**✅ CHECKLIST GENERAL DEL PROYECTO PPC**

**✅ Paso 1: Preparación del entorno**

**🧩 1.1 Instalar Python 3.10 o superior**

**🔧 Pasos:**

1. Ve a la web oficial de Python:  
   👉 <https://www.python.org/downloads/>
2. Descarga la versión **Python 3.10+** según tu sistema operativo.
3. Ejecuta el instalador y **marca la opción** ✅ Add Python to PATH.
4. Verifica que se instaló correctamente:
   * Abre la terminal o símbolo del sistema.
   * Escribe:
   * python --version
   * Debe devolver algo como: Python 3.10.12

**🖥️ 1.2 Instalar Visual Studio Code**

**🔧 Pasos:**

1. Ve a la web oficial de VSCode:  
   👉 <https://code.visualstudio.com/>
2. Descarga la versión para tu sistema operativo.
3. Instálalo con todas las opciones por defecto.
4. Abre VSCode y asegúrate de instalar:
   * La extensión **Python (Microsoft)**.
   * La extensión **Flask Snippets** (opcional).

**🗃️ 1.3 Instalar MySQL y crear base de datos**

**Opción A: Instalación local**

1. Instala **XAMPP** o **MySQL Workbench**.
   * XAMPP: <https://www.apachefriends.org/index.html>
2. Abre **phpMyAdmin** desde XAMPP o el navegador:
   * URL: <http://localhost/phpmyadmin>
3. Crea una base de datos llamada:
4. ppc\_portal
5. Crea un usuario MySQL o usa root para conexión local.

**Opción B: Usar SiteGround (más adelante en el paso 7)**

**📁 1.4 Crear estructura de carpetas del proyecto**

**🔧 Pasos:**

1. Abre Visual Studio Code.
2. Crea una nueva carpeta del proyecto:
3. ppc\_portal/
4. Dentro de esta carpeta, crea las siguientes subcarpetas y archivos:

ppc\_portal/

│

├── app.py

├── config.py

├── requirements.txt

│

├── /models/

│ ├── user.py

│ ├── ad.py

│ └── click.py

│

├── /templates/

│ ├── login.html

│ ├── ad.html

│ └── success.html

│

├── /static/

│ ├── styles.css

│ └── main.js (opcional)

**🐍 1.5 Crear entorno virtual Python**

**🔧 Pasos:**

1. Abre la terminal en VSCode (menú superior: Terminal > New Terminal).
2. Ejecuta:
3. python -m venv venv
4. Activa el entorno virtual:
   * En **Windows**:
   * venv\Scripts\activate
   * En **Mac/Linux**:
   * source venv/bin/activate
5. Verifica que está activo: aparece (venv) al inicio de la línea.

**📦 1.6 Instalar Flask y dependencias**

**🔧 Pasos:**

1. Asegúrate de estar en el entorno virtual activo.
2. Ejecuta:
3. pip install flask flask-mysqldb
4. Guarda las dependencias en requirements.txt:
5. pip freeze > requirements.txt

**📌 Final de Paso 1**

Una vez terminado este paso, tendrás:

* Python y MySQL listos.
* Entorno de desarrollo funcional en VSCode.
* Estructura base del proyecto creada.
* Entorno virtual y Flask instalados.

**✅ Paso 2: Crear base de datos MySQL**

Este paso consiste en crear la base de datos principal del sistema y las tres tablas necesarias: users, ads y clicks.

**🧮 2.1 Crear base de datos ppc\_portal**

**Opción A – Si usas XAMPP + phpMyAdmin:**

1. Inicia XAMPP > activa MySQL.
2. Abre el navegador y ve a:  
   👉 <http://localhost/phpmyadmin>
3. En el menú izquierdo, haz clic en "Nueva".
4. Escribe:
   * Nombre: ppc\_portal
   * Codificación: utf8\_general\_ci
5. Haz clic en "Crear".

**Opción B – Desde línea de comandos:**

CREATE DATABASE ppc\_portal CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

**📄 2.2 Crear tabla users**

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL

);

⚠️ Por ahora usaremos contraseña en texto plano. En producción, se debe cifrar con werkzeug.security.

