ACTIVIDAD INFORME TECNICO SOBRE CREACION DE APLICACIÓN SOBRE GESTION DE PELICULAS EN LENGUAJE PYTHON, FLASK Y MONGO

PRESENTADO A LA INSTRUCTORA: Zulma

REALIZADO POR APRENDICES:

Constanza Quira

Camilo Hurtado

Wesly Benavides

Santiago Ceron

Lilian Anacona

SERVICIO NACIONAL DE APRENDIZAJE SENA

CENTRO DE TELEINFORMATICA Y PRODUCCION INDUSTRIA

TECNOLOGO EN ANALISIS Y DESARROLLO DE SOFTWARE

POPAYAN 11 DE ABRIL DE 2024

INTRODUCCION

El presente informe tiene como objetivo detallar el desarrollo del proyecto “Películas Web Mongo" una aplicación web construida utilizando Python, Flask y MongoDB. Este proyecto tiene como finalidad ofrecer una plataforma que permita gestionar películas a través de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) tanto para las películas como para sus géneros. Además, se incorporan funcionalidades de registro de usuarios e inicio de sesión, con la implementación de características adicionales como el uso de DataTables para la visualización interactiva de datos y la integración de reCAPTCHA para garantizar la seguridad en el registro de usuarios.

Este informe describe el proceso de desarrollo, las tecnologías utilizadas, las funcionalidades implementadas y los objetivos alcanzados durante la realización del proyecto. Además, se expone el proceso de implementación de la aplicación en el servicio de hosting Render y su posterior subida a un repositorio en GitHub.

Objetivo General:

- Desarrollar una aplicación web de gestión de películas utilizando Python, MongoDB y Flask, que permita realizar operaciones CRUD sobre películas y géneros, gestionar el registro e inicio de sesión de usuarios y emplear herramientas como DataTables y reCAPTCHA para mejorar la interacción y la seguridad de la plataforma.

Objetivos Específicos:

1. Aprender a crear aplicaciones web utilizando Python, Flask y MongoDB, implementando funciones de CRUD para películas y géneros.
2. Desarrollar un sistema de registro e inicio de sesión de usuarios que permita la autenticación y autorización dentro de la aplicación.
3. Integrar herramientas como DataTables para la visualización eficiente de datos y reCAPTCHA para la seguridad en el registro de usuarios.
4. Desplegar la aplicación en el servicio de Render para su publicación en la web y subir el código fuente a un repositorio en GitHub para su versión y colaboración futura.

Desarrollo del Informe (Descripción Técnica):

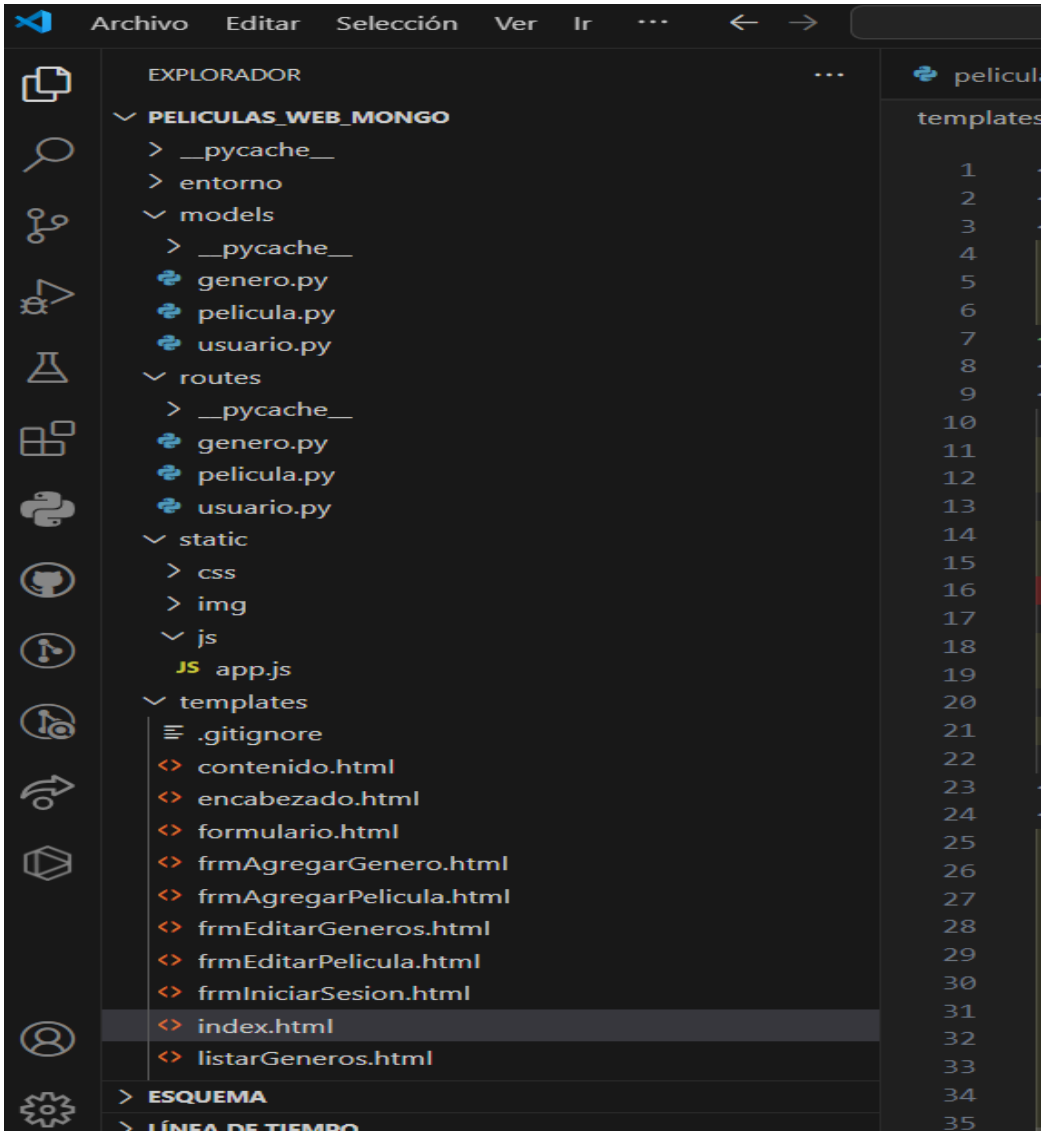
La actividad de desarrollo del proyecto “Películas Web Mongo" se llevó a cabo en el Centro de Teleinformática y Producción Industrial del SENA, ubicado en Popayán, Cauca, en el horario de 1 a 7 PM, entre el 26 de marzo y el 9 de abril. El proyecto estuvo dirigido por el instructor César Marino Cuéllar Chacón y estuvo destinado a los aprendices del 5° trimestre de la Tecnología en Análisis y Desarrollo de Software.

