Practico 4 @October 24, 2022

En este practico implementaremos lo visto en las ultimas clases:

- Precondición: practico 3
- Contenido:
  - **Primera parte:** creación de un **módulo** que englobe los componentes de homework.
  - **Segunda parte**: creación de un **custom pipe** para el filtrado de homeworks.
  - **Tercera parte:** creación de un **servicio**
  - **Cuarta parte:** creación de un **componente anidado**
  - **Quinta parte**: ruteo

Primera parte

1. **Abrimos el proyecto del practico 3**
2. **Vamos a crear un modulo que va a contener los componentes de homework para esto**
   a. Creamos dentro de app el directorio homeworks y movemos para dentro de este el directorio homeworks-list
   b. Nos movemos a dicho directorio y ejecutamos el comando

```
ng generate module homeworks
```

c. Modificamos el modulo para agregar el componente de listado de homeworks

```
@NgModule({
  declarations: [
    HomeworksListComponent],
  exports:[HomeworksListComponent],
  imports: [
```

```
    CommonModule,
    FormsModule
  ]

})
export class HomeworksModule { }
```

d. Agregamos el modulo homeworks al app module y quitamos el HomeworksListComponent

```
@NgModule({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule,
        FormsModule,
        HomeworksModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Segunda parte

1. **Vamos ahora a crear dentro de este directorio un pipe propio que filtra la lista de homeworks por descripcion, para esto:**
   a. Ejecutamos el comando

```
ng generate pipe homeworks-filter
```

b. modificamos el metodo transform de la clase HomeworksFilterPipe, para que se adapte a lo que necesitamos, en este caso queremos que nos devuelva la lista filtrada de homeworks (list: Array<Homework>) por el string descripción (arg: string).

```
import { Pipe, PipeTransform } from '@angular/core'; //0) importamos
import { Homework } from '../models/Homework';

@Pipe({
  name: 'homeworksFilter'
```

```
})
export class HomeworksFilterPipe implements PipeTransform {
  transform(list: Array<Homework>, arg: string): Array<Homework> {

  }
}
```

c. Agregamos la lógica del Pipe

```
import { Pipe, PipeTransform } from '@angular/core';
import { Homework } from '../models/Homework';

@Pipe({
  name: 'homeworksFilter'
})
export class HomeworksFilterPipe implements PipeTransform {

  transform(list: Array<Homework>, arg: string): Array<Homework> {
    return list.filter(
      x => x.description.toLocaleLowerCase()
        .includes(arg.toLocaleLowerCase())
    );
  }
}
```

2. **Agregamos el filtro al módulo para luego poder usarlo en el componente**

```
@NgModule({
 declarations: [
    HomeworksListComponent,
    HomeworksFilterPipe],
  exports:[HomeworksListComponent],
  imports: [
    CommonModule,
    FormsModule
  ]

})
export class HomeworksModule { }
```

5. **Agregamos el filtrado en el template**

Vamos a `homeworks-list.component.html` y donde usamos `*ngFor`, agregamos el filtrado:

```
<tr *ngFor="let aHomework of homeworks | homeworksFilter : listFilter">
```

6. **Ahora ya podremos filtrar en nuestra lista, corremos la aplicación y probamos**

Tercera parte

1. **Creamos el directorio services dentro de app**
2. **Creamos nuestro servicio**
   a. Vamos a `app/services`
   b. Lanzamos `ng generate service Homeworks`
3. **Modificamos la clase HomeworksService para agregar el metodo getHomeworks()**

```
import { Injectable } from '@angular/core';
import { Homework } from '../models/Homework';
import { Exercise } from '../models/Exercise';

@Injectable()
export class HomeworksService {

  constructor() { }

  getHomeworks():Array<Homework> {
    return [
      new Homework('1', 'Una tarea', 0, new Date(), [
        new Exercise('1', 'Un problema', 1),
        new Exercise('2', 'otro problema', 10)
      ]),
      new Homework('2', 'Otra tarea', 0, new Date(), [])
    ];
  }
}
```

4. **Registramos nuestro servicio a través de un provider**

Para registrar nuestro servicio en nuestro componente, debemos registrar un Provider.

En este caso, lo registraremos en el `HomeworksModule`

```
@NgModule({
  declarations: [
    HomeworksListComponent,
```

```
    HomeworksFilterPipe],
  exports:[HomeworksListComponent],
  imports: [
    CommonModule,
    FormsModule
  ],
  providers:[HomeworksService]

})
export class HomeworksModule { }
```

5. **Inyectamos el servicio en nuestro HomeworksListComponent**

La inyección la logramos a través del constructor de la clase, para ello hacemos en `homeworks-list.component.ts`:

Primero el import:

```
import { HomeworksService } from '../services/homeworks.service';
```

Y luego definimos el constructor que inyecta el servicio a la clase:

```
constructor(private _serviceHomeworks:HomeworksService) {
```

Para inicializar la lista de homeworks usaremos Hooks particularmente, el `OnInit` que se ejecuta luego de inicializar el componente, donde llamaremos el metodo del servicio.