**📄 2.3 Crear tabla ads**

CREATE TABLE ads (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

url TEXT NOT NULL,

pay\_per\_click DECIMAL(5,2) NOT NULL DEFAULT 0.10

);

Ejemplo: anuncio con pago de 0.10 por clic.

**📄 2.4 Crear tabla clicks**

CREATE TABLE clicks (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

ad\_id INT NOT NULL,

ip\_address VARCHAR(45) NOT NULL,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (ad\_id) REFERENCES ads(id)

);

**🧪 2.5 Insertar registros de prueba**

INSERT INTO users (email, password)

VALUES ('usuario@example.com', '123456');

INSERT INTO ads (title, url, pay\_per\_click)

VALUES ('Anuncio 1', 'https://anuncio1.com', 0.10);

**📝 2.6 Guardar todo en archivo schema.sql**

Crea un archivo en tu proyecto:

📄 ppc\_portal/database/schema.sql  
Y guarda allí todo lo anterior:

CREATE DATABASE IF NOT EXISTS ppc\_portal;

USE ppc\_portal;

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL

);

CREATE TABLE ads (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

url TEXT NOT NULL,

pay\_per\_click DECIMAL(5,2) NOT NULL DEFAULT 0.10

);

CREATE TABLE clicks (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

ad\_id INT NOT NULL,

ip\_address VARCHAR(45) NOT NULL,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (ad\_id) REFERENCES ads(id)

);

INSERT INTO users (email, password) VALUES ('usuario@example.com', '123456');

INSERT INTO ads (title, url, pay\_per\_click) VALUES ('Anuncio 1', 'https://anuncio1.com', 0.10);

**✅ Resultado del Paso 2**

Ya tienes:

* Base de datos ppc\_portal creada.
* Tablas con claves foráneas configuradas.
* Datos de prueba insertados.
* Script SQL schema.sql para reutilizar.

**✅ Paso 3: Configurar Backend en Flask**

Este paso incluye la configuración básica del backend, conexión con la base de datos y definición de rutas iniciales.

**🧩 3.1 Crear archivo config.py**

Este archivo contendrá la configuración de conexión con MySQL.

📄 ppc\_portal/config.py:

import os

# Configuración de MySQL

MYSQL\_HOST = 'localhost'

MYSQL\_USER = 'root'

MYSQL\_PASSWORD = 'tu\_contraseña' # Cambiar si tienes contraseña

MYSQL\_DB = 'ppc\_portal'

# Seguridad Flask

SECRET\_KEY = os.urandom(24) # O pon una cadena fija en producción

⚠️ En SiteGround cambia localhost por el host remoto de la base de datos cuando migres.

**📦 3.2 Crear y activar entorno virtual (si aún no lo hiciste)**

En terminal dentro del proyecto:

python -m venv venv

# Activar entorno:

# Windows:

venv\Scripts\activate

# Mac/Linux:

source venv/bin/activate

**📦 3.3 Instalar dependencias**

pip install flask flask-mysqldb

Luego guarda el archivo:

📄 ppc\_portal/requirements.txt

flask

flask-mysqldb

**🧠 3.4 Crear archivo principal app.py**

📄 ppc\_portal/app.py (estructura básica):

from flask import Flask, render\_template, request, redirect, session

from flask\_mysqldb import MySQL

import config

app = Flask(\_\_name\_\_)

app.config.from\_object(config)

mysql = MySQL(app)

@app.route('/')

def home():

return redirect('/login')

@app.route('/login')

def login():

return render\_template('login.html')

@app.route('/verify', methods=['POST'])

def verify():

email = request.form['email']

password = request.form['password']

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM users WHERE email=%s AND password=%s", (email, password))

user = cursor.fetchone()

cursor.close()

if user:

session['user\_id'] = user[0]

return redirect('/ad')

else:

return "Login incorrecto", 401

@app.route('/ad')

def ad():

if 'user\_id' not in session:

return redirect('/login')

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM ads LIMIT 1")

ad = cursor.fetchone()

cursor.close()

return render\_template('ad.html', ad=ad)

@app.route('/click/<int:ad\_id>', methods=['POST'])

def register\_click(ad\_id):

if 'user\_id' not in session:

return redirect('/login')

user\_id = session['user\_id']

ip\_address = request.remote\_addr

cursor = mysql.connection.cursor()

cursor.execute("INSERT INTO clicks (user\_id, ad\_id, ip\_address) VALUES (%s, %s, %s)", (user\_id, ad\_id, ip\_address))

mysql.connection.commit()

cursor.close()

return render\_template('success.html')

