CLASE 41

En esta práctica vamos a implementar un servidor web al cual se pueda acceder mediante un navegador web de manera interactiva mediante la incorporación de HTML, JavaScript y CSS.

El proyecto maven adjunto contiene un servidor web funcional el cual contiene el siguiente árbol de directorios:

```
./
├── pom.xml
└── src
    ├── main
    │   ├── java
    │   │   └── com
    │   │       └── mycompany
    │   │           └── app
    │   │               ├── App.java
    │   │               ├── FrontendSearchRequest.java
    │   │               ├── FrontendSearchResponse.java
    │   │               └── WebServer.java
    │   └── resources
    │       └── ui_assets
    │           ├── index.html
    │           ├── javascript.js
    │           └── style.css
    │
    └── test
        └── java
            └── com
                └── mycompany
                    └── app
                        └── AppTest.java
```

Para crear el archivo jar sólo debe ejecutar en la raíz:

```
mvn clean compile assembly:single
```

y como en ejemplos anteriores dentro de la carpeta `target` se encontrará el archivo JAR con todas las dependencias. Al ejecutar el archivo JAR se levantará un servidor web en el puerto 3000.

Acceda al servidor mediante un navegador web y retomando sus conocimientos en aplicaciones web, revise con un editor de texto los assets web que se encuentran en la carpeta `ui_assets` y verifique que se están incorporando en la aplicación web. Dele un vistazo también a los cuatro códigos Java disponibles en la carpeta `app` para darse una idea general del funcionamiento de esta aplicación web.

Posteriormente realice las siguientes actividades y suba al teams de su equipo las respuestas correspondientes (se recomienda también comentar su código con los conocimientos obtenidos al responderlas):

a) Intente acceder al servidor desde su teléfono celular y verifique que funciona correctamente. Es necesario que el firewall del equipo que ejecuta el servidor web permita las conexiones HTTP externas. También es posible que requiere ejecutar el archivo jar dentro de una consola DOS en lugar de usar WSL (mandar la captura de pantalla del celular como comprobante).
b) Qué devuelve el servidor al intentar acceder a los endpoint declarados en WebServer.java: `/status`, `/`, `/ui_assets` y `/procesar_datos`
c) Agregue las impresiones que considere necesarias en el código *WebServer.java* para determinar cuál es el primer método HTTP enviado del navegador hacia el servidor web.
d) ¿Cuál es el primer, segundo y tercer **asset** enviados del servidor hacia el navegador? En el método `sendResponse` agregue la impresión del número de bytes enviados para verificar que coinciden en tamaño con los archivos enviados.
e) ¿En el código del servidor cuál es el propósito de ejecutar `getClass().getResourceAsStream()` ?
f) ¿Para qué se ejecuta el método `addContentType()` ?
g) Después de recibir los assets, al introducir la frase en el navegador y dar click en enviar, ¿qué método HTTP se ejecuta, a qué endpoint accede y qué método del código Java se ejecuta?
h) En el inciso anterior ¿qué valor tiene el header “Content-type” cuando el mensaje llega al servidor? (imprimir su valor)
i) En el método `handleTaskRequest`: ¿Para qué se utiliza el método de Jackson `readValue`?¿qué se almacena en la variable `String frase`? ¿qué se almacena en el objeto `frontendSearchResponse`? ¿Para qué se utiliza el método de Jackson `writeValueAsBytes`?
j) En el servidor HTTP analizado con anterioridad, la línea final del método `sendResponse` era `exchange.close()` , ¿Por qué se omitió esta vez?
k) Además de los assets HTML, CSS y JS, el navegador solicita un cuarto asset. ¿Cuál es y que significa? Realice las modificaciones necesarias para que también se envíe el cuarto asset y se muestre en el navegador. Envíe la captura de pantalla correspondiente para demostrar que funciona.