

# INFORME PROYECTO RUEDA DE LA VIDA

MICHELLE RAMIREZ AGUDELO

YOSKAR ORDOSGOITTI IDLER

SEBASTIAN ARANGO TABORDA

MEDELLIN ANTIOQUIA

SENA (CTMA)

2025

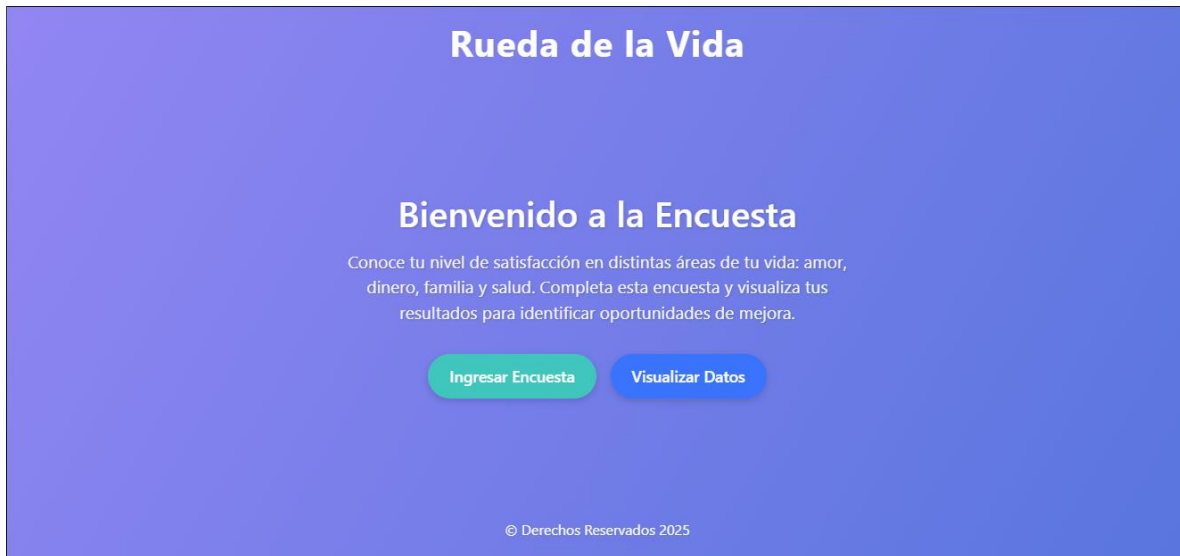
## INTRODUCCION

El presente informe detalla de manera integral el proceso de desarrollo del proyecto *La Rueda de la Vida*, una iniciativa que fusiona la tecnología y el autoconocimiento para evaluar aspectos fundamentales de la existencia humana. En esta versión del proyecto, se han incorporado nuevas funcionalidades a los formularios interactivos, permitiendo ahora el almacenamiento y gestión de datos a través de una base de datos diseñada específicamente para este propósito. Se documenta la creación de esta base de datos, la definición de sus campos y su integración con los formularios desarrollados con HTML, Bootstrap y CSS.

Además, se han implementado nuevos formularios para mejorar la experiencia del usuario: un formulario de bienvenida que direcciona a la encuesta y a los resultados, un formulario de resultados que muestra la evaluación obtenida y un formulario de despedida que almacena la información final en la base de datos. Todo el desarrollo se ha llevado a cabo dentro de un entorno virtual, optimizando la organización y ejecución del proyecto.

## DESARROLLO

### Formulario main.html



Se agrega un nuevo formulario de presentación que nos muestra un mensaje de bienvenida a la encuesta teniendo las opciones de dirigirnos a las encuestas o a ver los resultados de las personas que han hecho la encuesta.

### Código formulario main.html

```
122 <!-- Sección principal estilo "hero" -->
123 <div class="hero">
124   <h2>Bienvenido a la Encuesta</h2>
125   <p>
126     Conoce tu nivel de satisfacción en distintas áreas de tu vida: amor, dinero, familia y salud.
127     Completa esta encuesta y visualiza tus resultados para identificar oportunidades de mejora.
128   </p>
129   <div class="btn-container">
130     <a href="{{ url_for('personal') }}" class="btn btn-custom btn-ingresar">Ingresar Encuesta</a>
131     <a href="{{ url_for('ver_resultados') }}" class="btn btn-custom btn-visualizar">Visualizar Datos</a>
132   </div>
133 </div>
134
135 <!-- Footer -->
136 <footer>
137   <p>&copy; Derechos Reservados 2025</p>
138 </footer>
139
140 <!-- Bootstrap JS -->
141 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
142 </body>
143 </html>
144
```

## Formulario final FormFinal.html

### Encuesta Finalizada

Presiona el botón para guardar tus respuestas.