@app.route('/logout')

def logout():

session.clear()

return redirect('/login')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**📁 3.5 Crear carpetas templates/ y static/**

En la raíz del proyecto:

mkdir templates

mkdir static

**✅ Resultado del Paso 3**

Ya tienes:

* app.py configurado con rutas principales (/login, /ad, /click, /logout)
* Conexión a la base de datos MySQL funcional
* Configuración lista para agregar las páginas HTML
* Proyecto listo para ejecutar:
* python app.py

**✅ Paso 4: Backend – Módulos de lógica por clases (POO)**

**📁 Estructura esperada:**

ppc\_portal/

│

├── app.py

├── config.py

│

├── /models/

│ ├── user.py

│ ├── ad.py

│ └── click.py

**👤 4.1 Clase User – (Archivo: models/user.py)**

class User:

def \_\_init\_\_(self, id, email, password):

self.id = id

self.email = email

self.password = password

def authenticate(self, db\_cursor):

db\_cursor.execute(

"SELECT \* FROM users WHERE email=%s AND password=%s",

(self.email, self.password)

)

result = db\_cursor.fetchone()

if result:

self.id = result[0]

return True

return False

✅ Método authenticate() valida el login usando cursor de MySQL.

**📢 4.2 Clase Advertisement – (Archivo: models/ad.py)**

class Advertisement:

def \_\_init\_\_(self, id, title, url, pay\_per\_click):

self.id = id

self.title = title

self.url = url

self.pay\_per\_click = pay\_per\_click

@staticmethod

def get\_first(db\_cursor):

db\_cursor.execute("SELECT \* FROM ads LIMIT 1")

result = db\_cursor.fetchone()

if result:

return Advertisement(\*result)

return None

✅ Método get\_first() obtiene el primer anuncio disponible en la base de datos.

**🖱️ 4.3 Clase ClickTracker – (Archivo: models/click.py)**

class ClickTracker:

@staticmethod

def register\_click(db\_cursor, user\_id, ad\_id, ip):

db\_cursor.execute(

"INSERT INTO clicks (user\_id, ad\_id, ip\_address) VALUES (%s, %s, %s)",

(user\_id, ad\_id, ip)

)

✅ Método estático para registrar clics desde app.py.

**🧩 4.4 Modificar app.py para usar las clases**

**Al inicio de app.py, importar:**

from models.user import User

from models.ad import Advertisement

from models.click import ClickTracker

**Modifica /verify:**

@app.route('/verify', methods=['POST'])

def verify():

email = request.form['email']

password = request.form['password']

cursor = mysql.connection.cursor()

user = User(None, email, password)

if user.authenticate(cursor):

session['user\_id'] = user.id

return redirect('/ad')

else:

return "Login incorrecto", 401

**Modifica /ad:**

@app.route('/ad')

def ad():

if 'user\_id' not in session:

return redirect('/login')

cursor = mysql.connection.cursor()

anuncio = Advertisement.get\_first(cursor)

cursor.close()

return render\_template('ad.html', ad=anuncio)

**Modifica /click/<int:ad\_id>:**

@app.route('/click/<int:ad\_id>', methods=['POST'])

def register\_click(ad\_id):

if 'user\_id' not in session:

return redirect('/login')

user\_id = session['user\_id']

ip\_address = request.remote\_addr

cursor = mysql.connection.cursor()

ClickTracker.register\_click(cursor, user\_id, ad\_id, ip\_address)

mysql.connection.commit()

cursor.close()

return render\_template('success.html')

**✅ Resultado del Paso 4**

Ya tienes:

* Las 3 clases (User, Advertisement, ClickTracker) aplicando POO.
* Código más limpio, separado y reutilizable.
* app.py usando estas clases correctamente.

**✅ Paso 5: Frontend – Páginas HTML + estilos**

Las plantillas estarán en la carpeta templates/ y los estilos en static/.

