

Hito individual 1

Despliegue de aplicaciones

Sergio Felipe García

17/11/2023

ÍNDICE

ÍNDICE	1
Despliegue de aplicación.P1	2
Despliegue en Node:	2
Despliegue en Wordpress	5
Configuración y Despliegue de Certificados HTTPS	12
Certificados de Seguridad	12
Generación del certificado en Kali linux con openssl	12
Configuración para Apache:	13
Certificado de seguridad en node Windows	19
Instalación de Openssl	19
Validación Multimedia y Seguridad en el Desarrollo de la Aplicación	28
Visualización de formatos de imagen y vídeo	28
Archivo de arranque	29
Protección de archivos de configuración	31
Librería Crypto de Node	31
Cifrado del archivo de configuración.	31
Inspección de directorios	34
Desactivar la generación de listados de directorios	34
Manejo del error 403	35

Despliegue de aplicación.P1

Despliegue en Node:

Ya he hecho el set up para que el servidor tenga Node.js la versión 18.18.0 y tenía previamente instalado el paquete npm.

El comando `npm init -y` sirve para inicializar un nuevo proyecto de Node.js sin tener que responder manualmente a las preguntas del asistente de inicialización. Este comando crea un archivo `package.json`.

```
Your environment has been set up for using Node.js 18.18.0 (x64) and npm.

C:\Windows\System32>cd C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege

C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>npm init -y
Wrote to C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege\package.json:

{
  "name": "hito_individualdespliege",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>
```

Ejecuto `npm install` para instalar las dependencias del proyecto según lo especificado en tu archivo `package.json`. Up to date significa que todas las dependencias están ya instaladas y no hay cambios en las versiones especificadas

```
C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>npm install

up to date, audited 1 package in 2s

found 0 vulnerabilities
```

Lanzo la instalación de **Express** con `npm install express` este paquete de Node.js es un framework web para Node.js que simplifica el desarrollo de aplicaciones web y la creación de APIs.

```
C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>npm install express
added 62 packages, and audited 63 packages in 10s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>
```

En el fichero `index.js` se define la configuración un servidor web básico usando Express, en concreto para este proyecto, puertos, dependencias, maneja algunas rutas, y sirve archivos estáticos desde el directorio `/public`.

```
index.js / ...
// Importa las dependencias necesarias
const express = require('express');
const app = express();
const port = process.env.PORT || 3000; // Define el puerto en el que se ejecutará el servidor
app.use(express.static('public'));
// Define rutas y controladores
app.get('/', (req, res) => {
  res.send('¡Hola, mundo!');
  const myVariable = "Hola desde index.js";
});

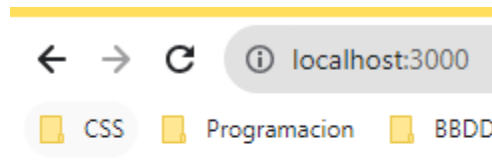
app.get('/', (req, res) => {
  res.redirect('/index.html');
  //res.sendFile(__dirname + '/public/design.html');//ok, pero sin css
});

// Arranca el servidor
app.listen(port, () => {
  console.log(`Servidor Express escuchando en el puerto ${port}`);
});
```

Para arrancar el servidor con esta configuración uso el comando `node index.js`

```
C:\Users\Sergio\Desktop\Sergio\hito_individualDespliege>node index.js
Servidor Express escuchando en el puerto 3000
```

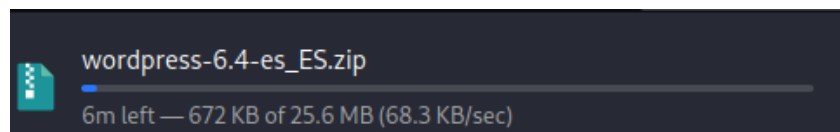
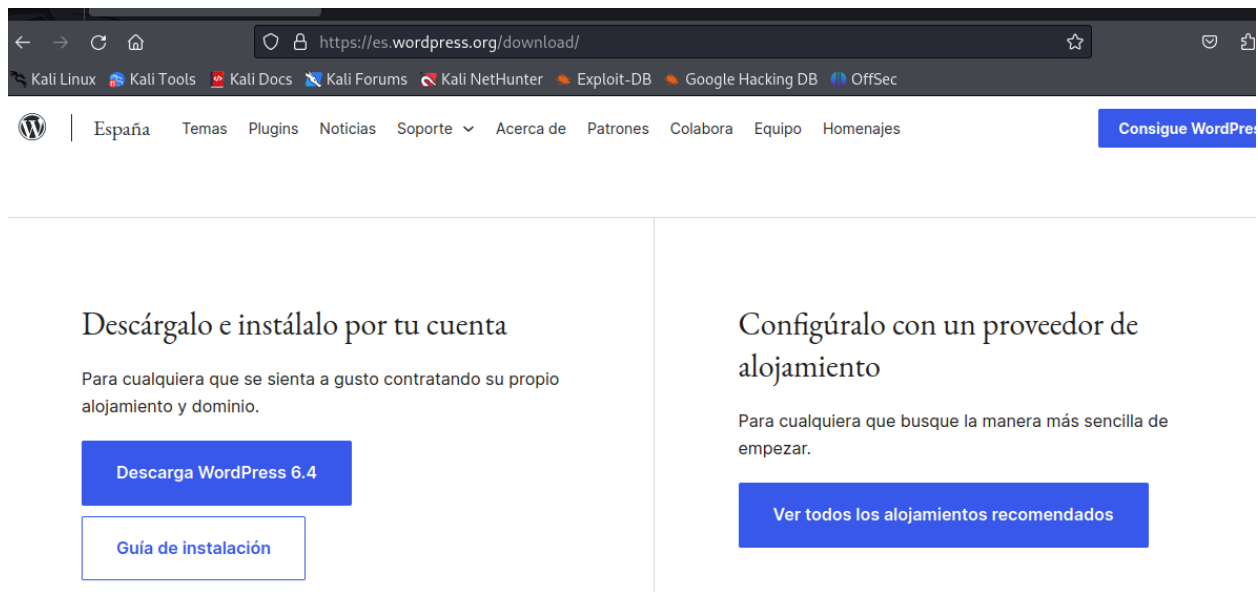
Una vez lanzado el comando node nos responde que el servidor Express está listo para recibir peticiones en el **puerto 3000**. A continuación lo compruebo en el navegador.



DESPLEGADO

Despliegue en Wordpress

Primero descargamos wordpress desde el sitio web oficial.



Luego compruebo si se ha descargado en la carpeta Downloads.

```
└─$ sudo su
[sudo] password for kali:
(root@kali)-[/home/kali]
# ls
Desktop  Downloads  Pictures  'system ctl apache2'  Videos
Documents Music      Public   Templates

(root@kali)-[/home/kali]
# cd Downloads

(root@kali)-[/home/kali/Downloads]
# ls
code_1.82.2-1694671812_amd64  wordpress
code_1.82.2-1694671812_amd64.deb  wordpress-6.3.2-es_ES.tar.gz
hook.js  wordpress-6.3.2-es_ES.zip
wincache-1.3.7.12-dev-5.6-nts-vc11-x86.exe  wordpress-6.4-es_ES.zip

(root@kali)-[/home/kali/Downloads]
# ls
code_1.82.2-1694671812_amd64
code_1.82.2-1694671812_amd64.deb
hook.js
wincache-1.3.7.12-dev-5.6-nts-vc11-x86.exe
wordpress
wordpress-6.4-es_ES.zip

(root@kali)-[/home/kali/Downloads]
#
```

y movemos el archivo a nuestra carpeta html, que es el lugar donde quiero desplegarlo.

```
(root@kali)-[/home/kali/Downloads]
# mv wordpress-6.4-es_ES.zip /var/www/html

(root@kali)-[/home/kali/Downloads]
#
```

extraemos los archivos comprimidos en el zip

```
(root@kali)-[/home/kali/Downloads]
# cd /var/www/html

(root@kali)-[/var/www/html]
# ls
index.html          sitio1             wordpress
index.nginx-debian.html sitio2            wordpress-6.4-es_ES.zip
latest-es_ES.tar.gz sitiokali         wordpress_old

(root@kali)-[/var/www/html]
# unzip wordpress-6.4-es_ES.zip
Archive:  wordpress-6.4-es_ES.zip
```

```
wordpress
wordpress-6.4-es_ES.zip
```

Desde dentro de la carpeta wordpress podemos ver los ficheros que se han descomprimido. Y parece que todo está correcto.

```
(root@kali)-[/var/www/html]
# cd wordpress

(root@kali)-[/var/www/html/wordpress]
# ls
index.php          wp-blog-header.php  wp-includes         wp-settings.php
license.txt        wp-comments-post.php wp-links-opml.php   wp-signup.php
readme.html        wp-config-sample.php wp-load.php         wp-trackback.php
wp-activate.php    wp-content          wp-login.php        xmlrpc.php
wp-admin           wp-cron.php         wp-mail.php
```

Tal y como indican los requisitos debemos tener una base de datos y una carpeta para guardar nuestros proyectos.

