



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO MATERIAL DE APOYO – ACTIVIDADES

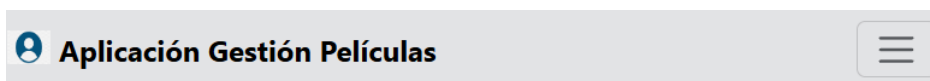
MATERIAL APOYO DESARROLLO DE APLICACIONES
EN PYTHON CON FLASK-MONGOALCHEMY

Creación del Front-End desde Python.

Pasos:











1. Partimos de que ya tenemos en el **backend** funcionando las siguientes rutas:
 - a. **GET /genero/**: lista todos los géneros de las películas
 - b. **POST /genero/**: agregar un genero.
 - c. **PUT /genero/**: actualizar un genero
 - d. **DELETE /genero/**: eliminar un genero
 - e. **GET /pelicula/**: lista todos películas
 - f. **POST /pelicula/**: agregar una película.
 - g. **PUT /película/**: actualizar una película
 - h. **DELETE /pelicula/**: eliminar una película

Funcionalidad Editar una Película: Partimos de la siguiente vista:



LISTADO DE PELÍCULAS

Agregar

Código	Título	Duración	Protagonista	Género	Acción
100	SuperMen	120	Camilinderere	Infantil	 
888	Rambo 5	130	Arnold	Comedia	 
234234	xxxxx	169	ddasdasd	Comedia	 
9999999	xxxxx	169	ddasdasd	Comedia	 
1000	La sombra	120	Pedro Infante	Terror	 

Tecnólogo en Análisis y Desarrollo de Software
Derechos Reservados @copyright



Al dar clic en el icono amarillo de cada película podemos ir a editar la película seleccionada. En el html el icono queda con una etiqueta **a** donde apuntamos a una **ruta** que se le agrega el id de la película. `/vistaEditarPelicula/{{p.id}}/`

```
<a href="/vistaEditarPelicula/{{p.id}}/">
  <i class="fa fa-edit text-warning" title="Editar"></i>
</a>
```

Código de la ruta: `/vistaEditarPelicula/{{p.id}}/`

```
Tabnine | Edit | Test | Explain | Document
@app.route("/vistaEditarPelicula/<string:id>/", methods=['GET'])
def mostrarVistaEditarPelicula(id):
    if ("user" in session):
        pelicula = Pelicula.objects(id=ObjectId(id)).first()
        generos = Genero.objects()
        return render_template("frmEditarPelicula.html",
                               pelicula=pelicula, generos=generos)
    else:
        mensaje="Debe primero ingresar con credenciales válidas"
        return render_template("frmIniciarSesion.html", mensaje=mensaje)
```

Para poder utilizar **ObjectId** debemos hacer la siguiente importación:

```
from bson.objectid import ObjectId
```



La ruta anterior renderiza el html llamado **frmEditarPelicula.html** cuya vista es la siguiente:

EDITAR PELÍCULA

Código:

100

✓

Título:

SuperMen

✓

Protagonista:

Camilinderere

✓

Duracion(minutos):

120

✓

Género:

Infantil

✓ ▾

Resumen:

yyyyyyyyyyyyyy

✓

Editar

Cancelar

Tecnólogo en Análisis y Desarrollo de Software
Derechos Reservados @copyright

El formulario para editar la película recibe los objetos **géneros** y la **película** consultada, los cuales son utilizados en el formulario para colocar los valores de la película que se quiere editar. Lo anterior nos indica que cada campo se le asigna al atributo value el valor correspondiente del objeto película.



