Programació Web

??? abstract “Duració i criteris d’evaluació”

```
Duració estimada: 10 hores

<hr />

| Resultat d'aprenentatge  | Criteris d'avaluació  |
| ------                    | -----                 |
| 4. Desenvolupa aplicacions Web embegudes en llenguatges de marques analitzant i incorporar
```

1. Mecanismes per al Manteniment de la Informació en Aplicacions Web

HTTP és un protocol **stateless**, sense estat. En les aplicacions web modernes, és essencial gestionar l’estat del client per proporcionar una experiència d’usuari fluida i personalitzada. Per això, se simula l’estat mitjançant l’ús de cookies, tokens o la sessió. L’estat és necessari per a processos com ara el carret de la compra, operacions associades a un usuari, etc… A continuació es detallen diversos mecanismes per mantenir aquesta informació, així com els seus avantatges i desavantatges.

Cookies

Les **cookies** són petits fitxers de text emmagatzemats al navegador de l’usuari. Són àmpliament utilitzades per mantenir l’estat del client entre sol · licituds HTTP, ja que HTTP és un protocol sense estat.

Avantatges de les Cookies

- **Persistència**: Les cookies poden mantenir-se durant períodes llargs definits per l’atribut `expires` o `max-age`.
- **Accessibilitat**: Es poden accedir des de qualsevol pàgina del mateix domini.
- **Simplicitat**: Fàcils d’implementar i gestionar.

Desavantatges de les Cookies

- **Seguretat**: Poden ser objectiu d’atacs com **XSS (Cross-Site Scripting)** si no es gestionen correctament.
- **Limitacions de mida**: Les cookies estan limitades a 4KB de dades.
- **Privadesa**: Les dades de l’usuari poden ser vulnerables si no es protegeixen adequadament.

Millors Pràctiques

- Utilitzar l'atribut `HttpOnly` per evitar l'accés a les cookies des de JavaScript.
- Configurar `SameSite` per prevenir atacs CSRF.
- Emmagatzemar únicament informació essencial i no sensible.

Sessions

Les **sessions** són un mecanisme que permet associar dades a un usuari durant una sessió específica. Aquestes dades s'emmagatzemen al servidor, mentre que el client només guarda un identificador de sessió.

Avantatges de les Sessions

- **Seguretat**: Les dades sensibles es mantenen al servidor.
- **Capacitat d'emmagatzematge**: No limitades per la mida de les cookies.
- **Compartició de dades**: Facilita la compartició d'estat entre diferents components d'una aplicació web.

Desavantatges de les Sessions

- **Escalabilitat**: Requereix memòria addicional al servidor, la qual cosa pot ser problemàtica amb molts usuaris.
- **Persistència limitada**: Normalment només duren mentre el navegador està obert, tret que es configuri d'una altra manera.

Millors Pràctiques

- Emmagatzemar únicament referències o identificadors a la sessió.
- Utilitzar eines com Redis per a una gestió eficient de sessions en entorns escalables.

Web Storage

El **Web Storage** és una API del navegador que permet emmagatzemar dades al client de manera més senzilla i amb més capacitat que les cookies tradicionals. Inclou dos mecanismes principals: `localStorage` i `sessionStorage`.

Característiques del Web Storage

- `localStorage`: Permet emmagatzemar dades que persisteixen fins que s'elimina explícitament, fins i tot després de tancar el navegador.
- `sessionStorage`: Les dades es mantenen només durant la sessió del navegador i s'esborren quan es tanca la pestanya o finestra.

Avantatges

- **Capacitat d'Emmagatzematge**: Ofereix més espai (fins a 5-10MB) que les cookies.
- **API Simple**: Proporciona una interfície senzilla per emmagatzemar i recuperar dades.

Usos Comuns

- **Manteniment de l'Estat del Client**: Pot guardar dades de sessió i preferències de l'usuari que no cal enviar al servidor.
- **Sincronització amb l'Entorn Servidor**: Es pot utilitzar per emmagatzemar dades que després es sincronitzen amb el servidor, millorant el rendiment i l'experiència d'usuari.

Consideracions de Seguretat

- **Accessibilitat**: Les dades són accessibles per JavaScript, per la qual cosa s'ha de tenir cura amb les vulnerabilitats XSS.
- **Dades Sensibles**: Evita emmagatzemar dades sensibles o confidencials.

Tot i que **Web Storage** s'executa al costat del client, entendre les seves funcionalitats pot ajudar els desenvolupadors del servidor a dissenyar sistemes més robustos i eficaços, on la cooperació entre client i servidor maximitza l'eficiència de l'aplicació web.

Tokens d'Autenticació

Els **JSON Web Tokens (JWT)** són un estàndard obert que defineix una manera compacta i autònoma d'enviar informació entre dues parts de manera segura com a objecte JSON. S'utilitzen habitualment per a l'autenticació en aplicacions web.

Avantatges dels JWT

- **Estatut autònom**: Porten tota la informació necessària, eliminant la necessitat de mantenir sessions al servidor.
- **Escalabilitat**: Milloren l'escalabilitat en aplicacions distribuïdes.
- **Seguretat**: Es poden signar digitalment per assegurar la seva autenticitat.

Desavantatges dels JWT

- **Revocació complexa**: Una vegada emesos, és difícil revocar-los sense mantenir una llista negra.
- **Sobrecarrega de dades**: Si els tokens són grans, poden afectar el rendiment, especialment en xarxes de baixa latència.

Millors Pràctiques

- Utilitzar signatura HS256 o RS256 per garantir la integritat del token.
- No emmagatzemar dades sensibles directament al token.

Cache del Navegador

El **cache del navegador** s'utilitza per emmagatzemar còpies temporals de recursos web per millorar el rendiment i reduir la càrrega del servidor.

Avantatges del Cache

- **Rendiment**: Redueix el temps de càrrega dels recursos.
- **Optimització**: Disminueix l'ample de banda requerit.

Desavantatges del Cache

- **Consistència**: Pot servir dades obsoletes si no es gestiona correctament.
- **Control**: Requereix configuració per evitar el caching indesitjat de dades dinàmiques.

Millors Pràctiques

- Configurar els encapçalaments HTTP correctament (`Cache-Control`, `ETag`) per gestionar l'actualització de recursos.

