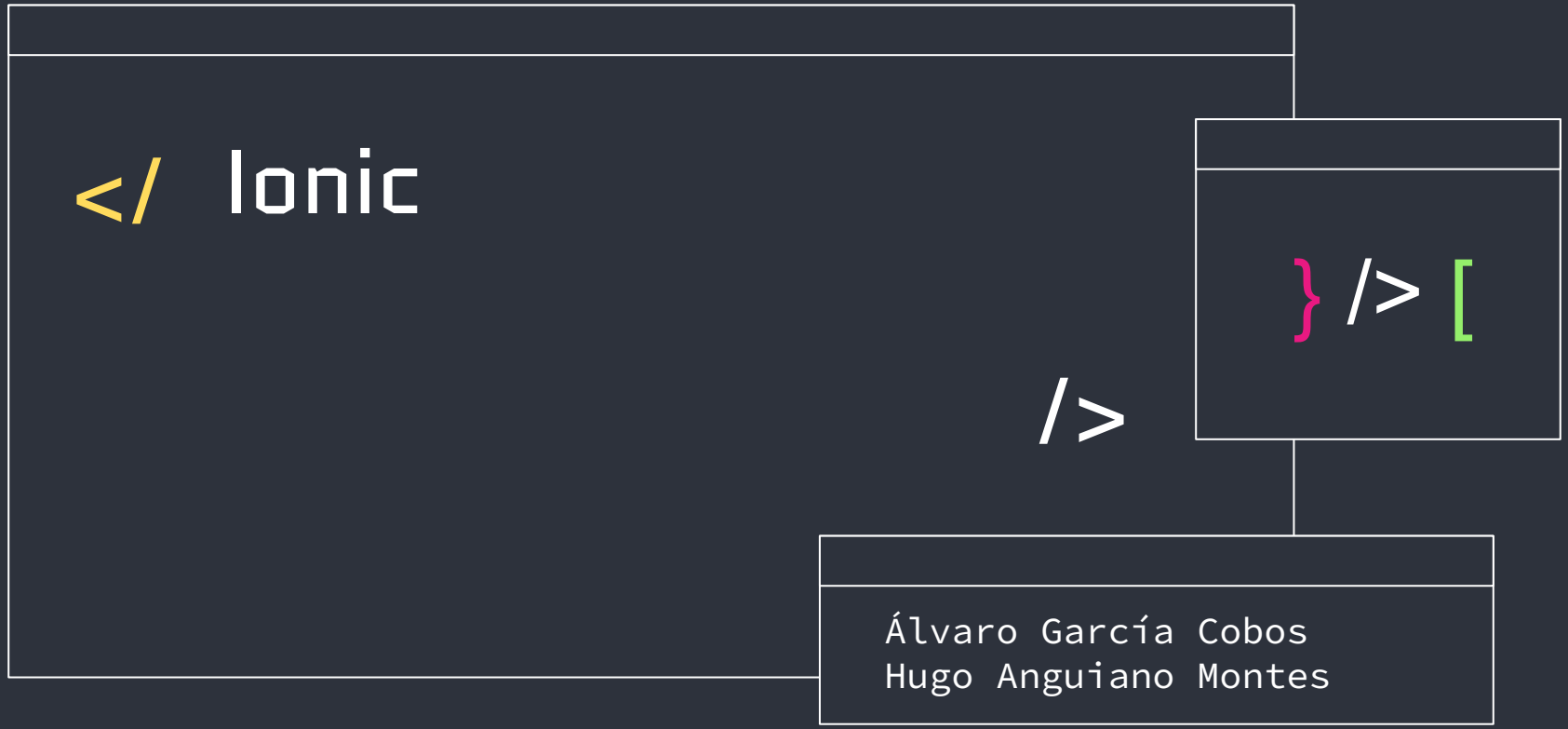


1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Instalación

```
PS C:\Users\hugoa\Desktop\Ionic> npm -v
10.8.2
PS C:\Users\hugoa\Desktop\Ionic> node -v
v20.19.5
PS C:\Users\hugoa\Desktop\Ionic>
```

```
PS C:\Users\hugoa\Desktop\Ionic> ng --version
20.3.3
PS C:\Users\hugoa\Desktop\Ionic> ionic --version
7.2.1
PS C:\Users\hugoa\Desktop\Ionic>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

¿Qué es Ionic?

01

} /> [

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

</ Ionic

/> **

} /> [

Ionic es un framework para crear aplicaciones para móvil usando ts, html y css. Un mismo código funciona en Android, IOS y web.
App móvil pero es una web metida dentro de una aplicación.
Con capacitor se pueden usar las funciones del teléfono como el GPS.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Capacitor

/> **

} /> [

Herramienta puente entre móvil y web.
Permite que la web funcione como una app real para Android e IOS.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Qué vamos a hacer

1º

- Crear proyecto.
- Desarrollar el proyecto en Vs code.
- Probar en la web (ionic serve).

2º

- Pasar a móvil (ionic build).
- Añadir a Android studio (ionic cap add android).
- Ionic cap sync (Sincronizar).
- Abrir en android(ionic cap open android).

Conclusion

Creas todo como si fuese una web y en 3 minutos con pocos comandos, lo tienes en una app móvil.
💯💯

</>

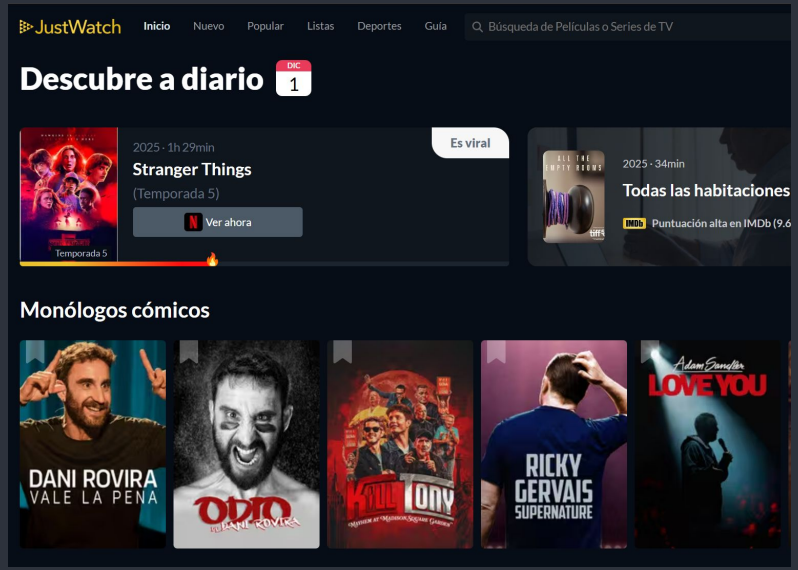
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ uso 💯% real de ionic

Se usa en muchas empresas porque permite ahorrar tiempo y recursos, sobretodo startups y empresas pequeñas.

JustWatch

Justwatch es una plataforma que muestra en qué sitio está disponible una película o serie. Tanto para web como para móvil.



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Estructura

- Angular Standalone → Organizado en componentes que no necesitan declararse en un módulo de Angular (NgModule).
- Componentes:
    - HTML (vista)
    - TypeScript (lógica)
    - CSS (styles)

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

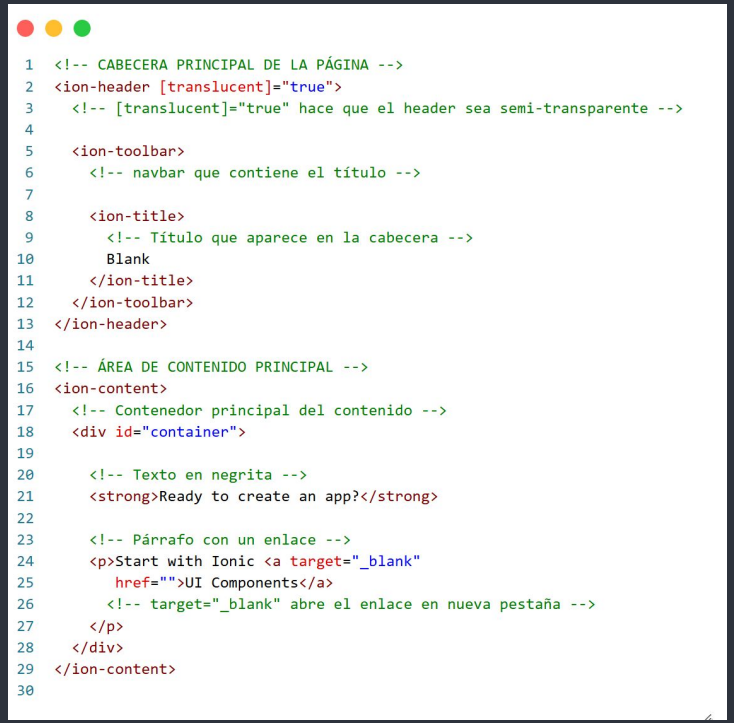
</ Estructura de un proyecto

La estructura es la misma que la de un proyecto de Angular. Lo único diferente en principio es la carpeta home/

- Usa etiqueta <ion-*>, no HTML

1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Estructura de un proyecto



