

Construyendo un Acceso Seguro

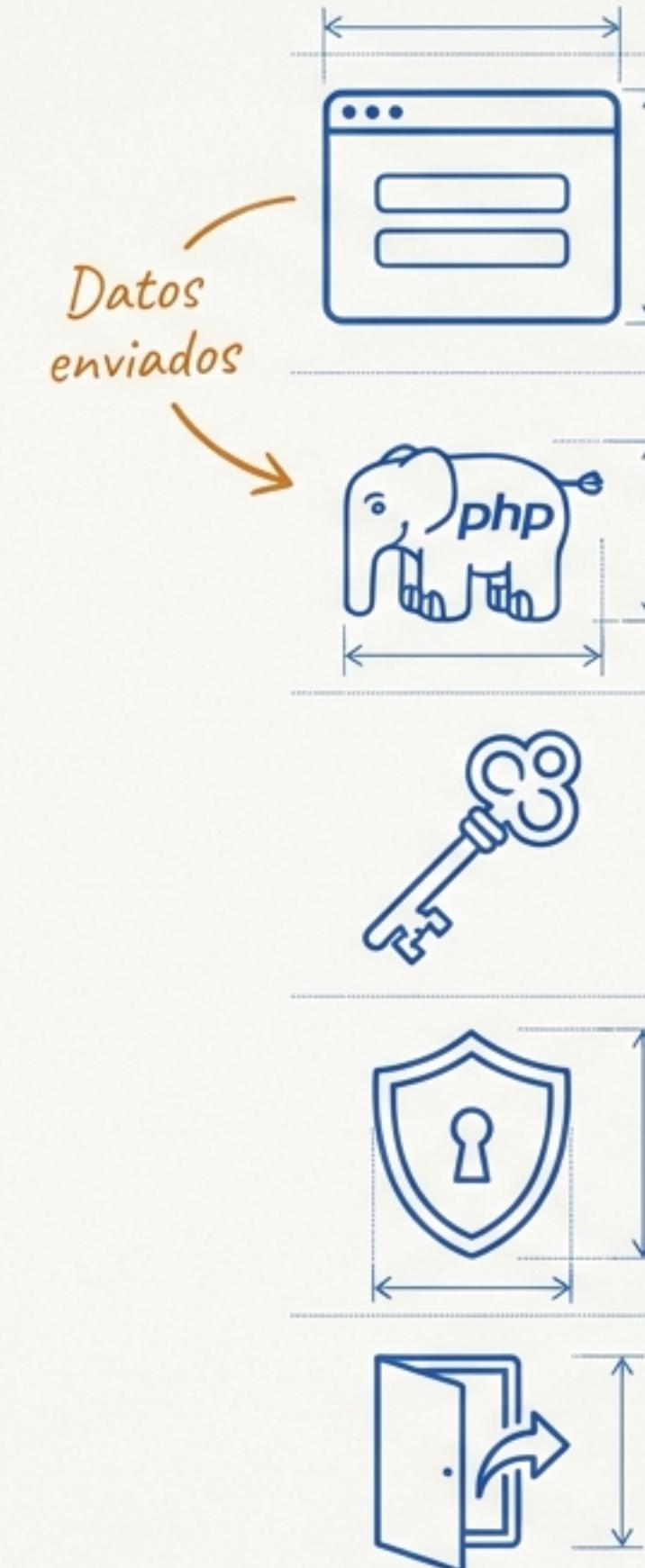
Guía Práctica de Login y Sesiones en PHP

Nivel Intermedio: Lógica de Aplicación

El Desafío: Nuestro Proyecto de Autenticación

Objetivo de la Actividad: Implementar un flujo de autenticación completo sin usar una base de datos.

Enunciado: "Login Básico sin Base de Datos".



Crear un formulario de login (Usuario y Contraseña).

Validar las credenciales contra un usuario fijo.

