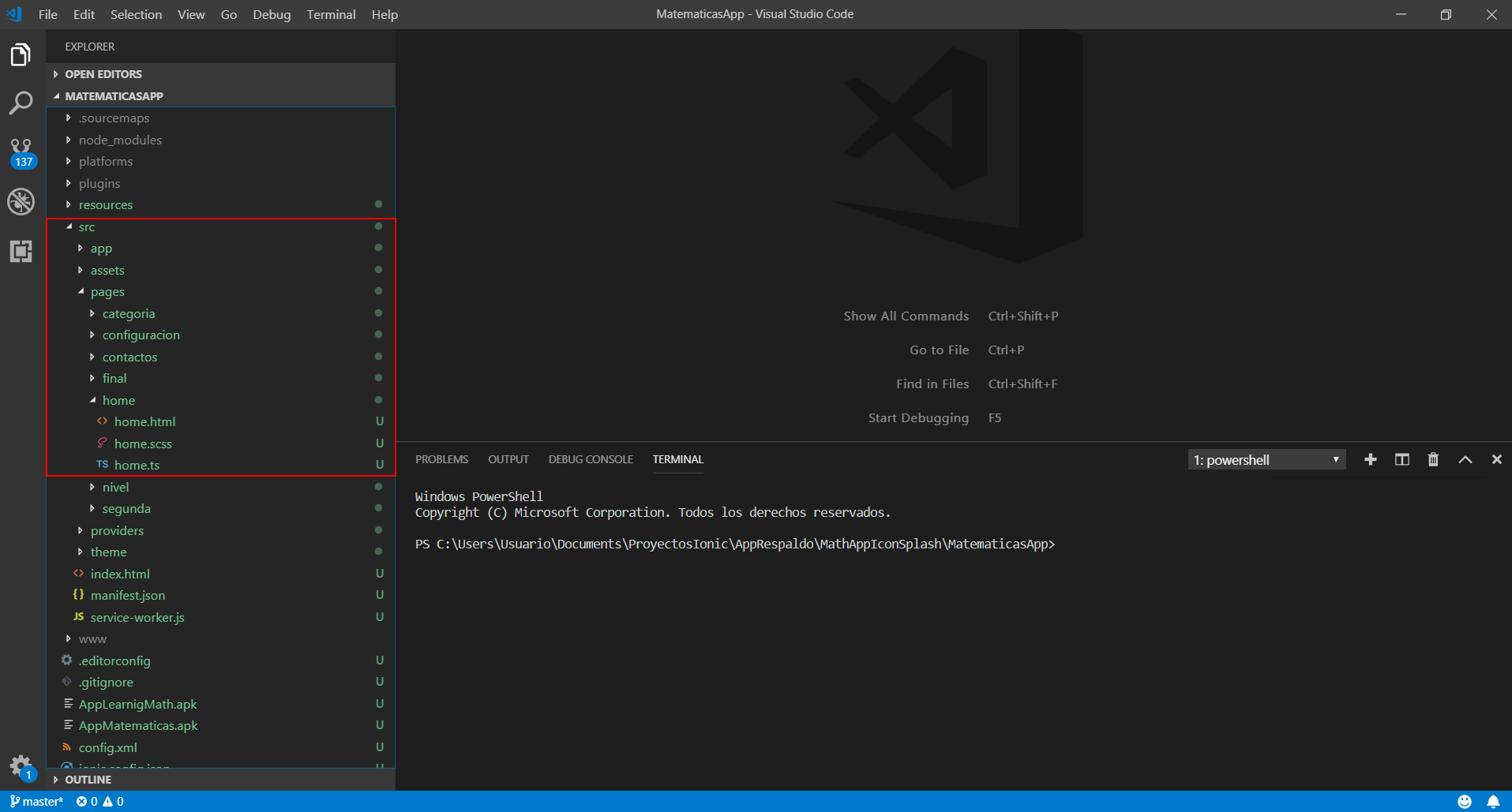


Figura 5. Estructura de un proyecto en IONIC 3

Como se puede observar en la figura 5, después de crear nuestro primer proyecto nos presenta la siguiente jerarquía de carpetas a lo largo de esta documentación se explicará la función de cada carpeta que se observa en la figura.

Empezaremos explicando la carpeta **src,** al abrir esta carpeta se puede observar un subconjunto de carpetas y archivos, dentro de ese subconjunto se puede observar la carpeta **pages** que es la carpeta que contiene las subcarpetas que contiene la parte del maquetado, estilos y funcionalidad de nuestra aplicación. Como ejemplo de esto se puede observar la carpeta home la misma que contiene los archivos con extensión .html, .ts, .scss.

# PÁGINA HOME

[home.html.-](http://home.html.-) este archivo contiene toda la parte de maquetación de la página principal, la estructura del archivo es como se observa en el siguiente ilustración 1 del 1 – 13 comprende la cabecera de la página, del 17-55 comprende todo el cuerpo de la página, aquí difiere un poco la estructura ya que se utiliza etiquetas propias del framework Ionic , en la línea 4 se consume la fuente de letra, de la línea 8-11 se implementa el **botón tres líneas** y el **título** de la página para el despliegue del menú, en la línea 19-21se ingresa una imagen con el logo de la escuela que sería la **subcabecera** de la página, en la línea 25-44 se implementa una etiqueta secction que contiene una etiqueta para el título **Modo de Juego** en la línea 27**,** en la línea 29-30 seingresa una imagena la pantalla principal de los tres niños pensando, con un párrafo debajo de elllos, en la line 33-35 se inserta una imagen que simula ser el avatar del juego, en la línea 40-41 se ingresa una caja de texto en la cual se debe escribir el nombre del jugador y con el botón **START** se envía el nombre a las páginas siguientes para validar el dato de entrado y verificar que el campo de texto se llenado caso contario no podrá dirigirse a la siguiente página del juego aquí se ha hecho uso de [ngModel] básicamente se trata de un enlace, entre algo que tienes en la definición del componente con un campo de formulario del template (vista) del componente, para mayor información revise el siguiente link que trata todo sobre ngModel <https://goo.gl/o7Y4Kh> y finalmente en la línea 48-56 se inserta un pie de página, con un logo de la universidad y sus respectivos derechos de copy right (footer).

