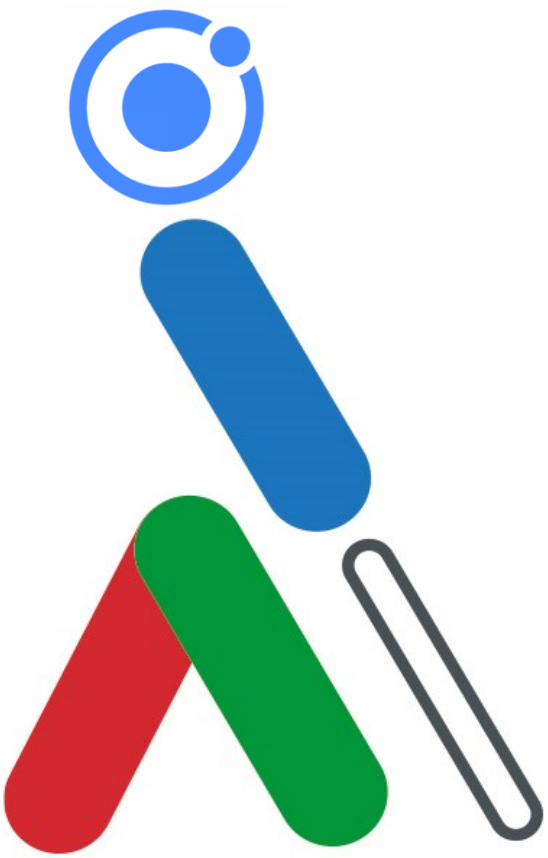

# IONIC

Exercicis d'Ionic

DESCRIPCIÓ BREU

Exercicis pas a pas per a la creació d'una aplicació amb el Framework Ionic.

Gacs

Ionic pas a pas

# Contenido

# Exercici 1

## 1. Introducció

El primer exercici serà la instal·lació del nostre entorn de treball per a poder crear aplicacions amb el framework Ionic. Aquest framework permet la creació d'aplicacions per a dispositius Android, iOS, Windows i navegadors al tractar-se d'uns framework multiplataforma.

Tot i que es tracta d'un framework multiplataforma pot incorporar codi nadiu dels tres sistemes operatius amb relativa facilitat per a garantir que l'aplicació es podrà executar a cada tipus de terminal.

## 2. NodeJS i NPM

El primer que haurem de fer serà instal·lar les aplicacions NodeJS i NPM. Per fer-ho anirem a la pàgina web de NodeJS i descarreguem la opció recomanada. Un cop descarregada l'aplicació executarem l'instal·lador.

Quan tinguem l'aplicació instal·lada obrirem un terminal i escriurem el següent:

```
$node-v
```

Al fer-ho es retornarà la versió de NodeJS i ens assegurarem de que la instal·lació s'ha realitzat de forma correcta i sense cap tipus de problema.

## 3. Ionic

Per a instal·lar Ionic obrirem el terminal i introduirem la següent línia de comandament:

```
$npm install -g @ionic/cli
```

La instal·lació d'Ionic durà uns quants minuts Durant els quals se'ns anirà mostrant tots els paquets que s'estan instal·lant al nostre equip. Quan la instal·lació finalitzi escriurem el següent comandament per a veure si tot s'ha instal·lat de forma correcta al nostre equip.

```
$ionic -v
```

Se'ns retornarà la versió d'Ionic per línia de comandament. En cas contrari voldrà dir que la instal·lació no s'ha realitat de forma correcta.

## 4. Visual Code

Per a facilitar la creació del codi és recomanable instal·lar un editor de text. EN aquest cas s'ha triat el IDEA Visual Code al tenir inserit un terminal que permet executar ordres de comandament per a anar escalant i desenvolupant les aplicacions.

Aquest IDE es gratuït i es pot descarregar des de la seva pàgina web. El IDE és gratuït i no cal que ens registrem per a poder utilitzar-ho. En cas de que es vulgui, podem instal·lar qualsevol altre IDE com a podria ser Atom o un editor de text. En aquest últim cas es recomana l'editor Notepad++ que també és gratuït.

## 5. Creant una carpeta

Un cop instal·lat tot l'entorn crearem una carpeta a la nostre unitat que contendrà totes les aplicacions que anem desenvolupant. Es recomana la creació d'aquesta carpeta per a tenir tots els nostres projectes localitzats. La creació de la carpeta la podem fer des de l'explorador de Windows o des de un terminal amb el comandament

```
$mkdir nom-carpeta
```

Veurem con ens apareix una nova carpeta.

## 6. Creació d'una aplicació

Per a crear la nostre primera aplicació obrirem un terminal i escriurem:

```
$ionic start firstApp blank --type=angular
```

Aquest comandament ens crearà un esquelet buit d'una aplicació que hem anomenat firstApp. Un cop creada l'estructura de l'aplicació escriure,

```
$cd firstApp
```

Per accedir a la carpeta de l'aplicació i un cop a dins escriurem

```
$code .
```

Aquest últim comandament ens obrirà el nostre projecta a Visual Codi. Aquesta primera aplicació es pot descarregar de forma gratuïta des del repositori de github de bcnitb.

# Exercici 2

## 1. Introducció

En aquest segon exercici farem una petita aplicació on el jugador haurà de calcular el resultat d'una operació matemàtica amb dos nombres que es generaran de forma aleatòria per la aplicació.

Així doncs crearem una aplicació anomenada **guessOperation** per a començar a fer l'aplicació. L'esquelet de l'aplicació es pot descarregar des del repositori de bcnitb.

