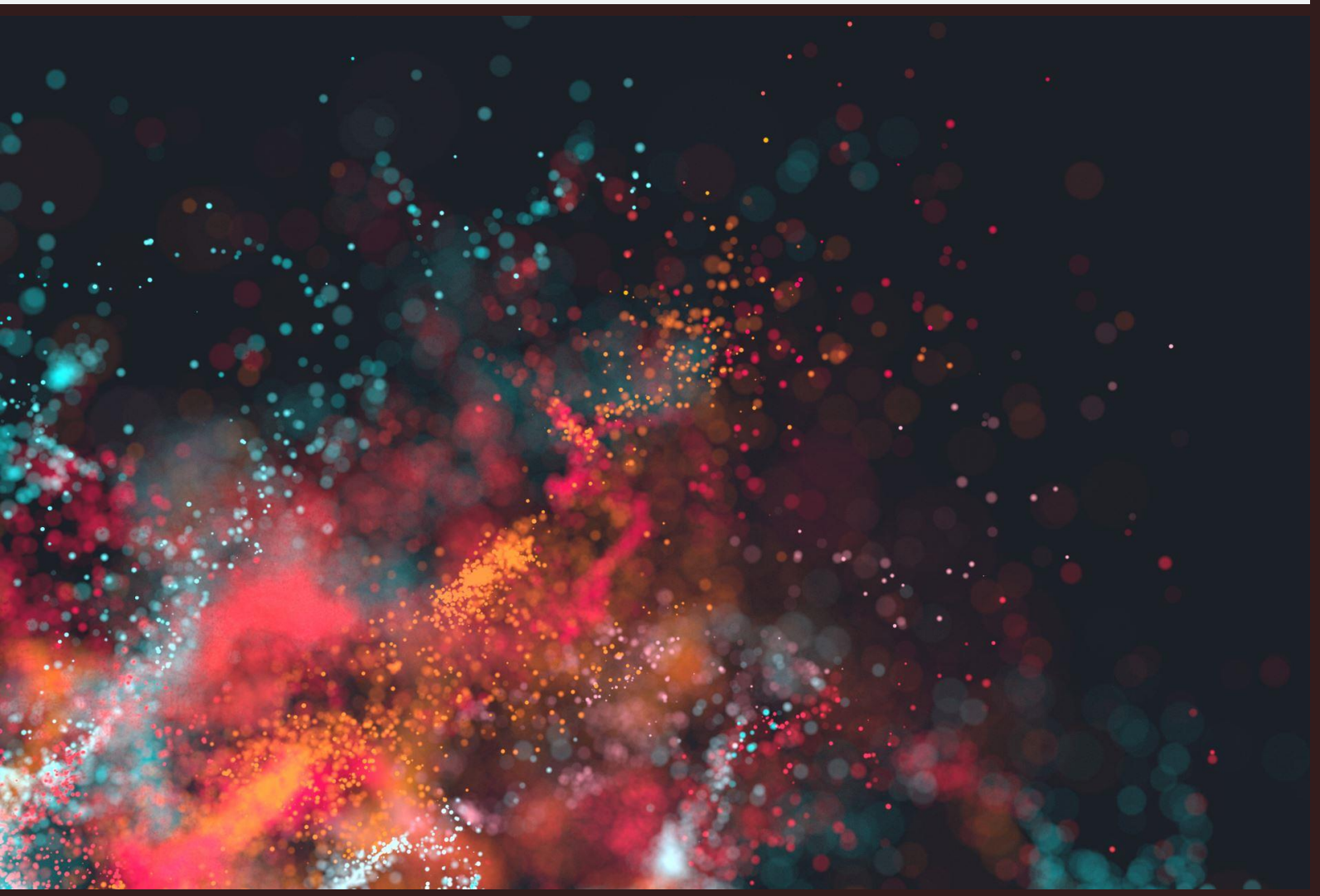


TEMA 3.7

Redireccionamiento y autenticación

REDIRECCIÓN

header - return

Redireccionamiento

- Uno de los usos del protocolo **HTTP** es indicarle al navegador que una página se ha mudado de localización.
- Se hace mediante un **response**, que le indica al navegador lo siguiente:
  - No te traigo ninguna página para que cargues
  - Todo ha ido bien, es que la página que pides me indica, que debes cargar otra página distinta
  - Aquí te indico cuál es la nueva página: Está en la siguiente **localización.**
- El navegador lee este response, y genera un request nuevo de tipo GET para solicitar la página indicada en el **Location**
- El **Location** puede ser una URL con parámetros GET

Estados de error del navegador

Cuando el navegador hace un request HTTP, puede recibir en el response los siguientes estados (entre otros):

- **200:** Significa que se ha encontrado la página y todo ha ido bien. El navegador cargará la página que viene en el response.
- **404:** Significa que no se ha encontrado la página. El navegador mostrará un mensaje de error indicando que no se ha encontrado la página
- **302**: La página ha cambiado de localización. Es lo que se conoce como "redireccionamiento". El navegador generará un nuevo request (GET) para navegar a la nueva localización.