```
ngOnInit(): void {
    this.homeworks = this._serviceHomeworks.getHomeworks();
}
```

6. **Corremos la aplicación y probamos, deberiamos ver los homeworks**

Cuarta parte

1.**Creamos nuesto componente StarComponent con el commando:** `ng generate component Star`

2. **Modificamos el codigo del componente StarComponent**

Dentro de `star.component.ts`, agregamos el siguiente código:

```
import { Component, OnChanges } from '@angular/core';

@Component({
  selector: 'app-star',
  templateUrl: './star.component.html',
  styleUrls: ['./star.component.css']
})
export class StarComponent implements OnChanges {

    rating: number = 4;
    starWidth: number;

    ngOnChanges():void {
        this.starWidth = this.rating * 86/5;
    }
}
```

3. **Creamos el Template para nuestro StarComponent y sus estilos**

Dentro de `star.component.html`, agregamos el siguiente código:

```
<div clas="crop"
    [style.width.px]="starWidht"
    [title]="rating">
    <div style="width:86px">
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
    </div>
</div>
```

A su vez, vamos a `star.component.css`, y agregamos el siguiente código:

```
.crop {
    overflow: hidden;
}

div {
    cursor: pointer;
}
```

4. **Agregamos el StarComponent al HomeworksModule si no se importo automaticamente**

```
@NgModule({
  declarations: [
    HomeworksListComponent,
    HomeworksFilterPipe,
    StarComponent],
  exports:[HomeworksListComponent],
  imports: [
    CommonModule,
    FormsModule,
    HttpClientModule

  ],
  providers:[HomeworksService]

})
export class HomeworksModule { }
```

1. **Usamos el StarComponent dentro de HomeworksListComponent**

En el template de nuestro componente de listado de tareas, es decir dentro de homeworks-list.component.htm, agregamos la celda rating a la tabla

```
<th>Rating</th>
```

Y luego dentro del body de la tabla, es decir, en el tag `tbody`, y dentro del `*ngFor` sobre las rows, agregamos esta última celda:

```
<td>
   <app-star></app-star>
</td>
```

6. **Usando input properties**

Vemos que se muestran 5 estrellas y no 4 como habíamos hardcodeado. Ni que tampoco se modifica el valor en el OnChanges (esto es porque el OnChanges se cambia cuando alguna input property de un componente se refresca). Vamos ahcer esto:

a. Ir a `app/models/homework.ts` y agregar la property rating

b. Ir a `app/homeworks-list/homeworks-list.component.ts` y agregar un valor de rating en las tareas que tengamos creadas

c. Ir al HTML del template y cambiar lo que teníamos antes por lo siguiente

```
<td>
    <app-star [rating]='aHomework.rating'></app-star>
</td>
```

7. **Definimos la Input Property Rating en StarComponent**

Para ello precisamos importar Input:

```
import { Component, OnChanges, Input } from '@angular/core';
```

Y luego agregar el decorador a la property `rating`:

```
@Input() rating: number;
```

8. **Levantamos eventos desde el componente anidado**

```
import { Component, OnChanges, Input, Output, EventEmitter } from '@angular/core';
```

Luego definimos la property del evento en nuestra clase del StarComponent:

```
@Output() ratingClicked: EventEmitter<string> = new EventEmitter<string>();
```

Luego hacemos el binding del evento en el template del evento:

```
<div class="crop"
    [style.width.px]="starWidth"
    title={{starWidth}}
    (click)='onClick()'>
    <div style="width: 86px">
        <span class="glyphicon glyphicon-star"></span>
```

```
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
    </div>
</div>
```

Y ahora volvemos al StarComponent y agregamos la funcion onClick que acabamos de declarar en nuestro template:

```
onClick(): void {
    this.ratingClicked.emit(`El raiting es ${this.rating}!`);
}
```

Finalmente, lo que haremos es agregar es la referencia al evento en el template:

```
<app-star [rating]='aHomework.rating'
    (ratingClicked)='onRatingClicked($event)'>
</app-star>
```

Y dentro del código del HomeworksListComponent definimos el callback que queremos que se ejecute cuando se haga click en nuestras estrellas:

```
onRatingClicked(message:string):void {
    this.pageTitle = 'HomeworksList ' + message;
}
```

Quinta parte

1. **Creamos el menu**

```
ng generate component menu
```

html

```
<div class="panel panel-primary">
  <div class="panel-heading">
    {{pageTitle}}
  </div>
  <td>
    <a [routerLink]="['/homeworks']"> {{"Homeworks list" | uppercase}} </a>
  </td>
</div>
<router-outlet></router-outlet>
```

component

```
pageTitle : string = 'Homework app';
```

en homeworks module agrego la ruta

```
const rootes:Routes = [
  { path: 'homeworks', component: HomeworksListComponent}]

@NgModule({
  declarations: [
    HomeworksListComponent,
    HomeworksFilterPipe,
    StarComponent,
    HomeworkDetailComponent,
    MenuComponent,
    ],
    exports: [HomeworksListComponent,
      RouterModule, MenuComponent],
    imports: [
      CommonModule,
      FormsModule,
      HttpClientModule,
      RouterModule.
      forRoot(rootes, {onSameUrlNavigation:'reload'})
    ],
  providers:[HomeworksService]

}
)
export class HomeworksModule {

}
```

app module html modificamos lo que teniamos, y dejamos el siguiente código

```
<app-menu></app-menu>
```

2. Creamos un HomeworkDetailComponent:

ng generate component homeworkDetail

`homework-detail.component.html`:

```
<!--Tabla de tareas -->
  <div class='table-responsive'>
    <table class='table'>
      <!--Cabezal de la tabla -->
      <thead>
      <tr>
        <th>Id</th>
        <th>Description</th>
        <th>DueDate</th>
        <th>Score</th>
        <th>Rating</th>
        <th>
        </th>
      </tr>
      </thead>
      <!--Cuerpo de la tabla-->
      <tbody>
        <td>{{aHomework?.id}}</td>
        <td>{{aHomework?.description | uppercase}}</td>
        <td>{{aHomework?.dueDate}}</td>
        <td>{{aHomework?.score}}</td>
        <td>
          <div>
                <table>
                  <thead>
                  <tr>
                    <th>Problem</th>
                    <th>Score</th>
                  </tr>
                  </thead>
                  <tbody>
                  <tr *ngFor='let aExercise of aHomework?.exercises'>
                    <td>{{aExercise.problem}}</td>
                    <td>{{aExercise.score}}</td>
                  </tr>
                  </tbody>
                </table>
            </div>
        </td>

      </tbody>
```

```
    </table>
</div>

  <div class='panel-footer'>
    <a class='btn btn-default' (click)="onBack()" style="width:80px">
      <i class='glyphicon glyphicon-chevron-left' ></i> Back
    </a>
  </div>
</div>
```

`homework-detail.component.ts`:

```
import { Component, OnInit } from '@angular/core';
import { Homework } from '../models/Homework';

@Component({
  selector: 'app-homework-detail',
  templateUrl: './homework-detail.component.html',
  styleUrls: ['./homework-detail.component.css']
})
export class HomeworkDetailComponent implements OnInit {
  pageTitle : string = 'Homework Detail';
aHomework : Homework | undefined = new Homework("2", "Otra tarea", 0, new Date(), [],
4);

  constructor() { }

  ngOnInit() {
  }
}
```

Y a su vez agregamos este componente en el HomeworksModule, primero haciendo el import y luego agregando `HomeworkDetailComponent` en el array de declarations:

```
import { HomeworkDetailComponent } from './homework-detail/homework-detail.component';
```

```
declarations: [
  AppComponent,
  HomeworksListComponent,
  HomeworksFilterPipe,
  StarComponent,
  WelcomeComponent,
  HomeworkDetailComponent
],
```

1. Seteamos el path en app.module.ts (AppModule)

En este caso el path sería: *homeworks/id*, indicando que ruta a un componente **HomeworkDetailComponent**. A su vez, le pasamos el parámetro id, con una barra y un dos puntos adelante (/:id). Si quisiéramos más parámetros, repetimos esto.

