

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA

ASIGNATURA:	PROGRAMACIÓN WEB 1							
TÍTULO DE LA PRÁCTICA:	<i>TRABAJO FINAL</i>							
NÚMERO DE PRÁCTICA:	8	AÑO LECTIVO:	2024 - B	NRO. SEMESTRE:	II			
FECHA DE PRESENTACIÓN	16/12/2024	HORA DE PRESENTACIÓN	<i>16:00:00 PM</i>					
INTEGRANTE (s):	<p>Karla Miluska Bedregal Coaguila Oroche Yajo Paola Fernanda</p>							
DOCENTE:	<i>M.Sc. Richart Escobedo Quispe</i>							

SOLUCIÓN Y RESULTADOS

I. PROYECTO

DOWNLOADERBYTE

Descripción:

El proyecto consiste en una aplicación web que permite a los usuarios registrarse, iniciar sesión y acceder a una plataforma para descargar videos de YouTube en diversos formatos (MP3, MP4 y AVI). Después de ingresar un enlace de YouTube, el usuario puede elegir el formato de descarga y proceder con la descarga. Una vez completada, los archivos descargados se almacenan en una base de datos, organizada según la lista seleccionada antes de la descarga.

La plataforma incluye un CRUD (Crear, Leer, Actualizar, Eliminar) que permite modificar el nombre de las canciones descargadas o eliminarlas de la lista. Además, el sistema ofrece la opción de cambiar propiedades de los videos, como resolución, proporción de imagen y codificación de audio. Las canciones modificadas se almacenan en un arreglo y se ponen nuevamente a disposición del usuario para su conversión.

Para hacer más ameno el proceso de descarga, se han integrado pequeños juegos en JavaScript, permitiendo al usuario disfrutar mientras se procesan los archivos. Además, la aplicación incluye una página de vista previa de los videos convertidos y descargados, proporcionando al usuario una experiencia completa desde la descarga hasta la visualización de los archivos.

II. EQUIPOS, MATERIALES Y TEMAS UTILIZADOS

- Sistema Operativo Ubuntu GNU Linux 22.04 Kernel 6.8.0-47-generic
- Visual Studio Code
- Editor de textos
- GitHub
- CGI::Session
- AJAX
- JSON
- XMLHttpRequest
- CGI (Common Gateway Interface)
- FFmpeg
- yt-dlp
- YouTube
- MariaDB
- MySQL
- Apache2
- UTF-8
- File::Spec
- Repositorio en github

III. URL DE REPOSITORIO GITHUB

<https://github.com/KarlaBedregal/Proyecto-Final-Pweb1.git>

Commits:

Commits

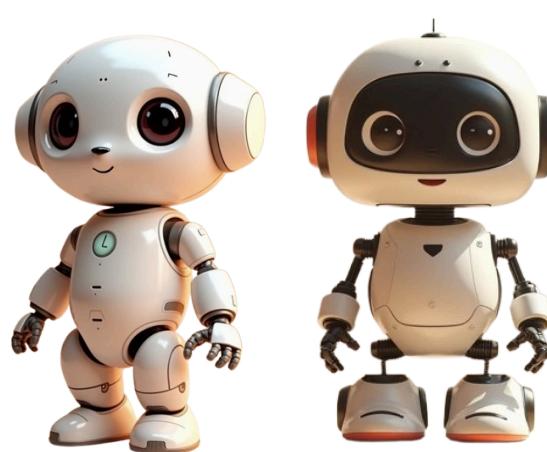
karla All users All time

- o- Commits on Dec 16, 2024
 - añadiendo mas archivos
KarlaBedregal committed 3 hours ago e483cb0
- o- Commits on Dec 15, 2024
 - actualizando backend
KarlaBedregal committed 17 hours ago 0eba7a5
 - incluyendo crud
KarlaBedregal committed yesterday 96c7f54
 - agregando cambios con la base de datos de canciones descargadas
KarlaBedregal committed yesterday c32dc72
- o- Commits on Dec 14, 2024
 - actualizando union descargador y sesion
KarlaBedregal committed 2 days ago 6e9a0c9
- o- Commits on Dec 13, 2024
 - Actualizando formularios y base de datos
KarlaBedregal committed 3 days ago 9b8cbdc
- o- Commits on Dec 11, 2024
 - añadiendo los perl del descargador
KarlaBedregal committed 5 days ago 2d49e5c
- o- Commits on Dec 10, 2024
 - Insertando los archivos iniciales
KarlaBedregal committed last week 6a2f5f8

