# ESCUELA DE INGENIERÍA DE BILBAO

## EXAMEN ORDINARIO DE GRADO – 31/05/2021

## ASIGNATURA: Sistemas Web

## GRADO: Ingeniería Informática de Gestión y Sistemas de Información

## Nombre y apellidos:

***Notas:***

- *Duración examen: 75 min.*
- *Algunas preguntas de test tienen **más de una** respuesta correcta. Para que la pregunta sea considerada correcta se deben marcar únicamente todas las respuestas correctas.*

**1.** **(0,5)** Cuando se emplea el método *POST* para enviar datos en el protocolo HTTP:

a. Hay que indicar la longitud de la petición en la cabecera *Content-Length*.
b. La respuesta puede tener un código 302/303.
c. Generalmente, los datos se incluyen en el cuerpo del mensaje de la petición.
d. El tamaño de los datos depende de la MTU de la capa inferior, es decir, del tamaño máximo del campo de datos de la capa TCP.
e. Hay que indicar el tipo de los datos que se envían en la cabecera *Content-Type*.

**2.** **(0,5)** Dado el siguiente fichero de despliegue de una aplicación web:

```
<servlet>
      <servlet-name>DoLoginServlet</servlet-name>
      <servlet-class>DoLogin</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>DoLoginServlet</servlet>
      <url-pattern>/do/Login</url-pattern>
<servlet-mapping>
```

¿Cuál de las siguientes URI será una petición correcta al servlet de nombre DoLoginServlet?

a. http://mydomain.com/myapp/doLogin
b. http://mydomain.com/myapp/DoLoginServlet/do/Login
c. http://mydomain.com/myapp/doLogin/DoLoginServlet
d. http://mydomain.com/myapp/DoLoginServlet/DoLogin
e. Ninguna de las anteriores

3. **(0.5)** A través de un formulario embebido dentro de una página HTML, un usuario envía unos datos a una aplicación alojada en Google App Engine. El servidor devuelve el código de respuesta *405 method not allowed*. ¿Dónde radica el problema?
   a. Los nombres de los campos indicados en el formulario no coinciden con los nombres de las variables indicadas en el servidor.
   b. Falta indicar la URI de acceso en el fichero *app.yaml*.
   c. *NullPointerException* a la hora de obtener la cabecera *Method-Type*.
   d. El método indicado en el formulario es *POST*, pero el método de la clase encargada de gestionar dicha petición es *get()*.
   e. El método indicado en el formulario es *GET*, pero el método de la clase encargada de gestionar dicha petición es *post()*.

4. **(0.5)** El objeto **HttpServletRequest**…
   a. Permite transformar un documento XML en una petición HTTP.
   b. Permite realizar peticiones HTTP desde código *javascript*.
   c. Permite realizar peticiones HTTP cuyas respuestas contienen un cuerpo del mensaje que está formateado únicamente en XML.
   d. Es el objeto que se utiliza en los métodos *doGet()* y *doPost()* de Tomcat para referenciar una petición HTTP.
   e. Es el objeto que se utiliza en los métodos *get()* y *post()* de *Google App Engine* para referenciar una petición HTTP.

5. **(0.5)** ¿En OAuth 2.0, a quién identifican las variables *consumer_key* y *consumer_secret*?
   a. Al usuario de la aplicación.
   b. Al programador y a la aplicación.
   c. A la aplicación.
   d. A la instancia del servidor de aplicaciones.
   e. Todas las anteriores.

6. **(0,5)** Para crear una aplicación web en *Google App Engine* que únicamente accede al API de Dropbox… (elige la respuesta más adecuada)
   a. Sólo es necesario registrar la aplicación en Dropbox.
   b. Es necesario crear un proyecto en Google Cloud y registrar la aplicación en Dropbox.
   c. Es necesario crear un proyecto en Google Cloud y unas credenciales para la aplicación tanto en Google como en Dropbox.
   d. Es necesario crear un proyecto en Google Cloud, iniciar App Engine en dicho proyecto y crear unas credenciales para la aplicación en Dropbox.
   e. La *callback_uri* o *redirect_uri* utilizada a la hora de registrar la aplicación en Dropbox debe coincidir con la indicada a la hora de crear las credenciales en Google.