El comando `mysql -u root -p` se utiliza para conectarse al servidor MySQL con el usuario "root" y solicitar la contraseña. `-u root`: Especifica el nombre de usuario (-u significa "user") que se utilizará para la conexión, en este caso, `root` y `-p`: Indica a MySQL que solicite la contraseña. Ahora ya se pueden lanzar comandos de `MySQL`, para ejecutar consultas y administrar la base de datos.

```
File Actions Edit View Help
(kali@kali)-[~]
$ sudo su
[sudo] password for kali: /home/kali/
(kali@kali)-[~]
# mysql -u root -p
Enter password: /var/www/html/
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.11.5-MariaDB-3 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Como podemos lanzar comandos ahora creamos la base de datos hito y un usuario con todos los permisos:

```
(kali@kali)-[~]
$ sudo su
[sudo] password for kali: /home/kali/
(kali@kali)-[~]
# mysql -u root -p
Enter password: /var/www/html/
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.11.5-MariaDB-3 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database hito;
Query OK, 1 row affected (0.001 sec)

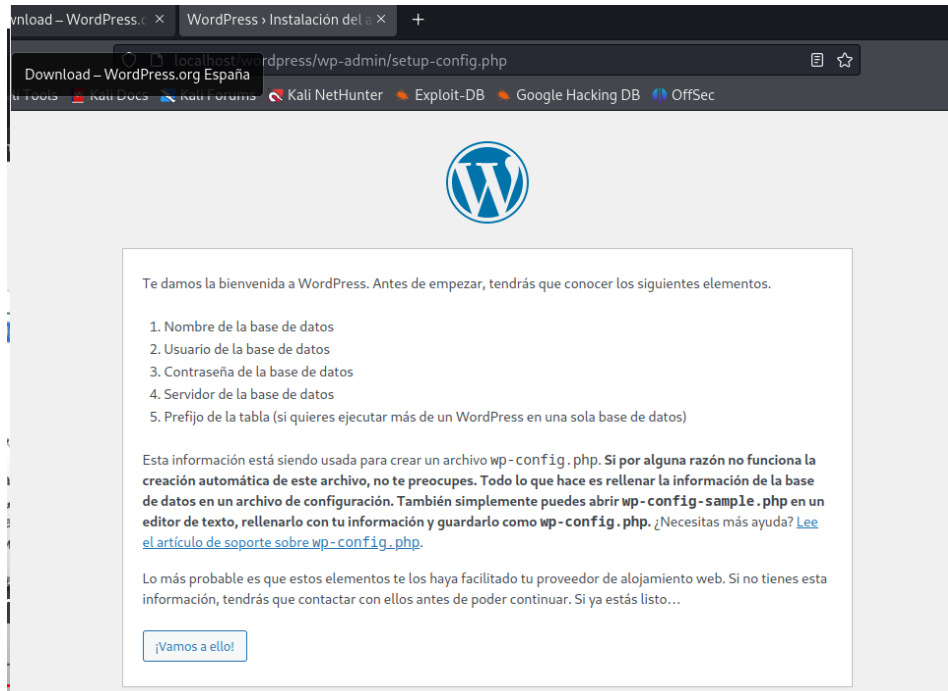
MariaDB [(none)]> grant all on hito.* to 'root'@'localhost' identified by '123';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```


Ahora que ya tenemos la bbdd funcionando, arranco el **servidor apache**.

```
(root@kali)-[/home/kali]
# service apache2 start
```

Una vez arrancado el apache ya puedo configurar **Wordpress** desde el navegador.



Rellenamos los campos que nos piden tal y como se muestra en las imágenes.

A continuación tendrás que introducir los detalles de tu conexión con la base de datos. Si no estás seguro de ellos, contacta con tu proveedor de alojamiento.

Nombre de la base de datos	<input type="text" value="hito"/>	El nombre de la base de datos que quieres usar con WordPress.
Nombre de usuario	<input type="text" value="root"/>	El nombre de usuario de tu base de datos.
Contraseña	<input type="password" value="..."/>	La contraseña de tu base de datos.
Servidor de la base de datos	<input type="text" value="localhost"/>	Si localhost no funciona, deberías poder obtener esta información de tu proveedor de alojamiento web.
Prefijo de tabla	<input type="text" value="wp_"/>	Si quieres ejecutar varias instalaciones de WordPress en una sola base de datos cambia esto.

La password es débil, pero es la que recuerdo en esta práctica, en el mundo real buscaría una más segura.

munao.

Información necesaria

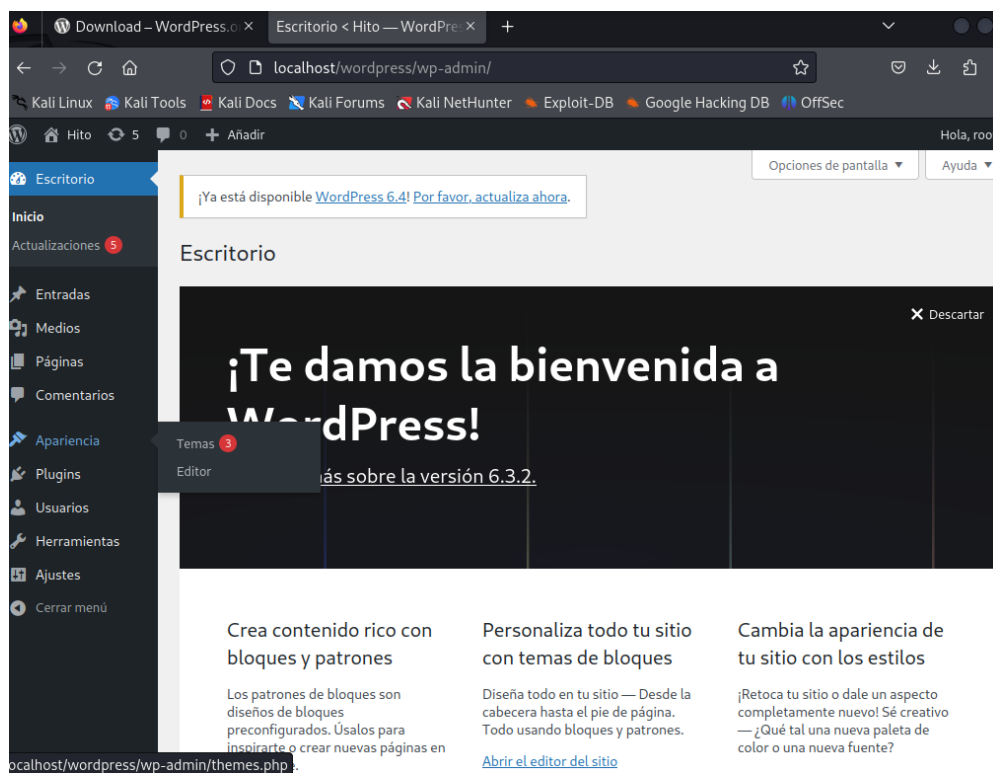
Por favor, proporciona la siguiente información. No te preocupes, siempre podrás cambiar estos ajustes más tarde.

Título del sitio	<input type="text" value="Hito"/>
Nombre de usuario	<input type="text" value="root"/> Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.
Contraseña	<input type="password" value="root1234"/> Muy débil
Confirma la contraseña	<input checked="" type="checkbox"/> Confirma el uso de una contraseña débil.
Tu correo electrónico	<input type="text" value="root@gmail.com"/> Comprueba bien tu dirección de correo electrónico antes de continuar.
Visibilidad en los motores de búsqueda	<input type="checkbox"/> Pedir a los motores de búsqueda que no indexen este sitio Depende de los motores de búsqueda atender esta petición o no.

