

Flask

1. Instalación

2. Aplicación básica Flask

3. Rutas con parámetros

4. Usar render_template()

5. Enviar datos al HTML con Jinja

6. Servir archivos estáticos

7. Jinja

8. Ejemplo completo



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



Flask

¿Qué es Flask?

Flask es un **framework ligero** para crear aplicaciones web en Python. Es ligero y flexible, ideal para proyectos pequeños o medianos. Te permite crear aplicaciones web de manera rápida y sencilla.

1. Instalación de Flask

Antes de empezar, asegúrate de tener **Python** instalado. Luego, instala Flask con el siguiente comando:

```
pip install flask
```

Para verificar que Flask se instaló correctamente, puedes ejecutar:

```
python -c "import flask; print(flask.__version__)"
```

Instalamos también el conector para mysql:

```
pip install mysqlclient
```

Estructura básica de una aplicación Flask:

La estructura típica de un proyecto Flask es algo así:

```
mi_proyecto/
├── app.py          # Archivo principal de la aplicación Flask
├── templates/      # Carpeta donde van las plantillas HTML
├── static/         # Carpeta para archivos estáticos (CSS, JS, imágenes)
└── db/             # (Opcional) Carpeta para manejo de bases de datos
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



2. Crear una Aplicación Flask Básica

Vamos a crear un "**Hola, Mundo**" con Flask.

🔧 Estructura del proyecto

```
mi_proyecto/
|--- app.py # Archivo principal de Flask
```

Código de app.py

```
from flask import Flask

# Crear la aplicación Flask
app = Flask(__name__)

# Definir una ruta (URL) y su función asociada
@app.route('/')
def home():
    return "¡Hola, Mundo desde Flask!"

# Ejecutar la aplicación
if __name__ == '__main__':
    app.run(debug=True)
```

◊ Explicación:

- **Flask(__name__):** Crea una instancia de Flask, es decir crea la aplicación.
- **@app.route('/')**: Define la ruta de la página principal ("").
- **app.run(debug=True)**: Inicia el servidor en modo **debug** para recargar cambios automáticamente.

🔧 Para ejecutar:

```
python app.py
```

Y luego, abre en tu navegador: **<http://127.0.0.1:5000/>**



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



3. Rutas con Parámetros

Podemos definir rutas que acepten parámetros en la URL.

❖ Ejemplo con parámetros en la URL

```
@app.route('/usuario/<nombre>')
def mostrar_usuario(nombre):
    return f"Hola, {nombre}!"
```

Hola, Ana!

Si accedes a <http://127.0.0.1:5000/usuario/Ana>, verás:

También podemos especificar el tipo de dato del parámetro:

```
@app.route('/numero/<int:num>')
def doble(num):
    return f"El doble de {num} es {num * 2}"
```

<http://127.0.0.1:5000/numero/5> devolverá

El doble de 5 es 10

Devolviendo HTML

Flask permite devolver **HTML** en las respuestas.

```
@app.route('/html')
def pagina_html():
    return """
<html>
    <head><title>Mi Página</title></head>
    <body>
        <h1>¡Bienvenido a Flask!</h1>
        <p>Esto es una página HTML generada con Flask.</p>
    </body>
</html>
"""
```

Al entrar a <http://127.0.0.1:5000/html>, verás una página con un **h1** y un **p**.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



4. Usar `render_template()` con HTML externo

Para separar la lógica de la presentación, Flask permite usar **archivos HTML** en la **carpeta templates**.

🔗 Estructura del proyecto

```
mi_proyecto/
|__ app.py
|__ templates/
|   |__ index.html
```

📌 Código de index.html (en templates/)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Página en Flask</title>
</head>
<body>
    <h1>Bienvenido a mi página con Flask</h1>
</body>
</html>
```

📌 Código de app.py para renderizar el HTML

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/pagina')
def pagina():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Ahora, al visitar <http://127.0.0.1:5000/pagina>, Flask renderizará index.html.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



5.Enviar Datos al HTML con **Jinja**

Flask usa **Jinja**, un motor de plantillas, para pasar variables a las páginas HTML. Jinja para generar HTML dinámico. Puedes pasar datos del backend a las plantillas con variables en Jinja.

