APPLICAZIONE academy_restful_web_service_consumer

L'Applicazione è sviluppata nell'ultima versione di Angular (con le standalone Components).

STRUTTURA DI UN PROGETTO ANGULAR 17/18.

package.json (contiene la lista delle dipendenze installate tramite npm) – non c'è nessuna variazione rispetto alle versioni precedenti di Angular.

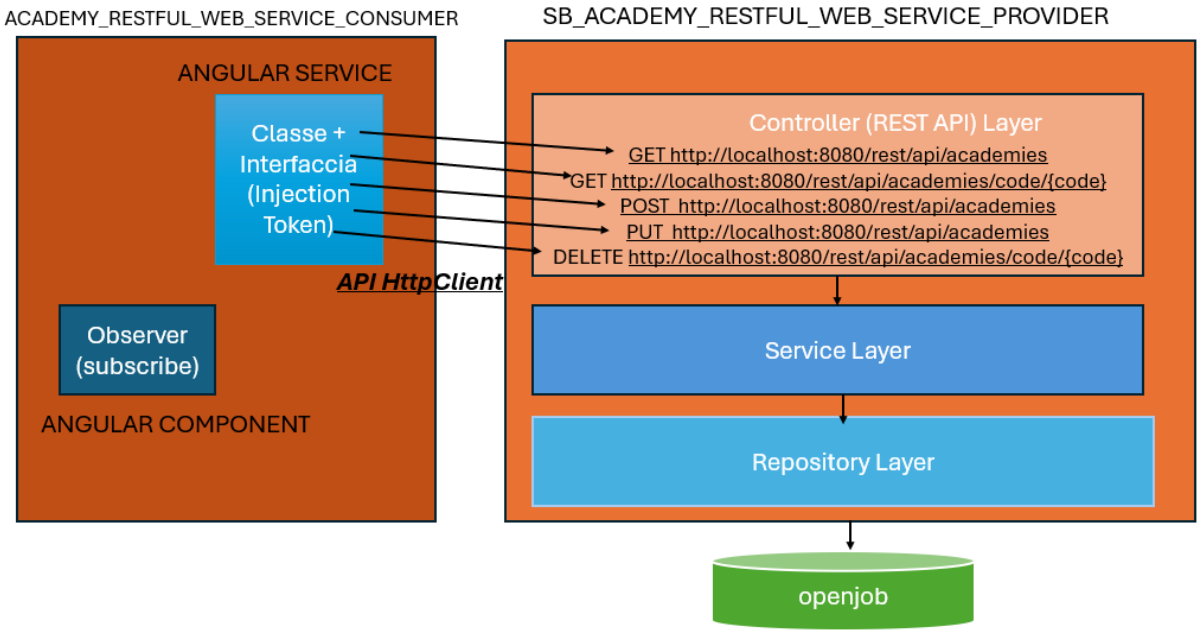
package-lock.json (contiene la lista delle dipendenze installate tramite npm e relativi link dell'npm registry dai quali le dipendenze vengono installate) – non c'è nessuna variazione rispetto alle versioni precedenti di Angular.

angular.json (è un file nel quale si possono fare configurazioni progettuali) – non c'è nessuna variazione rispetto alle versioni precedenti di Angular.

main.ts (file nel quale Node.js legge che allo start del progetto deve caricare l'AppComponent e deve esaminare il file app.config.ts) – c'è una variazione rispetto alle versioni precedenti di Angular, in cui il main.ts indicava a Node.js la necessità di caricare allo start il modulo AppModule. Nelle nuove versioni di Angular l'AppModule non è più obbligatorio. Le Component possono essere inserite in dei moduli,ma possono essere standalone, ed importare autonomamente moduli a seconda delle loro esigenze.

app.config.ts – non esisteva nelle versioni precedenti alla 17 di Angular. E' un file nel quale si può comunicare a Node.js di valutare la presenza di alcune API usate nel progetto, come ad esempio HttpClient per effettuare chiamate a Servizi rest.

