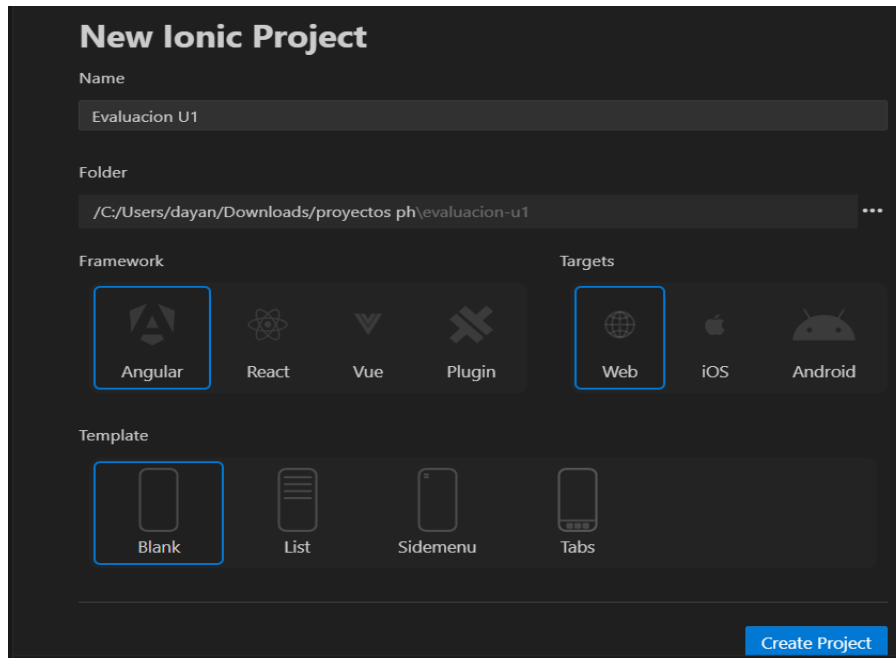
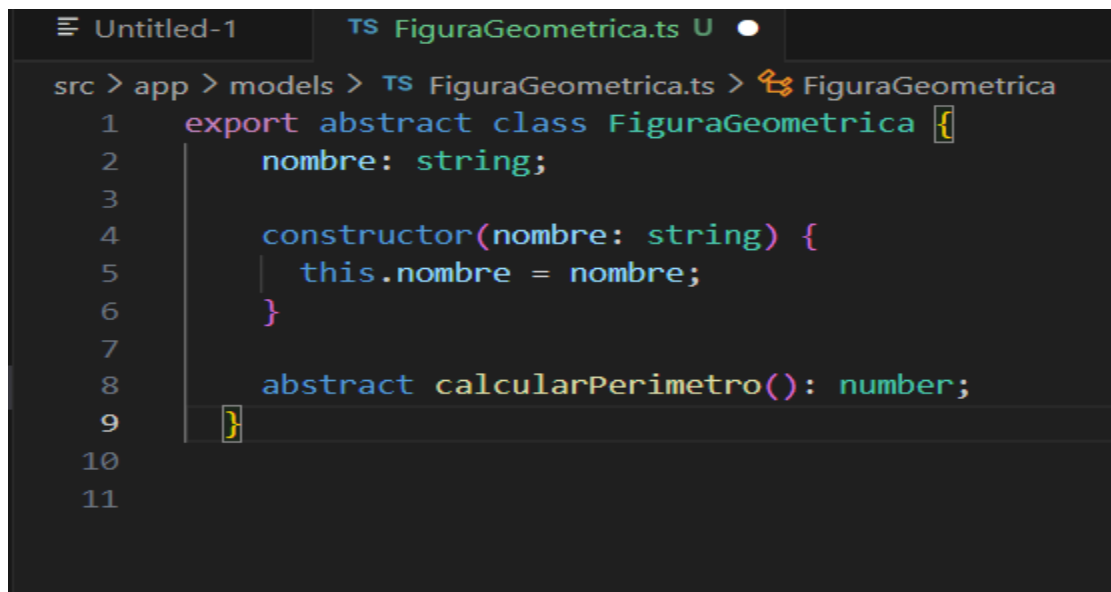


1. Crear proyecto de Ionic Framework usando la plantilla en blanco



2. Crear jerarquía de clases en Typescript según diagrama UML



-export abstract class FiguraGeometrica:

export: permite que esta clase pueda ser importada en otros archivos.

Abstract: indica que esta es una clase abstracta. Una clase abstracta no se puede instanciar directamente. Está diseñada para ser una base que otras clases pueden extender.

class FiguraGeometrica: define una clase llamada FiguraGeometrica.

-nombre: string; Define una propiedad llamada nombre de tipo string.

Esta propiedad almacenará el nombre de la figura geométrica.

-constructor(nombre: string):

Define un constructor para la clase FiguraGeometrica.

El constructor es una función especial que se llama cuando se crea una nueva instancia de la clase.

