



|  |
| --- |
| Implementando uma camada de serviço  {  constructor( |
|  | public restApi: RestApiService, |
|  | public router: Router |
|  | ) { } |
|  |  |
|  | ngOnInit() { } |
|  |  |
|  | addEmployee(dataEmployee) { |
|  | this.restApi.createEmployee(this.employeeDetails).subscribe((data: {}) => { |
|  | this.router.navigate(['/employees-list']) |
|  | }) |
|  | } |

}

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Employee } from '../shared/employee';

import { Observable, throwError } from 'rxjs';

import { retry, catchError } from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class RestApiService {

// Define API

apiURL = 'http://localhost:3000';

constructor(private http: HttpClient) { }

/\*========================================

CRUD Methods for consuming RESTful API

=========================================\*/

// Http Options

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json'

})

}

// HttpClient API get() method => Fetch employees list

getEmployees(): Observable<Employee> {

return this.http.get<Employee>(this.apiURL + '/employees')

.pipe(

retry(1),

catchError(this.handleError)

)

}

// HttpClient API get() method => Fetch employee

getEmployee(id): Observable<Employee> {

return this.http.get<Employee>(this.apiURL + '/employees/' + id)

.pipe(

retry(1),

catchError(this.handleError)

)

}

// HttpClient API post() method => Create employee

createEmployee(employee): Observable<Employee> {

return this.http.post<Employee>(this.apiURL + '/employees', JSON.stringify(employee), this.httpOptions)

.pipe(

retry(1),

catchError(this.handleError)

)

}

// HttpClient API put() method => Update employee

updateEmployee(id, employee): Observable<Employee> {

return this.http.put<Employee>(this.apiURL + '/employees/' + id, JSON.stringify(employee), this.httpOptions)

.pipe(

retry(1),

catchError(this.handleError)

)

}

// HttpClient API delete() method => Delete employee

deleteEmployee(id){

return this.http.delete<Employee>(this.apiURL + '/employees/' + id, this.httpOptions)

.pipe(

retry(1),

catchError(this.handleError)

)

}

// Error handling

handleError(error) {

let errorMessage = '';

if(error.error instanceof ErrorEvent) {

// Get client-side error

errorMessage = error.error.message;

} else {

// Get server-side error

errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;

}

window.alert(errorMessage);

return throwError(errorMessage);

}

}