

Unidad N° 5

Gestión de la información

Bases de datos con PHP

Introducción

- PHP brinda un soporte importante para permitir la conexión a varios motores de bases de datos.
- Además, el lenguaje incorpora su propio “gestor” de base de datos, a través de la extensión SQLite.
- Sin embargo, uno de los mejores aliados de PHP es MySQL, para lo cual incorpora la extensión MySQLi.
- Esta extensión tiene las siguientes características:
 - Incluye una interfaz orientada a objetos.
 - Permite utilizar consultas preparadas (disminuyen el tráfico de datos en la red).
 - Permite realizar subconsultas.

Bases de datos con PHP

- Provee funciones específicas para implementar transacciones.
- Brinda soporte para replicación.
- La extensión MySQLi provee 2 clases que permiten gestionar el trabajo con una base de datos:
 - Mysqli: gestionar la conexión.
 - Mysqli_result: gestionar recordset's.

Bases de datos con PHP

Conexión y desconexión del motor de base de datos

- La primera operación a realizar para poder efectuar consultas es conectarse al servidor MySQL.
- Esto se logra creando un objeto *mysqli*, el que requiere los siguientes parámetros:
 - Servidor: es la dirección IP del servidor MySQL. Si este se ejecuta en la misma computadora que el servidor web, entonces es 127.0.0.1 (o localhost).
 - Usuario: nombre de usuario válido, con permisos suficientes como para acceder a la base de datos.
 - Password del usuario que quiere conectarse.
 - Base de datos a la que intenta conectarse.

Bases de datos con PHP

- Una vez creado el objeto *mysqli*, se estableció correctamente la conexión y ya contamos con un identificador de la misma.
- Al abrir la conexión, se selecciona la base de datos indicada como parámetro.
- Luego, si se debe cambiar, se puede utilizar el método *select_db*, cuyo parámetro es el nombre de la base de datos a la que se conectará el motor.
- Una vez finalizado el trabajo con el motor de bases de datos MySQL, se debe cerrar la conexión. Esto se realiza invocando al método *close*.

Bases de datos con PHP

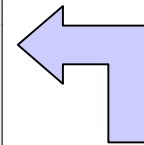
Algunos atributos de la clase MySQLi

Función	Descripción
<code>\$connect_error</code>	Devuelve una cadena con la descripción del último error de conexión.
<code>\$affected_rows</code>	Devuelve el número de registros afectados en la ejecución de la última consulta de acción SQL.
<code>\$insert_id</code>	Devuelve el valor del campo autoincremental del último registro insertado en una tabla.
<code>\$field_count</code>	Obtiene el número de campos de un recordset.
<code>\$server_info</code>	Devuelve la versión del servidor MySQL.

Bases de datos con PHP

Ejemplo

```
mysqli Object
(
    [affected_rows] => 0
    [client_info] => mysqlnd 8.1.6
    [client_version] => 80106
    [connect_errno] => 0
    [connect_error] =>
    [errno] => 0
    [error] =>
    [error_list] => Array
        (
        )
    [field_count] => 0
    [host_info] => localhost via TCP/IP
    [info] =>
    [insert_id] => 0
    [server_info] => 10.4.24-MariaDB
    [server_version] => 100424
    [sqlstate] => 00000
    [protocol_version] => 10
    [thread_id] => 21
    [warning_count] => 0
)
```



```
1 <!-- Codigo_133.php -->
2 <!DOCTYPE html>
3 <html lang="es">
4 <head>
5 <title>Ejemplo - Objeto MySQLi</title>
6 </head>
7 <body>
8 <section>
9 <article>
10 <?php
11 $con = new mysqli("localhost","root","") or die ("No se pudo conectar al server");
12 echo "<pre>";
13 print_r ($con);
14 echo "</pre>";
15 $con->close();
16 ?>
17 </article>
18 </section>
19 </body>
20 </html>
```

Bases de datos con PHP

Ejecución de consultas

- Las consultas a la base de datos se realizan mediante el método *query* de la clase *MySQLi*.
- Este método requiere como parámetro un string que contiene la consulta a realizar.
- Luego, si la consulta se ejecutó correctamente, el método devolverá 1 y, sino devolverá 0.
- Además, si se trata de una consulta de selección, devolverá un objeto *MySQLi_result* con los registros que cumplen las condiciones de la consulta.
 - El número de registros devueltos se puede determinar con el método *num_rows* de la clase *MySQLi_result*.

