

ejercicios-tema2-resueltos.pdf



VELARIS



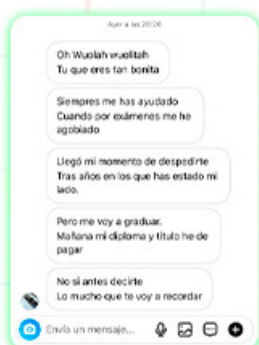
Sistemas Informaticos I



3º Grado en Ingeniería Informática



Escuela Politécnica Superior
Universidad Autónoma de Madrid



**Que no te escriban poemas de amor
cuando terminen la carrera**



*(a nosotros por
suerte nos pasa)*

WUOLAH

Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶▶▶



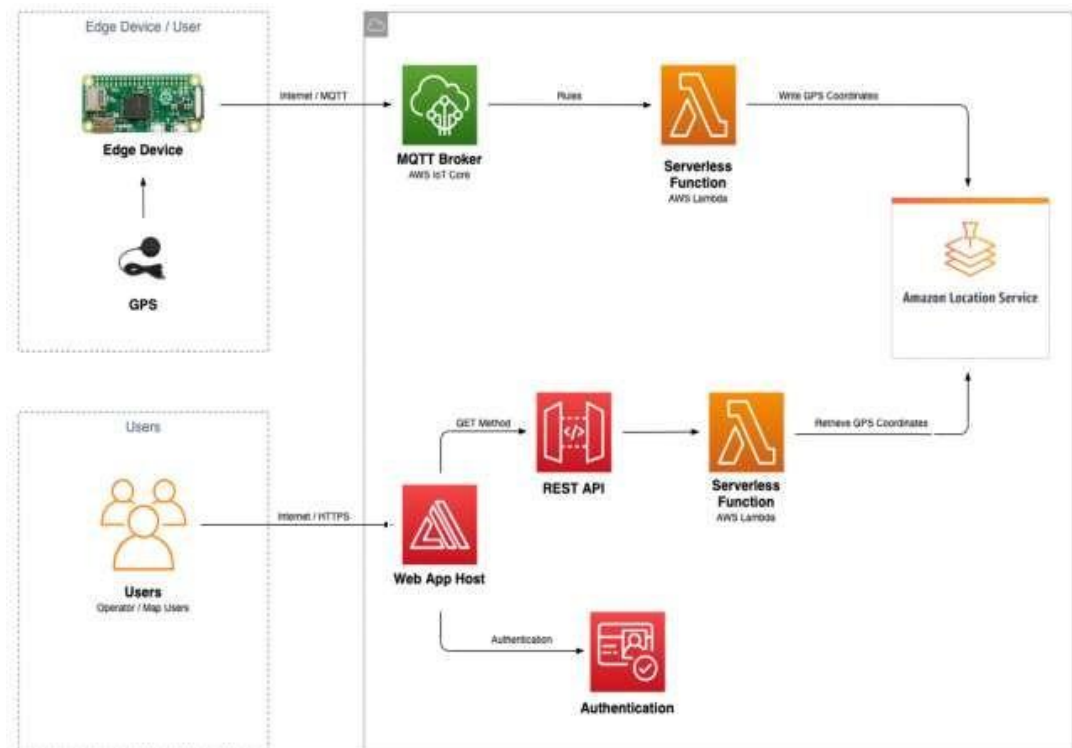
WUOLAH

(a nosotros por suerte nos pasa)

Sistemas Informáticos I

Tema 2: Hoja de ejercicios

Ejercicio 1. Describir y explicar brevemente la arquitectura que se muestra en la siguiente figura (adaptada de AWS) planteada como solución para un sistema web cuyo objetivo principal es mostrar la información de geolocalización de las rutas realizadas por una serie de vehículos en las últimas 24 horas, teniendo en cuenta que MQTT (*Message Queing Telemetry Transport*) es un protocolo de comunicación "machine to machine" sobre la pila TCP/IP que se basa en un servicio de mensajería push que sigue un esquema publicador-suscriptor.



