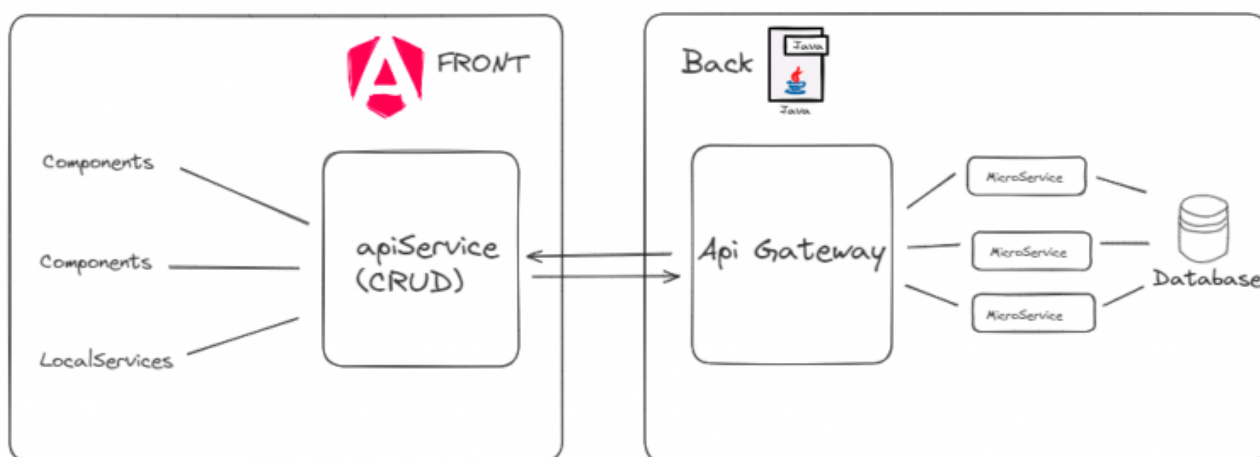


Esquema para el consumo de servicios

Para el consumo de servicios desde el front se debe tener en cuenta la convención para el consumo consulte artículo referente y tener en cuenta que la arquitectura consulte el artículo referente. A continuación se muestra un ejemplo de esquema de los consumos. Donde los distintos componentes tienen métodos que llaman a un servicio centralizado llamado apiService que contiene las operaciones para el CRUD.



Según la convención escogida para el consumo desde el BACK este sería un ejemplo de un body.

```
body = {
  page: 1,
  size: 20,
  searchsBy: [
    {
      key: 'identificacion',
      operation: 'GREATER_THAN',
      value: 0
    }
  ],
  ordersBy: [
    {
      column: 'identificacion',
      direction: 'ASC'
    }
  ]
};
```

Para el consumo estándar de un servicio se establecen 5 parámetros de los cuales obligatorio se debe pasar baseUrl que se encuentra en los entornos, endpoint que en este caso se define en el ejemplo dentro del objeto configTercero y opcionalmente pasamos params, headers y body. Los servicios creados en Spring requieren en algunos casos body y generalmente requieren los headers, como el ejemplo presentado en el artículo, mientras que los params se utilizan en la mayor parte de los casos.

para servicios del tipo get.

```
configTercero = {
  baseUrl: this.baseUrl,
  endpoint: "terceros/api/v1/consultar-terceros-paginado",
  params: null,
  headers: {
    "Authorization": "Bearer corantioquia-prod"
  },
  body: this.body,
};
```

El metodo `getData` recibe un endpoint como el descrito en `configTercero`, una `baseUrl`, `params`, un header, hace el consumo del metodo que se encuentra en `apiServices` pasandole dichos parametros, lo cual responde una vez suscrito como en el siguiente ejemplo con una data o un error, finalmente despues de la respuesta se desuscribe.

```
getData(endpoint: string, baseUrl: string, params?: any, header?: any): void {
  this.subscription = this.apiService.get<any[]>(endpoint, baseUrl, params, header).subscribe({
    next: (data) => {
      this.updateList(data);
    },
    error: (error) => {
      console.error('Error fetching data:', error);
    },
    complete: () => {
      this.subscription?.unsubscribe();
      this.progressBarService.hide();
    }
  });
};
```

La clase `ApiService` que se encuentra en la ruta `shared/components/services`, contiene los metodos del CRUD, para el consumo con el back, de ser necesario agregar nuevos metodos para implementaciones especificas se puede editar dicha clase.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, map } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ApiService {
  constructor(private http: HttpClient) { }

  get<T>(endpoint: string, baseUrl: string, params?: any, headers?:
  HttpHeaders): Observable<T> {
```

```
const url = `${baseUrl}/${endpoint}`;
return this.http.get<T>(url, { params, headers });
}

post<T>(endpoint: string, baseUrl: string, body: any, params?: any,
headers?: HttpHeaders): Observable<T> {
  const url = `${baseUrl}/${endpoint}`;
  return this.http.post<T>(url, body, { params, headers, responseType:
'json' }) as Observable<T>;
}

postText<T>(endpoint: string, baseUrl: string, body: any, params?: any,
headers?: HttpHeaders): Observable<string> {
  const url = `${baseUrl}/${endpoint}`;
  return this.http.post(url, body, { params, headers, responseType: 'text'
}) as Observable<string>;
}

put<T>(endpoint: string, baseUrl: string, body: any, headers?:
HttpHeaders): Observable<T> {
  const url = `${baseUrl}/${endpoint}`;
  return this.http.put<T>(url, body, { headers });
}

delete<T>(endpoint: string, baseUrl: string, headers?: HttpHeaders):
Observable<T> {
  const url = `${baseUrl}/${endpoint}`;
  return this.http.delete<T>(url, { headers });
}
}
```

From:
<http://wiki.adacsc.co/> - Wiki

Permanent link:
<http://wiki.adacsc.co/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:front:esquema&rev=1719507624>

Last update: 2024/06/27 17:00