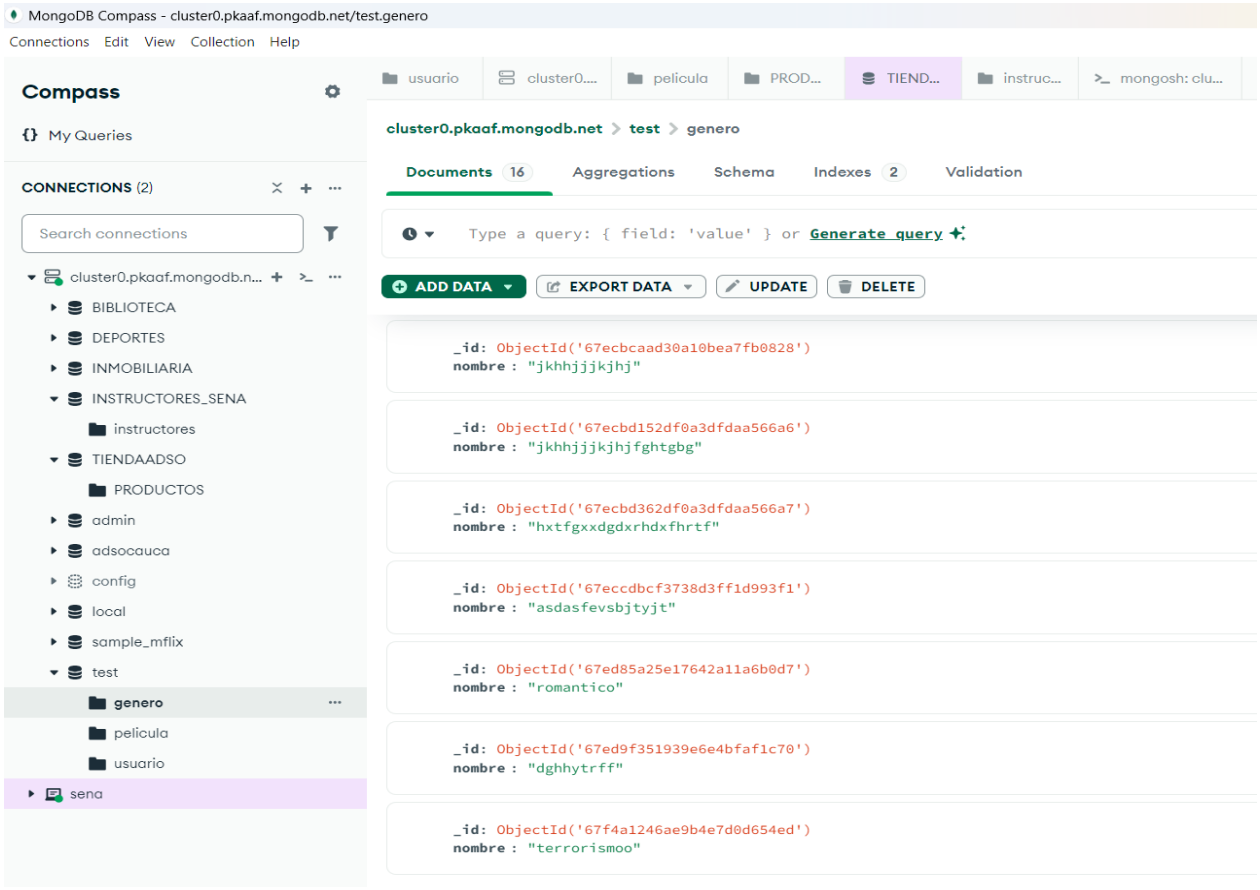
Durante las sesiones, se emplearon diversas herramientas y tecnologías para llevar a cabo el proyecto. Estas incluyeron Python, Flask y MongoDB para el desarrollo del backend, así como herramientas como DataTables para la visualización interactiva de los datos y reCAPTCHA para mejorar la seguridad del sistema.

