Web server

Se nos pide modelar la estructura de un **web server**.

Un web server atiende ***pedidos***. Cada pedido indica:

- la IP de quien hace el pedido, se puede manejar como un String.
- día y hora del pedido. Se recomienda usar para esto la clase `java.time.LocalDateTime`.
- la URL que se está requiriendo, p.ej. `"http://pepito.com.ar/documentos/doc1.html"`.

De una URL nos van a interesar estos datos, que se describen tomando `"http://pepito.com.ar/documentos/doc1.html"` como ejemplo.

- el protocolo, en este caso `"http"`.
- la ruta, en este caso `"/documentos/doc1.html"`.
- la extensión, en este caso `"html"`.

La ***respuesta*** a un pedido consiste de: tiempo que tardó en responder (en milisegundos), status code, y payload que es un String. También tiene que tener una referencia al pedido que la generó.
Los web servers que modelemos van a aceptar solamente el protocolo HTTP. Si el protocolo de la URL es distinto a `"http"` hay que devolver status code 501 (servicio no implementado) y payload vacío.

Un web server deriva todos los pedidos que acepta a un ***módulo***. Al web server se le pueden cargar la cantidad de módulos que corresponda.
Cuando se recibe un pedido, se le pregunta a todos los módulos que tiene configurados si lo pueden atender o no. El server deriva el pedido al primer módulo que le dice que sí. El módulo genera la respuesta cargándole payload y tiempo de respuesta, y la devuelve. El server le agrega a esa respuesta el status code 200 (OK), el dato del pedido que la generó, y la devuelve.
Si no hay ningún módulo que pueda atender el pedido, hay que devolver status code 404 (Not found) y payload vacio.

Para poder testear esta implementación, crear "***falsos módulos***" que trabajan por extensión. Para cada "falso módulo" se debe indicar: qué extensiones acepta (una colección de Strings), más un tiempo y un payload de respuesta.

A un web server también se le tienen que poder agregar ***analizadores***, que son objetos que registran y/o analizan distintos aspectos del tráfico que atiende el server.
Puede no haber ningún analizador, uno o muchos. Se tienen que poder agregar y quitar dinámicamente.
Ante cada pedido que atiende, el server le tiene a todos los analizadores que tenga asignados en ese momento la respuesta y el módulo que la generó.

Implementar los siguientes analizadores

1. **IPs sospechosas**.
Se le carga una colección de IPs sospechosas, y un *registro de actividad sospechosa*. Si el pedido que originó la respuesta proviene de una IP sospechosa, hay que pasarle ip, ruta de la URL y status code de respuesta.
Un registro de actividad sospechosa debe entender estos mensajes:

```
registrarActividad(ip, ruta, status code)
cantidadPedidosConRespuestaExitosa(ip)
cantidadPedidosConRespuestaNoExitosa(ip)
consultoRuta(ip, ruta)  // true o false
```

2. **Estadísticas**.
A este se le tiene que poder preguntar: tiempo de respuesta promedio, cantidad de pedidos entre dos momentos (fecha/hora) que fueron atendidos, cantidad de respuestas cuyo payload incluye un determinado String (p.ej. cuántas respuestas dicen "hola", lo que incluye "hola amigos" y "ayer me dijeron hola 4 veces"), porcentaje de pedidos con respuesta exitosa.

3. **Detección de demora en respuesta**.
Se setea con una demora mínima en milisegundos. Una respuesta cuyo tiempo de respuesta supere la demora mínima se considera demorada. Se le tiene que poder preguntar, para un módulo, la cantidad de respuestas demoradas.

Requerimiento

Que el Web Server atienda pedidos, devolviendo las respuestas que generan los “falsos módulos”, e interactuando con los analizadores.