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | <!--Inicio del header-->  <ion-header>  <!--Consume una fuente de letra -->  <link href="https://fonts.googleapis.com/css?family=Handlee" rel="stylesheet">  <ion-navbar>  <!-- <ion-title>  </ion-title>-->  <button ion-button menuToggle>  <ion-icon name="menu"></ion-icon>  </button>  <ion-title>Home</ion-title>  </ion-navbar>  </ion-header>  <!--Fin del header-->  <!--Inicio del content-->  <ion-content padding class="bodyHome">  <div id="encabezado">  <img src="../assets/imgs/logoescuela.png" alt="">  </div><br>  <!--Fin del encabezado-->  <!--Inicio del article-->  <article class="section">  <span>  <h2><b>Modo de Juego</b></h2>  </span>  <img src="../assets/imgs/niñosNum.png" alt="" id="kids">  <p class="parrP"><b>Adivina la respuesta de las operaciones matemáticas<br>¿Serás capaz de lograrlo?</b> </p>  <!--Inicio del avatar -->  <div id="avatar" class="section">  <img alt="" class="img-circle" src="../assets/imgs/animal.png">  </div>  <!--Inicio del avatar -->  <!--ngModel permite establecer un enlace con el campo del formulario con lo que se realiza en el método createMyForm(), esto permite validar el campo en caso que el usuario  desee emprezar con el juego sin escribir su nombre-->  <input placeholder="Escriba su nombre..." type="text" [(ngModel)]="nombre" /><br />  <input (click)="nextPage()" type="submit" [disabled]="!nombre" class="submit" value="START"/>  <!--Fin del formulario -->  </article>  <!--Fin del article -->  <!--Inicio del footer -->  <footer>  <ul>  <li><img src="../assets/imgs/logo1.png" alt="https://www.utpl.edu.ec/"></li>  <li class="copyright"> <b>**&copy;** Copyright 2018. Universidad Técnica Particular de Loja</b></li>  </ul>  </footer>  <!--Fin del footer -->  </ion-content>  <!--Fin del content--> | |

Código Fuente 1: Maquetado la página home

[home.ts.-](http://home.ts.-) este archivo contiene la funcionalidad de la página, como sus respectivos módulos, constructor, variables y métodos necesarios para el correcto funcionamiento de la página, en la figura 7 se puede observar su estructura.

En las líneas 1-5 se realiza la importación de las librerías necesarias, que establecen el enlace con home.html, obtener los parámetros, permitir la lectura de texto a voz, validar y crear formularios y controlar alertas, en las líneas 12-77 contiene toda la clase de **HomePage,** en la línea 14-15 se declara variables al muy interesante que ocurre aquí es que la variable nombre almacena el nombre que ingresamos en la página principal es es gracias al ngModel que se lo menciona anteriormente en donde basta al ngModel que se encuentra en la en home.html inicializarlo con el mismo nombre de la variable declarada en home.ts ([ngModel]=”nombre”), en la línea 18-31 se parametriza el constructor y dentro del mismo se inicializa los métodos que se desea que inicien junto con el constructor al ejecutar la aplicación. En la línea 34-40 se declara un método para generar una alerta al incio del juego estableciendo su respectivo título, texto y botón y al final aparece con una alerta que se presentan en las páginas web como se observa en la figura 6.

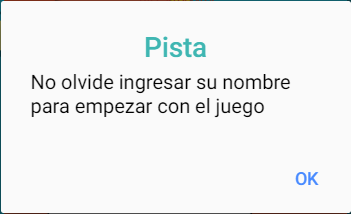


Figura 6. Alerta del juego

En las líneas 43-55 se establece un método para el reconocimiento de texto a voz que hace uso del módulo Async para trabajar con JavaScript asíncrono.

Nota: para instalar la librería de TextToSpeech en un proyecto se recomienda leer el siguiente manual mm <https://goo.gl/xWo4LS>. En las líneas 62-65 se crea un método para crear el campo de texto con su debida restricción de ser un campo obligatorio de llenar y por último en la página 71-73 se crea un método para realizar la navegación entre páginas con parámetros, en este caso se observa que se redirección la página **CatergoriasPage** junto con el parámetro nombre que almacena el nombre del jugador. Este método se activa cuando se activa el evento (click) desde home.thml línea 4.

Nota: Se puede realizar la navegación de páginas de dos formas mediante el método **setRoot** que una vez dirigido a la página siguiente no permite que se vuelva a la página anterior y el método **push** que es todo lo contario al método anterior.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | // Importación de librerias, se importa la libreria unicamente a utilizar  import { Component } from '@angular/core';  import { NavController, NavParams, AlertController } from 'ionic-angular';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { TextToSpeech, TTSOptions } from '@ionic-native/text-to-speech';  @Component({  selector: 'page-home',  templateUrl: 'home.html'  })  export class HomePage {  //Declaración de variables  myForm: FormGroup;  nombre: string;  // Constructor parametrizado  constructor(  public navCtrl: NavController,  public navParams: NavParams,  public formBuilder: FormBuilder,  private textToSpeech: TextToSpeech,  private alertCtrl: AlertController) {  // Estos métodos se ejecutan al iniciar la aplicación  this.sayText();  this.presentAlert();  this.myForm = this.createMyForm(); // almacena el método para crear el formulario  }  //Metodo para presentar una alerta al inicio del juego  presentAlert() {  let alert = this.alertCtrl.create({  title: 'Pista', // titulo de la alerta  subTitle: 'No olvide ingresar su nombre para empezar con el juego', // texto de la alerta  buttons: ['OK'] // botón de la alerta  });  alert.present(); //Visualización de la alerta en la pantalla principal  }  //Función para agregar TextToSpeech (permite que se lea un texto a voz)  async sayText() {  try {  const options: TTSOptions = {  text: "Escribe tu nombre para empezar el juego", //texto que emitira la aplicación  rate: **0.9**,// velocidad de la lectura de texto a Vox  locale: 'es-Es'  }  await this.textToSpeech.speak(options);  }  catch (e) {  console.error(e);  }  }  //Metodo opcional para guardar el nombre del jugador y presentarlo por consola  saveData1() {  console.log(this.nombre);  }  //Metodo para crear un formulario y especificar que el campo es requerido  private createMyForm() {  return this.formBuilder.group({  name: ['', Validators.required] //campo requerido  });  }  //Metodo para pasar a la siguiente pagina, enviando el nombre ingresado al inicio  nextPage() {  this.navCtrl.setRoot("CategoriaPage", { nombre: this.nombre });  }  } | |

[home.scss.-](http://home.scss.-) este archivo contiene los estilos o animaciones que contiene la página, por ejemplo colores, imágenes y bordes de la página.

Con la fusión de estos tres archivos se ha logrado realizar la primera página del juego como se observa en la figura 7.



Figura 7. . Página principal del juego

# PÁGINA CATEGORIA

[categoria.html.-](http://home.scss.-) En este archivo al igual que el anterior archivo con la misma extensión se realiza la maquetación de la página categoría, esta página le permite al usuario escoger una categoría según el tipo de operaciones que desee realizar, se presentan cuatro opciones Suma, Resta, División y Mixta (combinación de las tres operaciones matemáticas). En la las líneas 2-21 comprende el **encabezado de la página**, en donde en la línea 5-19 contiene un navbar que actúa como la barra de navegac-ión, que viene en la línea 7-18 con una lista no ordenada para colocar un botón de tres líneas (8-10), un título (11-13) y un icono home (14-16). En las líneas 21-54 se encuentra el cuerpo de la página que contiene una etiqueta article (25-53) la misma que contiene un título (29-32), una etiqueta div (30-52) que contiene los cuatro botones para seleccionar las respectivas operaciones de Suma (32-35), Resta (36-39), Multiplicación (40-44) y Mixta (45-51).