Sincronització Offline

La **sincronització offline** es refereix a la capacitat d'una aplicació web de funcionar sense connexió a Internet, sincronitzant dades quan es recupera la connexió.

Tècniques i Eines

- **IndexedDB**: Emmagatzema grans volums de dades estructurades dins del navegador.
- **Service Workers**: Gestionen peticions de xarxa, proporcionant funcionalitats offline i cache avançat.

Millors Pràctiques

- Gestionar conflictes de dades quan es torna a estar en línia.
- Utilitzar estratègies de sincronització optimitzades per minimitzar l'ample de banda i el temps de sincronització.

Seguretat de les Cookies i Sessions

La **seguretat** és fonamental en la gestió de cookies i sessions per protegir les dades dels usuaris de possibles atacs.

Pràctiques de Seguretat

- **CSRF**: Utilitzar tokens CSRF per validar sol · licituds d'accions sensibles.
- **XSS**: Sanear les dades d'entrada i utilitzar capçaleres de seguretat (`Content-Security-Policy`) per prevenir XSS.
- **Secure Flag**: Marcar les cookies amb l'atribut `Secure` perquè només s'enviïn a través de connexions HTTPS.

Conclusió La selecció del mecanisme adequat per al manteniment de l'estat en una aplicació web depèn de les necessitats específiques de l'aplicació, el volum de dades, els requisits de seguretat i l'arquitectura del sistema. Avaluar cada tècnica pel que fa a avantatges i desavantatges ajudarà a prendre decisions informades i construir aplicacions web més segures i eficients.

2.Exemples de Cookies i Sessions en PHP

Cookies

Les cookies s'emmagatzemen en el array global `$_COOKIE`. El que col · loquem dins del array, es guardarà en el client. Cal tindre present que el client pot no voler emmagatzemar-les.

Existeix una limitació de 20 cookies per domini i 300 en total en el navegador.

En PHP, per a crear una cookie s'utilitza la funció `setcookie`:

```php
<?php
setcookie(
    'nom_cookie',
    'valor_cookie',
    [
        'expires' => time() + 3600, // 1 hora
        'path' => '/',
        'domain' => '', // Domini actual
        'secure' => true, // Només HTTPS
        'httponly' => true, // Només accessible via HTTP
        'samesite' => 'Lax' // o 'Strict' o 'None'
    ]
);
?>
```

Destacar que el nom no pot contindre espais ni el caràcter `;`. Respecte al contingut de la cookie, no pot superar els 4 KB.

Consideracions de seguretat per a cookies: **HTTPOnly**: Assegura't que les cookies que contenen informació sensible no siguin accessibles per JavaScript utilitzant l'atribut HttpOnly.

Secure: Utilitza l'atribut Secure per assegurar que les cookies només es transmeten en connexions HTTPS.

SameSite: Defineix correctament l'atribut SameSite per a prevenir atacs CSRF (Cross-Site Request Forgery).

Per exemple, mitjançant *cookies* podem comprovar la quantitat de visites diferents que realitza un usuari:

```php
<?php
$accesosPagina = 0;
if (isset($_COOKIE['accesos'])) {
    $accesosPagina = $_COOKIE['accesos']; // recuperamos una cookie
    setcookie('accesos', ++$accesosPagina); // le asignamos un valor
}
?>
```

!!! tip "Inspeccionant les cookies" Si volem veure que contenen les cookies que tenim emmagatzemades en el navegador, es pot comprovar el seu valor en Dev Tools –> Application –> Storage

El temps de vida de les cookies pot ser tan llarg com el lloc web en el qual resideixen. Elles seguiran ací, fins i tot si el navegador està tancat o obert.

Per a esborrar una cookie es pot posar que expiren en el passat:

```php
<?php
setcookie(nombre, "", 1) // pasado
```

O que caduquen dins d'un període de temps deteminado:

```php
<?php
setcookie(nombre, valor, time() + 3600) // Caducan dentro de una hora
```

Comunicació amb cookies

S'utilitzen per a:

- Recordar els inicis de sessió
- Emmagatzemar valors temporals d'usuari
- Si un usuari està navegant per una llista paginada d'articles, ordenats d'una certa manera, podem emmagatzemar l'ajust de la classificació.

L'alternativa en el client per a emmagatzemar informació en el navegador és l'objecte LocalStorage.

Sessió

La sessió afig la gestió de l'estat a HTTP, emmagatzemant en aquest cas la informació en el servidor. Cada visitant té un ID de sessió únic, el qual per defecte s'emmagatzema en una cookie denominada `PHPSESSID`. Si el client no té les cookies actives, l'ID es propaga en cada URL dins del mateix domini. Cada

sessió té associat un magatzem de dades mitjançant el array global `$_SESSION`, en el qual podem emmagatzemar i recuperar informació.

La sessió comença en executar un script PHP. Es genera un nou ID i es carreguen les dades del magatzem:

Comunicació amb sessions

Les operacions que podem realitzar amb la sessió són:

```
<?php
session_start(); // carga la sesión
session_regenerate_id(true); // regenera el id
session_id() // devuelve el id
$_SESSION[clave] = valor; // inserción
session_destroy(); // destruye la sesión
unset($_SESSION[clave]; // borrado
```

Veurem mitjançant un exemple com podem inserir en un pàgina dades en la sessió per a posteriorment en una altra pàgina accedir a aqueixes dades. Per exemple, en `sesion.php` tindríem

```
<?php
// Iniciar sessió
session_start();
session_regenerate_id(true);

// Establir valors de sessió
$_SESSION['usuari'] = 'JohnDoe';
$_SESSION['rol'] = 'admin';

?>
```

I posteriorment podem accedir a la sessió en `sesion1.php`:

```
<?php
session_start();
// Recuperar valors de sessió
echo 'Usuari: ' . $_SESSION['usuari'] . '<br>';
echo 'Rol: ' . $_SESSION['rol'] . '<br>';

// Tancar sessió de forma segura
session_unset();  // Eliminar totes les variables de sessió
session_destroy();  // Destruir la sessió
?>
```

!!! note “Configurant la sessió en `php.ini`” Les següent propietats de `php.ini` permeten configurar alguns aspectes de la sessió:

```
  * `session.save_handler`: controlador que gestiona com s'emmagatzema (`files`)
  * `session.save_path`: ruta on s'emmagatzemen els arxius amb les dades (si tenim un clúste
```

```
  * `session.name`: nom de la sessió (`PHSESSID`)
  * `session.acte_start`: Es pot fer que s'autocarregue amb cada script. Per defecte està de
  * `session.cookie_lifetime`: temps de vida per defecte
```

Més informació en la documentació oficial.