Bases de datos con PHP

Ejemplo

```
1 <!-- Codigo_37.php -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="utf-8" />
6 <title>Ejemplo - Acceso a BD</title>
7 </head>
8 <body>
9 <section>
10 <article>
11 <?php
12 //Conexión al servidor MySQL
13 $con = new mysqli("localhost","root","","curso_php") or die ("No es posible conectarse al motor de BD");
14 //Preparación y ejecución de la consulta
15 $query = "INSERT INTO usuarios (Apellido, Nombres, Documento, Fecha_nac, Correo, Usuario, Pass) ";
16 $query.= "VALUES ('GONZALEZ','Adriana', 20140215, '1970-10-31', 'agonza@hotmail.com', 'adriana', 'adriana')";
17 $con->query($query) or die ("No se pudo ejecutar la consulta de selección");
18 //Impresión de mensaje aclaratorio
19 echo "El usuario fue dado de alta";
20 //Cierre de la conexión
21 $con->close();
22 ?>
23 </article>
24 </section>
25 </body>
26 </html>
```



Bases de datos con PHP

Recorrido de los registros

- La ejecución de una consulta de selección produce un objeto `MySQLi_result` (recordset) que, para recorrerlo, es necesario convertir cada registro en un arreglo o a un objeto.
- Para esto, se utilizan los métodos *fetch_array* y/o *fetch_object*. Si se utiliza *fetch_array*, se envía como parámetro el tipo de arreglo a crear:
 - Asociativo: `MYSQLI_ASSOC`
 - Enumerado: `MYSQLI_NUM`
 - Ambos: `MYSQLI_BOTH` (por defecto)
- Cuando no se utilice más el recordset, se debe liberar la memoria que ocupa → método *free*.

Bases de datos con PHP

Ejemplo

```
1 <!-- Codigo_38.php -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="utf-8" />
6 <title>Ejemplo - Acceso a BD</title>
7 </head>
8 <body>
9 <section>
10 <article>
11 <?php
12 require_once("contador.class.php");
13 //Conexión al motor
14 $con = new mysqli("localhost","root","","curso_php") or die ("No es posible conectarse al motor de BD");
15 //Preparación y ejecución de la consulta
16 $query = "SELECT Apellido, Nombres, Correo FROM usuarios WHERE Year(Fecha_nac)>1975 ORDER BY Apellido ASC, Nombres ASC";
17 $resultado = $con->query($query) or die ("No se pudo ejecutar la consulta de selección");
18 //Impresión de resultados
19 echo "<h2>Resultado de la consulta:</h2>";
20 if ($resultado->num_rows>0)
21 {
22     $conta = new contador();
23     echo '<h3>Se encontraron '.$resultado->num_rows.' personas:</h3>';
24     echo '<hr>';
25     //Recorrido del recordset
26     while ($registro = $resultado->fetch_object())
27     {
28         $conta->incremento();
29         echo '<strong>Registro N° '.$conta->valor_actual().'</strong><br>';
30         echo 'Apellido y nombre: '.$registro->Apellido.', '.$registro->Nombres.'<br>';
31         echo 'Correo electrónico: <a href="mailto:'.$registro->Correo.'">'.$registro->Correo.'</a>';
32         echo '<hr>';
33     }
34 }
35 else
36 {
37     echo '<h3>No se encontraron registros</h3>';
38 }
39 //Liberación de la memoria utilizada por el recordset y cierre de la conexión
40 $resultado->free();
41 $con->close();
42 ?>
43 </article>
44 </section>
45 </body>
46 </html>
```

Bases de datos con PHP

Ejemplo (continuación)



Bases de datos con PHP

Algunos métodos de la clase MySQLi_result

Función	Descripción
change_user	Permite cambiar el usuario y/o base de datos manteniendo la conexión establecida.
ping	Permite determinar si la conexión con el servidor se encuentra operativa. Si no es así, intentará reconectar.
fetch_row fetch_assoc	Similar a fetch_array, pero no es necesario especificar el tipo de arreglo.
fetch_object	Similar a fetch_array, pero trata al registro como un objeto y a cada campo como una propiedad de ese objeto.
data_seek	Permite posicionarse en algún registro específico del recordset.

