

Microsoft
Learn

STUDENT AMBASSADOR



Flask y el desarrollo de Aplicaciones Web con *Python*

Por Albert Jhonatan Quisbert Mújica



Agenda

- Introducción a *Flask*
- *Flask* vs Otros Frameworks
- Arquitectura de *Flask*
- Instalación y Configuración Básica
- Rutas y Manejo de Solicitudes
- *Templates* y *Jinja2*
- Formularios en *Flask*
- Bases de Datos y *SQLAlchemy*
- Demostración





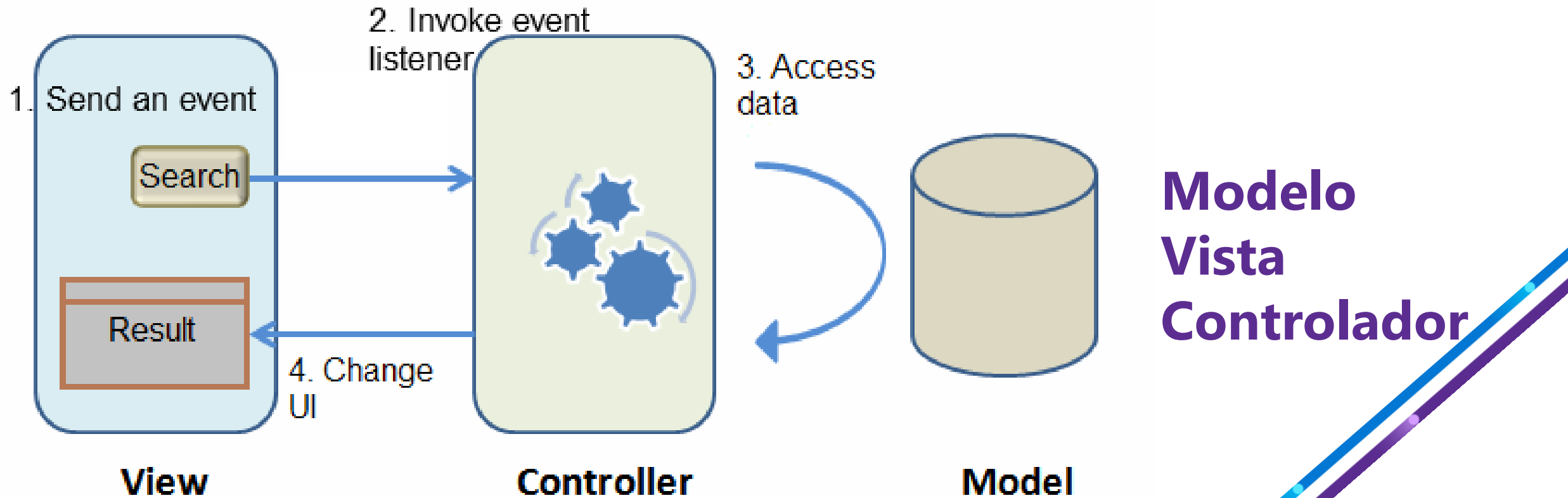
Introducción a *Flask*

¿Qué es *Flask*?

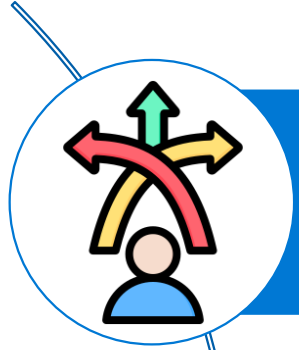


¿Qué es *Flask*?

Flask es un "*micro*" *Framework* para el desarrollo de aplicaciones web en *Python* bajo el patrón **MVC**.



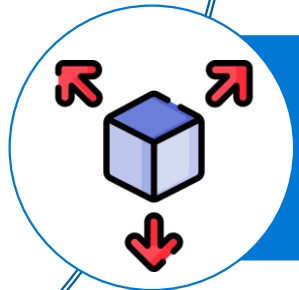
Características



Ligero y flexible.



Facilita el desarrollo rápido de aplicaciones web.



Extensible mediante múltiples módulos y librerías.



Flask vs Otros *Frameworks*

¿Por qué usar *Flask*?



