



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

### MATERIAL APOYO DESARROLLO DE APLICACIONES EN PYTHON CON FLASK-MONGOALCHEMY

#### APLICACIÓN GESTIÓN PELÍCULAS

A continuación se presenta material de para los siguientes requerimientos.

- Enviar correo Electrónico mediante librería Yagmail con archivos adjuntos.
- Uso de CORS
- Librería DataTables
- Publicación de la aplicación en la nube en render.
- Actualizar el recaptcha para que funcione en la url de la aplicación desplegada

#### Enviar Correo Electrónico con archivos adjuntos:

La librería permite adjuntar archivos de dos formas así:

1. **La primera forma** es que en el atributos content vaya una lista que contenga el mensaje y los archivos adjuntos. Para adjuntar archivos debemos colocar la ruta donde se encuentra el archivo que se quiere enviar en el correo electrónico. **Ejemplo:**

```
email = yagmail.SMTP("cesarmcuellar@gmail.com",os.environ.get("PASSWORD-ENVIAR-CORREO"),
                    encoding="utf-8")
asunto = "Ingreso al Sistema"
mensaje = f"Cordial saludo <b>{usuario.nombres} {usuario.apellidos}</b> \
Bienvenido a nuestro aplicativo Gestión películas. \
Enviamos Manual de usuario del aplicativo en formato pdf.<br><br>\
Cordialmente,<br><br><br> \
<b>Administración<br>Aplicativo Gestión Películas.</b>"
thread = threading.Thread(target=enviarCorreo,
                          args=(email, [usuario.correo,"ccuellar@sena.edu.co"],
                                asunto, [mensaje,"Manual.pdf"]))
thread.start()
```

De acuerdo con el código anterior estamos enviando como 4 párametro una lista que incluye el texto del mensaje y un archivo que se llama **Manual.pdf** que se encuentra en la raíz del proyecto.

```
Tabnine | Edit | Test | Explain | Document
def enviarCorreo(email=None, destinatario=None, asunto=None, mensaje=None):
    try:
        email.send(to=destinatario, subject=asunto, contents=mensaje)
    except Exception as error:
        print(str(error))
```



2. La **segunda opción** podemos agregar un 5 parámetro a la función `enviarcorreo` y a la hora de llamar la función enviamos el o los archivos en una lista como 5 parámetro así:

```
email = yagmail.SMTP("cesarmcuellar@gmail.com",os.environ.get("PASSWORD-ENVIAR-CORREO"),
| | | | | encoding="utf-8")
asunto = "Ingreso al Sistema"
archivosAdjuntos = ["Manual.pdf", "./static/imagenes/avatar.png"] ←
mensaje = f"Cordial saludo <b>{usuario.nombres} {usuario.apellidos}</b> \
| Bienvenido a nuestro aplicativo Gestión películas. \
| Enviamos Manual de usuario del aplicativo en formato pdf.<br><br>\
| Cordialmente,<br><br><br> \
| <b>Administración<br>Aplicativo Gestión Películas.</b>"
thread = threading.Thread(target=enviarCorreo,
| | | | | args=(email, [usuario.correo,"ccuellar@sena.edu.co"],
| | | | | asunto, mensaje, archivosAdjuntos))
thread.start() ↑
```

Como se muestra en el código anterior se incluyó un parámetro nuevo donde enviamos una lista de archivos.

```
Tabnine | Edit | Test | Explain | Document
def enviarCorreo(email=None, destinatario=None, asunto=None, mensaje=None, archivosAdjuntos=None):
| try:
|     email.send(to=destinatario, subject=asunto, contents=mensaje, attachments=archivosAdjuntos)
| except Exception as error:
|     print(str(error))
```

En la función `enviarCorreo` donde el objeto `email` llama al método `send`, se configura un parámetro llamado `attachments` el cual recibe una lista de archivos adjuntos a enviar al correo.

## USO DE CORS:

CORS (**Cross-Origin Resource Sharing**) es un mecanismo de seguridad implementado en los navegadores web que permite o restringe las solicitudes HTTP que se realizan desde un dominio diferente al del recurso solicitado. Específicamente, **CORS** se utiliza para controlar el acceso a recursos en un servidor desde un origen distinto al del servidor.