Transacciones

Definición

- Una transacción es un conjunto de consultas SQL que deben ejecutarse como si fuera una unidad → o se ejecutan todas o no se ejecuta ninguna.
- Se dice que una base de datos que implementa transacciones, supera la prueba ACID:
 - Atómica: todas las consultas dentro de la transacción se ejecutan correctamente o no se ejecutan.
 - Consistente: solo se escriben datos válidos en la base de datos (integridad referencial).
 - Aislada: los cambios serán visibles al finalizar la transacción.
 - Duradera: los cambios son permanentes.

Transacciones

- MySQL soporta el uso de transacciones. Para ello, incorpora las tablas INNODB.
- Las transacciones en MySQL se pueden implementar de dos maneras:
 - Método 1: Con las sentencias específicas de SQL
 - Begin: inicia la transacción.
 - Commit: confirma la transacción
 - Rollback: deshace la transacción
 - Método 2: Con los métodos incluidos en MySQLi
 - Begin_transaction: inicia la transacción.
 - commit: confirma la transacción.
 - rollback: deshace la transacción.

Transacciones

Esquema general

Transacción	Método 1	Método 2
Inicio	<code>\$conexion->query("Begin");</code>	<code>\$conexion->begin_transaction();</code>
Cuerpo (sentencias que la forman)	<code>\$conexion->query(\$consulta_1);</code> <code>\$conexion->query(\$consulta_n);</code>	
Fin	<code>If (todo_ok)</code> <code>{ \$conexion->query("Commit"); }</code> <code>else</code> <code>{ \$conexion->query("Rollback"); }</code>	<code>If (todo_ok)</code> <code>{ \$conexion->commit(); }</code> <code>else</code> <code>{ \$conexion->rollback(); }</code>

Transacciones

Ejemplo (Método 1)

```
1 <!-- Codigo_39.php -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="utf-8" />
6 <title>Ejemplo - Transacciones - Método 1</title>
7 </head>
8 <body>
9 <section>
10 <article>
11 <?php
12 //Método 1: utilizando las sentencias de SQL (Begin, Commit y Rollback)
13 $con = new mysqli("localhost","root","root","curso_php") or die ("No se pudo conectar al server");
14 $consulta_1 = "INSERT INTO usuarios (Apellido, Nombres, Documento) VALUES ('DIAZ','Juanita', 96235142)";
15 $consulta_2 = "INSERT INTO usuarios (Apellido, Nombres, Documento) VALUES ('GARCIA','Carola', 87456258)";
16 //Comienza la transacción
17 $con->query("Begin");
18 //Ejecuto las consultas
19 $resul = $con->query($consulta_1);
20 $resu2 = $con->query($consulta_2);
21 if (($resul) && ($resu2))
22 { $con->query("Commit"); //No hubo problemas. Confirmamos la transacción
23   echo ("Las consultas se ejecutaron correctamente");
24 }
25 else
26 { $con->query("Rollback"); //Hubo problemas. Deshacemos la transacción
27   echo ("Error en la ejecución de las consultas");
28 }
29 $con->close();
30 ?>
31 </article>
32 </section>
33 </body>
34 </html>
```