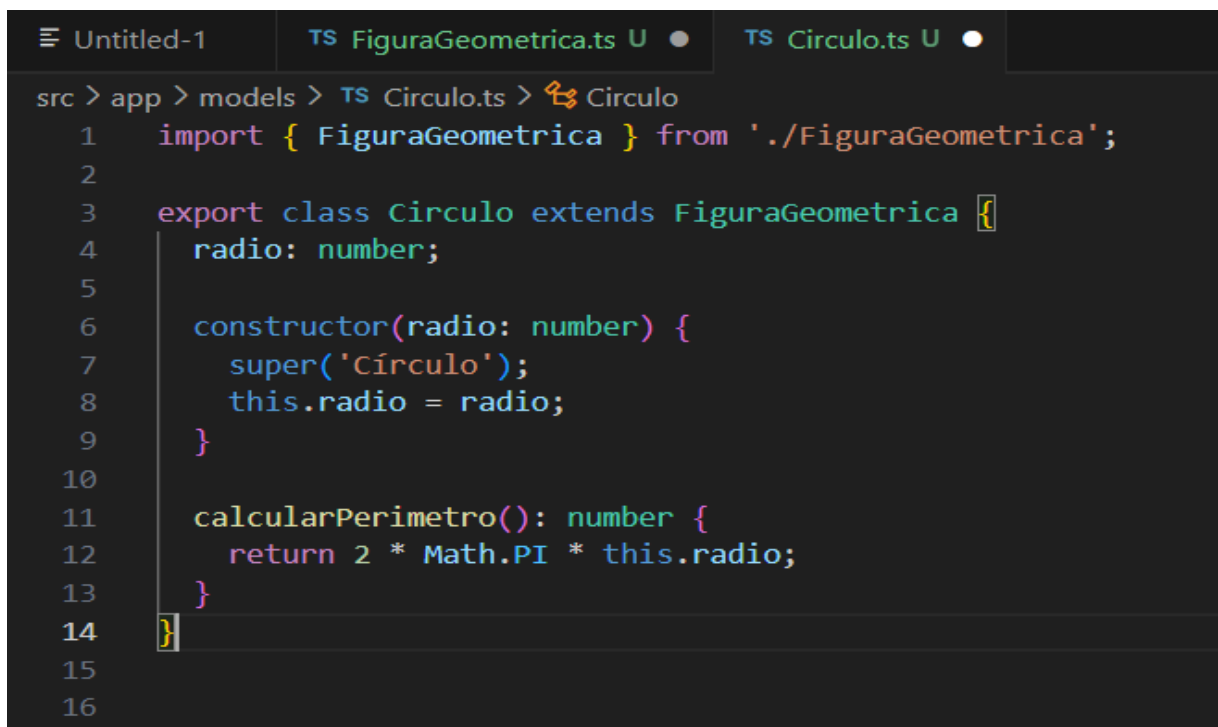
constructor(nombre: string): toma un argumento nombre de tipo string y asigna ese valor a la propiedad nombre de la instancia.


-abstract calcularPerimetro(): number;:

abstract: indica que este método no tiene una implementación en la clase FiguraGeometrica.

calcularPerimetro(): es un método abstracto que debe ser implementado por cualquier clase que extienda FiguraGeometrica.

: number: indica que este método debe devolver un valor de tipo number



```
src > app > models > TS Circulo.ts >  Circulo
1  import { FiguraGeometrica } from './FiguraGeometrica';
2
3  export class Circulo extends FiguraGeometrica {
4      radio: number;
5
6      constructor(radio: number) {
7          super('Círculo');
8          this.radio = radio;
9      }
10
11     calcularPerimetro(): number {
12         return 2 * Math.PI * this.radio;
13     }
14 }
15
16
```

-export class Circulo extends FiguraGeometrica { ... }:

Circulo: extiende FiguraGeometrica, lo que significa que hereda las propiedades y métodos de FiguraGeometrica.

-radio: number;: Define una propiedad llamada radio de tipo number.

-constructor(radio: number) { ... }:

Define un constructor para Circulo que toma un argumento radio de tipo number.