**7.** **(1)** Describe detalladamente la estructura de una petición HTTP y pon un ejemplo.

## FUNCIONAMIENTO DE HTTP: SOLICITUD DEL CLIENTE

**Sintaxis de una solicitud HTTP**

```
Método  URI HTTP/1.1
Cabeceras
CRLF
Cuerpo del mensaje
```

**solicitud HTTP del ejemplo**

```
GET /recurso HTTP/1.1
Host: sw2021.com:8080
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Desktop
```

**Método:** GET
El método describe el tipo de acción CRUD (Create, Read, Update and Delete) que se desea llevar a cabo sobre el recurso. En este caso, GET → Lectura.

**URI:** /recurso
La identificación del recurso puede realizarse con la URI completa o la URI relativa.

GET http://sw2021.com:8080/recurso HTTP/1.1

GET /recurso HTTP/1.1
Host: sw2021.com:8080

**Cabeceras:** caracterizan determinados aspectos del cliente y indican preferencias sobre la respuesta.
*Accept:* el navegador indica que acepta contenido en HTML.
*Accept-Encoding:* el navegador indica que prefiere contenido comprimido (en formato gzip), aunque también acepta contenido no comprimido (identity)
*Accept-Language:* el navegador indica que su preferencia de idioma es el inglés y su segunda opción el castellano.
*User-Agent:* el navegador se identifica como Mozilla sobre una plataforma Windows de escritorio.

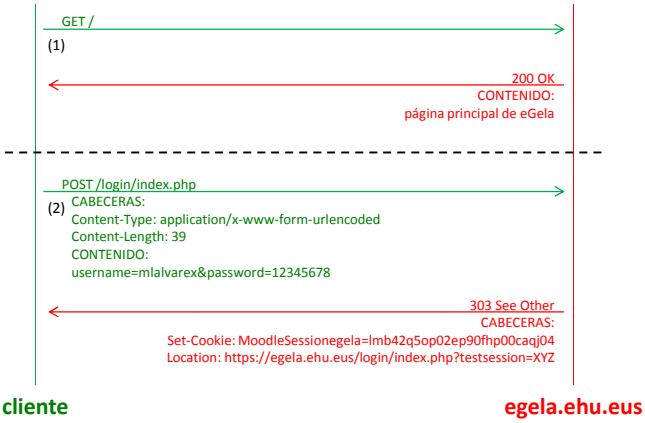
**Cuerpo del mensaje:** en este caso está vacío.

Oskar Casquero y María Luz Álvarez
Dpto. Ingeniería de Sistemas y Automática
2

**8.** **(1)** Si HTTP es un protocolo sin estado. ¿Cómo se mantiene un usuario identificado durante una sesión de navegación?

## EJEMPLO: CONEXIÓN CON EGELA

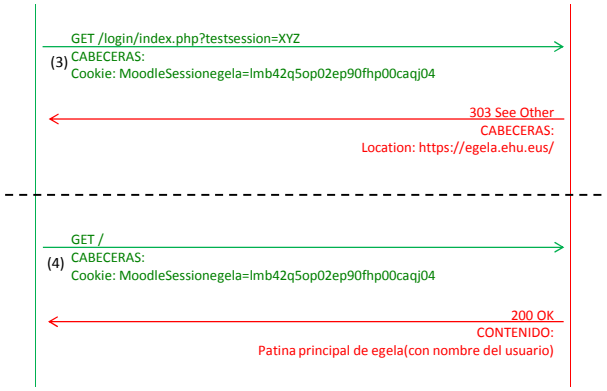
- Pasos (desde el punto de vista del navegador; con F12 en navegador se puede ver):




Oskar Casquero y María Luz Álvarez
Dpto. Ingeniería de Sistemas y Automática

## EJEMPLO:
## CONEXIÓN CON EGELA

- Pasos (desde el punto de vista del navegador; con F12 en navegador se puede ver):




Oskar Casquero y María Luz Álvarez
Dpto. Ingeniería de Sistemas y Automática

9. **(2)** Con la ayuda de la documentación adjunta del API de Gmail, escribe la **petición HTTP** que permite recuperar los mensajes no leídos con remitente sw2021@ehu.es.