Serialització en PHP

La serialització és el procés de convertir una estructura de dades o un objecte en una seqüència de caràcters que pot ser fàcilment emmagatzemada o transmesa i després reconstruïda. PHP proporciona dos funcions principals per a això: serialize() i unserialize().

1. serialize() La funció serialize() en PHP s'utilitza per a convertir una estructura de dades o un objecte en una representació de cadena.

```php
$data = array("a", "b", "c");
$serialized_data = serialize($data);
echo $serialized_data;
```

Eixida

```
a:3:{i:0;s:1:"a";i:1;s:1:"b";i:2;s:1:"c";}
```

2. unserialize() La funció unserialize() en PHP s'utilitza per a convertir la representació de cadena serialitzada de nou en una estructura de dades o un objecte.

```php
$original_data = unserialize($serialized_data);
print_r($original_data);
```

Eixida

```
Array
(
    [0] => a
    [1] => b
    [2] => c
)
```

Utilitzant amb Sessions Les sessions en PHP permeten emmagatzemar informació d'usuari per ser utilitzada en diverses pàgines. Pot ser útil serialitzar dades per a emmagatzemar-les en una sessió:

Iniciant una sessió i emmagatzemant dades serialitzades:

```php
session_start();
session_regenerate_id(true);

$data = array("a", "b", "c");
$_SESSION['data_serialitzada'] = serialize($data);
```

```php
session_start();

if (isset($_SESSION['data_serialitzada'])) {
$data = unserialize($_SESSION['data_serialitzada']);
print_r($data);
}
```

Consideracions de Seguretat: És crucial entendre que la funció unserialize() pot ser perillosa si s'usa amb dades que no són de confiança, ja que podria portar a l'execució de codi arbitrari. Per això, mai has de deserialitzar dades que vinguen d'una font desconeguda o no fiable sense validar-les prèviament.

3.Autenticació d'usuaris

Mecanismes d'Autenticació d'Usuaris

Mecanisme d'Autenticació	Característiques	Avantatges
Bàsica (usuari/contrasenya)	Es requereix un nom d'usuari i una contrasenya per accedir.	Fàcil d'implementar, àmpliament utilitzada.
Cookies	Emmagatzema informació d'autenticació en el navegador de l'usuari.	Persistència d'inici de sessió, personalització d'experiència.
Sessions	Manté l'estat d'autenticació en el servidor amb una identificació de sessió única.	Major seguretat, evita la necessitat d'emmagatzemar informació sensible al client.
OAuth	Permet als usuaris accedir a recursos sense compartir les seves credencials.	Seguretat millorada, experiència d'usuari simplificada.
JWT (JSON Web Token)	Utilitza tokens basats en JSON per a l'autenticació.	Lliure d'estat, fàcil de compartir entre diferents serveis.

Mecanisme d'Autenticació	Característiques	Avantatges
SAML (Security Assertion Markup Language)	Utilitza XML per a intercanviar dades d'autenticació entre l'usuari i el servei.	Integració amb sistemes d'autenticació empresarial, alt nivell de seguretat.
Autenticació multifactor (MFA)	Requereix múltiples formes de verificació (per exemple, contrasenya + codi SMS).	Seguretat augmentada, redueix el risc de compromís de comptes.

Exemple amb sessions Una sessió estableix una relació anònima amb un usuari particular, de manera que podem saber si és el mateix usuari entre dues peticions diferents. Si preparem un sistema de login, podrem saber qui utilitza la nostra aplicació.

```php
<?php
// Llista d'usuaris predefinits amb contrasenyes en text pla
$users = [
    'user1@example.com' => 'password1',
    'user2@example.com' => 'password2',
];

// Convertir les contrasenyes a un format encriptat
foreach ($users as $email => $password) {
    $users[$email] = password_hash($password, PASSWORD_BCRYPT);
}

// Formulari d'autenticació
if (isset($_POST['login'])) {
    $email = $_POST['email'];
    $password = $_POST['password'];

    if (isset($users[$email]) && password_verify($password, $users[$email])) {
        // L'usuari està autenticat
        session_start();
        $_SESSION['user'] = $email;
        echo "Login successful. Welcome, " . $email;
    } else {
        // Credencials incorrectes
        echo "Invalid email or password.";
    }
}
```

```
?>
<form method="post">
    Email: <input type="email" name="email" required>
    Password: <input type="password" name="password" required>
    <button type="submit" name="login">Login</button>
</form>
```

Aquest exemple mostra com mantenir l'estat de la sessió d'un usuari una vegada autenticat.

```
<?php
session_start();

if (!isset($_SESSION['user'])) {
    header("Location: login.php");
    exit();
}

// Mostra la pàgina només si l'usuari està autenticat
echo "Welcome, " . $_SESSION['user'];
?>
```

Finalment, necessitem l'opció de tancar la sessió que col · loquem en `logout.php`:

```
<?php
// Recuperamos la información de la sesión
session_start();

// Y la destruimos
session_destroy();
header("Location: index.php");
?>
```

!!! warning "Autenticació en producció" En l'actualitat l'autenticació d'usuari no es realitza gestionant la sessió direcamente, sinó que es realitza mitjançant algun framekwork que abstrau tot el procés o la integració de mecanismes d'autenticació tipus *OAuth*, com estudiarem en l'última unitat mitjançant *Laravel*.

