**Aplicación Flask - Gestión de Personas (Grupos de Vida)**

**📋 RESUMEN EJECUTIVO**

**Aplicación Flask modular** para gestión de personas en grupos de vida matrimoniales. Permite importar datos desde CSV, gestionar información personal y del cónyuge, y visualizar en formato de tableros tipo Kanban.

**🗂️ ESTRUCTURA DEL PROYECTO**

tu\_proyecto\_LIMPIO/

├── run.py # ← Punto de entrada

├── config.py # ← Configuración

├── requirements.txt # ← Dependencias

├── trello\_app.db # ← Base de datos SQLite

├── .env # ← Variables de entorno

├── app/ # ← Aplicación modular principal

│ ├── \_\_init\_\_.py # ← Factory pattern con blueprints

│ ├── models.py # ← Modelos de datos

│ ├── static/ # ← CSS, JS, imágenes

│ ├── templates/ # ← Templates HTML Jinja2

│ │ ├── auth/ # ← Templates de autenticación

│ │ ├── tableros/ # ← Templates de tableros

│ │ └── main/ # ← Templates principales

│ ├── auth/ # ← Blueprint de autenticación

│ │ ├── \_\_init\_\_.py

│ │ └── routes.py

│ ├── main/ # ← Blueprint principal

│ │ ├── \_\_init\_\_.py

│ │ └── routes.py

│ ├── tableros/ # ← Blueprint de tableros (PRINCIPAL)

│ │ ├── \_\_init\_\_.py

│ │ └── routes.py # ← Funciones principales

│ ├── api/ # ← Blueprint de API

│ │ ├── \_\_init\_\_.py

│ │ └── routes.py

│ └── utils/ # ← Utilidades

└── venv/ # ← Entorno virtual

**🎯 FUNCIONALIDADES PRINCIPALES**

**1. Gestión de Tableros**

* Crear, editar, eliminar tableros
* Organización tipo Kanban con listas y tarjetas
* Arrastrar y soltar para reorganizar

**2. Gestión de Personas (Tarjetas)**

* Información personal: nombre, dirección, teléfono, edad
* Información familiar: estado civil, número de hijos, edades
* **Información del cónyuge**: nombre, teléfono, edad, trabajo, fecha matrimonio
* Campos calculados y propiedades

**3. Importación/Exportación**

* **Descargar plantilla Excel/CSV** con campos del cónyuge
* **Importar datos** desde CSV con validación
* Procesamiento robusto con manejo de errores

**4. Autenticación**

* Login/logout de usuarios
* Registro de nuevos usuarios
* Sesiones de usuario

**🏗️ ARQUITECTURA TÉCNICA**

**Patrón de Diseño: Factory Pattern con Blueprints**

# app/\_\_init\_\_.py

def create\_app():

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

# Registrar blueprints

app.register\_blueprint(auth\_bp, url\_prefix="/auth")

app.register\_blueprint(main\_bp)

app.register\_blueprint(tableros\_bp, url\_prefix="/tableros")

app.register\_blueprint(api\_bp, url\_prefix="/api")

return app

**Modelos de Datos Principales:**

**Clase Tarjeta (Persona)**

class Tarjeta:

def \_\_init\_\_(self, nombre, apellido="", direccion="", telefono=""):

self.id = str(uuid.uuid4())

# Información personal

self.nombre = nombre

self.apellido = apellido

self.direccion = direccion

self.telefono = telefono

self.edad = None

self.estado\_civil = ""

self.numero\_hijos = 0

self.edades\_hijos = ""

# Información del cónyuge (CAMPOS AGREGADOS)

self.nombre\_conyuge = ""

self.telefono\_conyuge = ""

self.edad\_conyuge = None

self.trabajo\_conyuge = ""

self.fecha\_matrimonio = ""

**Método agregar\_persona (usa \*\*kwargs):**

def agregar\_persona(self, nombre, apellido="", direccion="", telefono="", \*\*kwargs):

persona = Tarjeta(nombre, apellido, direccion, telefono)

# Asignar campos adicionales automáticamente

for campo, valor in kwargs.items():

if hasattr(persona, campo):

setattr(persona, campo, valor)

self.tarjetas.append(persona)

return persona

**🔧 FUNCIONES CLAVE MODIFICADAS**

**1. Descarga de Plantilla (app/tableros/routes.py)**

@tableros\_bp.route("/descargar\_plantilla")

def descargar\_plantilla\_excel():