Código del html para editar la película:

```
5  {% block contenido %}
6      <script src="{ url_for('static', filename='js/app.js') }}"></script>
7      <div class="w-25" style="margin: 0 auto">
8          <form id="frmPelicula" class="was-validated" enctype="multipart/form-data">
9              <div>
10                 <h3 class="text-center fw-bold bg-dark text-white" style="font-size:21vw">EDITAR PELÍCULA</h3>
11             </div>
12             <div class="mb-3">
13                 <label class="fw-bold" for="txtCodigo">Código:</label>
14                 <input type="number" name="txtCodigo" id="txtCodigo" class="form-control"
15                     value="{{ pelicula.codigo }}" required>
16             </div>
17             <div class="mb-3">
18                 <label class="fw-bold" for="txtNombre">Título:</label>
19                 <input type="text" name="txtTitulo" id="txtTitulo" class="form-control"
20                     value="{{ pelicula.titulo }}" required>
21             </div>
22             <div class="mb-3">
23                 <label class="fw-bold" for="txtProtagonista">Protagonista:</label>
24                 <input type="text" name="txtProtagonista" id="txtProtagonista" class="form-control"
25                     value="{{ pelicula.protagonista }}" required>
26             </div>
27             <div class="mb-3">
28                 <label class="fw-bold" for="txtDuracion">Duracion(minutos):</label>
29                 <input type="number" name="txtDuracion" id="txtDuracion" min="30" max="200" class="form-control"
30                     value="{{ pelicula.duracion }}" required>
31             </div>
32
33             <div class="mb-3">
34                 <label class="fw-bold" for="cbGenero">Género:</label>
35                 <select name="cbGenero" id="cbGenero" class="form-select" value="{{ pelicula.genero.id }}" required>
36                     <option value="" selected>Selecione</option>
37                     {%for g in generos %}
38                         <option value="{{ g.id }}">{{ g.nombre }}</option>
39                     {%endfor%}
40                 </select>
41             </div>
42             <div class="mb-3">
43                 <label class="fw-bold" for="txtresumen">Resumen:</label>
44                 <textarea name="txtResumen" id="txtResumen" cols="30" rows="3" class="form-control"
45                     required>{{ pelicula.resumen }}</textarea>
46             </div>
47             <div>
48                 <button type="button" class="btn btn-dark" onclick="editarPelicula('{{ pelicula.id }}')">Editar</button>
49                 <a href="/peliculas/"><button type="button" class="btn btn-secondary">Cancelar</button></a>
50             </div>
51         </form>
52         <script>
53             document.getElementById('cbGenero').value = '{{ pelicula.genero.id }}'
54         </script>
55     </div>
56 {% endblock %}
```

El formulario en la parte inferior tiene un **botón** que tiene el texto **Editar**. Al dar clic lo envía a una función de javascript para que realiza la petición al servidor para la actualización de la película.



Código html del botón:

```
<button type="button" class="btn btn-dark"
onclick="editarPelicula('{pelicula.id}')">Editar</button>
```

Código de la función editarPelicula(id) en Javascript:

```
191  /**
192   * Función que se encarga de hacer la
193   * petición al servidor para actualizar
194   * una película de acuerdo con su id
195   * @param {*} id
196   */
197  function editarPelicula(id){
198      if(validarPelicula()){
199          const pelicula={
200              id: id,
201              codigo: document.getElementById('txtCodigo').value,
202              titulo: document.getElementById('txtTitulo').value,
203              protagonista: document.getElementById('txtProtagonista').value,
204              duracion: document.getElementById('txtDuracion').value,
205              resumen: document.getElementById('txtResumen').value,
206              genero: document.getElementById('cbGenero').value,
207              foto: ''
208          }
209          const url= "/pelicula/"
210          fetch(url, {
211              method: "PUT",
212              body: JSON.stringify(pelicula),
213              headers: {
214                  "Content-Type": "application/json",
215              }
216          })
217          .then(respuesta => respuesta.json())
218          .then(resultado => {
219              if (resultado.estado){
220                  location.href="/películas/"
221              }else{
222                  swal.fire("Edit Pelicula",resultado.mensaje,"warning")
223              }
224          })
225          .catch(error => {
226              console.error(error)
227          })
228      }else{
229          swal.fire("Edit Pelicula",mensaje,"warning")
230      }
231  }
```

La función anterior llama a una función llamada validarPelicula() que lo que hace es verificar que no se quede ningún campo vacío en el formulario.



Código Javascript de la función validarPelicula()

```
7 function validarPelicula(){
8     if (document.getElementById("txtCodigo").value==""){
9         mensaje="Debe ingresar código de la Película"
10        return false;
11    }else if(document.getElementById("txtTitulo").value==""){
12        mensaje="Debe ingresar Título de la Película"
13        return false;
14    }else if(document.getElementById("txtProtagonista").value==""){
15        mensaje="Debe ingresar el Protagonista de la Película"
16        return false;
17    }else if(document.getElementById('txtDuracion').value==""){
18        mensaje="Debe ingresar la duración de la Película"
19        return false;
20    }else if(document.getElementById('cbGenero').value==""){
21        mensaje="Debe seleccionar el género de la Película"
22        return false;
23    }else if(document.getElementById('txtResumen').value==""){
24        mensaje="Debe ingresar un pequeño resumen de la Película"
25        return false;
26    }else{
27        return true;
28    }
29 }
```

Si la película se actualiza bien y llega la variable estado como true, entonces lo redirecciona a la ruta donde se listan las películas, de lo contrario muestra un mensaje indicándonos el problema.