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | <!—Página para la sellecion de ategorías del Juego🡪  <ion-header>  <!—API para consumir un estilo de letra desde la dirección indicada🡪  <link href=”https://fonts.googleapis.com/css?family=Handlee” rel=”stylesheet”>  <ion-navbar>  <!—Lista desordenada que contien el bóton tres atego, titulo y el icono home🡪  <ul>  <li class=”men”><button ion-button menuToggle class=”toggle”>  <ion-icon name=”menu”></ion-icon>  </button> </li>  <li class=”titleCat”>  <ion-title>Categorias</ion-title>  </li>  <li class=”home”><button (click)=”goHomeC()” class=”bntH”>  <ion-icon name=”home” class=”iconH”></ion-icon>  </button></li>  </ul>  </ion-navbar>  </ion-header>  <!—Cuerpo de la página 🡪  <ion-content padding class=”bodyCategoria”>  <!—Contiene los botones para selccionar las ategorías del juego🡪  <article class=”section”>  <!—Titulo del cuerpo de la página🡪  <span>  <h2><b>Seleccione Categoría</b></h2>  </span>  <div class=”categorias”>  <div class=”cat1”>  <button ion-button block icon-end round outline (click)=”bntSuma()”>  <b>Suma</b>(<ion-icon name=”add”></ion-icon>)</button>  </div>  <div class=”cat1”>  <button ion-button block icon-end round outline (click)=”bntResta()”>  <b>Resta</b>(<ion-icon name=”md-remove”></ion-icon>)</button>  </div>  <div class=”cat1”>  <button ion-button block icon-end round outline (click)=”bntMultiplicacion()”>  <b>Multiplicación</b>(<ion-icon  name=”md-close”></ion-icon>)</button>  </div>  <div class=”cat1”>  <button ion-button block icon-end round outline (click)=”bntMixta()”>  <b>Mixta</b> (<ion-icon name=”add”>,</ion-icon>  <ion-icon name=”md-remove”>,</ion-icon>  <ion-icon name=”md-close”></ion-icon>)  </button>  </div>  </div>  </article>  </ion-content> | |

[categoria.ts.-](http://home.scss.-) El siguiente archivo contiene las respectivas librerías, componentes, clase, constructor, variables y métodos que permitirán dar funcionalidad a la página especialmente a los botones que el usuario seleccione, aquí se explicara lo más relevante como los métodos que le dan función a los botones igual que en el archivo anterior estos activan el evento (click) para llamar a la función desde el archivo categoría html a su correspondiente método que se encuentra en este archivo, como se puede observar en la línea 13-14 se declaran las variables, la variable nombre sirve para obtener el dato que se envió desde la página HomePage como se observa en la línea 16 y la variable operación permitirá almacenar el valor seleccionado por el usuario como se observa en la líneas 33, 37, 41 y 45 por ejemplo si selecciona la categoría suma se almacenara el valor de “suma” en la variable, en la línea 25-27 se declara un método para volver a la página principal que este método se activa cuando se presiona en el icono **home** que se visualiza en la página **CategoriaHome,** en las líneas 32-35 se declara el métodos **bntSuma(),** que permite almacenar el valor proveniente del botón suma y permite re direccionarse a la página Siguiente que en este caso sería NivelPage enviando ya nos solo el nombre del jugar sino que también envía el valor del botón como se observa en la línea 34,38,42 y 46, los siguientes métodos se implementan un lógica similar por lo que sería redundante explicar en lo único que difieren es en el valor que se envía por cada operación seleccionada.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | **import** { Component } from '@angular/core';  **import** { IonicPage, NavController, NavParams } from 'ionic-angular';  **import** { HomePage } from '../home/home';  @IonicPage()  @Component({  selector: 'page-categoria',  templateUrl: 'categoria.html',  })  **export** **class** CategoriaPage {  nombre = '';  operacion: **string**;  **constructor**(**public** navCtrl: **NavController**, **public** navParams: **NavParams**) {  **this**.nombre = navParams.get('nombre'); // aqui se recupera el parametro enviado desde la pagina HomePage (nombre del jugador)  }  ionViewDidLoad() {  console.log('ionViewDidLoad CategoriaPage');  }  //Metodo para volver ala pagina principal  goHomeC() {  **this**.navCtrl.setRoot(HomePage);  }  //Métodos o funciones para el paso de parametros de una pagina a otra en este caso se pasa el nombre del jugador y el tipo de operacion  // estos parametros son pasados a la pagina NivelPage  bntSuma() {  **this**.operacion = "suma";  **this**.navCtrl.push("NivelPage", { nombre: **this.nombre**, operacion: **this.operacion** });  }  bntResta() {  **this**.operacion = "resta";  **this**.navCtrl.push("NivelPage", { nombre: **this.nombre**, operacion: **this.operacion** });  }  bntMultiplicacion() {  **this**.operacion = "multiplicacion";  **this**.navCtrl.push("NivelPage", { nombre: **this.nombre**, operacion: **this.operacion** });  }  bntMixta() {  **this**.operacion = "mixta";  **this**.navCtrl.push("NivelPage", { nombre: **this.nombre**, operacion: **this.operacion** });  }  } | |

[categoria.scss.-](http://home.scss.-) en este archivo se encuentran los estilos y formatos que presenta la página como la imagen de fondo, el color de los contenedores, los estilos de botones entre otras cosas utilizadas para presentar una visualización agradable al usuario.

Con la fusión de estos tres archivos se ha logrado realizar la primera página del juego como se observa en la figura 8.