Exemple amb cookies

```
<?php
// Llista d'usuaris predefinits amb contrasenyes en text pla
$users = [
    'user1@example.com' => 'password1',
    'user2@example.com' => 'password2',
];
```

```php
// Convertir les contrasenyes a un format encriptat
foreach ($users as $email => $password) {
    $users[$email] = password_hash($password, PASSWORD_BCRYPT);
}

if (isset($_POST['login'])) {
    $email = $_POST['email'];
    $password = $_POST['password'];

    if (isset($users[$email]) && password_verify($password, $users[$email])) {
        // Establir una cookie d'autenticació
        setcookie("user", $email, time() + (86400 * 30), "/"); // 86400 = 1 dia
        echo "Login successful. Welcome, " . $email;
    } else {
        // Credencials incorrectes
        echo "Invalid email or password.";
    }
}

// Llegir la cookie
if (isset($_COOKIE['user'])) {
    echo "Welcome back, " . $_COOKIE['user'];
}
?>
<form method="post">
    Email: <input type="email" name="email" required>
    Password: <input type="password" name="password" required>
    <button type="submit" name="login">Login</button>
</form>
```

!!! warning "Seguretat en cookies" Les cookies són vulnerables a atacs com *Cross-Site Scripting (XSS)* i *Cross-Site Request Forgery (CSRF)*. Per a protegir-les, s'ha de configurar la cookie com a segura i només accessible a través de la web.

Exemple amb JWT Aquest exemple mostra com generar i verificar un JWT per a l'autenticació.

```php
<?php
function base64UrlEncode($data) {
    return rtrim(strtr(base64_encode($data), '+/', '-_'), '=');
}

function base64UrlDecode($data) {
    return base64_decode(strtr($data, '-_', '+/'));
```

```
}

function createJWT($header, $payload, $secret) {
    $headerEncoded = base64UrlEncode(json_encode($header));
    $payloadEncoded = base64UrlEncode(json_encode($payload));

    $signature = hash_hmac('sha256', "$headerEncoded.$payloadEncoded", $secret, true);
    $signatureEncoded = base64UrlEncode($signature);

    return "$headerEncoded.$payloadEncoded.$signatureEncoded";
}

function verifyJWT($jwt, $secret) {
    list($headerEncoded, $payloadEncoded, $signatureEncoded) = explode('.', $jwt);

    $signature = base64UrlDecode($signatureEncoded);
    $expectedSignature = hash_hmac('sha256', "$headerEncoded.$payloadEncoded", $secret, true

    if ($signature === $expectedSignature) {
        return json_decode(base64UrlDecode($payloadEncoded));
    }

    return false;
}

// Exemples d'ús
$header = ['alg' => 'HS256', 'typ' => 'JWT'];
$payload = ['email' => 'user1@example.com', 'exp' => time() + 3600];
$secret = 'your_secret_key';

$jwt = createJWT($header, $payload, $secret);
echo "JWT: " . $jwt . "\n";

$decoded = verifyJWT($jwt, $secret);
if ($decoded) {
    echo "JWT valid: " . json_encode($decoded) . "\n";
} else {
    echo "Invalid JWT.\n";
}
?>
```

Exemple amb MFA Aquest exemple mostra com afegir una capa addicional d'autenticació amb un codi MFA.

```
<?php
session_start();
```

```php
function sendMFACode() {
    $code = rand(100000, 999999);
    $_SESSION['mfa_code'] = $code;

    // Simular enviament de codi via email o SMS
    echo "Verification code: $code"; // En un entorn real, envia el codi per email o SMS.
}

function verifyMFACode($inputCode) {
    return isset($_SESSION['mfa_code']) && $inputCode == $_SESSION['mfa_code'];
}

if (isset($_POST['send_code'])) {
    sendMFACode();
}

if (isset($_POST['verify_code'])) {
    if (verifyMFACode($_POST['mfa_code'])) {
        echo "MFA successful.";
    } else {
        echo "Invalid verification code.";
    }
}
?>
<form method="post">
    <button type="submit" name="send_code">Send MFA Code</button>
</form>
<form method="post">
    MFA Code: <input type="text" name="mfa_code" required>
    <button type="submit" name="verify_code">Verify MFA Code</button>
</form>
```

Exemple amb OAuth Per a OAuth, es pot utilitzar un proveïdor extern com Google per autenticar els usuaris. Ho vorem més avant.

```php
<?php
require_once 'vendor/autoload.php';

$provider = new League\OAuth2\Client\Provider\Google([
    'clientId'     => 'your-client-id',
    'clientSecret' => 'your-client-secret',
    'redirectUri'  => 'your-redirect-url',
]);

if (!isset($_GET['code'])) {
```

```
    $authUrl = $provider->getAuthorizationUrl();
    $_SESSION['oauth2state'] = $provider->getState();
    header('Location: ' . $authUrl);
    exit;
} elseif (empty($_GET['state']) || ($_GET['state'] !== $_SESSION['oauth2state'])) {
    unset($_SESSION['oauth2state']);
    exit('Invalid state');
} else {
    $token = $provider->getAccessToken('authorization_code', [
        'code' => $_GET['code']
    ]);

    $user = $provider->getResourceOwner($token);
    $userData = $user->toArray();

    // Mostra la informació de l'usuari
    echo 'Hello, ' . $userData['name'];
}
?>
```

4. Referències Addicionals

A continuació es presenten diverses referències que poden ajudar-te a aprofundir en el tema de la gestió de sessions i cookies en PHP.

Documentació Oficial de PHP

- **Sessions a PHP**: La documentació oficial de PHP proporciona informació detallada sobre l'ús de sessions, incloent-hi exemples pràctics i consells de millors pràctiques.
  - Sessions a PHP
- **Cookies a PHP**: Trobaràs informació oficial sobre com treballar amb cookies en PHP, amb descripcions d'atributs com `HttpOnly`, `Secure` i `SameSite`.
  - Cookies a PHP

Articles i Blocs Tècnics

- **PHP Sessions: Tips & Tricks**: Aquest article discuteix tècniques avançades per gestionar sessions en PHP, amb exemples i millors pràctiques per a la seguretat.
  - PHP Sessions: Tips & Tricks
- **Handling Cookies Securely in PHP**: Una guia sobre com manejar cookies de manera segura, destacant estratègies per protegir les dades de l'usuari.
  - Handling Cookies Securely in PHP

Llibres sobre PHP

- **"Modern PHP: New Features and Good Practices"** de Josh Lockhart: Aquest llibre cobreix les novetats de PHP i les millors pràctiques, incloent la seguretat de sessions i cookies.
- **"PHP Objects, Patterns, and Practice"** de M. Zandstra: Proporciona una visió profunda sobre el disseny de programari amb PHP, incloent-hi aspectes de seguretat relacionats amb sessions i cookies.

