Desarrollar un aplicación Html-CSS-Javascript, ninguna tecnología más está permitida, ni jquery, ni ningún framework Javascript, ni controles externos.

La página debe recoger los datos de un formulario, estos serán validados correctamente pulsando el botón validar, una vez se verifiquen correctamente se mostrará en otra página de resultados.

La página debe solicitar los siguientes datos:

* + - Nombre (debe soportar 50 caracteres de longitud)
    - Teléfono (debe validarse usando una expresión regular)
    - Email (debe validarse para admitir solo emails correctos)
    - Contraseña (debe permitir al menos una letra debe ser mayúscula, debe tener un al menos un digito y un carácter especial y al menos debe tener 8 dígitos de longitud). El password no se debe visualizar en ningún momento.
* Los campos deben estar centrados en la página y conformar parte de una tabla en HTML, los estilos de los títulos y los campos deben ser diferentes.
* Todos los campos deben ser obligatorios, ninguno puede quedar vacío.
* Que es Rest y para qué sirve?
* Detallar la diferente entre get, pop, push, delete poner un ejemplo con código de cada uno de ellos usando como base el ejercicio anterior.
* ¿Como defino un método de autenticación en un Html?
* ¿Como se establece una clave de autorización proxy y para qué sirve?

**Tiempo límite hasta el 3 de febrero:** No hay escusas, si el día tres no lo tenéis, no os voy a insistir más como en ocasiones previas. (Este es un ejercicio que a un desarrollador web junior podría llevarle 30 minutos). Así que depende de vosotros, si queréis avanzar hacerlo, si por el contrario preferís centraros en otros temas porque los consideráis más importantes no lo hagáis y así yo no perderé el tiempo, una vez más os digo que esto es para vuestro futuro y por lo tanto es vuestra decisión. A mí me servirá para saber si vais a ser capaces algún día de progresar en otras tecnologías o si por el contrario solo os podre usar para hacer informes y formularios…

No hay ayudas, si no sabéis como resolverlo, buscarlo, todo está en internet, hay cientos de ejemplos, documentos y videos, el ejercicio está para que os rompáis un poco la cabeza que Dios os ha dado y sobre todo que aprendáis muchas cosas relacionadas…. ánimo!!

**Que es Rest y para qué sirve?**

REST: REPRESENTATIONAL STATE TRANSFER, es un interfaz entre sistemas que usa HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles.

**Métodos de petición HTTP:**

HTTP define un conjunto de **métodos de petición** para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados HTTP verbs. Cada uno de ellos implementan una semántica diferente, pero algunas características similares son compartidas por un grupo de ellos.

[**GET**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET)

El método GET  solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

[**POST**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

[**PUT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)

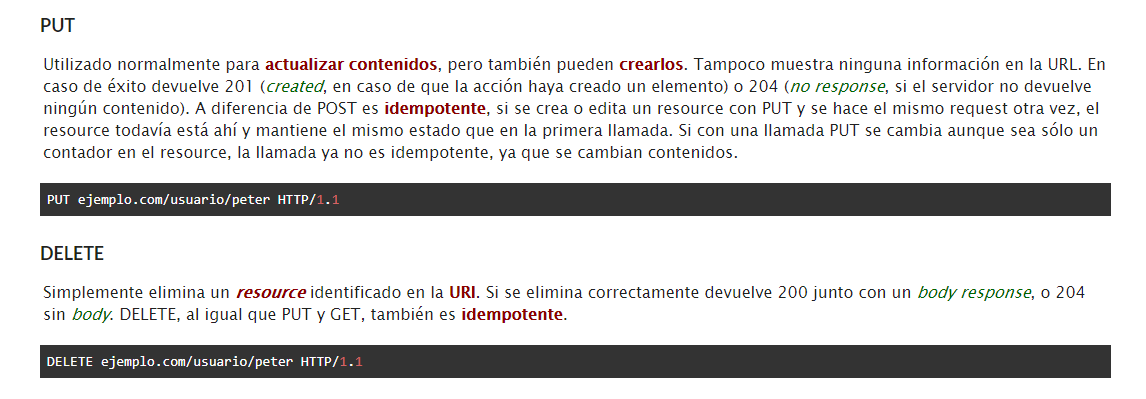
El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

[**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

El método DELETE borra un recurso en específico.