❖ Código de app.py con variables

```
@app.route('/saludo/<nombre>')
def saludo(nombre):
    return render_template('saludo.html', nombre=nombre)
```

❖ Código de templates/saludo.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Saludo</title>
</head>
<body>
    <h1>¡Hola, {{ nombre }}!</h1>
</body>
</html>
```

Si accedes a <http://127.0.0.1:5000/saludo/Ana>, verás

¡Hola, Ana!



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



6. Formularios y Métodos GET y POST

Flask permite manejar formularios con GET y POST.

🔗 Formulario en formulario.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Formulario</title>
</head>
<body>
    <h1>Introduce tu nombre:</h1>
    <form action="/procesar" method="post">
        <input type="text" name="nombre" required>
        <button type="submit">Enviar</button>
    </form>
</body>
</html>
```

🔗 Código en app.py para procesar el formulario

```
from flask import request

@app.route('/formulario')
def formulario():
    return render_template('formulario.html')

@app.route('/procesar', methods=['POST'])
def procesar():
    nombre = request.form['nombre']
    return f"¡Hola, {nombre}!"
```

- **GET:** Muestra el formulario.
- **POST:** Recoge los datos del formulario y los muestra.





6. Servir Archivos Estáticos (CSS, JS, Imágenes)

Para agregar estilos o imágenes, crea una carpeta static/.

💡 Estructura

```
mi_proyecto/
|__ app.py
|__ static/
|   |__ style.css
|__ templates/
|   |__ index.html
```

💡 Ejemplo de style.css (en static/)

```
body {
    background-color: lightblue;
    font-family: Arial, sans-serif;
}
```

💡 Referenciar CSS en index.html

```
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}>
```

Resumen

Concepto	Descripción
Instalar Flask	pip install flask
Crear rutas	@app.route('/')
Parámetros en URL	@app.route('/usuario/<nombre>')
Renderizar HTML	render_template('archivo.html')
Manejar formularios	request.form['campo']
Archivos estáticos	static/ (CSS, imágenes, JS)



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



7. JINJA

Con Flask podemos tomar datos de un formulario en Html, entonces ¿realmente para que nos hace falta Jinja?

Jinja no es estrictamente necesario para recibir datos de un formulario en Flask, pero **facilita la generación de contenido dinámico en las páginas HTML**. Aquí te explico sus principales ventajas:

Jinja no es estrictamente necesario para recibir datos de un formulario en Flask, pero **facilita la generación de contenido dinámico en las páginas HTML**. Aquí te explico sus principales ventajas:

1. Evita escribir HTML en Python

Sin Jinja, tendrías que construir las respuestas HTML en Python con return:

```
@app.route('/')
def index():
    return "<h1>Bienvenido a mi página</h1><p>Esta es una respuesta desde Flask.</p>"
```

Esto es **poco práctico**, especialmente cuando necesitas páginas más complejas.

Con **Jinja**, simplemente creas un archivo HTML en la carpeta templates y lo renderizas:

```
@app.route('/')
def index():
    return render_template('index.html')
```

Y el HTML queda limpio en templates/index.html:

```
<h1>Bienvenido a mi página</h1>
<p>Esta es una respuesta desde Flask.</p>
```

Esto **separa la lógica del servidor y la presentación**, facilitando el mantenimiento.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



2. Permite usar variables en el HTML

Si queremos mostrar un nombre en la página sin Jinja, tendríamos que hacer algo así:

```
@app.route('/')
def index():
    nombre = "Juan"
    return f"<h1>Hola, {nombre}</h1>"
```

Con Jinja, es más limpio:

```
@app.route('/')
def index():
    nombre = "Juan"
    return render_template('index.html', nombre=nombre)
```

Y en el HTML (index.html):

```
<h1>Hola, {{ nombre }}</h1>
```

3. Permite estructuras de control en HTML

Si tienes una lista de alumnos en Python:

```
@app.route('/')
def index():
    alumnos = ["Ana", "Carlos", "Sofía"]
    return render_template('index.html', alumnos=alumnos)
```

En Jinja puedes recorrer la lista en el HTML:

```
<ul>
    {% for alumno in alumnos %}
        <li>{{ alumno }}</li>
    {% endfor %}
</ul>
```

Sin Jinja, tendrías que construir el HTML en Python con concatenaciones de strings, lo cual es engoroso.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



¿Cuándo NO usar Ninja?

Si tu frontend se construye con **React, Vue, Angular o solo JavaScript con fetch()**, no necesitas Ninja porque los datos se manejan con JSON desde Flask.

Conclusión

Jinja facilita la generación de HTML dinámico de forma ordenada, sin necesidad de escribir HTML en Python. Separa la lógica del backend y el frontend, hace que el código sea más limpio y permite reusar plantillas. Es una excelente opción cuando usas **Flask como backend y HTML estático como frontend** sin frameworks JavaScript modernos.