Guies de Seguretat en Desenvolupament Web

- **OWASP Secure Coding Practices**: Aquesta guia de OWASP ofereix una ràpida referència sobre les millors pràctiques de codificació segura, útils per a protegir les teves aplicacions web.
  - OWASP Secure Coding Practices
- **OWASP Cheat Sheet on Session Management**: Una fitxa de referència ràpida de OWASP sobre la gestió de sessions, proporcionant consells de seguretat essencials.
  - OWASP Cheat Sheet on Session Management

Conferències i Vídeos Educatius

- **PHP Conference YouTube Channel**: El canal de YouTube de PHP Conference ofereix vídeos de conferències i xerrades que cobreixen una varietat de temes, incloent sessions i seguretat.
  - PHP Conference YouTube Channel
- **Laracasts**: Ofereix vídeos d'alta qualitat sobre desenvolupament en PHP i Laravel, amb temes rellevants per a sessions i cookies.
  - Laracasts

Tutorials Online i Plataformes Educatives

- **Stack Overflow PHP Sessions Tag**: Un lloc de consulta on trobar respostes a preguntes freqüents sobre la gestió de sessions en PHP.
  - Stack Overflow PHP Sessions Tag
- **Cursos a Udemy sobre PHP**: Cursos que cobreixen des de conceptes bàsics fins a avançats de PHP, incloent-hi la gestió de sessions i cookies.
  - Cursos a Udemy sobre PHP

5.Exercicis

Bateria d'Exercicis Solucionats per a la Unitat de Programació web

Exercici 1: Crear una sessió 1.Crea una pàgina PHP que inicialitze una sessió i emmagatzeme el nom i el rol d'un usuari en variables de sessió. Mostra aquests valors a la pàgina web.

Solució

```php
<?php
// Iniciar sessió
session_start();

// Emmagatzemar informació de l'usuari en la sessió
$_SESSION['nom'] = 'Joan';
$_SESSION['rol'] = 'Administrador';

echo 'Benvingut, ' . $_SESSION['nom'] . '<br>';
echo 'Rol: ' . $_SESSION['rol'] . '<br>';
?>
```

Exercici 2: Regenerar l'ID de sessió

1. Modifica l'exercici anterior per regenerar l'ID de sessió just després d'emmagatzemar la informació de l'usuari.

Solució

```php
<?php
// Iniciar sessió
session_start();

// Emmagatzemar informació de l'usuari en la sessió
$_SESSION['nom'] = 'Joan';
$_SESSION['rol'] = 'Administrador';

// Regenerar l'ID de sessió
session_regenerate_id(true);

echo 'Benvingut, ' . $_SESSION['nom'] . '<br>';
echo 'Rol: ' . $_SESSION['rol'] . '<br>';
?>
```

Exercici 3: Tancar la sessió

1. Escriu un script PHP que elimine totes les variables de sessió i destrueixi la sessió quan l'usuari tanqui sessió.

Solució

```php
<?php
// Iniciar sessió
```

```
session_start();

// Eliminar totes les variables de sessió
session_unset();

// Destruir la sessió
session_destroy();
?>
```

Exercici 4: Crear una cookie segura

1. Escriu un script PHP per crear una cookie que emmagatzeme el nom d'usuari amb els atributs de seguretat `HttpOnly`, `Secure` i `SameSite`.

Solució

```php
<?php
// Crear una cookie segura
setcookie(
    'nom_usuari',
    'Joan',
    [
        'expires' => time() + 3600, // 1 hora
        'path' => '/',
        'domain' => '', // Domini actual
        'secure' => true, // Només HTTPS
        'httponly' => true, // Només accessible via HTTP
        'samesite' => 'Lax' // o 'Strict' o 'None'
    ]
);
?>
```

Exercici 5: Llegir i modificar una cookie

1. Crea una pàgina PHP que llegeixca el valor d'una cookie anomenada `nom_usuari` i la modifique afegint un prefix de salutació.

Solució

```php
<?php
// Llegir el valor de la cookie
if (isset($_COOKIE['nom_usuari'])) {
    $nomUsuari = $_COOKIE['nom_usuari'];
    echo 'Hola, ' . $nomUsuari;
```

```
    // Modificar el valor de la cookie
    $salutacio = 'Hola, ' . $nomUsuari;
    setcookie('nom_usuari', $salutacio, time() + 3600, '/');
} else {
    echo 'Cookie not found.';
}
?>
```

Exercici 6: Aplicació de gestió d'usuaris amb sessions

1. Desenvolupa una aplicació PHP amb dos scripts: un per iniciar sessió i un altre per tancar sessió. L'aplicació ha de permetre que l'usuari introdueixca el seu nom d'usuari i vega un missatge de benvinguda una vegada haja iniciat sessió.

Solució

```php
<!-- login.php -->
<?php
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$nomUsuari = $_POST['nom_usuari'];

    // Emmagatzemar el nom d'usuari en la sessió
    $_SESSION['nom_usuari'] = $nomUsuari;

    // Redireccionar a la pàgina de benvinguda
    header('Location: welcome.php');
    exit();
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Iniciar Sessió</title>
</head>
<body>
    <h2>Iniciar Sessió</h2>
    <form method="post" action="login.php">
        <label for="nom_usuari">Nom d'usuari:</label>
        <input type="text" id="nom_usuari" name="nom_usuari" required>
        <button type="submit">Iniciar Sessió</button>
```

````
    </form>
</body>
</html>

```

```php
<!-- logout.php -->
<?php
session_start();

// Tancar sessió de forma segura
session_unset();
session_destroy();

// Redireccionar a la pàgina de login
header('Location: login.php');
exit();
?>
```

```php
<!-- wellcome.php -->
<?php
session_start();

if (!isset($_SESSION['nom_usuari'])) {
    // Redireccionar a la pàgina de login si no s'ha iniciat sessió
    header('Location: login.php');
    exit();
}

$nomUsuari = $_SESSION['nom_usuari'];
?>

<!DOCTYPE html>
<html>
<head>
    <title>Benvingut</title>
</head>
<body>
    <h2>Benvingut, <?php echo htmlspecialchars($nomUsuari); ?>!</h2>
    <p>Aquesta és la teva pàgina de benvinguda.</p>
    <a href="logout.php">Tancar Sessió</a>
</body>
</html>
````

```

### Exercici 7: Aplicació de gestió de preferències amb cookies

1. Crea una aplicació PHP que permeta als usuaris seleccionar el seu color preferit, emmagatzemant aquesta informació en una cookie. La pàgina hauria de mostrar el color preferit de l'usuari en futures visites.

Solució

