

Universidad Autónoma De Baja California

Facultad de ingeniería



Licenciatura en Sistemas Computacionales

Aplicaciones Web Con Base de Datos

Primer parcial: Examen práctico - Calculadora Angular

Catedrático

Rosendo Rafael Sosa Canales

Alumno

Carlos Francisco Espinoza Rivera

Mexicali, Baja California 9 de abril del 2022

Introducción

Las siguientes capturas de pantalla, representan los pasos que se han llevado a cabo para realizar la calculadora en framework Angular.

Instrucciones:

- **Paso 1:** Instalación de Angular CLI
- **Paso 2:** Inicializando nuestro Proyecto
- **Paso 3:** agregar nuestra plantilla HTML y estilos

Retos:

- **Paso 4:** Escuchar eventos de clic en los botones y obtener sus valores asociados.
- **Paso 7:** Visualización del valor de las variables en la plantilla.

Una vez que tenemos este proyecto los retos para aprobar este examen son los siguientes:

1. Investigar cómo utilizar el enlace de eventos para vincular estos métodos a la plantilla (ver referencias).
2. Explicar el funcionamiento lógico de los métodos anteriores y cómo funciona el enlace de variables así como los métodos. [Documento word o presentación].
3. Subir a GIT su versión, se toma en cuenta que se personalice.

Referencias:

<https://angular.io/guide/event-binding>
<https://angular.io/guide/property-binding>
<https://angular.io/guide/two-way-binding>

Desarrollo

1. Debemos instalar Angular en nuestro equipo.

comando: npm install -g @angular/cli

```
PS C:\USERS\CORE15\DESKTOP> NPM INSTALL -G @ANGULAR/CLI
CHANGED 193 PACKAGES, AND AUDITED 194 PACKAGES IN 215
23 PACKAGES ARE LOOKING FOR FUNDING
  RUN `NPM FUND` FOR DETAILS

FOUND 0 VULNERABILITIES
PS C:\USERS\CORE15\DESKTOP> |
```

2. Debemos instalar TypeScript con el siguiente comando: npm install -g typescript

```
PS C:\USERS\CORE15\DESKTOP> NPM INSTALL -G TYPESCRIPT
CHANGED 1 PACKAGE, AND AUDITED 2 PACKAGES IN 15

FOUND 0 VULNERABILITIES
PS C:\USERS\CORE15\DESKTOP> |
```

3. Cree mi proyecto con el siguiente comando ng new examen-calculadora-1138153.

```
PS C:\USERS\CORE15\DESKTOP> NG NEW EXAMEN-CALCULADORA-1138153
? WHICH STYLE SHEET FORMAT WOULD YOU LIKE TO USE? CSS
CREATE EXAMEN-CALCULADORA-1138153\ANGULAR.JSON (3159 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\PACKAGE.JSON (1089 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\README.MD (1078 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\TSCONFIG.JSON (863 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\EDITORCONFIG (274 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\.GITIGNORE (548 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\.BROWSERSLISTRC (600 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\KARMA.CONF.JS (1443 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\TSCONFIG.APP.JSON (287 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\TSCONFIG.SPEC.JSON (333 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\.VSCODE\EXTENSIONS.JSON (130 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\.VSCODE\LAUNCH.JSON (474 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\.VSCODE\TASKS.JSON (938 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\FAVICON.ICO (948 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\INDEX.HTML (310 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\MAIN.TS (372 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\POLYFILLS.TS (2338 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\STYLES.CSS (80 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\TEST.TS (745 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\ASSETS\.GITKEEP (0 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\ENVIRONMENTS\ENVIRONMENT.PROD.TS (51 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\ENVIRONMENTS\ENVIRONMENT.TS (658 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\APP\APP.MODULE.TS (314 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\APP\APP.COMPONENT.HTML (23332 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\APP\APP.COMPONENT.SPEC.TS (1016 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\APP\APP.COMPONENT.TS (230 BYTES)
CREATE EXAMEN-CALCULADORA-1138153\SRC\APP\APP.COMPONENT.CSS (0 BYTES)
✓ PACKAGES INSTALLED SUCCESSFULLY.
```