Comparación con otros *frameworks*



Framework
completo con
muchas
funcionalidades
por defecto.



Minimalista, da
mayor control al
desarrollador.

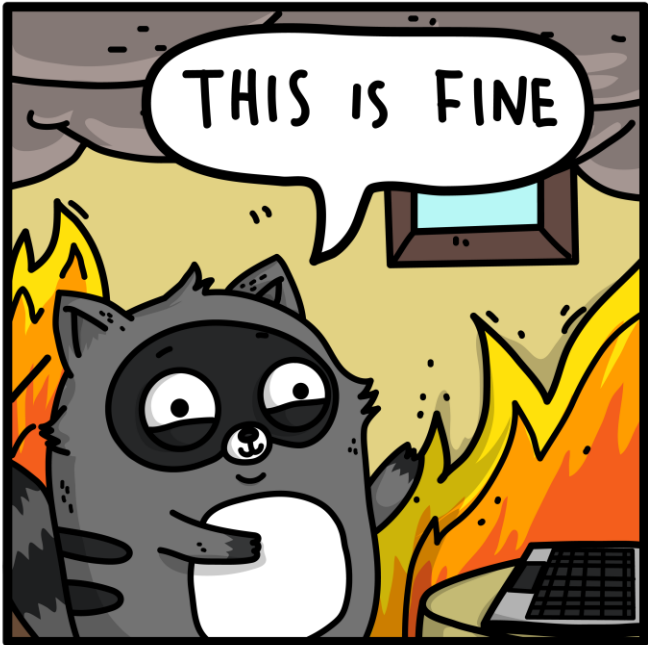
¿Cuándo usar *Flask*?

- Cuando se necesita más flexibilidad.
- Ideal para proyectos pequeños o medianos.

Flask vs Django



Framework	WSGI Framework	Full-stack Framework
Database	SQLAlchemy	Built-in ORM
Flexibility	Complete flexibility	Feature-packed
Design	Minimalistic Design	Batteries Included
Forms	Integrates with WTForms	Uses ModelForms
Template Engine	Jinja2 templating engine	Django Template language (DTL)
Learning curve	Simple to learn	Might take time to learn
Community support	Low community support	Big developer community
Applications	Netflix, Uber, Lyft, Red Hat	Pinterest, Instagram, Accenture