```php
<?php
// Llegir el color preferit de la cookie
$colorPreferit = isset($_COOKIE['color_preferit']) ? $_COOKIE['color_preferit'] : 'blau';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $colorPreferit = $_POST['color_preferit'];

    // Emmagatzemar el color preferit en una cookie
    setcookie('color_preferit', $colorPreferit, time() + 3600, '/');
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Preferències de Color</title>
</head>
<body style="background-color: <?php echo $colorPreferit; ?>">
    <h2>Preferències de Color</h2>
    <form method="post">
        <label for="color_preferit">Color preferit:</label>
        <select id="color_preferit" name="color_preferit">
            <option value="blau" <?php if ($colorPreferit === 'blau') echo 'selected'; ?>>Bl
            <option value="vermell" <?php if ($colorPreferit === 'vermell') echo 'selected';
            <option value="verd" <?php if ($colorPreferit === 'verd') echo 'selected'; ?>>Ve
        </select>
        <button type="submit">Guardar</button>
    </form>
</body>
</html>
```

Exercicis proposats

Exercici 1: Sistema de Carret de Compres sense Base de Dades

1. **Descripció:**
```

Desenvolupa una aplicació PHP que permeta als usuaris afegir productes a un carret de compres i mostrar el contingut del carret. Utilitza sessions per a mantindre l'estat del carret entre pàgines.

2. **Requisits:**

   - Crear una pàgina on l'usuari puga seleccionar productes.
   - Afegir els productes seleccionats a un carret emmagatzemat en una sessió.
   - Mostrar un resum del carret amb els productes afegits i les seues quantitats.
   - Permetre que l'usuari elimine productes del carret.

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Selecció de productes</title>
</head>
<body>
    <h1>Afegir productes al carret</h1>
    <form action="carret.php" method="POST">
        <label for="producte">Tria un producte:</label>
        <select name="producte" id="producte">
            <option value="Poma">Poma</option>
            <option value="Plàtan">Plàtan</option>
            <option value="Taronja">Taronja</option>
        </select>
        <input type="submit" value="Afegir al carret">
    </form>
    <a href="carret.php">Veure carret</a>
</body>
</html>
```

Exercici 2: Autenticació Bàsica d'Usuaris amb Sessions

1. **Descripció:**

   Crea una aplicació PHP que permeta als usuaris iniciar sessió mitjançant un formulari. Utilitza sessions per a mantindre l'estat d'autenticació de l'usuari i mostrar missatges personalitzats basats en aquest estat.

2. **Requisits:**

   - Crear un formulari d'inici de sessió que sol · licite el nom d'usuari i la contrasenya.
   - Emmagatzemar l'estat d'autenticació en una sessió després de verificar les credencials.

- Mostrar una pàgina de benvinguda personalitzada per a l'usuari autenticat.
- Proporcionar un enllaç per a tancar sessió i destruir la sessió.

Exercici 3: Recordatori d'Usuari amb Cookies

1. **Descripció:**

   Afig una funcionalitat de "recordar-me" a l'exercici anterior que emmagatzeme el nom d'usuari en una cookie i permeta a l'usuari ser recordat en futures visites al lloc web.

2. **Requisits:**

   - Afig una opció de "recordar-me" al formulari d'inici de sessió.
   - Emmagatzemar el nom d'usuari en una cookie quan l'opció és seleccionada.
   - Comprovar la cookie en futures visites i iniciar sessió automàticament si la cookie existeix.
   - Assegurar que les cookies es configuren amb atributs de seguretat adequats (`HttpOnly`, `Secure`, `SameSite`).

Exercici 4: Formulari de Contacte amb Protecció CSRF

1. **Descripció:**

   Desenvolupa un formulari de contacte que permeta als usuaris enviar missatges i implementa protecció CSRF per a assegurar que les sol · licituds siguen legítimes.

2. **Requisits:**

   - Crear un formulari de contacte amb camps per al nom, correu electrònic i missatge.
   - Generar i emmagatzemar un token CSRF en una sessió quan es carrega el formulari.
   - Incloure el token CSRF com a camp ocult en el formulari.
   - Verificar el token CSRF quan s'envia el formulari i mostrar un missatge de confirmació si és vàlid.
   - Mostrar un missatge d'error si el token CSRF no és vàlid o no existeix.

Exercici 5: Seguiment d'Activitat de l'Usuari amb Sessions

1. **Descripció:**

   Crea un sistema que registre les pàgines visitades per l'usuari durant una sessió i mostre aquesta informació quan l'usuari visita una pàgina d'activitat.

2. **Requisits:**

- Emmagatzemar una llista de pàgines visitades per l'usuari en una sessió.
- Actualitzar la llista de pàgines cada vegada que l'usuari visite una nova pàgina.
- Crear una pàgina que mostre l'historial de pàgines visitades durant la sessió actual.
- Assegurar que l'historial es restableix quan l'usuari tanca la sessió.

Solucions

6. Enunciats dels projectes

Per als dos projectes

1. **Autenticació de Jugadors:**
   - Implementa un sistema d'autenticació bàsic que permeta als jugadors iniciar sessió abans de començar el joc. Utilitza sessions per a mantindre l'estat d'autenticació.
   - Ha de servir per als dos jocs, "Penjat" i "4 en Ratlla".
   - Fes que una vegada autenticar l'usuari puga triar a quin joc vol jugar (en la mateix pàgina).
   - Protegeix els jocs per tal que no es puga jugar en cas de no estar autenticat.

Projecte Penjat

1. **Manteniment de l'Estat del Joc amb Sessions:**
   - Utilitza sessions per a emmagatzemar l'estat actual del joc, incloent la paraula a endevinar, lletres endevinades, i el nombre d'intents restants.
2. **Gestió de la Sessió del Joc:**
   - Afegeix funcionalitats per a reiniciar el joc en qualsevol moment, reinicialitzant les variables de sessió per a començar una nova partida.
   - Afegeix una opció per a tancar sessió i finalitzar la partida actual.
   - Afegeix un funció per a saver si el joc ha acabat, ja siga perquè s'han endevinat totes les lletres o s'haguen arribat al màxim número d'intents permesos.
   - Controla el final del joc
3. **Cookies per a Recordar Jugadors:**
   - Implementa cookies per a recordar els jugadors entre visites, permetent que l'usuari siga recordat si selecciona una opció de "Recordarme" durant l'inici de sessió.
4. **Seguretat i Autenticació:**
   - Implementa un sistema d'autenticació bàsic per a garantir que només els jugadors autenticats puguen accedir al joc.
   - Utilitza sessions per a mantenir l'estat d'autenticació i controlar l'accés a les funcionalitats del joc.

Consideracions Addicionals

- **Resiliència del Joc:** Implementa la lògica necessària per a manejar intents invàlids i mostrar missatges d'error adequats.
- **Millores d'Interfície:** Afegix un enllaç o botó per a tancar sessió i una opció per a reiniciar el joc.

Projecte "4 en Ratlla"

1. **Gestió de l'Estat de la Graella amb Sessions:**
   - Utilitza sessions per a emmagatzemar l'estat actual de la graella i el torn del jugador. Això permet mantenir la partida entre sol · licituds.
2. **Manteniment de la Sessió entre Jugadors:**
   - Emmagatzema l'identificador de cada jugador en sessions per a assegurar que el torn actual siga persistent entre sol · licituds.
   - Afegeix funcionalitats per a reiniciar el joc en qualsevol moment, reinicialitzant les variables de sessió per a començar una nova partida.
   - Afegeix una opció per a tancar sessió i finalitzar la partida actual.
   - Afegeix un funció per a saver si el joc ha acabat, ja siga perquè s'ha fet 4 en ratlla o s'haja completat el tauler.
   - Controla el final del joc.
3. **Cookies per a la Persistència d'Usuaris:**
   - Permet l'ús de cookies per a recordar els jugadors entre visites si han seleccionat "Recordar-me". Utilitza cookies per a emmagatzemar l'últim jugador autenticat.
4. **Seguretat i Autenticació:**
   - Implementa un sistema d'autenticació bàsic per a garantir que només els jugadors autenticats puguen accedir al joc.
   - Utilitza sessions per a mantenir l'estat d'autenticació i controlar l'accés a les funcionalitats del joc.
5. **Addicional**
   - Implementa la lògica per tal que el segon jugador siga la màquina (pots adaptar i/o millorar l'algorisme de baix).
   - Controla el joc per a que no es puga seguir jugant una vegada acabat.
   - Implementa un sistema de puntuació que otorgue 2 punts al guanyador i 1 a cadascú en cas d'empat.

