by Gabriel Alves Guimarães

Connecting Your Angular App to the Pokedex API

This e-book will guide you connecting your Angular application to the Pokedex API, allowing you to bring the exciting world of Pokémon to life.

Some slides were created using Gamma.app the remaining were designed by Gabriel Alves (me) using previous knowledge and Chat-GPT as a guide when necessary.

The previous knowledge was a boot camp provided by DIO in a partnership with Santander to create apps with Angular and Java.

Gamma.app was choose to provide the design to this presentation, slides created by Gamma.app have the ‘made with Gamma’ at the bottom-right.

I highly recommend this approach to create tutorials and e-books, but don’t trust in the IA, during the hole process it was necessary to review and adjust the content created many times and bring some of my self knowledge to overcome challenges within this project.

Why Learn this Skill?

Build Real-World Apps

Gain experience with real-world API interactions, a core skill for web development.

Engage Your Audience

Create interactive experiences that leverage the popularity and excitement of Pokémon.

Styling your App: Poké-themed design

1

Color Palette

Use vibrant colors inspired by the Pokémon world.

2

Fonts

Select playful fonts that evoke a sense of fun and adventure.

3

Icons

Incorporate Pokémon-themed icons to enhance the visual experience.

Getting Started: Setting up your Angular Project

1

1. Create a New Project

Utilize the Angular CLI to quickly create a new Angular project using the command "ng new my-pokedex-app".