## 2. Descripció de l'aplicació

L'aplicació que acabem de descarregar és l'esquelet de la futura aplicació. Per a veure l'aspecte actual de l'aplicació obrirem un terminal i escriurem el següent comandament:

```
$ionic serve
```

Al cap d'uns minuts se'ns obrirà el navegador d'internet que tenim per defecte i ens mostrarà l'aplicació com si fos una aplicació web.


*Figura 1. Aplicación web.*

En cas de que vulguem veure la seva experiència a un dispositiu mòbil pitjarem les tecles **Control + C** i escriurem el següent comandament escriurem:

```
$ionic serve -l
```

Es tornarà a obrir un navegador d'internet però veurem l'aplicació dins d'un terminal iOS i un Android.

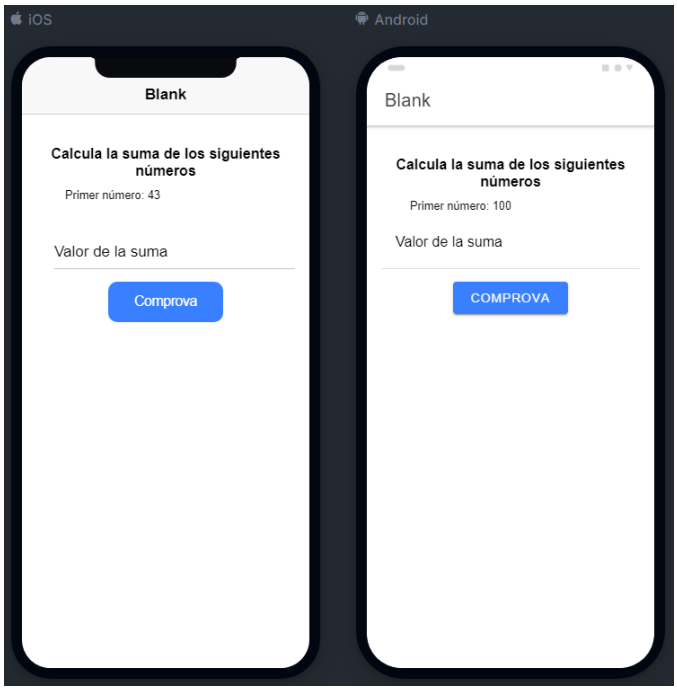


*Figura 2. Aplicació en dispositius mòbils.*

## 3. Conceptes

Si s'obre l'arxiu **home.page.html**, dins la carpeta **src/app/home** del projecte, es pot observar el codi en **HTML** de la vista. En aquest codi es pot veure que hi ha una etiqueta d'encapçalament **<h2>**, una de paràgraf **<p>** i un altre de formulari **<form>**, bàsicament. I dins la etiqueta de formulari hi ha una **<label>** i una d'un camp d'edició **<ion-input>**.

## 4. Exercici

Sota de la etiqueta de paràgraf hi ha un comentari amb el text TODO. El que es demana és que es substitueixi aquesta etiqueta per a que el mostri el següent text:

Segon número:

I un número aleatori que ha de ser diferent del número anterior. El resultat es mostra a la Figura 3.

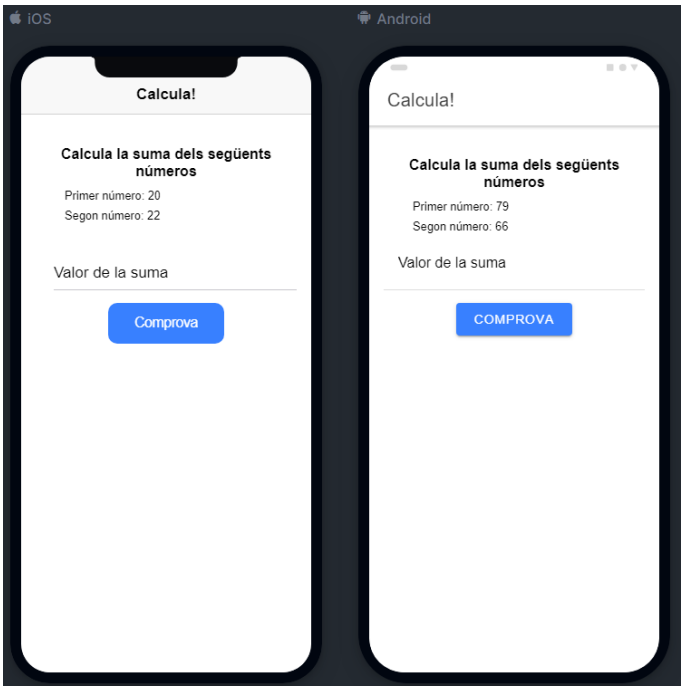


*Figura 3. Solució de l'exercici.*

# Exercici 3

## 1. Introducció

En aquest tercer exercici treballarem amb codi de TypeScript per a retocar la lògica de l'aplicació per a obtenir una resposta o una altre depenent de la resposta de l'usuari. La lògica de l'aplicació està al fitxer **home.page.ts** que es pot trobar a la mateixa direcció que el fitxer que hem vist a l'exercici 2, **src/app/home**.

Per a poder fer l'exercici haurem de descarregar la carpeta exercici 3 que trobarem al repositori de bcnitb.