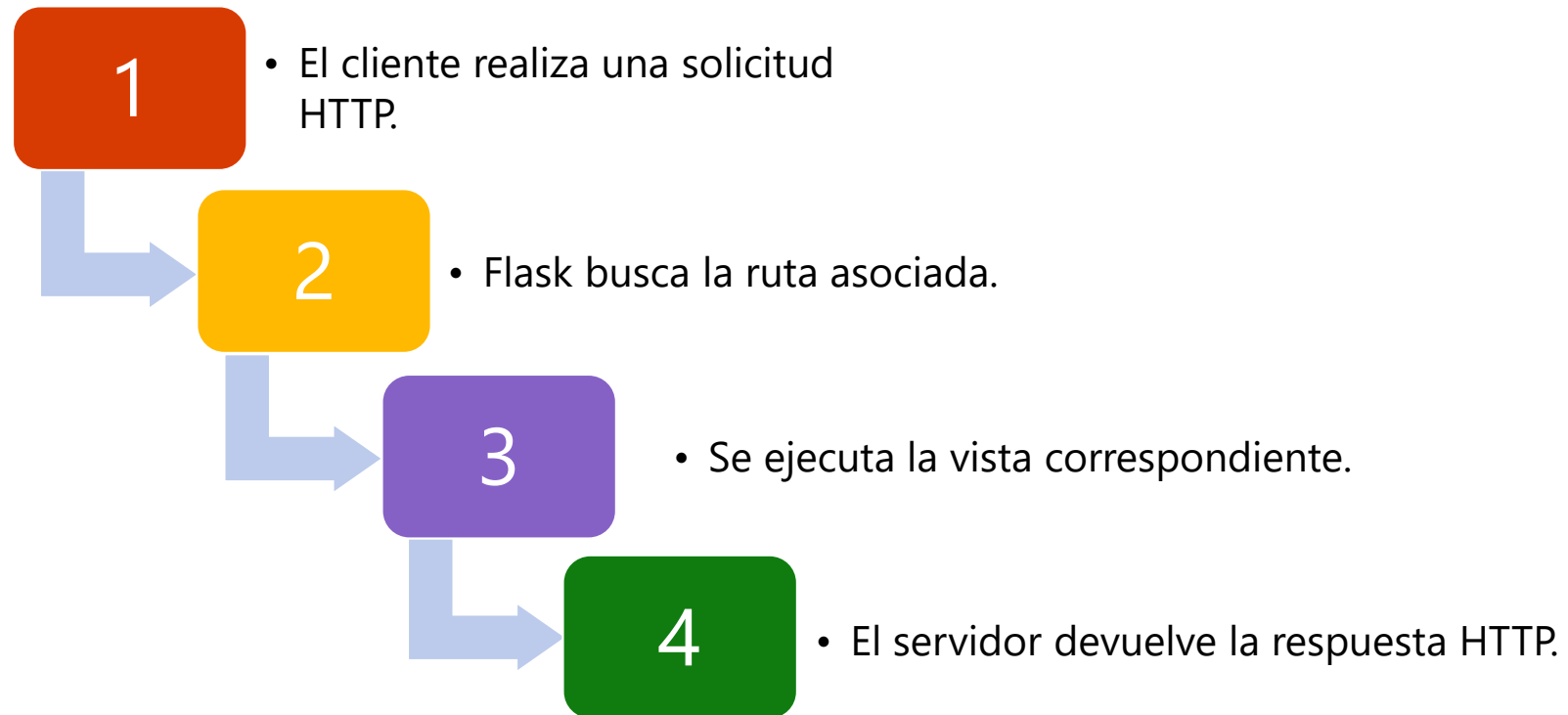
Arquitectura de *Flask*

Arquitectura de una aplicación



Arquitectura de una aplicación *Flask*

- Basado en **WSGI** (Web Server Gateway Interface).
- El ciclo de vida de una solicitud en Flask:





Instalación y Configuración Básica

Instalación de *Flask*



Instalación de *Flask*

Requisitos previos

- *Python* 3.X.
- pip (administrador de paquetes).
- IDE para *Python* o Editor de texto (*VS Code*).



Comando de instalación

`pip install flask`

The screenshot shows the official Flask project page on the Python Package Index (PyPI). The page has a blue header with a search bar and navigation links. The main content area is white and contains the following information:

- Flask 3.0.3**: The current version, marked as the "Versión más reciente" (Latest version) with a green checkmark.
- pip install Flask**: A button to install the package.
- Publicación: 7 abr 2024**: The release date.
- A simple framework for building complex web applications.**: A brief description.
- Navegación**: A sidebar with links to "Descripción de proyecto", "Histórico de versiones", and "Archivos de descarga".
- Descripción de proyecto**: A section titled "Flask" describing it as a lightweight WSGI web application framework.
- Installing**: A section titled "Installing" with instructions on how to install and update the package.

Estructura básica del proyecto *Flask*

```
|— app.py  
|— static/  
|— templates/  
|— venv/
```

app.py

- Archivo principal que contiene la lógica, rutas, métodos, etc.
- Tiene definido el inicio del servidor.

static/

- Carpeta con recursos multimedia, estilos y scripts de javascript que usará el proyecto.

templates/

- Carpeta con plantillas (archivos de HTML) que serán renderizados en el proyecto.

venv/

- Entorno virtual para el proyecto.

