Programación de Servicios y procesos

Tarea para PSP05.

Enunciado.

Ejercicio 1.

Modifica el ejemplo del servidor HTTP (Proyecto java **ServerHTTP**, apartado 5.1 de los contenidos) para que incluya la cabecera Date.

Como apunta el enunciado, la mayor parte del peso de este ejercicio, viene dado por el ejemplo propuesto en el tema, completamente funcional.

Lo adaptamos para añadirle una cabecera, mediante una nueva función

SERVIDORHttp.java

```
/*
 * TAREA PSP05. EJERCICIO 1.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que incluya la cabecera Date.
 * El servidor se basa en la versión 1.1 del protocolo HTTP.
 * Implementa solo una parte del protocolo con solo peticiones GET
 * y dos tipos de mensajes: peticiones de clientes a servidores y
respuestas
 * de servidores a clientes.
 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package servidorhttp;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;


/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */


/**
 * ************************************************************
 * Servidor HTTP que atiende peticiones de tipo 'GET' recibidas por el
puerto
 * 8066
 *
 * NOTA: para probar este código, comprueba primero de que no tienes
ningún otro
```

```
 * servicio por el puerto 8066 (por ejemplo, con el comando 'netstat'
si estás
 * utilizando Windows)
 */
public class ServidorHttp {

    /**
     *******************************************************
     * procedimiento principal que asigna a cada petición entrante un
socket
     * cliente, por donde se enviará la respuesta una vez procesada
     * @param args the command line arguments
     * @throws IOException, requerida al crear nuestro ServerSocket
     */
    public static void main(String[] args) throws IOException,
Exception {
        //Asociamos al servidor el puerto 8066
        ServerSocket socServidor = new ServerSocket(8066);
        //El método que escribirá en servidor las instrucciones
        imprimeDisponible();
        /**
         * @param socCliente socket de comunicación.
         */
        Socket socCliente;

        //ante una petición entrante, procesa la petición por el
socket cliente
        //por donde la recibe
        while (true) {
          //a la espera de peticiones
          socCliente = socServidor.accept();
          //atiendo un cliente
          System.out.println("Atendiendo al cliente ");
          //Salta al método que sirve al cliente la página
          procesaPeticion(socCliente);
          //cierra la conexión entrante
          socCliente.close();
          System.out.println("cliente atendido");
        }

    }//Fin clase main

    /**
     * Método que retorna la fecha y hora actual de un modo apropiado
para
     * usarlo en cabeceras HTTP
     * @return
     */
    public static String getDateValue(){
        DateFormat df = new SimpleDateFormat(
        "EEE, d MMM yyyy HH:mm:ss z", Locale.ENGLISH);
        df.setTimeZone(TimeZone.getTimeZone("GMT"));
        return df.format(new Date());
    }

    /**
   ***************************************************************
   * procesa la petición recibida
   *
   * @throws IOException
   */
```

```
  private static void procesaPeticion(Socket socketCliente) throws
IOException {
    /**
     * Variables locales
     * @param peticion String para el BufferReader.
     * @param html String recoge datos desde paginas.
     */
    String peticion;
    String html;

    //Flujo de entrada
    InputStreamReader inSR = new
InputStreamReader(socketCliente.getInputStream());
    //espacio en memoria para la entrada de peticiones
    BufferedReader bufLeer = new BufferedReader(inSR);

    //objeto de java.io que entre otras características, permite
escribir
    //'línea a línea' en un flujo de salida
    PrintWriter printWriter = new
PrintWriter(socketCliente.getOutputStream(), true);

    //mensaje petición cliente
    peticion = bufLeer.readLine();

    //para compactar la petición y facilitar así su análisis,
suprimimos todos
    //los espacios en blanco que contenga
    peticion = peticion.replaceAll(" ", "");

    //si realmente se trata de una petición 'GET' (que es la única que
vamos a
    //implementar en nuestro Servidor)
    if (peticion.startsWith("GET")) {
      //extrae la subcadena entre 'GET' y 'HTTP/1.1'
      peticion = peticion.substring(3, peticion.lastIndexOf("HTTP"));

      //si corresponde a la página de inicio
      if (peticion.length() == 0 || peticion.equals("/")) {
        //sirve la página
        html = Paginas.HTML_INDEX;
        printWriter.println(Mensajes.LINEA_INICIAL_OK);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
        printWriter.println("\n");
        printWriter.println(html);
      } //si corresponde a la página del Quijote
      else if (peticion.equals("/quijote")) {
        //sirve la página
        html = Paginas.HTML_QUIJOTE;
        printWriter.println(Mensajes.LINEA_INICIAL_OK);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
```

```
        printWriter.println("\n");
        printWriter.println(html);
      } //en cualquier otro caso
      else {
        //sirve la página
        html = Paginas.HTML_NO_ENCONTRADO;
        printWriter.println(Mensajes.LINEA_INICIAL_NOT_FOUND);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
        printWriter.println("\n");
        printWriter.println(html);
      }
    } // Fin if-else de peticiones GET
  } //Fin método procesaPeticion

  /**
   * ********************************************************
   * muestra un mensaje en la Salida que confirma el arranque, y da
algunas
   * indicaciones posteriores
   */
  private static void imprimeDisponible() {
    System.out.println("El Servidor WEB se está ejecutando y permanece
a la "
            + "escucha por el puerto 8066.\nEscribe en la barra de
direcciones "
            + "de tu explorador
preferido:\n\nhttp://localhost:8066\npara "
            + "solicitar la página de
bienvenida\n\nhttp://localhost:8066/"
            + "quijote\n para solicitar una página del
Quijote,\n\nhttp://"
            + "localhost:8066/q\n para simular un error");
  } //Fin método imprimeDisponible
}//Fin clase ServidorHttp
```