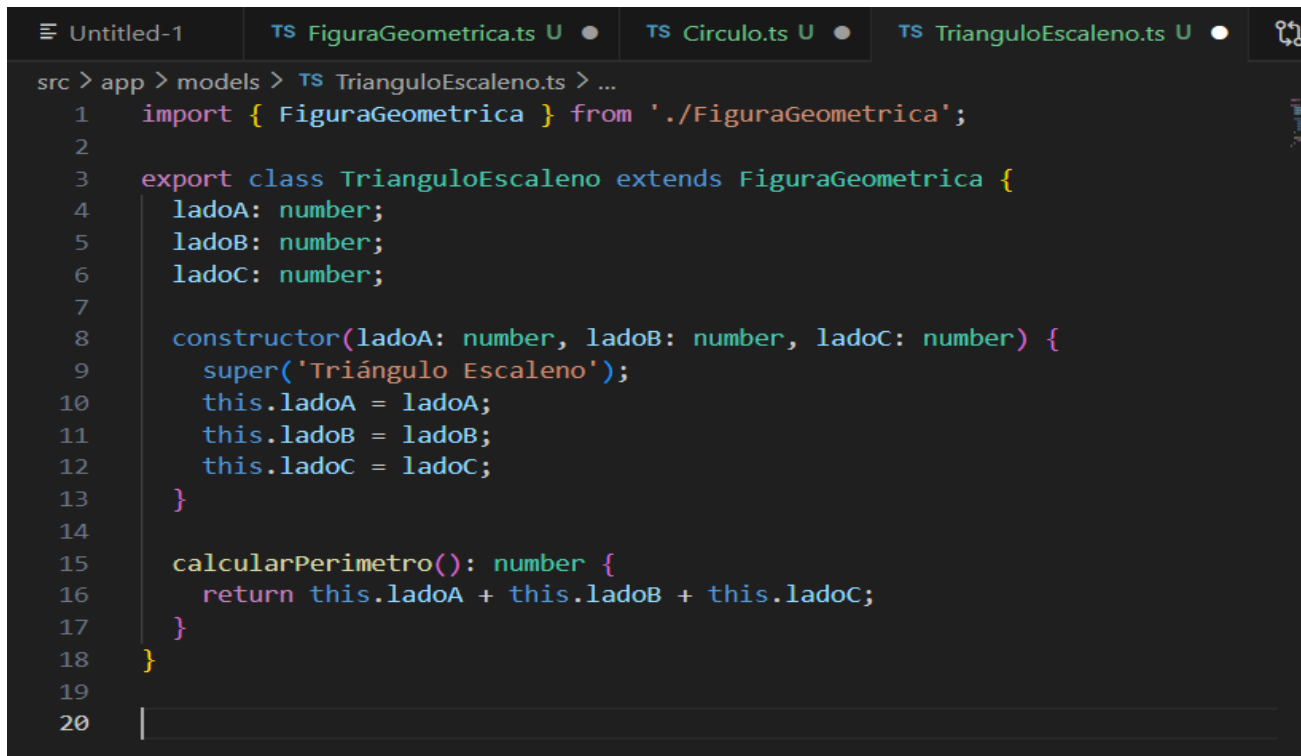
super('Círculo'); llama al constructor de la clase base (FiguraGeometrica) y le pasa el nombre de la figura ('Círculo').

this.radio = radio; asigna el valor del radio a la propiedad radio de la instancia.

-calcularPerimetro(): number { ... }:

Proporciona una implementación concreta del método abstracto calcularPerimetro().

Calcula el perímetro del círculo usando la fórmula $2 * \text{Math.PI} * \text{radio}$.



```
src > app > models > TS TrianguloEscaleno.ts > ...
1  import { FiguraGeometrica } from './FiguraGeometrica';
2
3  export class TrianguloEscaleno extends FiguraGeometrica {
4      ladoA: number;
5      ladoB: number;
6      ladoC: number;
7
8      constructor(ladoA: number, ladoB: number, ladoC: number) {
9          super('Triángulo Escaleno');
10         this.ladoA = ladoA;
11         this.ladoB = ladoB;
12         this.ladoC = ladoC;
13     }
14
15     calcularPerimetro(): number {
16         return this.ladoA + this.ladoB + this.ladoC;
17     }
18 }
19
20
```

-import { FiguraGeometrica } from './FiguraGeometrica';:

Esta línea importa la clase FiguraGeometrica desde el archivo FiguraGeometrica.ts.

Necesitamos esta importación porque TrianguloEscaleno extiende FiguraGeometrica.

-export class TrianguloEscaleno extends FiguraGeometrica { ... }:

export permite que esta clase pueda ser importada en otros archivos.

class TrianguloEscaleno define una nueva clase llamada TrianguloEscaleno.

extends FiguraGeometrica indica que TrianguloEscaleno es una subclase de FiguraGeometrica, heredando sus propiedades y métodos.

-ladoA: number; ladoB: number; ladoC: number;:

Estas líneas definen tres propiedades: ladoA, ladoB, y ladoC, todas de tipo number.

Estas propiedades representan los tres lados del triángulo escaleno.

-constructor(ladoA: number, ladoB: number, ladoC: number) { ... }:

Define un constructor para la clase TrianguloEscaleno.

constructor(ladoA: number, ladoB: number, ladoC: number) toma tres parámetros, cada uno representando uno de los lados del triángulo.

super('Triángulo Escaleno'); llama al constructor de la clase base (FiguraGeometrica) con el nombre de la figura ('Triángulo Escaleno').