2

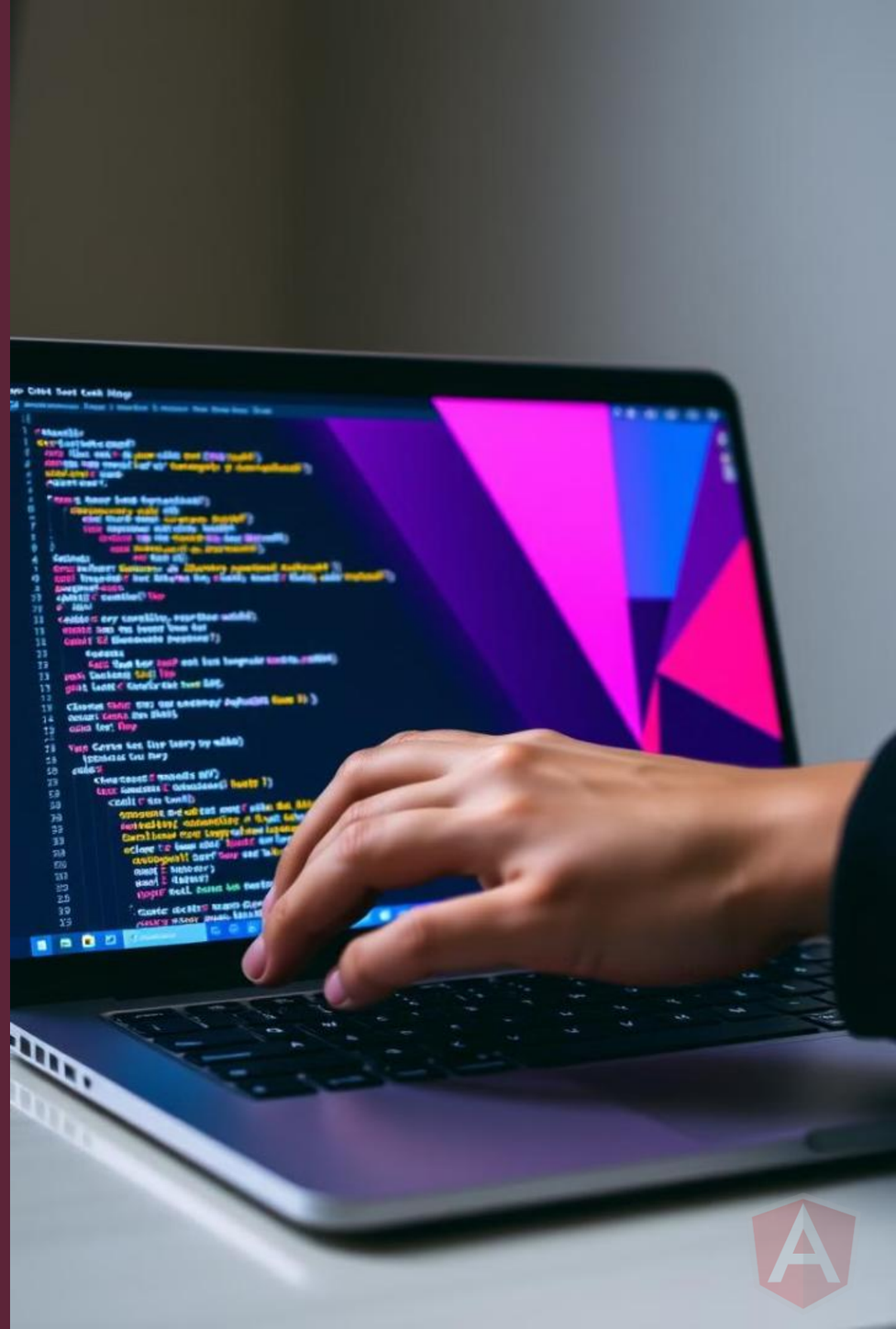
2. Configure Dependencies

Configure the necessary dependencies

3

3. Create Components

Create new components using the CLI



Create a New Project

Prerequisite (environment with visual studio code installed and internet connection)

1 - Install Node JS (LTS version) - https://nodejs.org/

Go to your application prompt of command (terminal) and check if node and NPM are installed using the commands:

```
CMD
node -v
npm -v
```

2 - Install the angular CLI globally

In the same terminal or in a new one, use the command

```
CMD
npm install -g @angular/cli
```

3 – Install the angular CDK component (in this project it'll be used to allow user to share the pokemon info)

```
CMD
npm install -g @angular/cdk
```

4 – Create a new project using the command:

```
CMD
ng new pokedex-app
```

