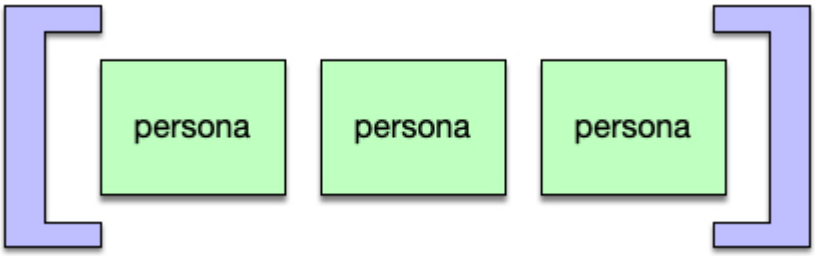
El concepto de JSON API es relativamente moderno . Cuando nosotros trabajamos con arquitecturas REST estamos muy acostumbrados a construirlas utilizando los niveles clásicos.

Hay situaciones en los que podemos usar nivel 2 o situaciones en las que podemos usar nivel 3 (HATEOAS) . Hasta aquí todo bastante clásico . Los problemas suelen surgir cuando estas arquitecturas se comienzan a complicar y necesitamos realizar operaciones más diversas . Pronto nos aparecerán las dudas , como ordenar , cómo paginar ,cómo incluir relaciones etc . La mayor parte de nuestras dudas no nos engañemos estarán relacionadas con las peticiones GET y las opciones de consulta que estas permiten.
Vamos a ver un ejemplo sencillo suponiendo que solicitamos la url de /personas.

```
[{
"nombre": "pedro",
"edad": "20",
"detalle": {
"estudios": "informatica"
},
{
"nombre": "gema",
"edad": "25",
```

```
"detalle": {
"estudios": "ingeniera"
}]
```

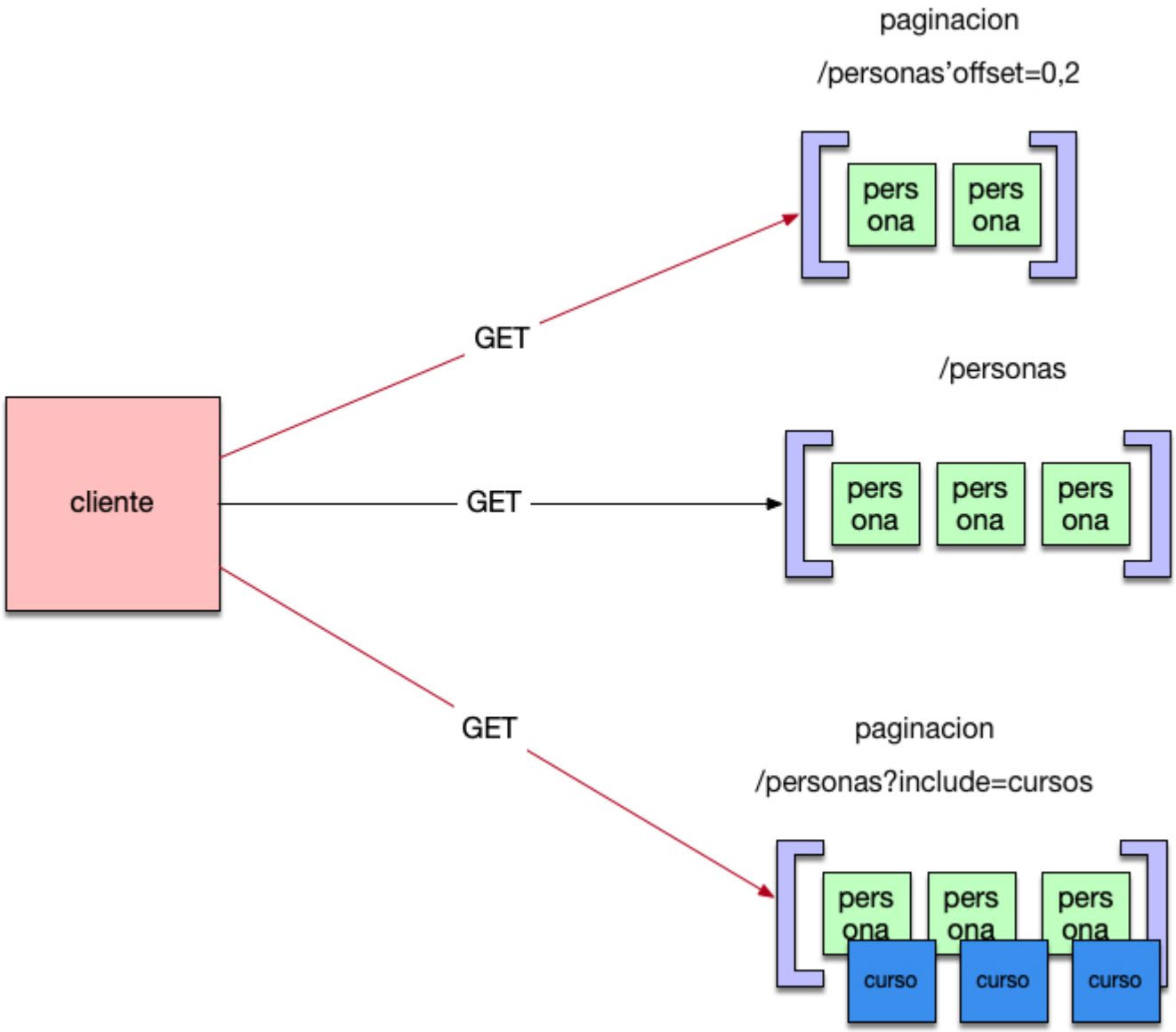
Esta puede ser una URL que nos lista las personas que tenemos en nuestra base de datos . Si nos preguntamos si la consulta es correcta , la respuesta es evidentemente que sí . Los datos nos son devueltos en formato JSON y tenemos acceso a la lista que deseamos.