**📁 Estructura esperada**

ppc\_portal/

│

├── /templates/

│ ├── login.html

│ ├── ad.html

│ └── success.html

│

├── /static/

│ └── styles.css

**🔐 5.1 Página login.html**

📄 templates/login.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login - Portal PPC</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<div class="container">

<h2>Inicia sesión para continuar</h2>

<form action="/verify" method="POST">

<input type="email" name="email" placeholder="Correo electrónico" required>

<input type="password" name="password" placeholder="Contraseña" required>

<button type="submit">Entrar</button>

</form>

</div>

</body>

</html>

**📢 5.2 Página ad.html**

📄 templates/ad.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Anuncio - Portal PPC</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<div class="container ad-box">

<h2>{{ ad.title }}</h2>

<p>Haz clic en el anuncio para continuar</p>

<form action="/click/{{ ad.id }}" method="POST">

<button type="submit">Ver más</button>

</form>

</div>

</body>

</html>

Puedes cambiar Ver más por una imagen de banner o botón visual si lo deseas.

**✅ 5.3 Página success.html**

📄 templates/success.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Gracias por tu visita</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

<meta http-equiv="refresh" content="3;url=/logout">

</head>

<body>

<div class="container">

<h2>¡Gracias por tu clic!</h2>

<p>Redirigiendo al portal...</p>

</div>

</body>

</html>

Redirige automáticamente al logout después de 3 segundos. Esto simula cerrar el acceso después del anuncio.

**🎨 5.4 Estilos CSS globales**

📄 static/styles.css

body {

font-family: Arial, sans-serif;

background-color: #f1f1f1;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: white;

padding: 2em;

border-radius: 10px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

text-align: center;

width: 300px;

}

input[type="email"],

input[type="password"] {

width: 90%;

padding: 0.8em;

margin: 0.5em 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #007bff;

color: white;

padding: 0.8em 1.2em;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 1em;

}

button:hover {

background-color: #0056b3;

}

.ad-box {

width: 400px;

}

**✅ Resultado del Paso 5**

Ya tienes:

* Plantilla de login funcional (login.html)
* Visualización de anuncios con botón (ad.html)
* Página de confirmación tras clic (success.html)
* Estilos aplicados de forma limpia y responsiva (styles.css)

**🔄 Prueba rápida local**

1. Ejecuta el proyecto:

python app.py

1. Entra a http://localhost:5000/
2. Usa email: usuario@example.com y clave 123456

**🔹 6. Pruebas locales**

* Ejecutar servidor local Flask (Comando: python app.py)
* Probar login con usuario válido (Navegador)
* Ver anuncio tras autenticación (Navegador)
* Hacer clic y confirmar registro del clic (Base de datos)
* Verificar redirección o página de éxito (Navegador)
* Verificar que IP, user\_id, ad\_id y timestamp se registran (phpMyAdmin)

**✅ Paso 6: Pruebas locales y validaciones**

Este paso consiste en probar manualmente todo el flujo del sistema para asegurar que:

* La app funciona correctamente.
* La base de datos guarda lo que debe.
* Las páginas HTML cargan bien y son usables.

**🧪 6.1 Ejecutar servidor local Flask**

1. Abre terminal en la raíz del proyecto.
2. Asegúrate de activar el entorno virtual:
3. # Windows
4. venv\Scripts\activate
5. # Mac/Linux
6. source venv/bin/activate
7. Ejecuta la aplicación:
8. python app.py
9. Accede desde tu navegador a:  
   👉 http://localhost:5000/

**🧪 6.2 Probar flujo completo**

**🔐 A. Login**

* Introduce:
  + Email: usuario@example.com
  + Contraseña: 123456
* Verifica:
  + Redirecciona correctamente a la página de anuncio (/ad)
  + Si colocas mal los datos, devuelve error 401

**📢 B. Ver anuncio**

* Verifica que:
  + El anuncio se muestra con su título (Anuncio 1)
  + Hay un botón para “Ver más”

**🖱️ C. Hacer clic en el anuncio**

* Haz clic en el botón.
* Verifica:
  + Redirecciona a /success (o directamente al destino en futuro)
  + Muestra mensaje de confirmación
  + Luego redirige a /logout tras 3 segundos

**📂 6.3 Verifica registros en la base de datos**

Abre tu herramienta MySQL (phpMyAdmin o MySQL Workbench):

**A. Verifica que el clic fue registrado:**

SELECT \* FROM clicks;

