

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from the bar, containing the date.

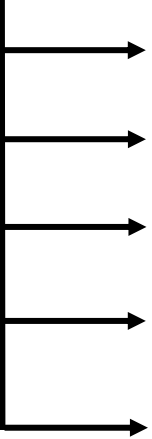
31-5-2023

DOCUMENTACION DEL PROYECTO

Several thin, dark blue curved lines sweep upwards from the bottom left corner of the page.

Sergio García Gordo
U-TAD

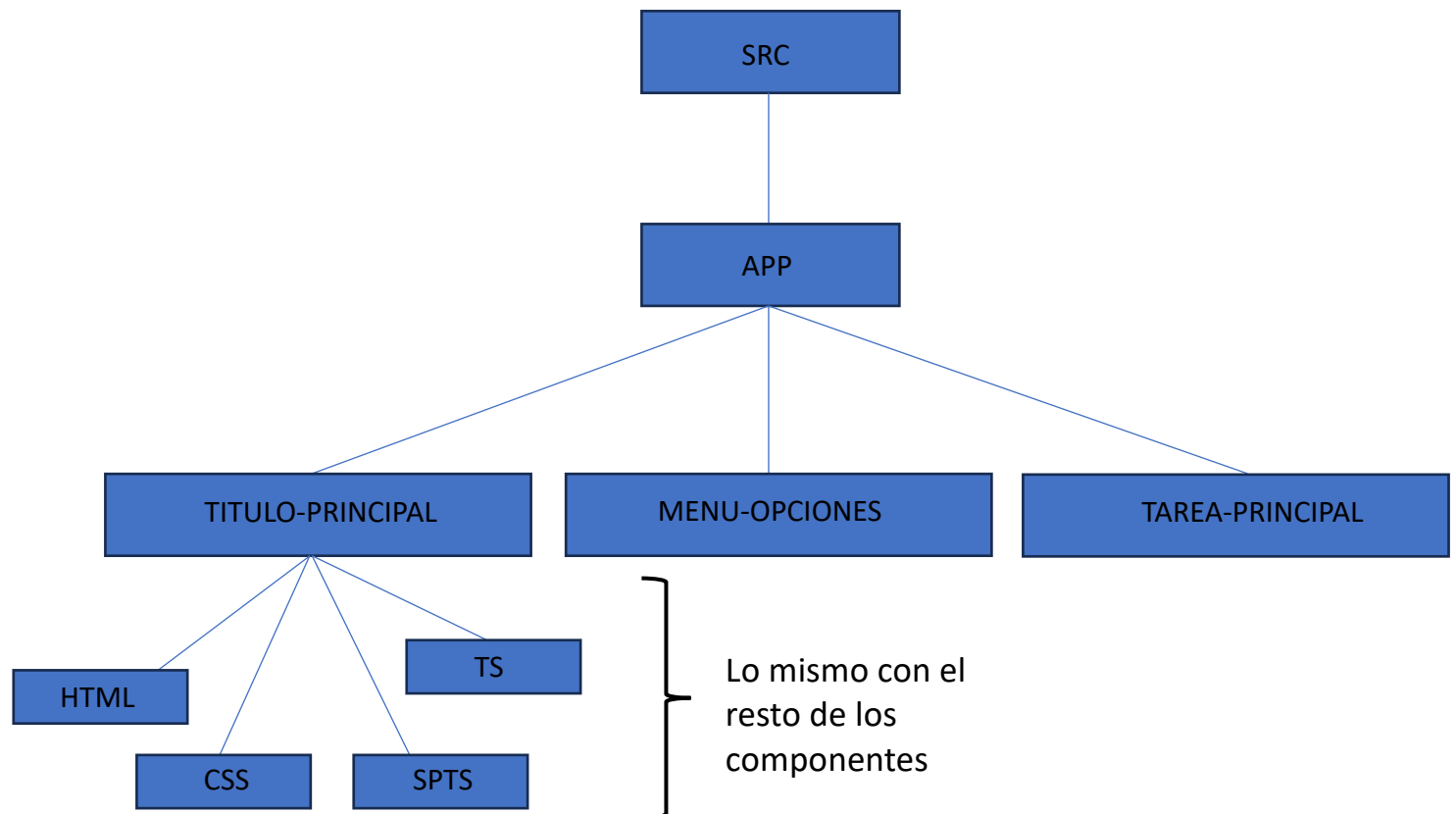
INDICE

- 
1. ESTRUCTURA DE LOS COMPONENTES
 2. ESTRUCTURA DE LAS PLANTILLAS
 3. ESTRUCTURA DEL CODIGO + FUNCIONES
 4. PROBLEMAS ENCONTRADOS + SOLUCIONES
 5. POSIBLES MEJORAS