Hasta aquí ningún problema . ¿Ahora bien es esta respuesta JSON flexible? . Esa es una pregunta muy diferente . En estos momentos nuestro programa cliente sea el que sea puede acceder a la lista esta claro.

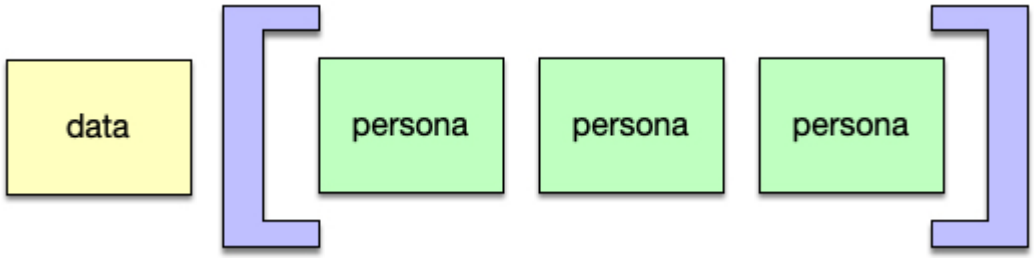
Sin embargo tiene sus limitaciones y una de las mas importantes es que la respuesta es muy plana .Simplemente se obtienen los datos y ya esta. Si un cliente quiere por ejemplo acceder al siguiente grupo de datos o quiere que la respuesta incluya relaciones o quiere … otras cosas. Estaremos ante una situación muy muy abierta y sobre todo por ahora poco flexible.

Nuestro cliente estará obligado a conocer de entrada el resto de URLS.