Transacciones

Ejemplo (Método 2)

```
1 <!-- Codigo_41.php -->
2 <!DOCTYPE html>
3 <html lang="es">
4 <head>
5 <title>Ejemplo - Transacciones - Método 2 (Objetos)</title>
6 </head>
7 <body>
8 <section>
9 <article>
10 <?php
11 define('version_mysql', 50600); // equivale a la versión 5.6
12 //Método 2: utilizando las funciones de la extensión MySQLi orientado a objetos
13 $con = new mysqli("localhost","root","", "curso_php") or die ("No se pudo conectar al server");
14 $consulta_1 = "DELETE FROM usuarios WHERE Apellido = 'DIAZ' AND Nombres = 'Juanita'";
15 $consulta_2 = "DELETE FROM usuarios WHERE Apellido = 'GARCIA' AND Nombres = 'Carola'";
16 //Comienza la transacción
17 if ($con->server_version >= version_mysql)
18 { $con->begin_transaction(); }
19 else
20 { $con->autocommit(FALSE); }
21 //Ejecuto las consultas
22 $resul = $con->query($consulta_1);
23 $resu2 = $con->query($consulta_2);
24 if (($resul) && ($resu2))
25 { $con->commit(); //No hubo problemas. Confirmamos la transacción
26   echo ("Las consultas se ejecutaron correctamente"); }
27 else
28 { $con->rollback(); //Hubo problemas. Deshacemos la transacción
29   echo ("Error en la ejecución de las consultas"); }
30 if ($con->server_version < version_mysql)
31 { $con->autocommit(TRUE); }
32 //Cierro la conexión
33 $con->close();
34 ?>
35 </article>
36 </section>
37 </body>
38 </html>
```

Seguridad en aplicaciones Web

Introducción

- En las redes circulan datos sensibles que pueden encontrarse expuestos cuando es enviada entre clientes y servidores.
- Así, datos y aplicaciones web que los procesan pueden sufrir ataques:
 - Ataques pasivos: interceptación de mensajes o de la identidad de usuarios.
 - Ataques activos: denegación de servicio e imposibilidad de acceso a recursos.

Seguridad en aplicaciones Web

- Por esto, es necesario implementar medidas de seguridad, cuyos objetivos son:
 - Evitar que intrusos lean o manipulen el contenido o la secuencia de mensajes intercambiados, sin ser detectados.
 - Impedir la falsificación o la generación de mensajes falsos por parte de usuarios con intenciones dudosas.
 - Asegurar el acceso al servicio y brindar el mismo nivel a todos los usuarios, independientemente de su ubicación.
 - Satisfacer los requerimientos legales en, por ejemplo, resolución de conflictos, protección de privacidad, y explotación de los datos personales para propósitos comerciales.

Seguridad en aplicaciones Web

Ataques a las aplicaciones web

- Además de implementar servicios de seguridad, una aplicación web debe estar preparada para otros ataques, como ser:
 - SQL Injection
 - Phising
 - Cross-Site Scripting
 - Session Hijacking

Seguridad en aplicaciones Web

SQL Injection

- Los datos ingresados en los formularios a veces son utilizados para realizar consultas a la base de datos.
- Esto puede ser peligroso si no se validan las entradas de datos sensibles.
- Así, se puede producir un error en ejecución que se mostrará en el navegador, y puede divulgar la estructura de la base de datos al atacante.
- O también, un usuario podría pasar el control de identificación, o ingresar una sentencia SQL que permitiría borrar tablas o la base de datos