IV. ESTRUCTURA DEL PROYECTO

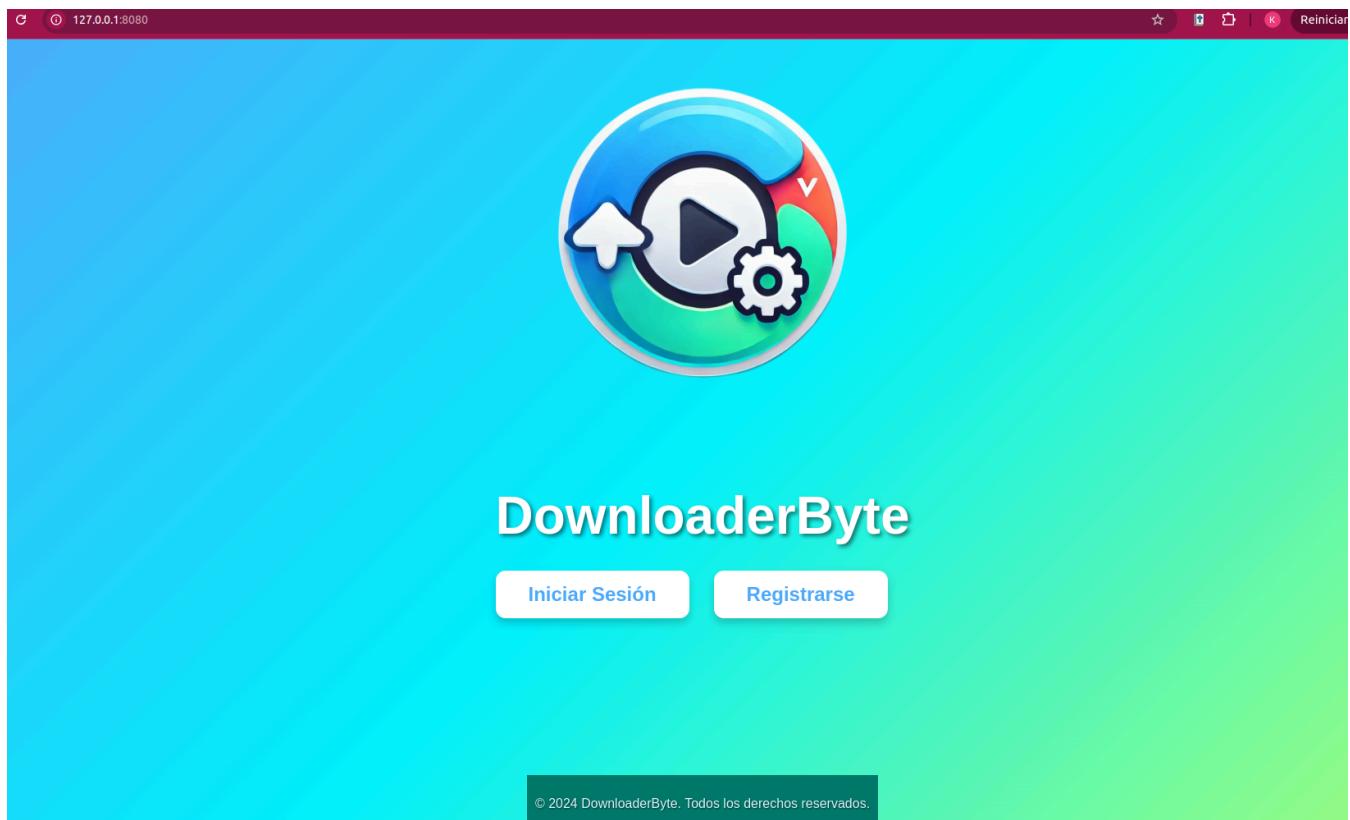
```
tnal/proyectoFinal$ tree
.
├── cgi-bin
│   ├── cerrar_sesion.pl
│   ├── config.pl
│   ├── delete_song.pl
│   ├── descargador.pl
│   ├── descargas.pl
│   ├── iniciar_sesion.pl
│   ├── registro.pl
│   ├── update_song.pl
│   └── ver_canciones.pl
├── Dockerfile
└── html
    ├── config.css
    ├── config.html
    ├── descargador.html
    ├── estilo1.css
    ├── estilosdescargador.css
    ├── index.html
    ├── iniciar_sesion.html
    ├── registro.html
    └── styles.css
    └── images
        ├── iconodownloaderbyte.png
        └── imagen.jpg
    └── init.sql
    └── readme.md
```

V. ICONO Y MASCOTAS DEL PROYECTO



VI. EJECUCIÓN DEL PROYECTO

PANTALLA 1



PANTALLA 2

127.0.0.1:8088/registro.html

Registrarse

Nombre: Apellido:

Fecha de Nacimiento: dd / mm / aaaa

Nombre de Usuario:

Correo Electrónico:

Contraseña: Confirmar Contraseña:

Edad:

Género: Masculino

Teléfono:

Registrar

PANTALLA 3

127.0.0.1:8088/iniciar_sesion.html

Iniciar Sesión

Nombre de Usuario:

Contraseña:

Iniciar Sesión

PANTALLA 4

127.0.0.1:8088/descargador.html

DOWNLOADERBYTE

Elije el nombre para tu lista:

Personalizar lista

Score: 1

URL del Video de YouTube:

Descargar en MP4

Descargar en MP3

Descargar en AVI

Historial

Mis Descargas

PANTALLA 5

Lista: dfsbdf

ID	Nombre Canción	URL	Directorio	Fecha Descarga	Acciones
1	On My Own	https://www.youtube.com/watch?v=TyUQiNmNug0	mp3	2024-12-16 20:43:53	Editar Eliminar

Lista: erht

ID	Nombre Canción	URL	Directorio	Fecha Descarga	Acciones
2	Luke Combs - Fast Car (Official Live Video)	https://www.youtube.com/watch?v=Fr7oYjnt3bM	mp3	2024-12-16 20:44:32	Editar Eliminar

[Volver](#)

[Cambiar las propiedades de mis descargas](#)

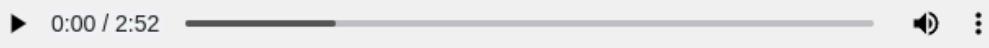
PANTALLA 6

Conversi?n en proceso...

Convertiendo el video 'dfsbd/On_My_Own-00001.mp3' de mp4 a mp4...

El video ha sido convertido con ?xito. :)

El video ha sido convertido con ?xito. Puedes descargarlo [aqu?.](#)