## 2. Descripció de l'aplicació

Si executem l'aplicació veurem que inicialment no hi ha cap modificació, però si introduïm un valor i premem el botó **Comprova** obtenim una o una altre resposta depenent de si el nostre resultat es correcte o no. Si introduïm el resultat correcte obtenim una confirmació a la pantalla tal i com es mostra a la Figura 4. Però si el resultat que introduïm es inferior al correcte obtindrem el missatge de la figura Figura 5.

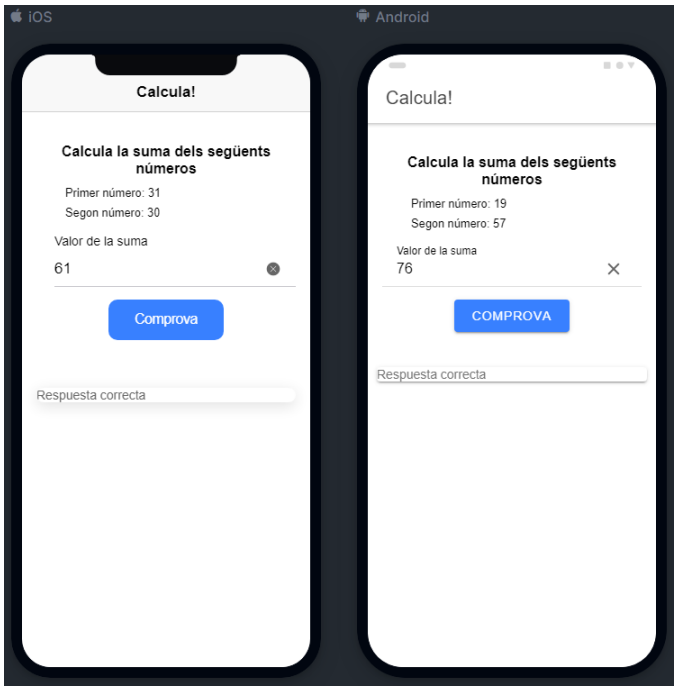


*Figura 4. Resultat correcte.*

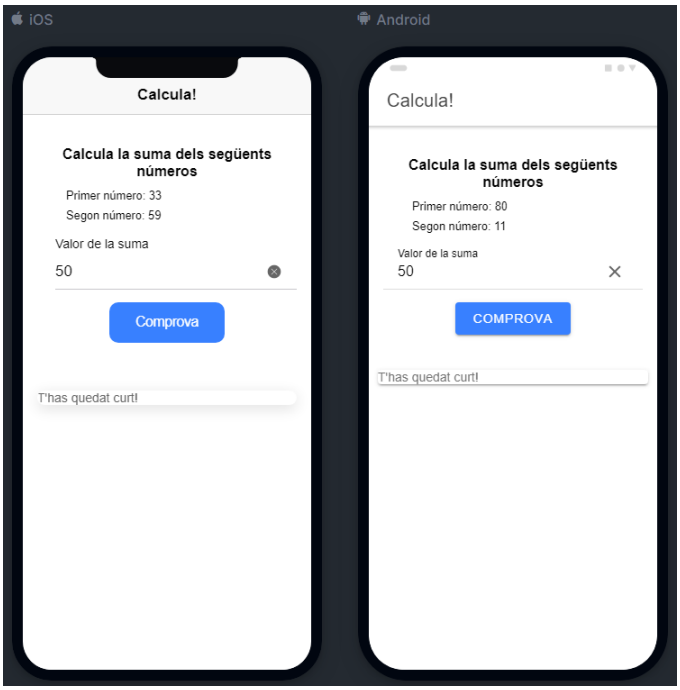


*Figura 5. Missatge si la suma és superior al valor introduït.*

## 3. Conceptes

Ara mateix, com ja hem vist, si introduïm un valor al prémer el botó de comprovació se'ns mostra un missatge sobre si el valor introduït es correcte o no.

La lògica d'aquest procediment es pot observar al fitxer **home.page.ts** que es troba a la carpeta **src/app/home**. Si obrim aquest fitxer i mirem la funció **checkAnswer()** veiem que aquesta està conformada per un bucle **if - else if**.

A dins d'un bucle **if** (si) s'entra quan es compleix la condició que hi ha dins dels parèntesis. Per exemple, el primer bucle if diu:

```
if(this.sum == this.num){
      this.result=this.resultArray[0];
}
```

A les línies superiors s'està dient: **fi** (si) el valor de la variable **sum** (suma dels nombre= és igual **(==**) al valor de la variable **num** (valor introduït) llavors la variable **result** agafa el primer valor de l'array **resultArray**.

La següent condició que es mostra **else if** indica que en cas de que la primera condició no es compleixi es revisi la segona; però que si la primera es compleix no es revisi la segona.

Sobre els array comentar que el primer element ocupa la posició 0 de l'array, ja que en programació es té en compte el nombre 0. La definició de l'array es pot observar al lloc on estan declarades totes les variables.

```
resultArray:string[]=["Respuesta correcta", "T'has quedat curt!"];
```

## 4. Exercici

Si executem l'aplicació i introduïm un valor superior al valor de la suma no se'ns retorna cap missatge. L'exercici consisteix en retoca la variable **resultArray** i la funció **checkAnswer( )** per

que es retorni el missatge: T'has passat de llarg!. Per això caldrà que es retoqui el codi on hi ha el comentari TODO.

Un cop finalitzat i executant l'aplicació la resposta en pantalla hauria de ser la següent.