Explicacion de la arquitectura

Componentes de la arquitectura

- Los equipos de GPS instalados en los camiones
- Los usuarios y operadores del sistema que ven geolocalizados en un mapa la posición en tiempo real de los camiones
- Los servicios en la nube basados en Amazon Web Services (AWS)

Como funcionan los servicios en la nube

- Las coordenadas en tiempo real son transmitidas al broker utilizando el protocolo MQTT a través de Internet
- El broker ejecuta una función Lambda (serverless) que graba las coordenadas GPS y los datos de cada camion en el servicio Amazon Location Service
- Los operadores de los mapas (que pueden ver la localización en tiempo real de cada camion), acceden con sus credenciales a un servidor web (Web App Host) usando el protocolo seguro HTTPS

- La aplicación web del servidor consume un servicio RESTful
- El servicio web ejecuta una función Lambda (serverless) que se trae los datos necesarios para posicionar los camiones en los mapas desde Amazon Location Service

Amazon Lambda es un servicio de computación sin servidor (serverless) de Amazon Web Services (AWS). Con Amazon Lambda, los desarrolladores pueden ejecutar código sin tener que provisionar o administrar servidores. En su lugar, los desarrolladores simplemente suben su código a Lambda y el servicio se encarga de ejecutarlo y escalarlo automáticamente. Es una herramienta muy flexible ya que te permite ejecutar código solo cuando es necesario, lo que reduce los costos y aumenta la escalabilidad.

El esquema publicador-suscriptor es un patrón de arquitectura de software en el cual los componentes de un sistema (llamados "publicadores") emiten eventos o mensajes a través de un canal común, y los componentes (llamados "suscriptores") se suscriben a esos eventos o mensajes para recibirlos y realizar alguna acción.

En este esquema, los publicadores no tienen conocimiento de quién está suscrito a sus eventos, y los suscriptores no tienen conocimiento de quiénes son los publicadores. Esto permite que los componentes se comuniquen de manera desacoplada, lo que facilita el desarrollo, el mantenimiento y la escalabilidad del sistema.

Este patrón se utiliza en diferentes tipos de sistemas, desde sistemas de mensajería asíncrona hasta sistemas de procesamiento de eventos en tiempo real.

Ejercicio 2. Durante la revisión de la arquitectura de un sistema distribuido con una elevada latencia entre su arquitectura web, desplegada en un proveedor cloud, y la base de datos de la que se alimenta, ubicada en un datacenter local (in-site), se identifica que el motor de búsqueda de la base de datos local satura constantemente debido a un elevado número de consultas. Se estima que la distancia entre los servicios cloud y el datacenter también podría suponer un problema a futuro. Proponer una arquitectura escalable en tamaño y geográficamente, teniendo en cuenta que los servicios que dan acceso a los datos están desarrollados en django y que los usuarios encargados del mantenimiento de los datos tienen acceso directo a la red del datacenter.

Detalles del problema

- Elevada latencia entre la arquitectura web y la base de datos (tarda mucho en responder o tardan mucho los datos en llegar desde la base de datos)
- La arquitectura web está en la nube
- La base de datos está en un datacenter local
- El motor de búsqueda satura por el elevado número de consultas

Arquitectura propuesta

- Mover las bases de datos locales a la nube, si es posible al mismo proveedor de servicios donde se encuentra la arquitectura web. La nube siempre va a ser escalable en tamaño y geográficamente, mucho más que si movemos la aplicación web a nuestro datacenter y nos ocupamos de provisionar, mantener y actualizar los servidores.
- Mover la capa de acceso a los datos (servicios que permiten a la nube llegar a los datos en el datacenter) a la nube igualmente. Tener en cuenta que los servidores donde se van a desplegar estos servicios van a utilizar Python como lenguaje de servidor, ya que Django es un framework desarrollado en Python
- Si la latencia siguiera siendo elevada, sustituir la capa de servicios por una capa más ligera (en Python o cualquier lenguaje soportado) incluyendo un broker de servicios que permita

**Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶**
(a nosotros por suerte nos pasa) 😊



WUOLAH



- seguridad, cache y mejor rendimiento.
- Si la latencia siguiera siendo elevada, eliminar la capa de acceso a los datos permitiendo que la arquitectura web accediera directamente a los datos

Ejercicio 3. Explicar de forma razonada si las siguientes afirmaciones son o no correctas:

- a) Los sistemas web que se ejecutan en Internet deben seguir un modelo de arquitectura en tres capas, mientras que las que lo hacen en una intranet, también pueden seguir un modelo en dos capas siempre que las restricciones de seguridad de la web lo permitan.

No es correcto. Todas las aplicaciones web deberán seguir un modelo de tres capas (presentación, negocio o datos) no importa si se ejecutan en un entorno de Intranet (dentro de la empresa) o en Internet. El modelo de tres capas es más fácil de desarrollar, mantener y escalar que un modelo de dos capas (presentación, negocio/datos) donde el acceso a datos por ejemplo se mezcle o se confunda con las funciones de negocio que ofrece el sistema web.

