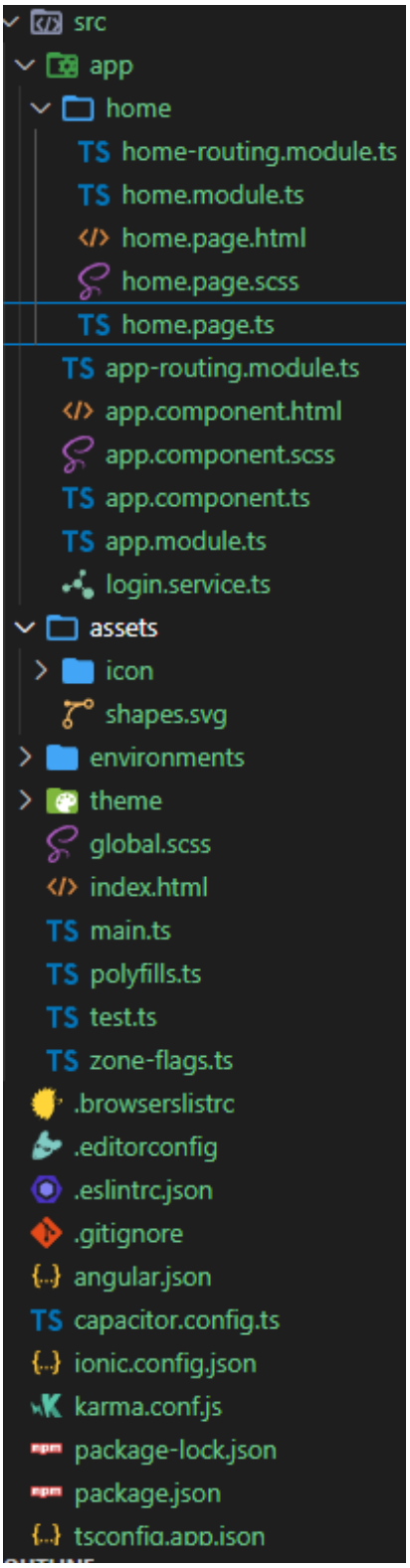
Guía resumen Ionic Angular

Esta guía contiene un resumen de los contenidos hasta ahora con ejemplos.

Ultima actualización: Septiembre 4

Estructura de un proyecto:


src
app
home
home-routing.module.ts
home.module.ts
home.page.html
home.page.scss
home.page.ts
app-routing.module.ts
app.component.html
app.component.scss
app.component.ts
app.module.ts
login.service.ts
assets
icon
shapes.svg
environments
theme
global.scss
index.html
main.ts
polyfills.ts
test.ts
zone-flags.ts
.browserslistrc
.editorconfig
.eslintrc.json
.gitignore
angular.json
capacitor.config.ts
ionic.config.json
karma.conf.js
package-lock.json
package.json
tsconfig.app.json

Carpeta Src:

Contiene código fuente de nuestro proyecto, aquí esta definido nuestro código general.

home.page.html

Vista plantilla de la pagina 'Home'

home.page.scss

Estilos aislados solos a la vista 'Home'

home.page.ts

Logica de la vista 'Home'

app.component.html

Plantilla de vista de la aplicación global, sobre esta vista se dibujan otras vistas usando el Angular Router

app-routing.module.ts

Aquí están definidas las rutas de la app y a que pagina/vista van dirigidas

app.component.ts

Lógica de la aplicación global

app.module.ts

Módulos globales de la aplicación

login.service.ts

Ejemplo de servicio angular

assets

Carpeta para guardar estáticos como imágenes o videos

theme

Definiciones de colores y variables de estilos de la app

global.scss

Estilos globales de la app

capacitor.config.ts

Configuración de nuestra app nativa

package.json

Definiciones de las dependencias del proyecto

Resumen de archivos comunes:

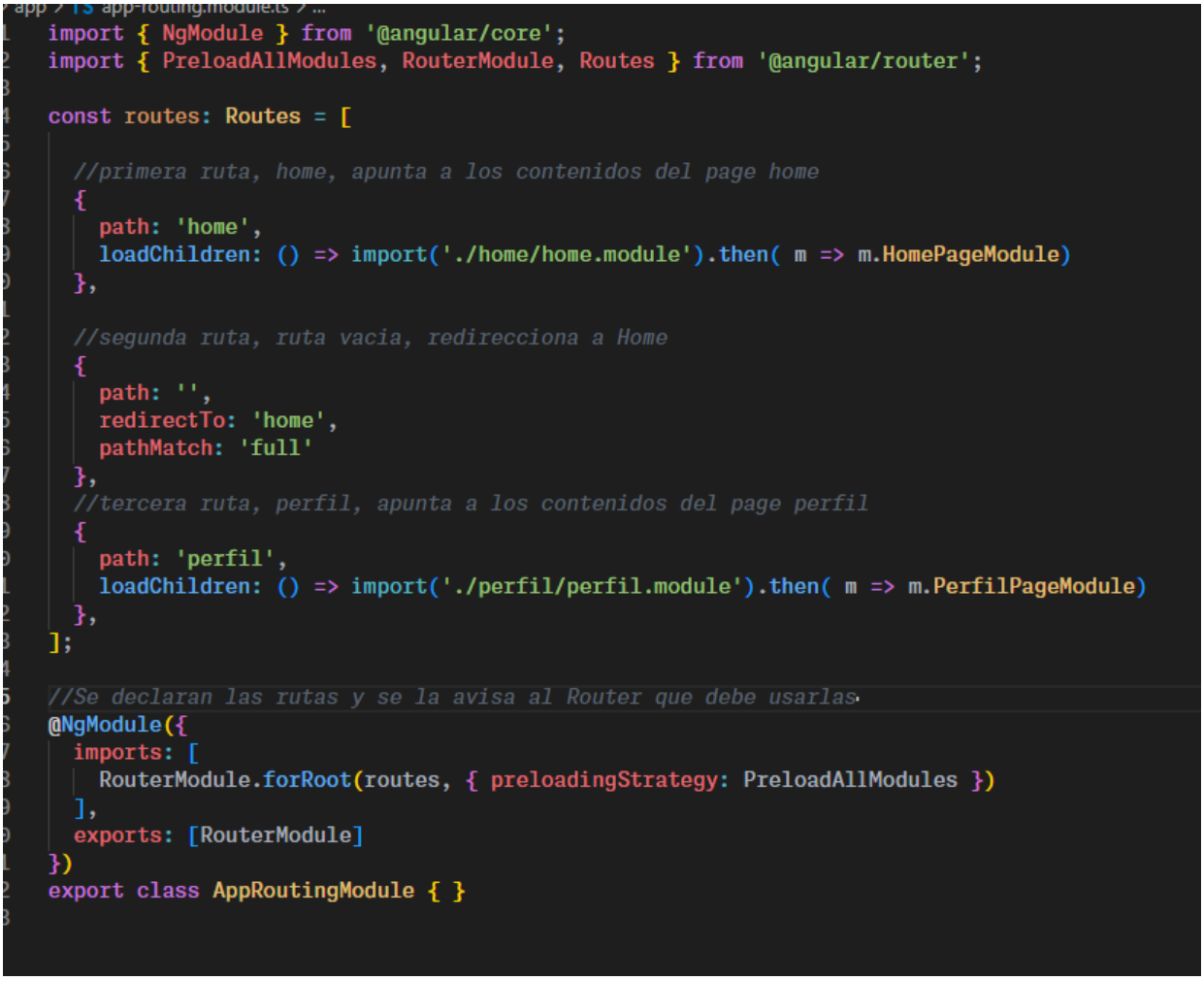
[nombre].page.ts:

```typescript
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage implements OnInit,OnDestroy,ViewDidEnter,ViewDidLeave {

  //Declaración de propiedades de esta pagina
  titulo:string = 'Mi pagina'
  ocultarImagen:boolean = false;
  frutas:string[] = ['Manzana','Pera','Frutilla']

  constructor() {
    //El codigo aqui se ejecuta primero que nada, justo antes de se inicie esta pagina
  }

  ngOnInit(){
    //El codigo aqui se ejecuta cuando el componente se crea
  }

  ngOnDestroy(): void {
      //El codigo aqui se ejecuta cuando el componente se destruye
  }

  ionViewDidEnter(): void {
      //El codigo aqui se ejecuta cuando la animación de entrar a esta vista ya termino
  }

  ionViewDidLeave(): void {
      //El codigo aqui se ejecuta cuando la animación de salir de esta vista ya termino
  }

  // un evento que podemos llamar desde el html
  clickOcultar(){
    //cambiamos esta propiedad y nuestra pagina reaccionara si es que se usa en el html
    this.ocultarImagen = true;
  }

}
```

[nombre].page.html

```
Go to component
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      {{titulo}}
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">

  <ion-button expand="block" color="primary" (click)="clickOcultar()">Ocultar imagen</ion-button>

  <!--Estructura if que depende de la propiedad ocultarImagen, -->
  <!--cuando llamemos el evento con el boton, este if cambiara de condición -->
  <!--y se eliminar el contenido del html-->
  @if(!ocultarImagen){
    <img src="/assets/imagen.png" alt="">
  }

  <ion-list>

    <!--Estructura for de Angular, iteramos sobre frutas y guardamos la iteración en i -->
    @for (i of frutas; track $index) {
      <ion-item>
        <ion-item>
          <ion-label>
            {{i}}
          </ion-label>
        </ion-item>
      </ion-item>
    }

  </ion-list>

</ion-content>
```

app.routing-module.ts



```ts
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [

  //primera ruta, home, apunta a los contenidos del page home
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
  },

  //segunda ruta, ruta vacia, redirecciona a Home
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  //tercera ruta, perfil, apunta a los contenidos del page perfil
  {
    path: 'perfil',
    loadChildren: () => import('./perfil/perfil.module').then( m => m.PerfilPageModule)
  },
];

//Se declaran las rutas y se la avisa al Router que debe usarlas
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Componentes Ionic:

https://ionicframework.com/docs/components

Estructuras de control Angular para el HTML:

Sentencia condicional if