Por defecto, los navegadores restringen las solicitudes **HTTP** entre orígenes distintos por razones de seguridad. Esto significa que un script que se ejecuta en un origen (como `https://mi-dominio.com`) no puede hacer solicitudes a un recurso en otro origen (como `https://otro-dominio.com`) a menos que el servidor de destino permita explícitamente estas solicitudes.

En aplicaciones Flask podemos incorporar una librería llamada **Flask-Cors** que facilita la habilitación de **CORS**. Al usar `flask-cors`, los desarrolladores pueden configurar fácilmente las políticas de **CORS** para sus aplicaciones, permitiendo o restringiendo el acceso según sea necesario



## Implementación de CORS en Flask

Instalar `pip install flask_cors`

En el archivo `app.py` donde se crea el objeto de la aplicación flask hacer la siguiente importación:

```
from flask_cors import CORS
```

Adicionalmente en ese mismo archivo agregar la siguiente línea la cuál permitirá **CORS** para todas las rutas.

```
CORS(app)
```

## Uso de Librería DataTables

Datatables es una librería de javascript que permite mejorar las tablas html, adicionalmente permite hacer búsquedas, ordenar, filtrar y páginar. También es posible exportar los datos de la tabla en diferentes formatos como csv, Excel o pdf.

Ejemplo de como se vería la lista de las películas usando datatables con la configuración básica.

 **Marino Chacón** 

### LISTADO DE PELÍCULAS

Agregar

10  entries per page

Search:

Código 	Título 	Duración 	Protagonista 	Género 	Acción 
12	sesesde	125	reres	sesese	 
100	SuperMen	120	Camilinderere	Infantil	 
888	Rambo 5	130	Arnold	Comedia	 
1000	La sombra	120	Pedro Infante	Terror	 
234234	xxxxx	169	ddasdasd	Comedia	 

Showing 1 to 5 of 5 entries 

  1  

**Tecnólogo en Análisis y Desarrollo de Software**  
**Derechos Reservados @copyright**



## Implementación DataTables:

Agregar al archivo html donde se listan las películas los siguientes CDN de acuerdo con las indicaciones del sitio oficial de DataTables.

```
<!--datatables-->
<script src="https://code.jquery.com/jquery-3.7.1.js"></script>
<script src="https://cdn.datatables.net/2.2.2/js/dataTables.min.js"></script>
<link rel="stylesheet" href="https://cdn.datatables.net/2.2.2/css/dataTables.dataTables.min.css">
<!--fin-->
```

Colocarle un **id** a la tabla para identificarla.

```
<table id="tbPelículas" class="table table-bordered mt-2" >
  <thead class="table-secondary text-center">
    <tr>
      <th>Código</th>
      <th>Titulo</th>
      <th>Duración</th>
      <th>Protagonista</th>
      <th>Género</th>
      <th>Acción</th>
```

En el archivo **app.js** agregar la siguiente línea que crea el objeto datatable con la tabla.

```
//crear el estilo de datatable a la tabla con id = tbPelículas
new DataTable('#tbPelículas');
```

También lo puede colocar al final del documento html así:

```
<script>
  //crear el estilo de datatable a la tabla con id = tbPelículas
  new DataTable('#tbPelículas');
</script>
```



## Publicación de la Aplicación en un servidor de Internet

Se realizará el proceso en **render.com**

### Pasos:

1. **Utilizar variables de entorno** para no exponer datos sensibles por seguridad, como por ejemplo la cadena de conexión a mongo atlas que contiene usuario y contraseña; igualmente la contraseña de aplicación creada para envío de correo electrónico, la clave pública y secreta para el uso del recaptcha. Estas variables se crean en un archivo **.env**, como se muestra a continuación con un ejemplo.

```
PASSWORD-BD=232132
USER-BD=usermongo
CORREO=cesarmcuellar@gmail.com
PASSWORD-ENVIAR-CORREO=zwakhyabcdtqqdnqe
SECRET-RECAPTCHA=6LdYkgYrAAAAA79Wmt8S-xxxxxVCmSpqTCHt1XUI
SITE-RECAPTCHA=6LdYkgYrAAAAECfyjeYzhxxxxxxEfkNJXVg6Yo
URI=mongodb+srv://${USER-BD}:${PASSWORD-DB}@runt.oudoapr.mongodb.net/?retryWrites=true&w=majority&appName=RUNT
```

2. **Instalar la librería gunicorn.** Está librería se requiere para la publicación en render.