Estructura básica del proyecto *Flask*

Código base de *app.py*

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
    return 'Bienvenido a Flask'
if __name__ == '__main__':
    app.run(port=5000, host='127.0.0.1', debug=True)
```

Importación de Flask

Creación de una aplicación de Flask

Definición de rutas

Método para ejecutarse en la ruta

Ejecución del servidor para el proyecto





Rutas y Manejo de Solicitudes

Rutas en *Flask*



Rutas en *Flask*

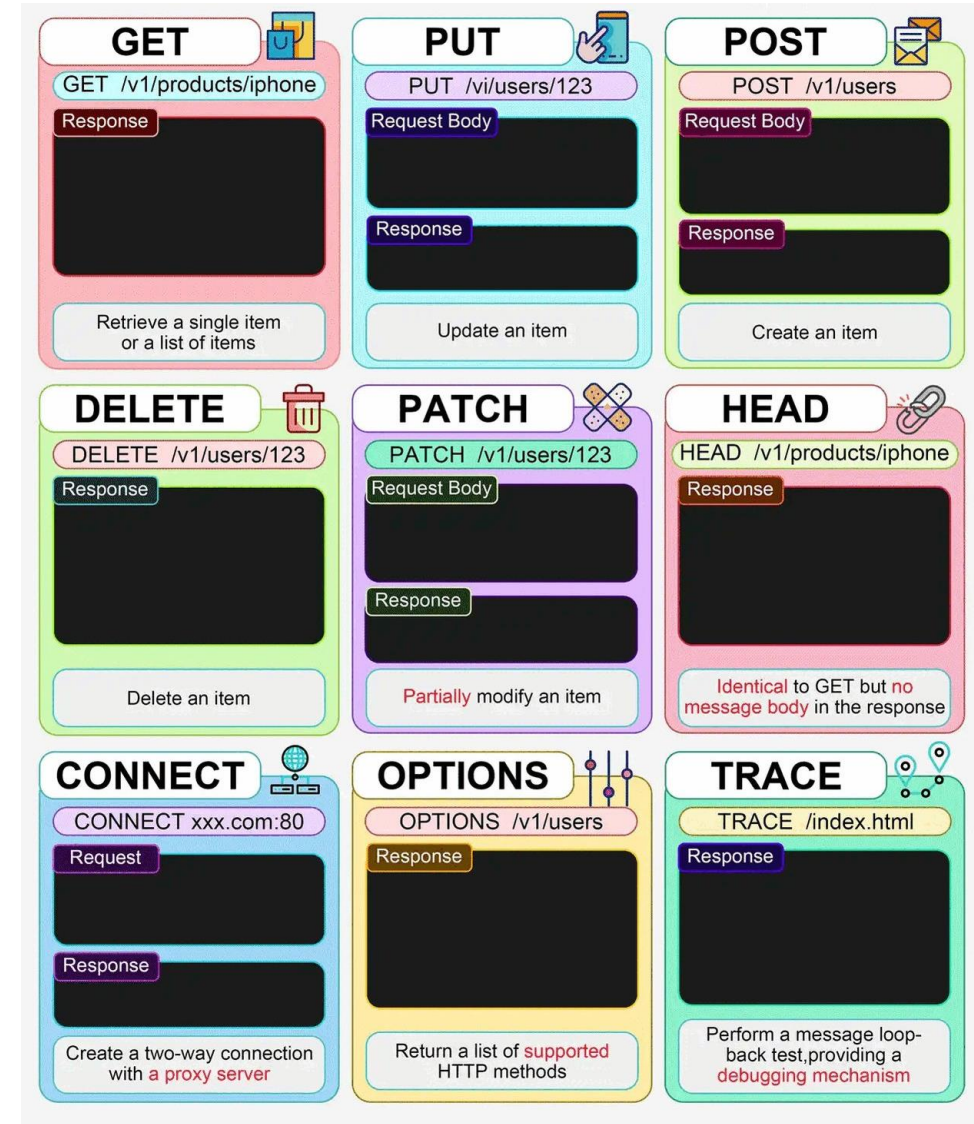
¿Qué son las rutas?

Definen las URLs que la aplicación responde.

Ejemplo de ruta básica

```
@app.route('/')
def index():
    return 'Bienvenido a Flask'
```

Métodos HTTP





Templates y Jinja2

Uso de *Templates*



Motor de plantillas

Flask usa *Jinja2* para manejar plantillas.

Uso de bloques y variables dentro de plantillas.



Jinja es una dependencia se encarga de generar y controlar las plantillas.

Se encarga de pasar **DATOS** por la plantilla para después renderizarla en el documento final.



Motor de plantillas

Ejemplo de renderizado de plantillas:


En app.py

```
@app.route('/')
def index():
    return render_template('index.html',
                           titulo='Página principal',
                           mensaje='Renderizado de plantillas con Jinja2')
```

En templates/index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ titulo }}</title>
</head>
<body>
    <h1>{{ mensaje }}</h1>
</body>
</html>
```