Jinja y Base de datos

Con flask podemos tomar datos de una base de datos y mostrarlos en un html sin usar jinja

Podemos hacerlo de dos formas:

Forma sin Ninja

1 Usar JSON y JavaScript (AJAX con Fetch/Axios)

- Flask devuelve datos en formato JSON.
- En el HTML, usamos **JavaScript** para hacer una petición y mostrar los datos.

2 Renderizar el HTML manualmente en Flask (no recomendado)

- Podemos construir el HTML como una cadena de texto en Python, pero esto es poco práctico y difícil de mantener.

Con Ninja

Jinja nos facilita la integración de datos de la base de datos en el HTML sin necesidad de usar **JavaScript**.



- Comparación: Jinja vs. JSON + JavaScript

Método	¿Cómo funciona?	Ventajas	Desventajas
Jinja (Renderizado en servidor)	Flask obtiene los datos y los "inyecta" directamente en la plantilla HTML antes de enviarla al navegador.	<input checked="" type="checkbox"/> Fácil de usar, no necesitas JavaScript. <input checked="" type="checkbox"/> Se integra directamente en el HTML. <input checked="" type="checkbox"/> Menos peticiones al servidor.	<input checked="" type="checkbox"/> Cada cambio en los datos requiere recargar la página. <input checked="" type="checkbox"/> No es ideal para aplicaciones dinámicas (como dashboards interactivos).
JSON + JavaScript (Fetch/Axios) (Renderizado en cliente)	Flask envía los datos en formato JSON y el navegador los procesa con JavaScript.	<input checked="" type="checkbox"/> Ideal para aplicaciones dinámicas. <input checked="" type="checkbox"/> No necesitas recargar la página para actualizar datos. <input checked="" type="checkbox"/> Funciona bien con frameworks como React/Vue.	<input checked="" type="checkbox"/> Requiere código JavaScript extra. <input checked="" type="checkbox"/> Puede ser más lento si hay muchas peticiones.

Conclusión

- Si quieres simplicidad y rapidez, usa **Jinja**.
- Si necesitas interactividad y datos en tiempo real, usa **JSON + JavaScript (Fetch/Axios)**.

Para aplicaciones simples o informes estáticos, **Jinja es suficiente**. Para aplicaciones dinámicas, **combinar Flask con JavaScript** es mejor

8. Ejemplo completo:

Flujo de trabajo.



Flask (Backend) + Jinja (Plantillas) = Backend + Frontend Básico

- **Flask** es el **backend**, donde se maneja la lógica de la aplicación, la conexión con la base de datos, las rutas y la gestión de solicitudes HTTP.
- **Jinja** es el **motor de plantillas** que Flask usa para generar dinámicamente **HTML** en el servidor. Con Jinja, puedes integrar los datos procesados en el servidor dentro del HTML, lo que te permite mostrar esos datos sin necesidad de usar tecnologías frontend más complejas como JavaScript o frameworks como React.

Flujo general:

1. **Flask** maneja las solicitudes del usuario, interactúa con la base de datos, procesa la lógica de negocio y luego envía los datos necesarios al **frontend** (en este caso, una plantilla HTML) usando **Jinja**.
2. **Jinja** permite integrar los datos del backend en el HTML, usando sintaxis especial dentro de las plantillas para mostrar listas, variables, tomar decisiones condicionales, etc. (como `{% for %}` para iterar sobre listas o `{{ }}` para mostrar variables).
3. En lugar de crear un **frontend dinámico** separado con JavaScript o un framework de frontend, **Jinja** te permite tener un frontend básico y dinámico directamente dentro de las plantillas HTML del servidor.



Resumen:

1. **Flask** se encarga de la lógica del backend: **Backend (Flask)** recupera los datos (en app.py).
2. Flask pasa esos datos a una **plantilla HTML** usando **Jinja**.
3. **Jinja** procesa esos datos y genera el HTML final, que es enviado al cliente (navegador).

Ventaja:

Usar **Flask + Jinja** es ideal para *proyectos más simples* o donde no necesitas un **frontend avanzado**. Te permite construir aplicaciones con un mínimo de código y esfuerzo, sin tener que aprender y trabajar con tecnologías frontend complejas.

Es una excelente opción cuando se busca simplicidad y rapidez en la creación de aplicaciones web.