For this project, choose CSS as the stylesheet format and don't enable SSR/SSG

Note: this project will use the latest approach Standalone Component

Open your Project and run it once!

1 - Go to your project folder using the command:

```
CMD
cd pokedex-app
```

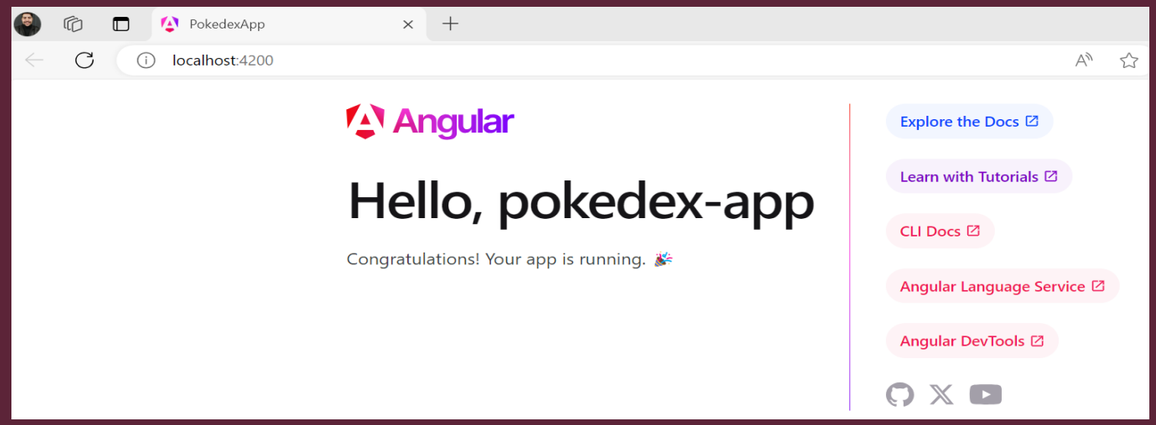
2 - Use the command below to open visual studio code

```
CMD
code .
```

3 – Test your project using the command below in the vs code terminal:

```
CMD
ng serve
```

Use the URL provided in the terminal after this command



Resetting template and Configure Dependencies

1 – Delete the content inside the file *app.component.html*, leaving only the router tag:

app.component.html

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

2 – Check if the property standalone is set as true inside the file *app.component.ts*, if not add it.

src > app > TS app.component.ts > ...

