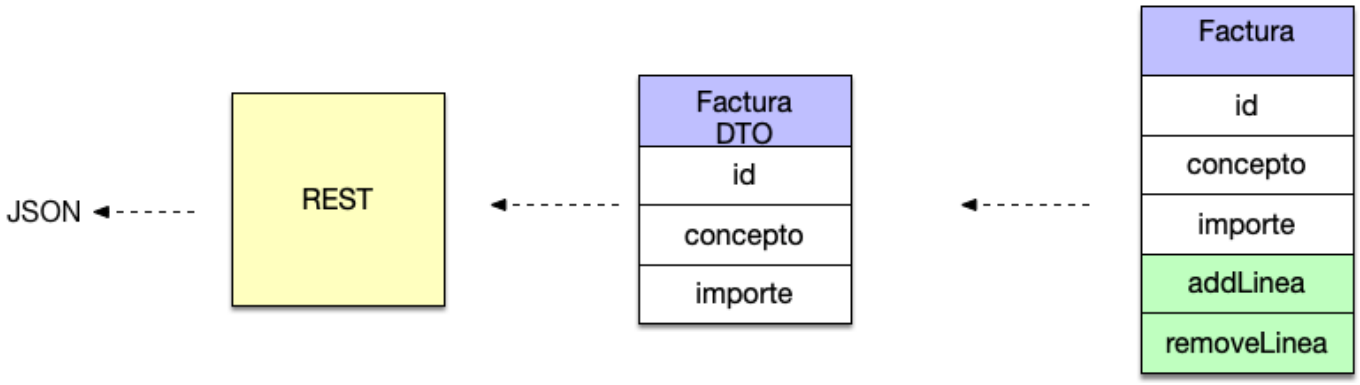
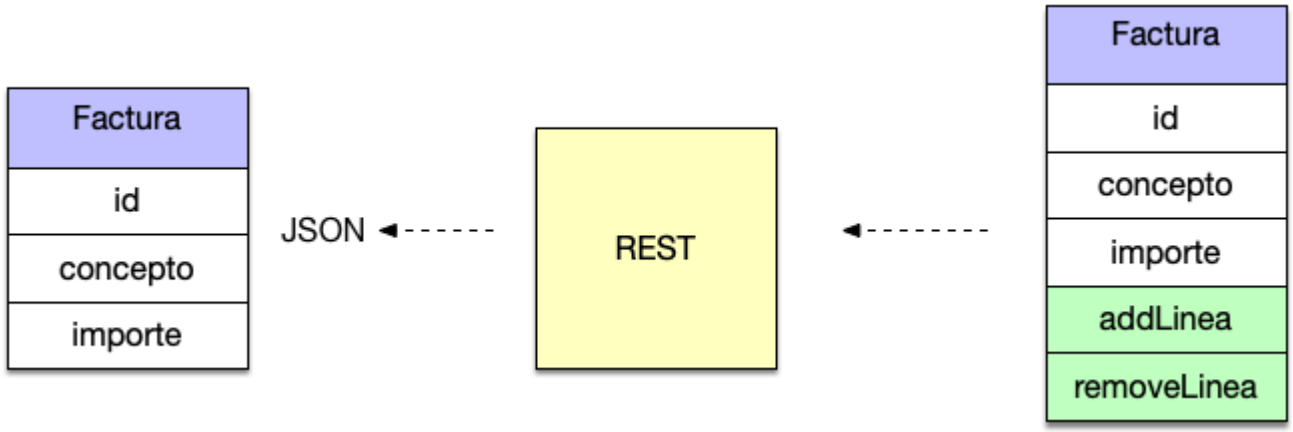
El concepto de REST DTO es muy habitual cuando trabajamos con arquitecturas REST . En este tipo de soluciones nos encontramos publicando recursos REST de forma continua y rapidamente nos vendrá a la mente una duda bastante razonable . ¿Qué publico mis objetos de negocio directamente con sus propiedades o publico unos DTO (Data Transfer Objects)en su lugar . Mucha gente nos comentará que es mejor usar DTOs ya que solo necesitamos las propiedades. Para ello probablemente usaremos una herramienta como Modelmapper .