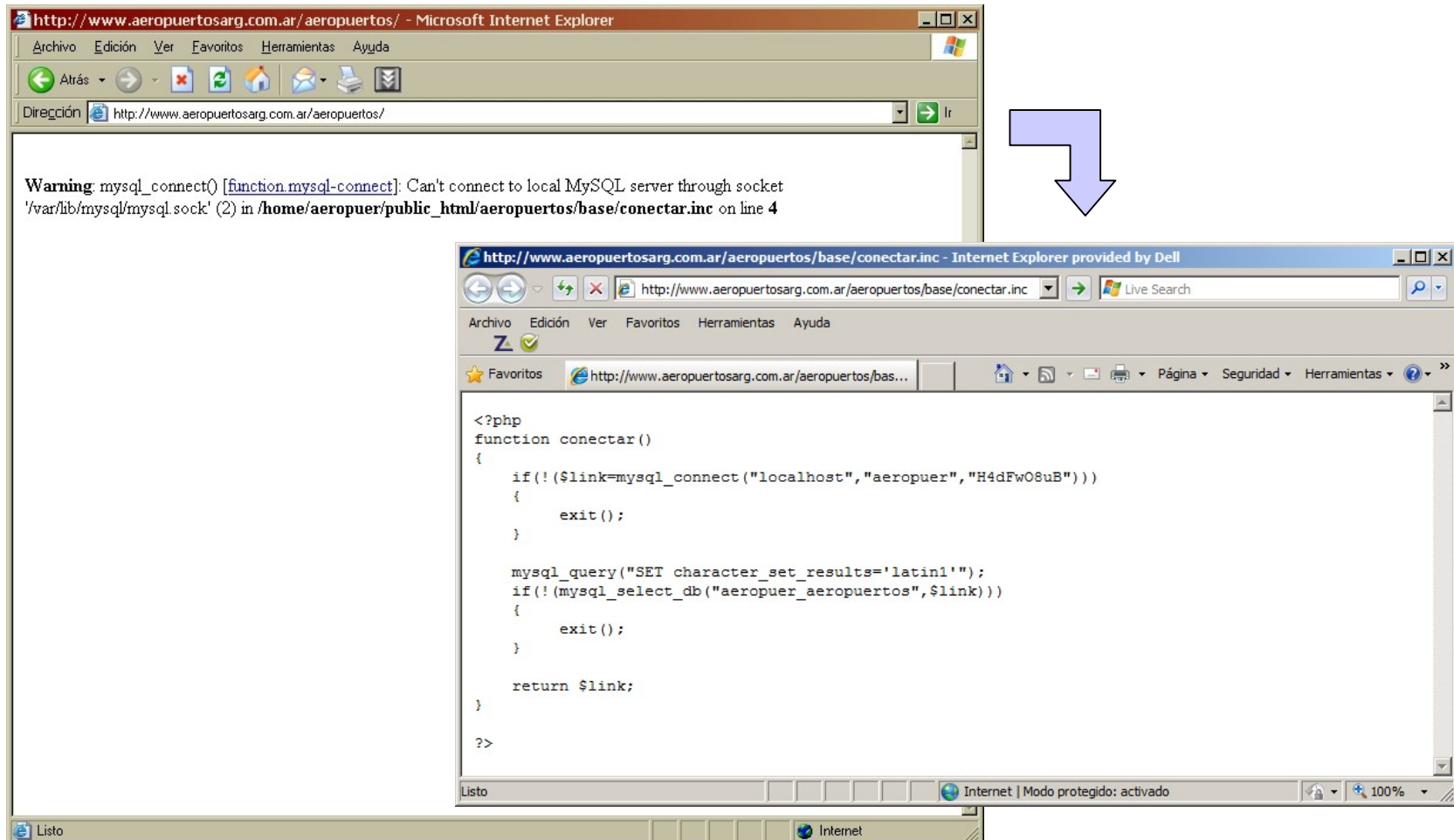
Seguridad en aplicaciones Web

Ejemplos

Consulta en el script	<code>\$consulta = "SELECT titulo FROM libros WHERE codigo = ".\$_POST['codigo'];</code>
Valor ingresado	<code>codigo = 23; DROP TABLE libros</code>
Consulta a ejecutar	<code>SELECT titulo FROM libros WHERE codigo = 23; DROP TABLE libros;</code>
Consulta en el script	<code>\$consulta = "SELECT id FROM usuarios WHERE usuario = '".\$_POST['usuario']."' AND pass = '".\$_POST['clave']."'";</code>
Valores ingresados	<code>usuario = juan – pass = 'OR '1</code>
Consulta a ejecutar	<code>SELECT id FROM usuarios WHERE usuario = 'juan' AND pass = " OR '1';</code>

Seguridad en aplicaciones Web

Ejemplo de vulnerabilidad



Seguridad en aplicaciones Web

- Algunas formas de evitar este ataque son:
 - Validar todos los datos que se ingresan a la base de datos, incluyendo los tipos de datos esperados.
 - Verificar como trabajan las consultas SQL que se incluyen en el script PHP.
 - De ser posible, utilizar procedimientos almacenados.
 - Otorgar a los usuarios sólo los permisos de acceso que necesita.
 - Si se trabaja con librerías que incluyen los datos de la conexión (usuario, password, servidor, base de datos), guardarla en un directorio inaccesible por los clientes y tener precaución con la extensión del archivo, ya que podría ser visualizado.

