









**Contenido**

[Resumen](#_heading=h.gjdgxs) **4**

[Objetivo](#_heading=h.30j0zll) **4**

[**Objetivos específicos**](#_heading=h.yvy322majdv2) **4**

[Alcance](#_heading=h.1fob9te) **4**

[Roles y responsabilidades](#_heading=h.3znysh7) **4**

[Requerimientos](#_heading=h.2et92p0) **4**

[**Restricciones**](#_heading=h.e3q3lbm5jouy) **5**

[**Matriz de requerimientos**](#_heading=h.vbpe47ukoyln) **5**

[**Herramientas de desarrollo**](#_heading=h.gn3jaeopnmh8) **6**

[**Requisitos de almacenamiento**](#_heading=h.5yvchxwzxlr7) **6**

[**Desarrollo**](#_heading=h.gp0prkbwtgzz) **6**

[**Servidor local**](#_heading=h.bhuqqrpa9a10) **6**

[**Base de datos**](#_heading=h.6rcz7rl8z2ak) **9**

[**Funcionalidad**](#_heading=h.benuxekt6mu2) **12**

[Fuentes de consulta](#_heading=h.26in1rg) **20**

## Resumen

Reporte que muestra el resultado del proyecto de desarrollo de un login que implemente verificación reCaptcha.

## Objetivo

Desarrollar un login que implemente verificación reCaptcha.

## Objetivos específicos

* Realizar un Login que realice la validación de un usuario.
* El Login debe implementar verificación reCaptcha.
* Por cada acceso inválido, el sistema mostrará un mensaje de error.
* Una vez hecho login o logout, el sistema no debe permitir que se acceda a la página de Bienvenida adelantando o devolviendo la página.

## Alcance

El reporte de actividades llegará a los servidores sociales del área de Seguridad en Sistemas y al responsable inmediato de estos para notificar y validar el progreso del trabajo realizado, así como servir de apoyo para la reproducción del sistema.

## Roles y responsabilidades

A continuación, se describen los roles y responsabilidades relacionados con la actividad de realización de una máquina virtual con un servidor local que distinga direcciones seguras con https.

Responsable del área de Seguridad en Sistemas: Revisar y aprobar el desarrollo de las actividades solicitadas en la máquina virtual

Servidor social: Ejecutar las tareas solicitadas por el responsable inmediato dentro de las máquinas virtuales.

## Requerimientos

* El sistema solicitará usuario, contraseña y la realización del Captcha para poder tener acceso.
* El sistema mostrará un mensaje de error siempre que se intente un acceso incompleto o inválido.
* Las contraseñas se almacenarán en la base de datos en codificación SHA1.
* El sistema solo permitirá el acceso una vez ingresadas credenciales válidas y habiendo llenado el requisito Captcha.
* El sistema no permitirá el acceso al index de Bienvenida de usuario adelantando o devolviendo la página.

## Restricciones

* El login no da acceso a una página con alguna otra función que no sea hacer login o logout.
* El sistema no permite el uso de caracteres especiales en sus campos.

## Matriz de requerimientos

# 

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requerimiento** | **Cumplimiento** | **Observaciones** |
| **1** | El sistema solicita los siguientes datos para conceder acceso: {Usuario, Contraseña} | COMPLETO | El requerimiento se cumple en un 100% |
| **2** | El sistema solicita al usuario una validación reCaptcha para conceder el acceso | COMPLETO | El requerimiento se cumple en un 100% |
| **3** | El sistema no permitirá el acceso si el usuario intenta acceder con credenciales inválidas. | COMPLETO | El requerimiento se cumple en un 100% |
| **4** | El sistema no permitirá el acceso si el usuario intenta acceder dejando algún campo vacío. | COMPLETO | El requerimiento se cumple en un 100% |
| **5** | El sistema no permitirá el acceso si el usuario intenta acceder sin haber llenado la validación Captcha. | COMPLETO | El requerimiento se cumple en un 100% |
| **6** | El sistema arrojará un mensaje de error cada intento que el usuario ingrese credenciales inválidas | COMPLETO | El requerimiento se cumple en un 100% |
| **7** | Los passwords almacenados deben guardarse en formato codificado SHA1 | COMPLETO | El requerimiento se cumple en un 100% |
| **9** | El sistema no permite el uso de caracteres especiales | COMPLETO | El requerimiento se cumple en un 100% |

