## Índice de contenido



# Implementación de código rodante para determinar la autenticidad de un cliente.

## Problemática:

Se desea realizar una comunicación cliente/servidor por el protocolo HTTP al tiempo de que sea lo mas segura posible evitando, el mayor problema consiste en el hecho que el protocolo HTTP enviá los datos en texto plano quedando al descubierto el usuario y contraseña al momento de logearse permitiendo que si un usuario malicioso lo desee pueda robar la identidad de otro dispositivo al obtener el usuario y contraseña.

## Solución propuesta.


Contemple la siguiente tabla clientes como se muestra, como se puede ver contiene diversos campos, esta tabla va a estar en el servidor y nos va a ayudar a asegurar que el cliente sea quien dice ser.

### Registro de clientes.

Al momento de registrar un cliente el servidor va a generar una clave la cual es **clave_cifrar** dicha clave se debe de guardar en el dispositivo con el cual queremos acceder a la aplicación.

## Login: Inicio de sesión.


El login lo va a realizar en los siguientes pasos:

1) El cliente enviá por el método POST información de su username y su password ambos cifrado s por md5.
   Esto es :

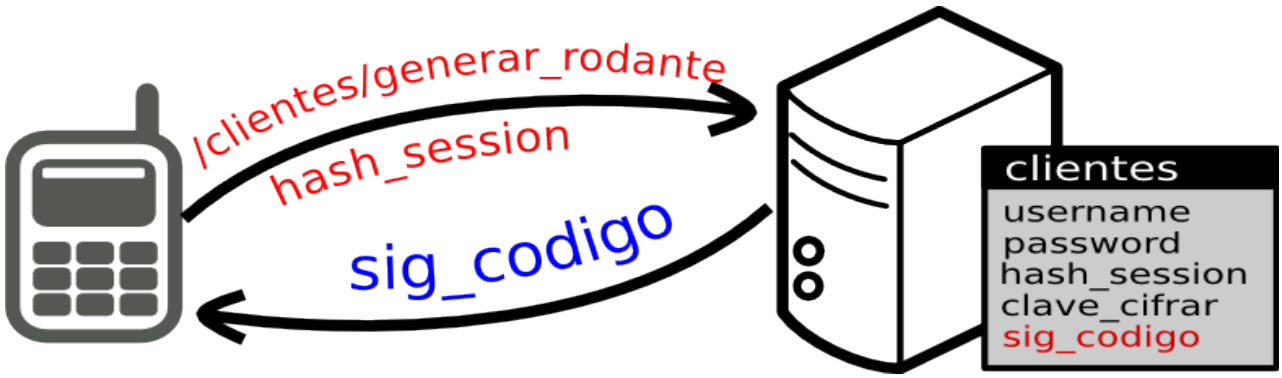
```
var usernameCifrado = md5($('#username'));
var passwordCifrado = md5($('#password'));
```

2) Si el username y password cifrado coincide con valores guardados en la base de datos, el servidor responde con un hash_session único.
3) El cliente recibe el hash_session y lo guarda para futuras solicitudes.

## Envió de datos sensibles.

Este proceso lo vamos a realizar en dos pasos:

### 1.- Preparar el envió de datos.




Aquí vamos a obtener un "*código rodante*"

1) El cliente va a realizar una solicitud del tipo POST a la url: /clientes/generar_rodante enviando la información de su **hash_session** obtenida previamente en el login.
2) El servidor a recibir esta información, busca si existe un cliente con dicha **hash_session** . De existir modifica su registro actualizando el atributo **sig_codigo** por uno nuevo el cual genera oratoriamente y se lo regresa al cliente.

## 2.- Envió de datos firmando la autenticidad del cliente.

El servidor no tiene otra forma de saber que el cliente es quien dice ser mas que el ***hash_session*** sin embargo este dato puede ser robado incluso el **username** y **password** cifrados los pueden extraer y usarlos para posteriormente iniciar sesión cuantas veces quieran.
Para solucionar esto esta el campo **clave_cifrar** el cual nunca es trasmitido por la red y por lo cual la única forma que alguien lo pueda obtener es accediendo físicamente al dispositivo o acatando al servidor para robar este dato. Esta clave_cifrar nos va a servir para que el cliente firme una solicitud que realice hacia el servidor indicando que el es realmente el cliente que inicio sesión y que ademas se encuentra registrado previamente en el servidor.
Los pasos que tendrán que hacer son los siguientes:

1) El cliente al realizar una solicitud sensible esta tendrá que estar siempre acompañada de su **hash_session** y **clave_verificacion** por lo cual debe genearar su clave de verificación quedando como:
```
clave_verificacion = md5(sig_codigo + clave_cifrar)
```

2) El servidor: Al recibir cualquier información sensibledebe validar la **clave_verificacion**. Al tener tanto la clave **sig_codigo** y la **clave_cifrar** del cliente puede obtener la **clave_verificacion** para comprobarlas, esto realizando el mismo proceso (`md5(sig_codigo + clave_cifrar)` ).
3) Si ambas claves; la que recibe y la que genera coinciden entonces el servidor puede asegurar que es el cliente quien dice ser y tras usar el **sig_codigo** el servidor hace nula esta clave quedando inservible para otra solicitud.