Guardar Encuesta

Volver al Menú Principal

Este nuevo formulario agrega muestre dos botones luego de finalizar con la encuesta, estos dos botones sirven para guardar la encuesta y volver al menú principal

## Código FormFinal.html

```
rueda-de-la-vida > venv > Código > templates > FormFinal.html > ...
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Final - Guardar Encuesta</title>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
7 </head>
8 <body>
9   <div class="container text-center my-5">
10    <h1>Encuesta Finalizada</h1>
11    <p>Presiona el botón para guardar tus respuestas.</p>
12    <form method="POST" action="{{ url_for('form_final') }}">
13      <button type="submit" class="btn btn-primary btn-lg">Guardar Encuesta</button>
14    </form>
15    <br>
16    <a href="{{ url_for('main') }}" class="btn btn-secondary">Volver al Menú Principal</a>
17  </div>
18  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
19 </body>
20 </html>
```

Resultados Guardados								
ID	Nombre	Edad	Departamento	Amor	Dinero	Familia	Salud	Acciones
1	sebastian arango	26	sistemas	14.499999999999998%	16.0%	14.499999999999998%	14.000000000000002%	Eliminar
2	carlos mora	31	contabilidad	12.5%	14.499999999999998%	16.0%	15.0%	Eliminar
3	karen morales	51	comercial	13.0%	14.000000000000002%	14.499999999999998%	14.000000000000002%	Eliminar
4	yoskar	25	antioquia	17.0%	15.0%	15.5%	12.0%	Eliminar

Volver al Menú Principal

Este formulario muestra los datos de las personas que realizaron la encuesta incluyendo su resultado en la encuesta también cuenta con un botón de eliminar y la opción de redirigirse al menú principal.

## Código resultados.html

```
26 <body>
27
28 <div class="container">
29   <h1 class="text-center">Resultados Guardados</h1>
30   <table class="table table-striped table-bordered table-hover">
31     <thead class="table-dark">
32       <tr>
33         <th>ID</th>
34         <th>Nombre</th>
35         <th>Edad</th>
36         <th>Departamento</th>
37         <th>Amor</th>
38         <th>Dinero</th>
39         <th>Familia</th>
40         <th>Salud</th>
41         <th>Acciones</th>
42       </tr>
43     </thead>
44     <tbody>
45       {% if respuestas %}
46       {% for r in respuestas %}
47       <tr>
48         <td>{{ r.id }}</td>
49         <td>{{ r.nombre_completo }}</td>
```

## Función convertir\_a\_numero

```
33
34 Tabnine | Edit | Test | Explain | Document
35 def convertir_a_numeros(lista):
36     """Convierte una lista de strings en una lista de enteros."""
37     return [int(x) for x in lista if x and x.isdigit()]
```

Esta función convierte una lista de cadenas de texto en una lista de números enteros. Para ello, verifica que cada elemento de la lista contenga únicamente dígitos antes de convertirlo en entero. Finalmente, retorna una nueva lista con los valores convertidos.

## Función calcular porcentaje

```

Tabnine | Edit | Test | Explain | Document
def calcular_porcentaje(lista):
    """Calcula el porcentaje de una categoría."""
    if not lista:
        return 0
    return (sum(lista) / 50) * 25 # 50 es el máximo puntaje posible

```

Esta función calcula el porcentaje de una categoría a partir de los valores de una lista. Primero, verifica si la lista está vacía y, en ese caso, retorna 0. Luego, suma los valores de la lista y los escala en función de un puntaje máximo de 50, devolviendo el resultado como porcentaje.