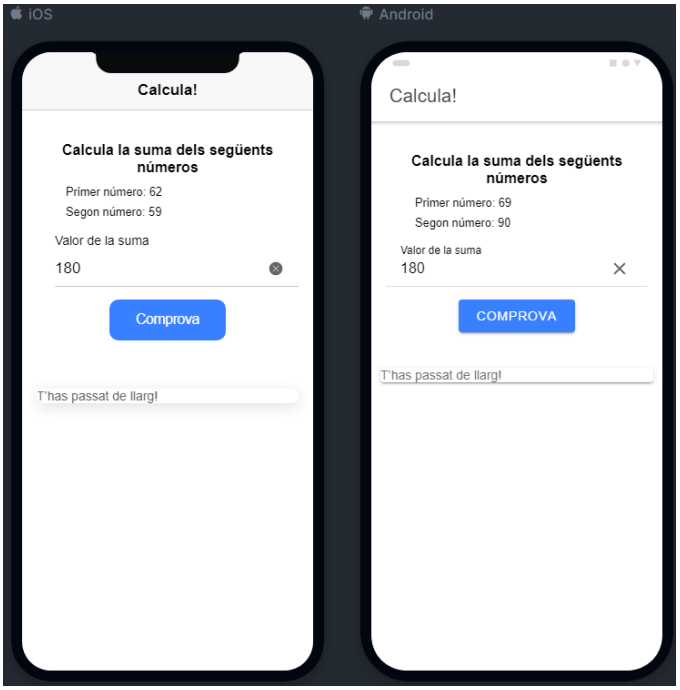


*Figura 6. Missatge per un valor més gran.*

# Exercici 4

## 1. Introducció

En aquest quart exercici farem que aparegui un dibuix amb la frase quan la resposta és incorrecte, per a donar una sortida gràfica al usuari. Com sempre, farem que la imatge sigui accessible, encara que en aquest cas la imatge serà decorativa i podríem evitar que el lector de pantalla la tingués en conte.

## 2. Descipció de l'aplicació

Per a fer l'exercici descarregarem el codi inicial del repositori de github de bcnitb i executarem l'aplicació per a veure el seu aspecte.


*Figura 7. Animació d'encert.*

## 3. Conceptes

Per fer que el formulari on introduir el valor de la suma desaparegui al introduir el valor correcte, s'han realitzat modificacions als arxius **home.page.html** i **home.page.ts**.

A l'arxiu **home.page.html** s'ha inclòs dins la etiqueta **<form>** la propietat ***ngIf**. Aquesta propietat realitza la mateixa funció que el bucle **if** que hem vist a l'exercici anterior. En aquest cas li hem dit a la propietat que el formulari ha de ser visible quan la variable **check** sigui **false**.

```
<form *ngIf="!check">
```

Per mostrar la imatge s'ha inclòs una nova línia a sota del missatge de text amb una propietat ***ngIf**. Aquesta línia és la que mostra l'animació quan s'encerta el resultat. L'animació que s'ha utilitzat és una animació lliure de la web giphy.

```
<div *ngIf="check" style="width:100%;height:0;padding-
bottom:45%;position:relative;">
        <span class="hidden">Animació de victòria</span>
```

```
        <iframe src="https://giphy.com/embed/MOWPkhRAUbR7i" width="100%"
height="100%" style="position:absolute" frameBorder="0" class="giphy-
embed" allowFullScreen></iframe>
      </div>
```

És al fitxer **home.page.ts** on es controla el valor de la variable **check**, aquesta variable només pot ser **true** o **false**. Com es pot veure a la funció **checkAnswer( )** la variable **check** modifica el seu valor quan s'introdueix el valor correcte.

```
if(this.sum == this.num){
      this.result=this.resultArray[0];
      this.check=true;
}
```

## 4. Exercici

En cas de que no s'introdueixi el valor correcte no hi ha cap animació i només es mostra el missatge, cosa que hem aconseguit a l'exercici anterior.

Per realitzar aquest exercici s'ha de seguir els següents passos:

1. Crear una nova variable **fail** de tipus boolean, com la variable **check**.
2. Modificar el valor de la variable **fail** a la funció **checkAnswer( )**.
3. Utilitzar la propietat ***ngIf** per mostrar/amagar l'animació.

Es recomana copiar el codi on es mostra l'animació de victòria i modificar l'enllaç que es mostra dins de src (`src="https://giphy.com/embed/MOWPkhRAUbR7i"`), on s'hauria d'introduir una nova adreça web. Es proposa l'adreça https://giphy.com/embed/Ly6FB6xRSJlW8.

Si hem realitzat l'excreció de forma correcta haurem d'obtenir el resultat de la Figura 8 si introduïm un valor erroni.