```
RouterModule.forRoot([{ path: 'homeworks/:id', component: HomeworkDetailComponent }]
```

2. Ruteamos al path

Agregamos en ***HomeworkDetailComponent***

```
@Input() routerLink: string | any[] ="";
```

Luego en el HTML de nuestro ***HomeworkDetailComponent***, ponemos un link (ancla) sobre el nombre, de manera de que cada vez que se haga click sobre el mismo, dicha ruta se resuelva y se le pase el parámetro asociado.

```
  <td><a [routerLink]="['/homeworks', aHomework.id]"> {{aHomework.id | uppercase}} </a
></td>
```

3. Leemos los parámetros de la ruta en el HomeworkDetailComponent

Leemos los parámetros de la ruta, usando el service ActivatedRoute de '@angular/router'.

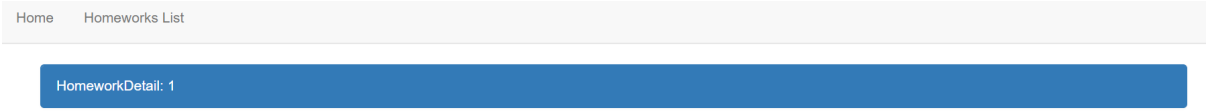
Lo inyectamos en nuestro componente para que use este servicio (el provider ya viene resuelto por el RouterModule que usamos la clase anterior):

```
constructor(private _currentRoute: ActivatedRoute, private serviceHomework:HomeworksSe
rvice) { }
```

1. Agarramos el parámetro de la ruta y lo ponemos en una variable privada, dicha lógica lo haremos en el OnInit (hay que implementar OnInit).

```
ngOnInit() : void {
  // let (es parte de ES2015) y define una variable que vive en este scope
```

```
  // usamos el nombre del parámetro que uamos en la configuración de la ruta y lo obte
nemos
  let id =+ this._currentRoute.snapshot.params['id'];
  // definimos el string con interpolacion
  this.pageTitle +=  `: ${id}`;
this.aHomework= this.serviceHomework.getHomeworks().find(x=>x.id==id.toString())
}
```



Detalle de homeworks

1. **Creamos un HomeworkDetailComponent:**

ng generate component homeworkDetail

`homework-detail.component.html`:

```
<div class="panel panel-primary">
    <div class="panel-heading">
        {{pageTitle}}
    </div>
</div>
```

`homework-detail.component.ts`:

```
import { Component, OnInit } from '@angular/core';
import { Homework } from '../models/Homework';

@Component({
  selector: 'app-homework-detail',
  templateUrl: './homework-detail.component.html',
  styleUrls: ['./homework-detail.component.css']
})
export class HomeworkDetailComponent implements OnInit {
  pageTitle : string = 'Homework Detail';
  homework : Homework;

  constructor() { }

  ngOnInit() {
```

```
  }
}
```

Y a su vez agregamos este componente en el HomeworksModule, primero haciendo el import y luego agregando `HomeworkDetailComponent` en el array de declarations:

2. **Seteamos el path en HomeworksModule**

En este caso el path sería: *homeworks/id*, indicando que ruta a un componente **HomeworkDetailComponent**. A su vez, le pasamos el parámetro id, con una barra y un dos puntos adelante (/:id). Si quisiéramos más parámetros, repetimos esto.

```
RouterModule.forRoot([{ path: 'homeworks/:id', component: HomeworkDetailComponent }]
```

3. **Ruteamos al path**

En el HTML de nuestro ***HomeworkDetailComponent***, ponemos un link (ancla) sobre el nombre, de manera de que cada vez que se haga click sobre el mismo, dicha ruta se resuelva y se le pase el parámetro asociado.

```
  <td><a [routerLink]="['/homeworks', aHomework.id]"> {{aHomework.id | uppercase}} </a
></td>
```

4. **Leemos los parámetros de la ruta n el HomeworkDetailComponent**

Leemos los parámetros de la ruta, usando el service ActivatedRoute de '@angular/router'.

Lo inyectamos en nuestro componente para que use este servicio (el provider ya viene resuelto por el RouterModule que usamos la clase anterior):

```
constructor(private _currentRoute: ActivatedRoute) {  }
```

Agarramos el parámetro de la ruta y lo ponemos en una variable privada, dicha lógica lo haremos en el OnInit (hay que implementar OnInit).

```
ngOnInit() : void {
  let id =+ this._currentRoute.snapshot.params['id'];
  this.pageTitle +=  `: ${id}`;
}
```