Mensajes.java

```
/*
 * TAREA PSP05. EJERCICIO 1.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que incluya la cabecera Date.
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package servidorhttp;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */
//Mensajes que intercambia el Servidor con el Cliente según protocolo
HTTP
public class Mensajes {
    /**
```

```
     * @param LINEA_INICIAL_OK  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     * @param LINEA_INICIAL_NOT_FOUND String con modificador final
para
     * asegurar el dato de respuesta inalterable.
     */
    /**
     * En este apartado mantenemos el código ofrecido por el temario
para
     * la resolución del ejercicio.
     * Sólo modifico el nombre de las variables, para acomodarlo a los
     * estándares, pues estaban en minúsculas.
     */
    public static final String LINEA_INICIAL_OK = "HTTP/1.1 200 OK";
    public static final String LINEA_INICIAL_NOT_FOUND =
            "HTTP/1.1 404 Not Found";
  //  public static final String lineaInicial_BadRequest =
  //          "HTTP/1.1 400 Bad Request";
} //Fin clase Mensajes
```

Pagina.java

```
/*
 * TAREA PSP05. EJERCICIO 1.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que incluya la cabecera Date.
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */

package servidorhttp;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */

/**
 *
**********************************************************************
******
 * clase no instanciable donde se definen algunos valores finales
 *
 * Sólo modifico el nombre de las variables, para acomodarlo a los
 * estándares, pues estaban en minúsculas.
 */

public class Paginas {

    /**
     * @param PRIMERA_CABECERA  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     * @param FECHA_CABECERA String con modificador final para
     * asegurar el dato de respuesta inalterable.
     * @param HTML_INDEX String con modificador final para asegurar
     * el dato de respuesta inalterable.
     * @param HTML_QUIJOTE String con modificador final para
     * asegurar el dato de respuesta inalterable.
```