```
<!-- CABECERA PRINCIPAL DE LA PÁGINA -->
<ion-header [translucent]="true">
  <!-- [translucent]="true" hace que el header sea semi-transparente -->

  <ion-toolbar>
    <!-- navbar que contiene el título -->

    <ion-title>
      <!-- Título que aparece en la cabecera -->
      Blank
    </ion-title>
  </ion-toolbar>
</ion-header>

<!-- ÁREA DE CONTENIDO PRINCIPAL -->
<ion-content>
  <!-- Contenedor principal del contenido -->
  <div id="container">

    <!-- Texto en negrita -->
    <strong>Ready to create an app?</strong>

    <!-- Párrafo con un enlace -->
    <p>Start with Ionic <a target="_blank"
       href="">UI Components</a>
      <!-- target="_blank" abre el enlace en nueva pestaña -->
    </p>
  </div>
</ion-content>

```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Creación de un proyecto

Paso 1.

- Abrir vs code
- Open folder (new)
- Abrir terminal

Paso 2.

- Ionic start
- Pick a framework (Angular)
- Nombre al proyecto

Paso 3.

- Starter template -> blank
- Standalone
- Proyecto instalado

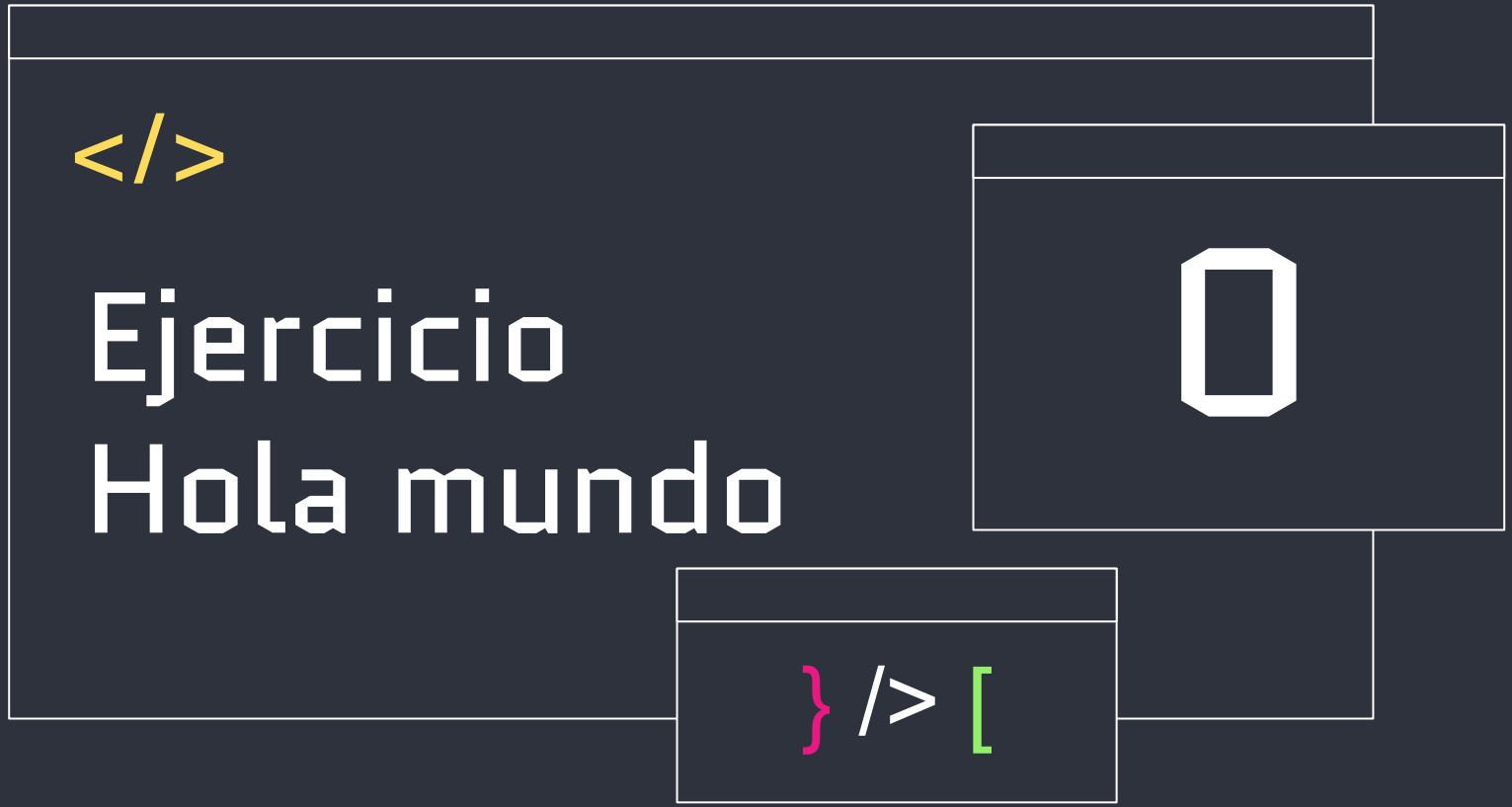
Paso 4.

- cd -> ruta del proyecto