4. Debemos acceder al proyecto que hemos creado.

```
Windows PowerShell
PS C:\USERS\CORE15\DESKTOP> CD EXAMEN-CALCULADORA-1138153
PS C:\USERS\CORE15\DESKTOP\EXAMEN-CALCULADORA-1138153> |
```

5. Para confirmar que nuestro proyecto se ha creado con todos los paquetes y complementos, debemos levantar nuestro servidor.

```
Windows PowerShell
PS C:\USERS\CORE15\DESKTOP> CD EXAMEN-CALCULADORA-1138153
PS C:\USERS\CORE15\DESKTOP\EXAMEN-CALCULADORA-1138153> NG SERVE --OPEN
✓ BROWSER APPLICATION BUNDLE GENERATION COMPLETE.

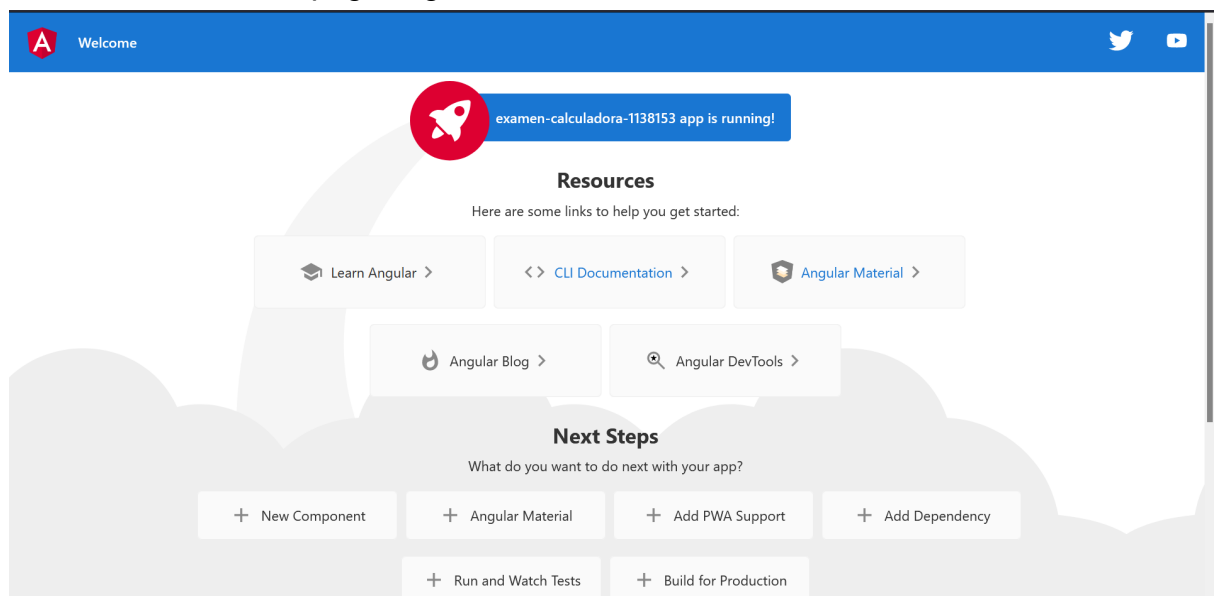
INITIAL CHUNK FILES | NAMES | RAW SIZE
VENDOR.JS | VENDOR | 1.70 MB
POLYFILLS.JS | POLYFILLS | 294.88 KB
STYLES.CSS | STYLES | 173.26 KB
MAIN.JS | MAIN | 48.03 KB
RUNTIME.JS | RUNTIME | 6.55 KB
INITIAL TOTAL | 2.21 MB

BUILD AT: 2022-04-09T08:13:08.333Z - HASH: 1F313737A4E6FD94 - TIME: 10195M5
** ANGULAR LIVE DEVELOPMENT SERVER IS LISTENING ON LOCALHOST:4200. OPEN YOUR BROWSER ON HTTP://LOCALHOST:4200/ **

/ COMPILED SUCCESSFULLY.
```

Nota: Nuestro puerto local será el 4200.

