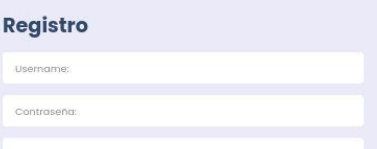
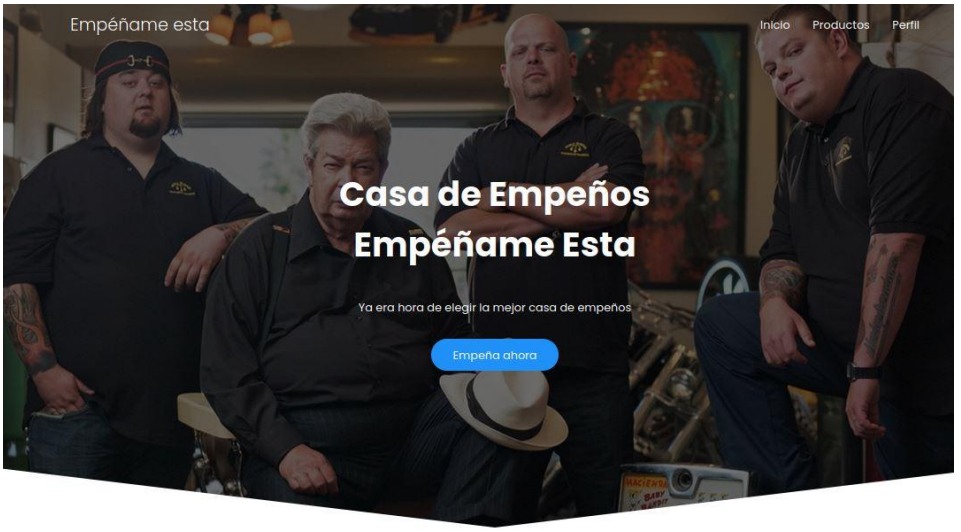
Casa de Empeños - Empéñame Esta

Enlace a Github:
https://github.com/EderSampayo/EmpenameEsta-SGSSI/tree/entrega_2

¿Cómo se utiliza el sistema?

La primera vez que entras en el sistema, te redireccionará a la página de **inicio de sesión**, donde hay que insertar el usuario y la contraseña. En el caso de no tener una cuenta, tendrías que pulsar en el botón de '**Registrarse**' e introducir los datos respetando el formato de cada uno de ellos.





Después de registrarse o loguearse, se te redireccionará a la página **principal**, donde podrás leer información sobre nuestra casa de empeños. Tanto en la parte superior a la derecha como en la parte inferior a la izquierda de la Web, tienes varias opciones.

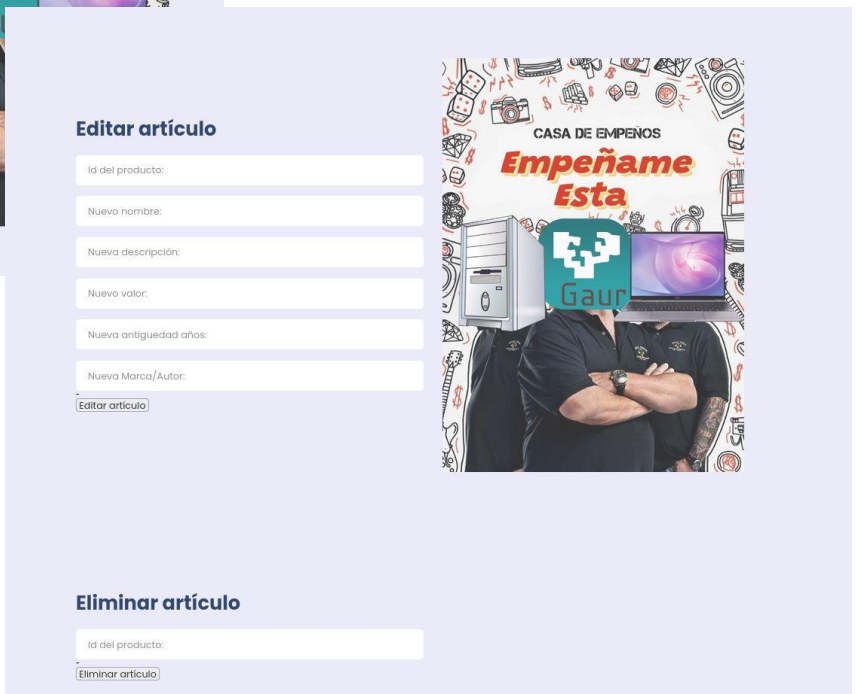
La primera es ‘Inicio’, la cual te llevará a la página principal.