## Herramientas de desarrollo

* Sistema Operativo
  + Debian GNU/Linux 10 (buster). Linux version 4.19.0-5-amd64 .
* Apache
  + Apache/2.4.38
* Postgresql
  + PostgreSQL 11.8.
* PHP
  + PHP 7.3.22.
* JQuery
  + JQuery 3.5.1.
* Complementos
  + Google reCaptcha - https://www.google.com/recaptcha/api.js
* Editor Atom
  + Atom 1.48.0.

## Requisitos de almacenamiento

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Requisitos** | | Apache | Postgres (instalación básica) | Base de datos | jQuery | Login |
| **Espacio en disco** | **Mínimo** | 50 MB | 40 MB | 750 MB | 90 KB | 500 MB |
| **Recomendado** | 80 MB | 50 MB | 1.5 GB | 100 KB | 600 MB |

## Desarrollo

## Servidor local

Para comenzar a trabajar, es necesario tener un entorno de desarrollo y pruebas propicio para poner a prueba el sistema, encontrar y reparar los fallos pertinentes. Para lograrlo, se instaló la versión 2.4 de Apache (la más reciente al momento del inicio del desarrollo de este proyecto). Dentro de la terminal se ejecutaron los siguientes comandos:

$ sudo apt update && sudo apt -y upgrade

$ sudo apt -y install apache2

Una vez descargados e instalados todos los paquetes pertinentes, el servicio de apache quedará activo automáticamente, para comprobarlo, ejecutar el siguiente comando:

$ systemctl status apache2

De no estar activo, ejecutar el siguiente comando:

$ systemctl start apache2

Una vez activado el servicio de apache, es necesario tener el archivo de configuración necesario para definir las características básicas de nuestro servidor; para ello, es necesario dirigirse a la ruta:

/etc/apache2/sites-available

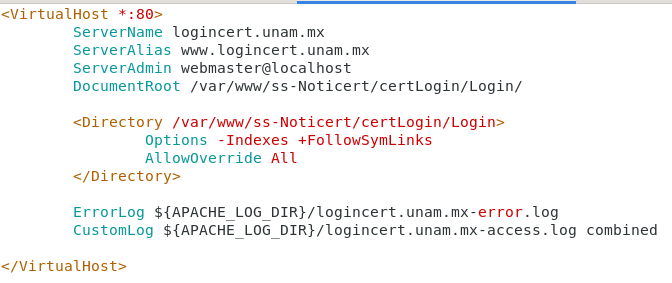
Dentro de esta ruta, encontraremos un archivo de configuración llamado “***000-default.conf***”, mismo que vamos a copiar para modificar. Es importante que como documento original no lo modifiquemos, pues es la plantilla básica de la página principal de nuestro servidor. Para copiar el archivo, ejecutar el comando:

$ sudo cp 000-default.conf logincert.unam.mx.conf

Una vez copiado el documento, vamo a modificarlo para adaptarlo a las necesidades de nuestro servidor, ejecutamos:

$ sudo vim logincert.unam.mx.conf

La terminal desplegará un espacio en el que podremos editar el contenido del archivo. Este aspecto es importante, porque determinaremos la dirección en la que nuestro sitio estará disponible, las características que tendrá y en qué archivos podremos dar seguimiento a las bitácoras de desarrollo.