**pip install gunicorn**

3. **Actualizar el archivo requirements.txt** que contendrá las librerías requeridas para el funcionamiento del proyecto.

**pip freeze > requirements.txt**

4. Verificar que el archivo **app.py** que inicializa la aplicación en el bloque **if** solo tenga el arranque de la aplicación. Debe quedar como se muestra a continuación. La **importación de rutas** que teníamos inicialmente dentro del if, las colocamos antes del if. Debemos dejar que quede así debido a que gunicorn es quien arranca el servidor, por lo tanto no se ejecuta el if. (incluso no es necesario tener el if).

```
#importas las rutas
from routes.usuario import *
from routes.genero import *
from routes.pelicula import *

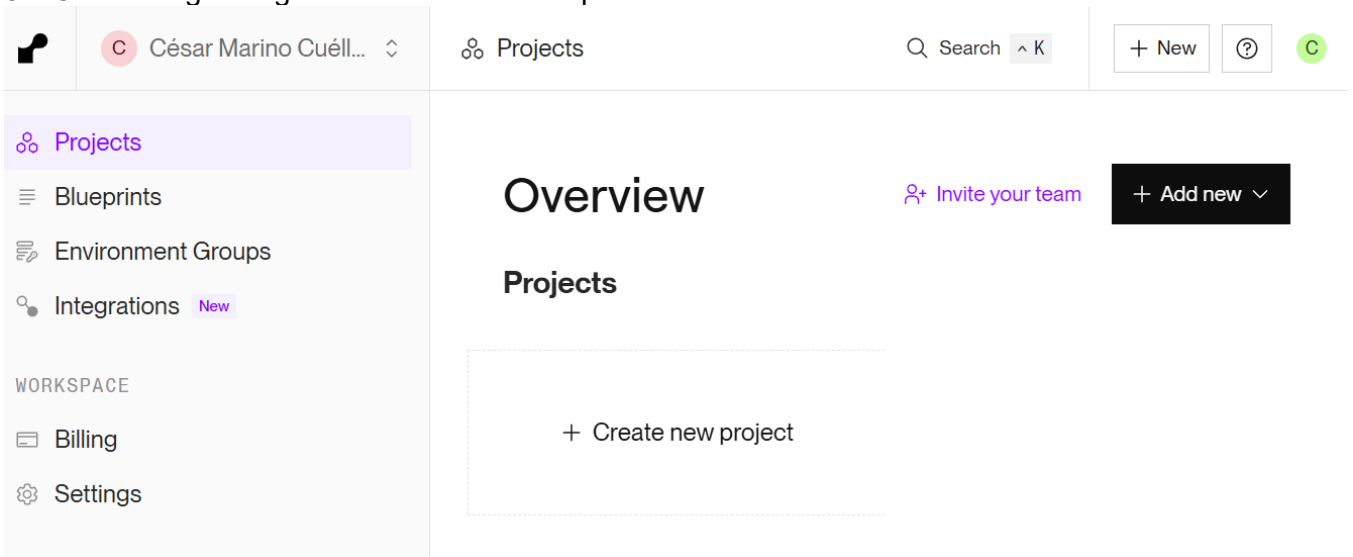
if __name__ == "__main__":
    app.run(port=5000, host="0.0.0.0", debug=True)
```



5. No olvidarnos de tener un archivo llamado **.gitignore**, en el cual vamos a colocar lo que queremos **excluir del push** al repositorio de github. **Ejemplo:** En el ejemplo se ha colocado el nombre de la carpeta del entorno que se llama entorno y también el archivo .env que contiene datos sensibles.

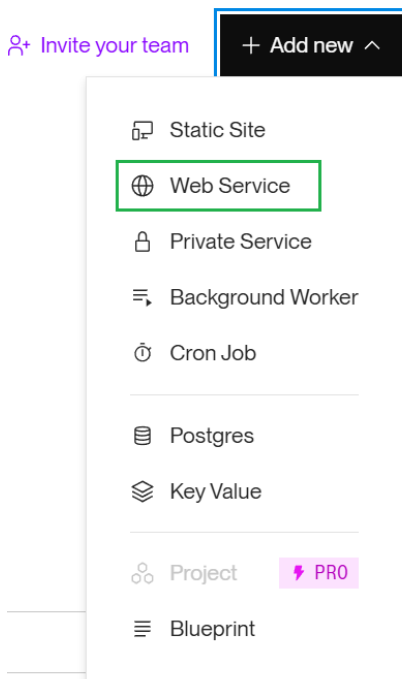
```
.gitignore
1  entorno
2  .env
```

6. Realizar los pasos para hacer el push a cada uno de sus repositorios en github.
7. Ingresar al render.com. Pueden crear un usuario con su correo electrónico o se pueden logear con github.
8. Cuando logren ingresar con su usuario aparece una interfaz así:





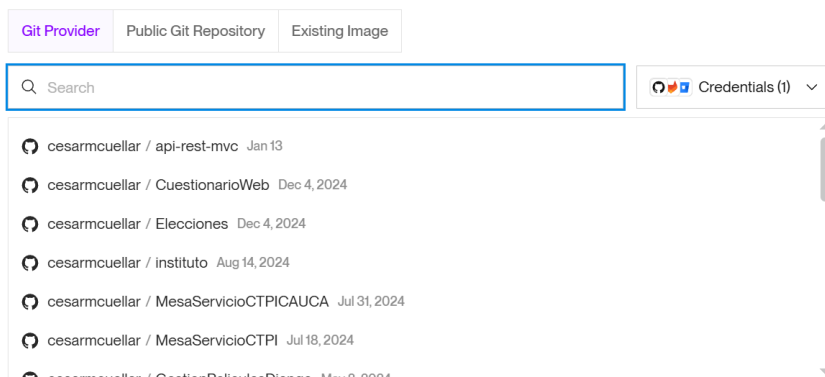
En el Botón **Add New**: aparecen unas opciones dar clic en Web Service.



Al dar clic en **WebService** aparece la siguiente interfaz:

You are deploying a Web Service

Source Code



Si usted ha ingresado con las **credenciales de GitHub** le aparecen en la pestaña **Git Provider** todos sus repositorios. Si es así, usted da clic en el repositorio, inmediatamente lo lleva a una interfaz para la configuración del servicio. Si no le aparecen los repositorios, dar clic en la pestaña Public Git Repository y le aparece la siguiente interfaz para que usted coloque la url del repositorio.



Git Provider

Public Git Repository

Existing Image

PR Previews and Auto-Deploy are available only for repositories configured with render.yaml

 `https://github.com/render-examples/clickhouse`

Connect →

Git Provider

Public Git Repository

Existing Image

PR Previews and Auto-Deploy are available only for repositories configured with render.yaml

 `https://github.com/CesarMCuellarCha/PELICULAS_FLASK_MONGOENGINE`

Connect →

Después de ingresar la url de su repositorio si es público, dar clic en connect donde lo lleva a la siguiente interfaz:

Source Code

 CesarMCuellarCha / PELICULAS\_FLASK\_MONGOENGINE

Edit


Name


A unique name for your web service.

FLASK\_PELICULAS

Project Optional

Add this web service to a [project](#) once it's created.

 Select a project...

 Select an environment...

Language

Python 3


Branch

The Git branch to build and deploy.

main

Region

Your services in the same [region](#) can communicate over a [private network](#). You currently have services running in Oregon.

 Oregon (US West)

5 existing services

Deploy in a new region +

Esas primeras opciones que aparecen se pueden dejar tal cual, la propiedad Name, la pueden cambiar si desean.





#### Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

e.g. `src`

#### Build Command

Render runs this command to build your app before each deploy.

```
$ pip install -r requirements.txt
```

#### Start Command

Render runs this command to start your app with each deploy.

```
$ gunicorn app:app
```

### Instance Type

For hobby projects

Free

\$0 / month

512 MB (RAM)

0.1 CPU

#### ⚠ Upgrade to enable more features

Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

En la imagen anterior nos aparece la propiedad **Build Command** que contiene el comando que ejecutará para instalar las librerías. Noten ustedes que el solo colocó que va a instalar las librerías de un archivo llamado **requirements.txt**. También aparece el **Start Command** que es el comando que arranca la aplicación `$ gunicorn app:app`. Por eso la instalación en nuestro proyecto de la librería gunicorn. En dicho comando el app de color verde o sea el primer app que aparece en el comando es el nombre del archivo que arranca nuestra aplicación, que para nuestro caso lo llamados así; pero si de pronto ustedes lo llamaron de otra forma deben cambiarlo.

En la parte inferior de la imagen aparece lo relacionado con el tipo de instancia **Instance Type**. Aquí debemos seleccuionar **Free**.