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router'

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
```

(not needed if your project is using Angular 19 or above).

3 – To ensure the necessary dependencies are installed, use the command:

Terminal

```
ng add @angular/common
```

If already installed the process will state it.

3 – Configure the file *app.config.ts* to add the necessary dependencies like HTTPClient and the routes.

src > app > TS app.config.ts > ...

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';
import { provideRouter } from '@angular/router';
import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: false }),
    provideHttpClient(withInterceptorsFromDi()),
    provideRouter(routes)]
};
```

Create Components and add routes

1 – Create the components using the command:

```
Terminal

ng generate component home --standalone
ng generate component search --standalone
ng generate component pokemon-detail --standalone
```

2 – Check if the property standalone is set as true for each one of the components, if not add it.

(not needed if your project is using Angular 19 or above).

3 – Configure the file *app.routes.ts* to add the routes to the components created for this project.

app.routes.ts

```
src > app > app.routes.ts > ...
1  import { Routes } from '@angular/router';
2  import { HomeComponent } from './home/home.component';
3  import { SearchComponent } from './search/search.component';
4  import { PokemonDetailComponent } from './pokemon-detail/pokemon-detail.component';
5
6  export const routes: Routes = [
7    { path: '', component: HomeComponent },
8    { path: 'search', component: SearchComponent },
9    { path: 'details/:id', component: PokemonDetailComponent },
10 ];
```

Understanding the Pokedex API

Endpoints

The Pokedex API offers various endpoints for retrieving Pokémon data.

Data Format

The API returns data in JSON format, making it easy to parse and display.

Authentication

The Pokedex API is freely accessible without any authentication requirements.

Displaying Pokemon Images and Details

Image

Use the provided image URLs in the API response to display each Pokémon's image.

Details

Extract and display relevant Pokémon details like name, and type.

List

Organize Pokémon information in a visually appealing list, allowing users to browse easily.

Interactivity: Allowing users to search by name and type and share the pokemon with other applications

1 Search Bar

Enable users to search for specific Pokémon by name and type

Fetching Pokemon Data: API calls in Angular

1. Import provideHttpClient withInterceptorsFromDi

Import the provideHttpClient withInterceptorsFromDi into your app.config.ts to enable API calls.

2. Inject the HttpClient

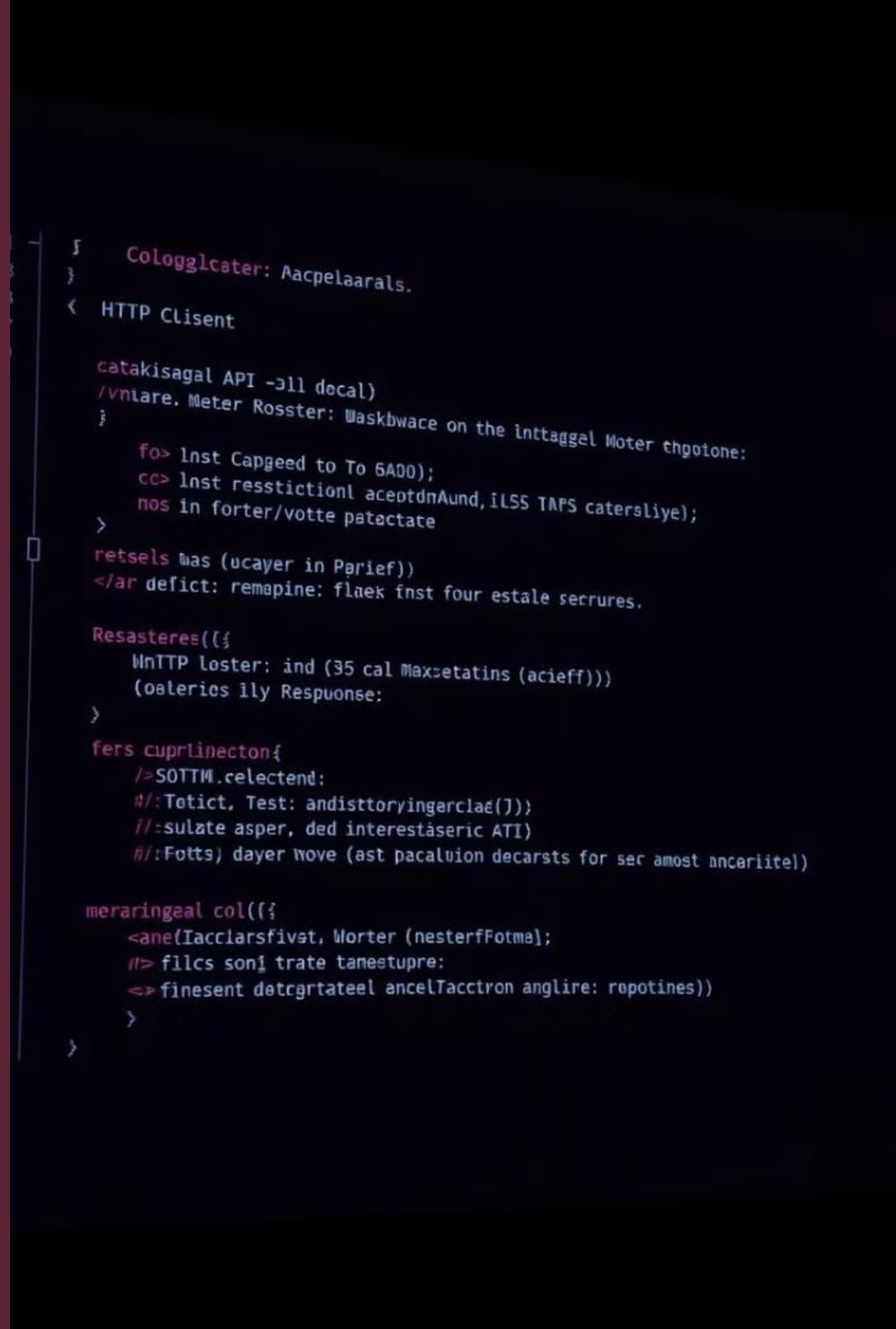
Inject the HttpClient into your component's constructor.