```
<!--Forma actual-->

@if(condicion){
```

```
  <div>
  <!-- Este div se mostrara si la condición es verdadera-->
  </div>
  }
<!--Legacy usando directivas-->
<div *ngIf="condicion">
  <!-- Este div se mostrara si la condición es verdadera-->
</div>
```

Ciclo for

```
<!--Forma actual-->
@for (i of iterable; track $index) {
  <p>
    <!-- Lo dentro del for se ejecutara n veces por el tamaño
del iterable, ademas la variable i contendra la iteración actual
-->
  </p>
}
<!--Legacy usando directivas-->
<p *ngFor="let i of iterable">
    <!-- Lo dentro del for se ejecutara n veces por el tamaño
del iterable, ademas la variable i contendra la iteración actual
-->
</p>
```

Interpolación

Nos permite colocar los valores de propiedades que manejemos dentro del *.page.ts directamente en el html, en el caso que el valor no sea string, angular implícitamente llamara el método .toString() sobre el objeto

```
<p>Mi nombre es {{nombre}}</p>

<!-- En este caso cambiara {{nombre}} por el valor que tenga la propiedad nombre dentro del ts  -->
```

() Enlace de eventos(salida o de HTML a Codigo ts)

Nos permite enlazar eventos html directamente a código dentro de nuestro *.page.ts

```
<button (click)="metodo1()"></button>

<!-- El evento click llamara al metodo metodo1() -->


<input (keyup)="tecleo()" name="nombre">

<!-- El evento tecla arriba, osea cuando se suelta una tecla llamara al metodo tecleo() del ts -->


<button (click)="cantidad-=1">Quitar uno</click>

<!-- Podemos llamar codigo sin pasar por un metodo igualmente -->
```

[] Enlace de atributos (entrada o de Codigo ts a HTML)

```
<img [src]="rutaUrl" alt="imagen">

<!-- Enlaza el atributo src de la imagen a la propiedad rutaUrl del codigo,

gracias al detector de cambios, si cambiamos la propiedad en el ts, cambiara en el html -->

<a [href]="urlPagina">Link que lleva a una url que esta en el ts</a>
```

[(ngModel)] Two way binding, enlace en ambos sentidos

ngModel es una directiva que se usa para crear un enlace bidireccional entre los datos en el componente y los elementos del formulario en la vista. Esto significa que cualquier cambio en el valor del campo de entrada en la vista se refleja en el modelo en el componente, y viceversa

ngModel es una propiedad de las etiquetas que tienen valor, como input, select, checkbox, etc

```
<ion-input label="Nombre" [(ngModel)]="nombre"></ion-input>

<!-- La propiedad nombre esta enlazada a este input, si el
codigo la cambia, sera cambiada en la vista

y si el usuario la cambia en la vista mediante su interacción,
el valor de nombre en el codigo tambien cambiara -->

<ion-checkbox [(ngModel)]="isChecked"></ion-checkbox>



<ion-select [(ngModel)]="opcionSeleccionada">

  <ion-select-option *ngFor="let o of opciones" [value]="o">{{
o}}</ion-select-option>

</ion-select>
```

Servicios Ionic:

Los servicios en Ionic se usan mediante la inyección de dependencias

```
nombreLocalDelServicio = inject(ClaseServicio); // esto debe ser
definido en las propiedades de la clase, arriba del constructor
```

AlertController

Se usa para generar mensajes en la app con apariencia y animación nativa

```
const alerta = await this.alertSrv.create({

   header:'Aviso',
```

```
   message:`Acción realizada de forma correcta`,
   buttons:[
    {
     text:'eliminar',
     role:'ok'
    }
   ]
  });
  await alerta.present();
  const resultado = await alerta.onDidDismiss();
// podemos incluir mas de un boton
// resultado.role contiene el role del boton que se cliqueo
```

Loader

El loader muestra un spinner que bloque la interación del usuario con la app mientras exista

```
const loader = await this.loaderSrv.create({
   message:'Cargando...',
   duration:999999
  });
  await loader.present();
  // aqui realizariamos una carga que toma tiempo
  await loader.dismiss();
```

Otros servicios útiles:

```
//Servicio para navegación
```

```
nav = inject(NavController)

//Servicio para información de la ruta actual

activeRoute = inject(ActivatedRoute)

//Modals

modals = inject(ModalController)

//toasts

toasts = inject(ToastController)
```

Pipes

Los pipes son extensiones para las plantillas HTML que nos permiten formatear o convertir los datos a mostrar a nuevas formas mas útiles.

```
<!-- Pipe de fecha, la propiedad hoy tiene la fecha actual como
objeto js, esto mostrara una fecha humanmente leible
https://v17.angular.io/api/common/DatePipe -->

<p>Hoy es {{hoy| date}}</p>

<!-- moneda, convertira algo como 1.425 en $1.425
https://v17.angular.io/api/common/CurrencyPipe-->

<p>Tu dinero es : {{dinero | currency }}</p>
```