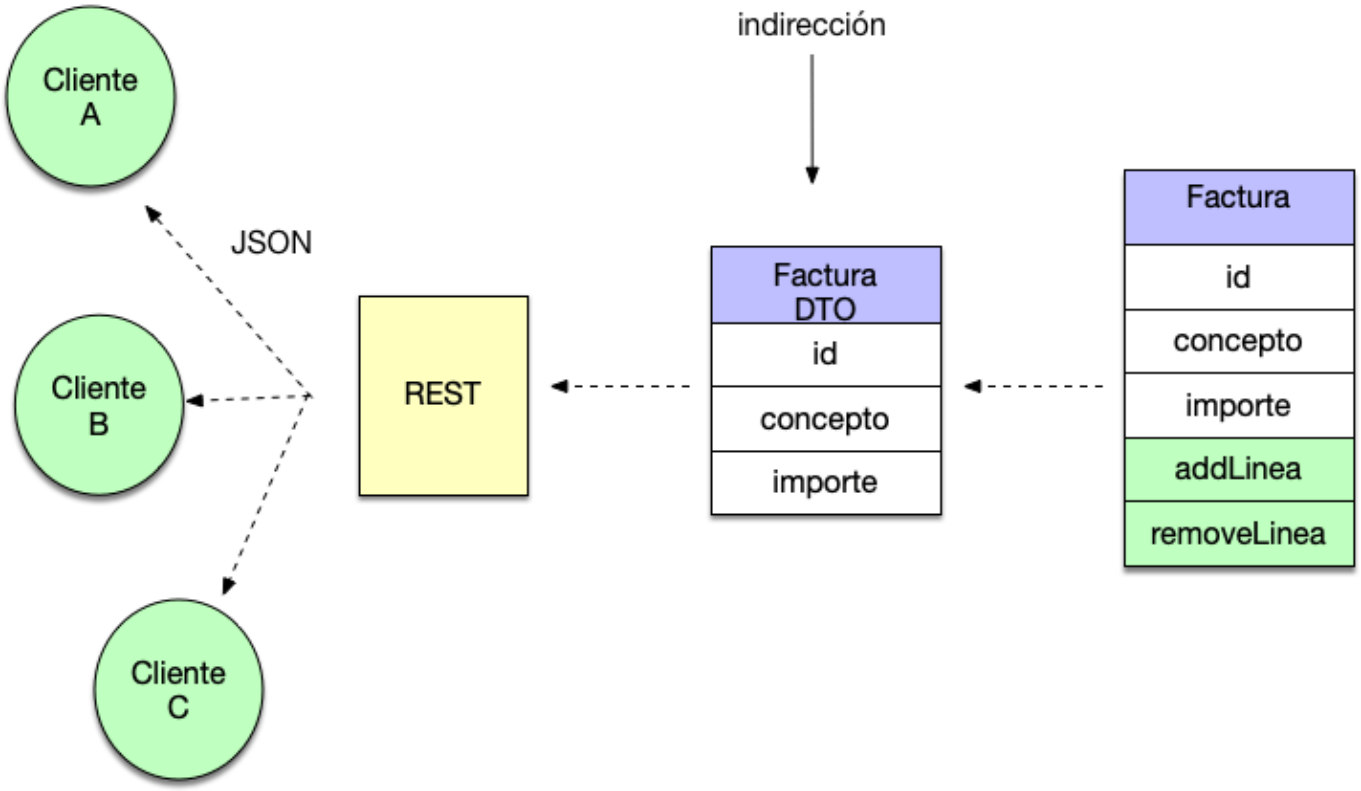
Ok esto puede ser cierto , pero hasta un punto determinado ya que al usar Arquitecturas REST nunca estamos publicando los métodos de los objetos de negocio. Es decir si solicitamos tener una lista de objetos en formato JSON estos únicamente tendrán las propiedades y por lo tanto serán DTOS de JavaScript por decirlo de alguna forma .



De esta forma nos ahorramos una capa importante de clases los DTO o Data Transfer Object y simplemente delegamos en los objetos de negocio que puede ser mucho más directos por decirlo de alguna forma . Ademas estos objetos de negocio suelen ser bastante estables a la hora de definir las propiedades ya que pertenecen al core del sistema . Una factura siempre es una factura y tiene más o menos la misma información. Hasta aquí todo parece correcto. Pero hay gente que nos vuelve a decir qué vale pero que los DTO no están mal. ¿Qué esta pasando , exactamente en este tipo de solución?.

REST DTO y JSON

Desde mi punto de vista ambas son posibles y si estuviéramos en un ejemplo sencillo probablemente el uso de objetos de negocio sería suficiente ya que se convertirían en DTOS JSON. Ahora bien no siempre las arquitecturas son tan definitivas . En muchas ocasiones las situaciones son más complejas por decirlo de alguna forma. En este caso la complejidad estriba en que la arquitectura REST que diseñamos probablemente sea consumida por un conjunto de clientes amplio. No solo eso sino que en muchas ocasiones ni siquiera tendremos consciencia de quién es el consumidor.



REST DTO e inmutabilidad

Es en estos casos en los que disponer de una capa REST DTO que nos permite añadir un nivel de indirección y controlar que propiedades y con que nombre se publican puede generar un mayor grado de inmutabilidad en el API REST algo que todos agradeceremos siempre en el futuro.

Otros artículos relacionados

1. Arquitecturas RESTFul y agregados
2. Spring REST CORS y su configuración
3. REST HTTP return codes y sus curiosidades
4. Arquitecturas REST y sus niveles