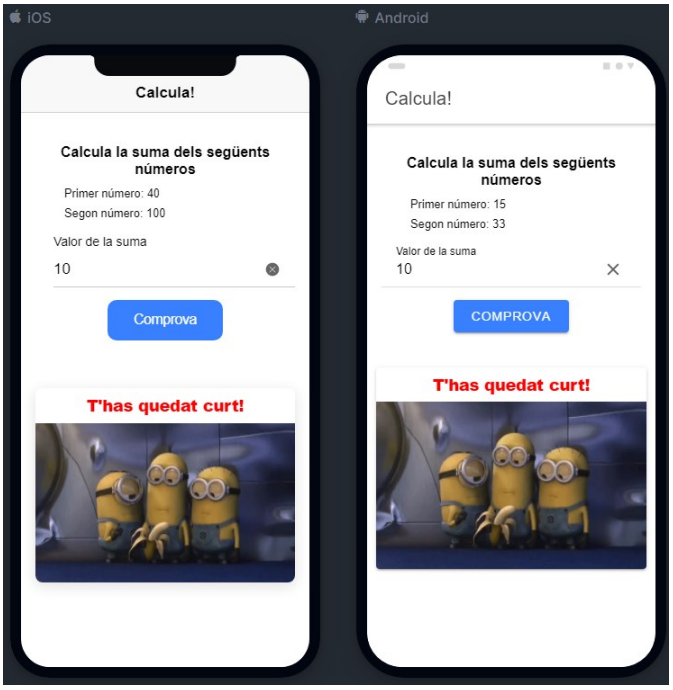


*Figura 8. Animació de resultat erroni.*

# Exercici 5

## 1. Introducció

L'aplicació comença a tenir bona cara i funciona, però tot i que la imatge és accessible als lectors de pantalla, un usuari del lector de pantalla haurà de realitzar un flick per a saber si ha calculat bé o malament el valor, amb la qual cosa estaria bé llançar un so per a donar una resposta ràpida a aquests usuaris.

## 2. Descripció de l'aplicació

Si descarregueu l'aplicació del repositori de bcnitb i l'executem, veurem que quan encertem la suma dels dos nombres es llança el so de gent aplaudint. D'aquesta forma donem una informació sonora als usuaris de lector de pantalla i puguin saber si han calculat bé l'operació o no.

## 3. Conceptes

Per que es reprodueixi un so a la nostre aplicació hem d'utilitzar una funció de TypeScript anomenada **Audio( )**. Per tant, si obrim el fitxer **home.page.ts** veurem que tenim una sèrie de variables noves que no estaven. Aquestes variables son:

```
sound:any;
audio=new Audio();
audioTime:any;
```

La primera variable, **sound**, és on es guardarà el so que llançarem al donar al botó per enviar la nostre resposta. La segona variable, **audio**, emmagatzema la funció **Audio( )**, que serà l'encarregada de reproduir el so. Per últim, la variable **audioTime** s'encarrega de posar el temps de duració del so a zero per pausar-ho.

Si baixem al final del codi observarem la nova funció **play( )** que és encarregada de carregar el so a la funció **Audio( )** i fer que es reprodueixi. Però abans de tot el que fa la funció és cridar a la funció **pauseAudio( )** per aturar qualsevol audio que s'estigui reproduint. Això es fa per evitar un solape dels sons.

La funció **play( )** té la següent forma:

```
play(sound:any) {
    this.pauseAudio(sound);

    this.audio.src=sound;
    this.audio.load();
    this.audio.play();

    this.audioTime=setTimeout(() =>{
      sound.playing=false;
    }, sound.duration*1000);
  }
```

Pel que fa a la funció **pauseAudio( )**, es de la següent forma:

```
private pauseAudio(selectedSound: any){
    clearTimeout(this.audioTime);
```

```
    this.audio.pause();
  }
```

Com podem observar, les funcions son força senzilles.

## 4. Exercici

En aquest exercici es demana que quan el valor introduït sigui erroni es llanci l'efecte sonor **fail.mp3** per dona una sortida sonora als usuaris de lectors de pantalla.

Els efectes sonors s'han descarregat de la web: http://www.sonidosmp3gratis.com/efectos, si es volen utilitzar uns altres efectes sonors ho podeu fer.

# Exercici 6

## 1. Introducció

L'aplicació ja està gairebé finalitzada. Podem fer-la una mica més atractiva si incluïm un marcador on es vagin sumant el nombre d'intents per a fer que l'usuari intenti millorar la seva marca.

## 2. Descripció de l'aplicació

L'aplicació per a realitzar l'exercici es pot descarregar, com sempre, al repositori de github de bcnitb i si l'executem veurem que ens mostra un marcador quan introduïm el resultat correcte.


*Figura 9. Puntuació obtinguda.*

## 3. Conceptes

En aquest cas s'ha afegit una nova variable anomenada **count** que ens servirà per a emmagatzemar el nombre d'intents que ha realitzat l'usuari fins aconseguir realitzar l'operació de forma correcte. Aquesta variable es pot trobar a sota de les variables numèriques de l'aplicació.

```
count:number=0;
```

A l'aplicació ara mateix només es modifica la variable si el valor introduït és el correcte i es fa de la següent manera:

```
++this.count;
```

A la línia superior estem indicant que a la variable **count** se li sumi un 1. Aquest codi també es podria escriure com:

```
this.count=this.count+1;
```

Que poder és una forma més clara d'escriure'l; encara que a la programació es tendeix a compactar el codi sempre que es pot.

Ens em de fixar que la modificació de la variable es realitza dins del bucle **if** que compleix la condició **this.sum==this.num**.

## 4. Exercici

L'exercici que es proposa és que el comptador també tingui en consideració quan s'ha introduït un valor erroni per la suma; ja que ara mateix només té en consideració quan s'introdueix el valor correcte i estem falsejant el resultat real.

# Exercici 7

## 1. Introducció

Gairebé ja tenim l'aplicació llesta; només ens caldria afegir dues coses més per a fer-la més atractiva motivant a l'usuari.

## 2. Descripció de l'aplicació

Si descarreguem l'aplicació del repositori de bcnitb i l'executem veurem que ens mostra un missatge motivador quan introduïm el resultat correcte. En cas de que encertem a la primera ens sortirà un missatge, i si necessitem dos intents el missatge serà un altre diferent.