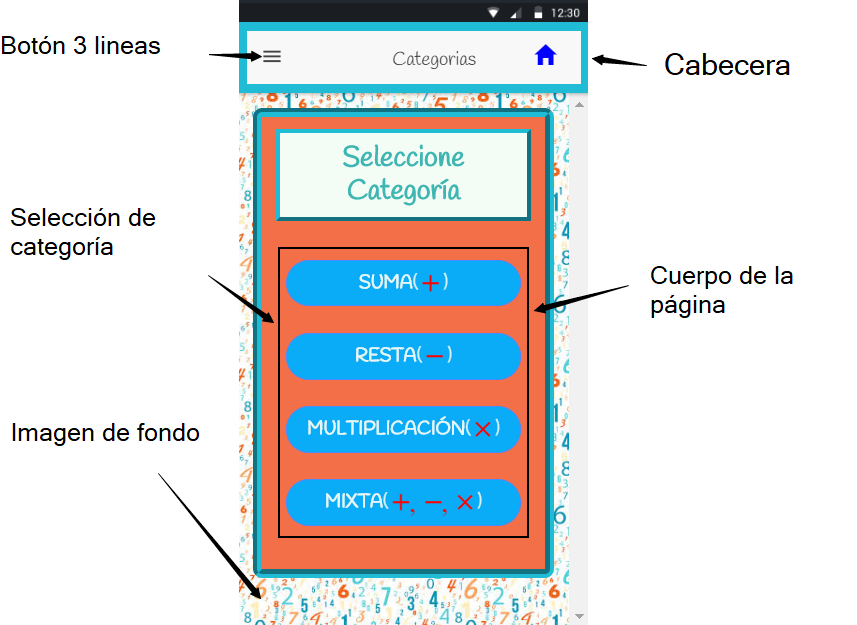


Figura 8. Página Categoría del juego

# PÁGINA NIVEL

[nivel.html.-](http://home.scss.-) este archivo es muy similar al archivo categoría.html que se lo explicó anteriormente, debido a que presentan una interfaz de usuario similar únicamente difieren en la selección de opciones, ya que en esta página se debe elegir el nivel del juego, el mismo que presenta una complejidad según el nivel selecciona, Nivel Fácil le permitirá al usuario realizar operaciones en un rango comprendido de números enteros de (1 a 10), el Nivel Medio permitirá al usuario realizar operaciones en un rango comprendido entre números enteros de (10 a 20), el Nivel Experto permitirá al usuario realizar operaciones matemáticas en un rango comprendido de (20 a 40) y finalmente se presenta la opción Nivel Profesional que permitirá que el usuario realice operaciones con un mayor grado de complejidad con números comprendidos en un rango de (50 a 10) números enteros.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <!--  Generated template for the NivelPage page.  See http://ionicframework.com/docs/components/#navigation for more info on  Ionic pages and navigation.  -->  <ion-header>  <link href="https://fonts.googleapis.com/css?family=Handlee" rel="stylesheet">  <ion-navbar>  <ul>  <li>  <ion-title>Nivel</ion-title>  </li>  <li class="homeN"> <button (click)="goHomeN()" class="bntH">  <ion-icon name="home" class="iconH"><br></ion-icon>  </button>  </li>  </ul>  </ion-navbar>  </ion-header>  <ion-content padding class="bodyNivel">  <!-- <h2><b>Seleccione Nivel de Complejidad</b></h2>-->  <article class="section">  <span>  <h2><b>Nivel de Juego</b></h2>  </span>  <div class="niveles">  <!--Botones para selccionar la complejidad del juego-->  <div class="niv"> <button ion-button block icon-start round outline (click)="bntLevel1()"><b>Fácil(1-10)</b></button></div>  <div class="niv"><button ion-button block icon-start round outline (click)="bntLevel2()"><b>Medio(10-20)</b></button></div>  <div class="niv"><button ion-button block icon-start round outline (click)="bntLevel3()"><b>Profesional(20-40)</b></button></div>  <div class="niv"><button ion-button block icon-start round outline (click)="bntLevel4()"><b> Experto(50-100)</b></button></div>  </div>  </article>  </ion-content> | |

[nivel.ts.-](http://home.scss.-) este archivo es muy similar al archivo categoría.ts, únicamente difieren en sus variables ya que como se observa en la línea 36, 41, 46 y 51 aquí se almacena el valor obtenido en el constructor de la clase y ahora se envía tres parámetros a SegundaPage que son nombre del usuario, categoría de juego y nivel de juego, igual que el archivo categoria.ts, estos parámetros son enviados por medio del constructor hacia la siguiente página que es SegundaPage

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | **import** { Component } from '@angular/core';  **import** { IonicPage, NavController, NavParams } from 'ionic-angular';  **import** { HomePage } from '../home/home';  /\*\*  \* Generated class for the NivelPage page.  \*  \* See https://ionicframework.com/docs/components/#navigation for more info on  \* Ionic pages and navigation.  \*/  @IonicPage()  @Component({  selector: 'page-nivel',  templateUrl: 'nivel.html',  })  **export** **class** NivelPage {  nombre: '';  operacion: '';  nivel: **number**;  **constructor**(**public** navCtrl: **NavController**, **public** navParams: **NavParams**) {  //this.userObject= this.navParams.data;  **this**.nombre = navParams.get('nombre');  **this**.operacion = navParams.get('operacion');  }  ionViewDidLoad() {  console.log('ionViewDidLoad NivelPage');  }  //Metodos para cambiar el nivel de juego y pasar los parametros a SegundaPage (nombre, oprecion escogida y nivel seleccionado)  bntLevel1() {  **this**.nivel = **1**;  **this**.navCtrl.push("SegundaPage", { nombre: **this.nombre**, operacion: **this.operacion**, nivel: **this.nivel** });  }  bntLevel2() {  **this**.nivel = **2**;  **this**.navCtrl.push("SegundaPage", { nombre: **this.nombre**, operacion: **this.operacion**, nivel: **this.nivel** });  }  bntLevel3() {  **this**.nivel = **3**;  **this**.navCtrl.push("SegundaPage", { nombre: **this.nombre**, operacion: **this.operacion**, nivel: **this.nivel** });  }  bntLevel4() {  **this**.nivel = **4**;  **this**.navCtrl.push("SegundaPage", { nombre: **this.nombre**, operacion: **this.operacion**, nivel: **this.nivel** });  }  //Method para volver a la página principal del juego  goHomeN() {  **this**.navCtrl.setRoot(HomePage);  }  } | |

[nivel.scss.-](http://home.scss.-) en este archivo se encuentran los estilos y formatos que presenta la página como la imagen de fondo, el color de los contenedores, los estilos de botones entre otras cosas utilizadas para presentar una visualización agradable al usuario.

Con la fusión de estos tres archivos se ha logrado realizar la primera página del juego como se observa en la figura 9.

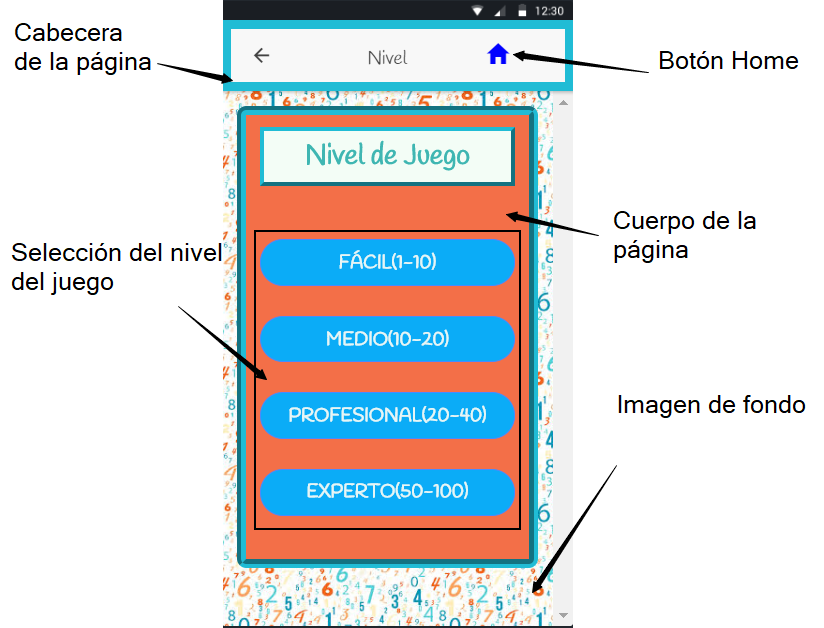
****

Figura 9. Página de niveles del juego

**PÁGINA SEGUNDA**

[nivel.html.-](http://home.scss.-) En este archivo se realiza la maquetación de la ventana que sirve para organizar la ubicación de cada uno de los elementos a utilizar, como podemos observar en las líneas 1-28 que comprende la cabecera de la pantalla de juego en donde se ha utilizado una lista no ordenada (7-19) para colocar una imagen de reloj (9-10) que presentará el tiempo que el usuario se demora en resolver las operaciones matemáticas, en la línea 12 se indica el puntaje obtenido por el estudiante de acuerdo a las respuestas correctas, en línea 14-16 se implenta un botón con su respectivo icono de home para volver a la ventana principal (HomePage), en la línea 21-23 se presenta un avatar que hace referencia al nombre del jugar ingresado al inicio del juego y finalmente en la línea 24 se presenta el número de operaciones matemáticas que debe realizar en este caso sería 5 operaciones a realizar.