Aquí tenemos la confirmación de que Wordpress se ha instalado. Por lo que ya tendríamos desplegado el proyecto.



Hago la comprobación desde el navegador.



Y compruebo el sitio está desplegado que está definitivamente desplegado



Configuración y Despliegue de Certificados HTTPS

Certificados de Seguridad

Los certificados de seguridad permiten que la información transmitida entre el usuario y el sitio web esté protegida y cifrada, manteniendo la privacidad de los datos, proporcionan una forma de verificar la identidad de la página, detectan cualquier intento de manipulación de la información durante la transmisión y son necesarios para habilitar el protocolo HTTPS.

Generación del certificado en Kali linux con openssl

Con los siguientes comandos genero un certificado auto-firmado (certificate.crt) con una clave RSA.:

```
openssl genpkey -algorithm RSA -out private.key
```

```
openssl req -new -key private.key -x509 -days 365 -out certificate.crt
```

```
(kali㉿kali)-[~]  
$ sudo su  
[sudo] password for kali:  
(root㉿kali)-[/home/kali]  
# openssl genpkey -algorithm RSA -out private.key  
openssl req -new -key private.key -x509 -days 365 -out certificate.crt
```

Estos comandos generan una clave privada RSA y un certificado auto-firmado que es válido por 365 días. Durante la generación el sistema hace algunas preguntas para completar la información del certificado (país, estado, organización, etc.).

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [AU]:es  
State or Province Name (full name) [Some-State]:.  
Locality Name (eg, city) []:.  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:hitocampus  
Organizational Unit Name (eg, section) []:.  
Common Name (e.g. server FQDN or YOUR name) []:localhost  
Email Address []:.
```

comprobamos que se ha generado un certificado que es un fichero llamado **certificate.crt**. y la clave privada.

```
(root@kali)-[/home/kali]
# ls
certificate.crt  Documents  Music      private.key  Templates
Desktop         Downloads  Pictures    Public       Videos
```

Configuración para Apache:

Muevo los archivos a un directorio accesible por Apache y me doy cuenta que en la estructura ya existe un directorio para esos fines:.

```
(root@kali)-[/etc/apache2]
# ls
apache2.conf      conf-enabled  mods-available  sites-available
apache2.conf.save  contrasena    mods-enabled     sites-enabled
certificate.crt    envvars       ports.conf
conf-available    magic         private.key

(root@kali)-[/etc/apache2]
# mv certificate.crt /etc/ssl/private

(root@kali)-[/etc/apache2]
# mv private.key /etc/ssl/private
```

Abre el archivo de configuración de Apache correspondiente a wordpress.pero primero hago una copia de seguridad. En **/etc/apache2/sites-available/000-default-ssl.conf** y realizamos las siguientes configuraciones para indicar donde estan los certificados y wordpress que se encuentra en:

```
(root@kali)-[/var/www/html]
# ls
index.nginx-debian.html  wordpress-6.4.1-es_ES.zip
index.html               wordpress
```

Modificaciones en el archivo quedan reflejadas de la siguiente manera: pero antes creo la carpeta ssl en apache2 y muevo los archivos

```
(root@kali)-[/etc/apache2]
# sudo mkdir /etc/apache2/ssl
```

```
(root@kali)-[/etc/apache2]
# mv /home/kali/certificate.crt /etc/apache2/ssl/
```

```
(root@kali)-[/etc/apache2/ssl]
# mv /home/kali/private.key /etc/apache2/ssl/
```

```
File Actions Edit View Help
GNU nano 7.2 /etc/apache2/sites-available/default-ssl.conf *
<VirtualHost *:443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html/wordpress
    SSLCertificateFile /etc/apache2/ssl/certificate.crt
    SSLCertificateKeyFile /etc/apache2/ssl/private.key

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf

    # SSL Engine Switch:
```

Me aseguro que el módulo SSL está funcionando correctamente ejecuto:

```

(root@kali)-[/home/kali]
# a2enmod ssl /etc/apache2
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
systemctl restart apache2
(root@kali)-[/home/kali]
# systemctl restart apache2
(root@kali)-[/home/kali]
# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
systemctl reload apache2
(root@kali)-[/home/kali]
# systemctl reload apache2

```

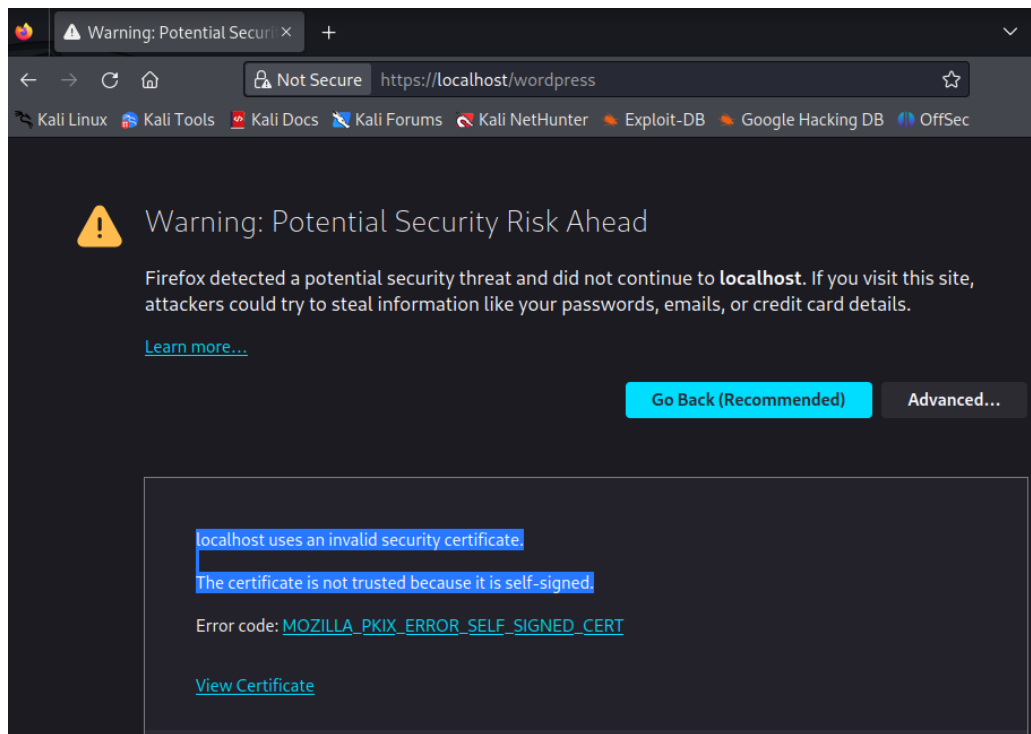
Reinicio el Apache