ARCHITETTURA APPLICATIVA FULLSTACK



COMPONENT ANGULAR REALIZZATE

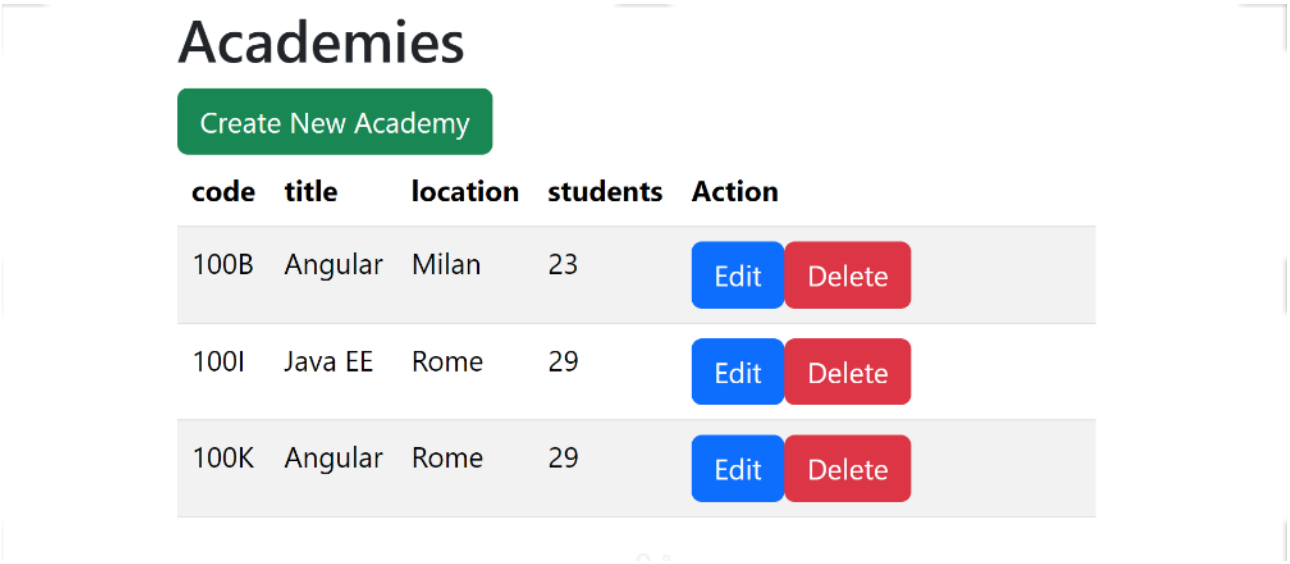


Figura 1 - Index Component (visualizza la lista delle Academies)

Figura 2 – CreateComponent (Component per l'inserimento di una nuova Academy)

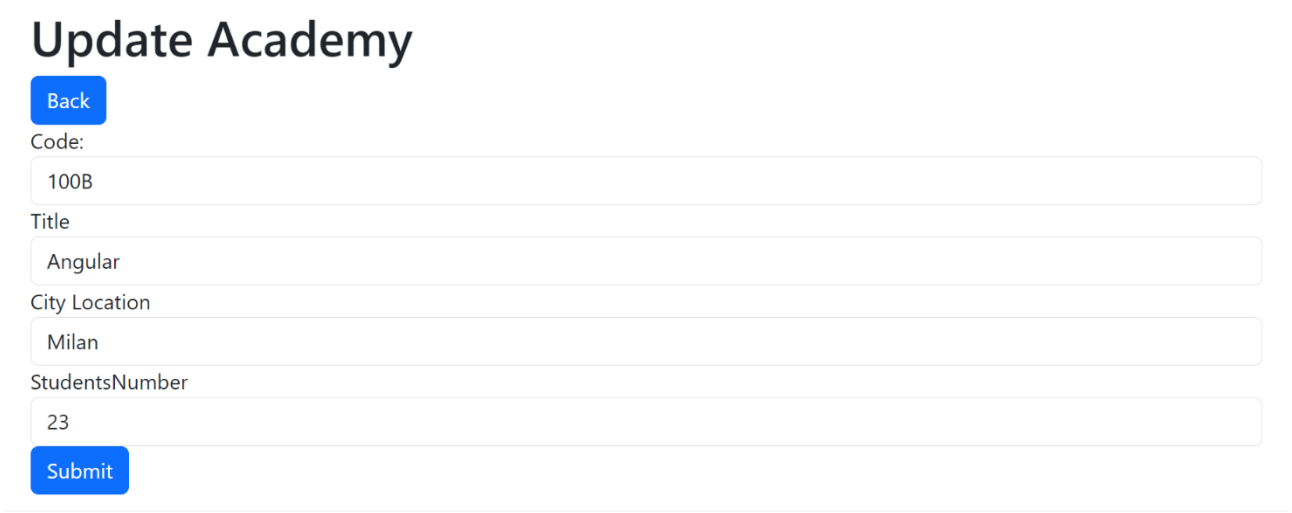


Figura 3 – EditComponent (Component per la modifica di una Academy già esistente)

