

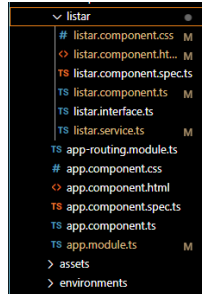
Documento Escrito de lo realizado durante
la prueba de postulación para la vacante
Fron-End

LA Katuna

CHRISTIAN ANTONIO ARIAS GRANIZO

1. Añadir una nueva columna que permita gestionar el estado DONE/INCOMPLETE

Se creo un select dentro del listado que permite gestionar el estado de cada ToDo para este fin se creo un front end en Angular agregando el componente Listar



En el cual se pudo conectar al aplicativo backend, para la gestión del CRUD, mediante la creación de un componente de tipo servicio(modelo)

```
constructor(private http: HttpClient) { }

getObjeto(): Observable<Objeto[]> {
  return this.http.get<Objeto[]>(OBJETO_API);
}
createObjeto(obj: Objeto): Observable<Objeto> {
  return this.http.put<Objeto>(OBJETO_API, obj);
}
deleteObjeto(obj: Objeto): Observable<Objeto> {
  return this.http.post<Objeto>(OBJETO_API, obj);
}
updateObjeto(obj: Objeto): Observable<Objeto> {
  return this.http.post<Objeto>(OBJETO_API, obj);
}
//para el estado
getObjetoEstado(): Observable<Objeto[]> {
  return this.http.get<Objeto[]>(OBJETOESTADO_API);
}
createObjetoEstado(obj: Objeto): Observable<Objeto> {
  return this.http.put<Objeto>(OBJETOESTADO_API, obj);
}
updateObjetoEstado(obj: Objeto): Observable < Objeto > {
  return this.http.post<Objeto>(OBJETOESTADO_API, obj);
}

logWithContext(): void {
  this.logger.error('Logging from ListarService');
}
```

Además, para luego proceder a conectarse mediante el archivo de tipo componente(controlador), para gestionar los elementos a utilizarse y mostrar en la vista (html)

```

11 export class ListarComponent implements OnInit {
12
13   obj: Objeto;
14   objjs: Objeto[];
15
16   constructor(
17     private objService: ListarService,
18     private logger: NGXLogger,
19   ) {
20     logger.partialUpdateConfig({ context: 'ListarComponent' });
21   }
22
23   ngOnInit(): void {
24
25   }
26
27   logWithContext(): void {
28     this.logger.error('Logging from CustomInstanceComponent');
29     this.objService.logWithContext();
30   }
31
32   private listObj() {
33     this.objService.getObjeto().subscribe((data: Objeto[]) => this.objjs = data, error => console.log('error en get Objeto', error));
34   }
35
36   onCreateObjeto(event: Objeto) {
37     this.objService.createObjeto(event).subscribe(
38       data => {
39         this.obj = Object.assign({}, this.obj, event);
40         this.listObj();
41       },
42     );
43   }
44   onUpdateObjeto(event: Objeto) {
45     this.objService.updateObjeto(event).subscribe(
46       data => {
47         this.obj = Object.assign({}, this.obj, event);
48         this.listObj();
49       },
50     );
51   }
52   onDeleteObjeto(event: Objeto) {
53     this.objService.deleteObjeto(event).subscribe(
54       data => {
55         this.obj = Object.assign({}, this.obj, event);
56         this.listObj();
57       },
58     );
59   }
60 }

```

Se agregó un parámetro para evitar que se solicite la inicialización de variables. En el archivo tsconfig.json

```

1  /* To learn more about this file see: https://angular.io/config/tsconf
2  {
3    "compileOnSave": false,
4    "compilerOptions": {
5      "baseUrl": "./",
6      "outDir": "./dist/out-tsc",
7      "forceConsistentCasingInFileNames": true,
8      "strict": true,
9      "strictPropertyInitialization": false,
10     "noImplicitOverride": true,
11     "noPropertyAccessFromIndexSignature": true,
12     "noImplicitReturns": true,

```

ToDoList

| Título | Descripción | Estado | Acciones |
|--------|-------------|--|--------------------------------------|
| | | <div> <div>Done</div> <div>Done</div> <div>Incomplete</div> </div> | <div> <div></div> <div></div> </div> |

Export Data

- Añadir un nuevo path que permita cambiar el estado de 1 elemento de la lista
Se agrego un path nuevo en el backend para gestionar el estado

```

33 | @GET
34 | public Set<ToDoList> listT(title) {
35 |     this.title = title;
36 |     return this.title;
37 | }
38 | @POST
39 | public Set<ToDoList> add(ToDoList element) {
40 |     toDoLists.add(new ToDoList( this.estado : element));
41 |     return toDoList;
42 | }
43 | @POST
44 | public Set<ToDoList> add(ToDoList element) {