La segunda es ‘Productos’, que de ser pulsada te llevará a la página de productos.

La tercera es ‘Perfil’, la cual te llevará a los datos de tu perfil, es decir, los datos que has insertado en el registro.

Por un lado, al entrar en la pestaña de Productos, podrás consultar todos los artículos que están registrados en la web, además de poder añadir y eliminar los artículos que desees.

Tanto para añadir como para eliminar productos tendrás que completar los campos utilizando el formato correcto.



Por otro lado, al entrar en la pestaña de Perfil, podrás consultar todos tus datos, y si quisieses editar alguno de ellos, tendrías que completar los campos para poder actualizarlos.

Entrega 2: Auditoría Sistema Web

Las auditorías se han hecho de dos formas:

1. PowerPoint del apartado seguridad web
2. Uso de OWASP ZAP

En el primer caso se ha ido apartado por apartado mirando las debilidades que podíamos encontrar en nuestra web y se han anotado para como se ve en los apartados que se encuentran a continuación arreglarlas.
En el caso de OWASP ZAP se ha utilizado para ver las diferencias de riesgos entre la primera entrega y la segunda, como se puede ver en el apartado de monitorización de la seguridad se ha reducido el riesgo eliminando todos los avisos que suponían un riesgo medio.
Para realizar esta comprobación primero se instaló OWASP ZAP, después se ejecutó el init.sh que se encuentra en la carpeta EmpenameEsta-SGSSI para iniciar nuestra web y para finalizar se insertó en el OWASP ZAP la dirección localhost:81 para comenzar el ataque.

Se han solucionado las siguientes vulnerabilidades:

Rotura de control de acceso:

Se ha creado un log para controlar los intentos de inicio de sesión a la página web. Cuando alguien introduce un usuario y contraseña, quedan registrados el usuario, ip, fecha y hora y resultado de la operación (es decir, si el login ha sido exitoso o no).

```
User: 172.18.0.1 - November 13, 2022, 12:04 am
Attempt: Failed
User: eder
-------------------------
User: 172.18.0.1 - November 13, 2022, 12:04 am
Attempt: Failed
User: eder
-------------------------
```

```php
//ENTREGA 2 (LOG DE ACCESOS)
//Something to write to txt log
$log  = "User: ".$_SERVER['REMOTE_ADDR'].' - '.date("F j, Y, g:i a").PHP_EOL.
"Attempt: ".($result[0]['success']=='1'?'Success':'Failed').PHP_EOL.
"User: ".$username.PHP_EOL.
"-------------------------".PHP_EOL;
//Save string to log, use FILE_APPEND to append.
file_put_contents('./Logs/log_'.date("j.n.Y").'.log', $log, FILE_APPEND);
//ENTREGA 2 (LOG DE ACCESOS)
```

Además, se ha añadido tokenización al sistema. Por ejemplo, en iniciosesion.php:

```php
<input type="hidden" name="CSRF_token" value="<?php echo $token; ?>"> <?php /*ENTREGA 2 TOKEN */ ?>
```

```php
//ENTREGA 2 (TOKEN)
    $token = md5(uniqid(rand(),true));
    $_SESSION['tokenLogin'] = $token;
//ENTREGA 2 (TOKEN)
```

```php
$_POST['CSRF_token'] == $_SESSION['tokenLogin']
```

Fallos criptográficos:

Para mayor seguridad con las contraseñas, estas han sido almacenadas con hashes y sal. Haciendo uso de las funciones *password_hash* y *password_verify* que vienen implementadas en PHP, cuando se va a guardar una contraseña, se *hashea* y se le añade sal antes de almacenarla en la base de datos.
De esta forma, si alguien obtiene acceso a la BD, no podría hacerse con las contraseñas de los usuarios. Esto se debe a que lo único que se almacenará será un criptograma que representa la contraseña original, y como la función hash no tiene inversa, no se puede obtener el contenido original desde el criptograma.

```php
//ENTREGA 2 (CONTRASEÑA HASHEADA)
$password_hasheada = password_hash($password, PASSWORD_DEFAULT);
```

```php
if(password_verify($password, $contraBD))
```

Para que las conexiones al sitio web sean seguras, se ha usado un certificado de servidor. Se ha cambiado el protocolo HTTP por el HTTPS. Con HTTP podría darse el caso de que, por ejemplo, alguien le robara la cookie de sesión a otro usuario por no estar encriptada. Al iniciar init.sh, se crearán una clave (.key), un certificado (.crt) y la solicitud de firma (.csr). El servidor está creado a través del archivo empenameesta.conf, el cual es parecido al que se utilizó en el laboratorio de apache2 y openSSL.

Inyección:

Para evitar la inyección en nuestra web se han tomado las medidas mostradas a continuación:

- Para comprobar que el usuario sea válido al registrarse, al iniciar sesión y para comprobar que no se ha cambiado indeseadamente se utiliza la función:
  ctype_alpha($username)
  Esta función comprueba si el valor de la variable es una única palabra formada por letras no acentuadas.
- Para comprobar que la contraseña sea válida y aumentar su seguridad, al registrarse se hacen las siguientes comprobaciones:
  - Se pide un tamaño mínimo de 5 caracteres.
  - Debe tener al menos un número.
  - Debe tener al menos una mayúscula.
  - Debe tener al menos una minúscula.
  - Debe tener al menos un carácter especial.

    - La contraseña no puede tener espacios en blanco.

  De esta manera nos aseguramos de que la contraseña no se puede encontrar entre las más inseguras y además evitamos la inyección indeseada.
  Al iniciar sesión no se inserta la contraseña en la BD por lo que no tenemos peligro de inyección.
- En nombre y apellidos se comprueba el string que se recibe mediante la siguiente función:

```
function areOnlyLetters( $mixed, $sCharsPermitidos = " ) : bool
{
        $pattern = "/^[a-zA-Zá-źÁ-Ź" . $sCharsPermitidos . "]+$/";
        return ( preg_match ( $pattern, $mixed ) );
}
areOnlyLetters($nomApe," ")
```

  Esta función comprueba que en nomApe solo hay letras y espacios.
- En DNI se comprueba el string que se recibe mediante la siguiente función para comprobar que es un DNI válido:

```
function es_dni_valido($dni){
        $dni_length = strlen((string)$dni);
        if($dni_length != 9)
        {
                return false;
        }
        if (preg_match("#^[0-9]{8}[A-Z]{1}+$#", $dni))  /* Si tiene el formato correcto */
        {
                $letter = substr($dni, -1);
                $numbers = substr($dni, 0, -1);
                if (substr("TRWAGMYFPDXBNJZSQVHLCKE", $numbers%23, 1) ==
                $letter && strlen($letter) == 1 && strlen ($numbers) == 8 )    /* Si la
                letra corresponde con los números*/
                {
                        return true;
                }
                return false;
        }
}
```

- En el caso del teléfono sólo se permiten números de 9 cifras mediante el uso de strlen y de is_numeric.
- Para el email se comprueba que sea un email válido.
- Para la fecha se comprueba que se inserta una fecha de tipo aaaa-mm-dd.
- Al añadir un producto, en descripción, en nombre del producto y en marca/autor se utiliza la misma función que en el caso de nombre y apellidos para comprobar que no se insertan ni números ni caracteres especiales.
- En valor y en antigüedad se utiliza la función is_numeric para comprobar si es un número o no.
- A la hora de eliminar el artículo y editar sus datos se pide un id que también se comprueba que es válido utilizando la función is_numeric.
- Cuando se quieren cambiar los datos del perfil se realizan las mismas comprobaciones que al registrarse.

- En todos los lugares en los que se hace un select y se utiliza el username se comprueba que no se ha modificado como aparece en el primer punto de este apartado.