Debe aparecer una fila nueva con:

* user\_id (por defecto: 1)
* ad\_id (por defecto: 1)
* ip\_address (por ejemplo: 127.0.0.1)
* timestamp

**🔐 6.4 Verificar seguridad básica**

**A. Protecciones activas**

* Si accedes a /ad sin haber iniciado sesión → redirige a /login
* Si accedes a /click/1 sin estar logueado → redirige a /login

**🧹 6.5 Validaciones visuales**

* Verifica que:
  + El diseño es limpio y centrado
  + Todos los estilos cargan bien
  + El botón no se puede presionar más de una vez (opcional con JS)
  + No hay errores en consola del navegador

**🧪 6.6 Validaciones técnicas (opcional para producción)**

* Usa la herramienta de desarrollador (F12) en navegador:
  + Verifica que no hay errores 404 (CSS, JS).
  + Red de solicitudes (Network) para confirmar respuesta 200 en cada ruta.
* Revisa consola de Flask para:
  + Ver errores o excepciones.
  + Confirmar que las rutas se ejecutan correctamente.

**✅ Resultado del Paso 6**

Ya confirmaste que:

* El sistema de login funciona.
* Se muestra un anuncio tras el login.
* Los clics se registran en MySQL.
* Hay protección de rutas.
* Todo corre en localhost correctamente.

**✅ Paso 7: Preparar para producción en SiteGround**

Este paso abarca desde la exportación de archivos y base de datos, hasta la configuración del entorno Flask y la conexión a la base de datos MySQL remota de SiteGround.

**🌐 7.1 Crear base de datos en SiteGround**

1. Accede a tu cuenta en SiteGround.
2. Ve a **Site Tools > Site > MySQL**.
3. Crea una nueva base de datos:
   * Nombre: ppc\_portal
4. Crea un nuevo usuario MySQL y asigna contraseña.
5. Asigna **todos los privilegios** al nuevo usuario para esa base de datos.
6. Anota:
   * Host (ej: xxx.siteground.net)
   * Nombre de BD, usuario y contraseña.

**🧾 7.2 Exportar base de datos local e importarla en SiteGround**

**Desde phpMyAdmin local:**

1. Exporta toda la base ppc\_portal como archivo .sql.

**Desde SiteGround:**

1. Ve a **phpMyAdmin** en tu panel SiteGround.
2. Selecciona la nueva base de datos creada.
3. Haz clic en **Importar** > selecciona el .sql exportado.
4. Confirma que las tablas (users, ads, clicks) se importaron.

**📂 7.3 Subir el proyecto al hosting**

**Opción A: Usando el File Manager**

1. En Site Tools > **Site > File Manager**.
2. Ve a la carpeta public\_html/.
3. Crea una subcarpeta si deseas (/ppc\_portal) o úsala directamente.
4. Sube todos los archivos y carpetas del proyecto comprimidos .zip.
5. Extrae el .zip desde el panel.

**Opción B: Usar FTP (con FileZilla)**

1. Configura tu acceso FTP desde Site Tools.
2. Sube todo el contenido a /public\_html/.

**⚙️ 7.4 Configurar config.py para producción**

📄 En ppc\_portal/config.py:

MYSQL\_HOST = 'tu\_host\_mysql.siteground.net' # distinto de localhost

MYSQL\_USER = 'tu\_usuario'

MYSQL\_PASSWORD = 'tu\_contraseña'

MYSQL\_DB = 'ppc\_portal'

# Usa una clave fija para producción

SECRET\_KEY = 'clave\_secreta\_segura'

**🔧 7.5 Configurar entorno Python en SiteGround**

Solo disponible en hosting **con soporte para Python (Startup, GrowBig o superior)**

1. Ve a **Dev > App Installer > Python App** en SiteGround.
2. Crea un nuevo entorno Python:
   * Version: 3.x
   * Carpeta raíz: /public\_html/ppc\_portal
3. En **"Startup File"**, escribe:
4. app:app

(Significa: desde app.py, usar la variable app)

1. Instala tus paquetes:
   * En terminal o SSH:
   * pip install flask flask-mysqldb
2. Si usas .htaccess, redirige al portal:
3. RewriteEngine On
4. RewriteRule ^$ http://proyecto.com/login [R,L]

**✅ 7.6 Prueba en línea**

1. Accede a:
2. https://proyecto.com/login
3. Ingresa:
   * Email: usuario@example.com
   * Password: 123456
4. Verifica que:
   * Muestra el anuncio correctamente.
   * El clic se registra en base de datos.
   * Todo funciona como en local.