Configuraci?n

Convertir De a

Seleccione el video

Propiedades del video de salida

Resoluci?n

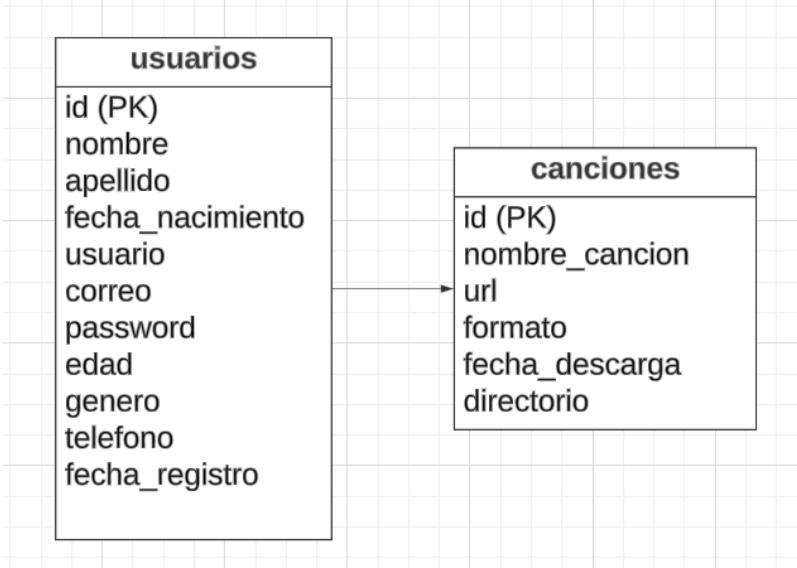
Proporci?n de la imagen

Propiedades del audio de salida

Codificaci?n de audio

Frecuencia

VI. DIAGRAMA DE BASE DE DATOS



VII. CÓDIGO FUENTE DEL PROYECTO

- PERL

1. registro.pl

Este código Perl recibe datos de un formulario web, valida que las contraseñas coincidan, que el teléfono tenga 9 dígitos y que no falten campos obligatorios. Luego, verifica si el usuario o correo ya existen en la base de datos. Si no están registrados, encripta la contraseña y guarda los datos en la base de datos. Si el registro es exitoso, redirige al usuario a la página de inicio de sesión.

```

1  #!/usr/bin/perl
2  use CGI qw(:standard);
3  use DBI;
4  use utf8;
5  binmode(STDOUT, ":utf8");
6
7  my $q = CGI->new;
8
9  # Configuración de conexión a la base de datos
10 my $dsn = "DBI:mysql:database=mi_base_de_datos;host=localhost";
11 my $usuario = "mi_usuario";
12 my $contraseña = "mi_contraseña";
13 my $dbh = DBI->connect($dsn, $usuario, $contraseña) or die "No se pudo conectar a la base de datos: $DBI::errstr";
14
15 # Obtenemos datos del formulario
16 my $nombre = param('nombre');
17 my $apellido = param('apellido');
18 my $fecha_nacimiento = param('fecha_nacimiento');
19 my $usuario = param('usuario');
20 my $correo = param('correo');
21 my $password = param('password');
22 my $confirmar_password = $q->param('confirmar_password');
23 my $edad = param('edad');
24 my $genero = param('genero');
25 my $telefono = param('telefono');
26
27 if ($password ne $confirmar_password) {
28     print $q->header(-type => 'text/html');
29     print $q->start_html('Error');
30     print "Las contraseñas no coinciden. Por favor, intente nuevamente.";
31     print $q->end_html;
32     exit;
33 }
34
35 if ($telefono !~ /\^d{9}\$/) {
36     print $q->header(-type => 'text/html');
37     print $q->start_html('Error');
38     print "El teléfono debe contener exactamente 9 dígitos numéricos.";
39     print $q->end_html;
40     exit;
41 }
42
43 if (!$nombre || !$apellido || !$fecha_nacimiento || !$usuario || !$correo || !$password || !$edad || !$genero || !$telefono) {
44     print "Content-type: text/html\n\n";
45     print header();
46     print "<h1>Error: Todos los campos son obligatorios.</h1>";
47     exit;
48 }
49
50 # Consulta para verificar si ya existe el usuario o correo
51 my $sth_check = $dbh->prepare("SELECT COUNT(*) FROM usuarios WHERE usuario = ? OR correo = ?");
52 $sth_check->execute($username, $email) or die "Error al verificar el usuario: $DBI::errstr";
53 my ($exists) = $sth_check->fetchrow_array();
54
55 if ($exists > 0) {
56     # Usuario ya registrado
57     print "Content-type: text/html\n\n";
58     print header();
59     print "<h1>Error: El usuario o correo ya está registrado.</h1>";
60     print "<p><a href='../iniciar_sesion.html'>Iniciar sesión</a></p>";
61 } else {
62     # Encriptar la contraseña
63     use Digest::SHA qw(sha256_hex);
64     my $hashed_password = sha256_hex($password);
65
66     my $sth_insert = $dbh->prepare("INSERT INTO usuarios (nombre, apellido, fecha_nacimiento, usuario, correo, password, edad, genero, telefono) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
67     $sth_insert->execute($nombre, $apellido, $fecha_nacimiento, $usuario, $correo, $hashed_password, $edad, $genero, $telefono);
68
69     # Respuesta al usuario
70     print "Content-type: text/html\n";
71     print "Location: ..../iniciar_sesion.html\n\n";
72     exit;
73
74     $sth_insert->finish();
75 }
76
77 # Cerrar la conexión a la base de datos
78 $sth_check->finish();
79 $dbh->disconnect();

```

2. iniciar_sesion.pl

Este código Perl maneja el inicio de sesión de un usuario. Recibe el nombre de usuario y la contraseña desde un formulario web, encripta la contraseña con SHA-256 y verifica si

las credenciales coinciden con algún registro en la base de datos. Si las credenciales son correctas, redirige al usuario a la página principal. Si son incorrectas, muestra un mensaje de error e invita al usuario a registrarse o intentarlo nuevamente.

```
1 #!/usr/bin/perl
2 use CGI qw(:standard);
3 use DBI;
4 use Digest::SHA qw(sha256_hex);
5 use utf8;
6 binmode(STDOUT, ":utf8");
7
8 # Configuración de conexión a la base de datos
9 my $dsn = "DBI:mysql:database=mi_base_de_datos;host=localhost";
10 my $usuario = "mi_usuario";
11 my $contraseña = "mi_contraseña";
12 my $dbh = DBI->connect($dsn, $usuario, $contraseña) or die "No se pudo conectar a la base de datos: $DBI::errstr";
13
14 # Obtener datos del formulario
15 my $username = param('usuario');
16 my $password = param('password');
17
18 my $hashed_password = sha256_hex($password);
19
20 my $sth = $dbh->prepare("SELECT * FROM usuarios WHERE usuario = ? AND password = ?");
21 $sth->execute($username, $hashed_password);
22
23 # Verificar si hay un usuario con las credenciales proporcionadas
24 if (my $row = $sth->fetchrow_hashref) {
25     # Redirigir a la página principal
26     print "Content-type: text/html\n";
27     print "Location: ../descargador.html\n\n";
28 } else {
29     print header();
30     print "<h1>Error de inicio de sesión</h1>";
31     print "<p>El nombre de usuario o la contraseña son incorrectos. Regístrate aquí -> <a href='../../registro.html'>Int";
32 }
33
34 $sth->finish();
35 $dbh->disconnect();
36
```