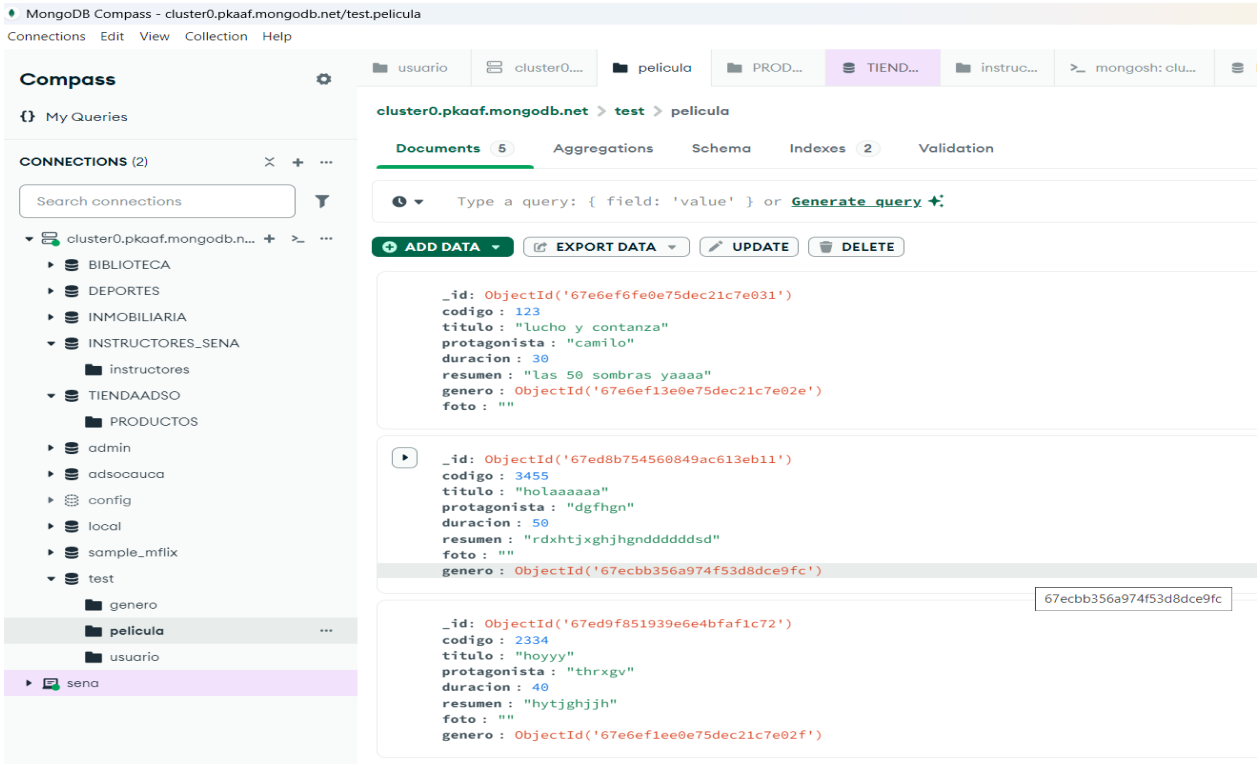
Pasos Realizados:

1. **Planificación y Diseño:**

- Se definieron los objetivos del proyecto, que incluyeron la creación de una aplicación que permitiera gestionar películas mediante operaciones CRUD y administrar usuarios con registro e inicio de sesión.
- Se estableció la estructura de la base de datos utilizando MongoDB, creando colecciones para las películas, géneros y usuarios.