La organización de los códigos de error puedes consultarla en la siguiente página:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Redirección con PHP: header("Location: …")

- En PHP podemos generar una redirección como respuesta al request de nuestra página.
- Para poder hacerlo, la **condición** es que no se haya enviado ninguna salida. Ni un doctype, ni html, ni siquiera un espacio en blanco o un enter.
- Se utilizará la función header, seguida de un return.
- Puedes ver el uso de esta función en el manual de PHP. Como parámetro podemos indicar un Location. Esto generará una redirección a esa localización
- Un ejemplo sería:

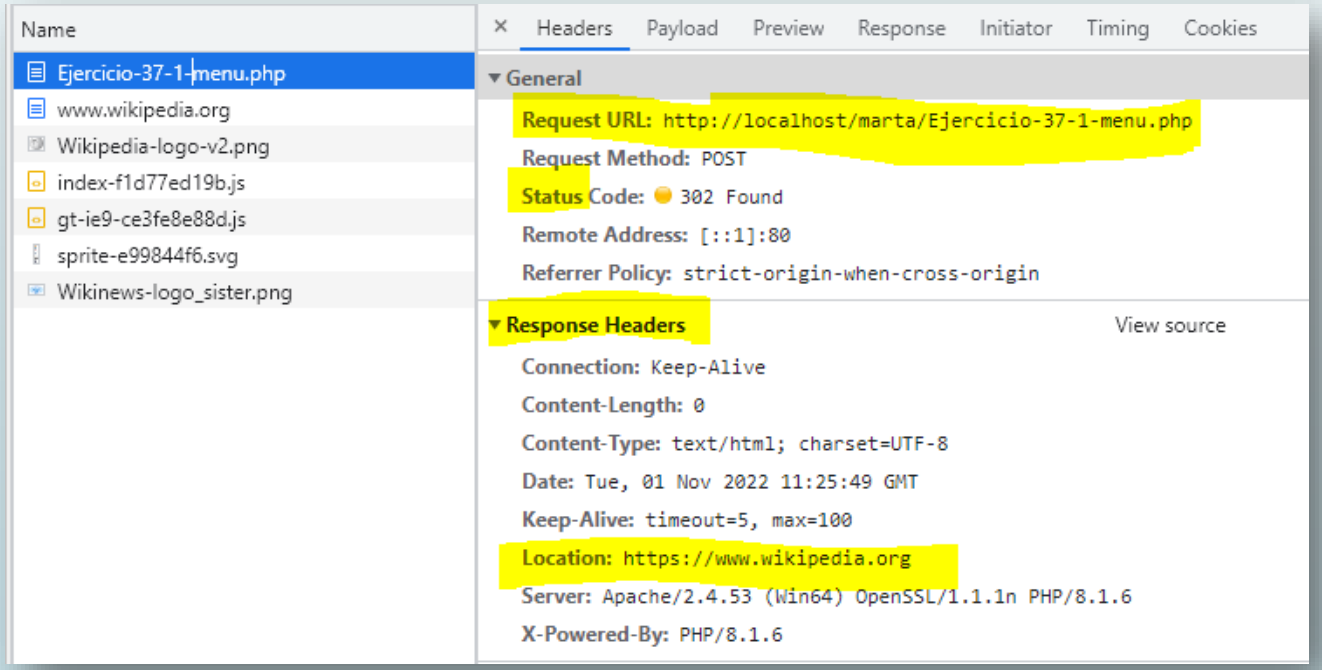
```
header('Location: http://www.wikipedia.org'); //Redirección a la Wikipedia
```

- Sin embargo, si redireccionas no puedes añadir salida. Por tanto:
  - **Antes de redireccionar** **NO SE PUEDE** escribir nada en la salida (ni un espacio en blanco)
  - **Después de redireccionar**, **ES NECESARIO** terminar el script PHP, para que no emita ninguna salida. No tiene ningún sentido enviar salida si el navegador va a redireccionar a otra página. Para ello podemos utilizar **return**.

Ejercicio 1:

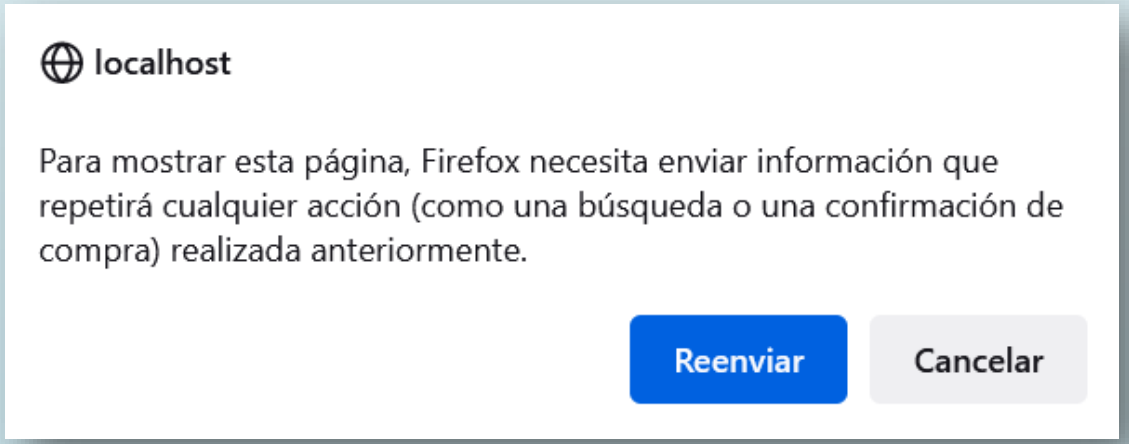
Menú

1. Haz un programa en PHP que simule un menú mediante redirecciones.
2. El usuario indicará un número en un campo de formulario POST, y navegará a la página indicada.
3. Abre el inspector, en el apartado de red, y observa la redirección
4. Refresca la página, y observa el mensaje



