Universidad de la Cuenca del Plata
Autorización Definitiva. Decreto Poder Ejecutivo Nacional N°091/2006

Actividad Áulica 25 - Desafío Halloween

CARRERA: Ingeniería en Sistemas de la Información.

MATERIA: Paradigmas y Lenguajes de Programación III.

COMISIÓN: “U” (única).

PROFESOR: Mgter. Encina Agustín.

ESTUDIANTE: Vera, Romina.

FECHA: 06-11-2025.

Objetivo del desafío

Diseñar e implementar una aplicación web mínima pero completa (PHP + MySQL) que permita registrar disfraces e implementar una votación autenticada, garantizando un solo voto por usuario y por disfraz. La solución debe incluir registro e inicio de sesión, panel de administración para gestionar (CRUD) los disfraces e interfaz temática de Halloween, aplicando buenas prácticas de seguridad (hash de contraseñas, CSRF, sanitización de entradas) y los métodos mysqli_* solicitados.

Paso 1 — Configuración de la base de datos

Objetivo del paso. Crear la base halloween y las tablas usuarios, disfraces y votos, definiendo claves primarias, unicidad y relaciones que garanticen un solo voto por usuario y por disfraz y permitan administrar disfraces e imágenes.

```
CREATE DATABASE IF NOT EXISTS halloween
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_general_ci;
USE halloween;

DROP TABLE IF EXISTS usuarios;
CREATE TABLE usuarios (
  id INT(11) NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL UNIQUE,
  clave TEXT NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB;

DROP TABLE IF EXISTS disfraces;
CREATE TABLE disfraces (
  id INT(11) NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL,
  descripcion TEXT NOT NULL,
  votos INT(11) NOT NULL DEFAULT 0,
  foto VARCHAR(20) NOT NULL,
  foto_blob BLOB NOT NULL,
  eliminado INT(11) NOT NULL DEFAULT 0,
  PRIMARY KEY (id)
) ENGINE=InnoDB;

DROP TABLE IF EXISTS votos;
CREATE TABLE votos (
  id INT(11) NOT NULL AUTO_INCREMENT,
  id_usuario INT(11) NOT NULL,
  id_disfraz INT(11) NOT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY uq_voto (id_usuario, id_disfraz),
  CONSTRAINT fk_voto_usuario FOREIGN KEY (id_usuario) REFERENCES
usuarios(id) ON DELETE CASCADE,
```

```
  CONSTRAINT fk_voto_disfraz FOREIGN KEY (id_disfraz) REFERENCES
disfraces(id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- Semillas opcionales (disfraces de ejemplo)
INSERT INTO disfraces (nombre, descripcion, votos, foto, foto_blob, eliminado) VALUES
('La Calabaza Misteriosa', 'Disfraz clásico con antifaz y capa naranja.', 0, 'imagen1.jpg', '',
0),
('Vampiro Elegante',       'Smokin, capa y lentes rojos. Muy retro.',    0, 'imagen2.jpg', '', 0),
('Bruja Urbana',           'Sombrero y escoba, estilo urbano.',          0, 'imagen3.jpg', '', 0);
```

- Motor y codificación. Se usa InnoDB (transacciones e integridad referencial) y utf8mb4 (soporta acentos y emojis).
- usuarios.
    - id PK autoincremental.
    - nombre UNIQUE: evita duplicar usuarios.
    - clave TEXT: almacena el hash de contraseña generado en la app.
- disfraces.
    - votos comienza en 0 y se incrementa al votar.
    - Estrategia de imágenes mixta: foto (archivo en /fotos) + foto_blob (binario). Si no hay archivo, puede renderizarse desde el BLOB.
    - eliminado es borrado lógico (permite ocultar sin borrar).
- votos.
    - UNIQUE (id_usuario, id_disfraz) aplica la regla de negocio: un solo voto por usuario/disfraz (capa de integridad en la BD).

- FKs con ON DELETE CASCADE: al eliminar un usuario o un disfraz, sus votos se borran automáticamente, evitando registros huérfanos.

Resultado del paso

Quedó creada la base halloween con sus tres tablas relacionadas y reglas de integridad que soportan el flujo de la aplicación: registro/login, catálogo de disfraces, y votación sin duplicados con limpieza automática de votos ante eliminaciones.