3. Make an API call

Use the HttpClient to make a GET request to the Pokedex API endpoint.

4. Handle the response

Parse the JSON data from the response and display it on your page.



Configuring Component – Home

HTML *(home.component.html)*

CMD

```html
<div class="home-container">
  <nav>
      <button (click)="redirect('https://pokeapi.co')">API Oficial</button>
      <button (click)="redirect('https://github.com/GabrielG1997')">GitHub</button>
    </nav>

    <main>
      <h1>Welcome to Pokedex</h1>
      <p>This project was created in order to practice the knowledge about angular
during a bootcamp provided by DIO about Generative IA with Copilot in partnership
with CAIXA, <br> One of the goals of this project was to create a simple e-book
tutorial about how to recreate it (you'll find the tutorial in the resources folder
of the github project) </p>
      <button (click)="goToSearch()">Buscar Pokémon</button>
    </main>
</div>
```

TypeScript *(home.component.ts)*

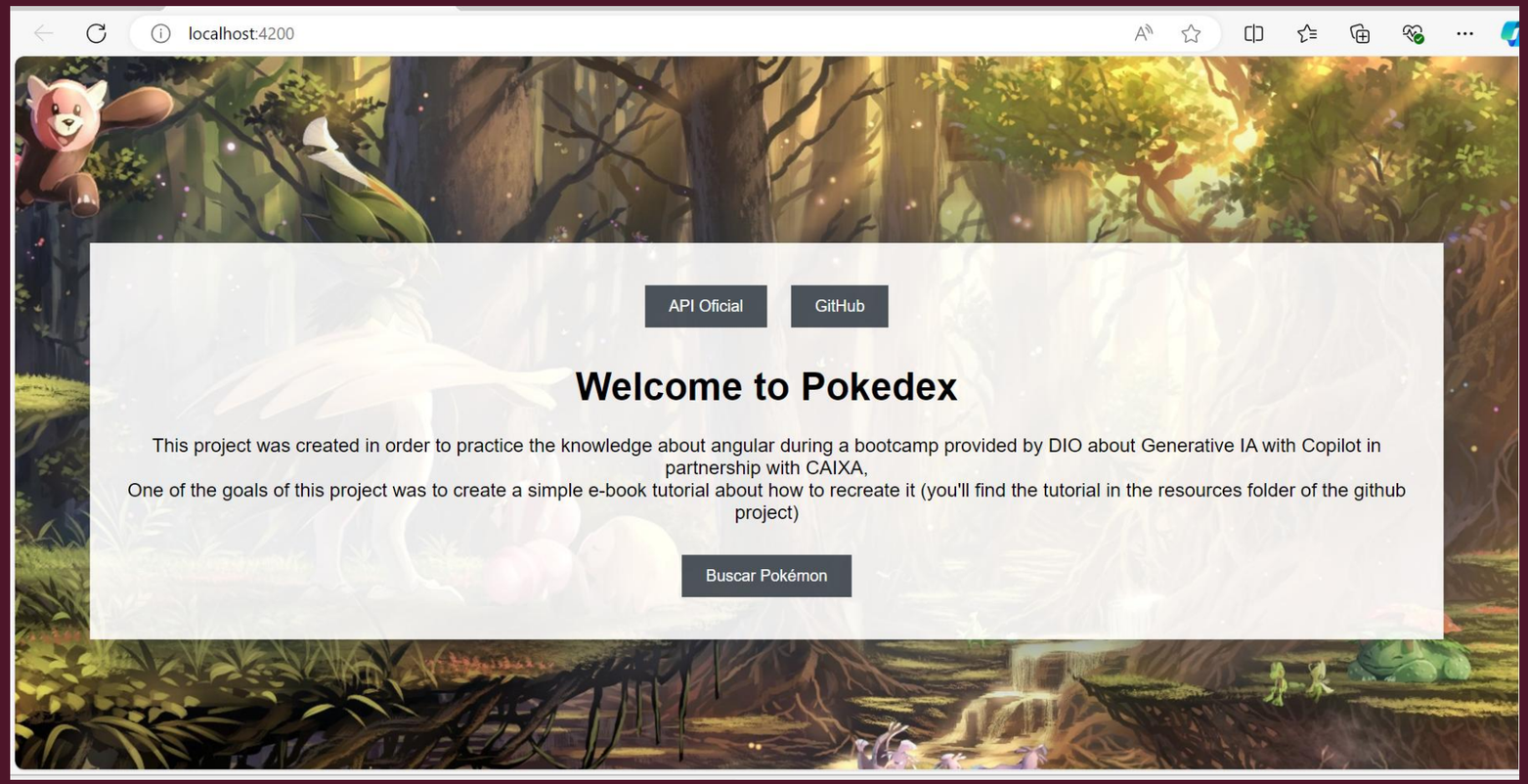
CMD

```typescript
import { Component } from '@angular/core';
import { Router } from '@angular/router';


@Component({
  selector: 'app-home',
  imports: [],
  templateUrl: './home.component.html',
  styleUrl: './home.component.css'
})
export class HomeComponent {
  constructor(private router: Router) {}
  goToSearch() {
    this.router.navigate(['/search']);
  }
  redirect(url: string) {
    window.open(url, '_blank');
  }
}
```

Component – Home - Explanation

1 Redirect Function

The home page contains only one method to redirect the user to the other pages or the GitHub / API docs.



Configuring Component – Search

HTML *(home.component.html)*

CMD

```
<div class="search-container">
    <h1>Search Pokemon</h1>
    <input [(ngModel)]="searchByIdTerm" placeholder="Search by Pokemon ID/name"/>
    OR
    <input [(ngModel)]="searchByTypeTerm" placeholder="Search by Pokemon Type"/>
    <button (click)="search()">Search</button>
    <button (click)="returnToHome()">Back to Home</button>

    <div class="results">
      <div *ngIf="isSingleResult && results.length" class="pokemon-card">
        <img [src]="results[0].sprites?.other?.dream_world?.front_default" alt="{{
results[0].name }}" />
        <h3>{{ results[0].name }}</h3>
        <p>ID: {{ results[0].id }}</p>
        <button (click)="viewDetails(results[0])">See Details</button>
      </div>

      <div *ngIf="!isSingleResult  && results.length" class="pokemon-list">
        <div *ngFor="let poke of results" class="poke-li">
          <img [src]="poke.sprites?.other?.dream_world?.front_default" alt="{{ poke.name }}"
class="poke-img" />
          <h2>{{ poke.name }}</h2>
          <h2>{{ poke.types[0].type.name }}</h2>
          <h3>ID: {{ poke.id }}</h3>
          <button (click)="viewDetails(poke)">See Details</button>
          </div>
      </div>
    </div>
</div>
```