Más abajo aparece lo de las variables de entorno:

#### Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

NAME\_OF\_VARIABLE

value

Generate



+ Add Environment Variable

Add from .env

Render nos permite agregar nuestras variables de entorno. Para ello dar clic en el botón **Add from .env**, y nos aparece la siguiente interfaz:



## Add from .env

Paste your .env contents to add multiple environment variables at once. Check out the [docs](#) for correct syntax.

```
KEY_1=VALUE_1  
KEY_2=VALUE_2  
KEY_3=VALUE_3
```

Choose a file ↕

Cancel

Add variables

## Add from .env

Paste your .env contents to add multiple environment variables at once. Check out the [docs](#) for correct syntax.

```
PASSWORD-BD=2312312  
USER-BD=usermongo  
CORREO=cesarmcuellar@gmail.com  
PASSWORD-ENVIAR-CORREO=zwakhyabcdqdnqe  
SECRET-RECAPTCHA=6LdYkgYrAAAAAJxxxxx-YMA5dVCmspqTCHt1XUI  
SITE-RECAPTCHA=6LdYkgYrAAAAECxxxxxx7gUBBxEfKJXVg6Yo  
URI=mongodb+srv://${USER-BD}:${PASSWORD-BD}@runt.oudoapr.mongodb.net/?  
retryWrites=true&w=majority&appName=RUNT
```

Choose a file ↕

Cancel

Add variables

El paso siguiente es ir a nuestro proyecto seleccionar las variables de entorno y copiarlos en el espacio indicado para ello. Después de copiarlas dar clic en el botón **Add Variables**.

Por último dar clic en el botón **Deploy Web Service**

Deploy Web Service

Después lo lleva a una interfaz donde render empieza a desplegar el aplicativo.

April 6, 2025 at 3:06 PM Building  
a2b1fa4 Actualización datatables

Cancel deploy

```
All logs Search Live tail GMT-5  
Apr 6 03:06:52 PM --> Checking out commit a2b1fa40612a1013069c202c0049/0a1c0000 in branch main  
Apr 6 03:06:53 PM --> Using Python version 3.11.11 (default)  
Apr 6 03:06:53 PM --> Docs on specifying a Python version: https://render.com/docs/python-version  
Apr 6 03:06:57 PM --> Using Poetry version 1.7.1 (default)  
Apr 6 03:06:57 PM --> Docs on specifying a Poetry version: https://render.com/docs/poetry-version  
Apr 6 03:06:57 PM --> Running build command 'pip install -r requirements.txt'...  
Apr 6 03:06:57 PM Collecting flask-mongoengine@ git+https://github.com/idoshr/flask-mongoengine.git@e244408acf440c4208f7ddcd6e5d819cb472e4da (from -r requirements.txt (line 13))  
Apr 6 03:06:57 PM Cloning https://github.com/idoshr/flask-mongoengine.git (to revision e244408acf440c4208f7ddcd6e5d819cb472e4da) to /tmp/pip-install-x6764iqo/flask-mongoengine_c9a60b54735848b5b3b3b266b0eadeb  
Apr 6 03:06:57 PM Running command git clone --filter=blob:none --quiet https://github.com/idoshr/flask-mongoengine.git /tmp/pip-install-x6764iqo/flask-mongoengine_c9a60b54735848b5b3b3b266b0eadeb  
Apr 6 03:06:58 PM Running command git rev-parse -q --verify 'sha^e244408acf440c4208f7ddcd6e5d819cb472e4da'  
Apr 6 03:06:58 PM Running command git fetch -q https://github.com/idoshr/flask-mongoengine.git e244408acf440c4208f7ddcd6e5d819cb472e4da  
Apr 6 03:06:58 PM Running command git checkout -q e244408acf440c4208f7ddcd6e5d819cb472e4da  
Apr 6 03:06:58 PM Resolved https://github.com/idoshr/flask-mongoengine.git to commit e244408acf440c4208f7ddcd6e5d819cb472e4da  
Apr 6 03:06:58 PM Installing build dependencies: started
```