Diseño inseguro:

Se ha creado un rate limit de los inicios de sesión, tanto para evitar bots, como para mejorar la seguridad contra los ataques de fuerza bruta y los ataques de denegación de servicio.

```php
session_start();

//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
if(isset($_SESSION["locked"]))
{
    $difference = time() - $_SESSION["locked"];
    if($difference > 10)
    {
        unset($_SESSION["locked"]);
        unset($_SESSION["login_attempts"]);
    }
}
//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
```

```php
//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
    if($_SESSION["login_attempts"] > 2)
    {
        $_SESSION["locked"] = time();
        ?>
        <h3 style="color: red;" class ="ErrorRegistro">Demasiados intentos fallidos. Por favor, espera 10 segundos</h3>
        <?php
    }
    else
    {
        ?>
            <input type="submit" name="InicSesion" value="Iniciar sesión">
        <?php
    }
//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
```

```php
//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
$_SESSION["login_attempts"] += 1;
//ENTREGA 2 (LIMITAR INTENTOS LOGIN)
```

Configuración de seguridad insuficiente:

Para combatir la configuración de seguridad insuficiente se utilizan las cabeceras HTTPS. En el caso de que el atacante quisiera utilizar una cabecera HTTP, el propio programa le redireccionará a HTTPS.

```
<VirtualHost *:80>
      Redirect / https://localhost:444
</VirtualHost>
```

Componentes vulnerables y obsoletos:

Anteriormente en el archivo *docker-compose.yml* se utilizaba la última versión de phpmyadmin sin saber exactamente cuál era (phpmyadmin:latest). Esto ha sido modificado, utilizando una versión fija y conocida para tener mayor control y seguridad.

```
phpmyadmin:
  image: phpmyadmin/phpmyadmin:5.2.0
```

Fallos de identificación y autenticación:

Como se ha mencionado en los apartados anteriores, las contraseñas se almacenan con hash y sal y no se permiten contraseñas básicas (se aplican varias restricciones). Además, las sesiones son invalidadas después de cierto tiempo y los intentos fallidos de login son retrasados (y también registrados en un log).

Para añadir un poco más seguridad en los perfiles de los usuarios, se ha añadido un botón de logout que permite cerrar la sesión en cualquier momento.

```
<input type="submit" name="Logout" value="Logout">
```

```
//ENTREGA 2 (BOTÓN LOGOUT)
if(isset($_POST['Logout']))
{
    session_unset();
    session_destroy();

    echo '<script type="text/javascript">window.location.replace("https://localhost:444/principal.php");</script>';
}
//ENTREGA 2 (BOTÓN LOGOUT)
```

Además, se ha creado un gestor de sesiones, el cual después de haber pasado un tiempo en la web desde que se ha iniciado sesión (en este caso 30 minutos), se invalidará la sesión para evitar que el usuario se dejé la sesión iniciada por inactividad.

En los php que no son login y register:

```
session_start();
if (isset($_SESSION['CREATED']) && (time() - $_SESSION['CREATED'] > 1800)) { //1800s = 30min
    // last request was more than 30 minutes ago
    session_unset();     // unset $_SESSION variable for the run-time
    session_destroy();   // destroy session data in storage
}
if (!isset($_SESSION['user_id'])) {
    echo '<script type="text/javascript">window.location.replace("https://localhost:444/iniciosesion.php");</script>';
}
```

En login y register:

```
session_start();
```

```
  ?>
  <h3 class ="OkRegistro">¡Te has logueado correctamente!</h3>
  <?php
  $_SESSION['user_id'] = $username;

  //ENTREGA 2 (SESIÓN EXPIRADA)
  $_SESSION['CREATED'] = time(); // registra en tiempo con el que se va a comparar si han pasado x minutos
  //ENTREGA 2 (SESIÓN EXPIRADA)
```

Fallos en la integridad de datos y software:

Para evitar los fallos en la integridad de datos y software, como se ha comentado anteriormente, hemos quitado todas las versiones latest y hemos puesto versiones específicas. Por lo tanto, antes de instalar una nueva versión, tendremos que ver los cambios que se han realizado para asegurarnos de que no incluye código malicioso.