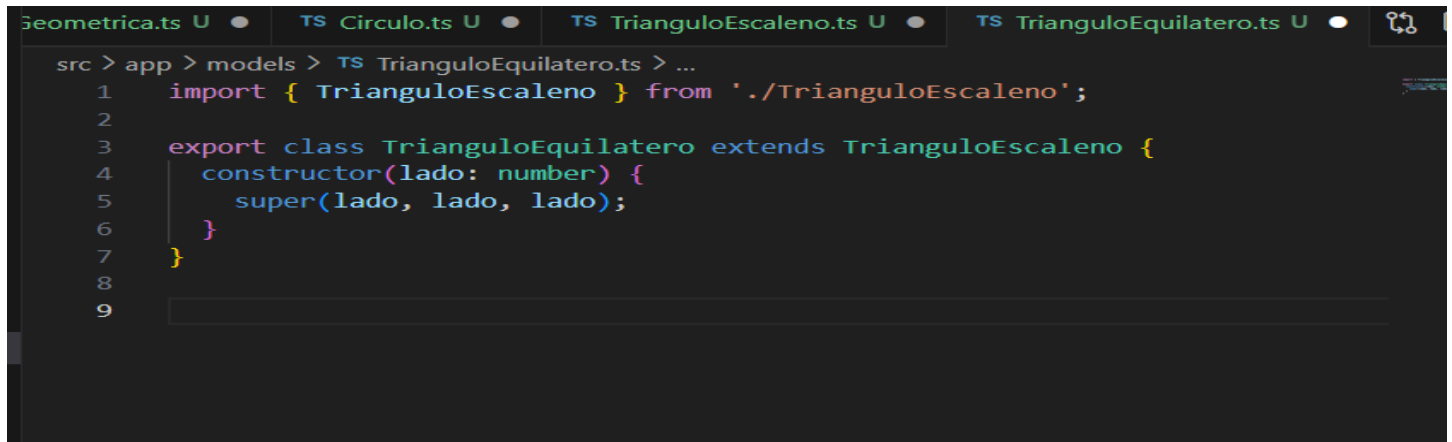
this.ladoA = ladoA; this.ladoB = ladoB; this.ladoC = ladoC; asigna los valores de los parámetros a las propiedades correspondientes de la instancia.

-calcularPerimetro(): number { ... }:

Este método proporciona una implementación concreta del método abstracto calcularPerimetro() definido en la clase base FiguraGeometrica.

calcularPerimetro(): number indica que este método no toma parámetros y devuelve un valor de tipo number.

return this.ladoA + this.ladoB + this.ladoC; calcula el perímetro del triángulo sumando los tres lados y devuelve el resultado.



```
5eometrica.ts U • TS Circulo.ts U • TS TrianguloEscaleno.ts U • TS TrianguloEquilatero.ts U •
src > app > models > TS TrianguloEquilatero.ts > ...
1  import { TrianguloEscaleno } from './TrianguloEscaleno';
2
3  export class TrianguloEquilatero extends TrianguloEscaleno {
4      constructor(lado: number) {
5          super(lado, lado, lado);
6      }
7  }
8
9
```

-import { TrianguloEscaleno } from './TrianguloEscaleno';:

Esta línea importa la clase TrianguloEscaleno desde el archivo TrianguloEscaleno.ts.

Necesitamos esta importación porque TrianguloEquilatero extiende TrianguloEscaleno.

-export class TrianguloEquilatero extends TrianguloEscaleno { ... }:

export permite que esta clase pueda ser importada en otros archivos.

class TrianguloEquilatero define una nueva clase llamada TrianguloEquilatero.

extends TrianguloEscaleno indica que TrianguloEquilatero es una subclase de TrianguloEscaleno, heredando sus propiedades y métodos.

-constructor(lado: number) { ... }:

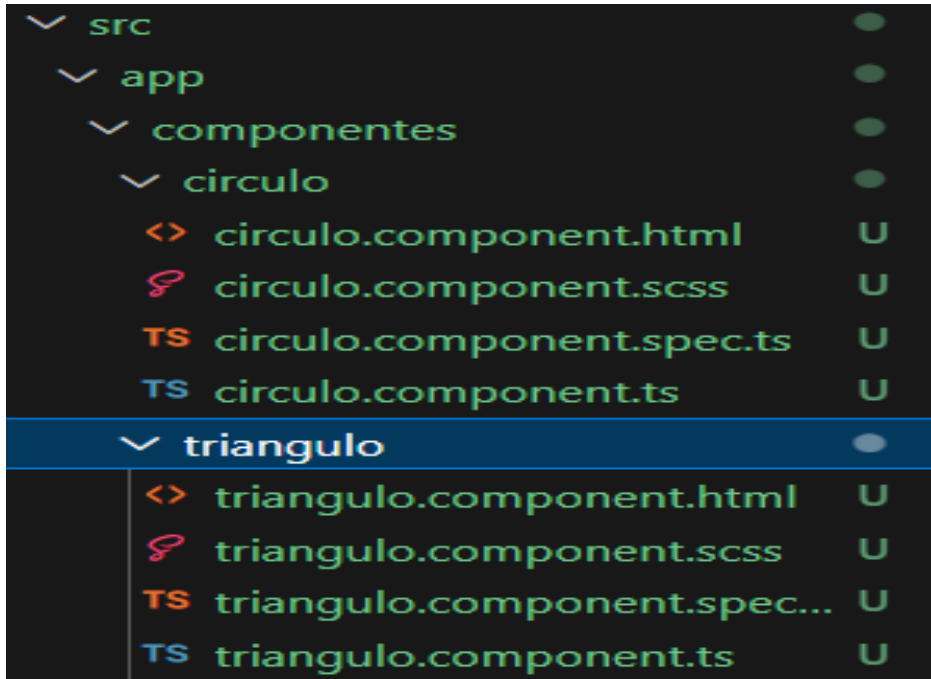
Define un constructor para la clase TrianguloEquilatero.

constructor(lado: number) toma un parámetro lado de tipo number, que representa la longitud de los tres lados del triángulo equilátero.

super(lado, lado, lado); llama al constructor de la clase base (TrianguloEscaleno) pasando el mismo valor lado para los tres lados (ladoA, ladoB, y ladoC).