Al final en la interfaz aparece **your service is live**, si todo ha quedado bien, como se muestra en la siguiente imagen.

```
All logs  Search  L
tertools-10.0.0 packaging-24.2 premailer-3.10.0 pymongo-4.11.3 python-dotenv-1.1.0 requests-2.32.3 urllib3-2.3.0 yagmail-0.15.29
Apr 6 03:07:12 PM [notice] A new release of pip is available: 24.0 -> 25.0.1
Apr 6 03:07:12 PM [notice] To update, run: pip install --upgrade pip
Apr 6 03:07:17 PM --> Uploading build...
Apr 6 03:07:24 PM --> Uploaded in 6.0s. Compression took 1.1s
Apr 6 03:07:24 PM --> Build successful 🎉
Apr 6 03:07:29 PM --> Deploying...
Apr 6 03:07:46 PM --> Running 'gunicorn app:app'
Apr 6 03:07:52 PM [2025-04-06 20:07:52 +0000] [75] [INFO] Starting gunicorn 23.0.0
Apr 6 03:07:52 PM [2025-04-06 20:07:52 +0000] [75] [INFO] Listening at: http://0.0.0.0:10000 (75)
Apr 6 03:07:52 PM [2025-04-06 20:07:52 +0000] [75] [INFO] Using worker: sync
Apr 6 03:07:53 PM [2025-04-06 20:07:52 +0000] [83] [INFO] Booting worker with pid: 83
Apr 6 03:07:53 PM 127.0.0.1 - - [06/Apr/2025:20:07:53 +0000] "HEAD / HTTP/1.1" 200 0 "-" "Go-http-client/1.1"
Apr 6 03:08:00 PM --> Your service is live 🎉
Apr 6 03:08:01 PM 127.0.0.1 - - [06/Apr/2025:20:08:01 +0000] "GET / HTTP/1.1" 200 3516 "-" "Go-http-client/2.0"
```

En esa misma ventana en la parte superior aparece la url donde quedó nuestro aplicativo funcionando, podemos dar clic para revisar.

WEB SERVICE

# FLASK\_PELICULAS

Python 3

Free

Upgrade your instance →

CesarMCuellarCha / PELICULAS\_FLASK\_MONGOENGINE main

<https://flask-peliculas-ishg.onrender.com>



## Prueba en el navegador:

flask-peliculas-ishg.onrender.com

### INICIAR SESIÓN

@ Username

Contraseña:

Error para el propietario del sitio: La clave de sitio no es válida

reCAPTCHA  
Privacidad · Condiciones


Ingresar Cancelar

Tecnólogo en Análisis y Desarrollo de Software  
Derechos Reservados @copyright

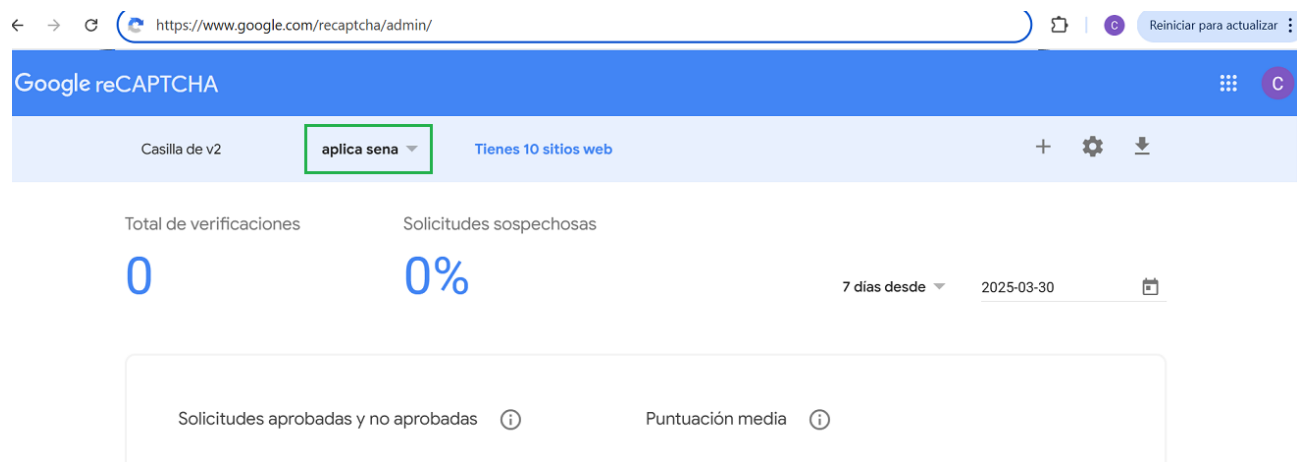
## ¿Porqué el recaptcha no está funcionando?