- b) Una aplicación web tradicional diseñada para una intranet es más eficiente en términos de tiempo de respuesta, ancho de banda y consumo de recursos, que una aplicación equivalente diseñada para Internet.

No necesariamente. Una aplicación web tanto en Internet como en Intranet va a depender de los recursos que le sean asignados para su correcto funcionamiento. Si esos recursos son capaces de escalar horizontalmente en función de la demanda mucho mejor. Esto es algo común en los servicios de la nube y poco común en las aplicaciones de Intranet.

Si yo despliegó una aplicación en Azure o AWS con recursos, podría ir mucho mejor que una aplicación de Intranet con una red de poco ancho de banda o con servidores antiguos de hace 20 años.

- c) Las aplicaciones web que se ejecutan en Internet son sistemas distribuidos con un cliente ligero (el navegador).

Es correcto. Son sistemas distribuidos porque generalmente no están ejecutándose en nuestro ordenador y utilizan como capa de presentación (interacción con el usuario) un navegador web.

Un sistema distribuido es un conjunto de componentes independientes y autónomos que trabajan juntos para lograr un objetivo común. Estos componentes se comunican y colaboran entre sí mediante una red de comunicaciones, y pueden estar físicamente ubicados en diferentes lugares.

Un sistema distribuido puede ser construido utilizando diferentes arquitecturas, como una arquitectura cliente-servidor o una arquitectura basada en servicios.

Los sistemas distribuidos tienen varias ventajas, incluyendo escalabilidad, tolerancia a fallas y mayor rendimiento, ya que pueden repartir la carga de trabajo entre varios componentes.

- d) Las aplicaciones web diseñadas para una intranet son sistemas centralizados cuyos componentes no pueden ejecutarse en máquinas que se encuentren fuera del hub central de máquinas interconectadas que definen y delimitan las reglas de seguridad de la red.

Incorrecto. Las aplicaciones web de Intranet suelen ser sistemas distribuidos, por ejemplo utilizan por separado servidores web, servidores de archivo y servidores de base de datos.

Un sistema centralizado es un sistema en el cual toda la información y todas las decisiones son controladas y gestionadas por una sola entidad central, en lugar de ser repartidas entre varios componentes o entidades.

Sistemas centralizados tienen algunas ventajas, como la simplicidad en su diseño y su fácil manejo, ya que solo existe un ente controlador. Sin embargo, tiene desventajas, como los puntos ciegos, un único punto de falla, si este se cae todo el sistema se para, y menor escalabilidad, ya que todas las peticiones y datos deben pasar por el sistema central. Por lo general al ser un único controlador escalan verticalmente (añadiendo más CPU, más memoria o más disco).

- e) Por simplicidad y homogeneidad, los sistemas Web han reemplazado a todos los otros sistemas distribuidos, encapsulando sus servicios.

Incorrecto. No todos los sistemas distribuidos están basados en servidores con protocolo HTTP. Existen otros sistemas que se usan en aplicaciones militares o industriales con arquitectura cliente-servidor y componentes separados que se comunican entre sí utilizando otros protocolos que no son HTTP.

Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

Ejercicio 4. Dado el contenido de los siguientes ficheros, asumiendo un procesamiento secuencial en el que primero se construye el HTML DOM completo y luego se muestra en pantalla, y teniendo en cuenta que la cache del navegador está activada, aunque inicialmente vacía, dibujar el diagrama de secuencia que represente el posible intercambio de mensajes entre el cliente y el servidor web cuando el primero solicita por GET el documento *home.html* al segundo:

home.html

```
<html><head><meta http-equiv="Content-Type" content="text/html">
  <link rel="stylesheet" type="text/css" href="css.css">
</head>

<body>
  <iframe src="cabecera.html" frameborder="0" scrolling="no" framespacing="0"
border="0" marginwidth="0" marginheight="0"></iframe>

  <div id="contenedorContenido">
    <div id="main">
      <div class="titulo">
        <div class="texto">
          <center>
            Contenido
          </center>
        </div>
      </div>
      
    </div>
  </body></html>
```

cabecera.html

```
<html><head><meta http-equiv="Content-Type" content="text/html">
  <link rel="stylesheet" type="text/css" href="css.css">
</head>

<body>
  <div id="header_tall">
    
  </div>
</body></html>
```

