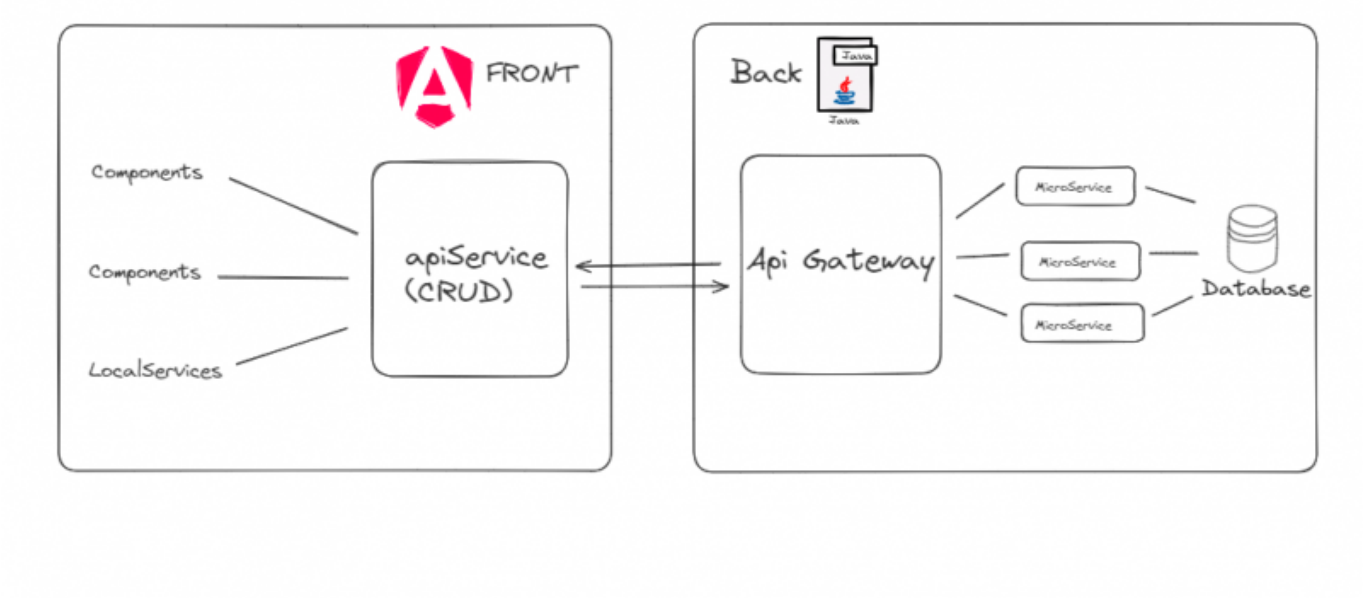
Esquema para el consumo de servicios

Para el consumo de servicios desde el front se deber tener en cuenta la convención para el consumo consulte articulo referente y tener en cuenta que la arquitectura consulte el articulo 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 parametros de los cuales obligatorio se debe pasar baseUrl que se encuentra en los enviroment, 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 srping requiere en algunos casos body y generalmente requieren los headers, como el ejemplo presentado en el artículo, mientras que los params se utiliza en la mayor parte de los caso

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 ecuentra en apiServices pasandole dichos parametros, lo cual responde una vez subscrito 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();
    }
  });
};
```