### Función procesar\_datos

```

Tabnine | Edit | Test | Explain | Document
def procesar_datos():
    """Convierte respuestas a números y calcula los porcentajes."""
    global respuestas_amor, respuestas_dinero, respuestas_familia, respuestas_salud
    global porcentaje_amor, porcentaje_dinero, porcentaje_familia, porcentaje_salud

    respuestas_amor = convertir_a_numeros(respuestas_amor)
    respuestas_dinero = convertir_a_numeros(respuestas_dinero)
    respuestas_familia = convertir_a_numeros(respuestas_familia)
    respuestas_salud = convertir_a_numeros(respuestas_salud)

    porcentaje_amor = calcular_porcentaje(respuestas_amor)
    porcentaje_dinero = calcular_porcentaje(respuestas_dinero)
    porcentaje_familia = calcular_porcentaje(respuestas_familia)
    porcentaje_salud = calcular_porcentaje(respuestas_salud)

```

Convierte las respuestas almacenadas en listas de números y calcula los porcentajes correspondientes.

Función main

```
0 @app.route("/")
1 def main():
2     return render_template("Main.html")
```

Muestra la página principal de la aplicación web.

Funciona personal

```
66 def personal():
67     global global_nombre, global_edad, global_departamento
68     if request.method == "POST":
69         global_nombre = request.form["nombre"]
70         global_edad = int(request.form["edad"])
71         global_departamento = request.form["departamento"]
72         print(f"Datos personales: {global_nombre}, {global_edad}, {global_departamento}")
73         return redirect(url_for("form_amor"))
74     return render_template("FormInfoPersonal.html")
```

Recibe y almacena los datos personales ingresados en el formulario, luego redirige al formulario de la categoría “amor”.

Funciona form\_amor

```
Tabnine | Edit | Test | Explain | Document
6 @app.route("/amor", methods=["GET", "POST"])
7 def form_amor():
8     global respuestas_amor
9     if request.method == "POST":
10         respuestas_amor = [
11             request.form.get("pregunta1"),
12             request.form.get("pregunta2"),
13             request.form.get("pregunta3"),
14             request.form.get("pregunta4"),
15             request.form.get("pregunta5")
16         ]
17         return redirect(url_for("form_dinero"))
18     return render_template("FormAmor.html")
19
```

Guarda las respuestas del formulario de la categoría “amor” y redirige al formulario de la categoría “dinero”.



### Función form\_dinero

```
Tabnine | Edit | Test | Explain | Document
0 @app.route("/dinero", methods=["GET", "POST"])
1 def form_dinero():
2     global respuestas_dinero
3     if request.method == "POST":
4         respuestas_dinero = [
5             request.form.get("pregunta1"),
6             request.form.get("pregunta2"),
7             request.form.get("pregunta3"),
8             request.form.get("pregunta4"),
9             request.form.get("pregunta5")
10        ]
11        return redirect(url_for("form_familia"))
12    return render_template("FormDinero.html")
13
```

Guarda las respuestas del formulario de la categoría “dinero” y redirige al formulario de la categoría “familia”.

### Función form\_familia

```
Tabnine | Edit | Test | Explain | Document
3
4 @app.route("/familia", methods=["GET", "POST"])
5 def form_familia():
6     global respuestas_familia
7     if request.method == "POST":
8         respuestas_familia = [
9             request.form.get("pregunta1"),
10            request.form.get("pregunta2"),
11            request.form.get("pregunta3"),
12            request.form.get("pregunta4"),
13            request.form.get("pregunta5")
14        ]
15        return redirect(url_for("form_salud"))
16    return render_template("FormFamilia.html")
17
```

Guarda las respuestas del formulario de la categoría “familia” y redirige al formulario de la categoría “salud”.

### Función form\_salud

```
18 @app.route("/salud", methods=["GET", "POST"])
19 def form_salud():
20     global respuestas_salud
21     if request.method == "POST":
22         respuestas_salud = [
23             request.form.get("pregunta1"),
24             request.form.get("pregunta2"),
25             request.form.get("pregunta3"),
26             request.form.get("pregunta4"),
27             request.form.get("pregunta5")
28         ]
29         return redirect(url_for("form_final"))
30     return render_template("FormSalud.html")
31
```

Guarda las respuestas del formulario de la categoría “salud” y redirige a la página final.