Debemos de ver una pagina igual a esta:



6. Ahora debemos agregar nuestra plantilla HTML y los estilos css.

```
Windows PowerShell
PS C:\USERS\CORE15\DESKTOP\EXAMEN-CALCULADORA-1138153> NG GENERATE COMPONENT CALCULATOR --SKIP-TESTS
SUPPORT FOR CAMEL CASE ARGUMENTS HAS BEEN DEPRECATED AND WILL BE REMOVED IN A FUTURE MAJOR VERSION.
USE '--SKIP-TESTS' INSTEAD OF '--SKIPTTESTS'.
CREATE SRC/APP/CALCULATOR/CALCULATOR.COMPONENT.HTML (25 BYTES)
CREATE SRC/APP/CALCULATOR/CALCULATOR.COMPONENT.TS (291 BYTES)
CREATE SRC/APP/CALCULATOR/CALCULATOR.COMPONENT.CSS (0 BYTES)
UPDATE SRC/APP/APP.MODULE.TS (412 BYTES)
PS C:\USERS\CORE15\DESKTOP\EXAMEN-CALCULADORA-1138153> |
```

7. En en el archivo `src/app/calculator/calculator.component.html` agregar el contenido que aparece en el siguiente link:
<https://gist.github.com/Xhendor/98fc2d9210a026e58edb23af8d0ea8cd>

8. Nuestra modificación debe verse de la siguiente manera:

```
calculator.component.html U
src > app > calculator > calculator.component.html > div.calculator
1 <div class="calculator">
2
3 <input type="text" class="calculator-screen" value="0" disabled />
4
5 <div class="calculator-keys">
6
7 <button type="button" class="operator" value="+">+</button>
8 <button type="button" class="operator" value="-">-</button>
9 <button type="button" class="operator" value="*">*</button>
10 <button type="button" class="operator" value="/">/</button>
11
12 <button type="button" value="7">7</button>
13 <button type="button" value="8">8</button>
14 <button type="button" value="9">9</button>
15
16
17 <button type="button" value="4">4</button>
18 <button type="button" value="5">5</button>
19 <button type="button" value="6">6</button>
20
21
22 <button type="button" value="1">1</button>
23 <button type="button" value="2">2</button>
24 <button type="button" value="3">3</button>
25
26
27 <button type="button" value="0">0</button>
28 <button type="button" class="decimal" value=".">.</button>
29 <button type="button" class="all-clear" value="all-clear">AC</button>
30
31 <button type="button" class="equal-sign" value="=">=</button>
32
33 </div>
34 </div>
35
```

9. En el archivo src/app/calculator/calculator.component.css agregar este contenido: <https://gist.github.com/Xhendor/69c1908dd01c4e6b676033a6bff12325>

```
calculator.component.html U # calculator.component.css U X
src > app > calculator > # calculator.component.css > calculator-keys
1 .calculator {
2   border: 1px solid #ccc;
3   border-radius: 5px;
4   position: absolute;
5   top: 50%;
6   left: 50%;
7   transform: translate(-50%, -50%);
8   width: 400px;
9 }
10
11 .calculator-screen {
12   width: 100%;
13   font-size: 5rem;
14   height: 80px;
15   border: none;
16   background-color: #252525;
17   color: #fff;
18   text-align: right;
19   padding-right: 20px;
20   padding-left: 10px;
21 }
22
23 button {
24   height: 60px;
25   background-color: #fff;
26   border-radius: 3px;
27   border: 1px solid #c4c4c4;
28   background-color: transparent;
29   font-size: 2rem;
30   color: #333;
31   background-image: linear-gradient(to bottom, transparent 50%, #000 50%, #000 50%, #000 50%);
32   box-shadow: inset 0 0 1px #000, inset 0 1px 0 #000, inset 0 1px 0 #000, inset 0 1px 0 #000;
33   text-shadow: 0 1px 0 #000;
34 }
35
36 button:hover {
37   background-color: #eaeaea;
38 }
39
40 operator {
```