Refrescando un formulario POST

- Cuando hacemos un POST, normalmente enviamos información para ser procesada en el servidor.
- Imaginemos que la información indica que se haga un pago.
- Imaginemos que, después de confirmar el pago, refrescamos la página.
- La página que estamos refrescando, es una orden de pago, por tanto, se enviaría de nuevo la orden de pago.
- Habitualmente los navegadores detectan la situación, y emiten un mensaje confirmando que queremos enviar de nuevo ese mismo POST.
- Tanto el mensaje (que es confuso) como la situación, son malos para nuestras aplicaciones.
- Podemos evitar que se dupliquen los POST mediante redirecciones, como vamos a explicar en la siguiente página



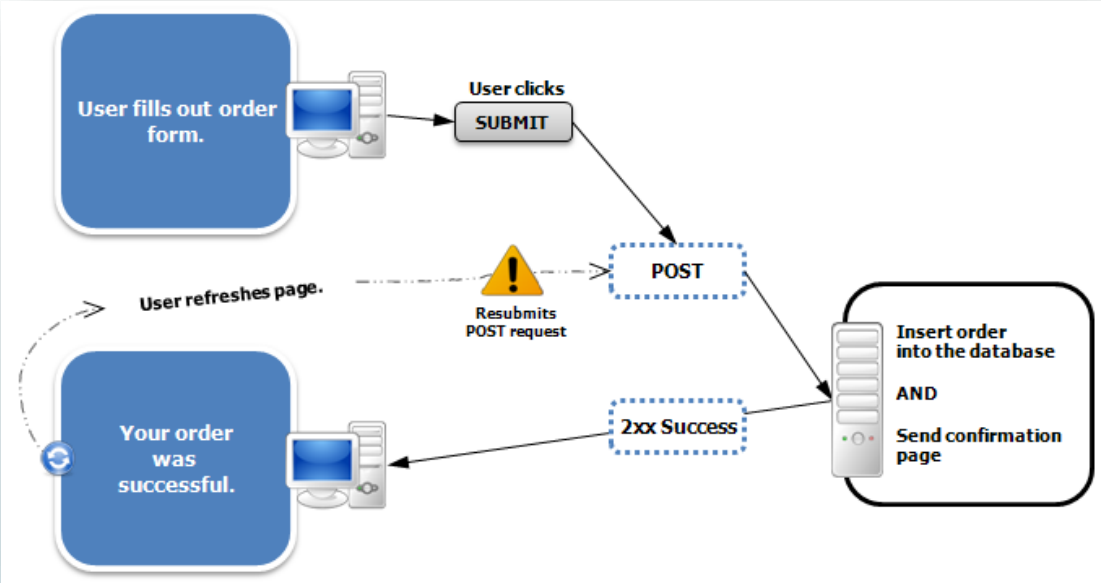
1. **Después de un POST, hemos refrescado el navegador.**
2. **Firefox detecta que se está *reenviando* un POST, con los mismos datos, y avisa por si es un pago, para que no se duplique!!**

Evitando POSTs dobles: Patrón POST-redirect-GET

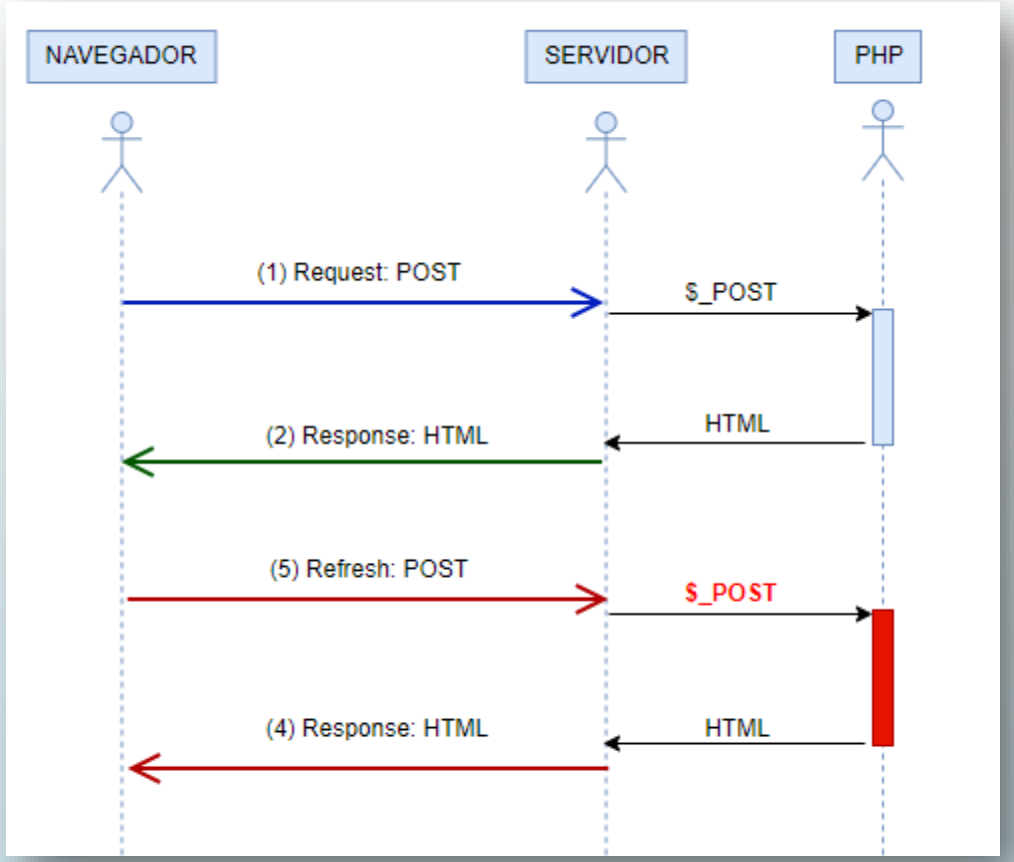
Para prevenir que un POST se reenvíe varias veces, y se repita algún proceso delicado como pueda ser un pago, se hace lo siguiente:

1. Cuando nuestro programa recibe los datos con el **POST**, *realiza el proceso* (validaciones, pagos, etc).
2. Luego envía un **Response** con una **redirección** a sí mismo, y *termina* con un **return**. No enviará salida visual, ni alcanzará el código de la parte de VISTA, porque le hemos hecho terminar antes. Si hubo validaciones o fallos, el mensaje correspondiente se guarda en una **sesión.**
3. El navegador, al recibir el Response anterior, realiza la redirección, esta vez con GET
4. Al recibir un GET, nuestro programa no realiza el proceso conforme a datos de POST. Lo que hará será ir a la parte de VISTA y mostrar la información. Los mensajes que pudiera haber, los recupera de la sesión.

Gráfico explicativo: Problema del POST

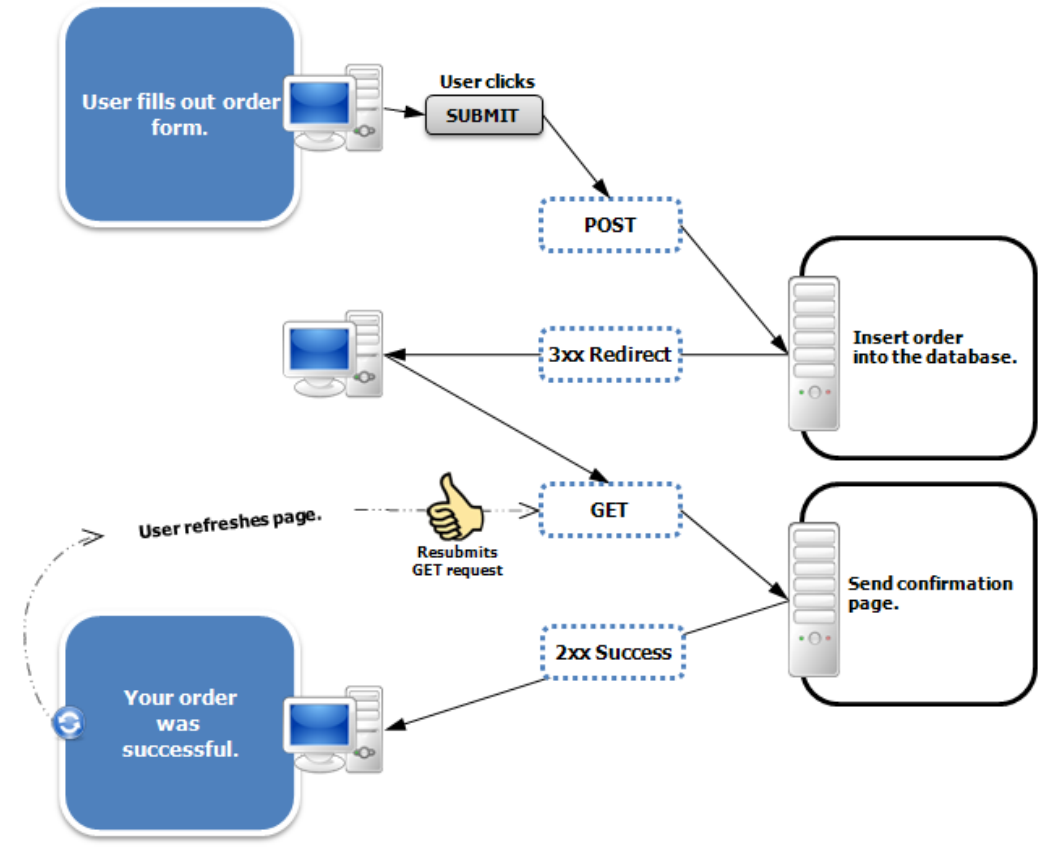


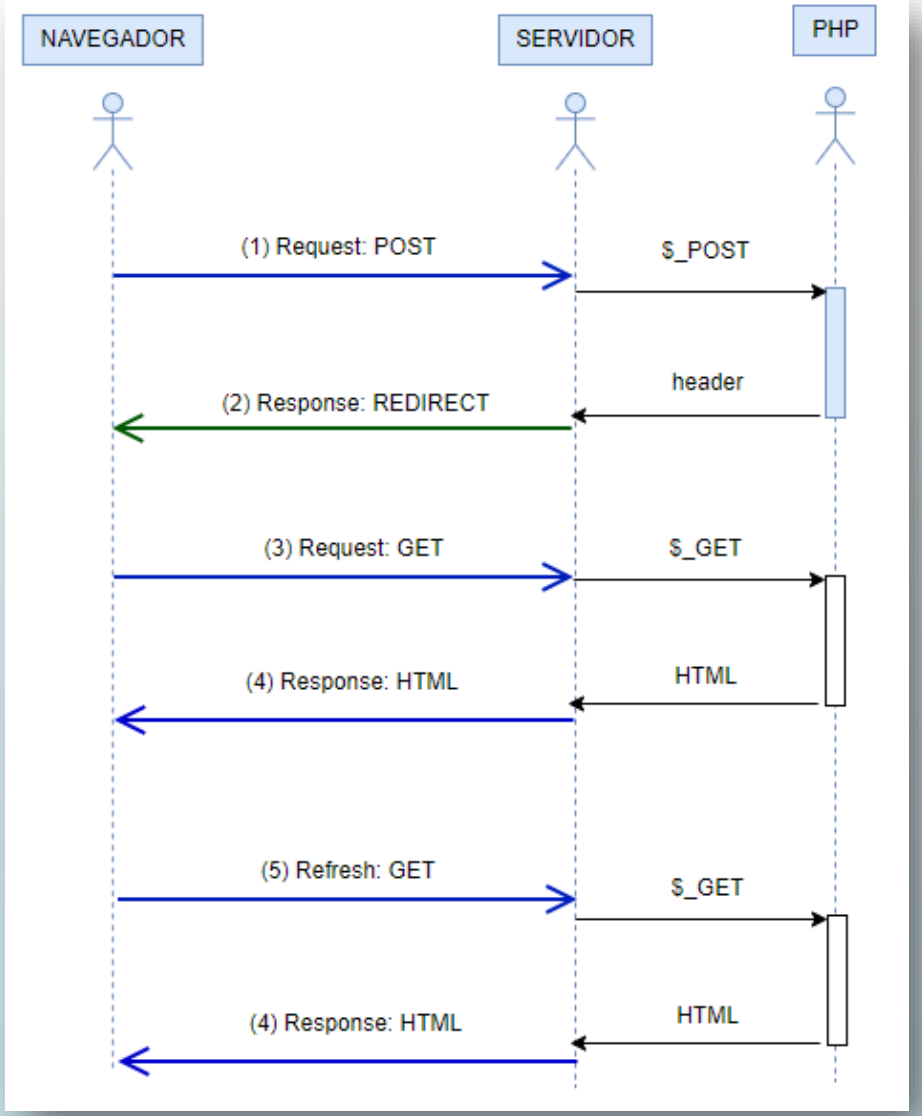
1. Se hace submit de un formulario con POST.
2. El programa en servidor realiza el proceso y responde que OK