CSS.CSS

```
iframe { border:none; width:100%; }
#header_tall {background:url(header_tall.gif) top repeat-x; padding-top: 20px;}
#contenedorContenido .titulo {background:url(neurona.jpg) top left no-repeat; }
```









Diagrama de secuencia (secuencia real en navegador Google Chrome)

Name	Status	Type	Initiator	Size	Time
home.html	200	document	Other	601 B	2 ms
css.css	200	stylesheet	home.html	213 B	9 ms
neurociencia.jpg	200	jpeg	home.html	101 kB	13 ms
cabecera.html	200	document	home.html	251 B	3 ms
neurona.jpg	200	jpeg	css.css	316 kB	8 ms
css.css	200	stylesheet	cabecera.html	213 B	3 ms
logoUAM.png	200	png	cabecera.html	374 kB	6 ms
header_tall.gif	200	gif	css.css	1.8 kB	2 ms

En la medida en que se va cargando el código de la página se van haciendo las

peticiones GET al servidor. Una vez que el recurso es descargado en el navegador, si el cache esta activado se queda en cache, de modo que cuando la pagina lo vuelve a solicitar el tiempo de acceso o descarga es menor.

Esto se ve facilmente si volvemos a cargar la misma pagina:

Name	Status	Type	Initiator	Size	Time
 home.html	304	document	Other	296 B	4 ms
 css.css	304	stylesheet	home.html	295 B	4 ms
 neurociencia.jpg	304	jpeg	home.html	298 B	3 ms
 neurona.jpg	304	jpeg	css.css	298 B	4 ms
 cabecera.html	304	document	home.html	296 B	5 ms
 css.css	304	stylesheet	cabecera.html	295 B	3 ms
 logoUAM.png	304	png	cabecera.html	298 B	4 ms
 header_tall.gif	304	gif	css.css	296 B	3 ms

En estos casos el codigo de status es 304, indicando que la pagina no ha sufrido modificaciones desde la ultima peticion y que el navegador ha guardado ese contenido en cache.

- 1) Descarga home.html
- 2) Al renderizar la pagina home.html se da cuenta que necesita la hoja de estilo y descarga a css.css
- 3) Igualmente se da cuenta que necesita la imagen neurociencia.jpg que esta en home.html y la descarga
- 4) Para poder pintar el iframe necesita descargar la pagina cabecera.html
- 5) Descarga a neurona.jpg porque la necesita como fondo del contendorContenido de la pagina home.html en el momento de aplicar la hoja de estilo
- 6) Al renderizar cabecera.html en el iframe, necesita de la hoja de estilo asi que la descarga
- 7) Descarga la imagen logoUAM.png como parte de la pagina cabecera.html
- 8) Al aplicar los estilos a la pagina cabecera.html descarga la imagen gif

Ejercicio 5. Dado el código HTML que se muestra a continuación, indicar de qué color se mostrará cada uno de los párrafos que aparecen en el código:

```
<!DOCTYPE html>
<html>
<head>
<style>
/* VER DETALLES EN https://www.w3schools.com/cssref/css_selectors.php */
div {
    color: green; /* Texto en verde para todos los div */
}
p {
    color: blue; /* Texto azul para todos los parrafos */
}
div ~ p {
    color: red; /* Texto rojo para todo parrafo precedido por un div */
}
div > p {
    color: magenta; /* Texto magenta para todo parrafo cuyo padre es un div */
}
```

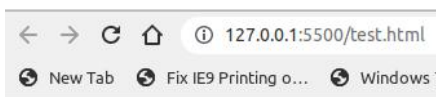


```

    div + p {
        color: gray; /* Texto gris para el primer elemento p que aparece
                       inmediatamente despues de un elemento div */
    }
</style>
</head>
<body>
<!-- Este texto sale en verde (los div no se afectan con nada)-->
<div>Párrafo 1</div>
    <!-- Este texto sale en gris (el parrafo esta justo despues de un div)-->
    <p>Párrafo 2</p>
    <div>
        <!-- Este texto sale en verde (los div no se afectan con nada)-->
        <div>Párrafo 3</div>
        <!-- Este texto sale en gris (el parrafo esta justo despues de un div)-->
        <p>Párrafo 4</p>
        <div>
            <!-- Este texto sale en magenta (p es hijo de un div)-->
            <p>Párrafo 5</p>
            <div>
                <!-- Este texto sale en magenta (p es hijo de un div)-->
                <p>Párrafo 6</p>
            </div>
            <!-- Este texto sale en gris (el parrafo esta justo despues de un div)-->
            <p>Párrafo 7</p>
        </div>
    </div>
<!-- Este texto sale en gris (el parrafo esta justo despues de un div)-->
<p>Párrafo 8</p>
<!-- Este texto sale en rojo (p tiene un div por delante)-->
<div><p>Párrafo 9</p></div>
</body>
</html>