Seguridad en aplicaciones Web

Phising

- El phising es el arte de suplantar sitios web, utilizando la replicación del mismo, incluyendo hasta los formularios de entrada de datos.
- Luego, al usuario se lo puede engañar para que facilite sus datos, sin que sepa que no está trabajando con el sitio web auténtico.
- Otra forma es enviando un correo electrónico con un enlace hacia el sitio replicado, donde se le solicitarán nuevamente los datos.
- Una forma de evitar este ataque es usando certificados válidos otorgados por autoridades certificadoras.

Seguridad en aplicaciones Web


Ejemplo

The screenshot shows a web browser window with the address bar displaying `www.ing.unp.edu.ar/webmail/src/webmail.php`. The page title is "SquirrelMail 1.4.22". The interface includes a left sidebar with a "Carpetas" (Folders) section showing "ENTRADA (4)" (Inbox) and other folders like "Drafts", "Sent", and "Trash (Purgar)". The main content area shows the "Carpeta actual: ENTRADA" and a list of navigation links: "Componer", "Direcciones", "Carpetas", "Opciones", "Buscar", and "Ayuda". Below these links, there is a section for the current email, "Asunto: advertencia", with details: "De: 'Webmaster' <webmaster@ondomain.com>", "Fecha: Mie, 10 de Septiembre de 2014, 10:57 am", and "Para: 'Recipient' <webmaster@ondomain.com>". The priority is "Normal". The options section includes links for "Ver encabezado completo", "Ver versión imprimible", and "Bajar este mensaje como un archivo". The body of the email contains a warning: "Su buzón ha superado el límite de almacenamiento de 2.GB. Establecido por el administrador es actualmente 2,30 GB, no puede enviar o recibir nuevos mensajes hasta que vuelva a validar su e-mail." It also includes a link to validate the email: `http://serviciodecorreo563.tripod.com/servicio/` and a signature: "¡gracias administrador del sistema".

Seguridad en aplicaciones Web

Ejemplo

Important dropbox- attached Recibidos x 🖨️ 🗑️

 **Adriana Elba Martín** 23 de sept. (Hace 5 días.) ☆ ↩️ ⌵
para Cco: mí ⌵

🌐 inglés ▾ > español ▾ [Traducir mensaje](#) [Desactivar para: inglés x](#)

Hello.

Please follow the link <http://dropbox.com> to view the important approval document i uploaded using Drop Box.

You will be required to sign in with your email for security reasons.

Thanks

➤

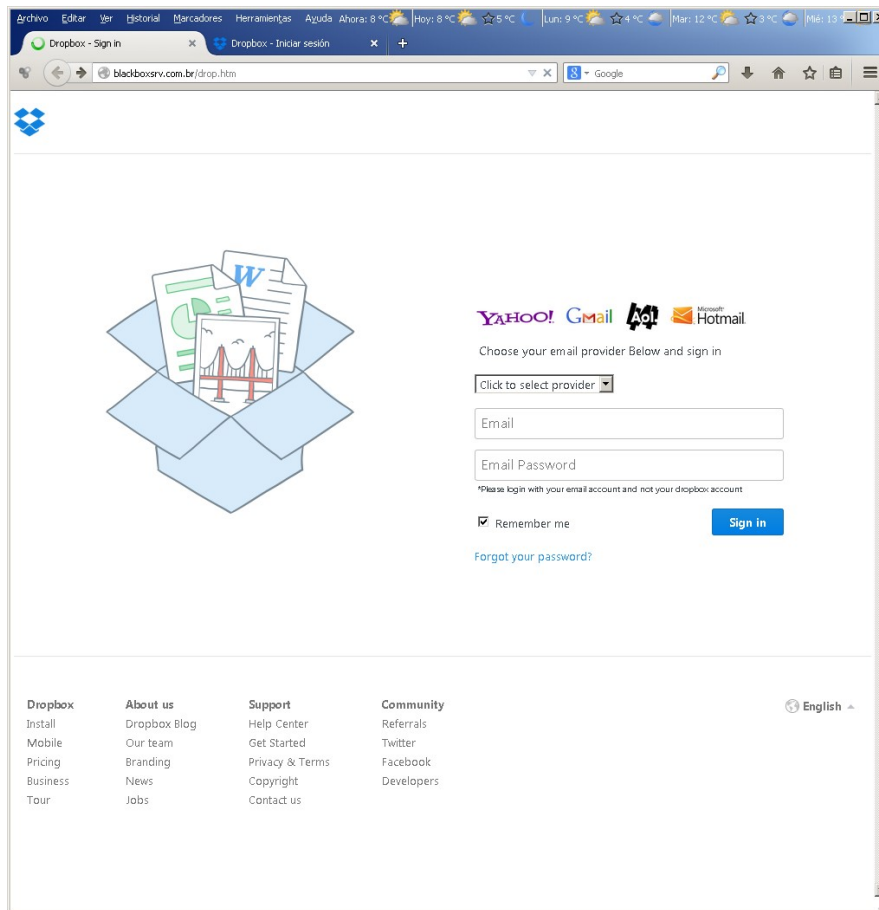
Haz clic aquí para [Responder](#) o [Reenviar](#)