Tipos de datos permitidos en el renderizado

- Cadenas (**str**).
 - Números (**int**, **float**).
 - Booleanos (**bool**).
 - Listas (**list**).
 - Diccionarios (**dict**).
 - Nulos (**None**).
 - Objetos.
 - Funciones.
- 

Motor de plantillas

Estructuras permitidas en plantillas:

- if

```
{% if condicion %}  
    <p>Bienvenido!, {{ user_name }}!</p>  
{% endif %}
```

- if – else

```
{% if condicion %}  
    <p>Bienvenido, {{ user_name }}!</p>  
{% else %}  
    <p>Please log in to continue.</p>  
{% endif %}
```

- if – elif – else

```
{% if user_role == 'admin' %}  
    <p>Bienvenido Admin, {{ user_name }}!</p>  
{% elif user_role == 'editor' %}  
    <p>Bienvenido Editor, {{ user_name }}!</p>  
{% else %}  
    <p>Bienvenido Invitado!</p>  
{% endif %}
```

- for

```
<ul>  
    {% for user in users %}  
        <li>{{ user }}</li>  
    {% endfor %}  
</ul>
```

- for con un índice

```
<ol>  
    {% for user in users %}  
        <li>{{ loop.index }}. {{ user }}</li>  
    {% endfor %}  
</ol>
```

- for sobre diccionarios

```
<ul>  
    {% for key, value in user_data.items() %}  
        <li>{{ key }}: {{ value }}</li>  
    {% endfor %}  
</ul>
```



Formularios en *Flask*

Manejo de Formularios



Manejo de Formularios


Formularios HTML y procesamiento en *Flask*.

Flask-WTF

Dependencia/extensión para manejar y validar formularios.

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired

class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Sign Up')
```

Two parallel diagonal lines, one blue and one purple, with small white dots, extending from the bottom right corner of the slide.



Bases de Datos y *SQLAlchemy*

Flask y Bases de Datos



Flask y Bases de Datos

SQLAlchemy

Dependencia/extensión para trabajar con un ORM en *Flask*.