3. El usuario refresca la página --> El POST es reenviado
4. A veces, el navegador avisa de que se va a reenviar, dando opción a cancelarlo. Imaginemos que el usuario no cancela.
5. El programa vuelve a realizar el proceso.... 2 veces.



En el diagrama de secuencia podemos ver que cada refresh vuelve a enviar el mismo POST, con lo que nuestro programa vuelve a ejecutar el código correspondiente al submit.
Si el código es un pago, vuelve a realizarse el pago.

Gráfico explicativo: Patrón POST-Redirect-GET



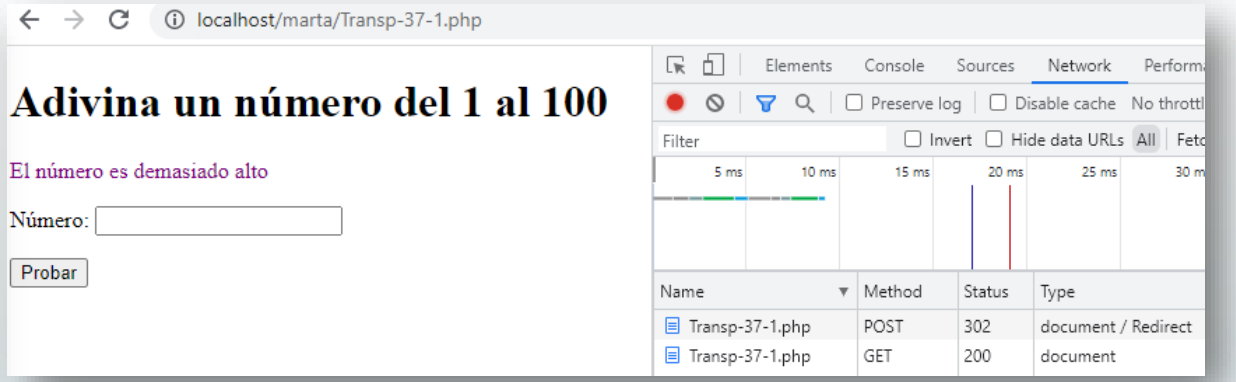


1. Se hace submit de un formulario con POST.
2. El programa en servidor realiza el proceso y con header pide redirección. No envía salida HTML
3. El navegador envía request del location, con GET.
4. El programa en servidor sólo emite HTML, no realiza proceso
5. El usuario refresca la página, se envía un GET
6. El servidor emite HTML, no realiza proceso