```

Salida en el navegador



Párrafo 1
 Párrafo 2
 Párrafo 3
 Párrafo 4
 Párrafo 5
 Párrafo 6
 Párrafo 7
 Párrafo 8
 Párrafo 9

Ejercicio 6. Realizar un documento HTML con un formulario como el de la figura:

Nombre:

Edad:

E-mail:

No se enviarán datos al servidor hasta que:

El campo para el nombre tenga un nombre válido de al menos 2 caracteres, no permitiéndose caracteres en blanco ni al principio ni al final.

El campo edad tenga una edad válida.

El campo e-mail tenga una dirección válida (al menos contenga un carácter "@")

Las validaciones se tienen que hacer de forma programática con *JavaScript*, no mediante validaciones en HTML 5

El formulario se enviará para su procesamiento a uno de los siguientes servicios <https://postman-echo.com/get> o <https://postman-echo.com/post>, el primero acepta peticiones en modo GET y el segundo en modo POST. Comparar el resultado obtenido en ambos casos, explicando la diferencia entre ambos métodos de envío.

Solucion con POST

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Formulario</title>
</head>
<body>
<form name="form1" onsubmit="return validarFormulario()" method="post"
  action="https://postman-echo.com/post">
  <p id="error"></p>
  <p>
    <label for="nombre">Nombre:</label><input type="text" id="nombre" name="nombre" />
  </p>
  <p>
    <label for="edad">Edad:</label> <input type="text" id="edad" name="edad" />
  </p>
  <p>
    <label for="email">Email:</label> <input type="text" id="email" name="email" />
  </p>
  <p><input type="submit" value="Enviar consulta" /></p>
</form>
<script>
function validarFormulario() {
// Parrafo donde se muestran los errores
let error = document.getElementById("error");
// Valor del campo nombre del formulario form1
```

Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolilah
Tu que eres tan bonita

```
let nombre = document.forms["form1"]["nombre"].value;
// Con trim() eliminamos caracteres en blanco delante y detras
if (nombre.trim().length < 2) {
    error.innerText = "El nombre no puede tener menos de 2 caracteres";
    return false;
}
// Valor del campo edad del formulario form1
let edad = document.forms["form1"]["edad"].value;
// /^ Comienzo de expresion regular
// Secuencia de al menos 1 digito
// $/ Fin de la expresion regular
let isnum = /^\d+$/.test(edad);
if (!isnum) {
    error.innerText = "La edad tiene que ser un valor numerico";
    return false;
}
edad = parseInt(edad); // Convertimos la edad en entero
if (edad < 1 || edad > 120) {
    error.innerText = "La edad no puede ser menor de 1 año ni mayor que 120";
    return false;
}
// Ver una validacion mas completa en:
// https://www.w3resource.com/javascript/form/email-validation.php
// Valor del campo email del formulario form1
let email = document.forms["form1"]["email"].value;
// /^ Comienzo de expresion regular
// Secuencia de al menos 1 caracter menos @ [^@]+
// Un @
// Secuencia de al menos 1 caracter menos @ [^@]+
// $/ Fin de la expresion regular
let isemail = /^[^@]+@[^@]+$/.test(email);
if (!isemail) {
    error.innerText = "El email no tiene un formato valido";
    return false;
}
return true;
</script>
</body>
</html>
```

La salida en el navegador es la siguiente:

← → ↺ 🏠 postman-echo.com/post

🔍 New Tab 📖 Fix IE9 Printing o... 🖥 Windows 7 Load... 📄 Descarga Office... 📧 megapost de enc... 📄 Papiroflexia: Dib... 🖨 System Ac

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "nombre": "juan perez",
    "edad": "21",
    "email": "juan.perez@gmail.com"
  },
  "headers": {
    "x-forwarded-proto": "https",
    "x-forwarded-port": "443",
    "host": "postman-echo.com",
    "x-amzn-trace-id": "Root=1-63bf9ba0-5ef6fed946ed6b8279e28df6",
    "content-length": "54",
    "cache-control": "max-age=0",
    "sec-ch-ua": "\"Not?A_Brand\";v=\"8\"",
    "Chromium": "v=\"108\"",
    "Google Chrome": "v=\"108\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "Linux",
    "upgrade-insecure-requests": "1",
    "origin": "http://127.0.0.1:5500",
    "content-type": "application/x-www-form-urlencoded",
    "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
    "sec-fetch-site": "cross-site",
    "sec-fetch-mode": "navigate",
    "sec-fetch-user": "?1",
    "sec-fetch-dest": "document",
    "referrer": "http://127.0.0.1:5500/",
    "accept-encoding": "gzip, deflate, br",
    "accept-language": "es-ES,es;q=0.9",
    "json": {
      "nombre": "juan perez",
      "edad": "21",
      "email": "juan.perez@gmail.com"
    },
    "url": "https://postman-echo.com/post"
  }
}
```