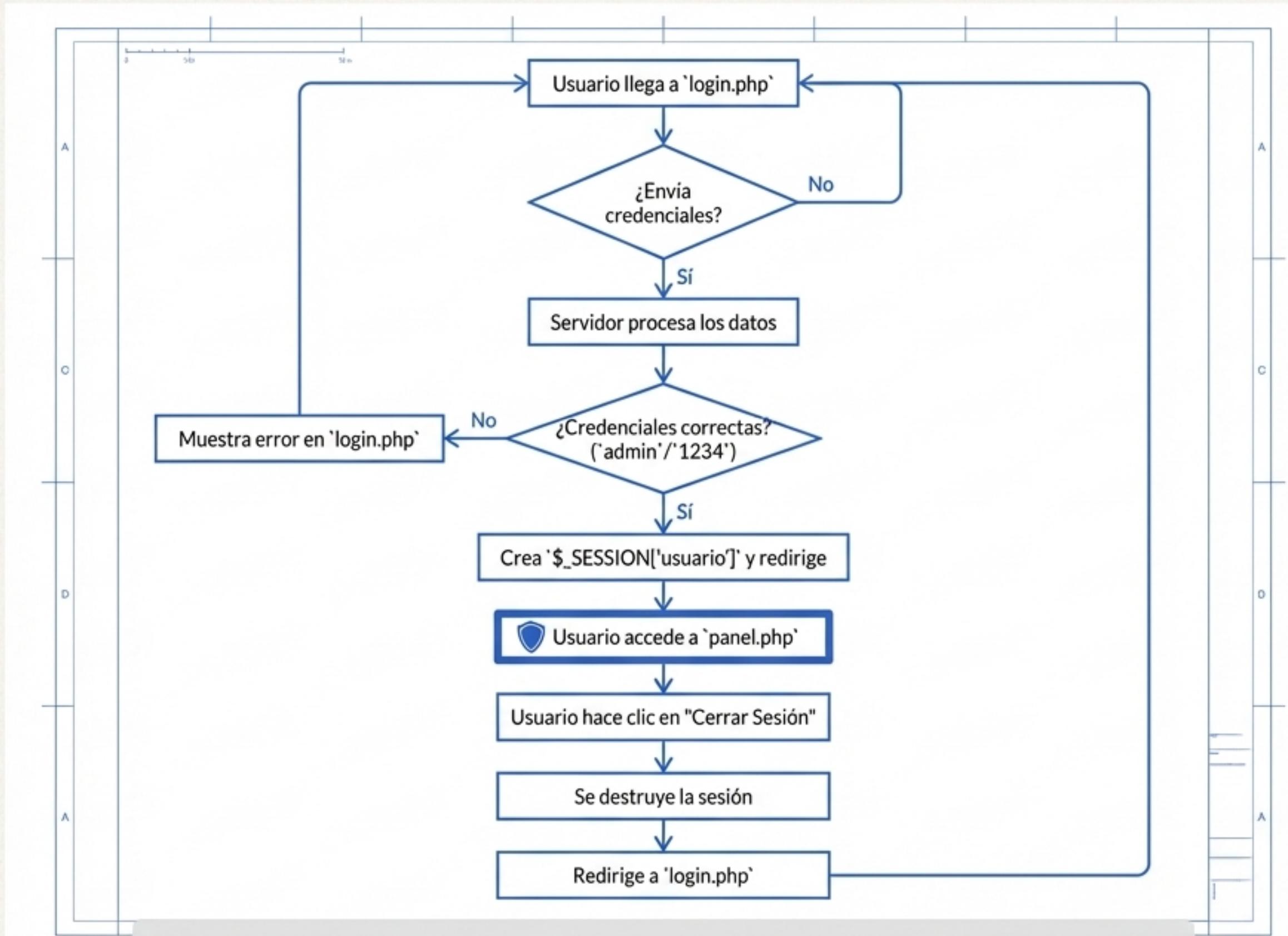
Si es correcto, iniciar una sesión de usuario.

