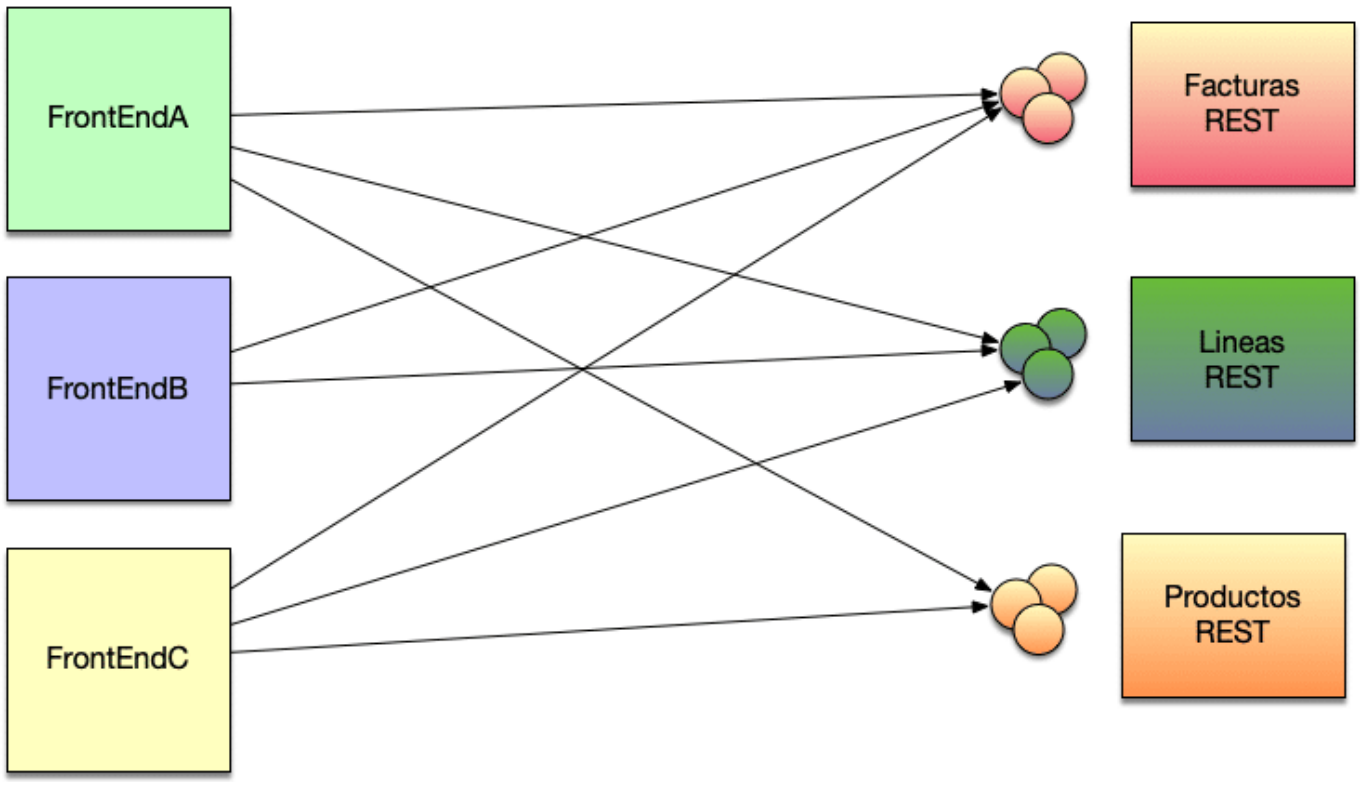
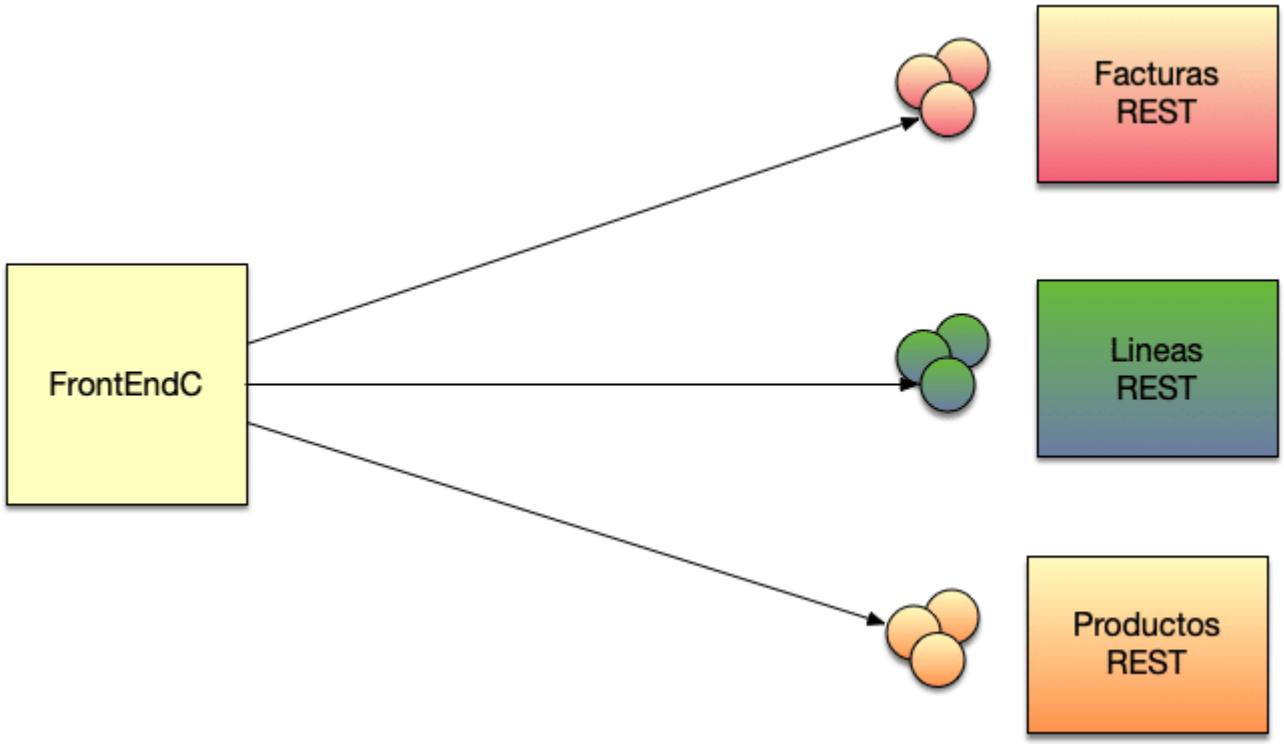
El patrón BFF hace referencia al concepto de Backend For FrontEnd. Cuando nosotros diseñamos muchas de nuestras aplicaciones nos encontramos en situaciones en las que tenemos diseñado un FrontEnd realizado en JavaScript contra un Backend desarrollado en la tecnología que nos plazca pero muchas veces enfocado a Servicios REST . En muchos casos el BackEnd esta diseñado de una forma neutra es decir publica una información de la que cualquier cliente puede beneficiarse . Eso sí cada cliente dispondrá de un interface de usuario diferente a otros . Al publicar la información de forma muy neutra todos los clientes se adaptarán a ella , pero cada uno de los cuales necesitará realizar un esfuerzo diferente para cargar los datos necesarios que presenta.



