Clase 4/6

#PAW

Idea: pasar cosas de la clase pasada a la práctica

Final: tiramos Spring WebMVC y usamos Jersey

nueva dependencia pom padre: org.glassfish.jersey.containers

La idea es dejar de usar el dispatcher en web.xml (front controller de spring) (lo demás, spring security etc, lo quiero dejar). (@Controller ya no sirve para nada).

con @Path indico a qué recursos responde este Controller.

ResponseBuilder ➜ patrón Builder

la implementación de GenericEntity guarda la información sobre la clase asociada ➜ va a poder serializar

convertiré mi modelo a DTO ➜ me da más versatilidad

mi API debe ser stateless ➜ todos mis requests autocontenidos.
Cuando uno tiene web apps grandes tener sesiones es un problema ➜ para soportar más tráfico cómo hace? ➜ escala verticalmente u horizontalmente. Verticalmente (más recursos (RAM, etc) pero es caro).
Yo quiero que mañana mi webapp sean X servidores y vos no tenes ni idea a quién te conectas (loadBalancer, reverseproxy, nginx). ➜ load balancer utiliza RoundRobin con StickySessions (mi cookie siempre le pega al mismo servidor, pero rompe mi premisa de distracción uniforme mi trafico). Entonces cómo hago si no tengo sesiones ni estado? ➜ necesito que todos mis requests sean autocontenidos (info sufí para autenticarme como licito ejecutor de esa acción). (Authorization header *no* es seguridad). ➜ HMAC firmo requests con timestamp o JWT (token único que id mi sesión, y puedo hacer que mute en cada request ➜ evito ataques de tipo *Replay*)

Recomendación de Juanma Sotuyo: usen JWT ➜ entrar al primer resultado (im feeling lucky).

A partir de ahora, siempre tecnología front….

Esta es la última clase que “codeamos a la par”

A partir de la clase que viene vamos a querer a empezar notas.