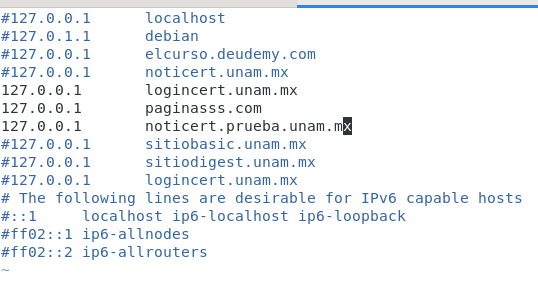
Dentro de las características del servidor, encontramos:

* VirtualHost \*:80: La etiqueta VirtualHost nos indica que las siguientes características serán aplicables para una página en entorno virtual local y que, específicamente, se comunicará con la red a través del puerto 80.
* ServerName: Que nos dará la dirección principal a la que un usuario se dirigirá cuando busque el sitio.
* ServerAlias: Es el nombre del sitio acompañado del prefijo “***www.***”
* ServerAdmin: Dirección de correo electrónico del administrador del servidor.
* DocumentRoot: Es la ruta de la carpeta donde se encuentran los archivos principales del sitio y lo hacen funcionar correctamente.
* La etiqueta “*Directory*” detalla un poco más a profundidad las características de los archivos que realizarán la función de index en el sitio.
* ErrorLog: Describe el nombre del archivo de bitácora de errores que va presentando la página durante su ejecución.
* CustomLog: Describe el nombre del archivo de bitácora de las funciones correctas que presenta el sitio durante su ejecución.

Ahora es necesario levantar el sitio, para ello es necesario levantar el sitio en el archivo de “hosts” con el comando:

vim /etc/hosts

En donde habilitaremos la dirección de nuestro sitio en la dirección por default del localhost:



Por último, se reinicia el servicio de apache con el comando:

$ sudo systemctl restart apache2

Si todo se ha configurado correctamente, el servicio de apache no debería arrojar ningún error y podremos acceder al sitio con la dirección que hemos registrado.

## Base de datos

La base de datos del sistema de Login es una base hecha en Postgresql; esta se encarga del almacenamiento de los usuarios y sus contraseñas. Primero se descargó e instaló Postgresql con el comando

$ sudo apt -y install postgresql

Una vez descargado e instalado, se puede verificar el estatus del servicio de postgresql con el comando:

$ systemctl status postgresql

Si no presenta ningún error, es necesario modificar la configuración por defecto del usuario y la contraseña, ya que al momento de realizar la instalación, el servicio tiene un usuario por defecto sin contraseña, por lo que ocuparlo resulta práctico, pero no tendría el nivel de seguridad mínimo que requiere nuestra base de datos, por lo que solucionamos el problema con los siguientes comandos, para crear un nuevo superusuario que nos permita tener un mejor control de la base de datos:

$ sudo -u postgres createuser --interactive

En donde “*-u*” nos indica el nombre del usuario que estamos creando, e “*--interactive*” solicita el nombre de usuario si no se especifica ninguno en la línea de comando. Posteriormente, nos pedirá completar los datos de registro del usuario:

$ sudo -u postgres createuser --interactive

Ingrese el nombre del rol a agregar: **postgres**

¿Será el nuevo rol un superusuario? (s/n) **S**

Con eso, se ha configurado a “postgres” como el superusuario de la base.

Para poder configurar la contraseña, primero hay que crear una base de datos para después limitar el acceso a los usuarios que quieran consumir dicha base de datos. Se ejecutan los siguientes comandos:

postgres=# \password

Ingrese la nueva contraseña:

Ingrésela nuevamente:

postgres=#

Una vez configurado el servidor de la base de datos, se configuró el archivo “***pg\_hba.conf***” que permite activar la función de validación de contraseñas con el comando:

$ sudo vim /etc/postgresql/11/main/pg\_hba.conf

Dentro del documento buscamos el apartado siguiente

# "local" is for Unix domain socket connections only

local all all md5

Entonces cambiamos el valor por defecto “md5” por el valor “peer” para activar la autenticación por contraseña, quedando de esta manera