Paso 2 — Desarrollo de la aplicación web

Objetivo del paso. Implementar la aplicación PHP + MySQL que permite listar disfraces, registrar/iniciar sesión, votar (solo usuarios autenticados, sin votos duplicados) y administrar disfraces (CRUD) con temática visual de Halloween.

Estructura de archivos

```
/ (raíz del proyecto)
├── index.php          # Listado + modal + botón Votar
├── registro.php       # Alta de usuarios
├── login.php          # Inicio de sesión
├── logout.php         # Cierre de sesión
├── votar.php          # Procesa la votación
├── admin.php          # CRUD de disfraces (solo admin)
├── config/
│   └── db.php         # mysqli_connect, charset
├── lib/
│   └── auth.php       # Sesiones, CSRF, helpers, require_login/admin
├── css/
│   └── halloween.css  # Estilos y tema Halloween
└── fotos/             # Imágenes subidas (accesible por web)
```

Páginas y responsabilidades

- index.php (pública)
    - Muestra grilla de disfraces no eliminados (eliminado=0) con nombre, descripción y votos.
    - Calcula porcentaje sobre el total con number_format($porc, 2, ',', '.').
    - Determina la imagen a mostrar:
        - Si existe archivo fotos/<nombre>, lo usa (file_exists()).
        - Si no, intenta BLOB (consulta foto_blob).
        - Si nada, placeholder.
    - Para cada disfraz, si el usuario está logueado, verifica si ya votó (consulta a votos) para deshabilitar el botón.
    - Abre un modal con detalle y acción de Votar (POST a votar.php con CSRF).
- registro.php (pública)
    - Formulario POST con csrf_field().
    - Valida campos requeridos; verifica duplicado de nombre.
    - Genera hash de contraseña (password_hash()), lo escapa (mysqli_real_escape_string()), inserta y logea automáticamente (set de $_SESSION).
    - Mensajes de error/éxito visibles en la UI.
- login.php (pública)

- Formulario POST con CSRF.
- Busca por nombre (escapado), valida hash con password_verify().
- Si OK, setea $_SESSION['user_id'] y $_SESSION['user_nombre']; redirige a index.php.

- logout.php
  - Limpia sesión y redirige a inicio.
  - votar.php (privada)
  - Requiere sesión: require_login().
  - Valida CSRF y que id_disfraz sea entero positivo.
  - Comprueba si ya existe voto (id_usuario, id_disfraz); si existe, no duplica.
  - Inserta en votos y hace UPDATE disfraces SET votos = votos + 1.
  - Redirige a index.php.
- admin.php (privada)
  - Solo admin: require_admin() (usuario con nombre admin).
  - Crear/Editar disfraz:
    - Valida nombre/descripcion.
    - Sube imagen: usa $_FILES['foto']['name'], explode('.'), end() para extensión, verifica subida con is_uploaded_file(), y guarda a /fotos con nombre único basado en time() + extensión (copy()).
    - Opcionalmente, lee el binario y lo guarda en foto_blob.

- Si reemplaza imagen, borra la anterior con unlink().

- Listado con columnas dinámicas (mysqli_num_fields(), mysqli_fetch_field_direct()).
- Eliminar: permite borrado físico del registro (y opcionalmente del archivo) o marcado lógico con eliminado.

Seguridad aplicada

- Sesiones centralizadas en lib/auth.php.
- CSRF: token por sesión, campo oculto en formularios y verificación en cada POST.
- Autenticación: password_hash() + password_verify().
- Validación de entrada:
  - Cast a int en IDs ((int)).
  - Escapes: mysqli_real_escape_string() para strings antes de armar SQL.
- Autorización:
  - require_login() para votar.
  - require_admin() para CRUD.

La unicidad de voto se protege dos veces: (a) chequeo en PHP previo a insertar y (b) UNIQUE (id_usuario, id_disfraz) en la BD (evita condiciones de carrera).