```

(root@kali)-[/etc/apache2/sites-available]
# sudo service apache2 restart

```

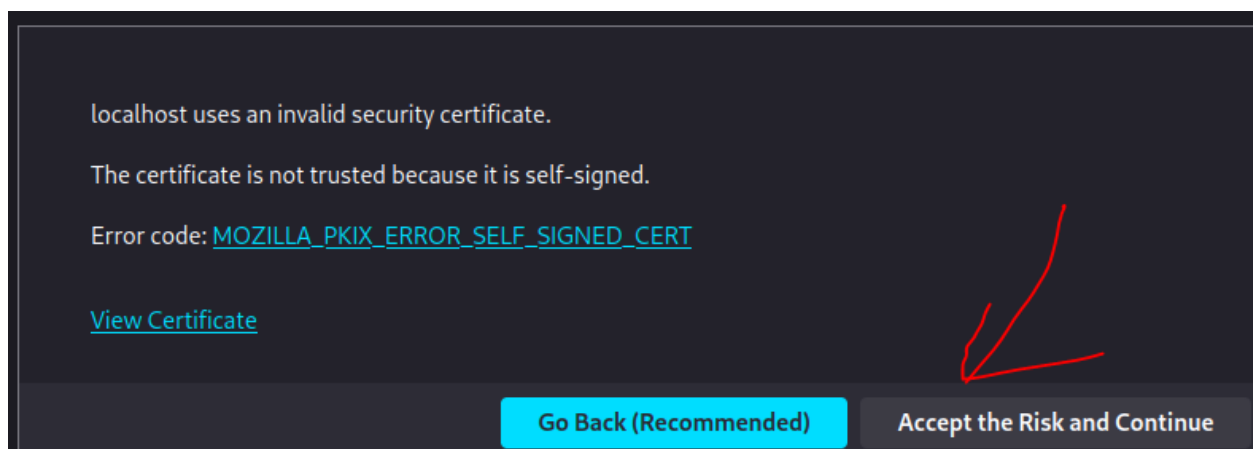
Compruebo si está funcionando correctamente desde el navegador. Y abro la pag esta vez con HTTPS y sale el siguiente mensaje:



Esto me parece correcto: indica que el certificado SSL para habilitar HTTPS en el localhost es autofirmado (bien para entornos de desarrollo), pero. Los navegadores web muestran este mensaje porque los certificados autofirmados **no son verificados por una autoridad de certificación de confianza.**

Para que no salga este mensaje todas la veces hacemos lo siguiente:

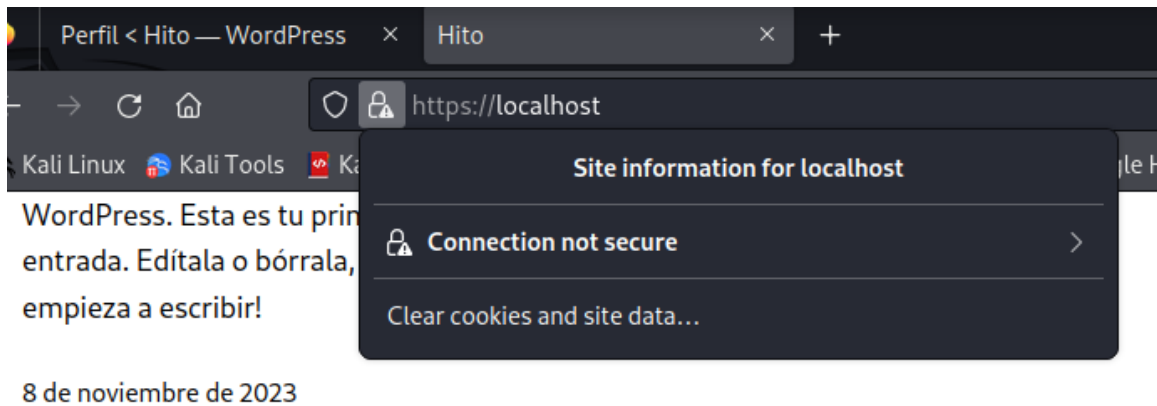
Como el navegador es Firefox :



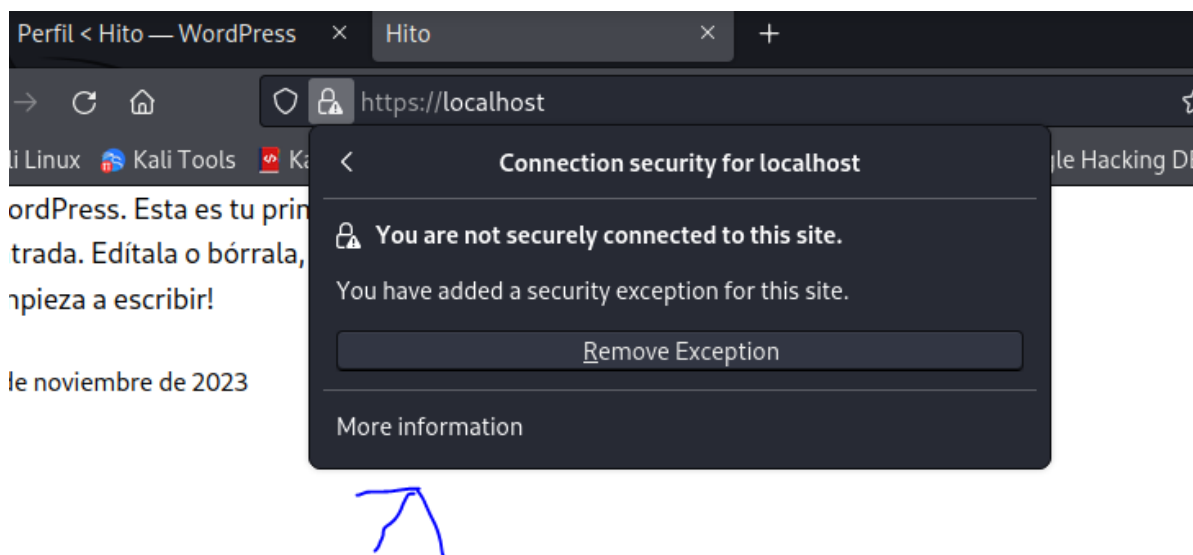
Haz clic en "Entiendo los riesgos".

Sigue las instrucciones para agregar la excepción de seguridad.

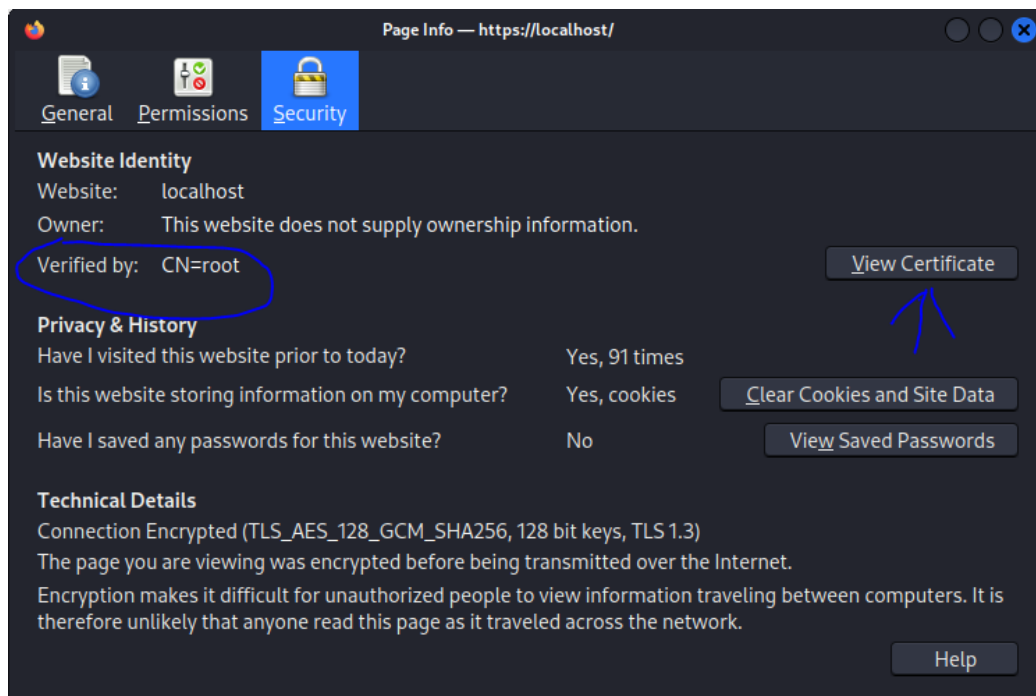
Le das a connection not secure



A more information para ver cual es nuestro certificado



aquí podemos ver toda la información del certificado de seguridad



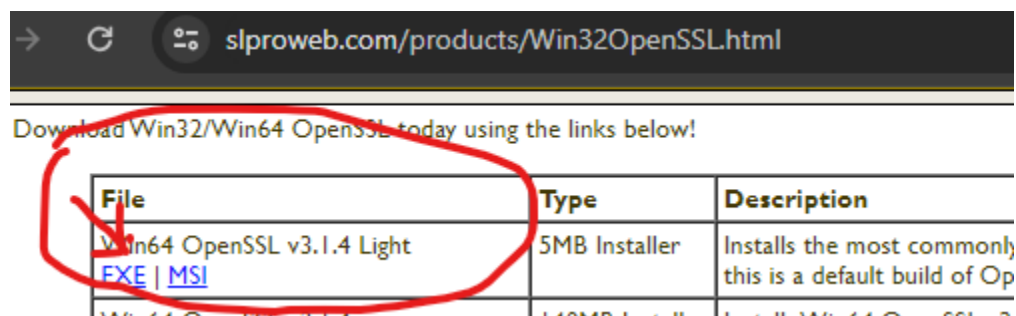
Common Name	root
Issuer Name	
Common Name	root
Validity	
Not Before	Mon, 21 Aug 2023 18:57:43 GMT
Not After	Thu, 18 Aug 2033 18:57:43 GMT
Subject Alt Names	
DNS Name	root
Public Key Info	
Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	B6:D7:E9:11:C8:4E:D6:12:65:9A:ED:2E:44:A5:E9:EB:D6:BD:02:DE:00:F1:CA:4...

Certificado de seguridad en node Windows

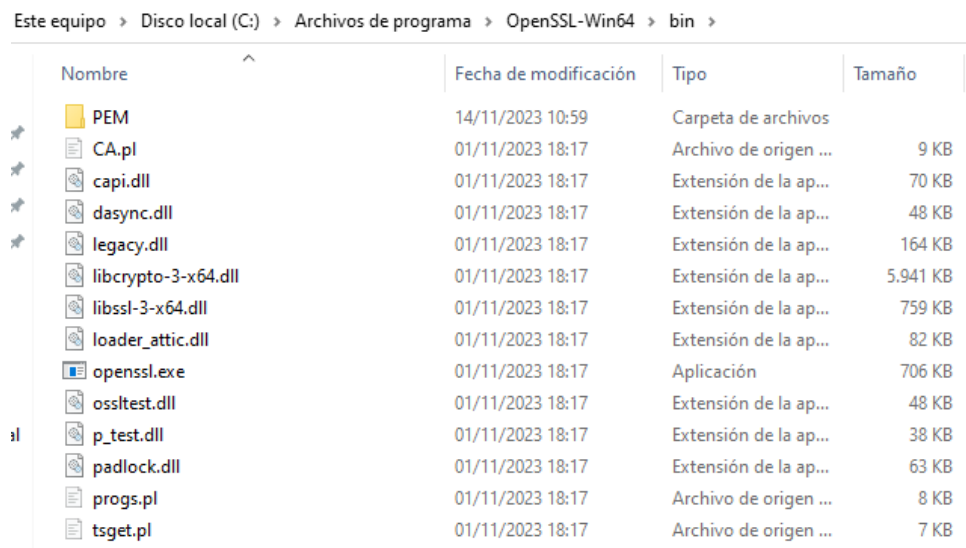
Instalación de Openssl

OpenSSL es una biblioteca de código abierto que proporciona implementaciones de protocolos criptográficos como SSL (Secure Sockets Layer) . Esta biblioteca incluye funciones y utilidades que permiten a los desarrolladores integrar seguridad en sus aplicaciones y servicios, ofreciendo cifrado, autenticación y otras operaciones criptográficas esenciales para garantizar la confidencialidad e integridad de la información transmitida a través de redes, especialmente en el contexto de Internet.

Primero nos vamos a la página web para instalarnos el Openssl y nos instalamos la versión más reciente



una vez instalado nos vamos a la carpeta OpenSSL-Win64 y vamos a la ruta que pone en la siguiente dirección de la imagen y la copiamos



luego nos abrimos el cmd como administrador y nos vamos a la carpeta con este comando