TypeScript *(home.component.ts)*

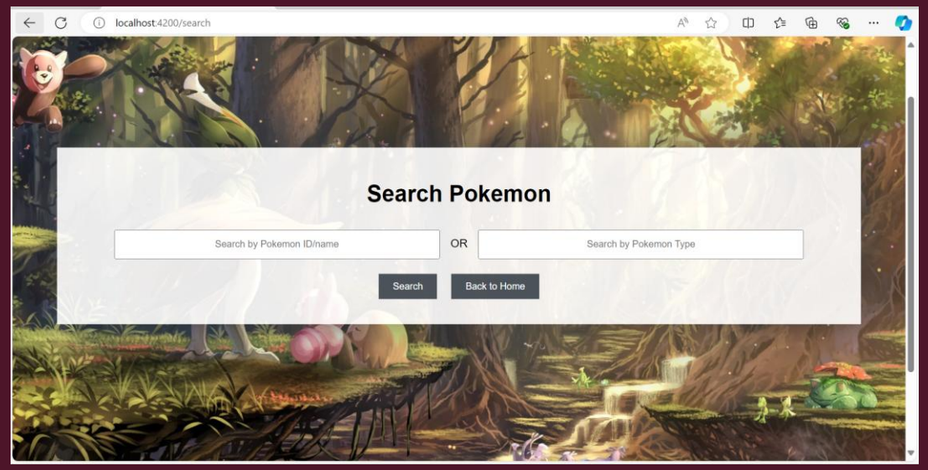
CMD

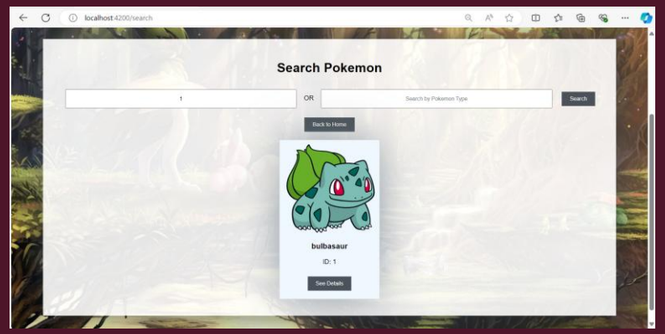
```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { forkJoin } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { throwError } from 'rxjs';
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root',
})
@Component({
  selector: 'app-search',
  imports: [FormsModule,CommonModule],
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css'],
})
export class SearchComponent {
  searchByIdTerm: string = '';
  searchByTypeTerm: string = '';
  results: any[] = [];
  apiUrl: string  = '';
  isSingleResult: boolean = false;

  constructor(private http: HttpClient, private router: Router) {}
  search() {
    if (this.searchByIdTerm.length > 0 && this.searchByTypeTerm.length == 0 ) {
      this.apiUrl = `https://pokeapi.co/api/v2/pokemon/${this.searchByIdTerm}`;
      this.isSingleResult = true;
    } else if(this.searchByTypeTerm.length > 0 && this.searchByIdTerm.length == 0) {
      this.apiUrl = `https://pokeapi.co/api/v2/type/${this.searchByTypeTerm}`;
      this.isSingleResult = false;
    }else if(this.searchByIdTerm.length > 0 && this.searchByTypeTerm.length > 0){
      alert('Search by ID/Name OR Type (Multiple search parameter detected)');
    }else{
      alert('Search by ID/Name or Type (search parameter is null)');
    }
    this.http.get(this.apiUrl).pipe(
      catchError((error) => {
        alert(`Error during requesition: ${error.error}: ${error.status}`);
        console.error('Error during requesition:', error);
        return throwError(() => new Error('Error to get data. Please, Try again.'));
      })
    ).subscribe((response: any) => {
      if (this.isSingleResult) {
        this.results = [response]; // Apenas um resultado
      } else {
        const requests = response.pokemon.map((p: any) =>
          this.http.get(p.pokemon.url).pipe(
            catchError((error) => {
              alert(`Error to fetch Pokemon details: ${p.pokemon.name} error: ${error}`);
              console.error(`Error to fetch Pokemon details: ${p.pokemon.name}`, error);
              return throwError(() => new Error('Error to fetch Pokemon details'));
            })
          )
        );
        forkJoin(requests).pipe(
          catchError((error) => {
            alert(`Error during Pokemon list processing, error: ${error}`);
            console.error('Error during Pokemon list processing: ', error);
            return throwError(() => new Error('Error during Pokemon list processing'));
          })
        ).subscribe({
          next: (pokemonDetails) => {
            this.results = pokemonDetails as any[];
          },
          error: (error) => {
            alert(`Error: ${error}`);
            console.error('Error: ', error);
          }
        });
      }
    });
  }
  viewDetails(pokemon: any) {
    if(this.isSingleResult){
      this.router.navigate(['/details', pokemon.id]);
    }else{
      this.router.navigate(['/details', pokemon.name]);
    }
  }
  returnToHome() {
    this.router.navigate(['']);
  }
}
```