STEP PROGETTUALI

1.CREAZIONE PROGETTO : ng new academy_restful_web_service_consumer

2. INSTALLAZIONE DIPENDENZA BOOTSTRAP

```
npm install bootstrap -save
```

3. CONFIGURAZIONE DIPENDENZA BOOTSTRAP IN angular.json

```
"node_modules/bootstrap/dist/css/bootstrap.min.css"
```

4. GENERAZIONE DI UN MODULO

```
ng generate module academy
```

5. GENERAZIONE DELLE COMPONENT DI PROGETTO

ng g c academy/index

ng g c academy/create

ng g c academy/edit

6. CONFIGURAZIONE DELLE ROTTE DI PROGETTO IN app.routes.ts

```
import { Routes } from '@angular/router';
import { IndexComponent } from './academy/index/index.component';
import { CreateComponent } from './academy/create/create.component';
import { EditComponent } from './academy/edit/edit.component';

export const routes: Routes = [
    { path: 'academy', redirectTo: 'academy/index', pathMatch: 'full'},
    { path: 'academy/index', component: IndexComponent },
    { path: 'academy/create', component: CreateComponent },
    { path: 'academy/:code/edit', component: EditComponent }
];
```

7. IMPLEMENTAZIONE INTERFACCIA MODELLO (SOTTO PACKAGE model academy.model.ts)

```
export interface Academy {

    code:string;
    title:string;
    cityLocation:string;
    studentsNumber:number;


}
```

8. CREAZIONE DI UNA CLASSE SERVICE (ng g s academy/academy) academy.service.ts

9. IMPLEMENTAZIONE DI UNA INTERFACCIA SERVICE (academyI.service.ts)

```
import { Observable } from "rxjs";
import { Academy } from "./model/academy.model";
import { InjectionToken } from "@angular/core";

export const academy_service_token = new
InjectionToken<AcademyServiceI>('academy_service_token');

export interface AcademyServiceI {


  getAcademies():Observable<any>;

  getAcademyByCode(code:string):Observable<any>;

  saveAcademy(academy:Academy):Observable<any>;

  updateAcademy(academy:Academy):Observable<any>;

  removeAcademy(code:string):Observable<any>;

}
```

10. IMPLEMENTAZIONE DELLA CLASSE SERVICE

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, Observable, throwError } from 'rxjs';
import { Academy } from './model/academy.model';
import { AcademyServiceI } from './academyI.service';

@Injectable({
  providedIn: 'root'
})
export class AcademyService implements AcademyServiceI{
```

```
  private apiURL = "http://localhost:8080/rest/api/academies";


  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json'
    })
  }

  constructor(private httpClient: HttpClient) { }


  getAcademies(): Observable<any> {

    return this.httpClient.get(this.apiURL)

    .pipe(
      catchError(this.errorHandler)
    )
  }

  getAcademyByCode(code:string): Observable<any> {

    return this.httpClient.get(this.apiURL + '/code/' + code)

    .pipe(
      catchError(this.errorHandler)
    )
  }


  saveAcademy(academy:Academy): Observable<any> {

    return this.httpClient.post(this.apiURL,JSON.stringify(academy),
this.httpOptions)

    .pipe(
      catchError(this.errorHandler)
    )
  }

  updateAcademy(academy:Academy): Observable<any> {

    return this.httpClient.put(this.apiURL,JSON.stringify(academy),
this.httpOptions)

    .pipe(
      catchError(this.errorHandler)
    )
```

```
  }


  removeAcademy(code:string):Observable<any>{

    return this.httpClient.delete(this.apiURL + '/code/' + code, this.httpOptions)

    .pipe(
      catchError(this.errorHandler)
    )
  }


  errorHandler(error:any) {

    let errorMessage = '';

    if(error.error instanceof ErrorEvent) {
      errorMessage = error.error.message;
    } else {
      errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
    }
    return throwError(()=>new Error(errorMessage));
 }