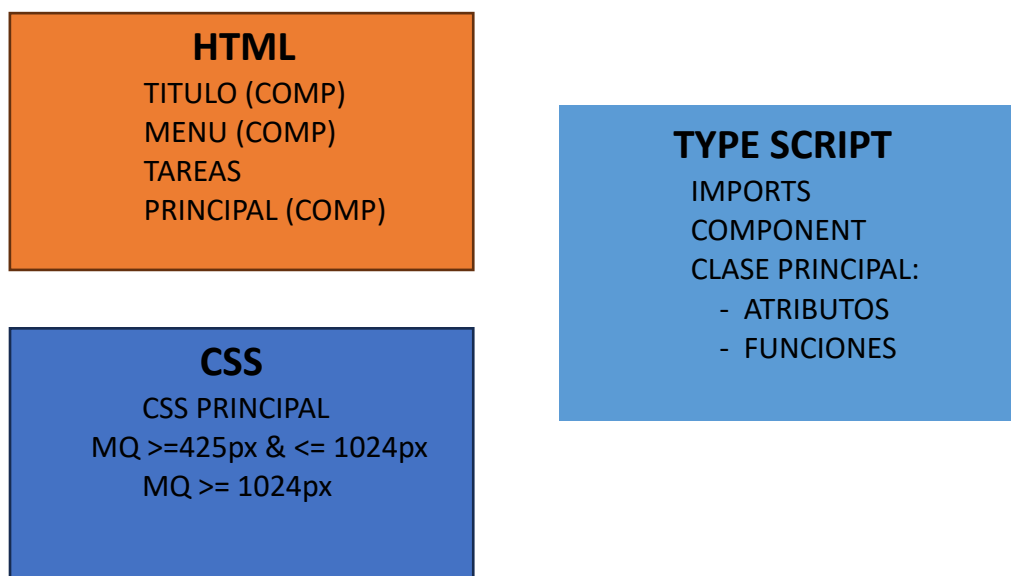
1. ESTRUCTURA DE LOS COMPONENTES

En mi aplicación, realmente, se utilizan pocos componentes, ya que cómo Angular es relativamente nuevo para nosotros, tampoco he querido hacer muchos, debido a que el tiempo que se tardaba en informarse sobre cómo conectar varios componentes entre ellos, podría haberme generado un problema de productividad, por lo que he optado por crear menos componentes.

Este es el esquema:



2. ESTRUCTURA DE LAS PLANTILLAS



3. ESTRUCTURA DEL CODIGO + FUNCIONES

El código HTML se estructura principalmente en divs, en los que se guardan cada elemento, agrupándolos por funcionalidad (el título en uno, la fecha y la hora en otro, el select en otro diferente etc....).

En cuanto al CSS, tampoco tiene gran complicación, ya que lo único que tiene, son los atributos de cada elemento del código HTML a partir de su id y unos media queries generales.