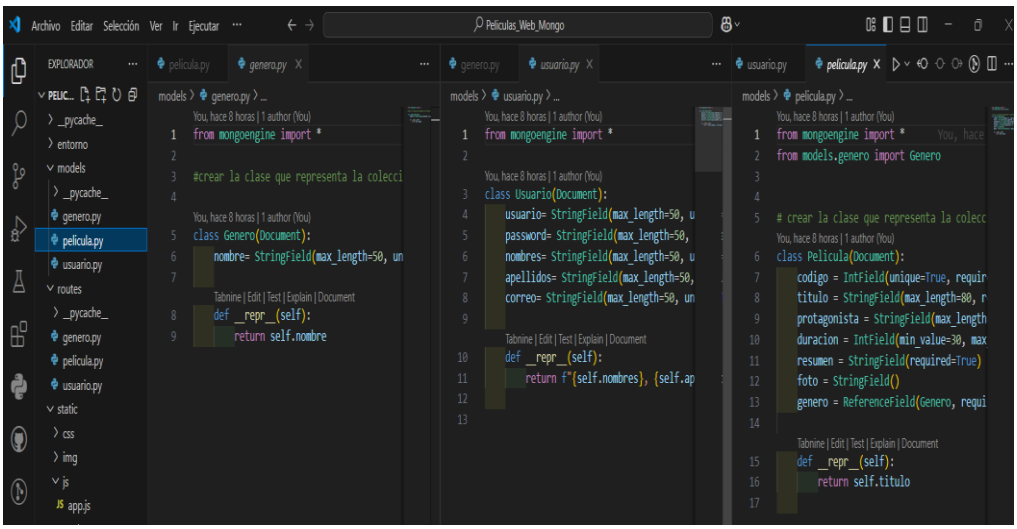



MongoDB Compass - cluster0.pkaaf.mongodb.net/test.genero
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
cluster0.pkaaf.mongodb.n...
BIBLIOTECA
DEPORTES
INMOBILIARIA
INSTRUCTORES_SENA
instructores
TIENDAADSO
PRODUCTOS
admin
adsocauca
config
local
sample_mflix
test
genero
pelicula
usuario
sena
usuario
cluster0....
pelicula
PROD...
TIEND...
instruc...
mongosh: clu...
cluster0.pkaaf.mongodb.net > test > genero
Documents 16
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE
_id: ObjectId('67ecbcaad30a10bea7fb0828')
nombre : "jkhhjjjkjhj"
_id: ObjectId('67ecbd152df0a3dfdaa566a6')
nombre : "jkhhjjjkjhjfghtgbg"
_id: ObjectId('67ecbd362df0a3dfdaa566a7')
nombre : "hxtfgxxdgdxrhdxfhrtf"
_id: ObjectId('67eccdbcf3738d3ff1d993f1')
nombre : "asdasfevsbjtyjt"
_id: ObjectId('67ed85a25e17642a11a6b0d7')
nombre : "romantico"
_id: ObjectId('67ed9f351939e6e4bfaf1c70')
nombre : "dghhytrff"
_id: ObjectId('67f4a1246ae9b4e7d0d654ed')
nombre : "terrorismoo"


MongoDB Compass - cluster0.pkaaf.mongodb.net/test.pelicula
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
cluster0.pkaaf.mongodb.n...
BIBLIOTECA
DEPORTES
INMOBILIARIA
INSTRUCTORES_SENA
instructores
TIENDAADSO
PRODUCTOS
admin
adsocauca
config
local
sample_mflix
test
genero
pelicula
usuario
sena
usuario
cluster0....
pelicula
PROD...
TIEND...
instruc...
mongosh: clu...
cluster0.pkaaf.mongodb.net > test > pelicula
Documents 5
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE
_id: ObjectId('67e6ef6fe0e75dec21c7e031')
codigo : 123
titulo : "lucho y contanza"
protagonista : "camilo"
duracion : 30
resumen : "las 50 sombras yaaaa"
genero : ObjectId('67e6ef13e0e75dec21c7e02e')
foto : ""
_id: ObjectId('67ed8b754560849ac613eb11')
codigo : 3455
titulo : "holaaaaaa"
protagonista : "dgfhgn"
duracion : 50
resumen : "rdxhtjxghjhgnddddddsd"
foto : ""
genero : ObjectId('67ecbb356a974f53d8dce9fc')
67ecbb356a974f53d8dce9fc
_id: ObjectId('67ed9f851939e6e4bfaf1c72')
codigo : 2334
titulo : "hoyyy"
protagonista : "thrxgv"
duracion : 40
resumen : "hytjghjjh"
foto : ""
genero : ObjectId('67e6ef1ee0e75dec21c7e02f')

2. **Desarrollo del Backend:**

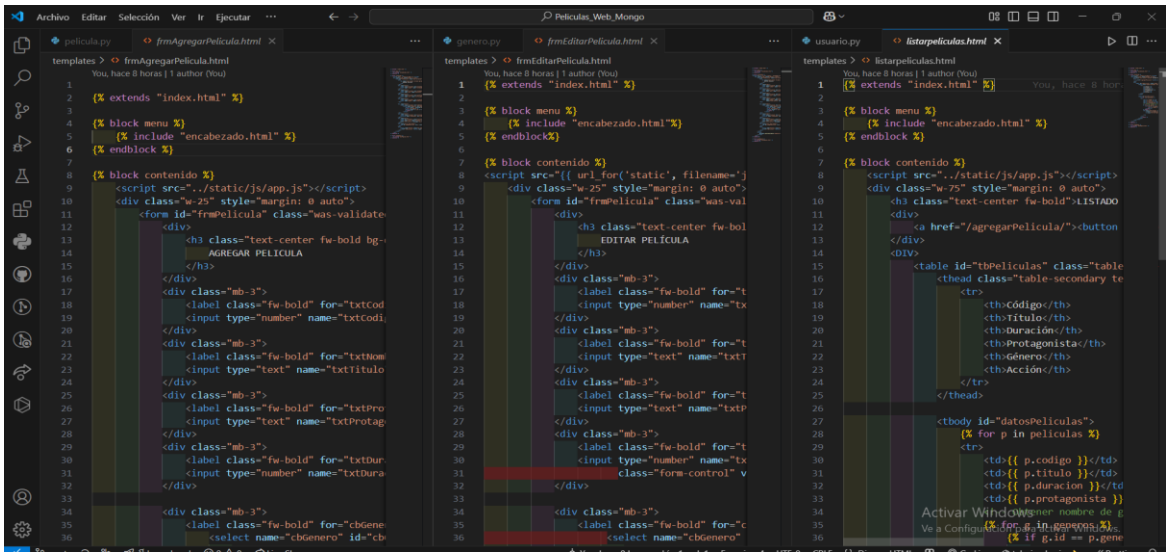
   - Se utilizó Flask para crear las rutas y controladores que manejarían las peticiones del cliente. Las rutas CRUD fueron configuradas para las películas y los géneros de películas.
   - Se implementó el sistema de registro de usuarios y autenticación utilizando sesiones.
   - Se integró reCAPTCHA en el formulario de registro para evitar bots y asegurar la validez de los registros.