Proteger una página de acceso exclusivo.

Implementar la funcionalidad de "Cerrar Sesión".

Flujo completo

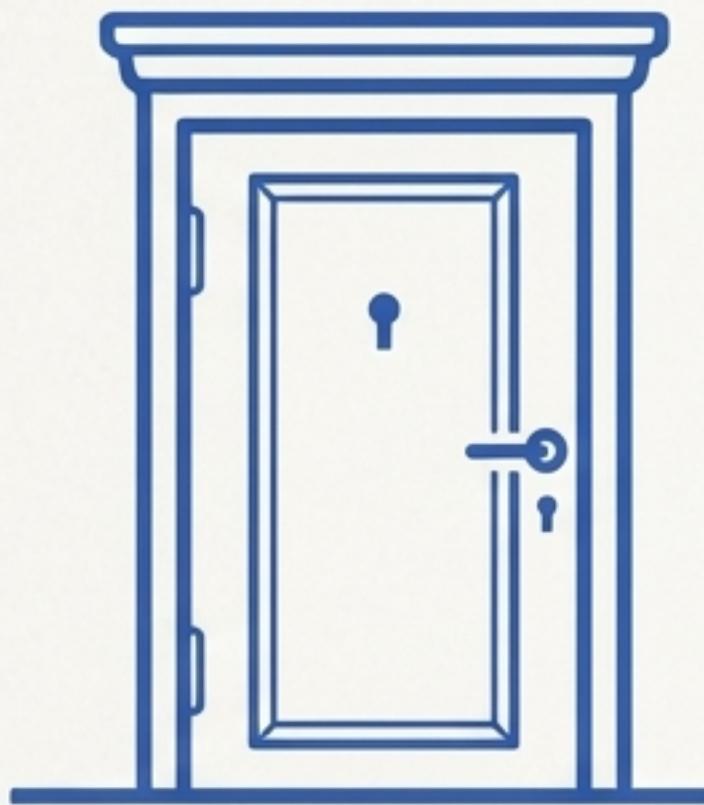
El Plano Arquitectónico: El Flujo del Usuario



Este diagrama es nuestra guía. Seguiremos cada paso para construir nuestro sistema.

Paso 1: La Puerta de Entrada (`login.php`)

Creando el Formulario HTML



Todo comienza con una interfaz. Este formulario es el único punto de acceso para el usuario. Usamos el método `POST` para enviar los datos de forma segura al servidor.

```
<form action="login.php" method="POST">
  <h2>Iniciar Sesión</h2>
  <label>Usuario:</label>
  <input type="text" name="usuario">

  <label>Contraseña:</label>
  <input type="password" name="password">

  <button type="submit">Entrar</button>
</form>
```

Clave para enviar datos que no deben ser visibles en la URL.

El atributo `name` es como PHP identificará este dato.

El Guardián en la Puerta: Procesando la Petición

Iniciando la Lógica del Servidor en PHP

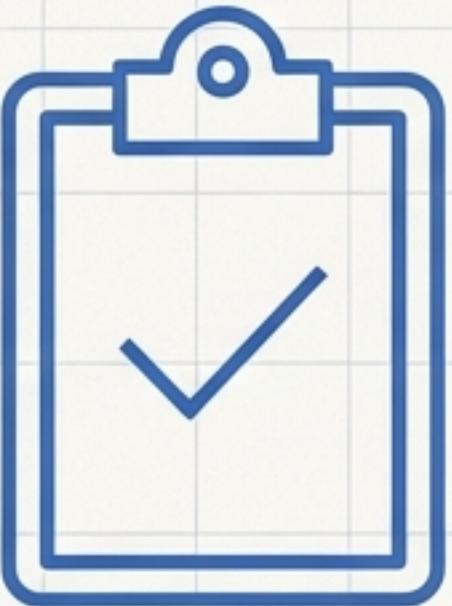
```
<?php
// Procesar el formulario cuando se envía
if($_SERVER['REQUEST_METHOD'] == 'POST'){
    // La lógica de validación irá aquí...
    $usuario = $_POST['usuario'];
    $password = $_POST['password'];
}
?>
```

Este bloque de código solo se ejecuta si el usuario ha enviado el formulario. Es el interruptor que activa nuestra seguridad.