HTTP request

```
GET https://www.googleapis.com/gmail/v1/users/userId/messages
```

Parameters

| Parameter name | Value | Description |
|---|---|---|
| **Path parameters** | | |
| `userId` | `string` | The user's email address. The special value **me** can be used to indicate the authenticated user. |
| **Optional query parameters** | | |
| `includeSpamTrash` | `boolean` | Include messages from **SPAM** and **TRASH** in the results. (Default: `false`) |
| `labelIds[]` | `list` | Only return messages with labels that match all of the specified label IDs. |
| `maxResults` | `unsigned integer` | Maximum number of messages to return. |
| `pageToken` | `string` | Page token to retrieve a specific page of results in the list. |
| `q` | `string` | Only return messages matching the specified query. Supports the same query format as the Gmail search box. For example, `"from:someuser@example.com rfc822msgid: <somemsgid@example.com> is:unread"`. Parameter cannot be used when accessing the api using the gmail.metadata scope. |

Authorization

This request requires authorization with at least one of the following scopes (read more about authentication and authorization).

Respuesta:

GET /gmail/v1/users/me/messages?q=sw2019%40gmail.com+is%3Aunread HTTP/1.1
Host: www.googleapis.com
Authorization: Bearer [YOUR_ACCESS_TOKEN]

**10. (1)** Para que una aplicación cree un evento en la cuenta de Google Calendar de un usuario, utilizamos el siguiente trozo de código.

```
scope = 'https://www.googleapis.com/auth/calendar'
calendarioID = "addressbook#contacts@group.v.calendar.google.com"
cabeceras = {}
cabeceras['User-Agent'] = 'Python Client'
# suponer que access_token se ha definido y obtenido correctamente antes
cabeceras['Authorization'] = 'Bearer ' + access_token
cabeceras['Content-Type'] = 'application/json'
url = 'https://www.googleapis.com/calendar/v3/calendars/'+calendarioID+'/events'
cuerpo = { 'end' : {'date':'2021-05-16'},
           'start' : {'date':'2021-05-15'},
           'description': 'Sistemas Web',
           'summary': 'ADIOS' }
cuerpo = json.dumps(cuerpo)
respuesta = requests.post(url, headers=cabeceras, data=cuerpo)
print(respuesta.status_code)
print(respuesta.content)
```

Al ejecutar la aplicación, nos devuelve el error **404 Not Found**. Las causas de esta respuesta pueden ser:

- Hay que indicar la longitud de los datos en la cabecera *Content-Length*.
- El cuerpo del mensaje no está correctamente formateado.
- El usuario no tiene un calendario con ese identificador.
- No se ha definido correctamente la cabecera *Content-Type.*
- Se ha definido mal el *scope* de la aplicación.

**Justifica** porque pueden ser validas o no **cada una de las opciones**.

Respuesta:

**1.- NO:** Con el método POST es necesario definir la *cabecera Content-Length*, y en este código el método requests.post() define la cabecera *Content-Length*.

**2.- NO:** El cuerpo del mensaje se formatea como json en el código.

*cuerpo = json.dumps(cuerpo)*

En caso de no formatear mensaje de error 400 BadRequest.

**3.- SI:** Esta es la causa del error, como el id del calendario aparece en la identificación del recurso, si el calendario no existe, el servidor no puede encontrar el recurso.

url = 'https://www.googleapis.com/calendar/v3/calendars/'+calendarioID+'/events'

**4.- NO**: La cabecera si está definida en el código:

cabeceras['Content-Type'] = 'application/json'

en caso de no estar definida no da ningún error.

**5.- NO:** El **scope es correcto y** permite accede a todo el contenido de calendar

scope = 'https://www.googleapis.com/auth/calendar'

Si el scope no es correcto, el error es: 403 Forbidden.

**11. (2)** Completa el siguiente diagrama de secuencia.