10. En el archivo de estilos global src/styles.css se agrega lo siguiente:

```
html {  
    font-size: 62.5%;  
    box-sizing: border-box;  
}  
  
*, *::before, *::after {  
    margin: 0;  
    padding: 0;  
    box-sizing: inherit;  
}
```

```
src > # styles.css > *
```

```
1  html {  
2      font-size: 62.5%;  
3      box-sizing: border-box;  
4  }  
5  *, *::before, *::after {  
6      margin: 0;  
7      padding: 0;  
8      box-sizing: inherit;  
9  }  
10
```

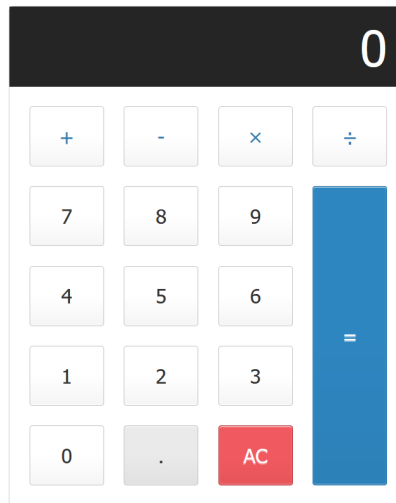
11. Al finalizar de realizar lo anterior en el archivo src/app/app.component.html borrar el contenido y agregar solo lo siguiente:

```
# styles.css M    <> app.component.html M X
```

```
src > app > <> app.component.html > app-calculator
```

```
1  <app-calculator></app-calculator>
```

12. Si recargamos nuestra página podremos ver los cambios que se han realizado.
-



13. Ahora debemos programar los eventos de escucha para los botones que tenemos en la plantilla, podemos ver que tenemos el siguiente patrón:
- dígitos (0-9),**
 - operadores (+, -, *, /, =),**
 - un punto decimal (.)**
 - y una tecla de reinicio.**
14. Abriendo el archivo que se encuentra en la siguiente ruta: `src/app/calculator/calculator.component.ts` debemos definir los siguientes atributos.

```
currentNumber = '0';  
firstOperand:any|null = null;  
operator:any|null = null;  
waitForSecondNumber = false;
```

Estos atributos se traducen de la siguiente manera:

- La variable **currentNumber** contiene la cadena que se mostrará en el elemento de entrada de resultados.
- La variable **firstOperand** contiene el valor del primer operando de la operación.
- La variable del **operator** contiene la operación.
- La variable **waitForSecondNumber** contiene un valor booleano que indica si el usuario ha terminado de escribir el primer operando y está listo para ingresar el segundo operando de la operación.

15. Los metodos para trabajar se encuentran en el siguiente link:

<https://gist.github.com/Xhendor/b733201693aaaa0158c75907d7a1f835>

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-calculator',
  templateUrl: './calculator.component.html',
  styleUrls: ['./calculator.component.css']
})
export class CalculatorComponent implements OnInit {

  constructor() { }
  currentNumber = '0';
  firstOperand : any | null = null;
  operator: any | null = null;
  waitForSecondNumber = false;

  ngOnInit(): void {
  }
  public getNumber(v: string){
    console.log(v);
    if(this.waitForSecondNumber)
    {
      this.currentNumber = v;
      this.waitForSecondNumber = false;
    }else{
      this.currentNumber === '0'? this.currentNumber = v: this.currentNumber += v;
    }
  }

  getDecimal(){
    if(!this.currentNumber.includes('.')){
      this.currentNumber += '.';
    }
  }
}
```

Métodos:

getNumber() que se utilizará para obtener el número actual.

getDecimal() que agrega el punto decimal al número actual.

doCalculation() que realiza el cálculo según el tipo de operador.

getOperation() que se usará para obtener la operación realizada.

clear() que se utilizará para borrar el área de resultados y restablecer los cálculos.

16. Los métodos que se han colocado dependen de los atributos, es necesario corregir los errores que aparecen:

```
private doCalculation(op:any,secondOp:any){
  switch (op){
    case '+':
      return this.firstOperand += secondOp;
    case '-':
      return this.firstOperand -= secondOp;
    case '*':
      return this.firstOperand *= secondOp;
    case '/':
      return this.firstOperand /= secondOp;
    case '=':
      return secondOp;
  }
}
```


17. Ahora debemos de poder llamar eventos al momento de presionar los botones.