```
C:\WINDOWS\system32>cd C:\Program Files\OpenSSL-Win64\bin
```

Introducir el comando para generar las claves

```
openssl req -nodes -x509 -newkey rsa:4096 -keyout server.key -out server.cer -days 365 -subj /CN=*.bioxor.net
```

A mi me daba un error el cual era que OpenSSL no puede encontrar el archivo de configuración openssl.cnf en la ubicación C:\Program Files\PostgreSQL\psqlODBC\etc\openssl.cnf.

```
C:\Program Files\OpenSSL-Win64\bin>openssl req -nodes -x509 -newkey rsa:4096 -keyout server.key -out server.cer -days 365 -subj /CN=*.bioxor.net
Can't open "C:\Program Files\PostgreSQL\psqlODBC\etc\openssl.cnf" for reading, No such file or directory
882F0000:error:80000002:system library:BIO_new_file:No such file or directory:crypto\bio\bss_file.c:67:calling fopen(C:\Program Files\PostgreSQL\psqlODBC\etc\openssl.cnf, r)
882F0000:error:10000080:BIO routines:BIO_new_file:no such file:crypto\bio\bss_file.c:75:
```

Para solucionar este error tuve que ir a la ruta y crear el archivo que me pedia con el block de notas:

Disco local (C:) > Archivos de programa > PostgreSQL > psqlODBC > etc				
Nombre	Fecha de modificación	Tipo	Tamaño	
openssl.cnf	15/11/2023 8:05	Archivo CNF	1 KB	

Investigando como debía hacer la configuración encontré la siguiente configuración:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
x509_extensions = v3_ca

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = US
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = YourState
localityName = Locality Name (eg, city)
localityName_default = YourCity
organizationName = Organization Name (eg, company)
organizationName_default = YourOrganization
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_max = 64

[ req_ext ]
subjectAltName = @alt_names

[ v3_ca ]
basicConstraints = CA:TRUE
subjectAltName = @alt_names

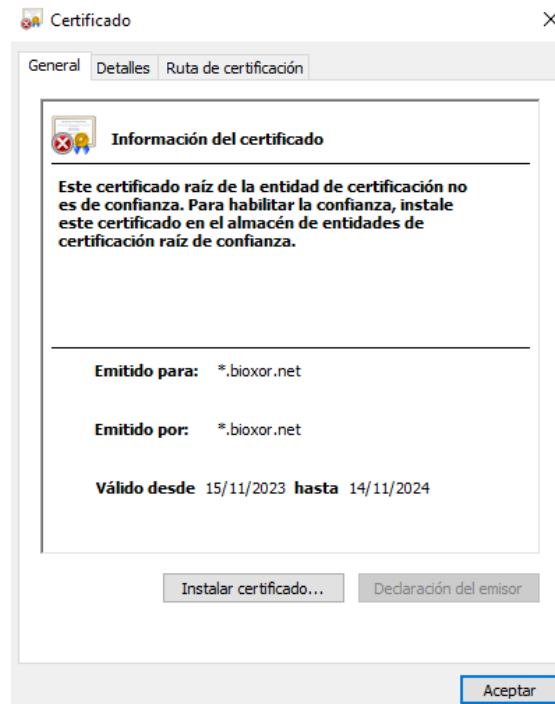
[ alt_names ]
DNS.1 = localhost
```

Explicación del archivo de configuración (basicamente contiene los datos que identifican al sitio web), la mayoría son opcionales:

- ☐ `default_bits = 2048`: Establece el tamaño predeterminado de los bits para las claves generadas en 2048 bits. Este es el tamaño comúnmente utilizado para claves RSA.
- ☐ `distinguished_name = req_distinguished_name`: Hace referencia a la sección `req_distinguished_name` que contiene información sobre el sujeto del certificado. Como estamos en un entorno de pruebas no es necesario poner nada.
- ☐ `req_extensions = req_ext`: Hace referencia a la sección `req_ext` que contiene extensiones para la solicitud de certificado. También es un parametro opcional.
- ☐ `x509_extensions = v3_ca`: Hace referencia a la sección `v3_ca` que contiene extensiones para el certificado X.509.
- ☐ `countryName`: País donde esta la empresa. Aquí podríamos poner España por ejemplo.
- ☐ `StateOrProvinceName`: Estado o provincia de la empresa .
- ☐ `LocalityName`: Localidad o ciudad.
- ☐ `organizationName`: Nombre de la organización por ejemplo COMPANY SL.
- ☐ `commonName`: Nombre común del sujeto, generalmente se usa para el nombre de dominio.
- ☐ `[v3_ca]`: Define extensiones para el certificado X.509.
- ☐ `basicConstraints = CA:TRUE`: Indica que este certificado es una Autoridad de Certificación (CA) y puede firmar otros certificados. Aunque esto no es realmente cierto al ser este un certificado autofirmado.
- ☐ `subjectAltName = @alt_names`: Hace referencia a la sección `alt_names` que define los nombres alternativos como DNS.
- ☐ `DNS.1 = localhost`: Especifica un nombre de dominio alternativo (DNS)

Una vez creado el fichero `.conf`, volví a introducir el comando para poder crear los certificados de seguridad y se crearon correctamente.