- Npm install por si faltan dependencias

Paso 5.

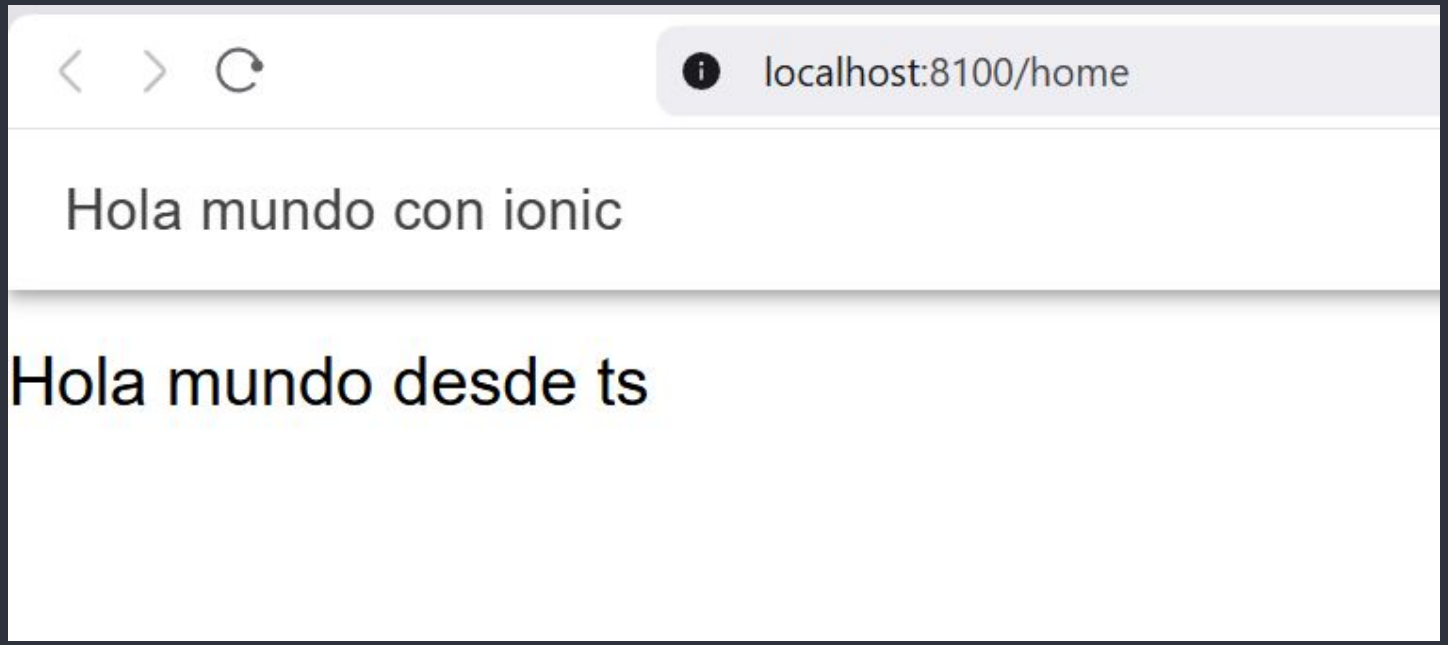
- Reset vs Code
- Ionic serve
- Abrir proyecto en la web

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1

</ Ejercicio hola mundo



1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1

</ Ejercicio hola mundo

```
export class HomePage {
  constructor() {}
  mensaje = "Hola mundo desde ts";
}
```

{{ }} : Angular detecta esa variable y la inserta en la lista

```
<ion-content>
<h2> {{mensaje}}</h2>
</ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Componentes

- Parte reutilizable
- Todo está hecho con componentes (propios de ionic (<ion-button>) o personalizados)
- En esta web de ionic hay componentes útiles para usar. Simplemente habría que generar el componente y copiar el código
- ¿Hace falta crear un componente para todo?

https://ionicframework.com/docs/components

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

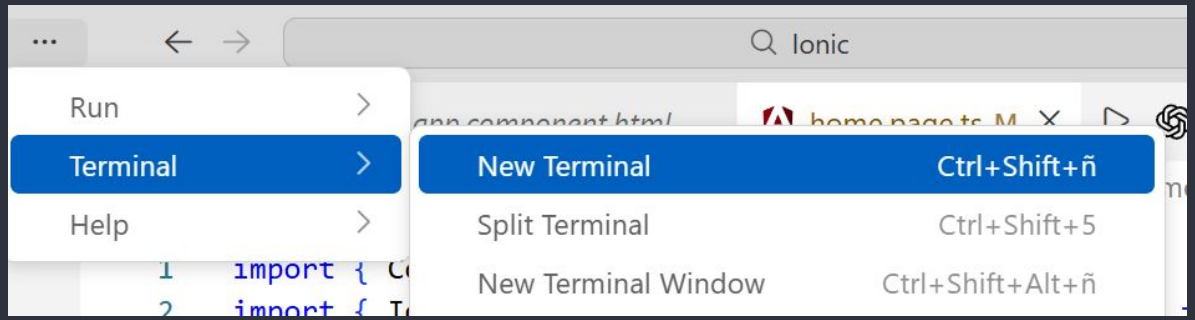
</>

Crear componente

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Crear componente

1.