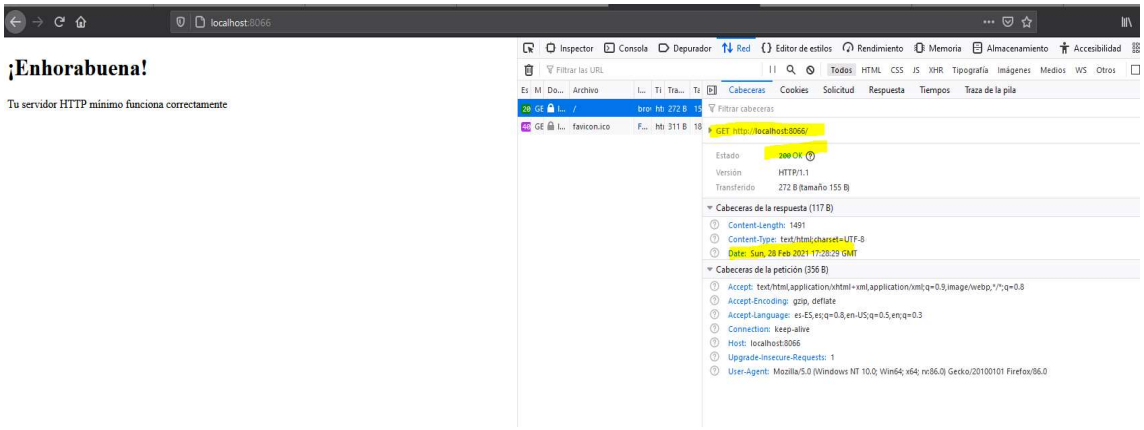
```
     * @param HTML_NO_ENCONTRADO  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     */

    /**
     * Declaración de las variables
     */
    public static final String PRIMERA_CABECERA =
          "Content-Type:text/html;charset=UTF-8";
    /**
     * Añadimos la cabecera Date, como nos han pedido para la tarea.
     */
    public static final String FECHA_CABECERA =
            "Date:" + ServidorHttp.getDateValue();
    //contenido index
    public static final String HTML_INDEX = "<!DOCTYPE html>"
            + "<html>"
            + "<head><title>index</title></head>"
            + "<body>"
            + "<h1>¡Enhorabuena!</h1>"
            + "<p>Tu servidor HTTP mínimo funciona correctamente</p>"
            + "</body>"
            + "</html>";
    //contenido quijote
    public static final String HTML_QUIJOTE = "<!DOCTYPE html>"
            + "<html>"
            + "<head><title>quijote</title></head>"
            + "<body>"
            + "<h1>Así comienza el Quijote</h1>"
            + "<p>En un lugar de la Mancha, de cuyo nombre no quiero "
            + "acordarme, no ha mucho tiempo que vivía un hidalgo de
los "
            + "de lanza en astillero, adarga antigua, rocín flaco y
galgo "
            + "corredor. Una olla de algo más vaca que carnero,
salpicón "
            + "las más noches, duelos y quebrantos (huevos con tocino)
los "
            + "sábados, lentejas los viernes, algún palomino de
añadidura "
            + "los domingos, consumían las tres partes de su hacienda.
El "
            + "resto della concluían sayo de velarte (traje de paño
fino), "
            + "calzas de velludo (terciopelo) para las fiestas con sus
"
            + "pantuflos de lo mismo, y los días de entresemana se
honraba "
            + "con su vellorí (pardo de paño) de lo más fino. Tenía en
su "
            + "casa una ama que pasaba de los cuarenta, y una sobrina
que "
            + "no llegaba a los veinte, y un mozo de campo y plaza,
que "
            + "así ensillaba el rocín como tomaba la podadera...</p>"
            + "</body>"
            + "</html>";
    //contenido noEncontrado
    public static final String HTML_NO_ENCONTRADO = "<!DOCTYPE html>"
            + "<html>"
```