Código de control: Patrón POST-redirect-GET

```php
<?php
// El usuario adivina el número 33
  session_start();
  if (isset ($_POST["numero"])) {
    $numero = (int)$_POST["numero"] + 0; // Por si viene vacío
    // Guardo el número en la sesión
    $_SESSION["numero"] = $numero;
    if ( $numero == 33 ) {
      $_SESSION["mensaje"] = "Has adivinado!! El número es 33";
    } else if ( $numero < 33 ) {
      $_SESSION["mensaje"] = "El número es demasiado bajo";
    } else if ( $numero > 33 ) {
      $_SESSION["mensaje"] = "El número es demasiado alto";
    }
    // Tras guardar los mensajes de validación en la sesión
    // Envío al navegador un response de redirección a este
    // mismo programa, pero con GET
    header("Location: Transp-37-1.php");
    return;
  }
?>
```

CONTROLADOR



1. El usuario pone 50 en el campo y pulsa "Probar" (submit)
2. El navegador envía un request POST con el parámetro "numero"
3. Mi programa inicia una sesión, recuperando los datos que hubiera en $_SESSION
4. Mi programa mira si se hizo un POST, en cuyo caso el parámetro "numero" estará en el $_POST
5. Si resulta que se hizo un POST, entonces valida el número, y prepara los mensajes de visualización. En este caso los mensajes se guardan en la sesión, ya que no vamos a visualizarlos ahora y en una variable se perderían.
6. Una vez hechas las validaciones, redirecciona y sale.
7. En este caso, nunca se ejecuta la parte de VISTA del programa, ni tampoco se emite salida HTML, sólo se emite con header una respuesta tipo "redirect" para que el navegador redireccione de nuevo a mi programa, esta vez con GET.
8. Cuando mi programa reciba ese GET, recuperará los datos de la sesión, pero no hará nada de la parte de control, irá directamente a la parte de VISTA.

Código: Patrón POST-redirect-GET

```php
<!-- VISTA - Sólo llegan las peticiones GET -->
<?php
  // Si había un mensaje almacenado en la sesión, lo muestro
  $miMensaje = isset($_SESSION["mensaje"]) ? $_SESSION["mensaje"] : "";
?>
<!DOCTYPE html>
<head>
  <title>Adivina</title>
  <style>
    .mensaje {
      color: purple;
    }
  </style>
</head>
<body>
  <h1>Adivina un número del 1 al 100</h1>
  <p class="mensaje"><?=$miMensaje?></p>
  <form method="post">
    <label for="numero">Número: </label>
    <input name="numero" id="numero" type="number"/>
  </p>
  <p>
    <input type="submit" value="Probar"/>
  </p>
  </form>
</body>
```

1. La parte de VISTA (a la izquierda), sólo se ejecuta si se llama al programa directamente, o bien si se llama mediante la redirección GET
2. Si en un request anterior se almacenó un mensaje en la sesión, este programa mostrará el mensaje
3. Si el usuario refresca, se enviará un GET, que es el último request que se hizo
4. Al recibir el GET, el programa pasa a la parte de VISTA, y muestra cualquier mensaje que hubiera almacenado en la sesión
5. Si se refresca, volverá a mostrarse el formulario, con el mensaje de la sesión, aunque esta vez no hayamos indicado un número

- Con el patrón POST-redirect-GET se soluciona el problema de refresco si habíamos hecho un request con POST.
- Se responde con HTML, sólo si el request era GET.
- NUNCA enviaremos salida si recibimos un POST.
- La salida se almacena en la sesión, y se muestra en el siguiente response.
- Problema: El mensaje está en la sesión, y cada vez que refresque seguirá apareciendo.
- Para que el mensaje salga sólo una vez, tengo que eliminarlo en cuanto lo muestre

Patrón POST-redirect-GET-Flash

```php
<!-- VISTA - Sólo llegan las peticiones GET -->
<?php
  // Si había un mensaje almacenado en la sesión, lo muestro
  $miMensaje = isset($_SESSION["mensaje"]) ? $_SESSION["mensaje"] : "";
  unset($_SESSION["mensaje"]);
?>
<!DOCTYPE html>
<html>
<head>
  <title>Adivina</title>
  <style>
    .mensaje {
      color: purple;
    }
  </style>
</head>
<body>
  <h1>Adivina un número del 1 al 100</h1>
  <p class="mensaje"><?=$miMensaje?></p>
  <form method="post">
    <label for="numero">Número: </label>
    <input name="numero" id="numero" type="number"/>
  </p>
  <p>
    <input type="submit" value="Probar"/>
  </p>
  </form>
```

La función unset elimina la entrada "mensaje" de $_SESSION y también del archivo de la sesión.