### Función form\_final

```
@app.route("/final", methods=["GET", "POST"])
def form_final():
    global global_nombre, global_edad, global_departamento
    if request.method == "POST":
        procesar_datos()
        nueva_respuesta = Respuesta(
            nombre_completo=global_nombre,
            edad=global_edad,
            departamento=global_departamento,
            amor=porcentaje_amor,
            dinero=porcentaje_dinero,
            familia=porcentaje_familia,
            salud=porcentaje_salud
        )
        db.session.add(nueva_respuesta)
        db.session.commit()
        print("✅ Respuesta guardada en la base de datos.")
        # Tras guardar, redirigimos al menú principal
        return redirect(url_for("main"))
    return render_template("FormFinal.html")
```

Procesa y almacena los datos recopilados de los formularios en la base de datos. Luego, redirige al menú principal.

### Función ver\_resultados

```
156 def ver_resultados():
157     respuestas = Respuesta.query.all()
158     print(f"Se encontraron {len(respuestas)} registro(s) en la base de datos.")
159     return render_template("Resultados.html", respuestas=respuestas)
160
```

Obtiene todas las respuestas almacenadas en la base de datos y las envía a la plantilla Resultados.html para su visualización.

### Función eliminar\_respuesta

```
60
    Tabnine | Edit | Test | Explain | Document
61 @app.route("/eliminar/<int:respuesta_id>", methods=["POST"])
62 def eliminar_respuesta(respuesta_id):
63     registro = Respuesta.query.get(respuesta_id)
64     if registro:
65         db.session.delete(registro)
66         db.session.commit()
67         print(f"Registro con ID {respuesta_id} eliminado.")
68     else:
69         print(f"No se encontró registro con ID {respuesta_id}.")
70     return redirect(url_for("ver_resultados"))
71
72
```

Elimina un registro específico de la base de datos según su ID. Si el registro existe, lo borra y confirma la acción. Si no, muestra un mensaje indicando que no se encontró el registro. Finalmente, redirige a la página de resultados.

## Clase respuesta

```
# Definir modelo de datos
class Respuesta(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True) # Agregar autoincrement=True
    nombre_completo = db.Column(db.String(100), nullable=False)
    edad = db.Column(db.Integer, nullable=False)
    departamento = db.Column(db.String(100), nullable=False)
    amor = db.Column(db.Float, nullable=False)
    dinero = db.Column(db.Float, nullable=False)
    familia = db.Column(db.Float, nullable=False)
    salud = db.Column(db.Float, nullable=False)
```

Define la estructura de los datos que se almacenarán en la base de datos, incluyendo campos como nombre, edad, departamento y porcentajes relacionados con diferentes aspectos.

## Función \_repr\_

```
Tabnine | Edit | Test | Explain | Document
def __repr__(self):
    return f'<Respuesta {self.nombre_completo} - Amor: {self.amor}% - Dinero: {self.dinero}% - Familia: {self.famili
```

Devuelve una representación en texto del objeto Respuesta, mostrando los valores almacenados de manera legible.

## Crea base de datos

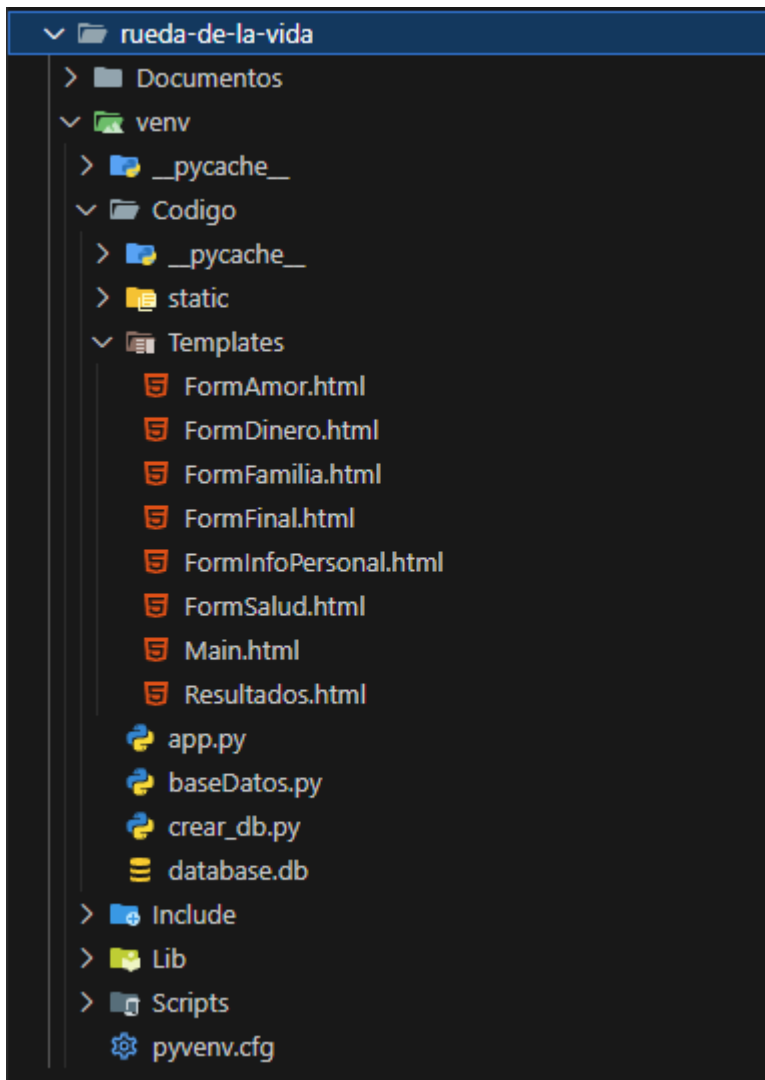
```
rueda-de-la-vida > venv > Codigo > crear_db.py
1  from app import app
2  from baseDatos import db
3
4  with app.app_context():
5      db.create_all()
6      print("✅ Base de datos creada exitosamente.")
```