}
```

NB : HttpClient è un Service built-in fornito da Angular per effettuare chiamate REST, e in quanto tale è iniettabile nel costruttore dei Service Custom. I metodi post, put, delete e get di HttpClient hanno come tipo di ritorno Observable. Observable è una API asincrona rxjs, libreria di front end trasversale, utilizzabile in qualunque applicazione e tecnologia Front End. Observable implementa il pattern Observer. Il Pattern Observer contempla la presenza di due attori applicativi : Observer, che si sottoscrivere per ricevere la notifica di eventi e li attende in modalità asincrona, e Observable, che notifica gli eventi. Observable ha un ciclo di vita rappresentato da 3 stati : NEXT (la chiamata REST è stata effettuata, ma la risposta non è ancora completata) ; COMPLETED (la risposta è stata completata con successo); ERROR (l'esito della risposta è negativo). Ogni Component che desidera invocare un Observable si deve sottoscrivere per farlo invocando il metodo subscribe Typescript.

11. IMPLEMENTAZIONE DELLA COMPONENT TS IndexComponent

```
import { Component, inject, Inject, OnInit } from '@angular/core';
import { Academy } from '../model/academy.model';
import { academy_service_token, AcademyServiceI } from '../academyI.service';
import { AcademyService } from '../academy.service';
import { CommonModule } from '@angular/common';
```

```
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-index',
  standalone: true,
  providers: [{ provide: academy_service_token, useClass: AcademyService }],
  imports: [CommonModule,RouterModule],
  templateUrl: './index.component.html',
  styleUrl: './index.component.css'
})
export class IndexComponent implements OnInit {


  academies: Academy[] = [];

  private academyService = inject<AcademyServiceI>(academy_service_token);

  constructor() {

  }

  getAcademies(): void {

    this.academyService.getAcademies().subscribe({
      next: (res) => {
        this.academies = res;
        console.log('Data fetched successfully', res);
      },
      error: (err) => {
        console.error('Error fetching data', err);
      }
    });

  }


  ngOnInit(): void {

   this.getAcademies();


  }

  removeAcademy(code: string) {

    this.academyService.removeAcademy(code).subscribe(res => {
      console.log(res.data);
      this.getAcademies();
    });
```

```
  }


}
```

12. IMPLEMENTAZIONE DELLA COMPONENT HTML index.component.html

```
<div class="container">
    <h1>Academies</h1>

    <a href="#" routerLink="/academy/create/" class="btn btn-success">Create New
Academy</a>

    <table class="table table-striped">
        <thead>
              <tr>
                <th>code</th>
                <th>title</th>
                <th>location</th>
                <th>students</th>
                <th width="250px">Action</th>
              </tr>
        </thead>
        <tbody>
              <tr *ngFor="let academy of academies">
                <td>{{ academy.code }}</td>
                <td>{{ academy.title }}</td>
                <td>{{ academy.cityLocation }}</td>
                <td>{{ academy.studentsNumber }}</td>
                <td>
                  <a href="#" [routerLink]="['/academy/', academy.code, 'edit']"
class="btn btn-primary">Edit</a>
                    <button type="button" (click)="removeAcademy(academy.code)"
class="btn btn-danger">Delete</button>
                </td>
              </tr>
        </tbody>
    </table>
 </div>
```

13. IMPLEMENTAZIONE DELLA COMPONENT HTML create.component.html

```
<div class="container">
    <h1>Save New Academy</h1>
```

```
    <form [formGroup]="form" (ngSubmit)="submit()">

        <div class="form-group">
            <label for="code">Code:</label>
            <input
                formControlName="code"
                id="code"
                type="text"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="body">Title</label>
            <input
                formControlName="title"
                id="title"
                type="text"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="cityLocation">City Location</label>
            <input
                formControlName="cityLocation"
                id="cityLocation"
                type="text"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="studentsNumber">Students Number</label>
            <input
                formControlName="studentsNumber"
                id="studentsNumber"
                type="number"
                class="form-control">
        </div>

        <button class="btn btn-primary" type="submit">Submit</button>
    </form>
</div>
```

14. IMPLEMENTAZIONE DELLA COMPONENT TS create.component.ts

```
import { Component, inject, OnInit } from '@angular/core';
import { FormControl, FormGroup, FormsModule, ReactiveFormsModule, Validators }
from '@angular/forms';
import { academy_service_token, AcademyServiceI } from '../academyI.service';
import { AcademyService } from '../academy.service';
```

```
import { Router } from '@angular/router';
import { CommonModule } from '@angular/common';
import { Academy } from '../model/academy.model';