Recuperamos los datos enviados desde el formulario usando la superglobal `'$_POST'`.

Verificando la Identidad: La Lista de Invitados

La Validación de Credenciales



Aquí es donde decidimos si el usuario tiene acceso. Comparamos lo que nos ha dado con lo que esperamos. Para este ejercicio, usamos valores fijos. En una aplicación real, esto se consultaría en una base de datos.

```
<?php
// Validar contra usuario fijo
if($usuario == 'admin' && $password == '1234'){
    // ACCESO CONCEDIDO
    // Próximo paso: crear la sesión y redirigir
} else {
    // ACCESO DENEGADO
    $error = "Usuario o contraseña incorrectos";
}
?>
```

El operador `&&` (Y lógico) es crucial. Exige que AMBAS condiciones, usuario y contraseña, sean verdaderas, verdaderas.

Concediendo el Acceso: La Llave Mágica (`\$_SESSION`)

Creando una Sesión de Usuario

Paso 1: Iniciar el motor de sesiones.

```
session_start();
```

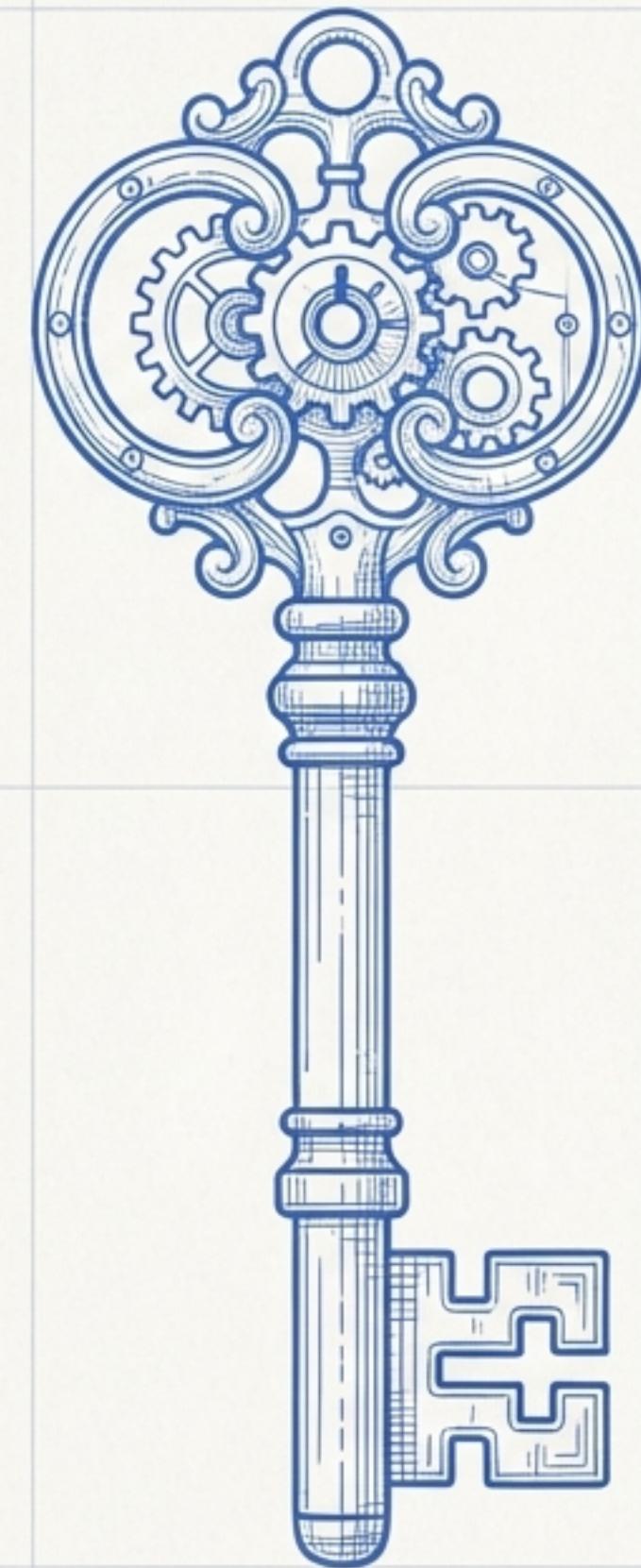
Este comando debe ser el primero en ejecutarse. Prepara el servidor para manejar la memoria de la sesión.