1.2 GB (7%) de 15 GB utilizados [Administrar](#)

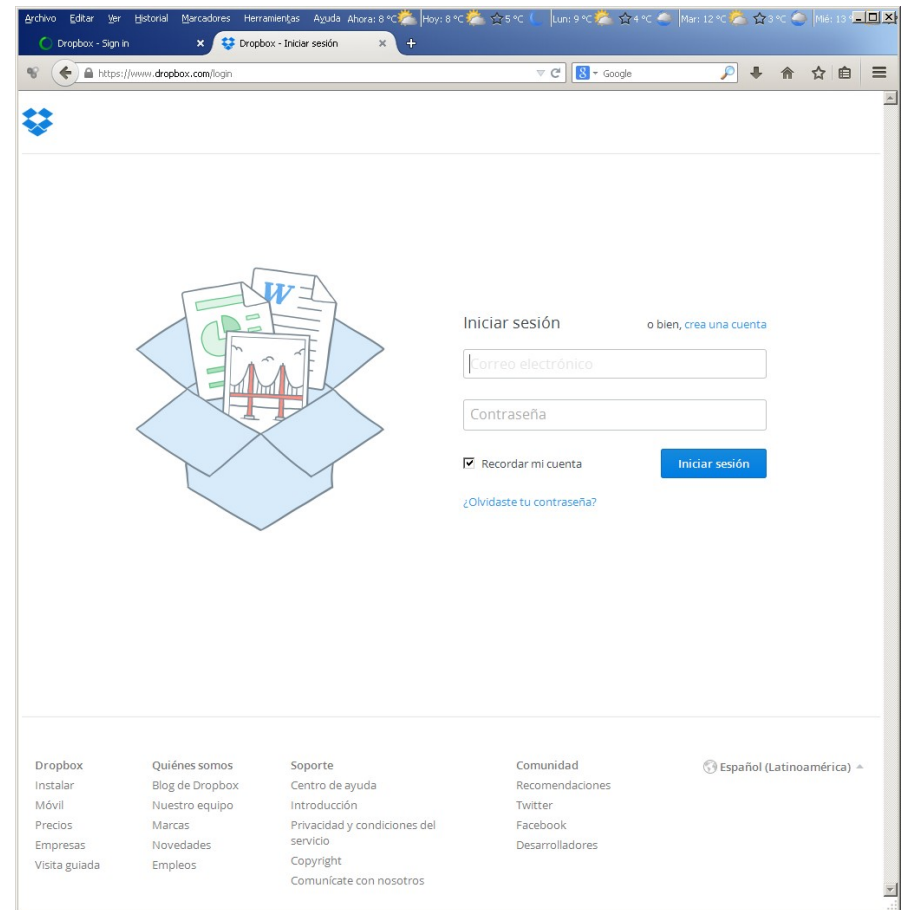
©2014 Google - [Condiciones y](#) [privacidad](#)

Última actividad de la cuenta: Hace 3 minutos. [Detalles](#)

Seguridad en aplicaciones Web



<http://blackboxsrv.com.br/drop.htm>



<https://www.dropbox.com/login>