Ejemplos: <https://diego.com.es/metodos-http>





Diferencia **POST vs PUT:** <http://notasjs.blogspot.com/2013/07/http-diferencia-entre-post-y-put.html>

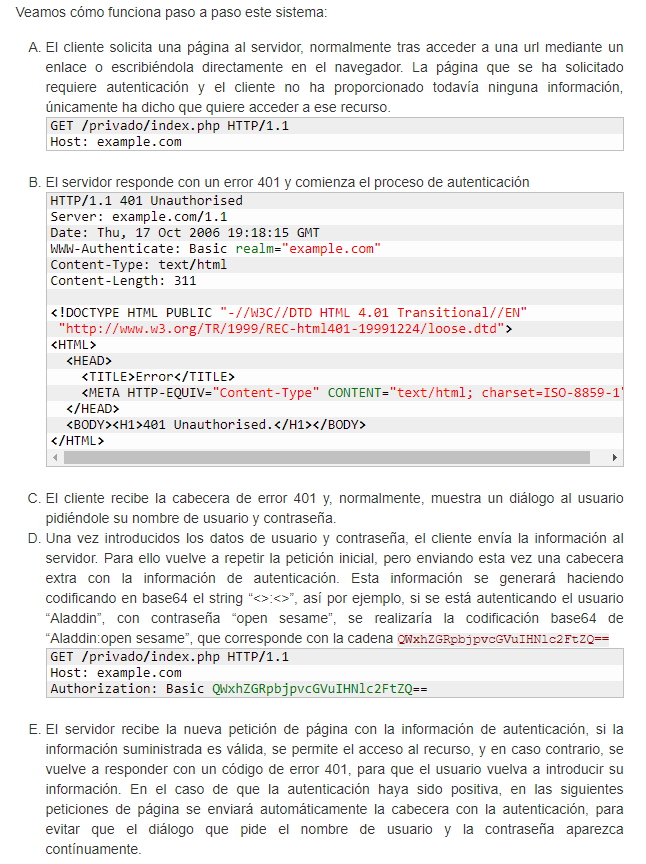
La diferencia no es que uno se utilice para crear y otro para actualizar. Los dos métodos pueden crear un recurso y los dos métodos pueden actualizarlo.

PUT se utiliza para poner un recurso en un lugar especificado (la URL). POST es mucho más general. POST simplemente envía una información al servidor para que éste la trate como considere oportuno. Puede crear un recurso nuevo, devolver una información, puede guardarlo en la base de datos, borrarlo, duplicarlo o modificarlo, puede crear 10 recursos de un tipo y dos de otro o puede no hacer absolutamente nada.

**Método de autenticación html:**

la autenticación de acceso básica es un método diseñado para permitir a un navegador web, u otro programa cliente, proveer credenciales en la forma de usuario y contraseña cuando se le solicita una página al servidor.

No está pensado para ser utilizado sobre líneas públicas, debido a que las credenciales que se envían desde el cliente al servidor, aunque no se envían directamente en texto plano, se envían únicamente codificadas en Base64, lo que hace que se puedan obtener fácilmente debido a que es perfectamente reversible, es decir, una vez que se posee el texto codificado es posible obtener la cadena original sin ningún problema, por lo que la información enviada no es cifrada ni segura.



**Autenticación PROXY:**

El mismo mecanismo de desafío y respuesta puede ser usada para autenticación por proxy. En este caso, es el proxy el que hace de intermediario y requiere la autenticación. En el caso de los proxys, el código de estado para requerir autenticación es **407**  (Proxy Authentication Required), la cabecera de respuesta Proxy-Authenticate contiene al menos un requerimiento aplicable en el proxy, y la cabecera de petición Proxy-Authorization es usada para proveer la credencial en el servidor proxy.

Cuando nos conectamos con un proxy, el servidor al que accedemos en realidad recibe la solicitud del proxy, en vez de recibirla directamente desde nuestro ordenador. Puede haber sistemas proxy que interceptan diversos servicios de Internet. Lo más habitual es el proxy web, que sirve para interceptar las conexiones con la web y puede ser útil para incrementar la seguridad, rapidez de navegación o anonimato.

El proxy web es un dispositivo que suele estar más cerca de nuestro ordenador que el servidor al que estamos accediendo. Este suele tener lo que denominamos una caché, con una copia de las páginas web que se van visitando. Entonces, si varias personas que acceden a Internet a través del mismo proxy acceden al primer sitio web, el proxy la primera vez accede físicamente al servidor destino, solicita la página y la guarda en la caché, además de enviarla al usuario que la ha solicitado. En sucesivos accesos a la misma información por distintos usuarios, el proxy sólo comprueba si la página solicitada se encuentra en la caché y no ha sido modificada desde la última solicitud. En ese caso, en lugar de solicitar de nuevo la página al servidor, envía al usuario la copia que tiene en la caché. Esto mejora el rendimiento o velocidad de la conexión a Internet de los equipos que están detrás del proxy.