@Component({
  selector: 'app-create',
  standalone: true,
  providers: [{ provide: academy_service_token, useClass: AcademyService }],
  imports: [CommonModule,ReactiveFormsModule],
  templateUrl: './create.component.html',
  styleUrl: './create.component.css'
})
export class CreateComponent implements OnInit{


  academy! : Academy;

  private academyService = inject<AcademyServiceI>(academy_service_token);
  private router: Router = new Router;
  form!: FormGroup;

  ngOnInit(): void {

      this.form = new FormGroup({
        code: new FormControl(''),
        title: new FormControl(''),
        cityLocation: new FormControl(''),
        studentsNumber: new FormControl(0)
      });

    }

    submit() {

      this.academyService.saveAcademy(this.form.value).subscribe((res: any) => {
        console.log('Academy created successfully!');
        this.router.navigateByUrl('academy/index');
      })

    }




  }
```

15. IMPLEMENTAZIONE DELLA COMPONENT TS edit.component.ts

```
import { Component, inject, OnInit } from '@angular/core';
import { Academy } from '../model/academy.model';
import { CommonModule } from '@angular/common';
import { academy_service_token, AcademyServiceI } from '../academyI.service';
import { AcademyService } from '../academy.service';
import { ActivatedRoute, Router } from '@angular/router';
import { FormControl, FormGroup, FormsModule, ReactiveFormsModule, Validators }
from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';

@Component({
  selector: 'app-edit',
  standalone: true,
  providers: [{ provide: academy_service_token, useClass: AcademyService }],
  imports: [CommonModule,ReactiveFormsModule],
  templateUrl: './edit.component.html',
  styleUrl: './edit.component.css'
})
export class EditComponent implements OnInit {

  private academyService = inject<AcademyServiceI>(academy_service_token);

  academy!: Academy;

  code!: string;

  form!: FormGroup;

  constructor(private route: ActivatedRoute, private router: Router) {

  }

  ngOnInit(): void {

    this.code = this.route.snapshot.params['code'];
    this.academyService.getAcademyByCode(this.code).subscribe((data) => {
      this.academy = data;
      console.log(this.academy);

    });

    this.form = new FormGroup({
      code: new FormControl(''),
      title: new FormControl(''),
      cityLocation: new FormControl(''),
      studentsNumber: new FormControl()
    });
```

```
  }

  submit() {

    this.academyService.updateAcademy(this.form.value).subscribe((res: any) => {
      console.log('Academy updated successfully!');
      this.router.navigateByUrl('academy/index');
    })

  }



}
```

16. IMPLEMENTAZIONE DEL TEMPLATE edit.component.html

```
<div class="container">
    <h1>Update Academy</h1>


    <form [formGroup]="form" (ngSubmit)="submit()">

        <div class="form-group">
            <label for="code">Code:</label>
            <input
                formControlName="code"
                id="code"
                type="text"
                readonly
                [(ngModel)]="academy.code"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="body">Title</label>
            <input
                formControlName="title"
                id="title"
                type="text"
                [(ngModel)]="academy.title"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="cityLocation">City Location</label>
            <input
                formControlName="cityLocation"
```

```
                id="cityLocation"
                type="text"
                [(ngModel)]="academy.cityLocation"
                class="form-control">
        </div>

        <div class="form-group">
            <label for="studentsNumber">StudentsNumber</label>
            <input
                formControlName="studentsNumber"
                id="studentsNumber"
                type="number"
                [(ngModel)]="academy.studentsNumber"
                class="form-control">
        </div>

        <button class="btn btn-primary" type="submit">Submit</button>
    </form>
</div>
```

[(NgModel)] è UN TWO WAY BINDING CHE CONSENTE DI RIFLETTERE DATI DI UN OGGETTO TYPESCRIPT TRA UN TEMPLATE E UNA CLASSE COMPONENT E VICEVERSA.

LE FORM ANGULAR ALL'INTERNO DELLE QUALI VIENE USATO L'[(NgModel)] VENGONO TERMINOLOGICAMENTE CHIAMATE DAL FRAMEWORK TEMPLATE DRIVEN FORMS.

17.CONFIGURAZIONE app.component.html