# "local" is for Unix domain socket connections only

local all all peer

Una vez configurado el archivo, se guardan cambios y reiniciamos el servicio de postgres con el siguiente comando:

$ sudo systemctl reload postgresql

Para comprobar que todo fluye sin problemas, ejecutamos el comando:

$ systemctl status postgresql

Si el servidor está en buen estado, procedemos a crear la base de datos que utilizaremos para almacenar la información de los usuarios. Esto lo lograremos importando el documento “***pg\_ss-login.sql***”, donde se encuentra toda la información necesaria para la creación de la base de datos con algunos datos ejemplo dentro de ella. El archivo “.*sql*” debe guardarse en la siguiente ruta para mayor facilidad de encontrar el archivo:

/etc/postgres/

Ya que accederemos a la base de datos como administrador, no es necesario modificar ningún permiso. Ejecutamos el siguiente comando que nos permitirá acceder a la terminal de postgres, nos solicitará la contraseña que hemos creado recientemente.:

$ su - postgres

Una vez hayamos accedido, debemos importar el volcado de la base de datos con el siguiente comando:

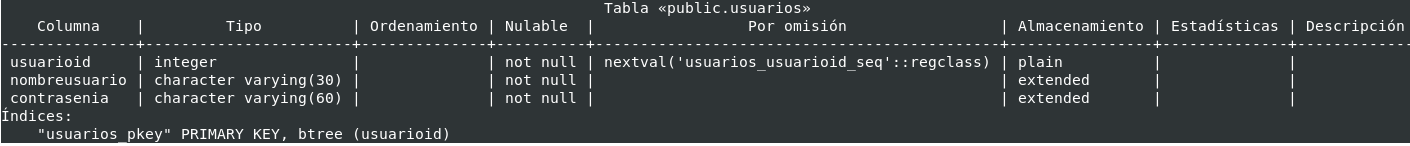
$ psql -U postgres < pg\_ss-login.sql > pg\_ss-login.log 2>&1

Que realizará la importación de la base en la sesión del usuario “postgres” y creará un documento “.log” con el nombre que determine el usuario (en este caso también se llama “pg\_ss-login”) para monitorear que todo funcione con esa importación de base de datos. Para verificar que todo ha resultado correctamente, ejecutamos el siguiente comando:

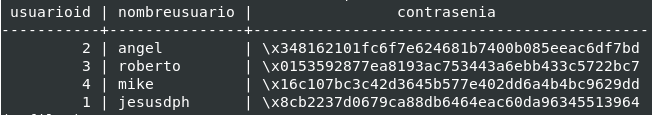
$ grep -i error pg\_ss-login.log

Si no muestra ningún error, podemos proseguir. El puerto de acceso a la base es el mismo que viene por defecto al instalar el manejador: “***5432***”, dentro, encontraremos la base de datos que hemos creado, llamada “***autenticación***” y el nombre de la única tabla dentro de ella es “***usuarios***”, para poder acceder, las credenciales de postgres son: usuario “***postgres***” con contraseña “***postgres***”. Dado que el desarrollo se hizo en un entorno completamente de prueba, no resultaba necesario reforzar los criterios de usuario y contraseña, pero se recomienda cambiarlos por cadenas más seguras en cuanto se suba a producción.

El cuerpo de la base de datos consta de solamente una tabla llamada noticias, compuesta por tres columnas: *usuarioid* de tipo numérico (int), *nombreusuario* de tipo cadena (varchar(30)) y *contrasenia* de tipo cadena (varchar(60)).



Dentro de la tabla se guardarán los registros de la página Logincert donde previamente se han agregado todas con sus respectivas contraseñas en formato codificado de SHA1..