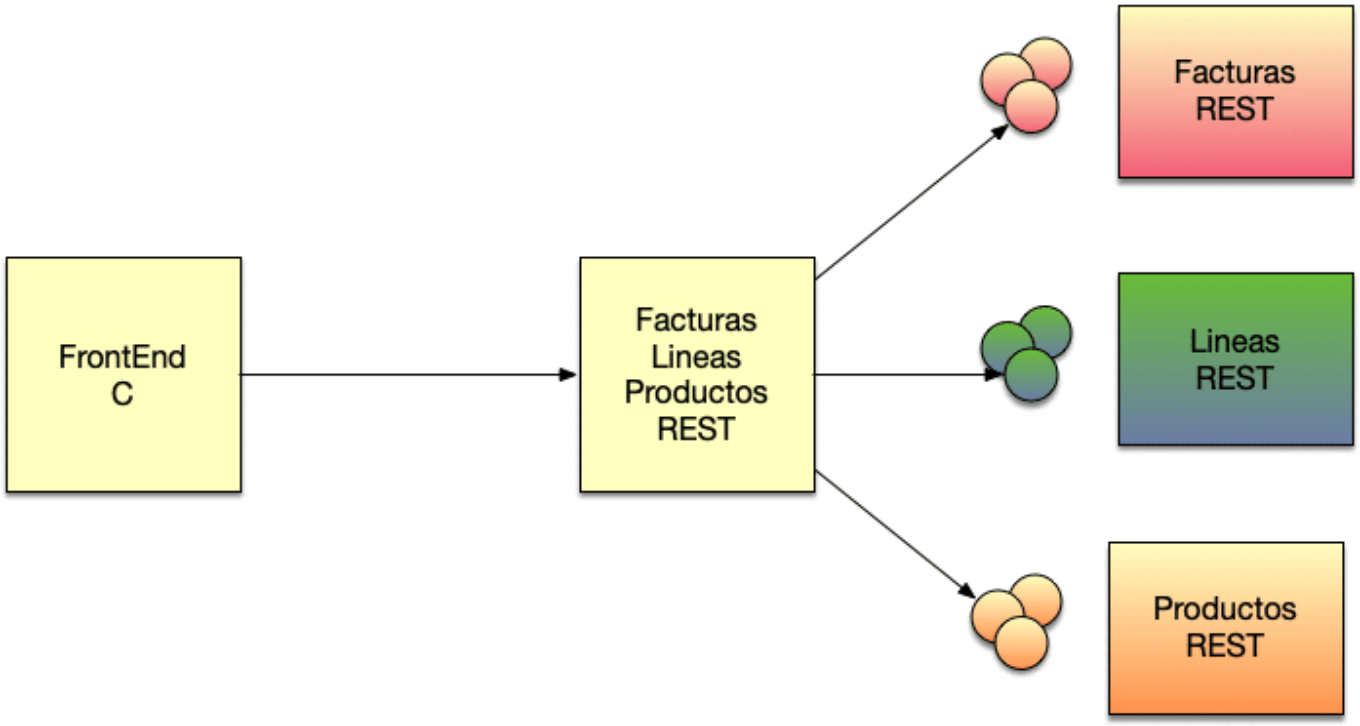
BFF y patronaje

Esto en principio es bastante correcto , pero hay situaciones en las que podemos necesitar realizar mejoras a nivel de simplicidad el interface de usuario y la carga de datos o incluso a nivel de rendimiento ya que puede ser que una vista concreta necesite los datos de una forma muy especifica para tener un mejor rendimiento. Ante estas situaciones , se puede

realizar una adaptación del backend de tal forma que este Backend publique la información de forma que sea mucho más fácil de consumir por parte del cliente . Por ejemplo imaginemos que el FrontEndC necesita acceder a las Facturas , a las Lineas y a los Productos todos de golpe. En una situación como esta se encontrará con que tiene la necesidad de consultar a un montón de servicios REST.



Esto hace que la gestión de peticiones al servidor sea relativamente compleja . En muchas ocasiones puede ser muy util rediseñar parte del lado servidor (Backend) para que publique la información que esas vistas necesitan concretamente.



En este caso estamos diseñando un Servicio REST que hace de fachada sobre el resto de servicios , adaptando los datos que el cliente realmente necesita y simplificando como este tiene que actuar y cargar la información . Tengamos en cuanta el patrón BFF cuando estamos trabajando en nuestro día a día.

Acabamos de ver como usar Java equals y hashcode

Otros artículos relacionados

1. Spring Boot REST JPA y JSON
2. REST API y su inmutabilidad
3. API REST estilos y homogeneidad