**📦 7.7 Checklist final de producción**

* ¿El sitio carga por HTTPS correctamente?
* ¿Login y logout funcionan?
* ¿Se registra el clic en la base de datos remota?
* ¿Redirige al éxito correctamente?
* ¿Las rutas están protegidas?
* ¿Has probado desde diferentes dispositivos?

**✅ Resultado del Paso 7**

Ya tienes el sistema:

* Funcionando en línea en proyecto.com
* Conectado a base de datos MySQL remota
* Ejecutando Python/Flask desde SiteGround
* Totalmente accesible para usuarios reales

**✅ Paso 8: Validaciones finales**

**🔁 8.1 Flujo completo de usuario (End-to-End Test)**

Simula el comportamiento de un visitante desde cualquier navegador:

1. Accede a:  
   👉 https://proyecto.com/login
2. Inicia sesión con:
   * Usuario: usuario@example.com
   * Contraseña: 123456
3. Verifica:
   * Redirección al anuncio /ad
   * Título del anuncio y botón se cargan
   * Al hacer clic, se redirige a /success
   * Luego al cerrar sesión automáticamente (/logout)

**🛡️ 8.2 Validar seguridad básica**

* **Protección de rutas**: si accedes directamente a /ad o /click sin estar logueado, debe redirigir a /login.
* **Sesiones**: al cerrar sesión (/logout) se destruye la sesión correctamente.
* **IP tracking**: cada clic debe registrar la IP del usuario (request.remote\_addr).
* **XSS y SQLi**: evita mostrar datos sin escapar en HTML (Flask lo hace por defecto).
* **Contraseñas**: por ahora están en texto plano. En próximas mejoras → usar hash (werkzeug.security.generate\_password\_hash()).

**📊 8.3 Validar base de datos**

Conéctate a phpMyAdmin o MySQL remoto (en SiteGround) y ejecuta:

SELECT \* FROM users;

SELECT \* FROM ads;

SELECT \* FROM clicks;

Confirma que:

* El usuario está registrado.
* El anuncio existe.
* El clic quedó guardado con:
  + user\_id
  + ad\_id
  + ip\_address
  + timestamp

**🔍 8.4 Validar comportamiento técnico**

* Verifica que no haya errores 404 o 500.
* Asegúrate de que no hay conflictos de rutas.
* Revisa la consola de desarrollador (F12):
  + En la pestaña "Network" y "Console"
* Los botones no deben poder usarse múltiples veces rápidamente (opcional: usar JS para deshabilitar tras clic).

**🧪 8.5 Validar desde múltiples dispositivos**

Prueba el sistema desde:

* Laptop/PC
* Teléfono móvil
* Tableta

Verifica:

* ✅ El diseño se adapta bien (estilo responsivo básico)
* ✅ La lógica del clic se ejecuta igual
* ✅ El navegador guarda cookies/sesión

**🧠 8.6 Pruebas de estrés mínimas (manual)**

1. Intenta abrir múltiples pestañas y hacer login desde cada una.
2. Intenta hacer múltiples clics al mismo anuncio.
3. Intenta borrar los cookies y volver a entrar.
4. Verifica si el sistema sigue funcionando de forma esperada.

**✅ 8.7 Checklist final del sistema en producción**

| **Ítem** | **Estado** |
| --- | --- |
| Login funcional con sesión | ✅ |
| Anuncio se muestra tras login | ✅ |
| Clic registra IP, ID y timestamp | ✅ |
| Logout destruye sesión | ✅ |
| Base de datos conectada y funcional | ✅ |
| Sistema visible en dominio público | ✅ |
| Archivos CSS cargan correctamente | ✅ |
| HTML limpio y sin errores visuales | ✅ |
| Redirecciones funcionan correctamente | ✅ |
| Base de datos protegida y no pública | ✅ |

**🚀 Sistema en línea 100% funcional**

Con esto concluye la versión inicial de tu sistema prototipo en Flask con publicidad PPC y autenticación.