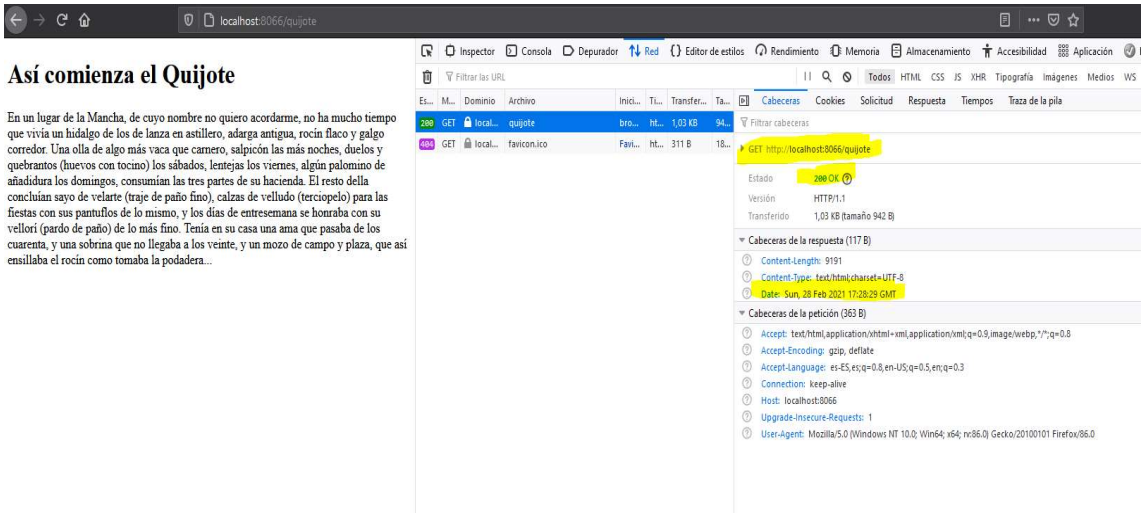
```
            + "<head><title>noEncontrado</title></head>"
            + "<body>"
            + "<h1>¡ERROR! Página no encontrada</h1>"
            + "<p>La página que solicitaste no existe en nuestro "
            + "servidor</p>"
            + "</body>"
            + "</html>";
} //Fin clase Paginas
```

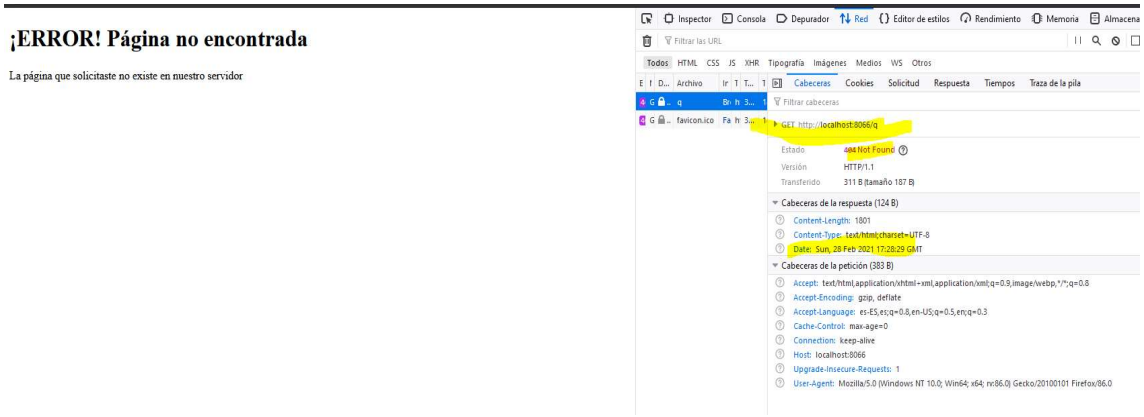
Al ejecutar, la consola nos muestra las distintas respuestas

```
run:
El Servidor WEB se está ejecutando y permanece a la escucha por el puerto 8066.
Escribe en la barra de direcciones de tu explorador preferido:

http://localhost:8066
para solicitar la página de bienvenida

http://localhost:8066/quijote
 para solicitar una página del Quijote,

http://localhost:8066/q
 para simular un error
Atendiendo al cliente
cliente atendido
Atendiendo al cliente
cliente atendido
BUILD STOPPED (total time: 48 seconds)
```

Desde el inspector, podemos ver como el nuevo header se muestra correctamente.







Ejercicio 2.

Modifica el ejemplo del servidor HTTP (Proyecto java **ServerHTTP**, apartado 5.1 de los contenidos) para que implemente multihilo, y pueda gestionar la concurrencia de manera eficiente.

Aprovecho ya las clases creadas para el ejercicio anterior y añadimos una nueva, siguiendo el esquema y pautas del tema en el apartado 5.2 para el manejo de hilos y creamos la clase **`HiloDespachador`**, para el manejo de los threads.