En la línea 20-91 se encuentra el cuerpo de la página la misma que contiene el mensaje de inicio del juego, la operación a realizar, las opciones que debe elegir y el botón para realizar la siguiente operación.

En la línea 32-38 se implementa una etiqueta de tipo ion-card que se presentara únicamente cuando aún no se ha seleccionado ninguna opción aún, en la línea 39-48 se ha hecho uso una vez más de la etiqueta ion-card pero esta vez se presentará únicamente cuando el usuario seleccione la respuesta correcta, caso contrario se presentará la etiqueta que se encuentra en las líneas 50-58 explicando la respuesta correcta que el usuario tenía que seleccionar, en las líneas 60-84 mediante una etiqueta article se ha colocado tres etiquetas input (62-64) de tipo number para los números y tipo text para el signo, en las líneas 66-82 se ha utilizado la etiqueta button para colocar respuestas aleatorias en los 4 botones y cada una activa un evento mediante (click)=”Funcion()”, que según la respuesta correcta llama al método que se encuentra en Segunda.ts para su respectiva verificación de la respuesta. Finalmente en las líneas 88-90 se presenta dos botones que aparecen en la pantalla del juego según la condición en el caso que aún no se hayan desarrollado aun todas la operaciones el botón Siguiente aparecerá constantemente, una vez finalizadas las operaciones aparecerá un boto finalizar que permite dirigirse a la Página FinalPage que contiene el los resultados del juego.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91 | <ion-header>  <link href="https://fonts.googleapis.com/css?family=Handlee" rel="stylesheet">  <ion-navbar>  <ion-title>  </ion-title>  <!--Lista no ordenada para mostrar el tiempo, el nombre del jugador, puntaje e intentos-->  <ul>  <li class="s">  <ion-icon class="reloj"><img class="relojArena" src="../assets/icon/Reg.png" alt=""></ion-icon>={{minutos}}:{{segundos}}  </li>  <li>Puntaje: {{score}}</li>  <li class="homeS"><button (click)="goHomeS()" class="bntH">  <ion-icon name="home" class="iconH"></ion-icon>  </button>  </li>  </ul>  <ul>  <li class="s">  <ion-icon class="nino"><img class="avatarS" src="../assets/icon/avatar.png" alt=""></ion-icon>= {{nombre}}  </li>  <li>Intentos: {{intento}}</li>  </ul>  </ion-navbar>  </ion-header>  <!--Cuerpo de la pagina en donde se muestran las operaciones matemáticas y retroalimentación del juego-->  <ion-content padding class="bodySegunda">  <!--Mensaje de inicio-->  <div class="welcome">  <ion-card \*ngIf="controlCorrecta == false && controlIncorrecta == false">  <ion-card-content class="contS">  Pongamos a prueba tus conocimientos matemáticos  </ion-card-content>  </ion-card>  </div>  <!--Esta tajeta aparecera si se cuemple la condicion de respuesta correcta-->  <ion-card \*ngIf="controlCorrecta == true">  <ion-card-header>  ¡¡¡Enhorabuena!!!  </ion-card-header>  <ion-card-content>  <img class="imagenAcertada" src="../assets/imgs/aplausos.gif" />  Has acertado, la respuesta es {{ numSum }}  </ion-card-content>  </ion-card>  <!--Esta tajeta aparecera si se cuemple la condicion de respuesta incorrecta-->  <ion-card \*ngIf="controlIncorrecta == true">  <ion-card-header>  ¡¡¡Respuesta Incorrecta!!!  </ion-card-header>  <ion-card-content>  <img class="imagenAcertada" src="../assets/imgs/tenor.gif" />  La respuesta correcta de {{num1}} {{signo}} {{num2}} es igual {{numSum}}  </ion-card-content>  </ion-card>  <!--Aqui se presentan las operaciones matemáticas con sus 4 opciones -->  <article class="section1">  <input type="number" [(ngModel)]="num1" [disabled]="true" placeholder="Numero 1">  <input type="text" [(ngModel)]="signo" [disabled]="true" placeholder="Signo">  <input type="number" [(ngModel)]="num2" [disabled]="true" placeholder="Numero 2"><br>  <div class="bloque1">  <button [disabled]="controlIncorrecta" [disabled]="controlCorrecta" ion-button (click)="comprobarNumeroSumaBotones1()"  name="bnt1">{{bnt1}}</button>  <button [disabled]="controlIncorrecta" [disabled]="controlCorrecta" ion-button (click)="comprobarNumeroSumaBotones2()"  name="bnt2">{{bnt2}}</button>  </div>  <div class="bloque1">  <button [disabled]="controlIncorrecta" [disabled]="controlCorrecta" ion-button (click)="comprobarNumeroSumaBotones3()"  name="bnt3">{{bnt3}}</button>  <button [disabled]="controlIncorrecta" [disabled]="controlCorrecta" ion-button (click)="comprobarNumeroSumaBotones4()"  name="bnt4">{{bnt4}}</button>  </div>  </article>  <!--Estos botones aparecen si se cumeple con la condición requerida-->  <!-- <button \*ngIf="bandera == true && intento > 0" ion-button block round (click)="reinicia1()">Siguiente</button>-->  <button id="bntNextFin" \*ngIf="bandera == true && intento > 0" ion-button block round (click)="reinicia1()">Siguiente</button>  <button id="bntNextFin" \*ngIf="intento == 0" ion-button block (click)="finalizar()">Finalizar</button>  </ion-content> | |