```
<div class="calculator">

  <input type="text" class="calculator-screen" value="0" disabled />

  <div class="calculator-keys">

    <button type="button" class="operator" value="+"></button>
    <button type="button" class="operator" value="-"></button>
    <button type="button" class="operator" value="*">&times;</button>
    <button type="button" class="operator" value="/">&divide;</button>

    <button type="button" value="7">7</button>
    <button type="button" value="8">8</button>
    <button type="button" value="9">9</button>

    <button type="button" value="4">4</button>
    <button type="button" value="5">5</button>
    <button type="button" value="6">6</button>

    <button type="button" value="1">1</button>
    <button type="button" value="2">2</button>
    <button type="button" value="3">3</button>

    <button type="button" value="0">0</button>
    <button type="button" class="decimal" value=".">.</button>
    <button type="button" class="all-clear" value="all-clear">AC</button>

    <button type="button" class="equal-sign" value="=">=</button>

  </div>
</div>
```

The button element represents a button labeled by its contents.
[MDN Reference](#)

Nota: Este componente se encuentra en
src/app/calculator/calculatorComponent0.html

```

<div class="calculator">

  <input type="text" class="calculator-screen" [value]="currentNumber" disabled />

  <div class="calculator-keys">

    <button type="button" (click)="getOperation('+')" class="operator" value="+">+</button>
    <button type="button" (click)="getOperation('-')" class="operator" value="-">-</button>
    <button type="button" (click)="getOperation('*')" class="operator" value="*">&times;</button>
    <button type="button" (click)="getOperation('/')" class="operator" value="/">&divide;</button>

    <button type="button" (click)="getNumber('7')" value="7">7</button>
    <button type="button" (click)="getNumber('8')" value="8">8</button>
    <button type="button" (click)="getNumber('9')" value="9">9</button>

    <button type="button" (click)="getNumber('4')" value="4">4</button>
    <button type="button" (click)="getNumber('5')" value="5">5</button>
    <button type="button" (click)="getNumber('6')" value="6">6</button>

    <button type="button" (click)="getNumber('1')" value="1">1</button>
    <button type="button" (click)="getNumber('2')" value="2">2</button>
    <button type="button" (click)="getNumber('3')" value="3">3</button>

    <button type="button" (click)="getNumber('0')" value="0">0</button>
    <button type="button" (click)="getDecimal()" class="decimal" value=".">.</button>
    <button type="button" (click)="clear()" class="all-clear" value="all-clear">AC</button>

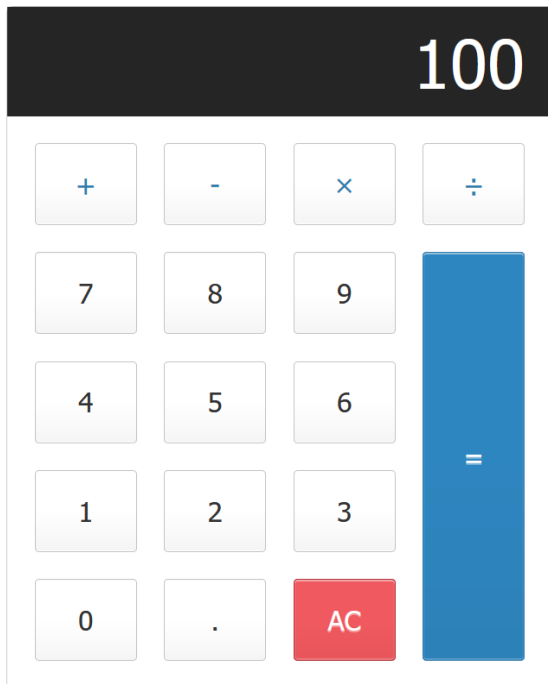
    <button type="button" (click) = "getOperation('=)" class="equal-sign" value="=">=</button>