ServidorHttpMultihilo.java

```
/*
 * TAREA PSP05. EJERCICIO 2.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que implemente multihilo, y pueda
gestionar
 * la concurrencia de manera eficiente.
 *
 * Para ello, usaremos la estructura propuesta en el apartado 5.2, que
nos
 * añade un archivo más para manejar los hilos, llamada
HiloDespachador,
 * que será una extensión de la clase Thread de Java, cuyo constructor
 * almacenará el socketCliente que recibe en una variable local
utilizada
 * luego por su método run() para tramitar la respuesta.
 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package servidorhttpmultihilo;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;

/**
```

```java
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */
/**
 * ************************************************************
 * Servidor HTTP que atiende peticiones de tipo 'GET' recibidas por el
puerto
 * 8066
 *
 * NOTA: para probar este código, comprueba primero de que no tienes
ningún otro
 * servicio por el puerto 8066 (por ejemplo, con el comando 'netstat'
si estás
 * utilizando Windows)
 */
public class ServidorHttpMultihilo {

    /**
     * @param socCliente socket de comunicación.
     * @param hilo objeto para el uso de threads
     */

    private static Socket socCliente;
    private static HiloDespachador hilo;
    /**
     ************************************************************
     * procedimiento principal que asigna a cada petición entrante un
socket
     * cliente, por donde se enviará la respuesta una vez procesada
     * @param args the command line arguments
     * @throws IOException, requerida al crear nuestro ServerSocket
     */
    public static void main(String[] args) throws
IOException,Exception {
        try {
            //Asociamos al servidor el puerto 8066
            ServerSocket socServidor = new ServerSocket(8066);
            //El método que escribirá en servidor las instrucciones
            imprimeDisponible();

            //acepta una petición entrante, procesa la petición por el
socket cliente
            //por donde la recibe
            while (true) {
              //a la espera de peticiones
              socCliente = socServidor.accept();
              //atiendo un cliente (AÑADIMOS A LA IMPRESION EL CLIENTE
CONCRETO)
              System.out.println("Atendiendo al cliente " +
socCliente.toString());

              /**
               * Siguiendo la estructura del ejemplo 5.2:
               * crea un nuevo hilo para despacharla por el
socketCliente que
               * le asignó.
               */
              hilo = new HiloDespachador(socCliente);
              hilo.start();
            }//Fin while
```

```
        } catch (IOException e) {
            e.getMessage();
        } //Fin try-catch
    } //Fin clase Main

    /**
     * Método que retorna la fecha y hora actual de un modo apropiado
para
     * usarlo en cabeceras HTTP
     * @return
     */
    public static String getDateValue(){
        DateFormat df = new SimpleDateFormat(
        "EEE, d MMM yyyy HH:mm:ss z", Locale.ENGLISH);
        df.setTimeZone(TimeZone.getTimeZone("GMT"));
        return df.format(new Date());
    }

    /**
   ***************************************
   * procesa la petición recibida
   * Eliminamos la restricción de acceso private para ser visible
desde Hilo
   * @throws IOException
   */
    static void procesaPeticion(Socket socketCliente) throws
IOException {
    /**
     * Variables locales
     * @param peticion String para el BufferReader.
     * @param html String recoge datos desde paginas.
     */
    String peticion;
    String html;

    //Flujo de entrada
    InputStreamReader inSR = new
InputStreamReader(socketCliente.getInputStream());
    //espacio en memoria para la entrada de peticiones
    BufferedReader bufLeer = new BufferedReader(inSR);

    //objeto de java.io que entre otras características, permite
escribir
    //'línea a línea' en un flujo de salida
    PrintWriter printWriter = new
PrintWriter(socketCliente.getOutputStream(), true);

    //mensaje petición cliente
    peticion = bufLeer.readLine();

    //para compactar la petición y facilitar así su análisis,
suprimimos todos
    //los espacios en blanco que contenga
    peticion = peticion.replaceAll(" ", "");

    //si realmente se trata de una petición 'GET' (que es la única que
vamos a
    //implementar en nuestro Servidor)
    if (peticion.startsWith("GET")) {
      //extrae la subcadena entre 'GET' y 'HTTP/1.1'
      peticion = peticion.substring(3, peticion.lastIndexOf("HTTP"));
```

```
      //si corresponde a la página de inicio
      if (peticion.length() == 0 || peticion.equals("/")) {
        //sirve la página
        html = Paginas.HTML_INDEX;
        printWriter.println(Mensajes.LINEA_INICIAL_OK);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
        printWriter.println("\n");
        printWriter.println(html);
      } //si corresponde a la página del Quijote
      else if (peticion.equals("/quijote")) {
        //sirve la página
        html = Paginas.HTML_QUIJOTE;
        printWriter.println(Mensajes.LINEA_INICIAL_OK);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
        printWriter.println("\n");
        printWriter.println(html);
      } //en cualquier otro caso
      else {
        //sirve la página
        html = Paginas.HTML_NO_ENCONTRADO;
        printWriter.println(Mensajes.LINEA_INICIAL_NOT_FOUND);
        printWriter.println(Paginas.PRIMERA_CABECERA);
        /**
         * Incluimos la cabecera date de la tarea.
         */
        printWriter.println(Paginas.FECHA_CABECERA);
        printWriter.println("Content-Length: " + html.length() + 1);
        printWriter.println("\n");
        printWriter.println(html);
      }
    } // Fin if-else de peticiones GET
  } //Fin método procesaPeticion

  /**
   * ************************************************************
   * muestra un mensaje en la Salida que confirma el arranque, y da
algunas
   * indicaciones posteriores
   */
  private static void imprimeDisponible() {

    System.out.println("El Servidor WEB se está ejecutando y permanece
a la "
            + "escucha por el puerto 8066.\nEscribe en la barra de
direcciones "
            + "de tu explorador
preferido:\n\nhttp://localhost:8066\npara "
            + "solicitar la página de
bienvenida\n\nhttp://localhost:8066/"
```

```
            + "quijote\n para solicitar una página del
Quijote,\n\nhttp://"
            + "localhost:8066/q\n para simular un error");
  } //Fin método imprimeDisponible
}//Fin clase ServidorHttpMultihilo
```