Tabla resumen de Ninja:

Estructura	Descripción	Ejemplo en Ninja
Mostrar datos	Mostrar una variable o valor en el HTML.	<code>{{ variable }}</code> Ejemplo: {{ nombre }} muestra el valor de la variable nombre.
Condicional IF	Condicional básico: se ejecuta si la condición es verdadera.	<code>{% if condicion %}...{% endif %}</code> Ejemplo: {% if edad >= 18 %} Eres mayor de edad {% endif %}
Condicional IF-ELSE	Condicional con una rama alternativa si la condición es falsa.	<code>{% if condicion %}...{% else %}...{% endif %}</code> Ejemplo: {% if edad >= 18 %} Adulto {% else %} Menor {% endif %}
Condicional IF-ELIF-ELSE	Condicional con múltiples opciones, similar a un "if-else if-else" en otros lenguajes.	<code>{% if condicion1 %}...{% elif condicion2 %}...{% else %}...{% endif %}</code> Ejemplo: {% if edad >= 18 %} Adulto {% elif edad >= 12 %} Adolescente {% else %} Niño {% endif %}
Bucle FOR	Bucle para iterar sobre una lista o un rango.	<code>{% for item in lista %}...{% endfor %}</code> Ejemplo: {% for nombre in nombres %}<p>{{ nombre }}</p>{% endfor %}
Bucle FOR con índice	Bucle con índice para acceder a la posición de los elementos.	<code>{% for item in lista %} {{ loop.index }}: {{ item }} {% endfor %}</code> Ejemplo: {% for i, nombre in enumerate(nombres) %} {{ i }}: {{ nombre }} {% endfor %}





Estructura	Descripción	Ejemplo en Jinja
Bucle con rango	Bucle que itera sobre un rango de números.	{% for i in range(1, 5) %}...{% endfor %} Ejemplo: {% for i in range(1, 6) %} <p>{{ i }}</p> {% endfor %}
Verificar si la lista no está vacía	Comprobar si una lista o conjunto tiene elementos.	{% if lista %}...{% endif %} Ejemplo: {% if nombres %} <p>Hay nombres</p> {% else %} <p>No hay nombres</p> {% endif %}
Mostrar el valor de un diccionario	Acceder y mostrar un valor de un diccionario usando claves.	{{ diccionario['clave'] }} Ejemplo: {{ persona['nombre'] }} muestra el valor asociado a 'nombre'.
Condicional con is	Condicional para verificar tipos o si una variable es nula.	{% if variable is none %}...{% endif %} Ejemplo: {% if nombre is not none %}<p>Nombre: {{ nombre }}</p> {% endif %}

Paso a paso:

Paso a paso para hacer una web simple con **Flask, Jinja y MySQL (en XAMPP)** que muestra un listado de alumnos y permite ver los detalles de un alumno seleccionado.

1. Instalación de Flask y Jinja

Antes de empezar, instala Flask y la librería para conectar con MySQL (usaremos [pymysql](#)):

```
pip install flask pymysql
```

2. Crear la base de datos en MySQL (XAMPP)

1. Abre XAMPP y asegúrate de que Apache y MySQL están corriendo.
2. Accede a phpMyAdmin (<http://localhost/phpmyadmin/>)
3. Ejecuta el siguiente SQL para crear la base de datos y la tabla de alumnos:





Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



```

CREATE DATABASE escuela;
USE escuela;

CREATE TABLE alumnos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    apellidos VARCHAR(50),
    genero ENUM('M', 'F'),
    edad INT,
    curso VARCHAR(20)
);

INSERT INTO alumnos (nombre, apellidos, genero, edad, curso) VALUES
('Juan', 'Pérez', 'M', 16, '10º'),
('María', 'Gómez', 'F', 15, '9º'),
('Carlos', 'López', 'M', 17, '11º');

```

O bien importa el archivo sql directamente usando PhpMyAdmin:

- Abre phpMyAdmin.
- Ve a la pestaña "**Importar**".
- Carga el archivo alumnos.sql.
- Haz clic en "**Continuar**".



alumnos.sql.txt

3.Estructura del Proyecto

Creamos la siguiente estructura de carpetas para nuestro proyecto.

```

mi_proyecto/
├── app.py          # Código principal de Flask
├── config.py       # Configuración de la base de datos
├── db/
│   └── conexion.py # Conexión con MySQL
└── templates/
    ├── index.html  # Página principal
    └── detalles.html # Página de detalles del alumno

```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Junta de Andalucía
Instituto Andaluz del
Deporte y la Juventud
www.andaluzdeldeporte.es