Si haces doble clic en el fichero del certificado observamos las propiedades y podemos ver que tiene validez de 1 año desde que lo he creado y que lo ha emitido bioxor.net como Autoridad de certificación.:



Coloco los archivos dentro de la carpeta de mi proyecto

```
EXPLORER:....  server.cer x
> node_modules
> public
index.js
package-lock.json
package.json
server.cer
server.key

server.cer
1  -----BEGIN CERTIFICATE-----
2  MIIFDzCCAvegAwIBAgIUdxTMXiRob80q9dROXUX0CTPXnW0wDQYJKoZIhvcNAQEL
3  BQAwFzEVMBMGA1UEAwMKi5iaW94b3IubmV0MB4XDTIzMTExNTA5MDAwNl0XDTI0
4  MTEwNDA5MDAwNl0wFzEVMBMGA1UEAwMKi5iaW94b3IubmV0MIICIjANBgkqhkiG
5  9w0BAQEFAAOCAg8AMIICgKCAgEAn250sen1PAsT7rr3WdZsKwdioQeZxwRJ+jvJ
6  AbQKMIDt3pUtSYfV487jnB4HElSMksNHM3G00jbvW018rfaoXlZ3wWklKF1qT/Zp
7  ke2fbbnzhfDBMc55EfrThEI8u1M30ApeFSXHJHJTk3emw8y9J52g86KrcsKoi
8  2+C+2GeW825FUDfapAYCFES26G8LMCwt+gq2ZaH8ZpYjQjokpEXP8QKJzbIhZxw2
9  3di7JNKA0AxpK+SmWNHnwmF12v5HJSwE+wHM6ULQRjSU/SX4++2D+1YfLVR+CpNe
10 serOiFcjZRineBTuvAQ3CnXcqQqZNZ+0kvFE75IBYqTW74C6d3Do1Ioz1DiTAWPa
```

Normalmente los servidores que utilizan https usan el puerto 443, modifíco este parámetro en mi archivo index.js sustituyendo el 80 que estaba por defecto.

Además le tengo que especificar al arrancar el servidor donde estan los archivos para que sea https con el siguiente código:

```
// Arranca el servidor
https.createServer({
  cert:fs.readFileSync('server.cer'),
  key:fs.readFileSync('server.key')
}, app).listen(port, () => {
  console.log(`Servidor Express escuchando en el puerto ${port}`);
});
```

¿Qué ocurre cuando abro mi sitio https con ese certificado instalado? pues que aparece este mensaje:



La conexión no es privada

Es posible que los atacantes estén intentando robar tu información de **localhost** (por ejemplo, contraseñas, mensajes o tarjetas de crédito). [Más información](#)

NET::ERR_CERT_AUTHORITY_INVALID



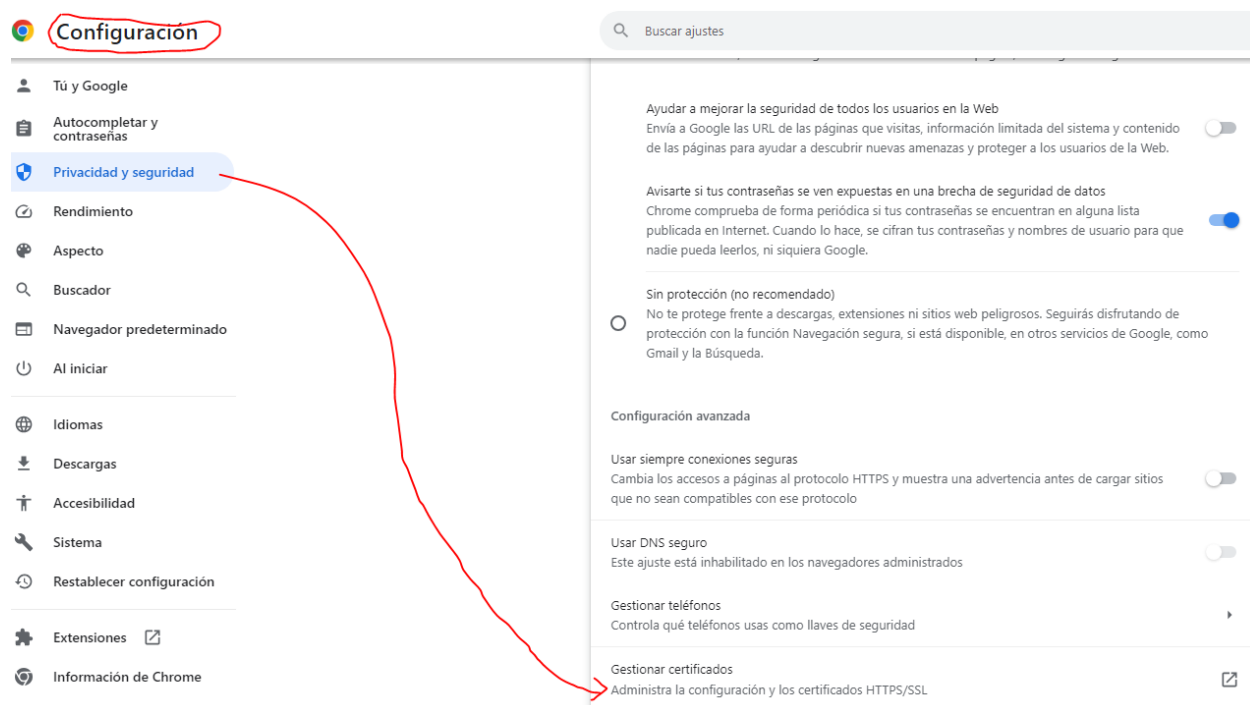
Para disfrutar del máximo nivel de seguridad en Chrome, [activa la protección mejorada](#).

Configuración avanzada

Volver para estar a salvo

Hay que tener en cuenta que Instalar un certificado autofirmado (self-signed) tiene estas implicaciones en términos de seguridad, y **una de ellas es que nuestro navegador no tiene a bixor registrado como CA** autoridad de certificación y por eso responde con ese mensaje al usuario final.

En la configuración del navegador un usuario podría ver lo que significa ese error:



Significado de cada símbolo de seguridad

Estos símbolos indican si Chrome ha establecido o no una conexión segura y privada con un sitio.

🔒 Predeterminado (Seguro)

ℹ Información o No es seguro

⚠ No es seguro o Peligroso

Te recomendamos que no introduzcas información personal ni privada en esta página. Si es posible, no utilices este sitio web.

No es seguro: ve con cuidado. La conexión de este sitio tiene algún problema de privacidad. Es posible que otro usuario pueda encontrar la información que envías o recibes a través de este sitio.

Peligroso: no utilices este sitio. Si aparece una pantalla de advertencia en color rojo que ocupa toda la página, significa que la función **Navegación segura** ha marcado el sitio como no seguro. El sitio podría hacer un uso inadecuado o abusivo de la información que recibe. También podría intentar instalar software dañino en tu ordenador. El uso de este sitio pone en riesgo tu privacidad y tu seguridad.

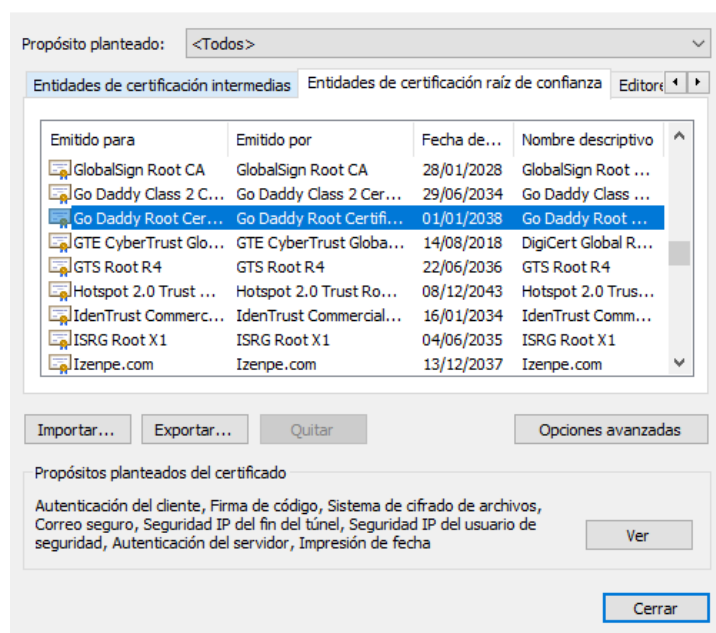
Sin embargo, a pesar de ser un mensaje llamativo en nuestro caso el certificado SSL/TLS que hemos instalado proporciona cifrado de extremo a extremo, lo que significa que la información transmitida entre el cliente y el servidor está protegida y no puede ser fácilmente interceptada por terceros.