## Funcionalidad

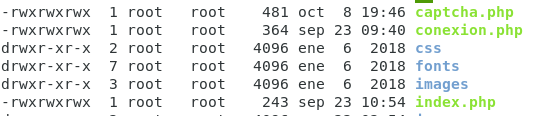
|  |  |  |
| --- | --- | --- |
| **Archivos** | **Función** | **Permisos (CHMOD)** |
| captcha.php | Contiene una relación de clave pública y clave privada que permitirá evaluar la validez del captcha en el documento “validarUsuario.php”. | Propietario: 4  Grupo: 4  Otros: 4 |
| index.php | Es la página de bienvenida, misma que nos da la opción de dirigirnos al formulario de login del archivo “login” o hacer logout si es que el usuario ha validado sus credenciales correctamente. | Propietario: 6  Grupo: 4  Otros: 4 |
| login.php | Formulario donde se reciben los datos de usuario, contraseña, captcha y que mostrará un mensaje si hay un error en la ejecución del archivo “validarUsuario.php” | Propietario: 6  Grupo: 4  Otros: 4 |
| logout.php | Página donde el usuario se puede desconectar de su sesión. | Propietario: 6  Grupo: 4  Otros: 4 |
| validarUsuario | Realiza la conexión con la base de datos y recibe los valores generados del archivo “login.php” (usuario, contraseña y captcha), realiza la consulta con la base de datos y devuelve un mensaje de error a la página de login o permite el acceso al usuario (dependiendo de si sus credenciales son correctas o no). | Propietario: 4  Grupo: 4  Otros: 4 |

Se recomienda que la descarga del sistema sea ***dentro*** en la ruta “***/var/www/https/***”, ya que suele ser lo más común, sin embargo existe la opción de realizarla en cualquier parte, sin embargo es muy importante recalcar lo siguiente: No importa en qué ruta se almacene el proyecto siempre y cuando dicha ruta se encuentre bien definida y especificada al momento de editar el documento “***logincert.unam.mx.conf***”, pues el apartado de *DocumentRoot* y los parámetros de la etiqueta *Document* deben coincidir con la ruta en donde se ha guardado.

Otro aspecto importante es verificar, es que cada archivo cuente con los permisos de acceso pertinentes como se muestra en la tabla. Para verificar esto, nos posicionamos sobre la carpeta general de nuestro proyecto y ejecutamos el comando

$ ls -la

Que nos listará todos los archivos contenidos en la carpeta, nos mostrará qué permisos tienen y a qué usuario y grupo pertenecen. Por ejemplo:



Si alguno de los archivos no tuviera los permisos mencionados en la tabla, es pertinente cambiarlos con el comando con la siguiente sintaxis:

$ sudo CHMOD [Permiso de propietario] [Permiso de grupo] [Permiso de otros] [nombre del archivo.php | .js | etc]

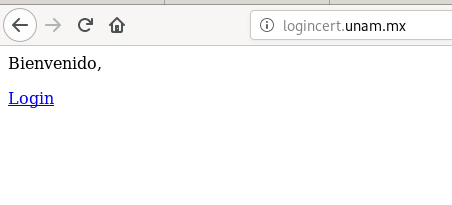
Por ejemplo:

$ sudo CHMOD 664 logout.php

Dependerá de las necesidades del usuario el nivel de permisos que concederá. A continuación una lista de la definición de los permisos:

* 0: Sin permisos
* 1: Ejecución
* 2: Escritura
* 3: Lectura y escritura
* 4: Lectura
* 5: Lectura y ejecución
* 6: Lectura y escritura
* 7: Lectura, escritura y ejecución

El sistema inicia al escribir la página “***paginasss.com***”, nombre de sitio que nosotros registramos durante el levantamiento del servidor Apache, en la página de bienvenida contenida en el archivo “***index.php***” que el servidor reconoce por defecto, donde se nos invita a realizar el login.

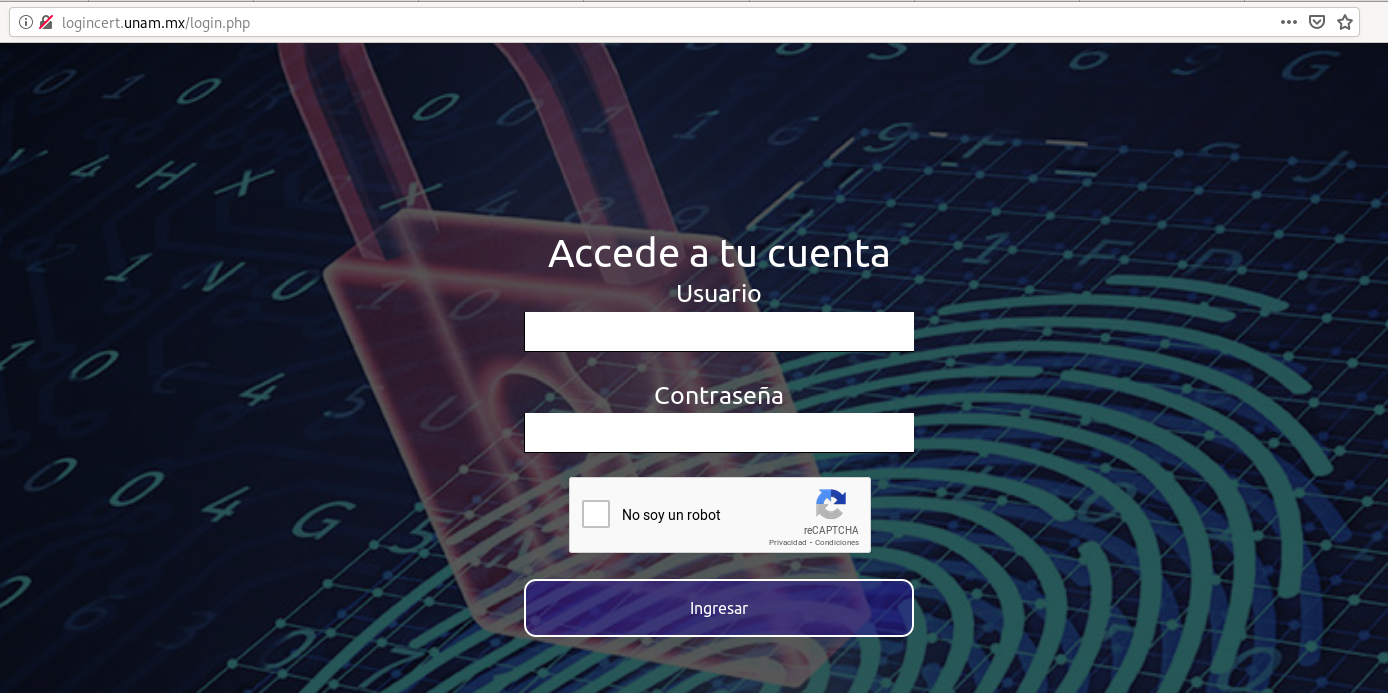


Al presionar click en Login, el sitio nos envía a la página de *login* principal, contenida en el archivo “***login.php***”, en donde es necesario un script y una línea en particular para que cumpla la función del Captcha.

<script src="<https://www.google.com/recaptcha/api.js>"></script>

requiere “captcha.php”;

Esta api y ese archivo recabarán los elementos necesarios para que la validación de estilo Captcha v2 se muestre en el sitio y con las claves necesarias, valide su confiabilidad.

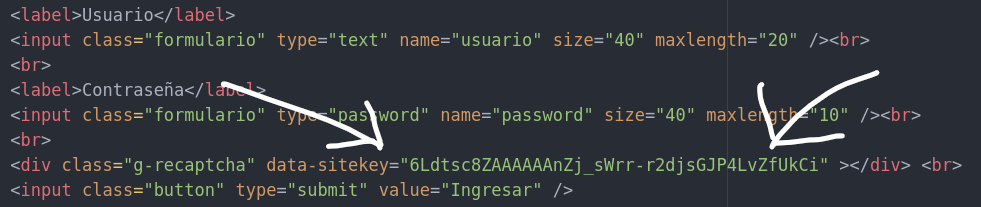


Es importante conocer de dónde proviene la verificación de captcha. Es un servicio de verificación distribuído y gestionado por Google, además de ser uno de los más populares para validar accesos.

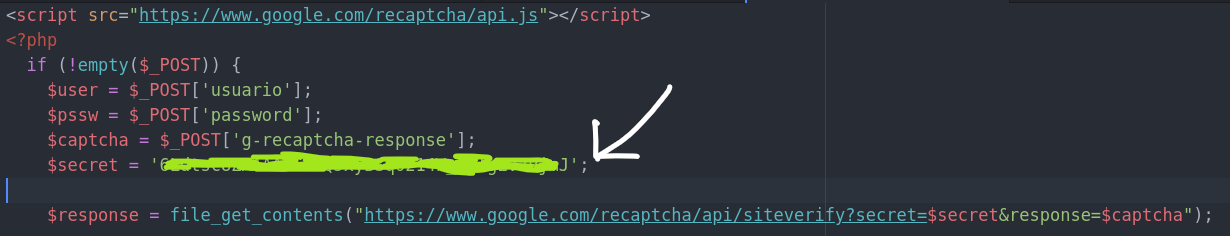
Para implementarlo, nos dirigimos a “<https://developers.google.com/recaptcha>”, donde nos registraremos con nuestra cuenta de google.com y nos dirigiremos al “Developer's Guide” (es necesario un conocimiento básico de inglés), donde nos dará una pequeña introducción de reCaptcha. Habrá que elegir la versión de la integración que nos interesa añadir a nuestro sitio, el más común suele ser reCaptcha v2 de Checkbox.

Será necesario registrar nuestro sitio y seguir las instrucciones que nos presente el sitio. Recibiremos al final dos llaves, una privada y una pública, estos son los elementos más importantes, porque son las que permitirán realizar la validación.

La llave pública la colocaremos en nuestro código principal de la siguiente manera.

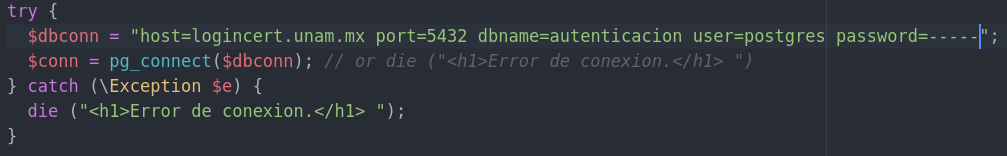


Y crearemos una función adicional para realizar la comparación con la llave privada



Donde se realizará también la validación de las credenciales de login.

En este punto podemos intentar acceder, puesto que todos los datos ingresados en el formulario, serán enviados a través del método POST al archivo “***validarUsuario.php***” el cual realiza la conexión con la base de datos para su consulta.

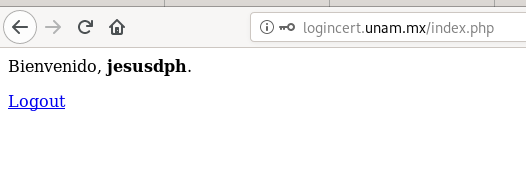


Una vez realizada la conexión con la base de datos “***autenticacion***”, el archivo “validarUsuraio.php” realizará la validación de los datos proporcionados por el usuario, empezando por reconocer caracteres especiales, verificar que no se encuentra vacío algún campo, validar que usuario y contraseña sean correctos y se encuentren relacionados ambos en la base de datos, por último, verifica que el captcha haya sido realizado a pesar de haber ingresado las credenciales adecuadas. En caso de haber accedido de manera correcta, con un usuario existente, una contraseña correcta y habiendo completado el captcha, “validarUsuario.php” redirigirá al índice principal de bienvenida de usuario.