  </div>

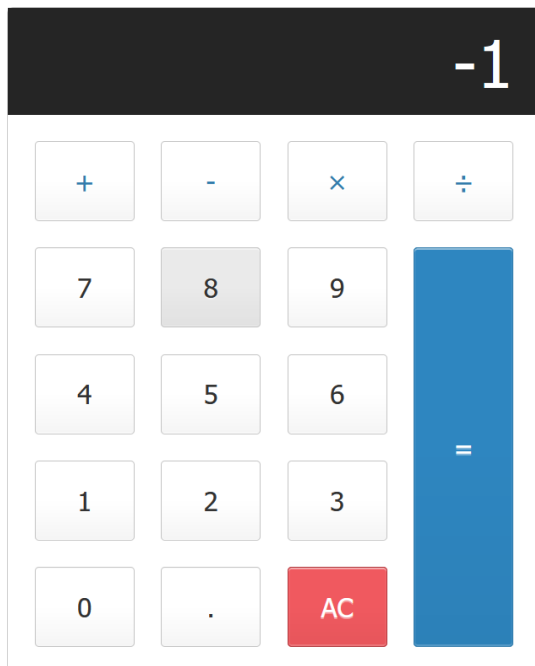
```

De esta manera utilizamos button(click) = “método” para poder llamar métodos que se encuentran .calculator.component.ts.

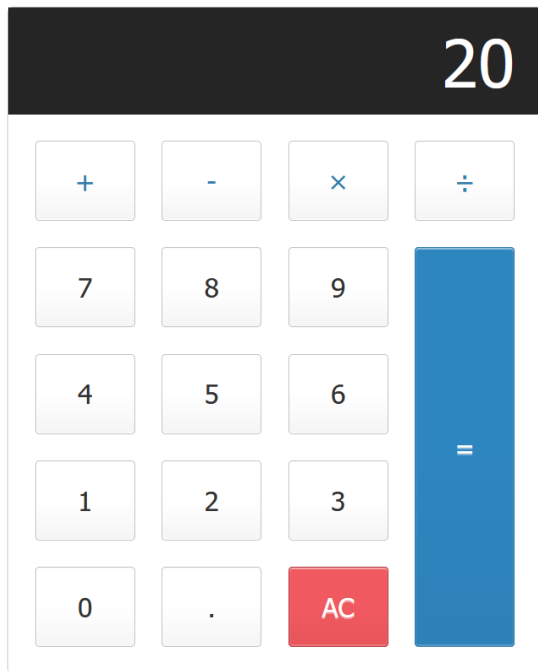
Realizaremos pruebas para verificar el funcionamiento. Al realizar la suma de $50+50$ nos dará el siguiente resultado:



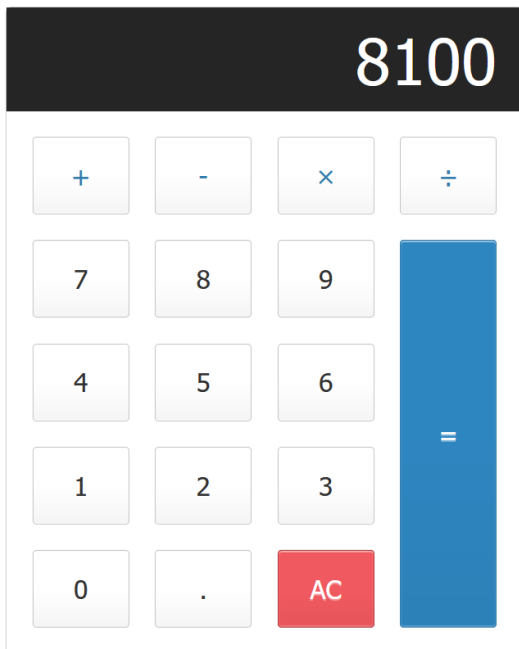
Al realizar una resta de $40-41$ nos dará el siguiente resultado:



Para la división utilizaremos los siguientes datos $100/5$



Para la multiplicación usaremos los siguientes datos 90×90



También podemos usar decimales:



Investigar cómo utilizar el enlace de eventos para vincular estos métodos a la plantilla (ver referencias).

De la siguiente manera podemos mandar a llamar eventos, los cuales en este caso se encuentran en nuestro archivo ts.

`<button class="btn btn-primary" (click)="clickAddTodo()">Add Todo</button>` al ser presionado mandar a llamar nuestros eventos, para ello debemos colocar nuestro método en el click. Además en estos metodos tambien podemos mandar parametros de la siguiente forma: `<button type="button" (click)="getOperation('+)" class="operator" value="+">+</button>` en donde el símbolo '+' es nuestro parámetro para el método `getOperation`.

Explicar el funcionamiento lógico de los métodos anteriores y cómo funciona el enlace de variables así como los métodos. [Documento word o presentación].