No está funcionando porque debemos configurarlo para que funcione en la url de nuestra aplicación, ya que cuando lo creamos solo colocamos localhost y 127.0.0.1

## Ingresa al navegador la siguiente url:

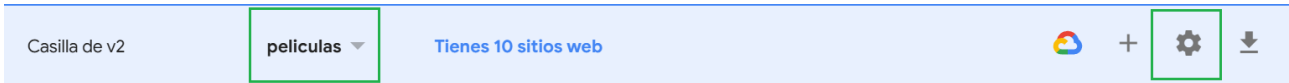
 <https://www.google.com/recaptcha/admin/>

## Nos aparece la siguiente interfaz:

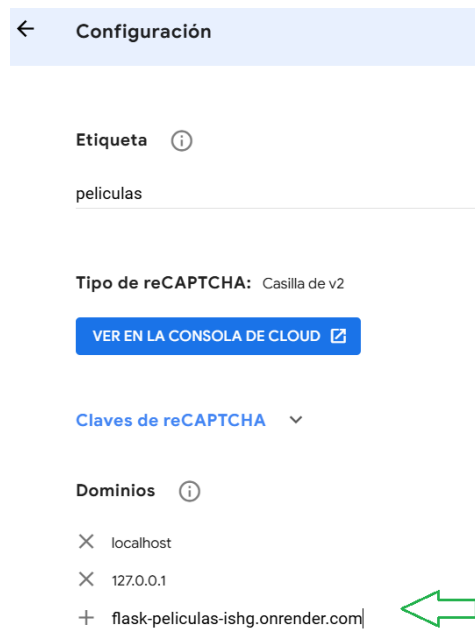




En la parte donde indica el cuadro en verde aparecen los **recaptcha** configurados para sus sitios web, en mi caso voy a seleccionar el creado para la aplicación y voy a agregar la url de la aplicación que se acaba de desplegar.

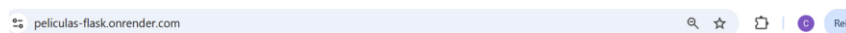


Seleccione películas y doy clic en el icono del lado derecho donde indica el cuadro en verde. Aparece la siguiente interfaz, donde en la parte de Dominios agrego la url de la aplicación que se acaba de desplegar. Se debe colocar sin el https como se muestra a continuación.



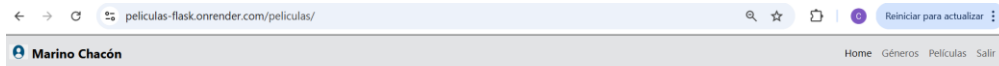
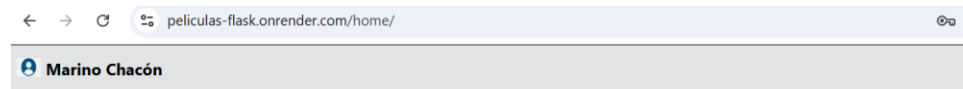
Después se da clic en el botón **Guardar** de la parte inferior.

Ahora podemos verificar si el recaptcha funciona en el sitio.





El paso siguiente es probar las funcionalidades ingresando con un usuario válido. Aquí unos ejemplos.





**Trabajo para realizar cada uno de ustedes.**

- **Implementar la funcionalidad de recuperar contraseña:** Dicha implementación debe generar una nueva contraseña de longitud de 8 caracteres y enviarla al usuario por correo electrónico. Recuerden que no solo es generarla, también deben actualizar la contraseña del usuario por la nueva generada.

- El miércoles **9 de abril de 2025** deben presentar toda la aplicación funcionando y desplegada en render.
  - Repositorio en Github
  - Despliegue en Render.

### Referencias:

- DataTables: <https://datatables.net/>
- Librería Yagmail: <https://github.com/kootenpv/yagmail>
- Librería Flask-Cors: <https://corydolphin.com/flask-cors/>
- Recaptcha Developer: <https://developers.google.com/recaptcha/intro?hl=es-419>
- Librería Google\_Recaptcha\_Flask: <https://github.com/brunolimas/Flask-Google-reCaptcha>
- Servidor para despliegue de aplicaciones: [www.render.com](http://www.render.com)

### CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
<b>Autor (es)</b>	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	06-04-2025