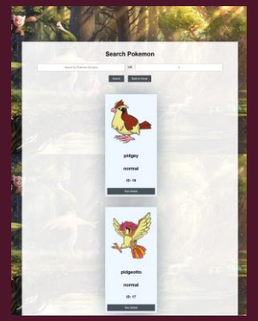
Component – Search - Explanation

1 Core concepts

- Data Binding:
  - Interpolation: {{ data }}
  - Property Binding: [property]="data"
  - Event Binding: (event)="handler()"
  - Two-Way Binding: [(ngModel)]="data“
- Directives:
  - ngIf: Conditional rendering.
  - ngFor: Looping through arrays.







This Component holds the most complex logic of this project and use different techniques to achieve its goal, allowing user to search by ID (id number or Pokémon name) and type (id number or type like water, grass, fire)

Configuring Component – Details

HTML *(home.component.html)*

CMD

```
<div *ngIf="pokemon" class="details-container">
    <img [src]="pokemon?.['sprites']?.['other']?.
['showdown']?.['front_default']" alt="{{ pokemon.name }}" />
    <h2>{{ pokemon.name }}</h2>
    <p>ID: {{ pokemon.id }}</p>
    <p>Tipo: {{ pokemon.types[0].type.name }}</p>
    <button [cdkCopyToClipboard]="pokemonDetailsToCopy"
(click)="share()">Copy to clipBoard</button>
    <button (click)="returnToSearch()">Back to
Search</button>
  </div>
```

TypeScript *(home.component.ts)*

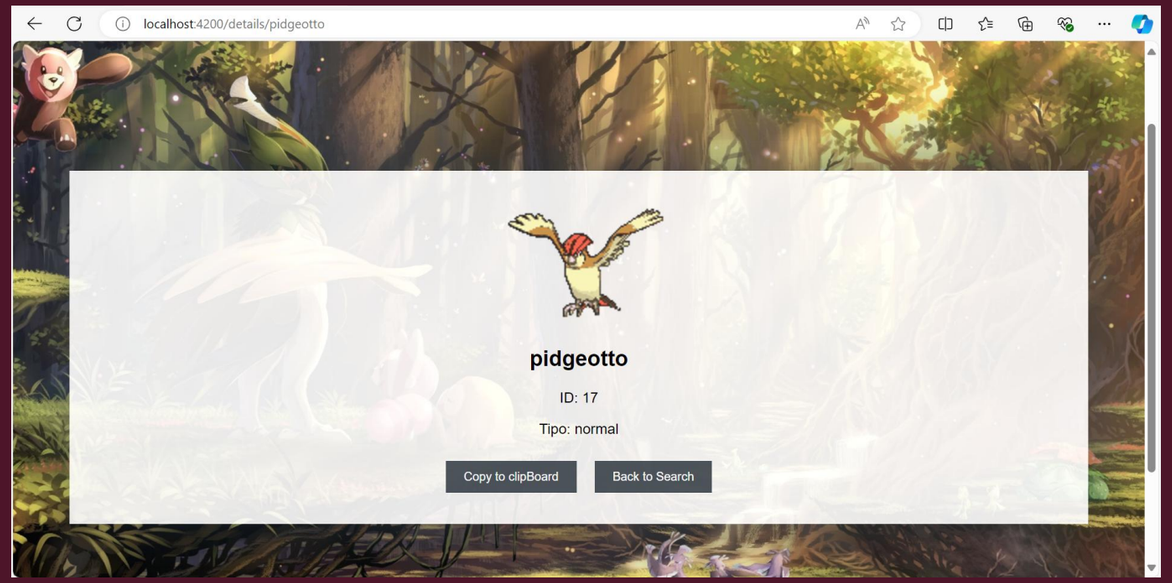
CMD

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ActivatedRoute, Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { ClipboardModule } from '@angular/cdk/clipboard'
import { Clipboard } from '@angular/cdk/clipboard';