2.

```
\Ionic> cd .\estructura-basica\
\Ionic\estructura-basica> ionic generate component checkbox
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Crear componente

```html
<ion-content>
  <ion-label> Activar checkbox?</ion-label>
  <ion-checkbox>[(ngModel)]="checked" </ion-checkbox>
  <p> Valor: {{checked}}</p>

</ion-content>

```

```typescript
@Component({
  selector: 'app-checkbox',
  templateUrl: './checkbox.component.html',
  imports: [IonicModule,FormsModule] ,
  styleUrls: ['./checkbox.component.scss'],
})
export class CheckboxComponent  implements OnInit {

  constructor() { }

  ngOnInit() {}

  checked =false;

}
```

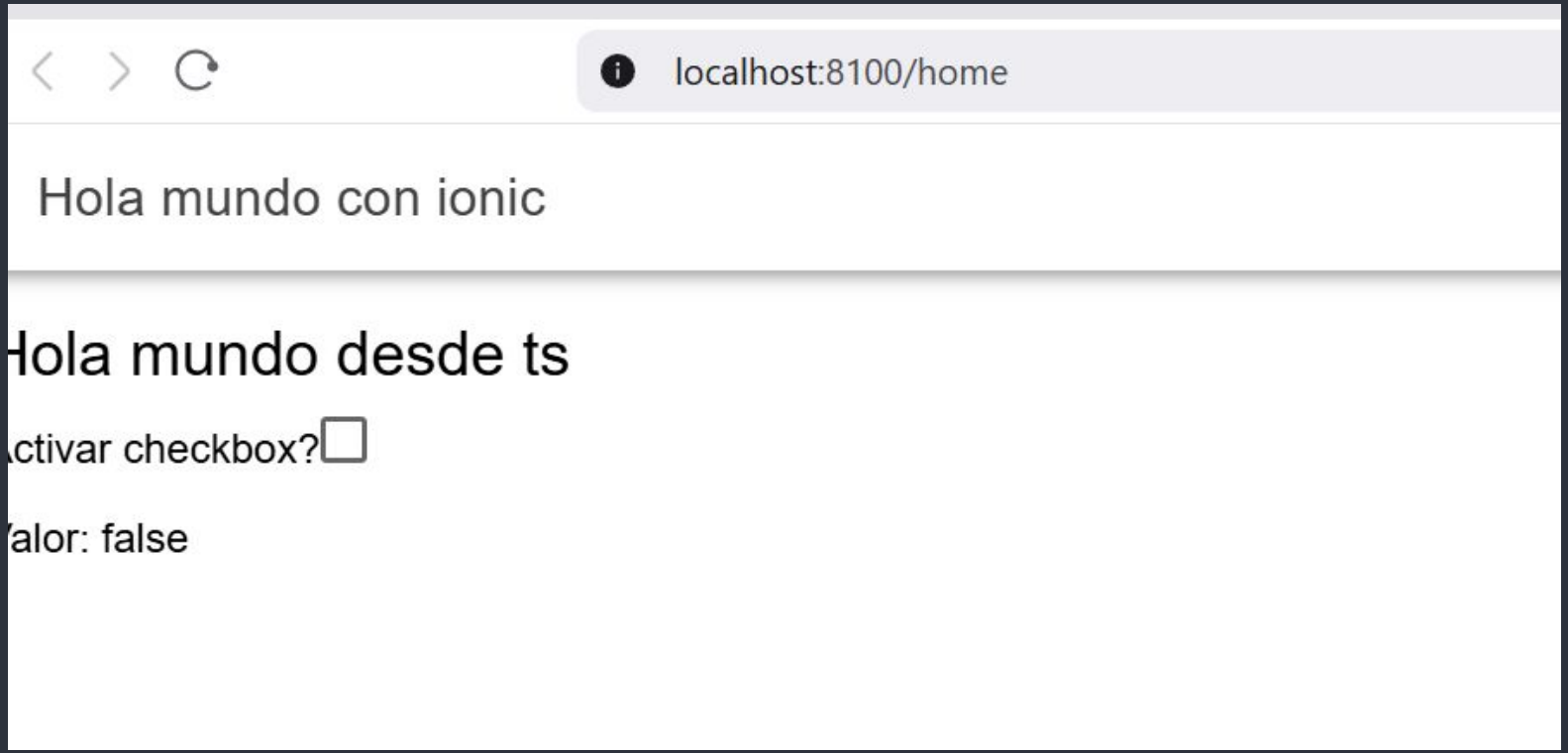
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Insertar componente

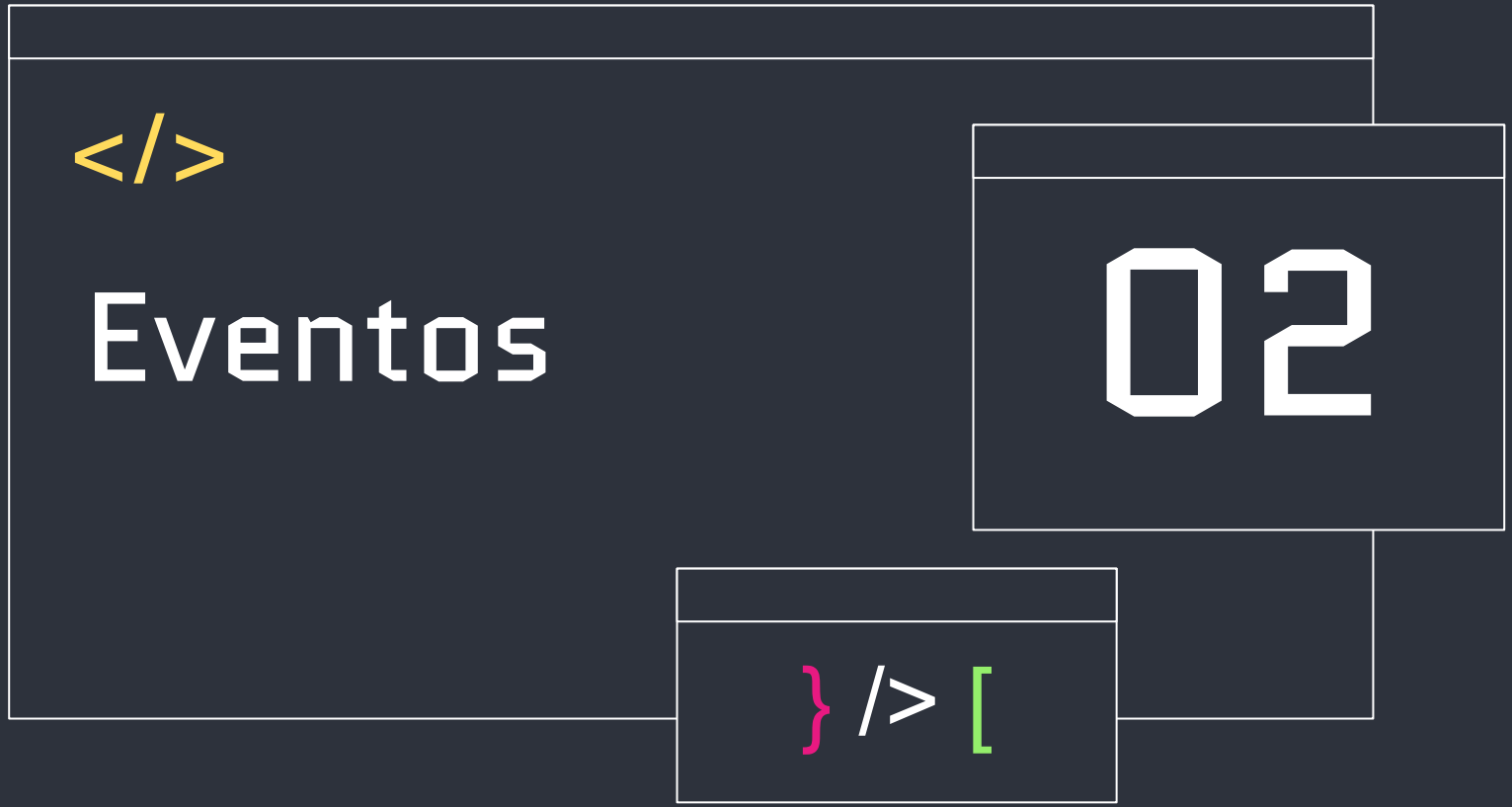
```
@Component({
  selector: 'app-home',
  lateUrl: 'home.page.html',
  eUrls: ['home.page.scss'],
  rts: [IonHeader, IonToolbar, IonTitle, IonContent,CheckboxComponent],

 class HomePage {
 ructor() {}
 aje = "Hola mundo desde ts";
```

```
Go to component | You, 1 second ago | 1 author (You)
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Hola mundo con ionic
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
<h2> {{mensaje}}</h2>