Este código crea la base de datos y sus tablas. Primero, importa `app` y `db` desde sus respectivos módulos. Luego, usa `app.app_context()` para establecer un contexto de aplicación, lo que permite que SQLAlchemy interactúe correctamente con la base de datos. Dentro de este contexto, `db.create_all()` genera las tablas definidas en los modelos de datos. Finalmente, imprime un mensaje confirmando que la base de datos fue creada exitosamente.

## Base de datos

TABLAS		id	nombre...	edad	departam...	amor	dinero	far
respuesta		Filtrar	Filtrar	Filtrar	Filtrar	Filtrar...	Filtrar...	
ROWID		1	1	sebastian arango	26	sistemas	14.499999999999998	16
# id		2	2	carlos mora	31	contabilidad	12.5	14.499999999999998
# nombre_completo		3	3	karen morales	51	comercial	13	14.000000000000002
# edad		4	4	yoskar	25	antioquia	17	15
# departamento		5						
# amor								
# dinero								
# familia								
# salud								

aquí se muestra la tabla respuesta de la base de datos SQLite utilizando la extensión SQLite Viewer en Visual Studio Code. Se pueden ver los registros almacenados con sus respectivos campos (id, nombre\_completo, edad, departamento, amor, etc.), lo que confirma que la base de datos ha sido creada y contiene datos.



En la imagen se puede observar la estructura organizada de los directorios y archivos del proyecto “rueda-de-la-vida”. Este proyecto está estructurado de la siguiente manera:

- venv/: Contiene el entorno virtual del proyecto, lo que permite gestionar dependencias de Python de forma aislada sin afectar otras instalaciones en el sistema. Dentro de este directorio se encuentran carpetas como Include, Lib, Scripts, y el archivo pyvenv.cfg, que son esenciales para el funcionamiento del entorno virtual.
- Codigo/: Es el directorio principal del código del proyecto. Dentro de él, encontramos:
  - static/: destinado a almacenar archivos estáticos como imágenes o CSS

- Templates/: Carpeta donde están almacenadas las plantillas HTML utilizadas en la aplicación, incluyendo formularios como FormAmor.html, FormDinero.html, Main.html, y Resultados.html, entre otros.
- app.py: Archivo principal de la aplicación, el punto de entrada donde se configura el servidor y las rutas.
- baseDatos.py y crear\_db.py: Scripts relacionados con la gestión de la base de datos.
- database.db: Archivo de la base de datos, que sugiere el uso de SQLite.

Gracias al uso de un entorno virtual (venv), el proyecto puede ejecutarse con versiones específicas de paquetes sin interferir con otros proyectos o configuraciones del sistema.

Vista de formularios

Main.html

Rueda de la Vida

Bienvenido a la Encuesta

Conoce tu nivel de satisfacción en distintas áreas de tu vida: amor, dinero, familia y salud. Completa esta encuesta y visualiza tus resultados para identificar oportunidades de mejora.