```
<router-outlet></router-outlet>
```

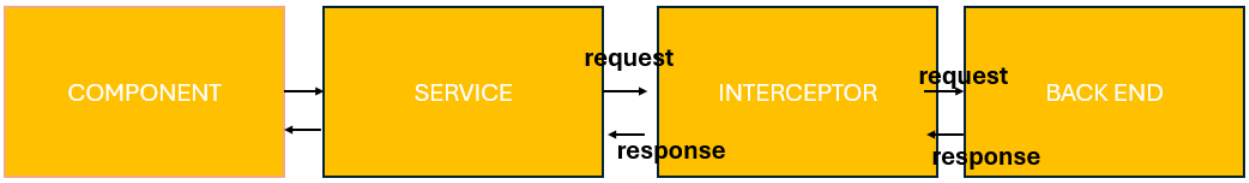
19. CONFIGURAZIONE app.config.ts

```
import { ApplicationConfig } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideAnimations } from '@angular/platform-browser/animations';

import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes), provideAnimations(), provideHttpClient()]
};
```

20. RUN DELL'APPLICAZIONE: npm start

21. AGGIUNTA DI UN INTERCEPTOR ANGULAR ALL'INTERNO DELL'APPLICAZIONE

GLI INTERCEPTOR SONO FUNZIONI PREDEFINITE ANGULAR (CUSTOMIZZABILI) CHE POSSONO ESSERE UTILIZZATE PER INTERCETTARE EVENTI DI RICHIESTA E/O RISPOSTA TRA L'APPLICAZIONE ANGULAR E APPLICAZIONI DI TERZE PARTI (APPLICAZIONI BACK END).



CREAZIONE INTERCEPTOR ALL'INTERNO DEL PERCORSO src/app/module

ng g interceptor logger

CODICE loggerinterceptor.interceptor.ts

```
import { HttpInterceptorFn } from '@angular/common/http';

export const loggerInterceptor: HttpInterceptorFn = (req, next) => {
  console.log(req.urlWithParams);
  return next(req);
};
```

CAMBIARE IL CODICE DEL FILE app.config.ts

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideAnimations } from '@angular/platform-browser/animations';
import { provideHttpClient, withFetch, withInterceptors } from '@angular/common/http';
import { loggerInterceptor } from './module/loggerinterceptor.interceptor';

export const appConfig: ApplicationConfig = {
```

providers: [provideRouter(routes), provideAnimations(), provideHttpClient(withInterceptors([loggerInterceptor]), withFetch())

]

};

FARE IL RUN DELL'APPLICAZIONE path http://localhost:8080/rest/api/academies

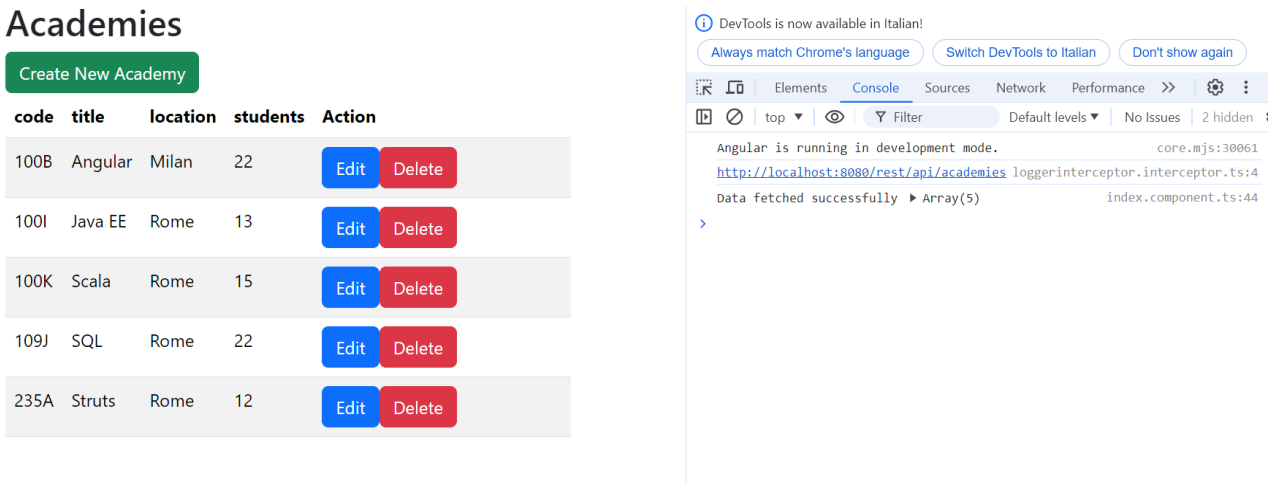


Figura 4 - Catch della url di chiamata verso il Servizio Rest