(component) CheckboxComponent
<app-checkbox></app-checkbox>

</ion-content>
```

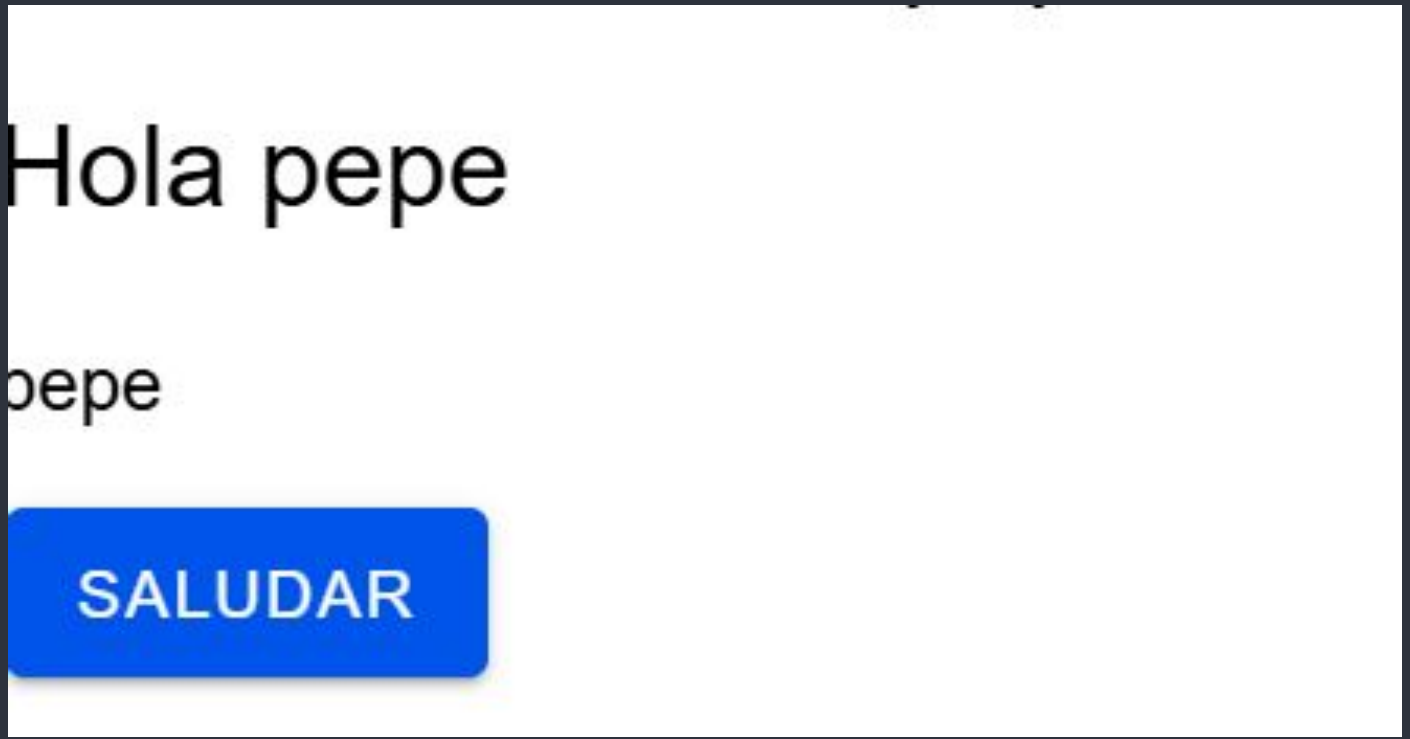
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1



1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1

</ HTML

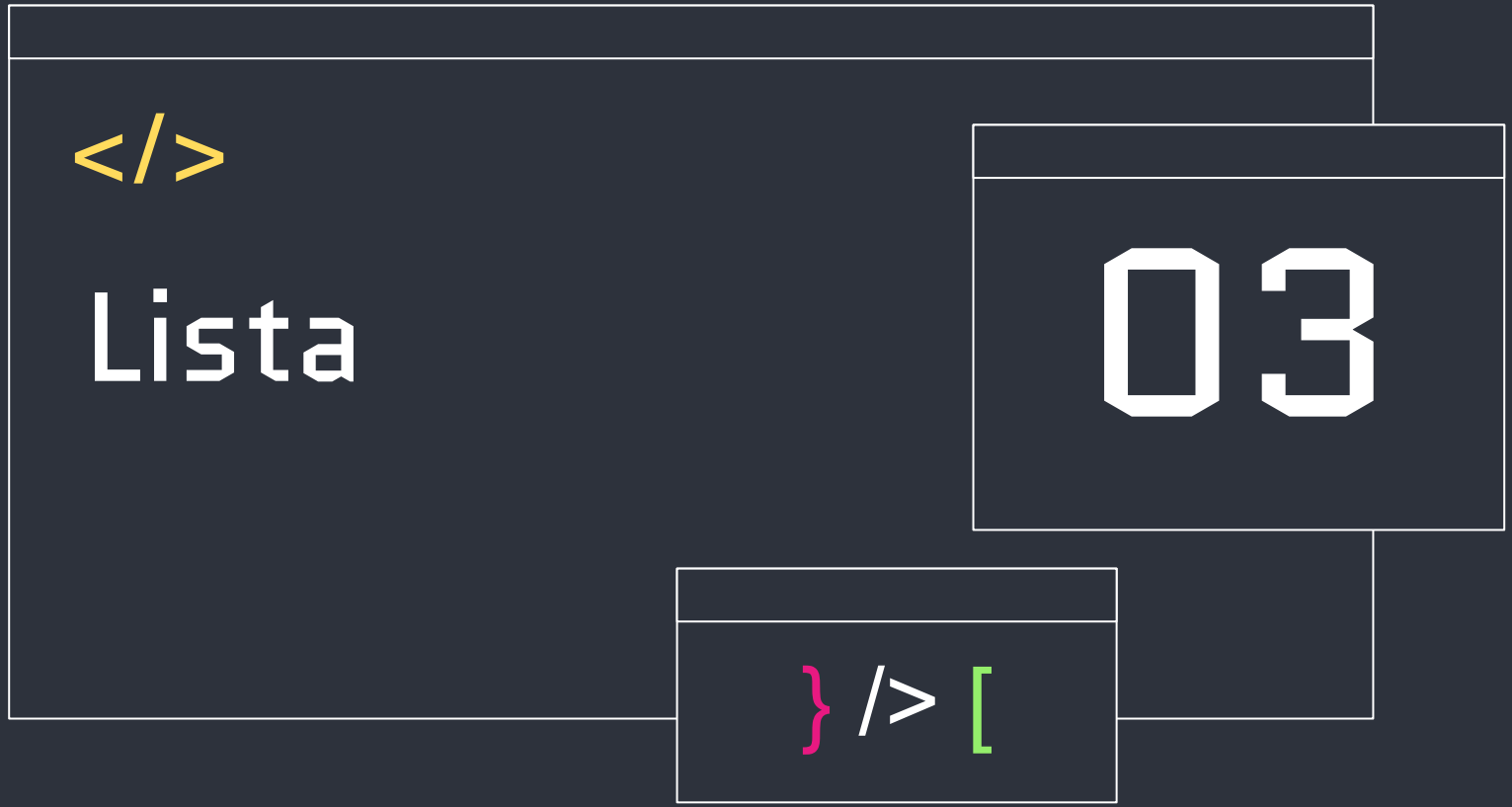
```
1  <ion-content>
2    <h2> {{mensaje}}</h2>
3    <ion-input placeholder="escbie tu nombre" [(ngModel)]="nombre">
4    </ion-input>
5    <ion-button (click)="saludar()">Saludar </ion-button>
6
7
8    <app-checkbox></app-checkbox>
9  </ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