Incluso con un certificado autofirmado, la conexión sigue siendo autenticada, lo que significa que el usuario puede confiar en que está conectado al servidor correcto y no a un servidor malicioso.

Pero existen estas desventajas:

Falta de Validación por una Autoridad de Certificación (CA): los certificados autofirmados como su nombre indica los emitimos nosotros con el proceso descrito anteriormente **NO están validados por una Autoridad de Certificación de confianza**. Este aspecto es importante por que estas actúan de **notario** dando la garantía de que la entidad que posee el certificado sea quien dice ser, por qué ha pasado un proceso de identificación.

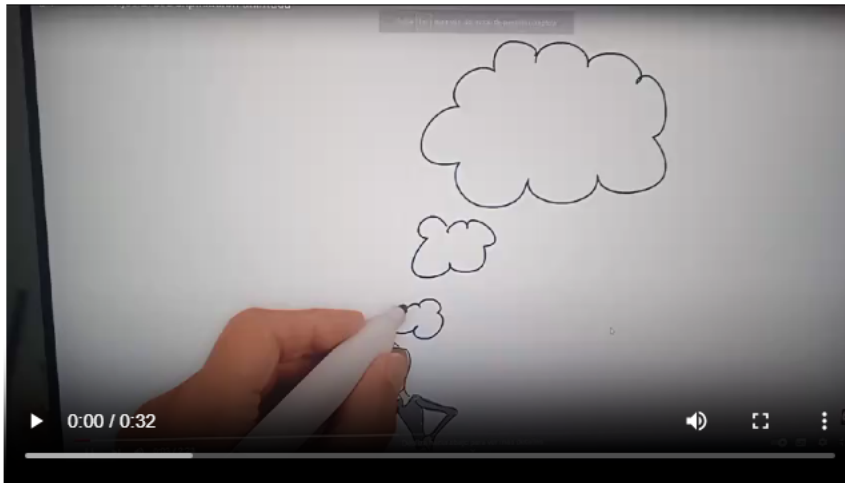
Por eso para poner en producción nuestro sitio es mejor utilizar Certificados de una CA de Confianza:



Aquí muestro el despliegue funcionando en https.



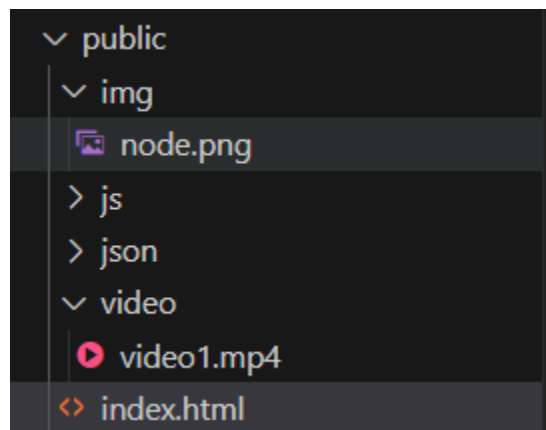
Funcionando



Validación Multimedia y Seguridad en el Desarrollo de la Aplicación

Visualización de formatos de imagen y vídeo

Para que en la página se pudieran visualizar el video y la imagen tuve que poner el video y la imagen en la carpeta public



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>Funcionando</h1>
  <video width="640" height="360" controls>
    <source src="video/video1.mp4" type="video/mp4">
  </video>
  
</body>
</html>
```

Archivo de arranque

Para verificar que el archivo de arranque se cargue correctamente y que la aplicación inicie sin problemas he implementado en el archivo index.js medidas para gestionar posibles errores durante el proceso de arranque y proporcionar mensajes informativos en caso de fallos.

Para ello he creado un archivo de arranque que muestro a continuación

```
// Configuraciones directas
const config = {
  database: {
    username: "sergio",
    password: "sergio"
  },
  // Otras configuraciones aquí...
};

// Lógica de inicialización
const initializeApp = () => {
  try {
    // Accede a la configuración específica, por ejemplo, datos de la base de datos
    const databaseUsername = config.database.username;
    const databasePassword = config.database.password;

    // Arranca el servidor después de la carga exitosa
    https.createServer({
      cert: fs.readFileSync('server.cer'),
      key: fs.readFileSync('server.key')
    }, app).listen(port, () => {
      console.log(`Servidor Express escuchando en el puerto ${port}`);
    });
  } catch (error) {
    // Manejo de errores durante el arranque
    console.error('Error durante el arranque:', error.message);

    // Puedes mostrar mensajes informativos al usuario
    console.error('La aplicación no pudo iniciar correctamente. Por favor, inténtalo de nuevo.');
```

// También puedes redirigir a una página de error o realizar otras acciones necesarias

```
  }
};

// Inicializar la aplicación
initializeApp();
```

Aquí estoy creando un objeto llamado **config** que almacena configuraciones para mi aplicación. En este caso, he incluido credenciales de base de datos, pero podría haber agregado más configuraciones según lo que necesite mi aplicación.

```
// Configuraciones directas
const config = {
  database: {
    username: "sergio",
    password: "sergio"
  },
  // Otras configuraciones aquí...
};
```

Ahora estoy definiendo una función llamada `initializeApp`, como punto de inicio para configurar mi aplicación. Dentro de ella, arranco el servidor con las configuraciones específicas (como las credenciales de la base de datos, indicar donde está el certificado para el server sea https y defino el puerto 443 como puerto seguro).

En caso de algún error durante este proceso (lo manejo en el bloque catch), el cual imprimirá un mensaje de error diciendo básicamente la aplicación no se pudo iniciar correctamente.

```
// Lógica de inicialización
const initializeApp = () => {
  try {
    // Accede a la configuración específica, por ejemplo, datos de la base de datos
    const databaseUsername = config.database.username;
    const databasePassword = config.database.password;

    // Arranca el servidor después de la carga exitosa
    https.createServer({
      cert: fs.readFileSync('server.cer'),
      key: fs.readFileSync('server.key')
    }, app).listen(port, () => {
      console.log(`Servidor Express escuchando en el puerto ${port}`);
    });
  } catch (error) {
    // Manejo de errores durante el arranque
    console.error('Error durante el arranque:', error.message);

    // Puedes mostrar mensajes informativos al usuario
    console.error('La aplicación no pudo iniciar correctamente. Por favor, inténtalo de nuevo.');
```

// También puedes redirigir a una página de error o realizar otras acciones necesarias

```
  }
};

// Inicializar la aplicación
initializeApp();
```

Protección de archivos de configuración

Librería Crypto de Node

La librería `crypto` en Node.js proporciona funciones para realizar operaciones criptográficas de cifrado Hashing. En una aplicación web, puedes utilizar cifrado y hashing para varios propósitos relacionados con la seguridad. Aunque en este proyecto al instalar un servidor https ya ciframos las comunicaciones entre el navegador del usuario y el servidor, podría utilizarse para:

- ☐ Almacenar contraseñas de usuarios de forma segura mediante funciones hash.
- ☐ Comprobación de integridad de datos, puedes coger un dato y pasarlo por un hash y si este se ha modificado o ha quedado incompleto podrías saberlo automáticamente si realizas la comparación del hash del dato previo u una vez este se ha transmitido. Si hubiese cambiado ese hash podrías decir que el dato no esta integro o que ha sido modificado ya que el hash deja huella.
- ☐ Cifrar información sensible antes de almacenarla en cookies si necesitas almacenar datos en el lado del cliente.
- ☐ Otra práctica muy común es cifrar los datos sensibles de la base de datos para que estos sean anónimos y no puedan leerse sin ser descifrados con una clave que solo posee un administrador. Por ejemplo datos médicos o biométricos.

Cifrado del archivo de configuración.

Para poder proteger mi archivo de configuración, instale la biblioteca **crypto** de Node.js usando npm.

