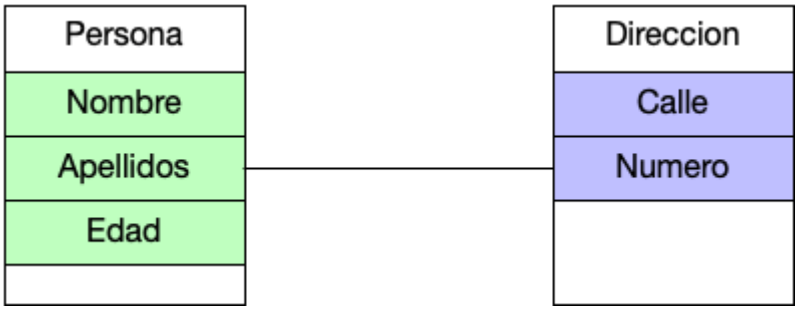
Cuando diseñamos un API REST , definimos como queremos que un recurso se presente de cara a una aplicación cliente. Es decir por ejemplo si tengo el concepto de Persona pues simplemente publico una serie de urls con metodos GET ,POST ,PUT y DELETE que se encargan de gestionar el recurso. ¿Ahora bien que hago cuando tengo relaciones y un Persona dispone de una dirección? . Recordemos que el concepto de Recurso REST es algo muy abierto . Tenemos varias opciones

Persona y Dirección

Una de las opciones más sencillas es publicar en una url el concepto de Persona.

`/personas`

Una vez tenemos publicado este concepto decidir que la dirección es un agregado de la persona.



Podemos acceder a el a través de una relación con una URL como :

`/personas/pedro/direccion`

De esta forma tendríamos una estructura muy limpia y ordenada . El problema que tiene esta estructura es que en algunos casos nos puede obligar a tener que realizar varias peticiones REST para obtener todos los datos que necesitamos de pedro. Por ejemplo podemos obtener la información de pedro con:

```
/personas/pedro
```

Esto nos devolverá la persona:

```
{
   "nombre": "pedro",
   "apellidos": "gomez",
   "edad": "20"
}
```

Si queremos acceder a la dirección simplemente ejecutamos otra petición Http y accedemos a ella:

```
/personas/pedro/dirección
```

Nos devolverá la dirección de pedro:

```
{
   "calle": "micalle",
   "numero": "1",
   "letra": "B"
}
```

Acabamos de hacer dos peticiones REST . No es algo que sea problemático pero puede ser que dependiendo del entorno suframos a nivel de rendimiento. Puede ser que una aplicación móvil prefiera realizar una única petición para obtener tanto los datos de la persona como su dirección

REST API y agregados

Para poder tener ese enfoque la información se tendrá que publicar como un agregado . Es decir una única url REST que en vez de devolver la persona únicamente ,adjunte ademas su dirección. Por ejemplo la url /personas/pedro devolvería los siguientes datos:

```
{
   "nombre": "pedro",
   "apellidos": "gomez",
   "edad": "20",
   "direccion": {
      "calle": "micalle",
      "numero": "1",
      "letra": "B"
   }
}
```

Dependiendo de nuestra situación deberemos aplicar una solución u otra a nuestros endpoints. La primera aporta una mayor flexibilidad y granularidad la segunda aporta un mayor rendimiento. Es la ventaja de usar REST y sus estilos.

Otros artículos relacionados

- Java Composicion y la reutilización del código
- Arquitecturas RESTFul y agregados
- Un ejemplo de JPA embedded objects
- ¿Que es CQRS ?
- Spring @PathVariable y segmentos de url