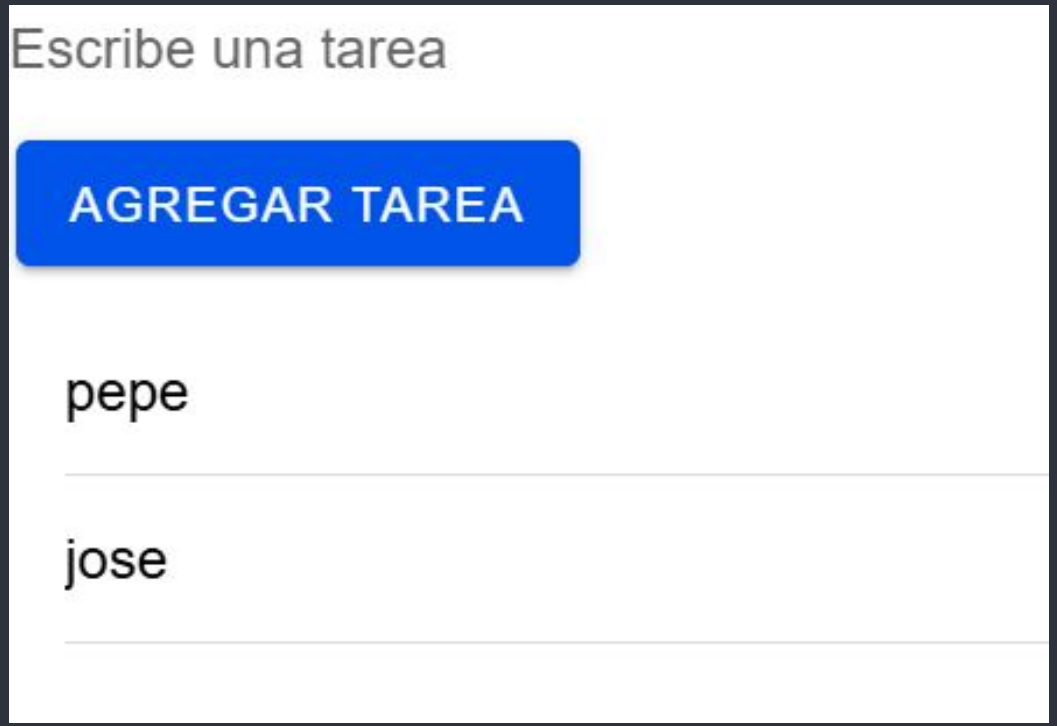
</ TS

```ts
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  imports: [
     CheckboxComponent,
     IonicModule, FormsModule],
})
export class HomePage {
  constructor() {}
  mensaje = "Hola mundo desde ts";
  saludar(){
    this.mensaje = "Hola "+this.nombre;
  }
  nombre ="";
}
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

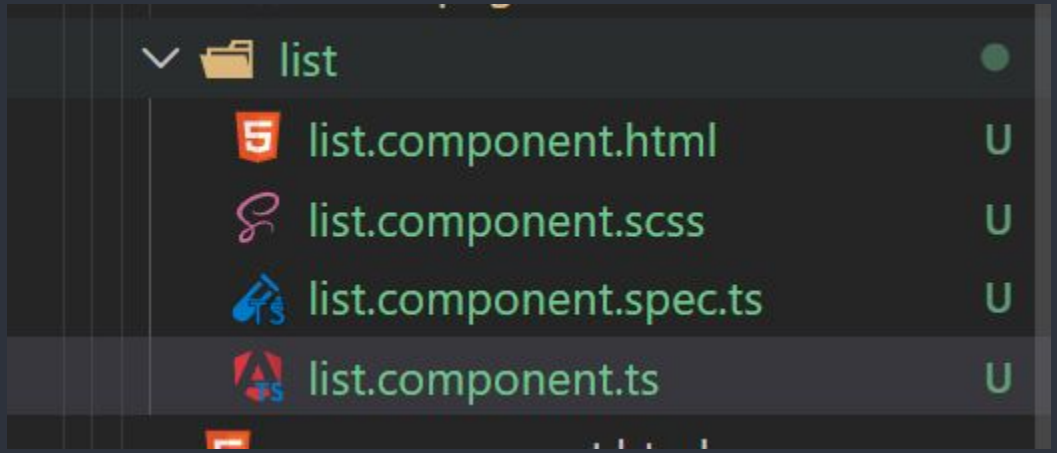


1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

</ HTML [list]

```
<ion-content>

  <ion-input placeholder="Escribe una tarea"[(ngModel)]="tarea" ></ion-ir
  <ion-button  (click)="agregarTarea()">Agregar tarea</ion-button>


   <ion-list>
    @for (tarea of tareas; track $index) {
      <ion-item>{{ tarea }}</ion-item>
}
   </ion-list>



</ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ ts [list]

```ts
@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',

  styleUrls: ['./list.component.scss'],
  imports: [IonContent, IonInput, IonButton,
    FormsModule,CommonModule,IonList,IonItem],
})
export class ListComponent  implements OnInit {

  constructor() { }

  ngOnInit() {}

  tareas :string[]= [];
  tarea ="";
  agregarTarea(){
    this.tareas.push(this.tarea);
    this.tarea ="";

  }

```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ HTML (home)

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Nuevo proyecto de Ionic
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <h2> Hola mundo {{mensaje}}</h2>
  <h2> {{mensaje}}</h2>
  <ion-input placeholder="Escribe tu nombre" [(ngModel)]="nombre"> </
  <ion-button (click)="saludar()"> Saludar </ion-button>




 <app-list></app-list>

  <app-checkbox > </app-checkbox>



</ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ ts[home]

```ts
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  imports: [IonHeader, IonToolbar, IonTitle, IonContent,
    CheckboxComponent, IonInput,FormsModule,IonButton, ListComponent ],
})
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Abrir en Android

04

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

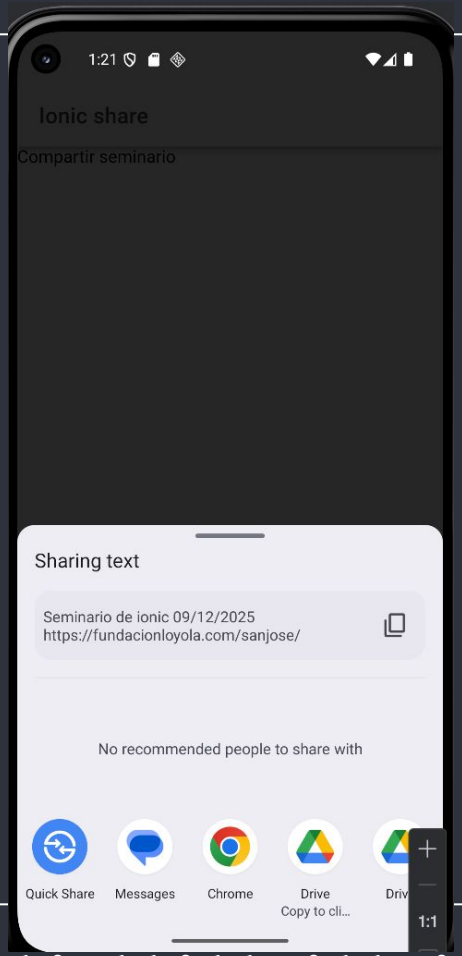
```
npm install @capacitor/share
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

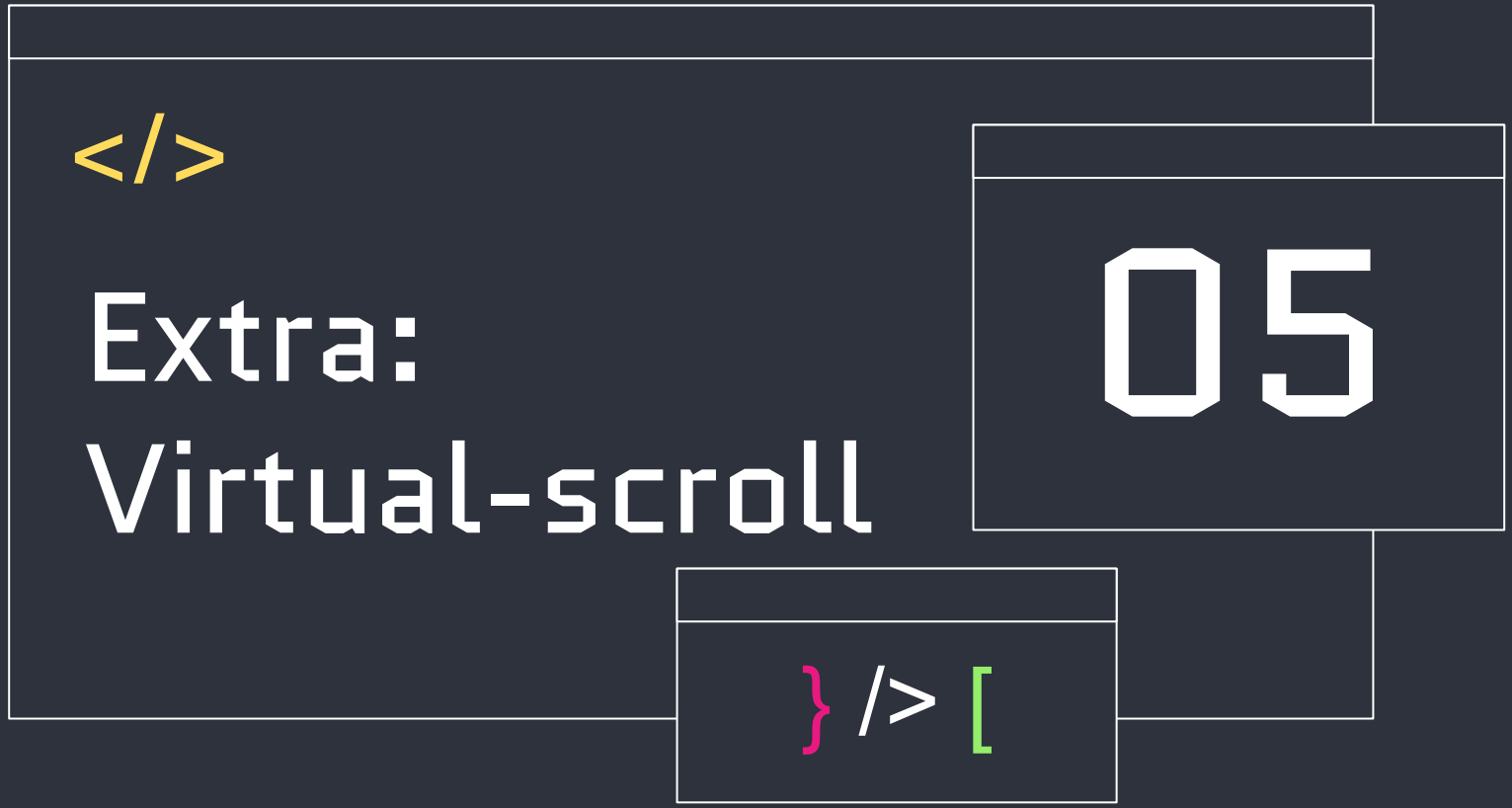
</ Pasos

PRIMERA VEZ	SI SE MODIFICA
- npm install - ionic build - ionic cap add android - ionic cap sync - ionic cap open android	- ionic build - ionic cap sync - ionic cap open android

1011 011 01 1011001 10 11011 011 01 110110 110111 1101



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1


05
Extra:
Virtual-scroll
} /> [

</ Definición

/> ** } /> [

Virtual Scroll es una técnica de optimización de rendimiento diseñada para manejar listas muy extensas de datos.