Uso de funciones

- mysqli_connect() → config/db.php
- mysqli_query() → consultas en todas las páginas
- mysqli_num_rows() → verificar usuario/voto existente
- mysqli_insert_id() → obtener ID recién creado (usuarios, disfraces)
- mysqli_num_fields() → render de columnas en admin.php
- mysqli_real_escape_string() → sanitizar entradas de texto
- $_FILES['foto']['name'] → nombre original de imagen en admin.php
- explode(".",$archivo) y end($extension) → extensión de archivo
- is_uploaded_file($_FILES['foto']['tmp_name']) → validar subida legítima
- time() → nombre único del archivo
- copy($_FILES['foto']['tmp_name'], "fotos/".$qu."."end($extension)) → mover imagen
- mysqli_error($con) → mostrar error controlado en admin
- unlink('fotos/' . $_POST['foto_actual']) → limpiar archivo viejo
- isset($_POST['nombre']) → validar campos en formularios
- file_exists("fotos/".$r['foto']) → decidir si se muestra archivo o BLOB
- number_format($r['Precio'],2,',','.') → adaptado: formateo de porcentaje en index.php

Flujo principal de usuario

- Registrarse (o iniciar sesión).
- Ver la lista de disfraces en index.php.
- Abrir el modal y votar (si está logueado).
- La app registra el voto y actualiza el contador del disfraz.
- El admin puede crear/editar/eliminar disfraces y manejar imágenes desde admin.php.

Resultado del paso

Aplicación operativa: catálogo visible, registro/login, votación controlada por usuario, y panel admin con subida de imágenes; todo con CSRF, hashes de contraseña y sanitización básica, cumpliendo la consigna.

Paso 3 — Personalización

Objetivo del paso. Darle identidad visual de Halloween a la app y permitir que los usuarios carguen imágenes de sus disfraces, manteniendo una experiencia responsive, accesible y segura.

Estilos y tema Halloween

- Paleta: negros y morados de fondo, acentos naranja calabaza (#ff6b35) y violeta para contraste.
- Componentes:
    - Header semitransparente con blur y borde luminoso.
    - Tarjetas (disfraces) con hover suave, micro-animaciones y “badge” de porcentaje.
    - Modal para ver detalle y votar.
- Formularios con inputs enfatizados y foco visible.
- Tipografía: sans-serif legible; contraste AA/AAA en textos clave.
- Animaciones: transiciones de opacidad/transform, “pulso” sutil en títulos, sin bloquear la interacción.

Resultado: archivo css/halloween.css con gradientes, sombras, animaciones, layout en grid y media queries para móviles.

Carga y manejo de imágenes

- Estrategia mixta (dos vías):
    1. Archivo en /fotos (campo disfraces.foto).
    2. BLOB (campo disfraces.foto_blob) opcional para portabilidad.
- Prioridad de renderizado en index.php:
    1. Si file_exists("fotos/".$r['foto']) → mostrar archivo.

2. Si no hay archivo pero foto_blob tiene datos → mostrar desde BLOB (base64).
3. Si nada → placeholder.

Validaciones y seguridad en la subida

- Servidor (admin.php):
  - Validar que el archivo provenga de una subida real: is_uploaded_file($_FILES['foto']['tmp_name']).
  - Derivar extensión segura:
  - $_FILES['foto']['name'] → explode('.', $archivo) → strtolower(end($parts)).
  - Generar nombre único con time() y copiar:
  - copy($_FILES['foto']['tmp_name'], "fotos/{$qu}.{$ext}").
  - Guardar el nombre en disfraces.foto; y opcionalmente, cargar el binario en foto_blob.
  - Limpieza al actualizar: si había imagen previa, unlink('fotos/' . $_POST['foto_actual']).

Accesibilidad y responsive

- Accesibilidad: etiquetas <label> asociadas, foco visible, colores con contraste, textos alternativos en <img alt="">.
- Responsive: grilla auto-fill; modal a una columna en ≤768 px; controles táctiles con targets ≥44 px.

Mapa de funciones usadas (lo exigido por la cátedra)

- $_FILES['foto']['name'] → nombre original.
- explode(".",$archivo) / end($extension) → obtener extensión.
- is_uploaded_file($_FILES['foto']['tmp_name']) → validar subida.
- time() → nombre único del archivo.
- copy($_FILES['foto']['tmp_name'], "fotos/".$qu."."end($extension)) → mover al directorio público.
- unlink('fotos/' . $_POST['foto_actual']) → borrar archivo anterior.
- file_exists("fotos/".$r['foto']) → decidir si mostrar archivo o BLOB.

Resultado del paso

La aplicación presenta una identidad Halloween consistente, es usable en móvil, y permite cargar/actualizar imágenes de forma segura y ordenada (archivo + BLOB), cumpliendo los ítems de personalización de la consigna.