3. **Desarrollo del Frontend:**

   - Se diseñó una interfaz de usuario sencilla y funcional utilizando HTML, CSS y JavaScript. A continuación, se detallan algunos fragmentos clave de código:

   HTML:


Archivo Editar Selección Ver Ir Ejecutar
Películas_Web_Mongo
película.py
frmAgregarPelicula.html
templates > frmAgregarPelicula.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html" %}
{% endblock %}
{% block contenido %}
<script src="../static/js/app.js"></script>

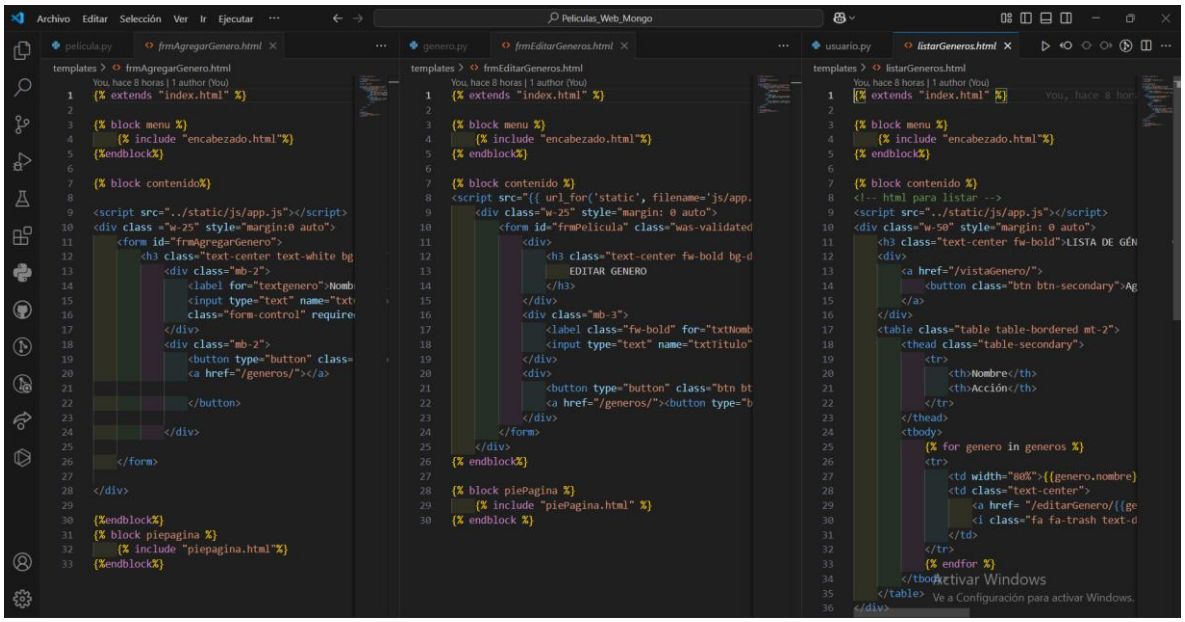
<form id="frmPelicula" class="was-validate
<h3 class="text-center fw-bold bg-
AGREGAR PELICULA
<label class="fw-bold" for="txtCod
<input type="number" name="txtCodi
<label class="fw-bold" for="txtNom
<input type="text" name="txtTitulo
<label class="fw-bold" for="txtPro
<input type="text" name="txtProtag
<label class="fw-bold" for="txtDur
<input type="number" name="txtDura
<label class="fw-bold" for="cbGene
<select name="cbGenero" id="cb
genero.py
frmEditarPelicula.html
templates > frmEditarPelicula.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html"%}
{% endblock%}
{% block contenido %}
<script src="{{ url_for('static', filename='j

<form id="frmPelicula" class="was-val
<h3 class="text-center fw-bol
EDITAR PELÍCULA
<label class="fw-bold" for="t
<input type="number" name="tx
<label class="fw-bold" for="t
<input type="text" name="txtT
<label class="fw-bold" for="t
<input type="text" name="txtP
<label class="fw-bold" for="t
<input type="number" name="tx
class="form-control" v
<label class="fw-bold" for="c
<select name="cbGenero"
usuario.py
listarpeliculas.html
templates > listarpeliculas.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html" %}
{% endblock %}
{% block contenido %}
<script src="../static/js/app.js"></script>

<h3 class="text-center fw-bold">LISTADO

<a href="/agregarPelicula/"><button

<DIV>
<table id="tbPeliculas" class="table
<thead class="table-secondary te
<tr>
<th>Código</th>
<th>Título</th>
<th>Duración</th>
<th>Protagonista</th>
<th>Género</th>
<th>Acción</th>
</tr>
</thead>
<tbody id="datosPeliculas">
{% for p in peliculas %}
<tr>
<td>{{ p.codigo }}</td>
<td>{{ p.titulo }}</td>
<td>{{ p.duracion }}</td
<td>{{ p.protagonista }}
{% for g in generos %}
{% if g.id == p.gene
Activar Windows
Ve a Configuración para activar Windows.


Archivo Editar Selección Ver Ir Ejecutar
Películas_Web_Mongo
película.py
frmAgregarGenero.html
templates > frmAgregarGenero.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html"%}
{%endblock%}
{% block contenido%}
<script src="../static/js/app.js"></script>

<form id="frmAgregarGenero">
<h3 class="text-center text-white bg

<label for="textgenero">Nomb
<input type="text" name="txt
class="form-control" require

<button type="button" class=

</button>

</form>

{%endblock%}
{% block piepagina %}
{% include "piepagina.html"%}
{%endblock%}
genero.py
frmEditarGeneros.html
templates > frmEditarGeneros.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html"%}
{% endblock%}
{% block contenido %}
<script src="{{ url_for('static', filename='js/app.