Esto es porque en un triángulo equilátero, todos los lados son iguales.

3. Crear 2 componentes Standalone de Angular dentro del proyecto: círculo y triángulo



4. Incorporar un elemento ion-select en el componente "home"
5. Usando la directiva *ngIf muestre el componente que corresponda según se seleccione en ion-select

```
<> home.page.html M X
src > app > home > <> home.page.html > ion-content
Go to component
1 <ion-header>
2   <ion-toolbar>
3     <ion-title>
4       Perímetro de Figuras Geométricas
5     </ion-title>
6   </ion-toolbar>
7 </ion-header>
8
9 <ion-content>
10  <ion-item>
11    <ion-label>Selecciona la figura</ion-label>
12    <ion-select [(ngModel)]="selectedFigura" placeholder="Seleccione uno">
13      <ion-select-option value="circulo">Círculo</ion-select-option>
14      <ion-select-option value="triangulo">Triángulo</ion-select-option>
15    </ion-select>
16  </ion-item>
17
18  <ng-container *ngIf="selectedFigura === 'circulo'">
19    <app-circulo></app-circulo>
20  </ng-container>
21
22  <ng-container *ngIf="selectedFigura === 'triangulo'">
23    <app-triangulo></app-triangulo>
24  </ng-container>
25 </ion-content>
```

Home.page.html:

-<ion-header> ... </ion-header>

Este bloque define la cabecera de la página.

<ion-header>: Contenedor para el encabezado de la página.

<ion-toolbar>: Barra de herramientas que contiene elementos de encabezado como títulos y botones.

<ion-title>: Título del encabezado. Aquí se muestra "Perímetro de Figuras Geométricas".

-<ion-content> ... </ion-content>

Este bloque define el contenido principal de la página.

<ion-content>: Contenedor principal para el contenido de la página.

<ion-item>: Contenedor para elementos de interfaz de usuario como etiquetas, entradas, etc.

<ion-label>: Etiqueta que describe el propósito del control asociado. Aquí dice "Selecciona la figura".

<ion-select>: Menú desplegable para seleccionar una opción.

[(ngModel)]="selectedFigura": Enlace de datos bidireccional a la propiedad selectedFigura del componente. Esto significa que cuando el usuario selecciona una opción, selectedFigura se actualiza con el valor seleccionado.

placeholder="Seleccione uno": Texto que se muestra cuando no se ha seleccionado ninguna opción.

<ion-select-option>: Opciones dentro del menú desplegable. Aquí hay dos opciones: "Círculo" y "Triángulo".

-*ngIf y <ng-container>

Estos elementos manejan la lógica de visualización condicional en Angular.

<ng-container>: Contenedor lógico que no produce ningún elemento HTML real, pero puede contener directivas estructurales como *ngIf.

*ngIf="selectedFigura === 'circulo'": Directiva que incluye el contenido del contenedor solo si selectedFigura es igual a 'circulo'.

<app-circulo></app-circulo>: Componente personalizado para manejar la lógica y la visualización de un círculo.

*ngIf="selectedFigura === 'triangulo'": Directiva que incluye el contenido del contenedor solo si selectedFigura es igual a 'triangulo'.

<app-triangulo></app-triangulo>: Componente personalizado para manejar la lógica y la visualización de un triángulo.

```
TS home.page.ts M X
src > app > home > TS home.page.ts > 📄 HomePage
1  import { Component } from '@angular/core';
2  import { IonicModule } from '@ionic/angular';
3  import { CommonModule } from '@angular/common';
4  import { FormsModule } from '@angular/forms';
5  import { CirculoComponent } from '../componentes/circulo/circulo.component';
6  import { TrianguloComponent } from '../componentes/triangulo/triangulo.component';
7
8  @Component({
9    selector: 'app-home',
10   templateUrl: './home.page.html',
11   styleUrls: ['./home.page.scss'],
12   standalone: true,
13   imports: [IonicModule, CommonModule, FormsModule, CirculoComponent, TrianguloComponent]
14 })
15 export class HomePage {
16   selectedFigura: string = ''; // Inicialización de la propiedad
17
18   constructor() {}
19 }
```

Home.page.ts:

Importaciones:

Component de @angular/core: Decorador que convierte una clase TypeScript en un componente de Angular.

IonicModule de @ionic/angular: Módulo que incluye componentes y servicios específicos de Ionic.

CommonModule de @angular/common: Módulo que proporciona directivas comunes de Angular como ngIf y ngFor.

FormsModule de @angular/forms: Módulo que permite el uso de formularios y la vinculación de datos en Angular.

CirculoComponent y TrianguloComponent: Importación de componentes personalizados para manejar la lógica y visualización de las figuras geométricas.