3. cerrar_sesion.pl

Este código Perl maneja el cierre de sesión de un usuario. Al ejecutarse, imprime un mensaje de éxito indicando que la sesión se ha cerrado correctamente y proporciona un enlace para regresar a la página principal. Utiliza la función header() para definir el encabezado HTTP y start_html() y end_html() para generar la estructura HTML de la página.

```
1 #!/usr/bin/perl
2 use CGI qw(:standard);
3 use strict;
4 use warnings;
5 use utf8;
6 binmode(STDOUT, ":utf8");
7
8 print header();
9 print start_html("Cerrar sesión");
10 print "<h1>Sesión cerrada exitosamente</h1>";
11 print "<p>Has cerrado tu sesión. Haz clic <a href='../../html/index.html'>aquí</a> para volver a la página principal</p>";
12 print end_html();
13
```

4. descargador.pl

Este código Perl gestiona la descarga de videos desde una URL proporcionada por el usuario. Primero, obtiene parámetros del formulario, como la URL del video y la acción

de descarga (MP4, MP3, o AVI). Si se proporciona un nuevo directorio, crea ese directorio en el sistema de archivos. Luego, dependiendo de la acción, ejecuta un comando `yt-dlp` para descargar el video en el formato seleccionado. Si la descarga es exitosa, guarda los detalles del video en una base de datos MySQL. Además, proporciona un enlace para ver las canciones descargadas y un botón para cerrar sesión. También maneja errores como URL inválidas o problemas al crear directorios.

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4  use CGI qw(:standard);
5  use File::Path qw(make_path);
6  use File::Spec;
7  use Encode;
8  use DBI;
9  use File::Basename;
10 use utf8; # Asegura que el código fuente esté en UTF-8
11 binmode(STDOUT, ":utf8");
12
13 my $cgi = CGI->new;
14
15 # Configuración de conexión a la base de datos
16 my $dsn = "DBI:mysql:database=mi_base_de_datos;host=localhost";
17 my $usuario = "mi_usuario";
18 my $contraseña = "mi_contraseña";
19 my $dbh = DBI->connect($dsn, $usuario, $contraseña) or die "No se pudo conectar a la base de datos: $DB
20
21
22 print $cgi->header('text/html; charset=UTF-8');
23 print $cgi->start_html('Descargar Video');
24
25 my $is_ajax = $cgi->param('directorio');
26
27 if ($is_ajax) {
28     # Solicitud AJAX: Solo actualizar el directorio y devolver la respuesta
29     my $directorio = $cgi->param('directorio') || 'Mis favoritos';
30     print $directorio;
31 } else {
32     # Parámetros del formulario
33     my $url = $cgi->param('url');
34     my $directorio = $cgi->param('select') || 'favoritos';
35     my $nuevo_directorio = $cgi->param('nuevo_directorio') || '';
36     my $accion = $cgi->param('action');
37
38     # Si se recibe un nuevo directorio, limpiamos caracteres no válidos
39     if ($nuevo_directorio) {
40         $nuevo_directorio =~ s/[^a-zA-Z0-9\_\-]//g; # Solo caracteres válidos
41         if ($nuevo_directorio eq '') {
42             print "<p>Error: El nombre del nuevo directorio no es válido.</p>";
43             print $cgi->end_html;
44             exit;
45     }
46 }
```

```

48 # Determina el directorio de descarga
49 my $path;
50 if ($nuevo_directorio) {
51     $path = File::Spec->catfile('/descargas', $nuevo_directorio);
52     # Crear el directorio si no existe y manejar errores
53     eval { make_path($path) unless -d $path; };
54     if ($@) {
55         print "<p>Error al crear el directorio '$nuevo_directorio': $@</p>";
56         print $cgi->end_html;
57         exit;
58     }
59     print "<p>Directorio personalizado creado: $nuevo_directorio</p>";
60 } else {
61     $path = File::Spec->catfile('/descargas', $directorio);
62     eval { make_path($path) unless -d $path; };
63     if ($@) {
64         print "<p>Error al crear el directorio '$directorio': $@</p>";
65         print $cgi->end_html;
66         exit;
67     }
68 }
69
70 # Verificar si la URL es válida
71 if ($url && $url =~ /^https?:\/\/[^\s]+$/) {
72     my $download_command;
73     my $extension;
74
75     # Construir el comando de descarga según la acción seleccionada
76     if ($accion eq 'descargar_mp4') {
77         $extension = 'mp4';
78         $download_command = "yt-dlp --no-overwrites -f mp4 --restrict-filenames -o '$path/%(title)s-%(a";
79     } elsif ($accion eq 'descargar_mp3') {
80         $extension = 'mp3';
81         $download_command = "yt-dlp --no-overwrites -x --audio-format mp3 --restrict-filenames -o '$pat";
82     } elsif ($accion eq 'descargar_avi') {
83         $extension = 'avi';
84         $download_command = "yt-dlp --no-overwrites --recode-video avi --restrict-filenames -o '$path/%";
85     } else {
86         print "<p>Error: Acción no válida.</p>";
87         print $cgi->end_html;
88         exit;
89     }

```

```

91     # Ejecutar el comando de descarga
92     my $download_output = `$download_command 2>&1`;
93     my $directorio_guardado = $nuevo_directorio || $directorio;
94     if ($? != 0) {
95         print "<p>Error al descargar el video: $download_output</p>";
96     } else {
97         print "<h2>Video descargado exitosamente en formato $extension.</h2>";
98         my $filename = "$path/" . (glob "$path/*.$extension")[0];
99         my $metadata_json = `yt-dlp -j $url`;
100
101        use JSON;
102        my $metadata = decode_json($metadata_json);
103
104        # Extraer los datos de la URL del video
105        my $nombre_cancion = $metadata->{title} || 'Desconocido';
106        $directorio_guardado = $nuevo_directorio || $directorio;
107
108        my $insertardatosvideos = $dbh->prepare("INSERT INTO canciones (nombre_cancion, url, formato");
109        $insertardatosvideos->execute($nombre_cancion, $url, $extension, $directorio_guardado);
110    }
111
112    print "<form action='/cgi-bin/ver_canciones.pl' method='get'>";
113    print "<input type='hidden' name='directorio' value='$directorio_guardado'>";
114    print "<input type='submit' value='Ver canciones descargadas'>";
115    print "</form>";
116
117 } else {
118     print "<p>Error: Proporcione una URL válida.</p>";
119 }
120
121 <form action="cerrar_sesion.pl" method="get">
122 |   <button type="submit">Cerrar Sesión</button>
123 </form>
124
125 }
126 # Finalizar HTML
127 print $cgi->end_html;

```

5. descargas.pl

Este código Perl gestiona la visualización, edición y eliminación de canciones almacenadas en una base de datos. Al principio, se conecta a la base de datos MySQL y obtiene la acción y el ID de la canción que se desea editar o eliminar. Si se recibe una solicitud de eliminación, elimina la canción correspondiente de la base de datos.

Luego, consulta y muestra una lista de directorios distintos donde se almacenan las canciones. Para cada directorio, consulta las canciones dentro de él y las muestra en una tabla con opciones para editar o eliminar cada canción. Los enlaces para editar y eliminar canciones usan funciones AJAX para interactuar con el servidor sin recargar la página.

Además, incluye un modal para editar el nombre de la canción, con un formulario que se activa al hacer clic en "Editar". Al enviar la actualización, se usa un XMLHttpRequest para enviar los datos al script update_song.pl y actualizar el nombre de la canción en la base de datos. Similarmente, el enlace de eliminar envía una solicitud para borrar la canción.

También hay un botón para volver atrás y otro para cambiar las propiedades de las descargas.

Este código incluye la interacción con la base de datos y la actualización dinámica de la página mediante AJAX, haciendo que la experiencia del usuario sea fluida y sin recargar la página.

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4  use CGI qw(:standard);
5  use DBI;
6  use utf8;
7  binmode(STDOUT, ":utf8");
8
9  my $cgi = CGI->new;
10
11 # Configuración de conexión a la base de datos
12 my $dsn = "DBI:mysql:database=mi_base_de_datos;host=localhost";
13 my $usuario = "mi_usuario";
14 my $contraseña = "mi_contraseña";
15 my $dbh = DBI->connect($dsn, $usuario, $contraseña) or die "No se pudo conectar a la base de datos: $DBI::errstr";
16
17 # Obtener la acción (editar, eliminar) y el ID de la canción a editar/eliminar
18 my $accion = $cgi->param('action') || '';
19 my $id_cancion = $cgi->param('id_cancion') || '';
20
21 # Si se recibe una acción de eliminar
22 if ($accion eq 'eliminar' && $id_cancion) {
23     my $sth = $dbh->prepare("DELETE FROM canciones WHERE id = ?");
24     $sth->execute($id_cancion);
25     print $cgi->header('text/html; charset=UTF-8');
26     print "<h3>Canción eliminada con éxito.</h3>";
27 }
28
29 # Consultar la base de datos
30 my $query_directorio = "SELECT DISTINCT directorio FROM canciones";
31 my $sth_directorio = $dbh->prepare($query_directorio);
32 $sth_directorio->execute();
33
34 print $cgi->header('text/html; charset=UTF-8');
35 print $cgi->start_html('Tabla de Canciones');
36
37 while (my $dir_row = $sth_directorio->fetchrow_hashref) {
38     my $directorio = $dir_row->{directorio};
39     print "<h2>Lista: $directorio</h2>";
40
41     # Consultar las canciones de ese directorio
42     my $query_canciones = "SELECT * FROM canciones WHERE directorio = ?";
43     my $sth_canciones = $dbh->prepare($query_canciones);
44     $sth_canciones->execute($directorio);
```

```
46     print "<table border='1'>";
47     print "<tr><th>ID</th><th>Nombre Canción</th><th>URL</th><th>Directorio</th><th>Fecha Descarga</th>";
48
49     while (my $row = $sth_canciones->fetchrow_hashref) {
50         print "<tr>";
51         print "<td>" . $row->{id} . "</td>";
52         print "<td>" . $row->{nombre_cancion} . "</td>";
53         print "<td>" . $row->{url} . "</td>";
54         print "<td>" . $row->{formato} . "</td>";
55         print "<td>" . $row->{fecha_descarga} . "</td>";
56
57         # Enlaces para editar y eliminar con AJAX
58         print "<td>";
59         print "<a href='#' onclick='editarCancion(" . $row->{id} . ", \" . $row->{nombre_cancion} . \"";
60         print "<a href='#' onclick='eliminarCancion(" . $row->{id} . ")'>Eliminar</a> |";
61         print "</td>";
62     }
63
64     print "</table>";
65 }
66
67 # Agregar el modal de edición
68 print <<'HTML';
69 <div id="modalEdit" style="display:none;">
70     <div style="background-color:#fff; padding:20px; width:300px; margin:auto; border:1px solid #ccc;
71         <h3>Editar Canción</h3>
72         <form id="editForm">
73             <input type="hidden" id="editId">
74             <label for="nombre_cancion">Nuevo nombre: </label>
75             <input type="text" id="editNombre" required>
76             <br><br>
77             <button type="button" onclick="submitEdit()">Actualizar</button>
78             <button type="button" onclick="cerrarModal()">Cerrar</button>
79         </form>
80     </div>
81 </div>
```

```

74 <div style="margin-top:20px;">
75   <button onclick="history.back()">Volver</button>
76   <form action="config.pl" method="get">
77     <button type="submit">Cambiar las propiedades de mis descargas</button>
78   </form>
79 </div>
80 <script>
81   function editarCancion(id, nombre) {
82     document.getElementById('editId').value = id;
83     document.getElementById('editNombre').value = nombre;
84     document.getElementById('modalEdit').style.display = 'block';
85   }
86   function cerrarModal() {
87     document.getElementById('modalEdit').style.display = 'none';
88   }
89   function submitEdit() {
90     var id = document.getElementById('editId').value;
91     var nombre = document.getElementById('editNombre').value;
92
93     var xhr = new XMLHttpRequest();
94     xhr.open('POST', 'update_song.pl', true);
95     xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
96     xhr.onreadystatechange = function () {
97       if (xhr.readyState == 4 && xhr.status == 200) {
98         alert('Canción actualizada');
99         cerrarModal();
100        location.reload();
101      }
102    };
103    xhr.send('action=editar&id_cancion=' + id + '&nombre_cancion=' + encodeURIComponent(nombre));
104  }
105  function eliminarCancion(id) {
106    if (confirm('¿Estás seguro de que quieres eliminar esta canción?')) {
107      var xhr = new XMLHttpRequest();
108      xhr.open('POST', 'delete_song.pl', true);
109      xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
110      xhr.onreadystatechange = function () {
111        if (xhr.readyState == 4 && xhr.status == 200) {
112          alert('Canción eliminada');
113          location.reload();
114        }
115      };
116      xhr.send('action=eliminar&id_cancion=' + id);
117    }
118  }
119 </script>
120 HTML

```

6. update_song.pl

Este script Perl actualiza el nombre de una canción en una base de datos MySQL. Obtiene el `id_cancion` y el `nombre_cancion` desde los parámetros de la solicitud, y si ambos están presentes, ejecuta una consulta UPDATE para modificar el nombre de la canción en la base de datos. Finalmente, responde al usuario con un mensaje indicando que la canción ha sido actualizada exitosamente.

```

1 #!/usr/bin/perl
2 use strict;
3 use warnings;
4 use CGI qw(:standard);
5 use DBI;
6 use utf8;
7 binmode(STDOUT, ":utf8");
8
9 my $cgi = CGI->new;
10 my $dbh = DBI->connect("DBI:mysql:database=mi_base_de_datos;host=localhost", "mi_usuario", "mi_contraseña");
11
12 my $id_cancion = $cgi->param('id_cancion');
13 my $nombre_cancion = $cgi->param('nombre_cancion');
14
15 if ($id_cancion && $nombre_cancion) {
16     my $sth = $dbh->prepare("UPDATE canciones SET nombre_cancion = ? WHERE id = ?");
17     $sth->execute($nombre_cancion, $id_cancion);
18 }
19
20 print $cgi->header('text/html; charset=UTF-8');
21 print "Canción actualizada con éxito.";
22

```

7. delete_song.pl

Este script Perl elimina una canción de una base de datos MySQL. Obtiene el id_cancion desde los parámetros de la solicitud, y si el ID está presente, ejecuta una consulta DELETE para eliminar la canción correspondiente en la base de datos. Luego, responde al usuario confirmando que la canción ha sido eliminada exitosamente.

```

1 #!/usr/bin/perl
2 use strict;
3 use warnings;
4 use CGI qw(:standard);
5 use DBI;
6 use utf8;
7 binmode(STDOUT, ":utf8");
8
9 my $cgi = CGI->new;
10 my $dbh = DBI->connect("DBI:mysql:database=mi_base_de_datos;host=localhost", "mi_usuario", "mi_contraseña");
11
12 my $id_cancion = $cgi->param('id_cancion');
13
14 if ($id_cancion) {
15     my $sth = $dbh->prepare("DELETE FROM canciones WHERE id = ?");
16     $sth->execute($id_cancion);
17 }
18
19 print $cgi->header('text/html; charset=UTF-8');
20 print "Canción eliminada con éxito.";
21
22

```

8. config.pl

Este código es un script en Perl que maneja un formulario web para configurar la conversión de videos. Utiliza CGI para crear la interfaz HTML, donde se pueden seleccionar opciones de conversión de formato, resolución, proporción de imagen y propiedades de audio. Los parámetros del formulario permiten elegir el formato de entrada y salida (como MP4, MP3, AVI), la resolución del video, la proporción de imagen, y la codificación de audio. Cuando el usuario selecciona un video y hace clic en "Convertir", el script ejecuta un comando ffmpeg para realizar la conversión del archivo

de video según las opciones seleccionadas. Los resultados se muestran en la página, indicando si la conversión fue exitosa o si hubo errores. Además, genera una lista de videos disponibles en el servidor, permitiendo que el usuario seleccione el video que desea convertir.

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4  use CGI qw(:standard);
5  use File::Spec;
6  use Encode;
7  use utf8;
8
9  my $cgi = CGI->new;
10
11 print $cgi->header('text/html; charset=UTF-8');
12 print $cgi->start_html('Configuración de mis descargas');
13
14 # Parámetros del formulario
15 my $formato_inicial = $cgi->param('selectmenuinicial') || 'mp4';
16 my $formato_final = $cgi->param('selectmenufinal') || 'mp4';
17 my $resolucion = $cgi->param('selectedresolucion');
18 my $proporcion_imagen = $cgi->param('selectedproporcionimagen');
19 my $audio_inicial = $cgi->param('selectedaudioinicial');
20 my $frecuencia = $cgi->param('selectedfrecuencia');
21 my $canales = $cgi->param('selectedcanales');
22 my $accion = $cgi->param('accion');
23
24 if ($frecuencia eq '44.1') {
25 |   $frecuencia = 44100;
26 } elsif ($frecuencia eq '48') {
27 |   $frecuencia = 48000;
28 } elsif ($frecuencia eq '96') {
29 |   $frecuencia = 96000;
30 } elsif ($frecuencia eq '192') {
31 |   $frecuencia = 192000;
32 }
33
34 if ($canales eq 'mono') {
35 |   $canales = 1;
36 } elsif ($canales eq 'stereo') {
37 |   $canales = 2;
38 } elsif ($canales eq '5.1') {
39 |   $canales = 6;
40 } elsif ($canales eq '7.1') {
41 |   $canales = 8;
42 }
43
44 if ($resolucion eq '480p') {
45 |   $resolucion = '640x480';
46 } elsif ($resolucion eq '720p') {
47 |   $resolucion = '1280x720';
48 } elsif ($resolucion eq '1080p') {
```

```
102     my $command;
103     if ($formato_final eq 'mp3') {
104         $command = "ffmpeg -i $input_file -vn -ar 44100 -ac $canales -b:a 192k $output_file";
105     } else {
106         $command = "ffmpeg -i $input_file -s $resolucion -aspect $proporcion_imagen -ar $frecuen
107     }
108     # Ejecutar el comando
109     my $output = `"$command 2>&1`;
110     if ($?) {
111         print "<p>Hubo un error al intentar convertir el video:</p><pre>$output</pre>";
112     } else {
113         print "<p>El video ha sido convertido con éxito. :)</p>";
114     }
115     if (-s '/tmp/ffmpeg_error.log') {
116         open my $fh, '<', '/tmp/ffmpeg_error.log';
117         my @error_log = <$fh>;
118         close $fh;
119         print "<p>Error de ffmpeg:</p><pre>@error_log</pre>";
120     } else {
121         print "<p>El video ha sido convertido con éxito. Puedes descargarlo <a href='/descargas/
122         print '<video width="640" height="360" controls>
123             <source src="/descargas/' . $video_descargado . '.convertido.' . $formato_final . '" typ
124             Your browser does not support the video tag.
125             </video><br><br>';
126     }
127 } else {
128     print "<p>No se seleccionó un video para convertir.</p>";
129 }
130 } else {
131     print "<h1>LOADING...</h1>";
132 }
133
134 # Ruta del directorio de descargas
135 my $path = '/descargas';
136 opendir(my $dh, $path) or die "No se pudo abrir el directorio '$path': $!";
137 my @categorias = readdir($dh);
138 closedir($dh);
```

```

141 # Comenzamos el formulario
142 print <<HTML;
143 <head>
144 |   <link rel="stylesheet" href="../html/config.css">
145 </head>
146 <h1>Configuración</h1>
147 <form action="/cgi-bin/config.pl" method="GET">
148   <strong>Convertir   De  &nbsp;&nbsp;</strong>
149   <select id="menuconfiginicial" name="selectmenuinicial">
150     <option value="mp4" @{{[ $formato_inicial eq 'mp4' ? 'selected' : '' ]}}>MP4</option>
151     <option value="mp3" @{{[ $formato_inicial eq 'mp3' ? 'selected' : '' ]}}>MP3</option>
152     <option value="avi" @{{[ $formato_inicial eq 'avi' ? 'selected' : '' ]}}>AVI</option>
153   </select>
154
155   <strong>&nbsp;a&nbsp;</Strong>
156
157   <select id="menuconfigfinal" name="selectmenufinal">
158     <option value="mp4" @{{[ $formato_final eq 'mp4' ? 'selected' : '' ]}}>MP4</option>
159     <option value="mp3" @{{[ $formato_final eq 'mp3' ? 'selected' : '' ]}}>MP3</option>
160     <option value="avi" @{{[ $formato_final eq 'avi' ? 'selected' : '' ]}}>AVI</option>
161   </select><br><br>
162
163   <strong>Seleccione el video</strong><br>
164   <select id="video" name="video_descargado">
165 HTML
166
167 # Generar las opciones del <select> con los videos disponibles de todas las categorías
168 foreach my $categoria (@categorias) {
169   next if $categoria =~ /^\. '/';
170
171   # Ruta completa para la categoría
172   my $directorio_categoria = "$path/$categoria";
173
174   # Comprobar si es un directorio
175   if (-d $directorio_categoria) {
176     opendir(my $dh_categoria, $directorio_categoria) or die "No se pudo abrir el directorio '$directorio_categoria'";
177     my @videos = readdir($dh_categoria);
178     closedir($dh_categoria);
179
180     # Si hay videos en esta categoría, los agregamos al <select>
181     foreach my $video (@videos) {
182       next if $video =~ /^\. /; # Excluir directorios ocultos
183
184       # Mostrar el video en la lista
185       print "<option value='$categoria/$video'>$video</option>\n";
186     }

```

9. ver_canciones.pl

El script Perl genera una página web que muestra una lista de archivos de un directorio específico dentro de /descargas, cuyo nombre se pasa como parámetro en la URL. Si no se proporciona un nombre de directorio, se usa un valor por defecto. Muestra los archivos en formato de lista HTML, omitiendo los archivos ocultos, y ofrece un enlace para regresar a la página principal.

```

1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4  use CGI qw(:standard);
5  use File::Spec;
6  use utf8;
7
8  my $cgi = CGI->new;
9  my $directorio = $cgi->param('directorio') || 'Mis favoritos';
10 my $path = File::Spec->catfile('/descargas', $directorio);
11
12 print $cgi->header('text/html; charset=UTF-8');
13 print $cgi->start_html('Canciones Descargadas');
14
15 print "<h1>Canciones Descargadas en '$directorio'</h1>";
16
17 opendir(my $dh, $path) or die "No se pudo abrir el directorio '$path': $!";
18 my @files = readdir($dh);
19 closedir($dh);
20
21 print "<ul>";
22 foreach my $file (@files) {
23     next if $file =~ /\. /; # Omitir archivos ocultos
24     print "<li>$file</li>";
25 }
26 print "</ul>";
27
28 print "<a href='/cgi-bin/descargador.pl'>Regresar a la página principal</a>";
29
30 print $cgi->end_html;
31

```

- init.sql

Este conjunto de sentencias SQL crea una base de datos llamada `mi_base_de_datos` si no existe, luego crea un usuario `mi_usuario` con contraseña `mi_contraseña`, otorgándole todos los privilegios sobre la base de datos. Después, define dos tablas: una para almacenar información de usuarios (`usuarios`), que incluye campos como nombre, correo, contraseña y teléfono, y otra para gestionar canciones descargadas (`canciones`), con atributos como nombre, URL, formato y el directorio donde se guarda. Además, ambas tablas tienen campos para registrar la fecha de creación, con restricciones de integridad como la comprobación de edad mínima en la tabla de usuarios.