[segunda.ts.-](http://home.scss.-) Este archivo contiene la líneas de código que permiten la funcionalidad del juego, como ya se ha explicado en archivos diferentes las primeras líneas son librerías, módulos y componentes propios de Angular, asi que empezaremos explicando desde la línea 12-33 donde se declara e inicializa variables en la donde se puede observar lo que realiza cada una de ellas cuál es su función dentro del juego. En las líneas 35-51 se crea un constructor con sus respectivos parámetros que permiten el paso de parámetros, obtener parámetros y presentar alertas, en la línea 75-81 se ha creado un método para generar el tiempo que un usuario se demora en realizar las 5 operaciones, en las líneas 86-145 se han creado métodos para comprobar el valor que posee el botón en caso de ser correcto se le agregará un puntaje de 150 y si no lo es pues no aumentara el contador, también permite saber el número de operaciones que va realizando el usuario. En la línea 148-150 se ha establecido un método para generar números aleatorios y ser almacenados en las variables num1 y num2 mismos valores que se presentarán en la interfaz del juego, en las líneas 155-167 se ha creado un método para reconocer el nivel del juego seleccionado por el usuario y en dependencia de eso se le asigna un rango de números, en las líneas 169-174 se ha creado un método para generar operaciones de suma, resta y multiplicación de manera aleatoria en este caso nos permite realizar operaciones mixtas cada vez que se seleccione la categoría mixta que posee una integración de estas tres operaciones, En las 177-187 se ha creado un método para reconocer que operación ha seleccionado el usuario y así mediante la aplicación establecer operaciones acordes a la categoría seleccionada, en las líneas 190-211 va relacionada con el método creado en las líneas 177-187 ya que en dependencia del signo lógico se establece la operación a realizar. En las líneas 214-224 se crea un método para asignar el valor a los botones de opciones según el nivel seleccionado, en las líneas 227-255 se crea un método para generar las respuesta correcta de la operación en forma aleatoria, en las lineas 258-264 se crea un método para controlar el número de operaciones que debe realizar el usuario, en las líneas 267-273 se crea un método que tiene relacione con el botón siguiente que permite el reinicio de variables y la generación de valores aleatorios. Finalmente en las lineas 272-286 se crea un método para pasar parámetros a la página FinalPage que presenta los resultados del usuario y que tiene relación con el botón finalizar.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283 | **import** { Component } from '@angular/core';  **import** { IonicPage, NavController, NavParams, AlertController } from 'ionic-angular';  **import** { HomePage } from '../home/home';  @IonicPage()  @Component({  selector: 'page-segunda',  templateUrl: 'segunda.html',  })  **export** **class** SegundaPage {  //Declaración de variables, para el lamcenamiento de sus respectivos valores  minutos: **number**; // Almacena los minutos del juego  segundos: **number**;// Alamcena los segundos del juego  nombre = ''; // Almacena el nombre del jugador  operacion: **string** = ''; //Alamcena el signo de la operacion matemática  nivel: **number**; //Alamacena el nivel del juego  bandera = **false**; //Cambia de estado cada vez que entra al metodo para comprobar el valor de los botones  controlCorrecta = **false**; //Cambia de estado cuando selcciona un boton con la respuesta incorrecta  controlIncorrecta = **false**;// Cambia de estado cuando selcceiona un boton con la respuesta correcta  controlAsignación: **number** = **1**; //Sirve como un contador para que los valores de los botones sean aleatorios  score: **number** = **0**; //Alamacena los puntos que obtiene el usuario  intento: **number** = **5**; // Numero de operaciones que se deben realizar  num1: **number**; //Alamacena el primer valor aleatorio  num2: **number**; //Alamacena el segundo valor aleatorio  numSum: **number**; // Alamacena la respuesta segun la operación realizada entre los dos números  signo: **String**; //Alamacena el valor del signo de operación a realizar  bnt1: **number**; //Alamacena el valor aleatroio  bnt2: **number**;//Alamacena el valor aleatroio  bnt3: **number**;//Alamacena el valor aleatroio  bnt4: **number**;//Alamacena el valor aleatroio  correcta: **number** = **0**; //Almacena el valor de las respuestas correctas  incorrecta: **number** = **0**; //Almacena el valor de las respuestas incorrectas  **constructor**(  **public** navCtrl: **NavController**, //Permite realizar el paso de valores de un constructor a otro  **public** navParams: **NavParams**, //Permite obtener las variables provenientes de otras páginas  **private** alertCtrl: **AlertController** //Visualización de alerta (aun no implementada en esta versión)  ) {  //Se almacena los valores del jugador que ha pasado de ventana en ventana  **this**.nombre = navParams.get('nombre');  **this**.operacion = navParams.get('operacion');  **this**.nivel = navParams.get('nivel');  //Se almacena el valor aleatorio para las operaciones  **this**.num1 = **this**.asignacionNivelnum();  **this**.num2 = **this**.asignacionNivelnum();  //Se reinicia el temporizador  **this**.resetTimer();  **this**.asignacionLevelBnt();  setInterval(() => **this**.tick(), **1000**);  }    ionViewDidLoad() {  console.log('ionViewDidLoad SegundaPage');    }  presentAlert() {  **let** alert = **this**.alertCtrl.create({  title: 'Respuesta', // titulo de la alerta  subTitle: 'RETROALIMENTACION', // texto de la alerta  buttons: ['OK'] // botón de la alerta  });  alert.present(); //Visualización de la alerta en la pantalla principal  }  //Método para resetear el tiempo (opcional aún no está implementado)  resetTimer() {  **this**.minutos = **0**;  **this**.segundos = **0**;  }  //controlar el tiempo cuando segundos es igual a 59 vuelve a cero y aumenta los minutos  tick() {  **if** (++**this**.segundos == **59**) {  **this**.minutos = **this**.minutos + **1**;  **this**.segundos = **0**;  }  }  //Comprobación de respuesta por cado botón que contenga la respuesta correcta, se asina puntaje caso contrario el puntaje permanece igual  comprobarNumeroSumaBotones1() {  **if** ((**this**.bnt1) == **this**.numSum) {  **this**.controlCorrecta = **true**;  **this**.score = **this**.score + **150**;  **this**.correcta = **this**.correcta + **1**;  } **else** {  **this**.controlIncorrecta = **true**;  **this**.incorrecta++;  }  **this**.intentos();  **this**.bandera = **true**;  }  //Comprobación de respuesta por cado botón que contenga la respuesta correcta, se asina puntaje caso contrario el puntaje permanece igual  comprobarNumeroSumaBotones2() {  **if** ((**this**.bnt2) == **this**.numSum) {  **this**.controlCorrecta = **true**;  **this**.score = **this**.score + **150**;  **this**.correcta = **this**.correcta + **1**;  } **else** {  **this**.controlIncorrecta = **true**;  **this**.incorrecta++;  }  **this**.intentos();  **this**.bandera = **true**;  }  //Comprobación de respuesta por cado botón que contenga la respuesta correcta, se asina puntaje caso contrario el puntaje permanece igual  comprobarNumeroSumaBotones3() {  **if** ((**this**.bnt3) == **this**.numSum) {  **this**.controlCorrecta = **true**;  **this**.score = **this**.score + **150**;  **this**.correcta = **this**.correcta + **1**;  } **else** {  **this**.controlIncorrecta = **true**;  **this**.incorrecta++;  }  **this**.intentos();  **this**.bandera = **true**;  }  //Comprobación de respuesta por cado boton que contenga la respeusta correcta, se asina puntaje caso contraio el puntaje permanece igual  comprobarNumeroSumaBotones4() {  **if** ((**this**.bnt4) == **this**.numSum) {  **this**.controlCorrecta = **true**;  **this**.score = **this**.score + **150**;  **this**.correcta = **this**.correcta + **1**;  } **else** {  **this**.controlIncorrecta = **true**;  **this**.incorrecta++;  }  **this**.intentos();  **this**.bandera = **true**;  }  //Generación de números aleatorios.  numAleatorio(a: **number**, b: **number**) {  **return** Math.round(Math.random() \* (b - a) + (a));  }  //Segun el parametro obtenido de la pagina NivelPage, se asigana el valor aleatorio apara las distintas opreaciones  //Ejemplo si eligio Fácil se obtendar un parametro con valor 1 ese parametro es comparado y se asina los valores correspondientes.  asignacionNivelnum() {  **var** result: **number**;  **if** (**this**.nivel == **1**) {  result = **this**.numAleatorio(**1**, **10**);  } **else** **if** (**this**.nivel == **2**) {  result = **this**.numAleatorio(**10**, **20**);  } **else** **if** (**this**.nivel == **3**) {  result = **this**.numAleatorio(**20**, **40**);  } **else** **if** (**this**.nivel == **4**) {  result = **this**.numAleatorio(**50**, **99**);  }  **return** result;  }  //Método para generar el opredor lógico de manera aleatoria.  letrasSleatorias() {  **var** aLetras = **new** Array('+', '-', 'x');  **var** cLetra = aLetras[Math.floor(Math.random() \* aLetras.length)];  **return** cLetra;  }  // Si en dependencia de la categoría que el jugador seleccione se asigna las operaciones  OperacionLogica(a, b) {  **var** result: **number**;  **if** (**this**.signo == "+") {  result = (parseInt(a) + parseInt(b));  } **else** **if** (**this**.signo == "-") {  result = (parseInt(a) - parseInt(b));  } **else** {  result = (parseInt(a) \* parseInt(b));  }  **return** result;  }  // Se compara la categoría que selecciono el jugador para asignarle el operador lógico  operMath(a, b) {  **var** result: **number**;  **if** (**this**.operacion == "suma") {  **this**.signo = "+";  result = (parseInt(a) + parseInt(b));  }  **else** **if** (**this**.operacion == "resta") {  **this**.signo = "-";  result = (parseInt(a) - parseInt(b));  }  **else** **if** (**this**.operacion == "multiplicacion") {  **this**.signo = "x";  result = (parseInt(a) \* parseInt(b));  }  // Este caso es especial, ya que se deben generar el operador lógico de manera aleatoria  **else** **if** (**this**.operacion == "mixta") {  **this**.signo = **this**.letrasSleatorias();  result = **this**.OperacionLogica(**this**.num1, **this**.num2);  }  **return** result;  }  //Method para asiganar el nivel de juego segun el parametro obtenido.  asignacionLevelBnt() {  **if** (**this**.nivel == **1**) {  **this**.asignacionN1();  } **else** **if** (**this**.nivel == **2**) {  **this**.asignacionN1();  } **else** **if** (**this**.nivel == **3**) {  **this**.asignacionN1();  } **else** **if** (**this**.nivel == **4**) {  **this**.asignacionN1();  }  }  //Method para asiganar la respuesta correcta en cada uno de los botones empieza por el bnt1 y termina en bnt4 (segun el nivel seleccionado)  asignacionN1() {  **this**.numSum = **this**.operMath(**this**.num1, **this**.num2);  **if** (**this**.controlAsignación == **1**) {  **this**.bnt1 = **this**.numSum;  **this**.bnt2 = **this**.numSum + **1**;  **this**.bnt3 = **this**.numSum + **2**;  **this**.bnt4 = **this**.numSum - **1**;  **this**.controlAsignación++  } **else** **if** (**this**.controlAsignación == **2**) {  **this**.bnt1 = **this**.numSum + **1**;  **this**.bnt2 = **this**.numSum;  **this**.bnt3 = **this**.numSum + **2**;  **this**.bnt4 = **this**.numSum - **1**;  **this**.controlAsignación++  } **else** **if** (**this**.controlAsignación == **3**) {  **this**.bnt1 = **this**.numSum + **1**;  **this**.bnt2 = **this**.numSum + **2**;  **this**.bnt3 = **this**.numSum;  **this**.bnt4 = **this**.numSum - **1**;  **this**.controlAsignación++  } **else** **if** (**this**.controlAsignación == **4**) {  **this**.bnt1 = **this**.numSum + **1**;  **this**.bnt2 = **this**.numSum + **2**;  **this**.bnt3 = **this**.numSum - **1**;  **this**.bnt4 = **this**.numSum;  **this**.controlAsignación = **1**;  }  }    //Method para controlar los intentos del juego  intentos() {  **if** (**this**.intento == **0**) {  console.log("fin del juego");  } **else** {  **this**.intento--;  }  }  //Reinicio de variables a su estado original, cada vez que presionamos el btoón siguiente  reinicia1() {  **this**.num1 = **this**.asignacionNivelnum();  **this**.num2 = **this**.asignacionNivelnum();  **this**.bandera = **false**;  **this**.controlCorrecta = **false**;  **this**.controlIncorrecta = **false**;  **this**.asignacionLevelBnt();  }  //Método para enviar los parametros obtenidos en SegundaPage hacia FinalPage  finalizar() {  **this**.navCtrl.setRoot("FinalPage", { nombre: **this.nombre**, score: **this.score**, segundos: **this.segundos**, minutos: **this.minutos**, correcta: **this.correcta**, incorrecta: **this.incorrecta** });  }  goHomeS() {  **this**.navCtrl.setRoot(HomePage);  }  } | |