Ayuda
en Acción
Impulsa Empleo Joven

4. Configuración de la Base de Datos ([config.py](#))

Crea un archivo config.py en la raíz del proyecto con los datos de conexión a MySQL:

```
MYSQL_HOST = 'localhost'
MYSQL_USER = 'root' # Usuario por defecto en XAMPP
MYSQL_PASSWORD = '' # Deja vacío si no tienes contraseña en MySQL
MYSQL_DB = 'escuela'
```



config.py.txt

5. Archivo de Conexión ([db/conexion.py](#))

Este archivo se encargará de conectar con la base de datos.

```
import pymysql
import config

def conectar():
    return pymysql.connect(
        host=config.MYSQL_HOST,
        user=config.MYSQL_USER,
        password=config.MYSQL_PASSWORD,
        database=config.MYSQL_DB,
        cursorclass=pymysql.cursors.DictCursor # Para recibir los datos en formato diccionario
    )
```



conexion.py.txt



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



6.Código Principal de Flask (app.py)

Este archivo define las rutas y la lógica de la aplicación. (servidor Flask)

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

app = Flask(__name__)

# Función para conectarnos a la base de datos
def conectar_bd():
    return pymysql.connect(
        host="localhost",
        user="root",           # Cambia esto si tu usuario es diferente
        password="",            # Pon la contraseña si tiene
        database="escuela",
        cursorclass=pymysql.cursors.DictCursor
    )

# Ruta principal: muestra el listado de alumnos
@app.route('/')
def index():
    conexion = conectar_bd()
    with conexion.cursor() as cursor:
        cursor.execute("SELECT id, nombre FROM alumnos") # Solo nombre e id
        alumnos = cursor.fetchall()
    conexion.close()

    return render_template("index.html", alumnos=alumnos)

# Ruta para mostrar los detalles de un alumno
@app.route('/ver_alumno', methods=['POST'])
def ver_alumno():
    alumno_id = request.form.get("alumno_id") # Obtiene el id del formulario
    conexion = conectar_bd()
    with conexion.cursor() as cursor:
        cursor.execute("SELECT * FROM alumnos WHERE id = %s", (alumno_id,))
        alumno = cursor.fetchone()
    conexion.close()

    return render_template("detalle.html", alumno=alumno)

if __name__ == '__main__':
    app.run(debug=True)
```



app.py.txt

Aquí tenemos dos rutas, la principal “/”, mandamos a index.html nombre e id de todos los alumnos, con “/ver_alumno” tomamos los datos del alumno seleccionado y los enviamos a detalle.html para ser mostrados.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



7. Plantilla Principal.

[templates/index.html](#)

Muestra la lista de alumnos y un botón para ver detalles.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Lista de Alumnos</title>
</head>
<body>
    <h1>Lista de Alumnos</h1>

    <form action="/ver_alumno" method="post">
        <label for="alumno">Selecciona un alumno:</label>
        <select name="alumno_id" id="alumno">
            {% for alumno in alumnos %}
                <option value="{{ alumno.id }}>{{ alumno.nombre }}</option>
            {% endfor %}
        </select>
        <button type="submit">Ver</button>
    </form>
</body>
</html>
```



index.html.txt

Lo que hace este HTML:

- Usa un **formulario** para enviar el ID del alumno al servidor.
- Cuando el usuario selecciona un alumno y pulsa "Ver", se envía el formulario con POST a la ruta /ver_alumno.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



Fondos Europeos



Crear la página de detalles del alumno

templates/detalle.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Detalles del Alumno</title>
</head>
<body>
    <h1>Detalles del Alumno</h1>
    <table border="1">
        <tr>
            <th>ID</th>
            <td>{{ alumno.id }}</td>
        </tr>
        <tr>
            <th>Nombre</th>
            <td>{{ alumno.nombre }}</td>
        </tr>
        <tr>
            <th>Apellidos</th>
            <td>{{ alumno.apellidos }}</td>
        </tr>
        <tr>
            <th>Género</th>
            <td>{{ alumno.genero }}</td>
        </tr>
        <tr>
            <th>Edad</th>
            <td>{{ alumno.edad }}</td>
        </tr>
        <tr>
            <th>Curso</th>
            <td>{{ alumno.curso }}</td>
        </tr>
    </table>

    <br>
    <a href="/">Volver</a>
</body>
</html>
```



detalle.html.txt

8. Ejecutar la Aplicación

Guarda todos los archivos y ejecuta el servidor Flask:

python app.py

Luego, abre en tu navegador:

<http://127.0.0.1:5000/>



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG