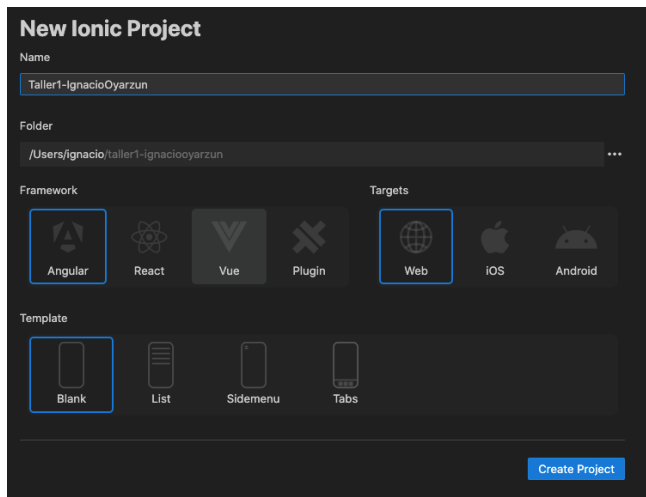


# Taller 1 – Programación Híbrida – IPLACEX

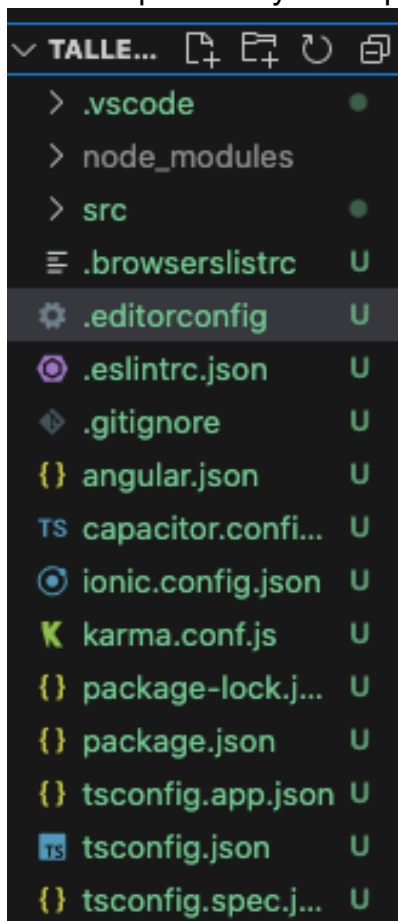
Ignacio Oyarzún  
Ingeniería en Informática  
23 de diciembre de 2024.

En esta ocasión nos encargaremos de explicar el desarrollo de nuestro primer taller de la asignatura de Programación Híbrida, donde logramos varios objetivos sin embargo costó implementar algunos elementos que se verán reflejados al final del informe.

Como primer paso procedemos a crear nuestro proyecto Ionic ya con todos los elementos necesarios instalados.



Dejamos todo configurado con las plantillas principales según lo establecido, damos inicio a nuestro proyecto y podemos observar que ya se nos creó nuestra aplicación y su respectiva distribución.



Posteriormente creamos nuestra carpeta “modelo” donde pondremos en práctica nuestra jerarquía de clases según lo visto en las instrucciones, como observación, en mi computador no logré abrir typescript desde el navegador, por lo que en el mismo VSCode utilicé el código de clases.

### Clase FiguraGeometrica

```
1  // Clase Principal Figura Geométrica
2
3  abstract class FiguraGeometrica
4  {
5      nombre:string;
6
7      constructor(nombre: string)
8      {
9          this.nombre = nombre;
10     }
11
12     abstract calcularPerimetro(): number;
13 }
```

### Clase Triangulo

```
1  // Clase Triangulo Escaleno que Hereda de Figura Geométrica
2
3  class TrianguloEscaleno extends FiguraGeometrica
4  {
5      ladoA: number;
6      ladoB: number;
7      ladoC: number;
8
9      constructor(ladoA: number, ladoB: number, ladoC: number)
10     {
11         super('Triángulo Escaleno');
12         this.ladoA = ladoA;
13         this.ladoB = ladoB;
14         this.ladoC = ladoC;
15     }
16
17     calcularPerimetro(): number
18     {
19         return this.ladoA + this.ladoB + this.ladoC;
20     }
21 }
22
23
24
25 //Clase Triangulo Equilátero que hereda de la Clase Triángulo Escaleno
26
27 class TrianguloEquilatero extends TrianguloEscaleno
28 {
29     constructor(lado: number)
30     {
31         super(lado, lado, lado);
32     }
33 }
```

Como podemos observar, clase triángulo hereda de la clase principal Figurateometrica, luego esta clase se subdivide en TrianguloEscaleno que hereda a TrianguloEquilatero.

Finalmente en términos de jerarquía tenemos nuestra clase circulo que hereda de la clase principal FiguraGeometrica.

```
1 // Clase que Hereda de Figura Geométrica
2
3 class Circulo extends FiguraGeometrica
4 {
5     radio: number;
6
7     constructor(radio: number)
8     {
9         super('Circulo');
10        this.radio = radio;
11    }
12
13    calcularPerimetro(): number
14    {
15        return 2 * Math.PI * this.radio;
16    }
17
18    resultado = this.calcularPerimetro();
19 }
```

Ya teniendo configurada nuestra jerarquía, procedemos a configurar nuestra página principal donde comenzará nuestra app web.

```
src > app > home > <> home.page.html > ion-content > div#container > strong
1 <ion-header [translucent]="true">
2   <ion-toolbar>
3     <ion-title>
4       Calculadora de Perímetros
5     </ion-title>
6   </ion-toolbar>
7 </ion-header>
8
9 <ion-content [fullscreen]="true">
10   <ion-header collapse="condense">
11     <ion-toolbar>
12       <ion-title size="large">Blank</ion-title>
13     </ion-toolbar>
14   </ion-header>
15
16   <div id="container">
17     <strong>Indicanos que figura quieres calcular</strong>
18   </div>
19
20   <ion-list>
21     <ion-item>
22       <ion-select label="Calculadora" placeholder="Seleccione una Figura"
23         (ionChange)="handleChange($event)">
24         <ion-select-option value="Circulo">Circulo</ion-select-option>
25         <ion-select-option value="Triangulo">Triangulo</ion-select-option>
26       </ion-select>
27     </ion-item>
28   </ion-list>
29
30   <app-triangulo *ngIf="componente == 'Triangulo'" />
31
32   <app-circulo *ngIf="componente == 'Circulo'" />
33
```

Como podemos observar se utiliza el evento “ionChange” el cual nos permitirá generar una acción al momento en que se seleccione el tipo de figura a calcular, así generar los distintos formularios, también utilizamos \*ngIf para mostrarnos el nombre del componente al ser seleccionado, todo esto mediante un elemento “IonList”.

Hablando del componente “ts” de nuestra página principal podemos encontrar lo siguiente:

```
src > app > home > ts home.page.ts > 📄 HomePage
1  import { Component, OnInit } from '@angular/core';
2  import { IonHeader, IonToolbar, IonTitle, IonContent, IonSelect, IonSelectOption, IonSelectCustomEvent } from '@ionic/core';
3  import { CirculoComponent } from '../circulo/circulo.component';
4  import { TrianguloComponent } from '../triangulo/triangulo.component';
5  import { IonSelectCustomEvent } from '@ionic/core';
6  import { CommonModule } from '@angular/common';
7
8  @Component({
9    selector: 'app-home',
10    templateUrl: 'home.page.html',
11    styleUrls: ['home.page.scss'],
12    imports: [IonHeader, IonToolbar, IonTitle, IonContent, IonSelect, IonSelectOption, IonSelectCustomEvent, CirculoComponent, TrianguloComponent]
13  })
14  export class HomePage implements OnInit {
15
16    componente = ""
17    constructor() {}
18
19    ngOnInit() {}
20
21    handleChange($event: IonSelectCustomEvent<SelectChangeEventDetail<any>>) {
22      this.componente = $event.target.value ?? ""
23    }
24
25  }
26
27
```

Importamos todos los elementos a utilizar en nuestra página, al momento de llamar a “IonChange” automáticamente se declara en nuestra sección “ts” de nuestra app.