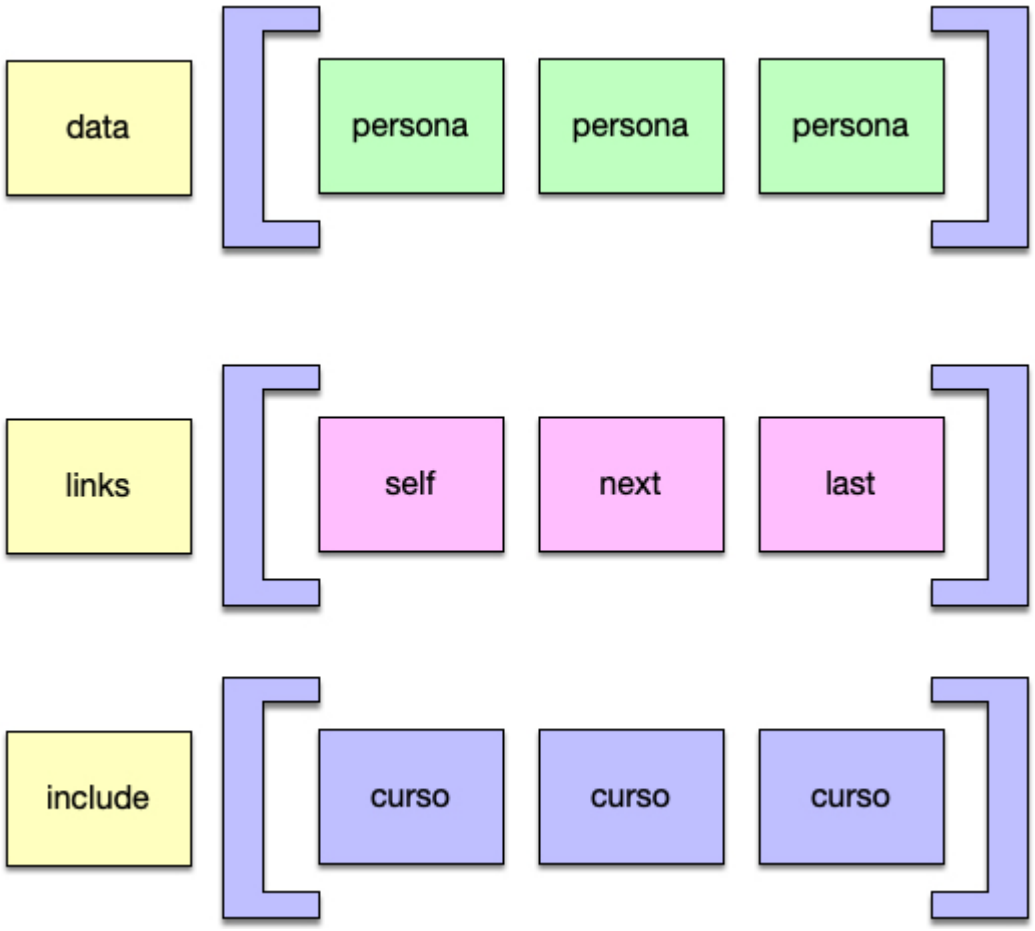


JSON API y su formato

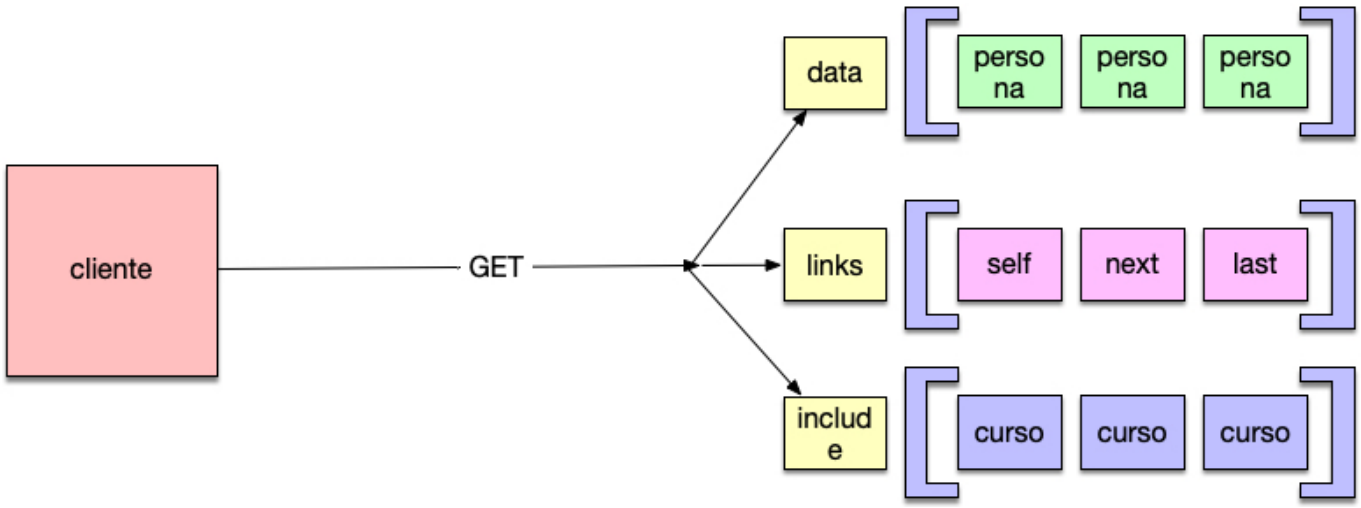
JSON API lo que propone es definir un API JSON que haga que estas respuestas adquieran un formato Standard. Por ejemplo en una estructura de JSON API los datos no se obtienen de una forma tan plana sino que vienen dentro de una estructura data.



Al ubicarlos dentro de esta estructura y solicitar a nuestros clientes que accedan a ellos estamos aportando flexibilidad ya que nos permite añadir a la petición GET subestructuras que complementen los datos expuestos . Dos de las mas habituales son links y includes . Que se encargan de incluir información sobre enlaces a datos complementarios así como información sobre elementos hijos que los datos originales necesiten.



Esto nos permitirá que nuestro cliente reduzca su acoplamiento de forma importante ya que a partir de este momento con la referencia a la estructura JSON que usa JSON api el resto de links no son necesarios.



Veámoslo en código:

```
 "links":[
"self": "localhost/personas?offset=1",
"next": "localhost/personas?offset=2",
"last": "localhost/personas?offset=10",

],
"data":[{
"nombre": "pedro",
"edad": "20",
"detalle": {
"estudios": "informatica"
}],
"included":[
```

```
"curso": "java",
]
```

El uso de JSON API nos puede ayudar a diseñar arquitecturas REST más flexibles.

1. REST DTO y JSON Arquitecturas Web y objetos
2. Spring REST CORS y su configuración
3. Introducción a JSON Web Token y la seguridad
4. https://jsonapi.org/format/