[segunda.scss.-](http://home.scss.-) en este archivo se encuentran los estilos y formatos que presenta la página como la imagen de fondo, el color de los contenedores, los estilos de botones entre otras cosas utilizadas para presentar una visualización agradable al usuario.

Con la fusión de estos tres archivos se ha logrado realizar la primera página del juego como se observa en la figura 10.

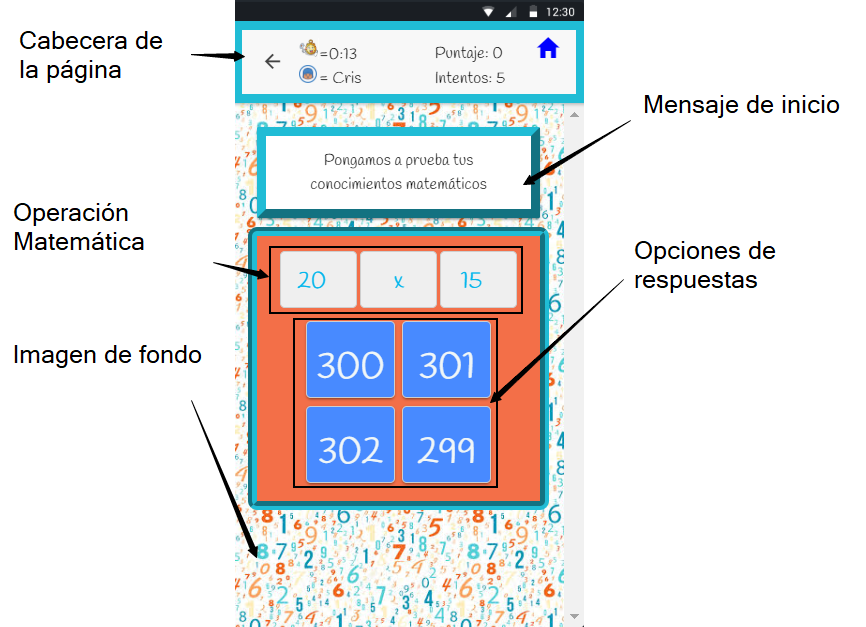


Figura 10. Página de operaciones matemáticas

**PÁGINA FINAL**

[final.html.-](http://home.scss.-) En este archivo se realiza la maquetación de la página insertando los elementos respectivos, como se observa en las líneas 7-13 tenemos toda la cabecera de la página la misma que contiene una etiqueta link para consumir un tipo de fuente mientras el dispositivo esté conectado a internet caso contrario se visualizara con el tipo de fuente de letra por defecto. En las líneas 14-58 se encuentra el cuerpo de la página la misma que consta, en las líneas 16-20 se presenta una etiqueta ion-card que aparecerá siempre que el número de respuestas sea = 5 caso contrario se presentará la etiqueta ion-card que se encuentra en las líneas 26-30 cuando las respuestas correctas está en un rango de 1 a 4 preguntas correctas, en el caso que el usuario no acertará la respuesta de ninguna operación entonces se presentará la etiqueta io-card que se encuentran en las líneas 38-44 En las líneas 46-52 se presenta una etiqueta article que abarca el puntaje obtenido, el tiempo de resolución de operaciones, el nombre del jugador y el número de respuestas correctas e incorrectas, finalmente en las líneas 54-57 se presenta dos botones que permiten salir de página actual a la página principal y el otro botón permite reiniciar el juego desde la página de selección de categorías.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | <!--  Página que muestra los resultados finales obtenidos por el jugador  See http://ionicframework.com/docs/components/#navigation for more info on  Ionic pages and navigation.  -->  <ion-header>  <link href="https://fonts.googleapis.com/css?family=Handlee" rel="stylesheet">  <ion-navbar>  <ion-title>Final</ion-title>  </ion-navbar>  </ion-header>  <ion-content padding class="bodyFinal">  <!--Imagen, se muestra cuando todas las operaciones básicas son correctas-->  <ion-card \*ngIf="correcta==5">  <ion-card-header>  ¡¡¡Felicitaciones!!!  </ion-card-header>  <ion-card-content>  <img class="imagenAcertada" src="../assets/imgs/acertadopng.png" />  Has alcanzado la máxima puntuación  </ion-card-content>  </ion-card>  <!--Imagen, se muestra cuando tiene de 2 a 4 aciertos-->  <ion-card \*ngIf="correcta >1 && correcta < 5">  <ion-card-header>  ¡¡¡Lo has hecho bien!!!  </ion-card-header>  <ion-card-content>  <img class="imagenAcertada" src="../assets/imgs/triste.png" />  Has alcanzado una puntuación regular  </ion-card-content>  </ion-card>  <!--Imagen, se muestra cuando todas las operaciones básicas son incorrectas-->  <ion-card \*ngIf="incorrecta>=4">  <ion-card-header>  ¡¡¡Deberás practicar más!!!  </ion-card-header>  <ion-card-content>  <img class="imagenAcertada" src="../assets/imgs/pesimopng.png" />  Seguramente en el próximo intento lo harás mejor  </ion-card-content>  </ion-card>  <!--Imagen, spresenta todo los datos del juego-->  <article class="section2">  <h3>Puntaje: {{score}}</h3>  <h3>Tiempo: {{minutos}}:{{segundos}}</h3>  <h3>Jugador: {{nombre}}</h3>  <h3>Respuestas Correctas: {{correcta}}</h3>  <h3>Respuestas Incorrectas: {{incorrecta}}</h3>  </article>  <!--Botones para manipular la salida y reinicio del juego-->  <div class="section4">  <button ion-button block (click)="jugarOtra()">Reiniciar</button>  <button ion-button block (click)="goHomeF()">Salir</button>  </div>  </ion-content> | |

[final.ts.-](http://home.scss.-) Este archivo contiene la parte de codificación del juego, en donde en las líneas 10-16 se realiza la declaración de variables para su respectivo almacenamiento de valores, en las líneas 21-26 se almacena las variables obtenidas de la página anterior (SegundaPage), en la línea 33-35 se crea un método para volver a reiniciar el juego que los regresa a la CategoriaPage, en el mismo que se envía como parámetro el nombre del jugador. Finalmente en las líneas 36-38 se crea un método para volver a la página principal del juego.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | **import** { Component } from '@angular/core';  **import** { IonicPage, NavController, NavParams } from 'ionic-angular';  **import** { HomePage } from '../home/home';  @IonicPage()  @Component({  selector: 'page-final',  templateUrl: 'final.html',  })  **export** **class** FinalPage {  //Declaración de variables  nombre: **string**;  score: **number**;  minutos: **number**;  segundos: **number**;  correcta: **number**;  incorrecta: **number**;  **constructor**(**public** navCtrl: **NavController**, **public** navParams: **NavParams**) {  // Aqui se extrae uno a uno los parametros obtenidos de SegundaPage  **this**.nombre = navParams.get('nombre'); //Almacena el nombre del jugador  **this**.score = navParams.get('score'); // Alamacena el puntaje del juego  **this**.segundos = navParams.get('segundos'); // Almacena los segundos del juego  **this**.correcta = navParams.get('correcta'); //Almacena el número de respuestas correctas  **this**.incorrecta = navParams.get('incorrecta'); //Almacena el valor de respuestas incorrectas  **this**.minutos = navParams.get("minutos"); //Almacena los minutos del juego  }  ionViewDidLoad() {  console.log('ionViewDidLoad FinalPage');  }  //Method para volver a la CategoriaPage, enviando como parametro el nombre del jugador  jugarOtra() {  **this**.navCtrl.setRoot('CategoriaPage', { nombre: **this.nombre** });  }  goHomeF() {  **this**.navCtrl.setRoot(HomePage);  }  } | |

[final.scss.-](http://home.scss.-) en este archivo se encuentran los estilos y formatos que presenta la página como la imagen de fondo, el color de los contenedores, los estilos de botones entre otras cosas utilizadas para presentar una visualización agradable al usuario.

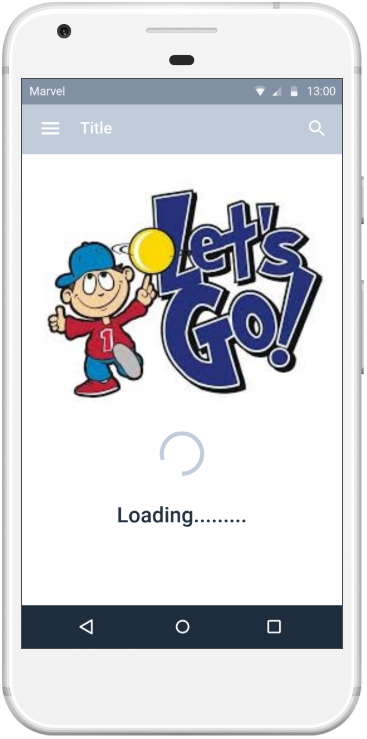
Con la fusión de estos tres archivos se ha logrado realizar la primera página del juego como se observa en la figura 11.



Figura 11. Página final del juego

**Anexos:**

**Mackup inicial del juego**



# Agregue el vínculo al proyecto aquí:

Vínculo para acceder al proyecto