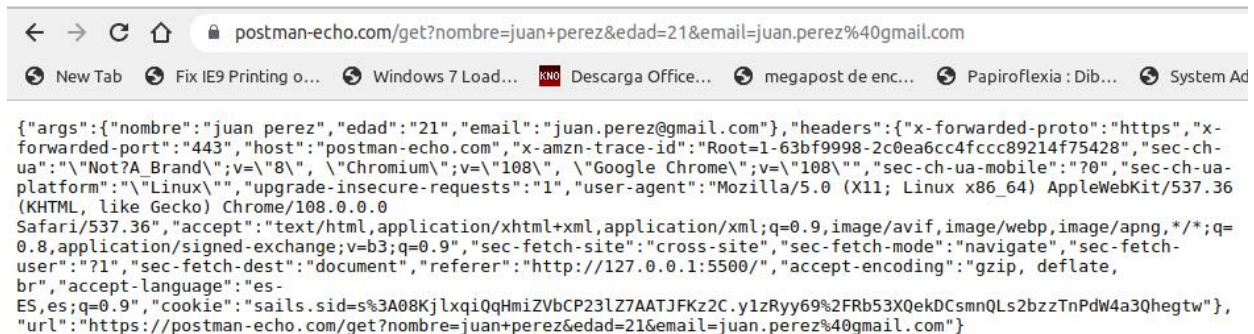
WUOLAH

Solucion con GET

Realizamos el siguiente cambio:

```
<form name="form1" onsubmit="return validarFormulario()" method="get"
  action="https://postman-echo.com/get">
```

La diferencia es que los campos del formulario pueden verse en la caja de URL del navegador (lo cual sería inapropiado para un formulario de login, por ejemplo):



```
{
  "args": {
    "nombre": "juan perez",
    "edad": "21",
    "email": "juan.perez@gmail.com"
  },
  "headers": {
    "x-forwarded-proto": "https",
    "x-forwarded-port": "443",
    "host": "postman-echo.com",
    "x-amzn-trace-id": "Root=1-63bf9998-2c0ea6cc4fccc89214f75428",
    "sec-ch-ua": "\"Not?A_Brand\";v=\\\"8\\\", \"Chromium\";v=\\\"108\\\", \"Google Chrome\";v=\\\"108\\\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"Linux\"",
    "upgrade-insecure-requests": "1",
    "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
    "sec-fetch-site": "cross-site",
    "sec-fetch-mode": "navigate",
    "sec-fetch-user": "?1",
    "sec-fetch-dest": "document",
    "referrer": "http://127.0.0.1:5500/",
    "accept-encoding": "gzip, deflate, br",
    "accept-language": "es-ES,es;q=0.9",
    "cookie": "sails.sid=s%3A08KjlxqiQqHmiZVbCP23lZ7AATJFKz2C.y1zRyy69%2FRb53XQekDCsmnQLs2bzzTnPdW4a3Qhegtw"
  },
  "url": "https://postman-echo.com/get?nombre=juan+perez&edad=21&email=juan.perez%40gmail.com"
}
```

Ejercicio 7. Dado el siguiente código HTML. Indica que se vería en el navegador tras cargarse la página completamente.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>

$( document ).ready(function() {

    $(':input[required]').siblings('input').addClass('backgroundMarker').
    append($('<span>').text('*')).
    addClass('requiredMarker');

});