Paso 2: Guardar la prueba de identidad.

```
// Dentro de if($usuario == 'admin' && ...)  
$_SESSION['usuario'] = 'admin';
```

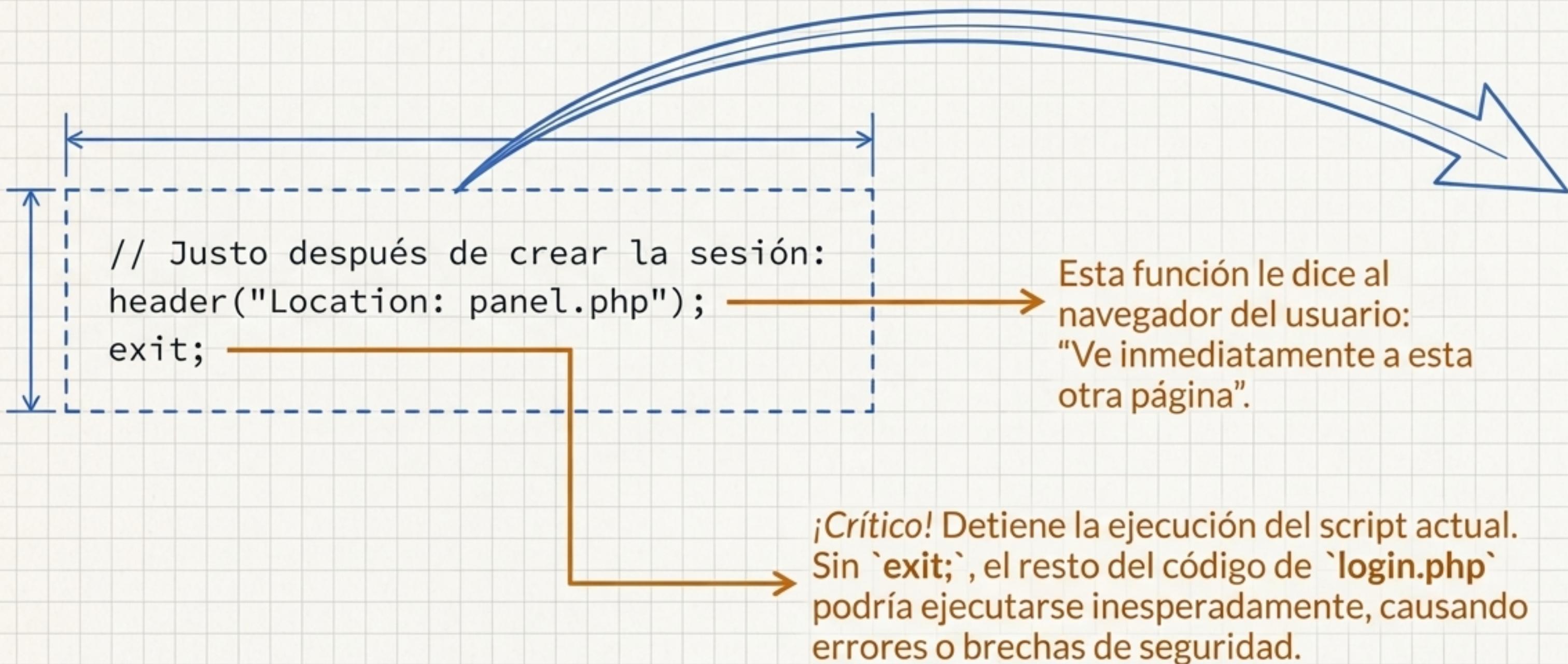
Asignamos un valor ('admin') a una clave ('usuario') en el array super global `\$_SESSION`. Esto es lo que persiste.