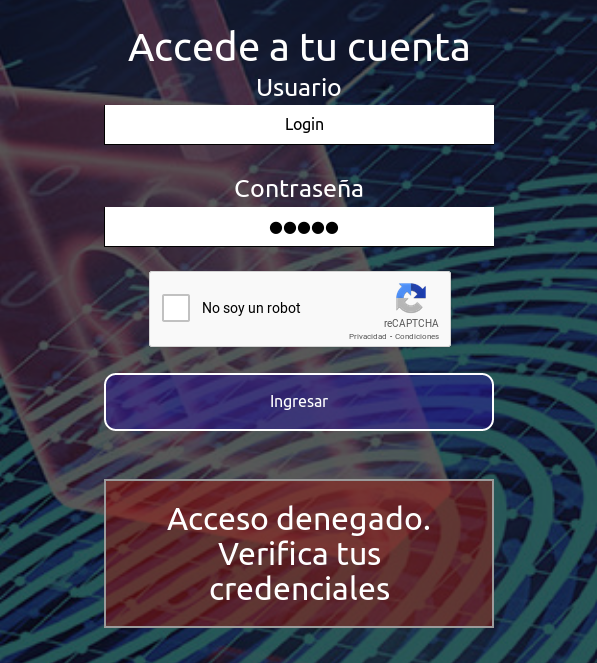
Cuando realizamos un acceso correcto, habiendo completado todos los campos y el captcha:



El login se realiza de manera correcta y accedemos sin problema a la página principal de bienvenida del usuario, presentando el mensaje “Bienvenido [usuario]”.



En caso de haber metido un dato incorrecto como usuario o contraseña:

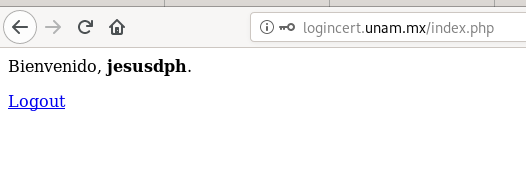


El login lanzará un mensaje de error correspondiente al tipo de error encontrado. Y, por ejemplo, si accedemos con datos correctos, pero sin completar el captcha:

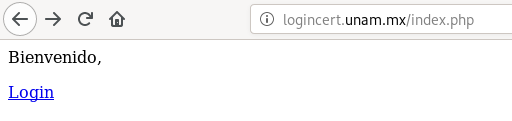


El login lanza un mensaje de error especificando lo que falló en el acceso, en este caso muestra que faltó completar el captcha.

Suponiendo que logramos realizar un Login correcto por completo, al hacer logout, no debe permitirnos volver a la página de bienvenida.



Por lo que no volverá al login de bienvenida del usuario, solo la bienvenida general, por lo que no se logra acceder desde otro sitio.



## Fuentes de consulta

* Youtube
  + LexterXPS. (2016). Login y Autenticación de Tipos de Usuario con PHP + MySQL + jQuery AJAX. Lugar de publicación: Youtube: https://www.youtube.com/watch?v=sueRUsVF2U8&t=1s
* Apache
  + Apache. (2020). VirtualHost Examples. Lugar de publicación: apache.org: https://httpd.apache.org/docs/2.4/vhosts/examples.html
* PHP
  + PHP. (2020). Pg\_connect. Lugar de publicación: php.net/manual: <https://www.php.net/manual/es/function.pg-connect>
  + PHP. (2020). “$\_POST”. Lugar de publicación: php.net/manual: https://www.php.net/manual/es/reserved.variables.post
* Postgres
  + PostgreSQL. (-). PostgreSQL 9.2.24 Documentation. Lugar de publicación: PostgreSQL: <https://www.postgresql.org/docs/9.2/app-createuser.html>
  + Techlandia. (2018). Cómo restablecer una contraseña PostgreSQL. Lugar de publicación: Techlandia: https://techlandia.com/restablecer-contrasena-postgresql-como\_366486/
* W3SCHOOLS
  + w3schools. (-). HTML Styles. Lugar de publicación: w3schools: https://www.w3schools.com/html/html\_styles.asp