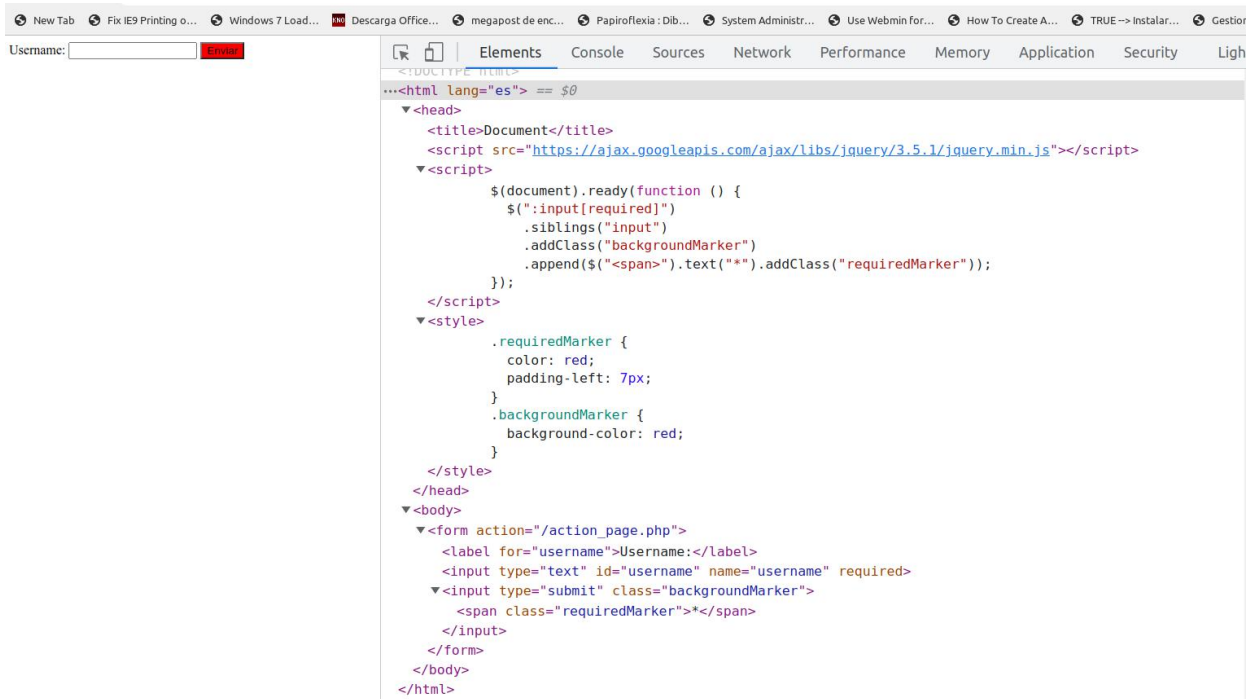
</script>

<style>
.requiredMarker {
color: red;
padding-left:7px;
}
.backgroundMarker {
background-color: red;
}
</style>

<form action="/action_page.php">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
  <input type="submit">
</form>

</body>
</html>
```

El resultado de cargar esta pagina en el navegador es el siguiente:



Una vez que la pagina se ha cargado, se ejecuta el script que hace lo siguiente:

- Filtra los elementos de tipo input con atributo required
- Busca los hermanos o nodos adyacentes con siblings que sean de tipo input para seleccionar el boton de submit
- Añade al input de tipo submit la clase backgroundMarker que lo pinta con color de fondo rojo
- Añade a este input una etiqueta span con un "*" para indicar que rellenar el campo en este formulario es obligatorio(ERROR: el nodo se inserta en el DOM dentro de la etiqueta input que es un elemento de tipo vacio - no tiene apertura y cierre - y por lo tanto no vemos nada).

Lo correcto seria:

```
<script>
$(document).ready(function () {
  $(":input[required]")
    .before($("<span>").text("*").addClass("requiredMarker"))
    .siblings("input")
    .addClass("backgroundMarker");
});
</script>
```

El resultado al cargar la pagina seria:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Document</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script>
      $(document).ready(function () {
        $(":input[required]")
          .before($("<span>").text("*").addClass("requiredMarker"))
          .siblings("input")
            .addClass("backgroundMarker");
      });
    </script>
    <style>
      .requiredMarker {
        color: red;
        padding-left: 7px;
      }
      .backgroundMarker {
        background-color: red;
      }
    </style>
  </head>
  <body>
    <form action="/action_page.php">
      <label for="username">Username:</label>
      <span class="requiredMarker">*</span>
      <input type="text" id="username" name="username" required>
      <input type="submit" class="backgroundMarker">
    </form>
  </body>
</html>