Decorador @Component:

selector: 'app-home': Nombre del selector que se utiliza para instanciar este componente en una plantilla.

templateUrl: './home.page.html': Ruta al archivo HTML que define la plantilla de este componente.

styleUrls: ['./home.page.scss']: Ruta al archivo de estilos SCSS específico de este componente.

standalone: true: Indica que este componente es autónomo y no necesita declararse en un módulo de Angular tradicional.

imports: [IonicModule, CommonModule, FormsModule, CirculoComponent, TrianguloComponent]: Módulos y componentes que este componente necesita para funcionar.

Clase del Componente HomePage:

export class HomePage: Define la clase HomePage que será el controlador de la lógica para la plantilla home.page.html.

selectedFigura: string = ";;: Define e inicializa la propiedad selectedFigura que se utilizará para almacenar la figura seleccionada por el usuario.

constructor() {}: Constructor de la clase. En este caso, no realiza ninguna acción, pero puede ser usado para inyectar servicios o inicializar variables si fuera necesario.

6. Crear formularios dentro de cada componente (círculo y triángulo)


```
<> circulo.component.html U ●
src > app > componentes > circulo > <> circulo.component.html > ion-card
Go to component
1  <ion-card>
2    <ion-card-header>
3      <ion-card-title>Círculo</ion-card-title>
4    </ion-card-header>
5    <ion-card-content>
6      <ion-img src="assets/circulo.png"></ion-img>
7      <ion-item>
8        <ion-label>Radio</ion-label>
9        <ion-input [(ngModel)]="radio" type="number"></ion-input>
10     </ion-item>
11     <ion-button (click)="calcular()">Calcular Perímetro</ion-button>
12     <ion-label *ngIf="resultado">Perímetro: {{ resultado }}</ion-label>
13   </ion-card-content>
14 </ion-card>
```

-<ion-card> ... </ion-card>

Este bloque define una tarjeta (ion-card) que contendrá los elementos necesarios para calcular el perímetro de un círculo.

-<ion-card-header> ... </ion-card-header>

<ion-card-header>: Define el encabezado de la tarjeta.

<ion-card-title>: Contiene el título del encabezado, en este caso "Círculo".

-<ion-card-content> ... </ion-card-content>

Este bloque contiene los elementos principales de la tarjeta.

<ion-img src="assets/circulo.png"></ion-img>

Propósito: Muestra una imagen del círculo.

src="assets/circulo.png": Especifica la ruta de la imagen a mostrar.

<ion-item> ... </ion-item>

Propósito: Contiene el campo de entrada del radio y su etiqueta.

<ion-label>Radio</ion-label>: Etiqueta que indica que el siguiente campo de entrada es para el radio.

<ion-input [(ngModel)]="radio" type="number"></ion-input>:

Propósito: Campo de entrada para que el usuario introduzca el valor del radio.

[(ngModel)]="radio": Enlace de datos bidireccional que conecta el valor del input con la propiedad radio en el componente Angular. Esto significa que cuando el usuario escribe en el input, la propiedad radio se actualiza automáticamente con el valor ingresado.

type="number": Especifica que el tipo de entrada es numérico.

`<ion-button (click)="calcular()">Calcular Perímetro</ion-button>`

Propósito: Botón que, al ser presionado, ejecuta la función `calcular()` en el componente Angular.

`(click)="calcular()"`: Vincula el evento de clic del botón con la función `calcular()`, que debe estar definida en el componente.

`<ion-label *ngIf="resultado">Perímetro: {{ resultado }}</ion-label>`

Propósito: Muestra el perímetro calculado si está disponible.

`*ngIf="resultado"`: Directiva de Angular que hace que esta etiqueta solo se muestre si `resultado` tiene un valor.

`{{ resultado }}`: Interpolación de Angular que muestra el valor de `resultado` dentro de la etiqueta.

```
TS círculo.component.ts U X
src > app > componentes > círculo > TS círculo.component.ts > ...
1  import { Component } from '@angular/core';
2  import { IonicModule } from '@ionic/angular';
3  import { FormsModule } from '@angular/forms';
4  import { CommonModule } from '@angular/common';
5  import { Circulo } from '../../models/Circulo';
6
7  @Component({
8    selector: 'app-circulo',
9    templateUrl: './circulo.component.html',
10   styleUrls: ['./circulo.component.scss'],
11   standalone: true,
12   imports: [IonicModule, FormsModule, CommonModule]
13 })
14 export class CirculoComponent {
15   radio: number = 0;
16   resultado: number = 0;
17
18   calcular() {
19     if (this.radio) {
20       const circulo = new Circulo(this.radio);
21       this.resultado = circulo.calcularPerimetro();
22     }
23   }
24 }
```

Importaciones:

Component de `@angular/core`: Decorador que convierte una clase TypeScript en un componente Angular.

IonicModule de `@ionic/angular`: Módulo que contiene componentes y servicios específicos de Ionic.

FormsModule de `@angular/forms`: Módulo que permite el uso de formularios y la vinculación de datos en Angular.

CommonModule de @angular/common: Módulo que proporciona directivas comunes de Angular como ngIf y ngFor.

Circulo de ../../models/Circulo: Importa la clase Circulo desde el archivo de modelos, que se utiliza para encapsular la lógica del cálculo del perímetro de un círculo.