```

Y se agrego el elemento en el front-end que permita gestionarlo

```

ToDoList > ToDoList_Front > src > app > listar > TS listar.service.ts > ListarService > createObjetoEst
32 | return this.http.put<Objeto>(OBJETO_API, obj);
33 | }
34 | //para el estado
35 |
36 | createObjetoEstado(obj: Objeto): Observable<Objeto> {
37 |     return this.http.put<Objeto>(OBJETO_API, obj);
38 | }
39 |
40 |
41 |
42 | logWithContext(): void {
43 |     this.logger.error('Logging from ListarService');
44 | }
45 | }

```

Así como también el elemento html en la vista



3. Añadir un nuevo path que permita buscar por título

Se agrego un path en el back-end con el fin de filtrar el contenido por el título ingresado en la barra de búsqueda

```

@GET
public Set<ToDoList> listT(title) {
    if (value.getTitle().equals(element.getTitle())) {
        return element.getTitle()
    }else
        return null
}

```

Así como también se usó el elemento search de la vista para obtener el valor de búsqueda

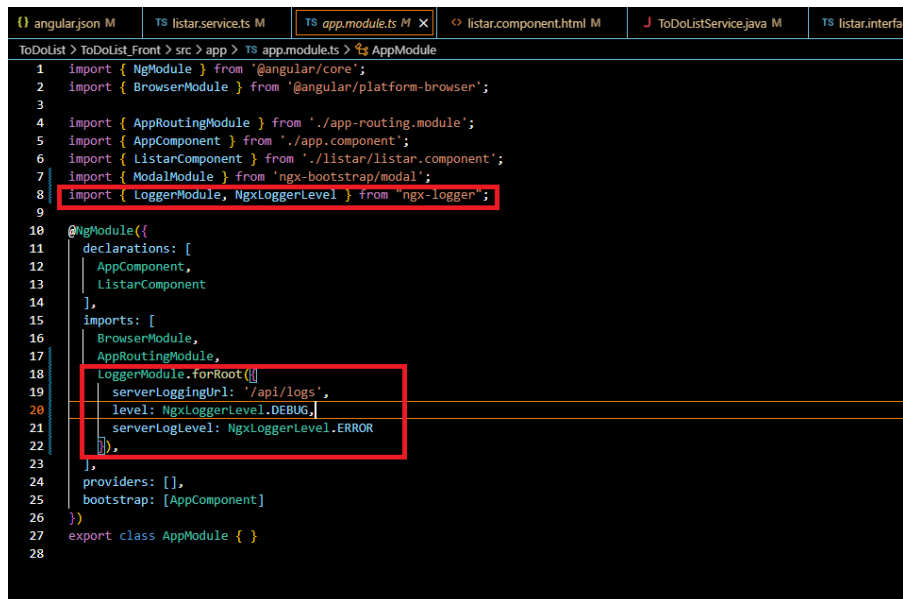


```
angular.json M TS listar.service.ts M X listar.component.html M J ToDoListService.java M TS listar.interface.ts M
ToDoList > ToDoList_Front > src > app > listar > TS listar.service.ts > ListarService > getTitle
11 headers: new HttpHeaders({ 'Content-Type': 'application/json' })
12 }
13
14
15
16 export class ListarService {
17   private logger: NGXLogger;
18
19   constructor(private http: HttpClient) { }
20
21   @ViewChild('search') srch: ElementRef;
22
23   getTitle(obj: Objeto): Observable<Objeto> {
24     return this.http.post<Objeto>(OBJETO_API, this.srch);
25   }
26   getObjeto(): Observable<Objeto> {
```

4. Añadir Logger en todo para todo el código
Se agrego el componente
Ngx-logger

```
npm install --save ngx-logger
```

se lo añadió al archivo Modules



```
angular.json M TS listar.service.ts M TS app.module.ts M X listar.component.html M J ToDoListService.java M TS listar.interfac
ToDoList > ToDoList_Front > src > app > TS app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { ListarComponent } from './listar/listar.component';
7 import { ModalModule } from 'ngx-bootstrap/modal';
8 import { LoggerModule, NGXLoggerLevel } from 'ngx-logger';
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     ListarComponent
14   ],
15   imports: [
16     BrowserModule,
17     AppRoutingModule,
18     LoggerModule.forRoot({
19       serverLoggingUrl: '/api/logs',
20       level: NGXLoggerLevel.DEBUG,
21       serverLogLevel: NGXLoggerLevel.ERROR
22     }),
23   ],
24   providers: [],
25   bootstrap: [AppComponent]
26 })
27 export class AppModule { }
28
```

Así como también a cada uno de los elementos del proyecto

```
43 |  
44 | }  
45 |  
46 |  
47 | logWithContext(): void {  
48 |     this.logger.error('Logging from ListarService');  
49 | }  
50 | }
```