Si observamos el TypeScript tiene una interfaz con una serie de datos, dos atributos declarados dentro de la clase y el resto de los métodos que dotan de funcionalidad (no total) al programa.

```
> tarea-principal > > tarea-principal.component.html > > div#misTareas > > div#descripcion > > div#tarea > > div#idInput > > textarea#input
<div id="tituloDiv">
  
  <p id="tareas">MIS TAREAS</p>
</div>
<div id="divBotones">
  <button id="botonCrear" (click)="ponTarea()">AÑADIR</button>
  <button id="botonEliminar" (click)="eliminarTareas()">ELIMINAR</button>
</div>

<div id="descripcion">
  <p id="vaya" *ngIf="vayaDisp">iVaya!, está todo muy tranquilo por aquí...</p> The attribute name of [ *ngIf ] must be in lowercase.
  <div id="tarea" *ngFor="let tarea of tareas; index as i" > The attribute name of [ *ngFor ] must be in lowercase.
    <div id="titulo">
      <input type="text" id="tituloCab" [(ngModel)]="tarea.titulo"> The attribute name of [ [(ngModel)] ] must be in lowercase.
      <input type="checkbox" id="check" [(ngModel)]="tarea.eliminada"> The attribute name of [ [(ngModel)] ] must be in lowercase.
    </div>

    <div id="tareaCabecera">
      <div id="tiempo">
        <input type="date" id="fecha" [(ngModel)]="tarea.fecha"> The attribute name of [ [(ngModel)] ] must be in lowercase.
        <input type="time" id="hora" [(ngModel)]="tarea.hora"> The attribute name of [ [(ngModel)] ] must be in lowercase.
      </div>
    </div>

    <div id="selec">
      <select id="seleccion" [(ngModel)]="tarea.estado"> The attribute name of [ [(ngModel)] ] must be in lowercase.
        <option value="porHacer">POR HACER</option>
        <option value="enProceso">EN PROCESO</option>
        <option value="terminado">TERMINADO</option>
      </select>
    </div>

    <div id="idInput">
      <!--Con Rows = 5 le decimos el numero de lineas que queremos que tenga el input (para que haya saltos de)-->
      <textarea placeholder="Info Extra" id="input" rows="5" [(ngModel)]="tarea.infoExtra"></textarea> The attribute name of [ [(ngModel)] ] must be in lowercase.
      <br>
      <button type="button" id="botonGuardar" (click)="guardarTarea(tarea)">GUARDAR</button>
    </div>
  </div>
```

```
tarea-principal.component.css — src
tarea-principal.component.html  TS tarea-principal.component.ts  # tarea-principal.component.css x
app > tarea-principal > # tarea-principal.component.css > {} @media screen and (min-width: 1024px)
176     height: 30px;
177     width: 20px;
178     margin-right: 10px;
179     margin-top: 3px;
180     color: blue;
181 }
182
183
184 #botonGuardar {
185     height: 30px;
186     width: 90px;
187     background-color: transparent;
188     color: blue;
189     border: 2px solid white;
190     border-radius: 4px;
191     cursor: pointer;
192     border-color: blue;
193     transition: background-color 0.7s;
194 }
195
196 #botonGuardar:hover {
197     background-color: blue;
198     color: white;
199 }
200
201
202 > @media screen and (min-width: 425px) and (max-width: 1024px) {--
416 }
417 > @media screen and (min-width: 1024px){--
637 }
```

```
tarea-principal > TS tarea-principal.component.ts > TareaPrincipalComponent
@Component({
  selector: 'app-tarea-principal',
  templateUrl: './tarea-principal.component.html',
  styleUrls: ['./tarea-principal.component.css']
})
export class TareaPrincipalComponent {
  tareas: Tarea[] = []; //Array de tareas
  vayaDisp: boolean = true; //display de ¡Vaya!, parece que esta muy tranquilo... para que aparezca o desaparezca cuando se metan tareas

  ponTarea() { //METODO PARA AÑADIR TAREAS
    const nuevaTarea: Tarea = { //Tarea con sus campos
      titulo: '',
      eliminada: false,
      fecha: '',
      hora: '',
      estado: '',
      infoExtra: ''
    };
    this.tareas.push(nuevaTarea); //Enviamos la tarea a array de Tareas
    this.displayVaya(); //Llamamos al metodo que oculta la frase
  }

  eliminarTarea(tarea: Tarea) { //parameto que es la tarea a eliminar
    const indice = this.tareas.findIndex(t => t === tarea); //busca el indice de la tarea y se utiliza una funcion de comparacion que compara
    if (indice !== -1) { //Si es diferente a -1, significa que la tarea se encuentra en la lista
      this.tareas.splice(indice, 1); //Elimina la tarea del array
      this.guardarTareasEnLocalStorage(); //actualiza la localStorage
    }
    this.displayVaya(); //Vuelve a salir la frase
  }

  eliminarTareas() {
    this.tareas = this.tareas.filter(tarea => !tarea.eliminada); //filter crear un array que contiene las tareas que no han sido marcadas con
    //Se evalua si la propiedad "eliminada" de cada tarea es false, si lo es se meten en el array
    this.guardarTareasEnLocalStorage();
    this.displayVaya();
  }

  displayVaya(): void {
    this.vayaDisp = this.tareas.length === 0; //Si this.tareas.length es igual a 0, this.vayaDisp se establecerá en true y se mostrará. De lo
  }

  guardarTareasEnLocalStorage() {
```

4. PROBLEMAS ENCONTRADOS + SOLUCIONES

PROBLEMAS

- ENTENDIMIENTO DEL FUNCIONAMIENTO DE ANGULAR
- CONEXIÓN ENTRE COMPONENTES

SOLUCIONES

- PREGUNTAR EN CLASE ASÍ CÓMO BUSCAR INFORMACIÓN EN INTERNET

5. POSIBLES MEJORAS

El código en mi opinión está bastante bien hecho. Lo único que le falta es filtrar por fecha y demás, así como mejorar la parte del diseño de las tareas. Sin embargo, aunque todavía sigo teniendo tiempo para entregarlo, este trabajo le he hecho para ponerme a prueba e intentar aprender lo que pueda de angular de cara al futuro, además de que tengo la evaluación aprobada.

Es por eso por lo que entrego esta tarea ya, aunque le falten cosas por mejorar.

Dicho esto, sólo me queda darte las gracias por haberme enseñado cómo funciona la vida empresarial, ya que todo esto me va a servir de aprendizaje para el día en el que esté en una de ellas y sin duda, lo agradezco.