Nuestros métodos son llamados directamente desde el archivo `calculator.components.ts` y de esta forma se pueden ejecutar al presionar un botón.

Por ejemplo, el método **`getNumber()`** nos proporciona el primer número que el usuario presiona, para luego esperar el segundo número. De esta forma podemos seguir ingresando números sin que estos se sustituyan pues comenzamos a concatenar una vez nuestro **`waitForSecondNumber`** sea verdadero.

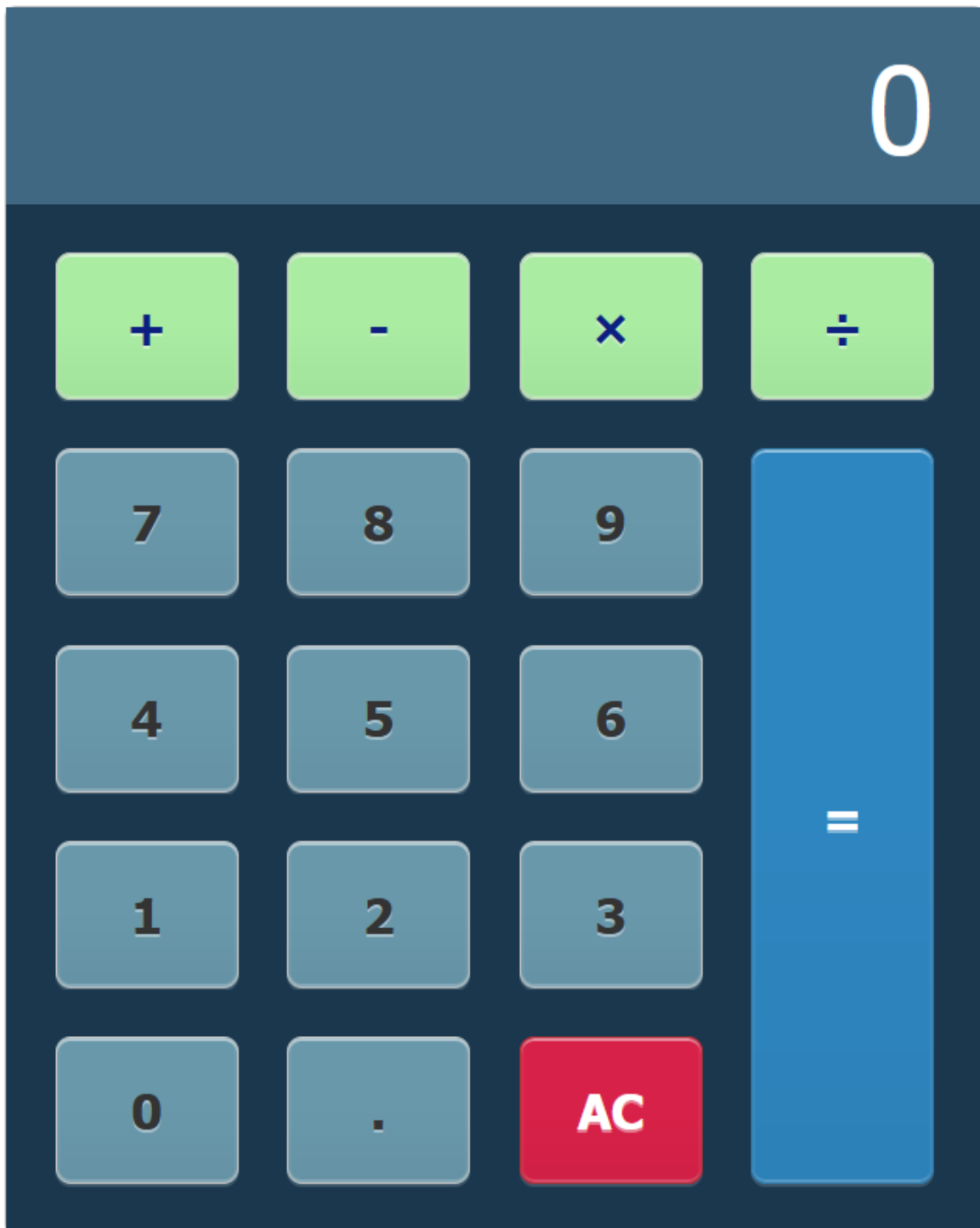
`getDecimal()` nos retorna un punto decimal cuando este es invocado desde la parte visual por medio de un botón, ya que cuando nosotros presionamos este botón enviamos por parámetro un punto decimal. De este modo, si nuestro parámetro sea diferente de un punto decimal, este se concatenan con la cadena que ya hemos enviado.

`doCalculation()` realiza las operaciones por medio de los dos parámetros y concatena los números que vayamos ingresando.

`getOperation` realiza las operaciones por medio de un switch que nos ayuda a entrar la las diferentes operaciones, de nuevo este método se activa al presionar un botón y al enviar un parametro, ya sea como suma +, resta -, division /, o multiplicacion *, etc.

El método **`clear()`** nos ayuda a limpiar nuestra pantalla al momento de presionar nuestro botón AC, reinicia nuestras variables al valor por defecto.

Le di un estilo diferente a mi calculadora, y quedó de la siguiente manera:



Utilizando gitHub para subir nuestro repositorio:

```
MINGW64/C:/Users/ene5/Desktop/examen-calculadora-1138153
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/app/app.component.html
        modified:   src/app/app.module.ts
        modified:   src/styles.css

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/app/calculator/

no changes added to commit (use "git add" and/or "git commit -a")

jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (master)
$ git add
git: 'add.' is not a git command. See 'git --help'.
The most similar command is
    add

jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (master)
$ git add .
warning: LF will be replaced by CRLF in src/app/app.module.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/styles.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/app/calculator/calculator.component.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/app/calculator/calculator.component.html.
The file will have its original line endings in your working directory
jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (master)
$ git commit -m "Examen de la primera unidad"
git: 'commit' is not a git command. See 'git --help'.
The most similar command is
    commit

jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (master)
$ git commit -m "Examen de la primera unidad"
[master f9f9a61] Examen de la primera unidad
6 files changed, 201 insertions(+), 485 deletions(-)
rewrite src/app/app.component.html (100%)
create mode 100644 src/app/calculator/calculator.component.css
create mode 100644 src/app/calculator/calculator.component.html
create mode 100644 src/app/calculator/calculator.component.ts

jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (master)
$ git remote add origin https://github.com/Shepharz/examen-calculadora-1138153.git
$ git branch -M main

jvelaz@LAPTOP-76377202H MINGW64 ~/Desktop/examen-calculadora-1138153 (main)
$ git push -u origin main
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 8 threads
Compressing objects: 100% (42/42), done.
Writing objects: 100% (48/48), 201.59 KiB | 8.98 MiB/s, done.
Total 48 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (0/0), done.
To https://github.com/Shepharz/examen-calculadora-1138153.git
 * [new branch]    main -> main
branch 'main' set up to track 'origin/main'.
```

Link: <https://github.com/Shepharz/examen-calculadora-1138153>

Conclusiones:

Aprendí el uso de los eventos por medio de botones, y reforcé lo que he aprendido en clase sobre TypeScript, sin embargo considero que aun me hace falta mucho para poder manejarlo desde cero. Este examen proporciona todos los elementos básicos de cómo realizar métodos con parámetros y con base a ello, entrenar con Angular el cual es un framework que me interesa mucho aprender.