```
C:\Users\Sergio\Desktop\Sergio\Hitodespliege>npm install crypto
up to date, audited 66 packages in 4s
11 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```


Una vez instalado , creo un archivo `encryptConfig.js`, en el cual:

- Detallo el fichero que deseo encriptar, en el ejemplo `config.json`
- Genero una clave secreta que utilizo para encriptar el archivo. Solo el que tenga esa clave podrá desencriptar
- Encriptar el archivo utilizando dicha clave
- Por último guardo la clave en el archivo encriptado.

```
js > JS encryptConfig.js > ...
const fs = require('fs');
const crypto = require('crypto');

// Lee el archivo de configuración no encriptado
const configFile = fs.readFileSync(__dirname + '/config.json', 'utf-8');

// Genera una clave secreta para la encriptación
const secretKey = crypto.randomBytes(32); // Puedes ajustar la longitud según tus necesidades

// Crea un objeto de cifrado con la clave secreta
const cipher = crypto.createCipher('aes-256-cbc', secretKey);

// Encripta el archivo de configuración
const encryptedConfig = cipher.update(configFile, 'utf-8', 'hex') + cipher.final('hex');

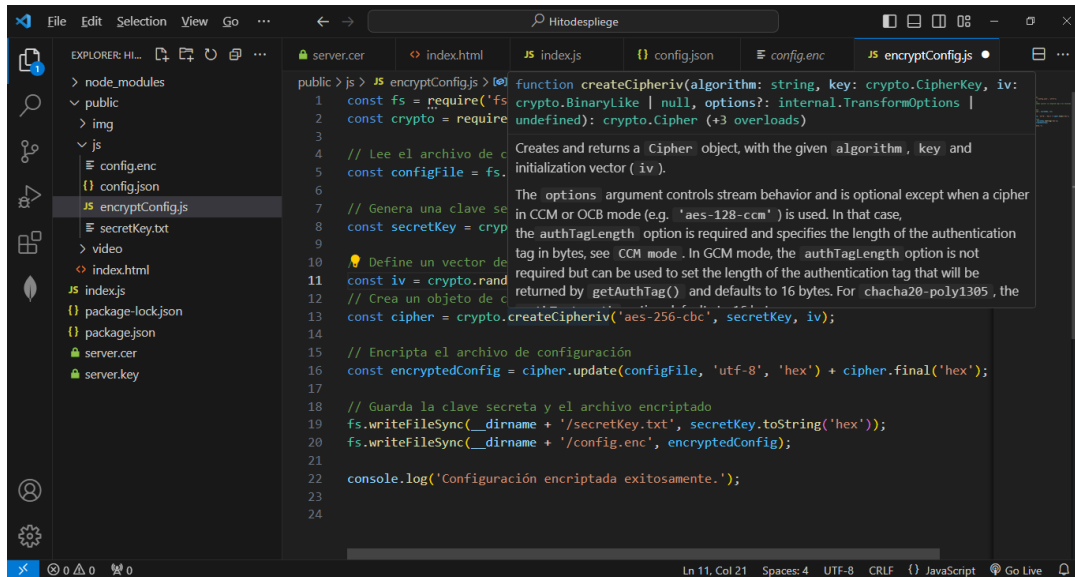
// Guarda la clave secreta y el archivo encriptado
fs.writeFileSync(__dirname + '/secretKey.txt', secretKey.toString('hex'));
fs.writeFileSync(__dirname + '/config.enc', encryptedConfig);

console.log('Configuración encriptada exitosamente.');
```

Lo lanzó y encriptó con éxito el fichero, aunque me doy cuenta de que la función `createCipher` ha quedado obsoleta según la documentación de node.

```
C:\Users\Sergio\Desktop\Sergio\Hitodespliege>node public/js/encryptConfig.js
Configuración encriptada exitosamente.
(node:9560) [DEP0106] DeprecationWarning: crypto.createCipher is deprecated.
(Use `node --trace-deprecation ...` to show where the warning was created)
```

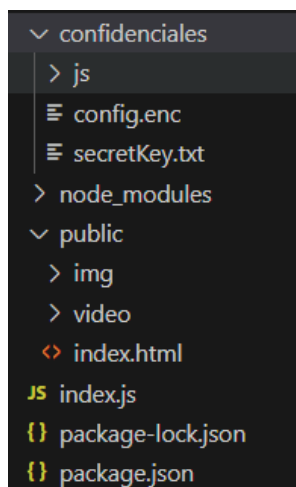
Modifico el código siguiendo las instrucciones de la documentación y vuelvo a lanzar la encriptación.



```
public > js > JS encryptConfig.js > 10
1  const fs = require('fs');
2  const crypto = require('crypto');
3
4  // Lee el archivo de configuración
5  const configFile = fs.readFileSync('config.json', 'utf-8');
6
7  // Genera una clave secreta
8  const secretKey = crypto.randomBytes(32);
9
10 // Define un vector de inicialización (iv)
11 const iv = crypto.randomBytes(16);
12 // Crea un objeto de cifrado
13 const cipher = crypto.createCipheriv('aes-256-cbc', secretKey, iv);
14
15 // Encripta el archivo de configuración
16 const encryptedConfig = cipher.update(configFile, 'utf-8', 'hex') + cipher.final('hex');
17
18 // Guarda la clave secreta y el archivo encriptado
19 fs.writeFileSync(__dirname + '/secretKey.txt', secretKey.toString('hex'));
20 fs.writeFileSync(__dirname + '/config.enc', encryptedConfig);
21
22 console.log('Configuración encriptada exitosamente.');
```

```
C:\Users\Sergio\Desktop\Sergio\Hitodespliege>node public/js/encryptConfig.js
Configuración encriptada exitosamente.
```

Para verificar que todo ha ido correctamente, hay que asegurarse de que los archivos `secretKey.txt` y `config.enc` hayan sido creados en el directorio especificado. Estos archivos contienen la clave secreta y la configuración encriptada, respectivamente. Y los he creado en una carpeta NO publica que he nombrado `confidenciales`.



Inspección de directorios

El directorio `public` como su nombre indica es público luego, ahora me dispongo a proteger el contenido de la carpeta `confidenciales` y el resto del contenido.

Antes de protegerlo trato de inspeccionar la carpeta `confidenciales` para ver que ocurre y obtengo el siguiente mensaje:



Desactivar la generación de listados de directorios

El mensaje "`Cannot GET /confidenciales`" indica que estás intentando acceder a la ruta `/confidenciales` en tu aplicación, pero no se ha definido ninguna rutina para manejar esa solicitud específica.

Para evitar la inspección de directorios y proporcionar una capa adicional de seguridad, voy a configurar el servidor para evitar que se muestren los contenidos de los directorios.

En el caso de un servidor Node.js con Express, puedes lograr esto utilizando el middleware `express.static`.

`express.static` con la opción `{ index: false }` se utiliza para desactivar la generación automática de listados de directorios. Esto significa que si alguien intenta acceder directamente a la URL de un directorio, no se mostrará una lista de archivos en ese directorio.

```
// Configuración para evitar la inspección de directorios
app.use('/confidenciales', express.static(path.join(__dirname, 'confidenciales'), { index: false }));
```

Manejo del error 403

Cuando un usuario trata de acceder a un directorio o a una página específica que tenga error de permisos, vamos a manejar el **error 403 Forbidden**

En mi práctica voy a utilizar, **app.use** para interceptar cualquier solicitud a la ruta **"/confidenciales"**. Esto mismo podríamos utilizar para cualquier directorio que creamos y que no deseáramos que fuera público. **app.use** no solo intercepta la solicitud, también envía una respuesta con un código de estado 403 (**Acceso prohibido**) y aprovecho a sacar un mensaje indicando que el acceso está prohibido.

Aquí está el código:

```
app.use('/confidenciales', (req, res, next) => {  
  res.status(403).send('Acceso prohibido. No tienes permisos para acceder a esta ruta.');
```

A continuación, muestro el resultado de este paso.