```php
function jugar(&$graella,$jugadorActual){

        $opponent = $jugadorActual === 1 ? 2 : 1;

        // Comprovar si pots guanyar
        for ($col = 1; $col <= COLUMNES; $col++) {
            if (isValidMove($graella, $col)) {
```

```
            $tempBoard = $graella;
            $coord = ferMoviment($tempBoard, $col, $jugadorActual);

            if (fi_joc($tempBoard, $coord)) {
                return ferMoviment($graella,$col,$jugadorActual); // Guanyar immediatame
            }
        }
    }

    // Comprovar si l'oponent pot guanyar i bloquejar
    for ($col = 1; $col <= COLUMNES; $col++) {
        if (isValidMove($graella, $col)) {
            $tempBoard = $graella;
            $coord = ferMoviment($tempBoard, $col, $opponent);
            if (fi_joc($tempBoard, $coord )) {
                return ferMoviment($graella,$col,$jugadorActual); // Bloquejar
            }
        }
    }

    // Estratègia: buscar el millor moviment
    // Podem afegir més lògica aquí per seleccionar el millor moviment
    $possibles = array();
    for ($col = 1; $col <= COLUMNES; $col++) {
        if (isValidMove($graella, $col)) {
            $possibles[] = $col;
        }
    }
    if (count($possibles)>2) {
        $random = rand(-1,1);
    }
    $middle = (int) (count($possibles) / 2)+$random;
    $inthemiddle = $possibles[$middle];
    return ferMoviment($graella, $inthemiddle, $jugadorActual);