Calculadora de Perímetros

Calculadora

Selecione una Figura ▼

Indicanos que figura quieres calcular

Calculadora

☐ Circulo

☐ Triangulo

CANCEL

OK

Ya con las recientes imágenes podemos observar que tenemos nuestra página principal ya lista, dándonos el paso a configurar nuestros formularios.

```
Go to component
1 <ion-card>
2   <ion-card-header>
3     <ion-card-title>Círculo</ion-card-title>
4   </ion-card-header>
5
6   <ion-card-content>
7     <ion-img src="https://dicciomat.com/wp-content/uploads/2024/06/como-se-calcula-el-perimetro-del-circulo.png" style="width: 100px; height: 100px; border-radius: 50%;"/>
8     <p>El perímetro de un círculo es la longitud de la línea que forma su borde.</p>
9     <ion-item>
10      <ion-label position="floating">Radio (cm)</ion-label>
11      <ion-input type="number" [(ngModel)]="radio"></ion-input>
12    </ion-item>
13
14    <ion-button (click)="calcularPerimetro()">CALCULAR PERÍMETRO</ion-button>
15
16    <ion-text>El perímetro es: {{ resultado }} cm</ion-text>
17  </ion-card-content>
18 </ion-card>
```

En el caso del formulario círculo utilizamos una etiqueta “Ion-Card” según lo solicitado en la prueba para crear nuestro formulario la cual contiene una etiqueta “Ion-Img” con una imagen de referencia además de nuestro input en el cual el usuario entregará la información de tipo número la cual puede escribir o seleccionar, en este caso también utilizamos un botón con el evento “Click” el cuál invocará a nuestro método. Al final el resultado se muestra mediante “Ion-Text”.

```
1 import { CommonModule } from '@angular/common';
2 import { Component, OnInit } from '@angular/core';
3 import { IonCard, IonItem, IonLabel, IonImg, IonButton, IonInput, IonCardTitle, IonCardHeader, IonCardContent, IonText } from '@ionic/angular';
4 import { FormsModule } from '@angular/forms';
5
6 @Component({
7   selector: 'app-circulo',
8   templateUrl: './circulo.component.html',
9   styleUrls: ['./circulo.component.scss'],
10   standalone: true,
11   imports: [IonText, IonCardContent, IonCardHeader, IonCardTitle, IonButton, IonImg, IonCard, IonItem, IonLabel, IonInput]
12 })
13 export class CirculoComponent implements OnInit {
14
15   resultado: number = 0;
16   radio: number = 0;
17   constructor() {}
18
19   ngOnInit() {}
20
21   calcularPerimetro() {
22     this.resultado = 2 * Math.PI * this.radio;
23   }
24 }
25
26
27
28
29
30
```

En nuestra sección “ts” procedemos a declarar nuestro método además de importar todo lo necesario para el correcto funcionamiento de nuestro programa, además de inicializar nuestras variables.

En la siguiente página procederemos a analizar nuestros códigos del componente “Triángulo”.