<form id="frmPelicula" class="was-validated

<h3 class="text-center fw-bold bg-d
EDITAR GENERO
<label class="fw-bold" for="txtNomb
<input type="text" name="txtTitulo"
<button type="button" class="btn bt
<button type="b
</form>
{% endblock%}
{% block piePagina %}
{% include "piePagina.html" %}
{% endblock %}
usuario.py
listarGeneros.html
templates > listarGeneros.html
{% extends "index.html" %}
{% block menu %}
{% include "encabezado.html"%}
{% endblock%}
{% block contenido %}
<!-- html para listar -->
<script src="../static/js/app.js"></script>

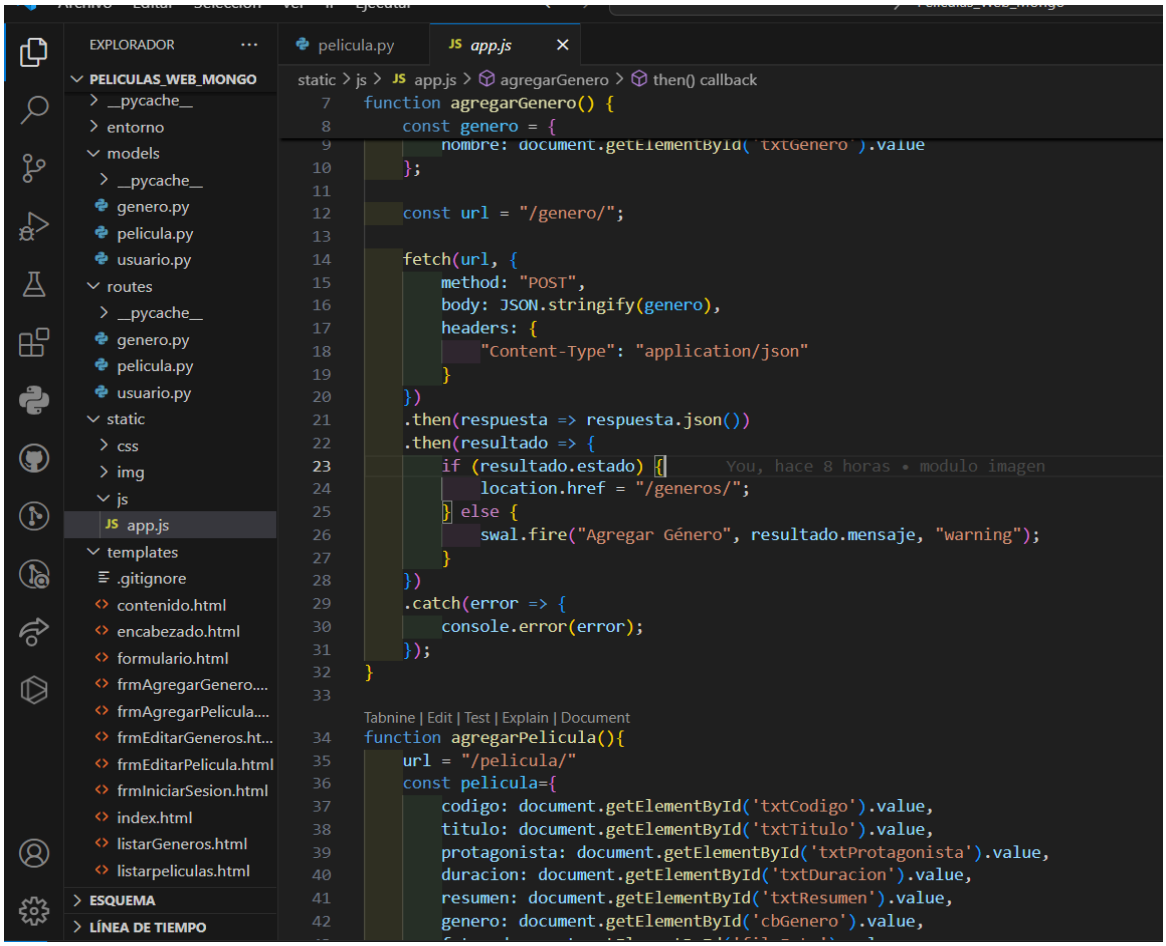
<h3 class="text-center fw-bold">LISTA DE GÉN

<button class="btn btn-secondary">Ag

<table class="table table-bordered mt-2">
<thead class="table-secondary">
<tr>
<th>Nombre</th>
<th>Acción</th>
</tr>
</thead>
<tbody>
{% for genero in generos %}
<tr>
<td width="80%">{{genero.nombre}
<td class="text-center">
<a href= "/editarGenero/{{ge
<i class="fa fa-trash text-d
</td>
</tr>
{% endfor %}
</tbody>
</table>