    return -1; // Totes les columnes estan plenes
}
```

Consideracions Addicionals

- **Lògica de Torn de Jugadors:** Implementa la lògica necessària per a canviar de torn entre els jugadors utilitzant sessions.
- **Reinici de Partida:** Afegeix una opció per a reiniciar el joc, que ha de restablir les sessions i permetre començar de nou.
- **Interfície d'Usuari Amigable:** Assegura't que la interfície d'usuari és clara i proporciona indicacions visuals dels moviments i torns dels ju-

gadors.

Rúbrica d'Avaluació

Criteri	Excel · lent (4)	Bé (3)	Adequat (2)	Insuficient (1)
Funcionalitat del Joc	El joc està completament funcional i sense errors.	El joc està majoritàriament funcional amb errors mínims.	El joc és funcional, però conté errors significatius.	El joc no és funcional o està incomplet.
Ús de Sessions	Sessions ben implementades per a mantenir l'estat del joc.	Sessions utilitzades correctament amb alguns problemes menors.	Sessions utilitzades, però amb deficiències importants.	No s'han utilitzat sessions o són incorrectes.
Ús de Cookies	Cookies ben utilitzades per a recordar els jugadors.	Cookies utilitzades adequadament amb algunes millores possibles.	Cookies utilitzades amb limitacions evidents.	No s'han utilitzat cookies o són incorrectes.
Autenticació d'Usuaris	Autenticació segura i efectiva implementada.	Autenticació implementada amb alguns problemes.	Autenticació present però amb deficiències notables.	No s'ha implementat autenticació o és incorrecta.
Interfície d'Usuari	Interfície atractiva i fàcil d'utilitzar.	Interfície clara amb algunes millores possibles.	Interfície funcional però poc intuïtiva.	Interfície confusa i difícil d'utilitzar.
Seguretat	Totes les mesures de seguretat implementades correctament.	Seguretat adequada amb algunes millores possibles.	Mesures de seguretat bàsiques implementades.	No s'han tingut en compte mesures de seguretat.
Comentaris i Codi	Codi ben comentat i fàcilment llegible.	Codi clar amb comentaris adequats.	Codi llegible però amb pocs comentaris.	Codi desordenat i sense comentaris.
Innovació i Creativitat	El projecte mostra innovació significativa.	Algunes idees creatives han estat implementades.	Alguna creativitat present, però limitada.	Cap creativitat o innovació en el projecte.

Criteri	Excel · lent (4)	Bé (3)	Adequat (2)	Insuficient (1)
Punts addicionals	1 punt per cadascúna aconseguida.			

Explicació dels Criteris

Funcionalitat del Joc

- **Excel · lent (4):** El joc funciona completament segons les especificacions, amb totes les funcionalitats implementades i sense errors. Els jugadors poden interaccionar amb el joc tal com s'esperava i totes les accions es realitzen correctament.
- **Insuficient (1):** El joc no és funcional, falten parts importants del codi o el joc no es pot jugar correctament.

Ús de Sessions

- **Excel · lent (4):** Les sessions s'utilitzen eficaçment per a mantenir l'estat del joc i de l'usuari entre sol · licituds. Les dades de la sessió es gestionen de manera adequada per a preservar l'experiència de l'usuari.
- **Insuficient (1):** No s'han utilitzat sessions o la seva implementació és incorrecta, la qual cosa afecta negativament l'experiència de l'usuari.

Ús de Cookies

- **Excel · lent (4):** Les cookies s'utilitzen de manera efectiva per a recordar els jugadors entre sessions, amb la configuració adequada d'atributs de seguretat (`HttpOnly`, `Secure`, `SameSite`).
- **Insuficient (1):** No s'han utilitzat cookies o la seva implementació és incorrecta, amb una configuració de seguretat deficient.

Autenticació d'Usuaris

- **Excel · lent (4):** L'autenticació dels usuaris és segura i eficient, amb mecanismes adequats per a validar les credencials i protegir la informació dels usuaris.
- **Insuficient (1):** No s'ha implementat un sistema d'autenticació o el sistema present és insegur i defectuós.

Interfície d'Usuari

- **Excel · lent (4):** La interfície d'usuari és atractiva, clara i fàcil de navegar, proporcionant una experiència d'usuari òptima.

- **Insuficient (1):** La interfície és confusa, difícil d'utilitzar o inacabada.

Seguretat

- **Excel · lent (4):** Totes les mesures de seguretat necessàries han estat implementades, incloent-hi la validació de dades d'entrada, protecció CSRF i altres pràctiques de seguretat recomanades.
- **Insuficient (1):** No s'han tingut en compte mesures de seguretat, deixant el projecte vulnerable a possibles atacs.

Comentaris i Codi

- **Excel · lent (4):** El codi està ben comentat, és llegible i segueix bones pràctiques de programació.
- **Insuficient (1):** El codi està desordenat, difícil de llegir i manca de comentaris explicatius.

Innovació i Creativitat

- **Excel · lent (4):** El projecte mostra un alt grau d'innovació i creativitat, oferint característiques o enfocaments únics que milloren el joc.
- **Insuficient (1):** El projecte manca de creativitat o innovació, seguint únicament les instruccions bàsiques sense cap valor afegit.

Notes Addicionals

- **Consistència del Codi:** S'espera que el codi sigui consistent, utilitzant convencions de noms adequades i estructures de codi clares.
- **Adaptació de les Necessitats:** Els criteris poden ser adaptats segons les necessitats específiques del curs o dels projectes individuals.

7. Autoavaluació: Gestió de Sessions i Cookies

Exercici 1: Funcions de les Cookies

Pregunta: Quina és la funció principal de les cookies en el context del desenvolupament web?

Opcions: a) Guardar els fitxers de l'usuari al servidor. b) Emmagatzemar informació del client per personalitzar l'experiència d'usuari. c) Executar codi al servidor. d) Controlar la velocitat de la connexió a Internet.

Exercici 2: Seguretat de les Cookies

Pregunta: Quins atributs de seguretat haurien de tindre les cookies per protegir-les contra atacs?

Opcions: a) Secure b) HttpOnly c) SameSite d) CrossSite

Exercici 3: Creació de Sessions en PHP

Pregunta: Quin dels següents passos és necessari per iniciar una sessió en PHP?

Opcions: a) Cridar a la funció session_start(). b) Utilitzar la funció session_open(). c) Assignar un valor a la variable $_SESSION. d) No es necessita cap funció especial.

Exercici 4: Manteniment d'Informació en Sessions

Pregunta: Com es pot mantindre la informació d'un usuari durant la sessió d'una aplicació web?

Opcions: a) Utilitzant la variable global $GLOBALS. b) Utilitzant la variable $_SESSION. c) Utilitzant la variable $_COOKIE. d) Utilitzant arxius temporals al servidor.

Exercici 5: Funcions de PHP per a Sessions

Pregunta: Quina funció de PHP es fa servir per destruir una sessió?

Opcions: a) session_destroy() b) session_unset() c) session_delete() d) session_end()

Exercici 6: Avantatges de les Sessions

Pregunta: Quins són els avantatges d'utilitzar sessions en lloc de cookies per a mantenir l'estat de l'usuari?

Opcions: a) Les sessions poden emmagatzemar més informació perquè es guarden al servidor. b) Les sessions són més segures perquè no s'envien al client. c) Les sessions redueixen la càrrega del servidor. d) Les sessions no necessiten ser configurades amb atributs de seguretat.

Exercici 7: Autenticació d'Usuaris

Pregunta: Quina és la pràctica recomanada per assegurar la identitat d'un usuari en una aplicació web?

Opcions: a) Utilitzar noms d'usuari i contrasenyes emmagatzemades com a cookies. b) Utilitzar sessions per mantenir l'estat d'autenticació després d'iniciar sessió. c) Emmagatzemar la contrasenya de l'usuari a la URL. d) No utilitzar cap forma d'autenticació.