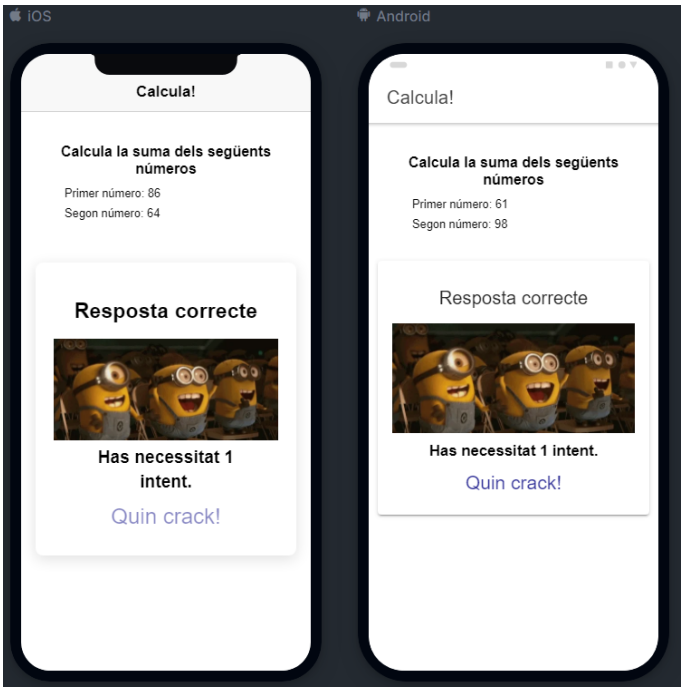


*Figura 10. Missatge de victòria.*

## 3. Descripció de l'aplicació

Si obrim el fitxer amb el codi de TypeScript (home.page.ts) veurem que s'han creat dues variables nomes de tipus **string**. Aquesta variables son **msg** i **msgArray**.

```
msg:string;
msgArray:string[]=["Quin crack!", "Gairebé perfecte!"];
```

La primera variable serà la que contingui el text a mostrar per pantalla depenent de la puntuació obtinguda o la segona és un array on s'emmagatzemen tots els valors possibles de la variable **msg**.

Per consignar els diferents valors de la variable **msg** s'ha creat un bucle de tipus **switch**. Aquest bucle és molt similar al bucle **if** que es va veure a l'Exercici 3. La diferencia es que en un bucle **switch** la comparació es realitza dins el mateix bucle.

```
switch(this.count){
      case 1:
        this.msg=this.msgArray[0];
        break;
```

```
      case 2:
        this.msg=this.msgArray[1];
        break;
      default:
        this.msg="oooops!";
        break;
    }
```

En el bucle que s'acaba de mostrar es dius que la condició és el valor de la variable **count**, i que depenent d'aquest valor s'executi un cas (**case**) o un altre, i si no es compleix cap cas llavors s'executi el cas per defecte (**default**).

Com es pot veure, en aquest tipus de bucle no hi ha cap signe d'equivalència.

A banda d'això, també s'ha modificat l'arxiu de la part visual de l'aplicació: home.page.html, on s'han creat dues noves classes i s'ha fet que el text es mostri d'un color o un altre depenent del valor de la variable **count**. Per realitzar aquesta tasca s'ha utilitzar la propietat **[ngClass]**.

```
<p [ngClass]="{'good': count==1, 'bad': count>1}" class="message">
      {{ msg }}
</p>
```

A les línies superior es diu que en cas que la variable **count** valgui 1, s'utilitzi la classe **good** de la fulla d'estils i que si es superior a 1 s'utilitzi la classe **bad**.

## 4. Exercici

En aquest exercici es demanen dues coses, així que haurem de retocar dos arxius diferents.

### 4.1. Exercici a

En aquest primer apartat crearem fins a 3 missatges diferents per obtenir un total de 5 respostes depenen de la puntuació de l'usuari.

### 4.2. Exercici b

En cas de que hagin calgut més de 3 intents per a introduir el valor correcte es mostrarà la classe **bad**.

# Exercici 8

## 1. Introducció

Si descarreguem l'aplicació de l'exercici 8 del repositori de bcnitb veurem que ja tenim gairebé enllistada l'aplicació. Encara però tenim un petit inconvenient. Hem posat 5 missatges diferents depenent dels intents necessaris fins introduir el valor correcte de la suma, però... Què passa si han calgut 6 o més intents?

La respostes és que es llançaria el valor per defecte (**default**), però no estaria millor limitar el nombre d'intent? Això és el que farem en aquest exercici. Una altre cosa que farem és que l'aplicació ens vagi informant dels intents.

## 2. Descripció de l'aplicació

Si llancem l'aplicació veurem que a sota del segon número aleatori es mostra el text **N Intents**. Si no em realitzat cap intent ens indica que el nombre d'intents és zero. També ens mostra una animació de joc perdut.