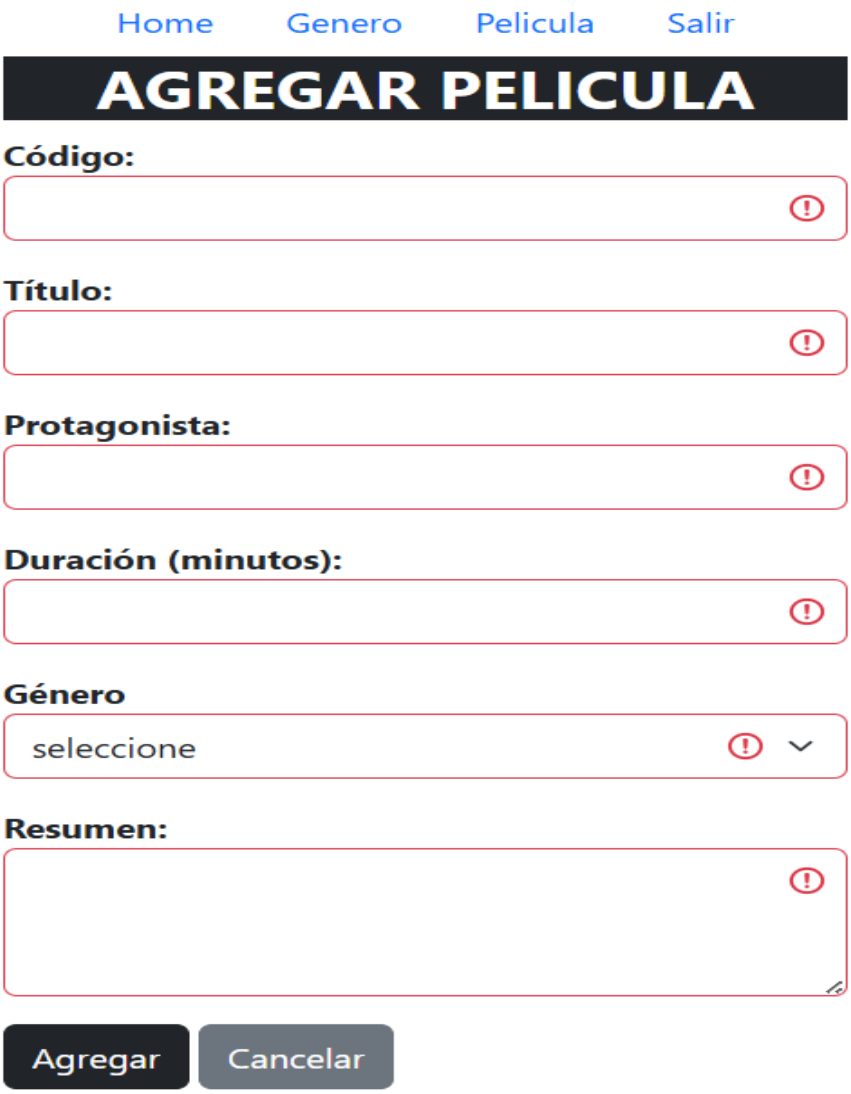
Activar Windows
Ve a Configuración para activar Windows.

JavaScript:

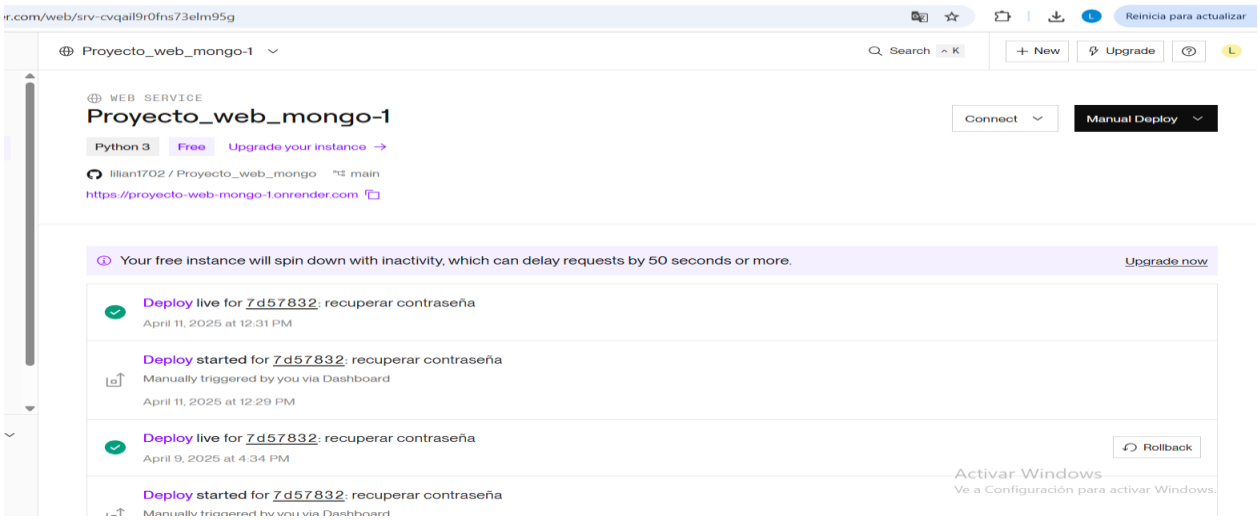

EXPLORADOR
PELICULAS_WEB_MONGO
__pycache__
entorno
models
__pycache__
genero.py
pelicula.py
usuario.py
routes
__pycache__
genero.py
pelicula.py
usuario.py
static
css
img
js
app.js
templates
.gitignore
contenido.html
encabezado.html
formulario.html
frmAgregarGenero....
frmAgregarPelicula....
frmEditarGeneros.ht...
frmEditarPelicula.html
frmIniciarSesion.html
index.html
listarGeneros.html
listarpeliculas.html
ESQUEMA
LÍNEA DE TIEMPO
pelicula.py
app.js
static > js > app.js > agregarGenero > then() callback
function agregarGenero() {
const genero = {
nombre: document.getElementById('txtGenero').value
};
const url = "/genero/";
fetch(url, {
method: "POST",
body: JSON.stringify(genero),
headers: {
"Content-Type": "application/json"
}
})
.then(respuesta => respuesta.json())
.then(resultado => {
if (resultado.estado) {
You, hace 8 horas • modulo imagen
location.href = "/generos/";
} else {
swal.fire("Agregar Género", resultado.mensaje, "warning");
}
})
.catch(error => {
console.error(error);
});
}
Tabnine | Edit | Test | Explain | Document
function agregarPelicula(){
url = "/pelicula/"
const pelicula={
codigo: document.getElementById('txtCodigo').value,
titulo: document.getElementById('txtTitulo').value,
protagonista: document.getElementById('txtProtagonista').value,
duracion: document.getElementById('txtDuracion').value,
resumen: document.getElementById('txtResumen').value,
genero: document.getElementById('cbGenero').value,


INICIAR SESIÓN
Username
12345
Contraseña:
No soy un robot
reCAPTCHA
Privacidad - Términos
Ingresar
Cancelar
recuperar contraseña

4. **Pruebas y Depuración:**
    - Se realizaron pruebas de funcionalidad para verificar que todas las rutas y características, como el CRUD y la autenticación, funcionaran correctamente.
    - Se solucionaron errores encontrados durante las pruebas y se mejoró la estabilidad de la aplicación.

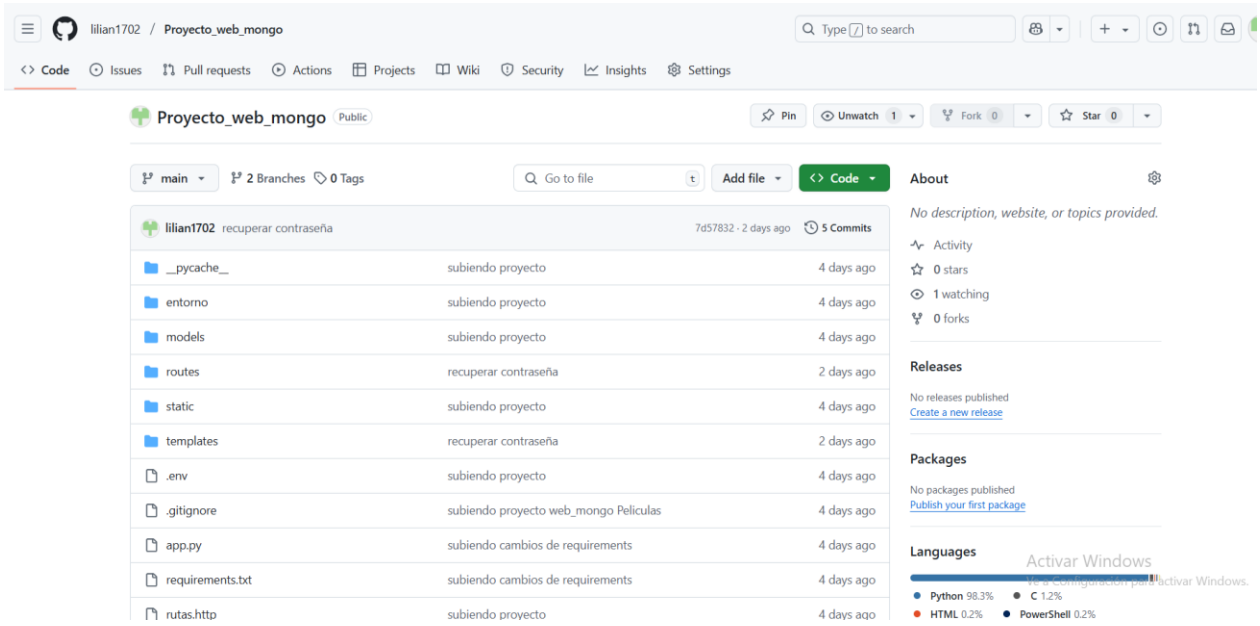


5. **Despliegue en Render:**
    - Se configuró y desplegó la aplicación en Render, asegurándose de que la aplicación estuviera accesible de forma pública.



6. **Subida al Repositorio GitHub:**

   - El código fuente del proyecto se subió a un repositorio en GitHub para facilitar el acceso, seguimiento y futuras colaboraciones.



Conclusiones

- El desarrollo de la aplicación "Gestión Películas" permitió una comprensión más profunda de las tecnologías Flask, Python y MongoDB, así como el uso de herramientas adicionales como DataTables y reCAPTCHA.
- El proceso de implementación en Render y el uso de GitHub para el control de versiones ofrecieron una experiencia valiosa en la gestión de proyectos y el trabajo colaborativo en la nube.
- A través de este proyecto, los aprendizajes adquiridos sobre el desarrollo web y el uso de herramientas modernas fueron fundamentales para mejorar las habilidades técnicas de los participantes.

Recomendaciones para Futuras Prácticas

1. Continuar con la exploración de tecnologías complementarias, como frameworks de frontend (por ejemplo, React o Vue.js), para mejorar la interactividad y la experiencia del usuario.
2. Ampliar las funcionalidades de la aplicación, como la integración de un sistema de comentarios para las películas o la implementación de un sistema de puntuación.
3. Realizar pruebas de carga y optimización para mejorar el rendimiento de la aplicación a medida que crece en términos de usuarios y datos.
4. Fomentar el trabajo colaborativo en GitHub, utilizando ramas y pull requests para gestionar de manera eficiente el código y el desarrollo del proyecto.