Código Python que responde a la petición PUT para actualizar una película:

```
41
42 @app.route("/pelicula/", methods=['PUT'])
43 def updatePelicula():
44     try:
45         mensaje=None
46         estado=False
47         if request.method=='PUT':
48             datos= request.get_json(force=True)
49             #obtener pelicula por id
50             pelicula = Pelicula.objects(id=datos['id']).first()
51             #actualizar sus atributos
52             pelicula.codigo = datos['codigo']
53             pelicula.titulo=datos['titulo']
54             pelicula.protagonista= datos['protagonista']
55             pelicula.resumen = datos['resumen']
56             pelicula.foto=datos['foto']
57             genero = Genero.objects(id=datos['genero']).first()
58             if genero is None:
59                 mensaje="No se actualizó el genero."
60             else:
61                 pelicula.genero=genero
62                 pelicula.save()
63                 mensaje = f"{mensaje} Película Actualizada"
64                 estado=True
65             else:
66                 mensaje="No permitido"
67     except Exception as error:
68         mensaje=str(error)
69         mensaje="No es posible actualizar ya existe película con ese código."
70
71     return {"estado":estado, "mensaje": mensaje}
72
```













Funcionalidad Eliminar Película: Partimos de la siguiente vista

 **Aplicación Gestión Películas**



LISTADO DE PELÍCULAS

Agregar

Código	Título	Duración	Protagonista	Género	Acción
100	SuperMen	120	Camilinderere	Infantil	 
888	Rambo 5	130	Arnold	Comedia	 
234234	xxxxx	169	ddasdasd	Comedia	 
9999999	xxxxx	169	ddasdasd	Comedia	 
1000	La sombra	120	Pedro Infante	Terror	 

Tecnólogo en Análisis y Desarrollo de Software
Derechos Reservados @copyright

Al dar clic en el **icono rojo** de cada película, lo direcciona a una función de javascript para que realice la petición al servidor para eliminar la película de acuerdo con el id que se le pasa como parámetro.

```
<i class="fa fa-trash text-danger" title="Eliminar"
onclick="deletePelicula('{{p.id}}')"></i>
```



Código javascript de la función deletePelicula(id)

```
233 /**
234  * Función que realiza la petición al servidor
235  * para eliminar una película de acuerdo con su id
236  * @param {*} id
237  */
238 function deletePelicula(id){
239     Swal.fire({
240         title: "¿Está usted seguro de querer eliminar la Película",
241         showDenyButton: true,
242         confirmButtonText: "SI",
243         denyButtonText: "NO"
244     }).then((result) => {
245         /* Read more about isConfirmed, isDenied below */
246         if (result.isConfirmed) {
247             const pelicula={
248                 id: id,
249             }
250             const url= "/pelicula/"
251             fetch(url, {
252                 method: "DELETE",
253                 body: JSON.stringify(pelicula),
254                 headers: {
255                     "Content-Type": "application/json",
256                 }
257             })
258             .then(respuesta => respuesta.json())
259             .then(resultado => {
260                 if (resultado.estado){
261                     location.href="/películas/"
262                 }else{
263                     swal.fire("Delete Pelicula",resultado.mensaje,"warning")
264                 }
265             })
266             .catch(error => {
267                 console.error(error)
268             })
269         }
270     });
271 }
```




Código Python que responde a la petición del cliente para eliminar una película

```
Tabnine | Edit | Test | Explain | Document
@app.route("/pelicula/", methods=['DELETE'])
def deletePelicula():
    try:
        mensaje=None
        estado=True
        if request.method=="DELETE":
            datos=request.get_json(force=True)
            pelicula=Pelicula.objects(id=datos['id']).first()
            if(pelicula is None):
                mensaje="No es posible eliminar pelicula con ese id"
            else:
                pelicula.delete()
                estado=True
                mensaje="Pelicula Eliminada correctamente"
        else:
            mensaje="No permitido"
    except Exception as error:
        mensaje=str(error)

    return {"estado": estado, "mensaje": mensaje}
```

Funcionalidad para ingresar a la aplicación con un usuario registrado en la base de datos.

Crear un modelo para la colección Usuario: Crear archivo usuario.py en la carpeta models

```
models > usuario.py > ...
1  from mongoengine import *
2
3  class Usuario(Document):
4      usuario = StringField(max_length=50, required=True, unique=True)
5      password = StringField(max_length=50,required=True)
6      nombres = StringField(max_length=50,required=True)
7      apellidos = StringField(max_length=50,required=True)
8      correo = EmailField(required=True, unique=True)
9
10
11  Tabnine | Edit | Test | Explain | Document
12  def __repr__(self):
13      return f"{self.nombres} {self.apellidos}"
```



Agregar manualmente un usuario a la base de datos para poder hacer la prueba y validación de ingreso.