HiloDespachador.java

```
/*
 * TAREA PSP05. EJERCICIO 2.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que implemente multihilo, y pueda
gestionar
 * la concurrencia de manera eficiente.
 *
 * Para ello, usaremos la estructura propuesta en el apartado 5.2, que
nos
 * añade un archivo más para manejar los hilos, llamada
HiloDespachador,
 * que será una extensión de la clase Thread de Java, cuyo constructor
 * almacenará el socketCliente que recibe en una variable local
utilizada
 * luego por su método run() para tramitar la respuesta.
 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package servidorhttpmultihilo;

import java.io.IOException;
import java.net.Socket;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */
public class HiloDespachador extends Thread {
    private Socket socketCliente;

    public HiloDespachador(Socket socketCliente) {
        this.socketCliente = socketCliente;
    }
    public void run() {
        /**
         * Aprovechamos la estructura del ejemplo, añadiendo el
         * proceso de atención y cerrado del cliente.
         */

        try{
            //Llamamos al método del servidor para procesar la
petición
            ServidorHttpMultihilo.procesaPeticion(socketCliente);
            //cierra la conexión entrante
            socketCliente.close();
            System.out.println("cliente atendido");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}//Fin clase HiloDespachador
```

Mensajes.java

```
/*
 * TAREA PSP05. EJERCICIO 2.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que implemente multihilo, y pueda
gestionar
 * la concurrencia de manera eficiente.
 *
 * Para ello, usaremos la estructura propuesta en el apartado 5.2, que
nos
 * añade un archivo más para manejar los hilos, llamada
HiloDespachador,
 * que será una extensión de la clase Thread de Java, cuyo constructor
 * almacenará el socketCliente que recibe en una variable local
utilizada
 * luego por su método run() para tramitar la respuesta.
 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */

package servidorhttpmultihilo;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */

//Mensajes que intercambia el Servidor con el Cliente según protocolo
HTTP

public class Mensajes {

    /**
     * @param LINEA_INICIAL_OK  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     * @param LINEA_INICIAL_NOT_FOUND String con modificador final
para
     * asegurar el dato de respuesta inalterable.
     */

    /**
     * En este apartado mantenemos el código ofrecido por el temario
para
     * la resolución del ejercicio.
     * Sólo modifico el nombre de las variables, para acomodarlo a los
     * estándares, pues estaban en minúsculas.
     */

    public static final String LINEA_INICIAL_OK = "HTTP/1.1 200 OK";
    public static final String LINEA_INICIAL_NOT_FOUND =
            "HTTP/1.1 404 Not Found";
  //  public static final String lineaInicial_BadRequest =
  //          "HTTP/1.1 400 Bad Request";
}//Fin clase Mensajes
```

