Manual de Usuario: Conexión entre Backend y Frontend en Angular (Ionic)

Este documento describe paso a paso cómo estructurar un proyecto Angular con Ionic para establecer una conexión con un backend mediante servicios HTTP. Se detalla cómo organizar las carpetas, configurar la URL base, crear servicios y clases de modelo, e implementar métodos para operaciones CRUD.

---

1. Estructura del proyecto

Dentro del directorio `src/`, se recomienda establecer la siguiente estructura de carpetas:

```
src/
├── view/         // Contiene las vistas de la aplicación
├── component/    // Componentes reutilizables
└── service/      // Servicios para comunicación con el backend
```

Esta organización mejora la mantenibilidad y escalabilidad del proyecto.

---

2. Configuración de entorno: URL del backend

Para establecer la URL base de las peticiones HTTP, se debe modificar el archivo `environment.ts` ubicado en `src/environments/`.

Archivo: environment.ts

```
export const environment = {
  production: false,
  apiHost: 'http://localhost',
  apiPort: '8080',
  apiPrefix: '/api'
};
```

Este archivo permite definir valores de configuración que varían según el entorno. En este caso, se define la dirección del servidor backend, el puerto y el prefijo del API.

---

3. Registro del cliente HTTP en main.ts

Es necesario registrar el proveedor `HttpClient` para habilitar el uso de servicios HTTP. Esto se realiza en el archivo `main.ts`.

Archivo: main.ts

```
import { bootstrapApplication } from '@angular/platform-browser';
import { RouteReuseStrategy, provideRouter, withPreloading, PreloadAllModules } from
'@angular/router';
import { IonicRouteStrategy, provideIonicAngular } from '@ionic/angular/standalone';
import { provideHttpClient } from '@angular/common/http';
```

```
import { routes } from './app/app.routes';
import { AppComponent } from './app/app.component';

bootstrapApplication(AppComponent, {
  providers: [
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
    provideIonicAngular(),
    provideRouter(routes, withPreloading(PreloadAllModules)),
    provideHttpClient()
  ],
});
```

Con esta configuración, el módulo `HttpClient` estará disponible en toda la aplicación para hacer peticiones HTTP.

---

4. Creación de carpeta para servicios

Dentro de la carpeta `app/`, se debe crear manualmente una subcarpeta llamada `service/`. En esta carpeta se alojarán los archivos relacionados con la lógica de conexión al backend.

---

5. Creación del archivo api-url.ts

Dentro de la carpeta `service/`, se debe crear un archivo llamado `api-url.ts`. Este archivo se encargará de construir la URL base a partir de las variables definidas en `environment.ts`.

Archivo: api-url.ts

```
import { environment } from 'src/environments/environment';

const { apiHost, apiPort, apiPrefix } = environment;

export const API_BASE_URL = ${apiHost}:${apiPort}${apiPrefix
```

Este archivo centraliza la URL base para que pueda reutilizarse en los distintos servicios del proyecto.

---

6. Generar el servicio CRUD

Se debe generar un archivo de servicio llamado `crud.service.ts` dentro de la carpeta `service` con el siguiente comando:

```
ionic generate service crud
```

Este archivo contendrá los métodos para interactuar con el backend a través de HTTP.

---

7. Crear clase modelo para representar datos

Se debe generar una clase que represente la estructura de los datos que se gestionarán. En este caso, se creará una clase llamada `Home`:

```
ionic generate class home
```

Archivo: home.ts

```
export class Home {
    id: number;
    fullname: string;
    email: string;
    phone: number;
}
```

---

8. Definición de métodos en el servicio

Dentro del archivo `crud.service.ts`, se implementan los métodos que permiten realizar operaciones CRUD mediante `HttpClient`.

Archivo: crud.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { API_BASE_URL } from './api-url';
import { Home } from '../home/home';
import { Subject } from 'rxjs';

@Injectable({ providedIn: 'root' })
export class CrudService {

  public dataChanged$ = new Subject<void>();

  constructor(private http: HttpClient) {}

  getAll(endpoint: string) {
    return this.http.get(${API_BASE_URL}/${endpoint});
  }

  getById(endpoint: string, id: number) {
    return this.http.get(${API_BASE_URL}/${endpoint}/${id});
  }

  create(endpoint: string, data: Home) {
    return this.http.post(${API_BASE_URL}/${endpoint}, data);
  }

  update(endpoint: string, id: number, data: Home) {
    return this.http.put(${API_BASE_URL}/${endpoint}/${id}, data);
  }

  delete(endpoint: string, id: number) {
    return this.http.delete(${API_BASE_URL}/${endpoint}/${id});
  }
}
```