Crear archivo **usuario.py** en la carpeta **routes** con las siguientes rutas:

```
routes > usuario.py > iniciarSesion2
1  from app import app,db
2  from flask import Flask, render_template, request, session
3  from flask_mongoengine import MongoEngine
4  from models.usuario import Usuario
5
6  Tabnine | Edit | Test | Explain | Document
7  @app.route("/")
8  def inicio():
9      return render_template("frmIniciarSesion.html")
10
11  Tabnine | Edit | Test | Explain | Document
12  @app.route("/iniciarSesion/", methods=['POST'])
13  def iniciarSesion():
14      mensaje = ""
15      if request.method=='POST':
16          try:
17              username=request.form['txtUser']
18              password=request.form['txtPassword']
19              usuario = Usuario.objects(usuario=username,password=password).first()
20              if usuario:
21                  session['user']=username
22                  return render_template("contenido.html")
23              else:
24                  mensaje="Credenciales no válidas"
25          except Exception as error:
26              mensaje=str(error)
27      return render_template("frmIniciarSesion.html", mensaje=mensaje)
28
```

Crear el formulario para iniciar la sesión

INICIAR SESIÓN

@

👁

Ingresar

Cancelar



Código del Formulario para iniciar sesión:

```
2  {% block contenido %}
3      <script src="https://www.google.com/recaptcha/api.js" async defer></script>
4      <div class="iniciarSesion w-25 bg-body-secondary mt-3" style="margin: 0 auto">
5          <h3 class="text-center fw-bold">INICIAR SESIÓN</h3>
6          <form action="/iniciarSesion/" method="post" class="w-75" method="post" style="margin: 0 auto">
7              <div class="input-group mb-3">
8                  <span class="input-group-text">@</span>
9                  <div class="form-floating">
10                     <input type="text" class="form-control" name="txtUser" id="txtUser"
11                     placeholder="Username">
12                     <label for="txtUser">Username</label>
13                 </div>
14             </div>
15             <div class="input-group mb-3">
16                 <span class="input-group-text"><i class="fa fa-eye"></i></span>
17                 <div class="form-floating">
18                     <input type="password" class="form-control" name="txtPassword" id="txtPassword"
19                     placeholder="Password">
20                     <label for="txtPassword">Contraseña:</label>
21                 </div>
22             </div>
23             <div class="mt-2 mb-2">
24                 <button class="btn btn-dark">Ingresar</button>
25                 <button class="btn btn-secondary">Cancelar</button>
26             </div>
27         </form>
28     </div>
29     {%if mensaje %}
30     <script>
31         Swal.fire("Inicio de Sesión", '{{mensaje}}', "info")
32     </script>
33     {% endif %}
34 {%endblock %}
```

Nota: Recuerden que para trabajar las sesiones debemos crear un **secretkey** al objeto **app**.

En el archivo app crear algo así: El valor del secret_key puede ser cualquiera.

```
app = Flask(__name__)
app.secret_key = "1234567890aeiou"
```



Validar para que no ingresen a rutas sin haber iniciado sesión.

Ejemplo ruta que lista las películas

```
Tabnine | Edit | Test | Explain | Document
@app.route("/peliculas/", methods=['GET'])
def listarPeliculas():
    if ("user" in session):
        peliculas = Pelicula.objects()
        generos = Genero.objects()
        print(generos)
        return render_template("listarPeliculas.html",
                                peliculas=peliculas,
                                generos=generos)
    else:
        mensaje="Debe primero ingresar con credenciales válidas"
        return render_template("frmIniciarSesion.html", mensaje=mensaje)
```

El código anterior primero valida si la variable **user** está creada como variable de sesión. Dicha variable debe ser creada cuando iniciamos sesión así:

```
Tabnine | Edit | Test | Explain | Document
@app.route("/iniciarSesion2/", methods=['POST'])
def iniciarSesion2():
    mensaje = ""
    if request.method=='POST':
        try:
            username=request.form['txtUser']
            password=request.form['txtPassword']
            usuario = Usuario.objects(usuario=username,password=password).first()
            if usuario:
                session['user']=username
                return render_template("contenido.html")
            else:
                mensaje="Credenciales no válidas"
        except Exception as error:
            mensaje=str(error)

    return render_template("frmIniciarSesion.html", mensaje=mensaje)
```

Para que se puedan crear las variables de sesión debemos importar session de la librería flask

```
from flask import Flask, render_template, request, session
```



Dicha importación la debemos hacer en todos los archivos donde hayan rutas que hagan referencia a las variables de sesión.

Cerrar la sesión: Utilizada para salir del sistema. Crearla en el archivo usuario.py de la carpeta routes.

```
Tabnine | Edit | Test | Explain | Document
@app.route("/salir/")
def exit():
    session.clear()
    mensaje="Ha cerrado la sesión de forma"
    return render_template("frmIniciarSesion.html",mensaje=mensaje)
```

Nota: Pendientes para la próxima sesión de clase

- Implementar el recaptcha
- Uso de variables de entorno
- Publicar la aplicación en un servidor internet

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	02-04-2025