Ingresar Encuesta

Visualizar Datos

© Derechos Reservados 2025

FormInfoPersonal.html

Información Personal

Nombre Completo

Ingrese su nombre completo

Edad

Ingrese su edad

Departamento

Ingrese su departamento

Continuar



## formAmor.html

### Formulario de Amor

1. ¿Qué tan satisfecho/a estás con tu vida amorosa actual?

Selecciona un número

2. ¿Qué tan presente y estable sientes el amor en tu vida?

Selecciona un número

3. ¿Qué tan bien te comunicas con tu pareja o con las personas de interés romántico?

Selecciona un número

4. ¿Qué tan conectado/a emocionalmente te sientes con tu pareja o con alguien especial?

Selecciona un número

5. ¿Qué tan importante consideras el amor en tu felicidad y bienestar general?

Selecciona un número

Continuar

## FormDinero.html

### Formulario de Dinero

1. ¿Ahorras e inviertes al menos el 20% de tus ingresos cada mes?

Selecciona un número

2. ¿Tienes un presupuesto mensual y lo sigues disciplinadamente?

Selecciona un número

3. ¿Puedes cubrir un gasto inesperado sin endeudarte?

Selecciona un número

4. ¿Tienes claro cómo hacer crecer tu dinero a través de inversiones?

Selecciona un número

5. ¿Tienes un plan financiero a largo plazo con metas definidas?

Selecciona un número

Continuar

## formFamilia.html

### Formulario de Familia

1. ¿Qué tan satisfecho/a estás con tu relación actual con tu familia?

Selecciona un número

2. ¿Qué tan presente y unido/a te sientes con tu familia?

Selecciona un número

3. ¿Qué tan buena es la comunicación dentro de tu familia?

Selecciona un número

4. ¿Qué tan apoyado/a y comprendido/a te sientes por tu familia?

Selecciona un número

5. ¿Qué tan importante consideras a tu familia en tu felicidad y bienestar general?

Selecciona un número

Continuar

## formSalud.html

### Formulario de Salud

1. ¿Estás comiendo de manera equilibrada y nutritiva?

Selecciona un número

2. ¿Cómo calificarías tu nivel de energía diaria?

Selecciona un número

3. ¿Cuántas horas de sueño duermes en promedio cada noche?

Selecciona un número

4. ¿Realizas alguna actividad física regularmente?

Selecciona un número

5. ¿Cómo gestionas el estrés y las emociones en tu vida diaria?

Selecciona un número

Continuar

formFinal.html

## Encuesta Finalizada

Presiona el botón para guardar tus respuestas.

Guardar Encuesta

Volver al Menú Principal

Resultados.html

Resultados Guardados

ID	Nombre	Edad	Departamento	Amor	Dinero	Familia	Salud	Acciones
1	sebastian arango	26	sistemas	14.499999999999998%	16.0%	14.499999999999998%	14.000000000000002%	Eliminar
2	carlos mora	31	contabilidad	12.5%	14.499999999999998%	16.0%	15.0%	Eliminar
3	karen morales	51	comercial	13.0%	14.000000000000002%	14.499999999999998%	14.000000000000002%	Eliminar
4	yoskar	25	antioquia	17.0%	15.0%	15.5%	12.0%	Eliminar
5	g	4	g	14.000000000000002%	15.5%	16.5%	17.0%	Eliminar

Volver al Menú Principal

## **CONCLUSION**

Este proyecto consiste en una aplicación web desarrollada con Flask y SQLAlchemy que permite gestionar registros en una base de datos SQLite. A lo largo del desarrollo, se han implementado funcionalidades clave como la creación de la base de datos, la definición de un modelo de datos con atributos específicos, y la gestión de registros mediante rutas que permiten visualizar, agregar y eliminar datos.

El sistema almacena información de usuarios, incluyendo su nombre, edad, departamento y valores numéricos asociados a diferentes categorías como amor, dinero, familia y salud. Además, se ha utilizado SQLite Viewer en Visual Studio Code para verificar la estructura y contenido de la base de datos.

En conclusión, este proyecto es un ejemplo sólido de cómo combinar Flask, SQLAlchemy y SQLite para gestionar información de manera estructurada y eficiente, sentando las bases para futuras ampliaciones y mejoras, como la integración de una interfaz más avanzada o la adición de nuevas funcionalidades de análisis de datos.