Fallos en la monitorización de la seguridad:

Para tener una monitorización adecuada se han almacenado los logins en un registro y se han realizado auditorías con la herramienta de pentesting ZAP.

Fallos encontrados mediante OWASP ZAP en la entrega 1:
Riesgo medio:

- Absence of Anti-CSRF Tokens
- Application Error Disclosure
- Content Security Policy (CSP) Header Not Set
- Hidden File Found
- Missing Anti-clickjacking Header
- Parameter Tampering

Riesgo bajo:

- Cookie No HttpOnly Flag
- Cookie without SameSite Attribute
- Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
- Server Leaks Version Information via "Server" HTTP Response Header Field
- X-Content-Type-Options Header Missing
- Loosely Scoped Cookie
- Modern Web Application
- User Agent Fuzzer
- User Controllable HTML Element Attribute (Potential XSS)

Fallos encontrados mediante OWASP ZAP en la segunda entrega:
Riesgo bajo:

- Absence of Anti-CSRF Tokens
- Application Error Disclosure
- Content Security Policy (CSP) Header Not Set
- Hidden File Found
- Missing Anti-clickjacking Header
- Parameter Tampering
- Cookie No HttpOnly Flag
- Cookie without SameSite Attribute
- Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
- Server Leaks Version Information via "Server" HTTP Response Header Field
- X-Content-Type-Options Header Missing
- Content-Type Header Missing
- Modern Web Application

- User Agent Fuzzer
- User Controllable HTML Element Attribute (Potential XSS)

Se ha mejorado la seguridad, ya que podemos ver como hemos corregido los errores de riesgo medio.

Fallos en la seguridad física:

Antes al intentar iniciar sesión y registrarse se mostraba la contraseña, lo cual permitía que otra persona la pudiese ver, ahora se ha cambiado eso para que aparezcan unos puntos en lugar de los caracteres escritos y se da la posibilidad al usuario de ver la contraseña pulsando un checkbox.

```php
<input type="password" name="Password" placeholder="Contraseña:" class="footer__input" id="inputPassword">
<?php //ENTREGA 2 (OCULTAR PASSWORD) ?>
<input type="checkbox" onclick="ocultarPassword()">Ver Contraseña
<?php //ENTREGA 2 (OCULTAR PASSWORD) ?>
```

js > JS ocultarPassword.js

```javascript
function ocultarPassword() {
    var x = document.getElementById("inputPassword");
    if (x.type === "password") {
      x.type = "text";
    } else {
      x.type = "password";
    }
  }

```

Fuentes para la creación del sistema:

- Primera entrega:

Vídeo utilizado para la estructura html y el diseño de css:
https://www.youtube.com/watch?v=3S_FKLBjOsI

Uso de php y sql:
https://www.youtube.com/watch?v=cka0J41iJY0

Mostrar datos en php a través del SELECT de MySQL:
https://www.youtube.com/watch?v=nPAp-gT5gPI

- Segunda entrega:

Comprobar datos de entrada, por ejemplo: Nombre y apellidos
https://www.codedevelium.com/validar-string-solo-letras/

Formato y algoritmo para validar el DNI:
https://gist.github.com/sjimenez77/4485483
https://www.adaweb.es/validar-dni-con-php/

Validación de Date:
https://stackoverflow.com/questions/19271381/correctly-determine-if-date-string-is-a-valid-date-in-that-format

Función para almacenar contraseñas con hash y sal
https://www.php.net/manual/en/function.password-hash.php

Función para verificar contraseñas almacenadas con hash
https://www.php.net/manual/en/function.password-verify.php

Requisitos en contraseñas:
https://stackoverflow.com/questions/42467243/regex-strong-password-the-special-characters

Expirar sesión:
https://stackoverflow.com/questions/520237/how-do-i-expire-a-php-session-after-30-minutes

Ocultar contraseña:
https://www.w3schools.com/howto/howto_js_toggle_password.asp

Limitar accesos fallidos:
https://www.youtube.com/watch?v=tH7dzGnrSI8&t=74s

Log de logins fallidos:
https://stackoverflow.com/questions/19898688/how-to-create-a-logfile-in-php