Paginas.java

```
/*
 * TAREA PSP05. EJERCICIO 2.
 * Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP,
apartado
 * 5.1 de los contenidos) para que implemente multihilo, y pueda
gestionar
 * la concurrencia de manera eficiente.
 *
 * Para ello, usaremos la estructura propuesta en el apartado 5.2, que
nos
 * añade un archivo más para manejar los hilos, llamada
HiloDespachador,
 * que será una extensión de la clase Thread de Java, cuyo constructor
 * almacenará el socketCliente que recibe en una variable local
utilizada
 * luego por su método run() para tramitar la respuesta.
 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package servidorhttpmultihilo;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 14/01/2021
 * @version 1
 */

/**
 ****************************************************************
 * clase no instanciable donde se definen algunos valores finales
 *
 * Sólo modifico el nombre de las variables, para acomodarlo a los
 * estándares, pues estaban en minúsculas.
 */

public class Paginas {
    /**
     * @param PRIMERA_CABECERA  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     * @param FECHA_CABECERA String con modificador final para
     * asegurar el dato de respuesta inalterable.
     * @param HTML_INDEX String con modificador final para asegurar
     * el dato de respuesta inalterable.
     * @param HTML_QUIJOTE String con modificador final para
     * asegurar el dato de respuesta inalterable.
     * @param HTML_NO_ENCONTRADO  String con modificador final para
asegurar
     * el dato de respuesta inalterable.
     */

    /**
     * Declaración de las variables
     */
    public static final String PRIMERA_CABECERA =
            "Content-Type:text/html;charset=UTF-8";
    /**
     * Añadimos la cabecera Date, heredada del punto 1 de la tarea.
     */
```

```
    public static final String FECHA_CABECERA =
            "Date:" + ServidorHttpMultihilo.getDateValue();
    //contenido index
    public static final String HTML_INDEX = "<!DOCTYPE html>"
            + "<html>"
            + "<head><title>index</title></head>"
            + "<body>"
            + "<h1>¡Enhorabuena!</h1>"
            + "<p>Tu servidor HTTP mínimo funciona correctamente</p>"
            + "</body>"
            + "</html>";
    //contenido quijote
    public static final String HTML_QUIJOTE = "<!DOCTYPE html>"
            + "<html>"
            + "<head><title>quijote</title></head>"
            + "<body>"
            + "<h1>Así comienza el Quijote</h1>"
            + "<p>En un lugar de la Mancha, de cuyo nombre no quiero "
            + "acordarme, no ha mucho tiempo que vivía un hidalgo de
los "
            + "de lanza en astillero, adarga antigua, rocín flaco y
galgo "
            + "corredor. Una olla de algo más vaca que carnero,
salpicón "
            + "las más noches, duelos y quebrantos (huevos con tocino)
los "
            + "sábados, lentejas los viernes, algún palomino de
añadidura "
            + "los domingos, consumían las tres partes de su hacienda.
El "
            + "resto della concluían sayo de velarte (traje de paño
fino), "
            + "calzas de velludo (terciopelo) para las fiestas con sus
"
            + "pantuflos de lo mismo, y los días de entresemana se
honraba "
            + "con su vellorí (pardo de paño) de lo más fino. Tenía en
su "
            + "casa una ama que pasaba de los cuarenta, y una sobrina
que "
            + "no llegaba a los veinte, y un mozo de campo y plaza,
que "
            + "así ensillaba el rocín como tomaba la podadera...</p>"
            + "</body>"
            + "</html>";
    //contenido noEncontrado
    public static final String HTML_NO_ENCONTRADO = "<!DOCTYPE html>"
            + "<html>"
            + "<head><title>noEncontrado</title></head>"
            + "<body>"
            + "<h1>¡ERROR! Página no encontrada</h1>"
            + "<p>La página que solicitaste no existe en nuestro "
            + "servidor</p>"
            + "</body>"
            + "</html>";
}// Fin clase Paginas
```

Podemos ver que al ejecutar el servidor maneja correctamente hilos de forma simultánea, atendiendo y cerrando cada proceso.