@Component({
  selector: 'app-pokemon-detail',
  imports: [CommonModule, ClipboardModule],
  templateUrl: './pokemon-detail.component.html',
  styleUrl: './pokemon-detail.component.css'
})
export class PokemonDetailComponent {
  pokemon: any;
  pokemonDetailsToCopy: string = '';
  constructor(
    private route: ActivatedRoute,
    private http: HttpClient,
    private router: Router,
    private clipboard: Clipboard
  ) {
    const id = this.route.snapshot.paramMap.get('id');
    this.loadPokemon(id);
  }
  loadPokemon(id: string | null) {
    const apiUrl = `https://pokeapi.co/api/v2/pokemon/${id}`;
    this.http.get(apiUrl).subscribe((response) => (this.pokemon = response));
  }
  share(){
    this.pokemonDetailsToCopy = JSON.stringify({
      details:`https://pokeapi.co/api/v2/pokemon/${this.pokemon.id}`,
      id: this.pokemon.id,
      name: this.pokemon.name,
      image: this.pokemon.sprites?.front_default,
    });
    this.clipboard.copy(this.pokemonDetailsToCopy);
    alert('Pokémon shared!');
  }
  returnToSearch() {
    this.router.navigate(['/search']);
  }
}
```

Component – Details - Explanation

1 CDK library

- This page uses the CDK Library to allows the user to share the Pokémon information copying the Pokémon info to the clipboard.



There are other forms to solve this challenge, but using the CDK library was the easiest way to me.

Configuring Global CSS

stylesheet *(style.css)*

```css
body {
  background-image: url('https://img.goodfon.com/original/1920x1080/8/39/karamimame-pokemon-pokemony-pokemon-les.jpg');

  background-size: cover;
  background-position: center;
  background-repeat: repeat-y;
  height: 100vh;
  font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  .home-container,
  .search-container,
  .details-container {
    background-color: rgba(255, 255, 255, 0.9);
    text-align: center;
    align-content: center;
    margin: 20% 5%;
    padding: 2%;
    box-shadow: 0px -1px 107px -8px rgba(40, 54, 64, 0.945);
    -webkit-box-shadow: 0px -1px 107px -8px rgba(40, 54, 64, 0.945);
    -moz-box-shadow: 0px -1px 107px -8px rgba(40, 54, 64, 0.945);
  }
  .details-container img {
    width: 18%;
  }
  .search-container input{
    margin: 1%;
    padding: 1% 1%;
    text-align: center;
    width: 40% ;
  }

  button {
    background-color: rgba(42, 51, 59, 0.83);
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    margin: 10px;
```

```css
  button:hover{
    background-color: rgba(40, 97, 147, 0.83);
  }
  .pokemon-card {
    border: 1px solid #ddd;
    padding: 10px;
    margin: 10px;
    display: inline-block;
    text-align: center;
    background-color: aliceblue;
    box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
    -webkit-box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
    -moz-box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
  }
  .poke-li{
    padding: 25%, 25%;
    margin: 5% 38%;
    height: 15%;
    width: 25%;
    display: grid;
    background-color: aliceblue;
    border: 1px solid #ddd;
    box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
    -webkit-box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
    -moz-box-shadow: 0px -1px 107px -8px rgba(98, 116, 129, 0.801);
  }
  .poke-img{
    padding: 9%;
    width: 80%;
    height: 80%;
    margin: 1% 1%;
  }
```

Conclusion: Mastering Poké-powered Angular Apps

By connecting the Angular app to the Pokedex API, you've gained valuable experience with API interactions, built an engaging application, and explored the exciting world of Pokémon.

Keep learning, keep exploring, and continue building amazing applications with Angular and the Pokedex API!

Connect with me on:

LinkedIn:

GitHub:

Any suggestions or feedbacks are welcome, thanks for your time