`\$_SESSION` es un array especial que persiste entre diferentes páginas. Al guardar 'usuario', creamos una "llave" digital que el usuario llevará consigo mientras navega.



Enviando al Área VIP: La Redirección

Usando `header()` para Controlar el Flujo



Arquitectura Completa de `login.php`

Uniendo Todas las Piezas

```
<?php
session_start();

// 1. Si ya está logueado, redirigir
if(isset($_SESSION['usuario'])){
    header("Location: panel.php");
    exit;
}

// 2. Procesar el formulario
if($_SERVER['REQUEST_METHOD'] == 'POST'){
    $usuario = $_POST['usuario'];
    $password = $_POST['password'];

    if($usuario == 'admin' && $password == '1234'){
        $_SESSION['usuario'] = 'admin';
        header("Location: panel.php");
    } else {
        $error = "Usuario o contraseña incorrectos";
    }
}
?>
<!-- HTML del formulario aquí --&gt;</pre>
```

Observa el primer bloque: si el usuario ya tiene una sesión activa (`isset($_SESSION['usuario'])`), lo enviamos directamente al panel. Esto mejora la experiencia de usuario.

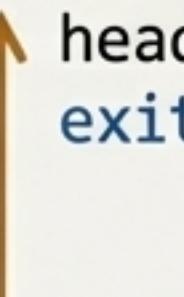
Paso 2: El Área Restringida (`panel.php`)

El Guardián Permanente en la Entrada



Esta página debe protegerse a sí misma. Lo primero que hace es preguntar: '¿Tienes la llave (la sesión)?'. Si la respuesta es no, expulsa al visitante.

```
<?php  
session_start();  
  
// Seguridad: si no está logueado, volver al login  
if(!isset($_SESSION['usuario'])){  
    header("Location: login.php");  
    exit;  
}  
?>
```



La exclamación `!` invierte la lógica: 'Si la sesión de usuario NO existe...'. Esta es la línea de defensa más importante de nuestra página protegida.

Dentro del Panel: Contenido Protegido

Personalizando la Experiencia del Usuario

Una vez verificado el acceso, podemos mostrar el contenido exclusivo y usar los datos de la sesión para personalizar la página.

```
<h2>Panel de Control</h2>  
  
<p>Bienvenido al Panel</p>  
<p>Usuario: <?php echo $_SESSION['usuario']; ?></p>  
<p>Esta es una página protegida</p>
```