```
from flask_sqlalchemy import SQLAlchemy
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)
```

Demo

Desarrollemos un pequeño ejemplo



Algunas recomendaciones:



- Usar un entorno virtual por cada proyecto en *Python*.

Comando: `python -m venv [nombre_entorno]`

- Activar el nuevo entorno virtual.

Comando: `.[nombre_entorno]\Scripts\activate`

- Instalar todas las dependencias necesarias después de iniciar el entorno virtual (usar el comando ***pip install***).

- ☐ *Flask* (<https://pypi.org/project/Flask/>).

- ☐ *Bootstrap-Flask* (<https://pypi.org/project/Bootstrap-Flask/>).

- ☐ *Flask-SQLAlchemy* (<https://pypi.org/project/Flask-SQLAlchemy/>).

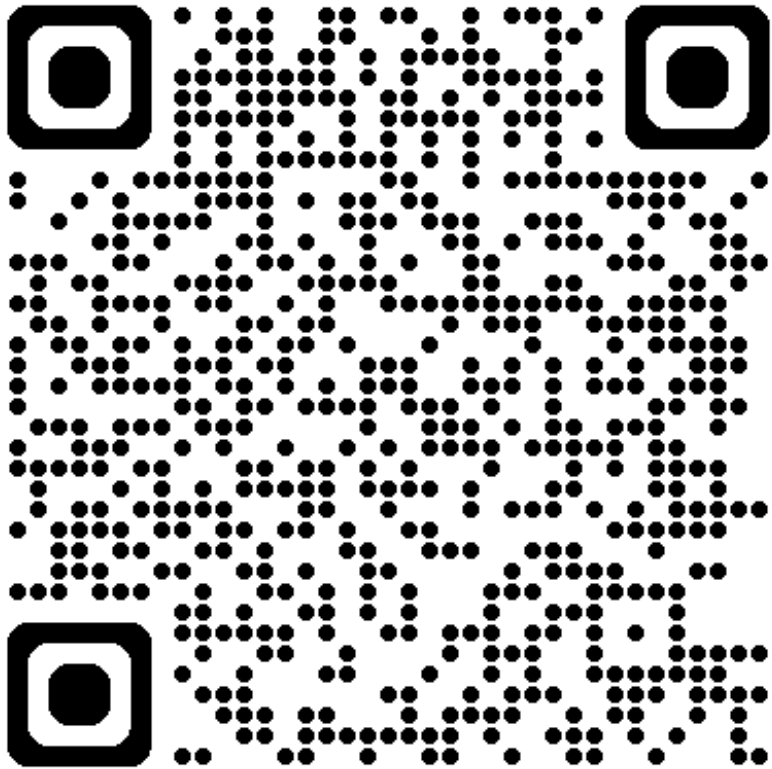
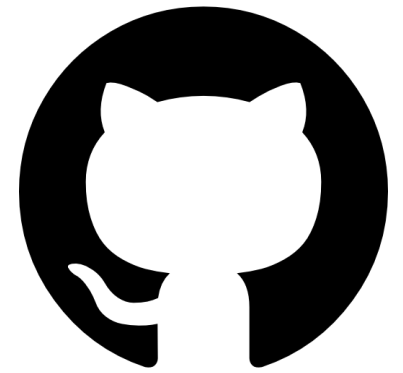
- ☐ *Flask-WTF* (<https://pypi.org/project/Flask-WTF/>).

Extensiones para VS Code

- Auto Rename Tag.
- Auto Close Tag.
- Error Lens.
- flask-snippets.
- indent-rainbow.
- Jinja.
- Prettier.
- Python (mucho muy importante!).
- SQLite Viewer.



Repositorio del proyecto en GitHub

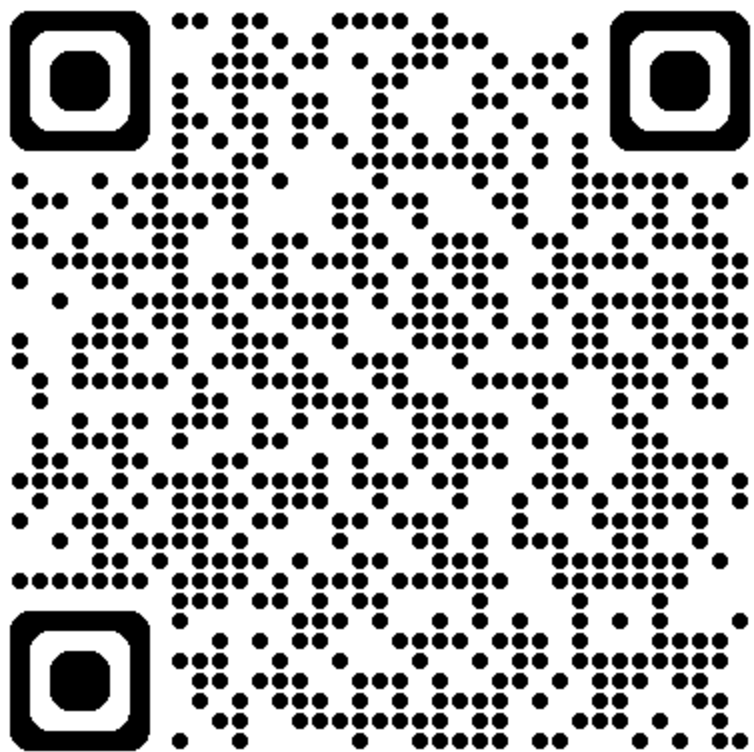


Pasos para replicar el proyecto:

1. Clonar/Descargar el contenido del repositorio en un directorio.
2. Con *VS Code*, abrir el directorio con todos los elementos extraídos.
3. Crear y activar un entorno virtual.
4. Instalar todas las dependencias que se encuentran en el archivo [*requeriments.txt*](#).

Comando: *pip install -r requeriments.txt*

5. ¡Ejecutar el proyecto!



Responda nuestra encuesta de dos preguntas

Puedes contactarme en:



Albert Jhonatan Quisbert Mújica



Albert Jhonatan Quisbert Mújica



@tkey128



AlbertJQM



Google Developer Groups
El Alto



¡GRACIAS POR TU ATENCIÓN!