En este caso reutilizamos la misma estructura de Ionic utilizada para nuestro formulario anterior, con la diferencia que cambiamos las descripciones y la imagen, cabe mencionar que en ambas utilizamos CSS para poder regular el tamaño de las imágenes.

```
Go to component
1 <ion-card>
2   <ion-card-header>
3     <ion-card-title>Triángulo</ion-card-title>
4   </ion-card-header>
5
6   <ion-card-content>
7     <ion-img src="https://www.neurochispas.com/wp-content/uploads/2021/03/formula-del-perimetro-de-un-trianguulo-recta"
8     <p>El perímetro de un círculo es la suma de las longitudes de sus tres lados. En otras palabras, es la medida tot
9     <ion-item>
10      <ion-label position="floating">Lado A (cm)</ion-label>
11      <ion-input type="number" [(ngModel)]="LadoA"></ion-input>
12    </ion-item>
13    <ion-item>
14      <ion-label position="floating">Lado B (cm)</ion-label>
15      <ion-input type="number" [(ngModel)]="LadoB"></ion-input>
16    </ion-item>
17    <ion-item>
18      <ion-label position="floating">Lado C (cm)</ion-label>
19      <ion-input type="number" [(ngModel)]="LadoC"></ion-input>
20    </ion-item>
21
22    <ion-button (click)="calcularPerimetro()">CALCULAR PERÍMETRO</ion-button>
23
24    <ion-text>El perímetro es: {{ resultado }} cm</ion-text>
25  </ion-card-content>
26 </ion-card>
```

Procedemos a analizar nuestro código de la sección ts.

```
1 import { Component, OnInit } from '@angular/core';
2 import { IonHeader, IonToolbar, IonTitle } from '@ionic/angular/standalone';
3 import { CommonModule } from '@angular/common';
4 import { IonCard, IonItem, IonLabel, IonImg, IonButton, IonInput, IonCardTitle, IonCardHeader, IonCardContent, IonText
5 import { FormsModule } from '@angular/forms';
6
7 @Component({
8   selector: 'app-triangulo',
9   templateUrl: './triangulo.component.html',
10  styleUrls: ['./triangulo.component.scss'],
11  standalone: true,
12  imports: [IonTitle, IonToolbar, IonCard, IonItem, IonLabel, IonImg, IonButton, IonInput, IonCardTitle, IonCardHeader,
13 })
14 export class TrianguloComponent implements OnInit {
15
16   LadoA: number = 0
17   LadoB: number = 0
18   LadoC: number = 0
19   resultado: number = 0
20   constructor() { }
21
22   ngOnInit() {}
23
24   calcularPerimetro()
25   {
26     this.resultado = this.LadoA + this.LadoB + this.LadoC;
27   }
28
29
30 }
31
```

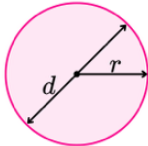
Como podemos notar, importamos todo lo necesario, además de inicializar nuestras variables y declarar el método a utilizar al momento de calcular el perímetro de un triángulo.

En esta última página procederemos a mostrar el funcionamiento de nuestros formularios.

Calculadora de Perímetros

CalculadoraCirculo

Círculo



$$P = 2 \cdot \pi \cdot r$$
$$= \pi \cdot d$$

El perímetro de un círculo es la longitud de la línea que forma su borde.

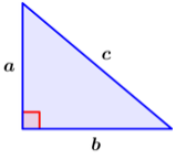
Radio (cm)  
5

CALCULAR PERÍMETROEl perímetro es: 31.41592653589793 cm

Calculadora de Perímetros

CalculadoraTriángulo

Triángulo



$$p = a + b + c$$

El perímetro de un círculo es la suma de las longitudes de sus tres lados. En otras palabras, es la medida total del contorno o borde del triángulo

Lado A (cm)  
1

Lado B (cm)  
1

Lado C (cm)  
1

CALCULAR PERÍMETROEl perímetro es: 3 cm

Con esto damos por finalizado el taller 1, las dificultades presentadas fueron principalmente la complicación de importar cada elemento a utilizar, además de complicaciones al utilizar \*ngIf para mostrar el resultado, no logré que se escondiera el resultado, para que apareciera después de apretar el botón, por lo que agradecería un feedback sobre esta prueba.