Decorador @Component:

selector: 'app-circulo': Nombre del selector que se utiliza para instanciar este componente en una plantilla.

templateUrl: './circulo.component.html': Ruta al archivo HTML que define la plantilla de este componente.

styleUrls: ['./circulo.component.scss']: Ruta al archivo de estilos SCSS específico de este componente.

standalone: true: Indica que este componente es autónomo y no necesita declararse en un módulo de Angular tradicional.

imports: [IonicModule, FormsModule, CommonModule]: Módulos que este componente necesita para funcionar.

Clase del Componente CirculoComponent:

export class CirculoComponent: Define la clase CirculoComponent que será el controlador de la lógica para la plantilla circulo.component.html.

radio: number = 0: Propiedad que almacena el valor del radio ingresado por el usuario, inicializada en 0.

resultado: number = 0: Propiedad que almacena el resultado del cálculo del perímetro, inicializada en 0.

calcular(): Método que calcula el perímetro del círculo utilizando la fórmula $2 \times \pi \times \text{radio}$

if (this.radio): Verifica si radio tiene un valor (es decir, no es null ni undefined ni 0).

const circulo = new Circulo(this.radio): Crea una nueva instancia de la clase Circulo pasando el valor del radio.

this.resultado = circulo.calcularPerimetro(): Llama al método calcularPerimetro de la instancia circulo y asigna el resultado a la propiedad resultado.

```
<> triangulo.component.html U X
src > app > componentes > triangulo > <> triangulo.component.html > ion-card
Go to component
1 <ion-card>
2   <ion-card-header>
3     <ion-card-title>Triángulo Escaleno</ion-card-title>
4   </ion-card-header>
5   <ion-card-content>
6     <ion-img src="assets/triangulo.png"></ion-img>
7     <ion-item>
8       <ion-label>Lado A</ion-label>
9       <ion-input [(ngModel)]="ladoA" type="number"></ion-input>
10    </ion-item>
11    <ion-item>
12      <ion-label>Lado B</ion-label>
13      <ion-input [(ngModel)]="ladoB" type="number"></ion-input>
14    </ion-item>
15    <ion-item>
16      <ion-label>Lado C</ion-label>
17      <ion-input [(ngModel)]="ladoC" type="number"></ion-input>
18    </ion-item>
19    <ion-button (click)="calcular()">Calcular Perímetro</ion-button>
20    <ion-label *ngIf="resultado">Perímetro: {{ resultado }}</ion-label>
21  </ion-card-content>
22 </ion-card>
```

`<ion-card>`: Es un componente de Ionic que se utiliza para mostrar contenido estructurado de manera visualmente atractiva, similar a una tarjeta. En este caso, encapsula todo el contenido relacionado con la información de un triángulo escaleno.

`<ion-card-header>`: Este elemento define la cabecera de la tarjeta. En tu caso, contiene el título del card que es "Triángulo Escaleno".

`<ion-card-title>`: Es un componente que se utiliza dentro de `<ion-card-header>` para mostrar un título específico. En tu caso, muestra el texto "Triángulo Escaleno".

`<ion-card-content>`: Es donde se coloca el contenido principal de la tarjeta. Aquí se encuentran todos los elementos que forman parte de la información del triángulo escaleno.

`<ion-img>`: Es un componente de Ionic para mostrar imágenes. En este caso, muestra una imagen del triángulo que se encuentra en la ruta especificada por `src="assets/triangulo.png"`.

`<ion-item>`: Este componente se utiliza para representar un ítem dentro de una lista de elementos. Aquí se usa para crear un formulario donde se ingresan los lados del triángulo (lado A, lado B, lado C).

`<ion-label>`: Sirve para agregar etiquetas de texto descriptivas dentro de los ítems de `<ion-item>`.

`<ion-input>`: Es un componente de entrada de datos que permite al usuario ingresar valores. Se utiliza con `[(ngModel)]` para enlazar los valores de entrada a las variables `ladoA`, `ladoB` y `ladoC` en tu controlador (probablemente una clase TypeScript).

<ion-button>: Componente de Ionic que representa un botón interactivo. En este caso, al hacer clic en el botón se activa la función calcular() que realiza el cálculo del perímetro del triángulo.

*ngIf="resultado": Es una directiva estructural de Angular que muestra o oculta un elemento en función de una condición. En este caso, se muestra el texto "Perímetro: {{ resultado }}" solo si la variable resultado tiene algún valor (es decir, después de calcular el perímetro).

```
TS triangulo.component.ts U X
src > app > componentes > triangulo > TS triangulo.component.ts > ...
1  import { Component } from '@angular/core';
2  import { IonicModule } from '@ionic/angular';
3  import { FormsModule } from '@angular/forms';
4  import { CommonModule } from '@angular/common';
5  import { TrianguloEscaleno } from '../../models/TrianguloEscaleno';
6
7  @Component({
8    selector: 'app-triangulo',
9    templateUrl: './triangulo.component.html',
10   styleUrls: ['./triangulo.component.scss'],
11   standalone: true,
12   imports: [IonicModule, FormsModule, CommonModule]
13 })
14 export class TrianguloComponent {
15   ladoA: number = 0; // Inicialización de propiedades
16   ladoB: number = 0; // Inicialización de propiedades
17   ladoC: number = 0; // Inicialización de propiedades
18   resultado: number = 0; // Inicialización de propiedades
19
20   calcular() {
21     if (this.ladoA && this.ladoB && this.ladoC) {
22       const triangulo = new TrianguloEscaleno(this.ladoA, this.ladoB, this.ladoC);
23       this.resultado = triangulo.calcularPerimetro();
24     }
25   }
26 }
```

Importaciones:

@angular/core: Importa Component de Angular, necesario para definir componentes.