```

Ahora el script hace lo siguiente:

- Filtra los elementos de tipo input con atributo required
- Como solo va a encontrar uno, delante inserta el elemento span al que añade la clase requiredMarker que pinta el asterisco en color rojo
- Busca los hermanos o nodos adyacentes con siblings que sean de tipo input para seleccionar el boton de submit
- Añade la clase backgroundMarker que pinta el color de fondo del boton en rojo

Ejercicio 8. En el contexto de un sistema distribuido basado en la WWW, cuál de las siguientes tecnologías relacionarías en exclusiva con el lado del cliente, cuál en exclusiva con el lado del servidor y cuál con ambos.

AJAX	<input checked="" type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
CGI	<input type="checkbox"/>	CLIENTE	<input checked="" type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
CSS	<input checked="" type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
DHTML	<input checked="" type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
HTML 5	<input checked="" type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
JavaScript	<input type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input checked="" type="checkbox"/>	AMBOS
JSON	<input type="checkbox"/>	CLIENTE	<input type="checkbox"/>	SERVIDOR	<input checked="" type="checkbox"/>	AMBOS
PHP	<input type="checkbox"/>	CLIENTE	<input checked="" type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
Python	<input type="checkbox"/>	CLIENTE	<input checked="" type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS
SQL	<input type="checkbox"/>	CLIENTE	<input checked="" type="checkbox"/>	SERVIDOR	<input type="checkbox"/>	AMBOS

Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶▶▶



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempre me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolilah
Tu que eres tan bonita

WSGI [] CLIENTE [X] SERVIDOR [] AMBOS
XML [] CLIENTE [] SERVIDOR [X] AMBOS

Ejercicio 9. Utilizando AJAX, JQuery y Flask, se desea construir una aplicación web que pida al usuario que introduzca dos números, los mande al servidor para que los sume y actualice de forma asíncrona el documento HTML en el cliente con el resultado de la suma. El resultado de la suma debe situarse, utilizando JQuery, después de los dos puntos de la cadena "Resultado:" situada debajo del formulario.

Dado el código suministrado a continuación, completa los espacios marcados como 1, 2 y 3 para que la aplicación satisfaga los requisitos especificados.

```
<!DOCTYPE html>
<html>
<head>
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
</script>

<script>
  function cargarRespuesta() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {

      1

    }

  }
</script>
</head>

<body>
  <form name="Formulario">
    Valor 1: <input type="text" id="valor1" size=1>
    Valor 2: <input type="text" id="valor2" size=1>
    <button type="button" onclick="cargarRespuesta()">Calcular</button>
  </form>
  <div id="resultado">Resultado: </div>
</body>
</html>
```

```
from flask import Flask
```

```
app = Flask(__name__)
```

3

```
if __name__ == '__main__':
    app.run('127.0.0.1', 8800, debug=True)
```

WUOLAH

CODIGO HTML

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
function cargarRespuesta() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        // BLOQUE 1 -----
        // readState == 4 Es que la operacion se ha completado
        // status == 200 La operacion se ha completado correctamente
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            $("#resultado").text("Resultado: " + xhttp.responseText);
        }
        //-----
    };
    // BLOQUE 2 -----
    let params = "valor1=" + document.forms["Formulario"]["valor1"].value +
        "&valor2=" + document.forms["Formulario"]["valor2"].value;
    xhttp.open("GET", "http://127.0.0.1:8800?" + params, true);
    xhttp.send();
    // -----
}
</script>
</head>
<body>
    <form name="Formulario">
        Valor 1: <input type="text" id="valor1" size="1" />
        Valor 2: <input type="text" id="valor2" size="1" />
        <button type="button" onclick="cargarRespuesta()">Calcular</button>
    </form>
    <div id="resultado">Resultado:</div>
</body>
</html>
```

CODIGO PYTHON

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from flask import Flask
app = Flask(__name__)
# Bloque 3 -----
import flask
from flask import request

@app.route('/', methods=['GET'])
def calcular():
    valor1 = request.args.get("valor1")
    valor2 = request.args.get("valor2")
    result = str(int(valor1)+int(valor2))
    resp = flask.Response(result)
    resp.headers['Access-Control-Allow-Origin'] = '*'
```

```
    return resp
#-----

if __name__ == "__main__":
    app.run('127.0.0.1', 8800, debug=True)
```