```

1  -- Crear la base de datos
2  CREATE DATABASE IF NOT EXISTS mi_base_de_datos;
3
4  CREATE USER IF NOT EXISTS 'mi_usuario'@'localhost' IDENTIFIED BY 'mi_contraseña';
5  GRANT ALL PRIVILEGES ON mi_base_de_datos.* TO 'mi_usuario'@'localhost';
6  FLUSH PRIVILEGES;
7
8  -- Usar la base de datos recién creada
9  USE mi_base_de_datos;
10
11 -- Crear la tabla de usuarios
12 CREATE TABLE IF NOT EXISTS usuarios (
13     id INT AUTO_INCREMENT PRIMARY KEY,
14     nombre VARCHAR(50),
15     apellido VARCHAR(50),
16     fecha_nacimiento DATE,
17     usuario VARCHAR(100) UNIQUE NOT NULL,
18     correo VARCHAR(100) UNIQUE NOT NULL,
19     password VARCHAR(255) NOT NULL,
20     edad INT CHECK (edad >= 5),
21     genero VARCHAR(1) NOT NULL,
22     telefono VARCHAR(9) NOT NULL,
23     fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
24 );
25
26 -- Crear la tabla de canciones descargadas
27 CREATE TABLE IF NOT EXISTS canciones (
28     id INT AUTO_INCREMENT PRIMARY KEY,
29     nombre_cancion VARCHAR(255) NOT NULL,
30     url VARCHAR(255) NOT NULL,
31     formato VARCHAR(10) NOT NULL,
32     fecha_descarga TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
33     directorio VARCHAR(255) NOT NULL
34 );
35

```

- Comandos de construcción del dockerfile

docker build -t iproyecto .

docker run -d -p 8092:80 -p 3307:3306 --name ciproyecto iproyecto

VIII. CONCLUSIONES/RECOMENDACIONES

Conclusiones

- Aplicación completa y fácil de usar: La aplicación permite a los usuarios registrarse, iniciar sesión y descargar videos de YouTube de forma sencilla. Además, se pueden convertir los videos a diferentes formatos y gestionar los archivos descargados, todo desde una sola plataforma.
- Interactividad y personalización: Gracias al sistema CRUD, los usuarios pueden cambiar el nombre de las canciones o eliminarlas, lo que les da más control sobre sus descargas. Además, la opción de cambiar las propiedades de los videos (como la resolución y la calidad del audio) es una gran ventaja, ya que hace que los archivos sean aún más personalizados.
- Entretenimiento durante la espera: Una de las cosas que hace que la aplicación sea más divertida es que agregamos juegos básicos en JavaScript. Esto ayuda a que el tiempo de espera para descargar o convertir videos no sea aburrido, lo que mejora la experiencia del usuario.

- Facilidad de uso: La aplicación está diseñada para que los usuarios puedan navegar de forma fácil y rápida. No importa si alguien es nuevo en esto o ya tiene algo de experiencia con las descargas de videos, la interfaz es intuitiva y facilita todo el proceso.
- Tecnologías bien utilizadas: Usar herramientas como CGI, AJAX, FFmpeg y yt-dlp fue clave para que la aplicación funcione de manera eficiente y rápida. Cada tecnología tiene un papel importante y contribuye a que el proceso de descarga y conversión sea fluido.

Recomendaciones

- Mejorar el rendimiento: Aunque la aplicación funciona bien con pocos usuarios, si más personas empiezan a usarla al mismo tiempo, podría volverse más lenta. Sería bueno optimizarla para manejar más tráfico sin que se cuelgue o se demore mucho.
- Más opciones de personalización: Aunque ya hay algunas opciones para cambiar las propiedades de los videos, podría ser útil agregar más características, como elegir diferentes calidades de video o añadir subtítulos si están disponibles.
- Hacer el diseño más atractivo: Aunque la aplicación funciona bien, su diseño podría ser más bonito y moderno. Mejorar la apariencia de los botones, las imágenes y las opciones haría que la aplicación sea más atractiva para los usuarios.
- Agregar más fuentes: Actualmente, la aplicación solo permite descargar de YouTube, pero sería genial agregar otras plataformas, como Vimeo, para que los usuarios tengan más opciones.
- Mejorar la gestión de sesiones: El uso de CGI::Session está bien para la gestión de usuarios, pero en el futuro podríamos pensar en algo más avanzado, como tokens JWT, para manejar las sesiones de una manera más segura y eficiente, especialmente si la aplicación crece y tiene más usuarios.

RÚBRICA DE CALIFICACIÓN

ÍTEM	DESCRIPCIÓN	EXCELENTE	PROCESO	DEFICIENTE	AUTO-EVALUACIÓN
Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	4	2	1	3
Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente hasta llegar al código final del requerimiento del laboratorio.	4	2	1	4
Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación). Si no se le entregó pregunta, usted	4	2	1	3

	recopile información relevante para el laboratorio desde diferentes medios, referenciandola correctamente (máximo 2 caras).				
Ortografía	El documento no muestra errores ortográficos.	4	2	1	4
Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	2	1	3
	CALIFICACIÓN	20	10	5	16

REFERENCIAS Y BIBLIOGRAFÍA

CICEI. (n.d.). *Tutorial de Perl: Capítulo 6 - Operadores aritméticos.* Retrieved from <https://www2.iib.uam.es/bioinfo/curso/perl/tutoriales/cicei/cap6.htm>

Perl 5 Porters. (n.d.). *perlop - Perl operators.* Retrieved from <https://perldoc.perl.org/perlop.html>

<https://docs.google.com/document/d/1v5hXZU88QdhELTcDjdEpnfZCaej-FnZM/edit#heading=h.gidgxs>

<https://github.com/rescobelodoulasalle/docker/tree/main>

https://github.com/KarlaBedregal/calculadora_CGI/commits/main/