Su funcionamiento consiste en renderizar en el DOM únicamente los elementos que son visibles en la pantalla (más un pequeño margen de seguridad), en lugar de cargar toda la lista a la vez.

A medida que el usuario se desplaza, el sistema recicla los contenedores existentes: toma los que salen de la pantalla por arriba y los mueve hacia abajo rellenandolos con los nuevos datos, manteniendo el consumo de memoria bajo y la aplicación fluida.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Instalar dependencias

/> **

} /> [

```
PS C:\Users\alvar\Desktop\1Trimestre\Seminarios\SeminarioIonic\proyectoListas> npm install @angular/cdk
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ home.page.ts

```ts
import { Component } from '@angular/core';
import { IonHeader, IonToolbar, IonTitle, IonContent, IonItem, IonLabel } from '@ionic/angular/standalone';
//Añadir import
import { ScrollingModule } from '@angular/cdk/scrolling';


You, 29 seconds ago | 1 author (You)
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  imports: [IonHeader, IonToolbar, IonTitle, IonContent, ScrollingModule, IonItem, IonLabel], //Añadir aquí
})
export class HomePage {
  //Generamos una lista de 100 elementos
  items = Array.from({ length: 100 }).map((_, i) => `Item #${i}`);
}
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ home.page.html

```
1  <ion-header>
2    <ion-toolbar>
3      <ion-title>Lista Virtual</ion-title>
4    </ion-toolbar>
5  </ion-header>
6
7  <ion-content>
8    <ion-list>
9      <cdk-virtual-scroll-viewport itemSize="56" class="ion-content-scroll-host">
10       <ion-item *cdkVirtualFor="let item of items">
11         <ion-label>{{ item }}</ion-label>
12       </ion-item>
13     </cdk-virtual-scroll-viewport>
14   </ion-list>
15 </ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ home.page.scss

El **cdk-virtual-scroll-viewport** necesita tener una altura definida, de lo contrario colapsará a 0px y no verás nada

```scss
cdk-virtual-scroll-viewport {
  height: 100%;
  width: 100%;
}
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Extra: Infinite-Scroll

06

} /> [

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Definición

/> **

} /> [

El scroll infinito (o desplazamiento continuo) es una técnica de diseño web donde el contenido se carga automáticamente mientras el usuario se desplaza hacia abajo, creando una exper
iencia de navegación sin fin, sin botones de "siguiente página", común en redes sociales como Instagram y TikTok para mantener al usuario enganchado mostrando más y más contenido sin interrupción.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ home.page.ts

```ts
import { Component, OnInit } from '@angular/core';
import { InfiniteScrollCustomEvent, IonAvatar, IonContent, IonInfiniteScroll, IonInfiniteScrollContent, IonItem, IonLabel, IonList, IonHeader, IonToolbar, IonTitle } from '@ionic/angular/standalone';

You, 1 minute ago | 1 author (You)
@Component({
  selector: 'app-example',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  imports: [IonAvatar, IonContent, IonInfiniteScroll, IonInfiniteScrollContent, IonItem, IonLabel, IonList, IonHeader, IonToolbar, IonTitle],
})
export class HomePage implements OnInit {
  items: string[] = [];

  ngOnInit() {
    this.generateItems();
  }

  private generateItems() {
    const count = this.items.length + 1;
    for (let i = 0; i < 50; i++) {
      this.items.push(`Item ${count + i}`);
    }
  }

  onIonInfinite(event: InfiniteScrollCustomEvent) {
    this.generateItems();
    setTimeout(() => {
      event.target.complete();
    }, 500);
  }
}
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ home.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Lista Infinita</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    @for (item of items; track item; let index = $index) {
    <ion-item>
      <ion-label>{{ item }}</ion-label>
    </ion-item>
    }
  </ion-list>
  <ion-infinite-scroll (ionInfinite)="onIonInfinite($event)">
    <ion-infinite-scroll-content></ion-infinite-scroll-content>
  </ion-infinite-scroll>
</ion-content>
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Diferencia entre virtual-scroll e infinite-scroll

/> **

} /> [

El Infinite Scroll carga contenido nuevo dinámicamente a medida que el usuario llega al final de la lista (añadiendo elementos al DOM), ideal para feeds como Twitter o Facebook, mientras que el Virtual Scroll solo renderiza los elementos visibles en la pantalla, reciclando los nodos del DOM para un rendimiento superior y evitando la sobrecarga del DOM.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Ejercicio para casa

07

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Lista de tareas avanzada

Objetivo: Replicar la interfaz y lógica de una lista de tareas.

Requisitos Funcionales:

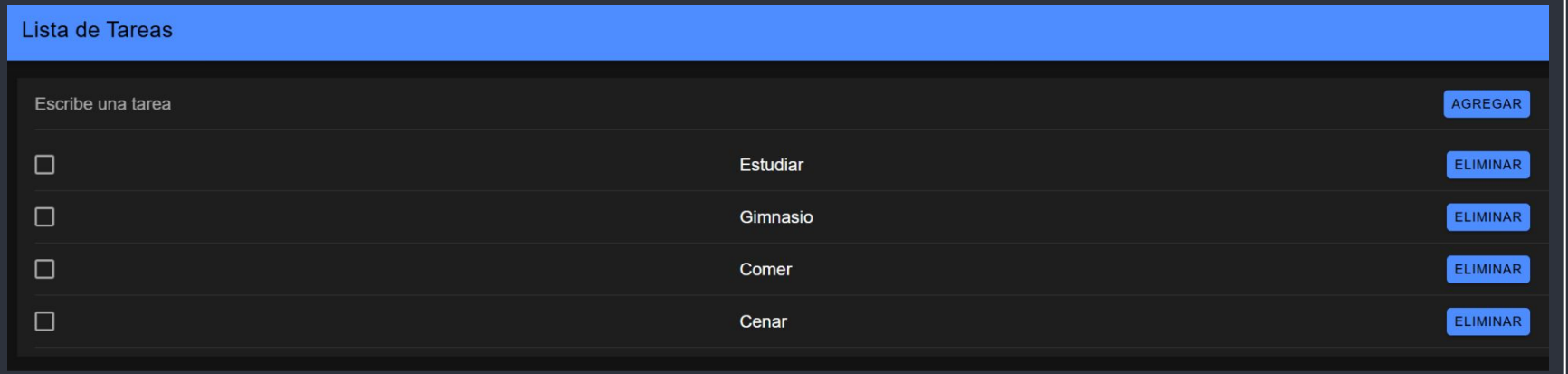
- **Agregar:** Input de texto + Botón "Agregar" (validar que no esté vacío).
- **Listar:** Mostrar las tareas dinámicamente debajo del formulario.
- **Acciones por ítem:**
  - Checkbox: Marcar tarea.
  - Botón Eliminar: Borrar la tarea de la lista.
- Abrir en Android studio (adjuntando captura).

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Lista de tareas avanzada

/> **

} /> [



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1