- Hemos modificado el programa para que la visualización del mensaje sea como un **flash**, es decir, se vea sólo una vez.
- El código asigna el contenido de la sesión a una variable. Luego elimina el mensaje de la sesión.
- El contenido de la variable se muestra en el HTML de salida.
- Cuando se envía el response, se envía el mensaje, pero en posteriores refrescos, éste ha sido eliminado de la sesión por lo que hay nada para mostrar.
- A este patrón se le llama POST-redirect-GET-Flash, y consigue que los mensajes sólo se muestren una vez.
- Este patrón mantiene las ventajas de no mostrar salida como respuesta a un POST

AUTENTICACIÓN

session_start()

Sesión vs login

- No es lo mismo tener una sesión que hacer login.
- Lo habitual es iniciar una **sesión** en el mismo instante en que te conectas a la aplicación por primera vez.
- La **cookie de sesión** se envía con el primer response, y queda establecida la sesión, a veces incluso sin información asociada. Sería un fichero en blanco.
- El código de sesión se almacena en la cookie para identificación del navegador por el servidor.
- Un **login**, en cambio, guarda la información del usuario en la sesión. Esta información no se guarda en el navegador, como la cookie, sino en el servidor.
- Un **logout** elimina de la sesión la información del usuario. La sesión sigue identificada en la cookie del navegador, pero en el servidor no está la información del login del usuario.

Patrón de login: session-POST-redirect-GET-flash

- Cuando navegas a una página de login, lo primero que se establece es una sesión, y en el primer response vendrá la cookie de sesión.
- Luego, al rellenar el formulario, siempre con POST, se envían los datos de login y password:
  - El programa identifica la información, y comprueba que login y password son correctos.
  - Si son correctos almacena en la sesión el login, y redirecciona a otro página.
  - Si no son correctos guarda un mensaje de error en la sesión y se redirecciona a sí mismo.
  - En cualquier caso, nunca ejecutará el código de vista tras recibir un request con POST
- Una vez aceptado el login, la información del usuario permanece en la sesión.
- Cualquier programa de la aplicación sabe que estás logado porque la información del usuario está en la sesión

Código de login (1): session-POST-redirect-GET-flash

CONTROL

```php
<?php
  session_start(); // Siempre !!
  // Si hicieron submit al formulario de login
  if ( isset($_POST["usuario"]) && isset($_POST["password"])) {
    unset($_SESSION["usuario"]); // Cierro la sesión anterior
    // Valido la información
    if (empty($_POST["usuario"])) {
      $_SESSION["mensaje"] = "Debes indicar un usuario";
      header("Location: Transp-37-2-login.php");
      return;
    } else if ($_POST["password"]=="1234") {
      $_SESSION["usuario"] = $_POST["usuario"];
      $_SESSION["mensaje"] = "Bienvenido a nuestra aplicación!!";
      header("Location: Transp-37-2-app.php");
      return;
    } else {
      $_SESSION["mensaje"] = "Password incorrecta";
      header("Location: Transp-37-2-login.php");
      return;
    }
  }
?>
```

1. Lo primero se inicia la sesión. En todos los casos. Si no existía, se creará y se enviará la cookie de sesión en el response.
2. En caso de que el usuario haya enviado los datos de login y password, los datos estarán en el POST, y se va a validar la información y a hacer login en caso de que sea correcta.
3. En caso de que el request no sea un POST del formulario, no se ejecuta ningún código de control, y se pasa a la parte de vista.

- $_SESSION["usuario"]: Es el identificativo del usuario, se guarda en la sesión (servidor). Cuando nuestros programas lo encuentren, significará que ha conseguido hacer login y está logado.
- $_SESSIO["mensaje"]: Se guardan los mensajes a mostrar. Como no podemos mostrar salida cuando se hace un POST, sino que redireccionamos, necesitamos guardar aquí el mensaje para poder mostrarlo en un posterior request(GET)-response

Código de login (2): session-POST-redirect-GET-flash

VISTA

```php
<?php
  $mensaje = isset($_SESSION["mensaje"]) ? $_SESSION["mensaje"] : "";
  unset($_SESSION["mensaje"]); // Lo elimino para el flash
?>
<!DOCTYPE html>
<html>
  <head>
    <title>LOGIN</title>
  </head>
  <body>
    <h1>Haz login</h1>
    <p style="color: purple"><?=$mensaje?></p>
    <form method="post">
      <p>
        <label for="usuario">Usuario: </label>
        <input type="text" id="usuario" name="usuario"/>
      </p>
      <p>
        <label for="password">Password</label>
        <input type="password" id="password" name="password"/>
      </p>
      <p>
        <input type="submit" value="Log in"/>
      </p>
    </form>
  </body>
</html>
```

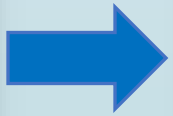
1. A la parte de vista llegamos sólo por request de tipo GET (primera llamada o redirección)
2. Si había un mensaje almacenado en la sesión, se muestra.
3. Para que el mensaje sólo aparezca una vez, cuando debe, y no en posteriores refresh, eliminamos el mensaje de la sesión una vez utilizado.