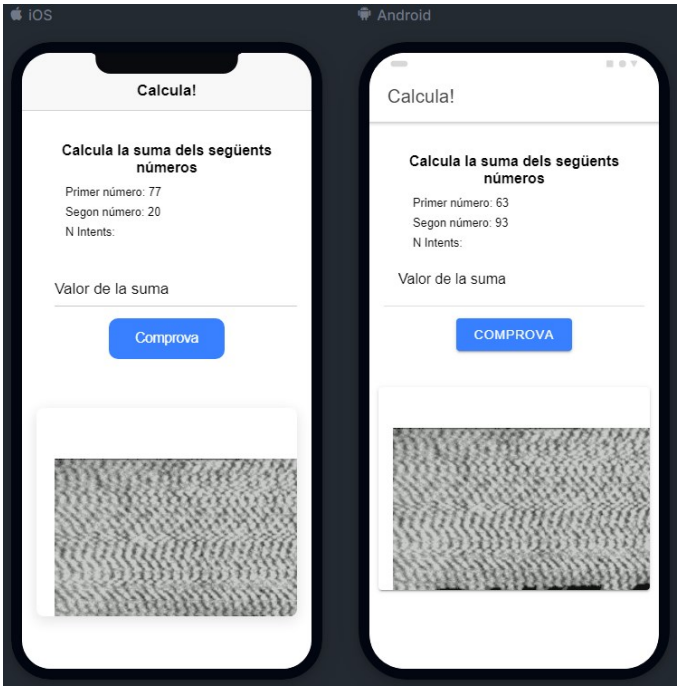


*Figura 11. Comptador de l'aplicació.*

## 3. Concepte

L'animació de **GameOver** es pot localitzar a sota de l'animació d'error. El codi és molt semblant al de les dues altres animacions. Per que es mostri aquesta animació només en cas de pèrdua del joc, haurem d'utilitzar una condició complexa del tipus A o B, que es representa amb **||**. Així que caldrà pensar una mica com la construïm.

```
<div *ngIf="A || B">
```

Pel que fa al nombre d'intents, aquest es recollen a la variable **count**, la qual és visible sempre i de la qual ja hem parlat en exercicis anteriors.

## 4. Exercici

Com hem dit anteriorment, hem de fer dos petits exercicis.

### 4.1. Exercici a

El primer exercici serà fer que el text d'intents només surti quan hàgim introduït un valor i no abans, és a dir, que només es mostri quan el nombre d'intents sigui 1 o superior.

### 4.2. Exercici b

El segon exercici es fer que el nombre d'intents sigui tingui un nombre màxim de 5, i que després del cinquè intents no ens deixi introduir cap altre valor i es mostri una pantalla de **Game Over**[1].

[1] La pantalla ja està fet i només s'ha de mostrar. Si es vol, es pot modificar aquesta pantalla.

# Exercici 9

## 1. Introducció

Ara mateix l'aplicació ja funciona, però quan finalitzem la partida, és a dir, fem més de 5 intents sense encertar o encertem el resultat, ens desapareix el botó **Comprova** i no podem fer res més. Això és un inconvenient si volem torna a intentar.

Per això, a aquest exercici crearem un nou botó que es mostrarà quan acabem el joc i d'aquesta forma poder tornar a intentar-ho.

## 2. Descripció

Si descarreguem l'aplicació del repositori de bcnitb veurem que l'aplicació és igual tal i com l'hem deixat a l'exercici anterior i que només s'ha afegit un so en cas de que hàgim perdut o guanyat la partida.

## 3. Conceptes

Per aquest exercici s'ha creat una funció anomenada **newGame( )**. Aquesta funció es pot observer al final de l'arxiu **home.page.ts** i la seva funció és reiniciar les variables de l'aplicació.

Per a reiniciar una variable el que s'ha de fer és assignar-li el valor inicial a la variable al pitjar sobre el botó que estigui enllaçat amb la dunció. Un exemple de reinici de dues variables es pot veure a les següent línies.

```
newGame(){
    this.num=null;
    this.num1=this.randomNum(0,100);
    this.num2=this.randomNum(0,100);
}
```

Com es port observar, el reinici de variables es molt semblant a la declaració d'una variable però sense marcar la seva tipología.

## 4. Exercici

En aquest cas farem 3 exercicis per a poder desenvolupar tot el que cal per a finalitzar l'aplicació.

### 4.1. Exercici a

El primer de tot serà la creació d'un botó que anomenarem **Reiniciar**[2]. Per a fer aquest nou botó haurem d'obrir l'arxiu home.page.html i mirar el codi del botó que hi ha al formulari. Aquest botó haurà de crida a una funció anomenada **newGame( )**.

### 4.2. Exercici b

Si executem l'aplicació veurem que el botó apareix sempre en pantalla; no està malament donar l'opció de reiniciar la partida quan es vulgui; però al tenir només 5 intents, seria millor que aquest botó només aparegui quan es finalitza la partida. Per tant, s'ha d'incloure un atribut ***ngIf** a l'etiqueta.

Penseu en quina ha de ser la condició o condicions per que es mostri el botó.

[2] Si es vol que el botó estigui centrat s'ha de crear una etiqueta ** ** amb la classe (**Class**) **txt-center**.

### 4.3. Exercici b

La darrera part d'aquest exercici es realitzarà a l'arxiu home.page.ts. A aquest fitxer veurem que hi ha una funció anomenada **newGame( )** al final de l'arxiu. Aquesta funció reinicia totes les variables al seu valor d'inici, però està inacabada. El que s'ha de fer és reiniciar les variables que manquen.

# Apèndix I: Afegir un nou sumant[3]

Si hem realitzar l'últim exercici de forma correcta ara l'aplicació seria plenament funcional i es podria reiniciar tantes vegades com es volgués. En aquest apèndix durem l'aplicació un pas més enllà i es demana que es modifiqui la mateixa per a que es puguem sumar tres números aleatoris enlloc de dos.