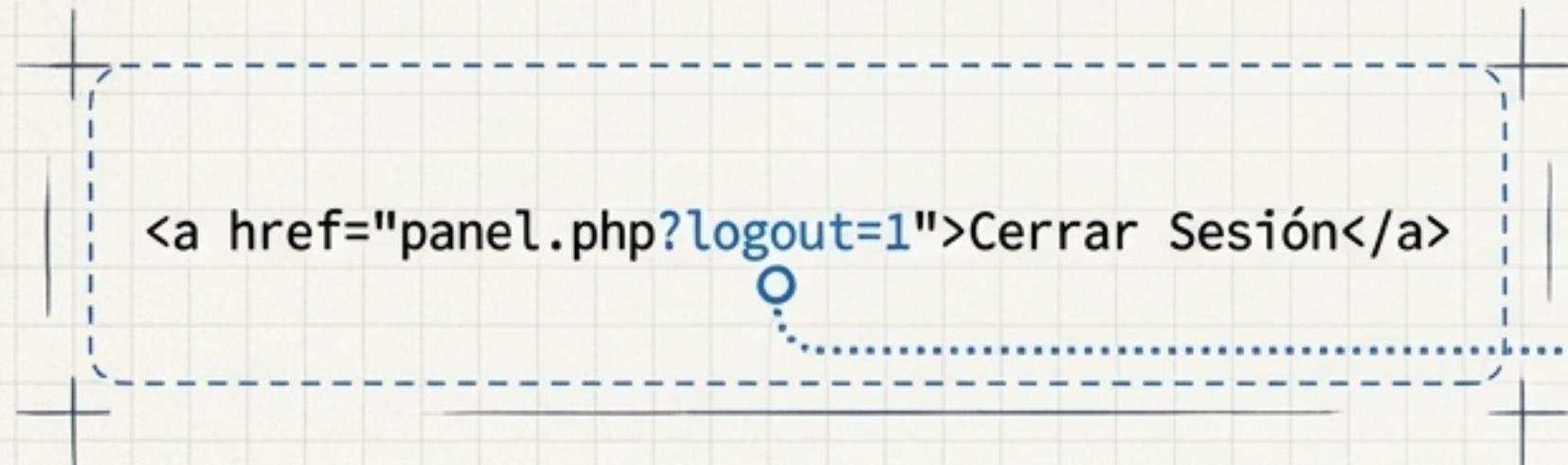
Aquí recuperamos el nombre de usuario de la sesión para mostrar un saludo personalizado. La sesión actúa como un puente de datos entre páginas.

La Salida Controlada: El Botón de Logout

Permitiendo al Usuario Cerrar su Sesión

Lado Izquierdo (El Enlace)

Añadimos un simple enlace. La clave está en el parámetro GET que le pasamos a la URL (`?logout=1`).



Lado Derecho (La Detección en PHP)

El servidor busca este parámetro en la URL para saber que debe iniciar el proceso de cierre de sesión.

```
// Procesar cierre de sesión  
if(isset($_GET['logout'])){  
    // Lógica para destruir la sesión aquí...  
}
```

Usamos `$_GET` porque `logout` es un parámetro en la URL, no de un formulario POST.