@ionic/angular: Importa módulos de Ionic necesarios para la interfaz de usuario.

@angular/forms: Importa módulos de Angular para manejar formularios, incluyendo ngModel.

@angular/common: Importa módulos comunes de Angular.

TrianguloEscaleno: Importa la clase TrianguloEscaleno desde el archivo TrianguloEscaleno ubicado en ../../models/.

Decorador @Component:

selector: Define el selector CSS que Angular usará para identificar este componente en otros archivos de plantilla.

templateUrl: Especifica la ubicación del archivo HTML que define la plantilla del componente.

styleUrls: Especifica la ubicación del archivo SCSS que define los estilos para el componente.

standalone: true: Esto es una propiedad personalizada que no es estándar en Angular. Podría ser una configuración específica del proyecto para marcar que este componente es independiente o autónomo en su funcionalidad.

imports: Importa los módulos necesarios para este componente, que incluyen IonicModule, FormsModule y CommonModule.

Clase TrianguloComponent:

ladoA, ladoB, ladoC, resultado: Son propiedades de la clase TrianguloComponent. ladoA, ladoB, y ladoC son inicializadas a 0, y resultado a 0. Estas variables almacenan los lados del triángulo y el resultado del cálculo del perímetro respectivamente.

calcular(): Método que se invoca cuando se hace clic en el botón "Calcular Perímetro". Verifica si los valores de ladoA, ladoB y ladoC son diferentes de cero (lo cual se interpreta como un valor válido para un lado de un triángulo). Si son válidos, crea una instancia de TrianguloEscaleno con estos lados y llama al método calcularPerimetro() de la clase TrianguloEscaleno, asignando el resultado a this.resultado.

7. Usar el evento “click” sobre un elemento ion-button para desencadenar el cálculo del perímetro

El código <ion-button (click)="calcular()">Calcular Perímetro</ion-button>: es un componente de botón en Ionic que tiene un evento de clic asociado a él.

<ion-button>: Este es un componente de Ionic que representa un botón interactivo en la interfaz de usuario. Es parte de la librería de componentes de Ionic y proporciona estilos y funcionalidades predefinidas para crear botones consistentes en aplicaciones móviles y web.

(click)="calcular()": Esta es una directiva de Angular que se usa para escuchar el evento de clic en el botón. Cuando el usuario hace clic en el botón, se ejecuta la función calcular() que está definida en el componente TypeScript asociado.

Calcular Perímetro: Este es el texto visible dentro del botón. Es lo que el usuario verá y sobre lo que puede hacer clic para activar la acción definida por la función calcular().

8. Usar ion-card e ion-image para colocar más información de la figura geométrica

Esta función esta en triangulo.component.html y circulo.component.html

<ion-img src="assets/circulo.png"></ion-img> y <ion-img src="assets/triangulo.png"></ion-img>

<ion-img>: es un componente específico de Ionic diseñado para manejar imágenes de manera eficiente en aplicaciones híbridas. Aunque se parece a la etiqueta estándar HTML , Ionic proporciona este componente para aprovechar las funcionalidades adicionales que puede ofrecer, como la carga diferida de imágenes y el soporte para varias resoluciones de pantalla.

src="assets/circulo.png": El atributo src especifica la ruta de la imagen que se va a mostrar. En este caso, "assets/circulo.png" indica que la imagen circulo.png se encuentra en la carpeta assets.

Perímetro de Figuras Geométricas

Selecciona la figura

Selecione uno ▼

☐

Círculo

☐

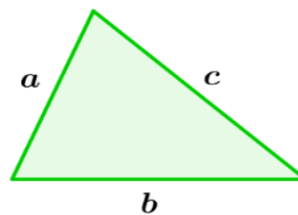
Triángulo

CANCEL

OK

Perímetro de Figuras Geométricas

Triángulo Escaleno



Lado₁₀
A

Lado₂₀
B

Lado₁₅
C

CALCULAR PERÍMETRO

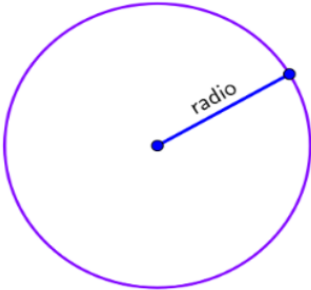
Perímetro: 45

Perímetro de Figuras Geométricas

Selecciona la figura

Círculo ▼

Círculo



Radio15

CALCULAR PERÍMETRO

Perímetro: 94.24777960769379