- $mensaje: Almacena temporalmente el mensaje a mostrar en HTML, así podemos borrarlo de la sesión. Esto implementa el patrón flash (mostrar el mensaje sólo una vez, no en posteriores refrescos)
- $_SESSION["mensaje"]: Mensaje que se almacenó en un request-response anterior. Como hemos llegado a la vista, significa que el request fue con GET
- Si los hay, vamos a mostrar los mensajes que nos hubieran redirigido

Código de aplicación:

```php
<?php
  session_start(); // Siempre !!
  // Compruebo que estoy logada
  $usuario="";
  if ( isset($_SESSION["usuario"]) ) {
    $usuario = "Usuario: " . $_SESSION["usuario"];
    $mensaje = isset($_SESSION["mensaje"]) ? $_SESSION["mensaje"] : "";
    unset($_SESSION["mensaje"]);
  } else {
    $mensaje = "Haga login para iniciar sesión";
  }
?>
<!DOCTYPE html>
<html>
  <head>
    <title>APLICACION</title>
  </head>
  <body>
    <h1>Aplicación</h1>
    <p style="color:darkgrey"><?=$usuario?></p>
    <p style="color: purple"><?=$mensaje?></p>
      <p>
        <a href="Transp-37-2-login.php">Login</a>
      </p>
      <p>
        <a href="Transp-37-2-logout.php">Logout</a>
      </p>
  </body>
</html>
```

- En este ejemplo, la aplicación no tiene funcionalidad, sólo muestra un mensaje de bienvenida y da la opción a hacer login o logout.
- Si no existe usuario en la sesión, no estamos logados, y se muestra un mensaje para indicar que se haga login.
- Si existe el usuario, estamos logados, y vamos a mostrar el mensaje que hubiera en la sesión.



Patrón de logout: POST-redirect-borrar sesion-redirect

- El programa de mi aplicación identifica que estoy logado porque la información del login (nombre de usuario) está en la sesión.
- Si no encuentra el nombre del usuario en la sesión, indicará que no estoy logado y no permitirá acceder a la funcionalidad.
- Para realizar logout, basta con redireccionar a un programa de logout que:
  - Inicia la sesión
  - Destruye la sesión, borrando todos los datos
  - Redirecciona al programa de login, que permitirá logarse de nuevo

```php
<?php
  session_start(); // Siempre !!
  session_destroy();
  header("Location: Transp-37-2-app.php");
?>
```

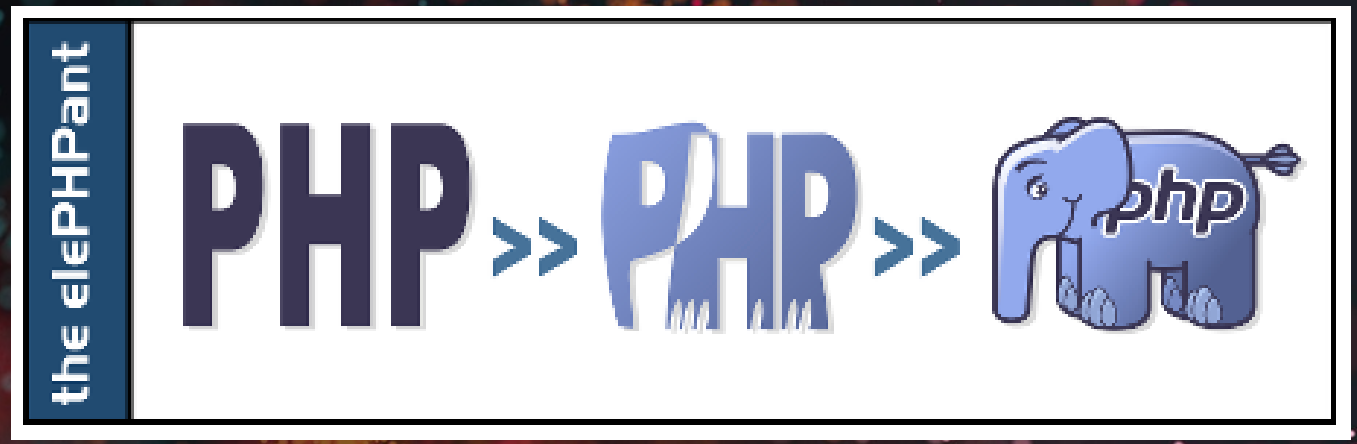
- **El código de logout es muy sencillo.**
- **Identifica la sesión, y luego la destruye. Esto borrará toda la información que se hubiera almacenado en la misma.**
- **Finalmente redirecciona al programa de aplicación**

Ejercicio 2:

Login-logout

Siguiendo el patrón **sesion-POST-redirect-GET-flash**, haz una aplicación en PHP con los siguientes programas:

1. **login.php**: Un programa de login. Utiliza una sesión para almacenar el usuario.
2. **logout.php**: Un programa de logout. Eliminará toda la información de la sesión.
3. **app.php**: Un programa de aplicación que ofrecerá un formulario de aspecto similar al formulario 1 del ejercicio 4 de cookies y sesiones que hay en el aula virtual.
    1. Este formulario no llamará a otro formulario, guardará los datos directamente en la sesión
    2. Este programa ofrecerá dos botones, uno para guardar los datos, otro para hacer logout
    3. Si no estamos logados (por ejemplo, si entramos directamente), sólo mostrará un mensaje indicando que debemos logarnos, y la opción de navegar a "login.php"



https://www.php.net/docs.php