Com a guia, s'han de realitzar modificacions al fitxers **home.page.html** i **home.page.ts**. Aquestes modificacions que s'han de realitzar son:

## 1. Home.page.html.

- Afegir una nova etiqueta **<p>** on mostrar el nou número a sumar.
- Modificar el valor de la propietat **max** de l'etiqueta **<ion-input>**.

## 2. Home.page.ts.

- Crear una nova variable **num3**.
- Modificar la funció **calculateSum( )**.
- Modificar la funció **newGame( )**.

Si s'ha realitzat de forma correcta les modificacions indicades, l'aparença de l'aplicació deuria ser semblant a la Figura 12.

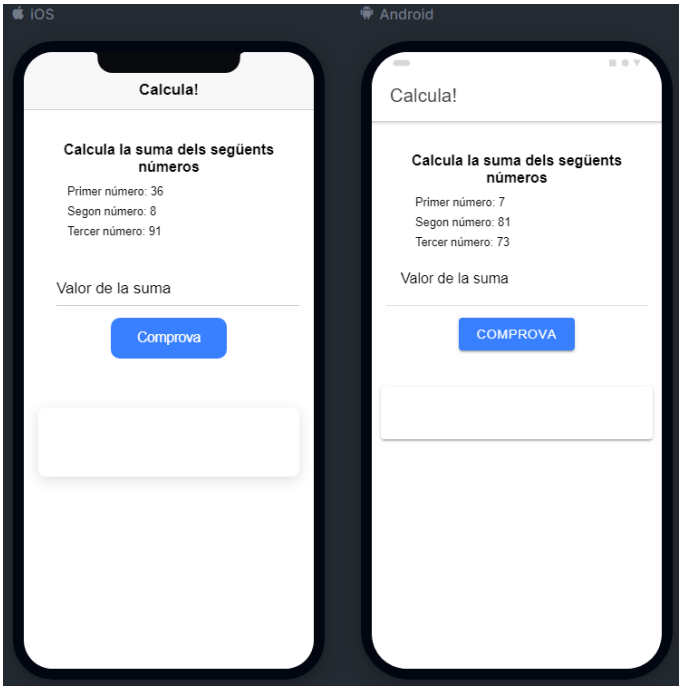


*Figura 12. Aspecte final de l'aplicació.*

Al repositori de bcnitb podeu descarregar l'aplicació finalitzada i fer sobre aquest fitxer les modificacions pertinents per a realitzar l'apèndix.

[3] Es por veure la solució al repositori bcnitb.

# Apèndix II: Multiplicant valors

En aquest apèndix es demana que s'agafi l'aplicació finalitzada, que es pot descarrega des del repositori de bcnitb i modificar la mateixa per que es demani la multiplicació del dos nombres que es mostren de manera aleatòria al iniciar-se l'aplicació.

Com a guia, s'han de realitzar modificacions al fitxers home.page.html i home.page.ts. Aquestes modificacions que s'han de realitzar son:

## 3. Home.page.html.

- Modificar el text **Calcula la suma dels...** per **Calcula el producte dels...**
- Modificar el text de la etiqueta **<ion-label>**: **Valor de la suma** per **Valor del producte**.

## 4. Home.page.ts.

- Modificar el nom de la variable **sum** per **prod**[4].
- Modificar la funció **calculateSum( )** que s'anomenarà **calculateProd( )** i fer les modificacions adients.

Nota: si es vol fer més fàcil la multiplicació, haurem de modificar el límit de la variable **num1** o **num2**. D'aquesta forma podem imposar es calculi un valor aleatori entre 0 i 100 i l'altre entre 0 i 10.

Si hem programat de forma correcte les modificacions, la nostre aplicació hauria de tenir el següent aspecte.

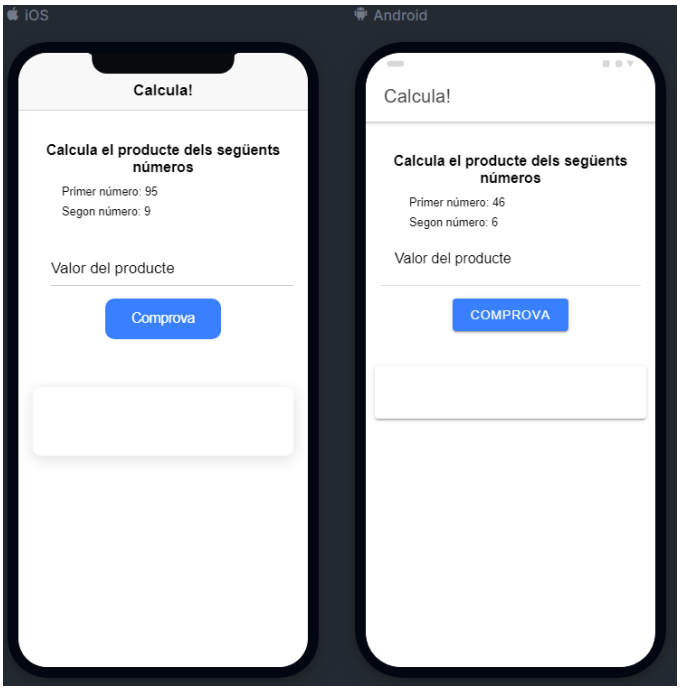


*Figura 13. Aplicació producte.*

L'arxiu amb la solució a aquest apartat es pot descarregar des de el repositori de bcnitb.

[4] S'ha de tenir en compte que la modificació del nom de la variable sum per prod s'ha de realitzar a rotes les línies del fitxer on aparegui la variable.