Revocando el Acceso: Destruyendo la Sesión

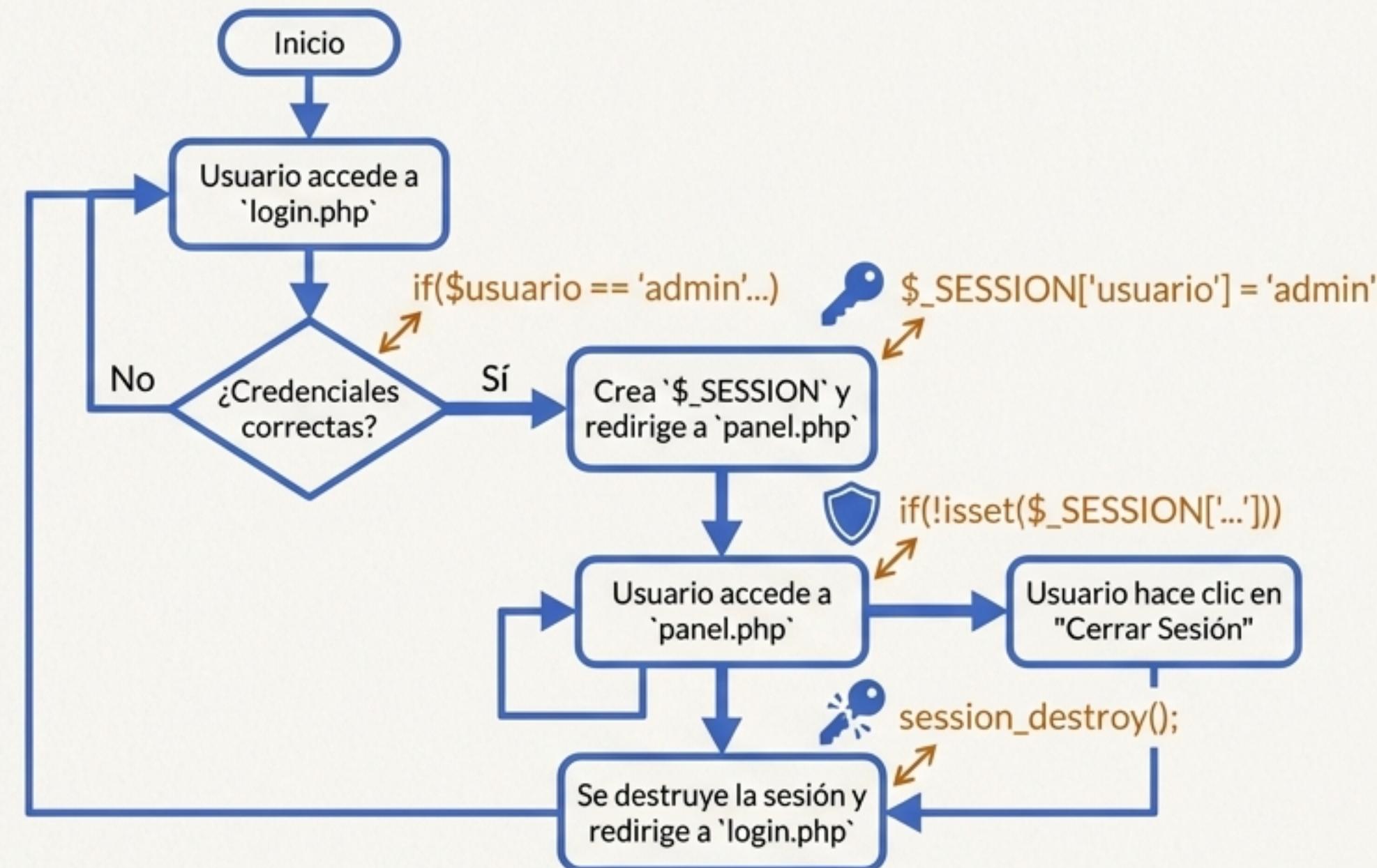
`session_destroy()` vs `unset()`

```
// Dentro de if(isset($_GET['logout']){  
  
    // 1  
    session_destroy();  
    // 2  
    unset($_SESSION['usuario']);  
  
    // 3  
    header("Location: login.php");  
    exit;  
}
```



1. **session_destroy()** : Elimina toda la información de la sesión del almacenamiento del servidor. Es como quemar la tarjeta de acceso.
2. **unset(\$_SESSION['usuario'])** : Elimina la variable del array `\$_SESSION` en el script actual. Es una buena práctica para asegurar que no se use accidentalmente después.

El Plano Terminado: Vista Completa del Sistema



Cada pieza de nuestro código tiene un propósito claro dentro de este flujo, creando un sistema coherente y seguro.

Principios de una Arquitectura Segura

Conceptos Clave que Has Dominado Hoy



Flujo de Control Explícito: Usamos `header()` para guiar al usuario a través de la aplicación de manera predecible.



Gestión de Estado con Sesiones: `$_SESSION` nos permite recordar quién es el usuario a través de múltiples peticiones HTTP.



Protección de Rutas: Cada página sensible es responsable de verificar la autorización antes de mostrar su contenido.



Separación de Intereses: El HTML presenta la información, y el PHP maneja la lógica, incluso dentro del mismo archivo.