"""Descargar template de Excel con campos del cónyuge"""

contenido\_csv = """Nombre,Dirección,Teléfono,Edad,Estado Civil,Num Hijos,Edades Hijos,Nombre Cónyuge,Edad Cónyuge,Teléfono Cónyuge,Trabajo Cónyuge,Fecha Matrimonio

Juan Pérez,Calle 123,555-0123,35,Casado,2,"5,8",María Pérez,32,555-0130,Maestra,2018-06-15"""

**2. Importación de CSV (app/tableros/routes.py)**

# Lectura de campos del cónyuge del CSV

nombre\_conyuge = fila.get('Nombre Cónyuge', '').strip()

telefono\_conyuge = fila.get('Teléfono Cónyuge', '').strip()

# Llamada con campos del cónyuge

nueva\_persona = lista\_encontrada.agregar\_persona(

nombre=nombre,

apellido=apellido,

# ... otros campos ...

nombre\_conyuge=nombre\_conyuge,

telefono\_conyuge=telefono\_conyuge,

# ... campos adicionales ...

)

**3. Visualización en Templates (app/templates/tableros/ver.html)**

{% if tarjeta.nombre\_conyuge %}

<br><small style="color: #f59e0b;">💑 {{ tarjeta.nombre\_conyuge }}</small>

{% endif %}

**🚀 CÓMO EJECUTAR**

# 1. Navegar al proyecto

cd /Users/joserea/tu\_proyecto\_LIMPIO

# 2. Activar entorno virtual

source venv/bin/activate

# 3. Ejecutar aplicación

python3 run.py

# 4. Acceder en navegador

# http://127.0.0.1:5001

**📊 ESTADO ACTUAL DEL SISTEMA**

**✅ FUNCIONANDO CORRECTAMENTE:**

* ✅ Aplicación Flask modular ejecutándose
* ✅ Autenticación de usuarios
* ✅ Gestión de tableros y listas
* ✅ Creación y edición de tarjetas/personas
* ✅ **Modelo de datos con campos del cónyuge**
* ✅ **Plantilla CSV con campos del cónyuge**
* ✅ **Importación CSV procesando campos del cónyuge**
* ✅ **Visualización de información del cónyuge en tarjetas**
* ✅ Drag & drop para reorganizar
* ✅ Base de datos persistente

**🔧 MEJORAS IMPLEMENTADAS RECIENTEMENTE:**

1. **Campos del cónyuge agregados al modelo Tarjeta**
2. **Función de importación actualizada** para leer campos del cónyuge
3. **Plantilla CSV mejorada** con columnas del cónyuge
4. **Template actualizado** para mostrar información del cónyuge
5. **Proyecto limpio** con solo archivos esenciales

**📝 NOTAS TÉCNICAS IMPORTANTES**

**Arquitectura actual:**

* ✅ **Modular con blueprints** (no monolítica)
* ✅ **Factory pattern** en app/\_\_init\_\_.py
* ✅ \*\*Método agregar\_persona con **kwargs** (flexible para campos nuevos)
* ✅ **Templates organizados por blueprint**

**Base de datos:**

* **Tipo:** Storage en memoria con persistencia
* **Archivo:** trello\_app.db
* **Modelos:** Tablero, Lista, Tarjeta
* **UUIDs:** Sí, usa UUIDs en lugar de IDs enteros

**🎯 CAMPOS DEL CÓNYUGE - ESPECIFICACIONES**

**Campos en modelo Tarjeta:**

* nombre\_conyuge (str)
* telefono\_conyuge (str)
* edad\_conyuge (int)
* trabajo\_conyuge (str)
* fecha\_matrimonio (str)

**Columnas en CSV:**

* "Nombre Cónyuge"
* "Teléfono Cónyuge"
* "Edad Cónyuge"
* "Trabajo Cónyuge"
* "Fecha Matrimonio"

**Visualización:**

* Aparece debajo de información personal
* Ícono: 💑
* Color: #f59e0b (naranja)
* Solo se muestra si hay datos del cónyuge

**💡 PARA DESARROLLADORES**

**Cómo agregar nuevos campos:**

1. Agregar al modelo Tarjeta en app/models.py
2. Actualizar función de importación en app/tableros/routes.py
3. Actualizar plantilla CSV en descargar\_plantilla\_excel()
4. Actualizar template de visualización

**Estructura de rutas:**

* / → Redirige a tableros
* /auth/\* → Login, registro, logout
* /tableros/\* → Gestión de tableros y personas
* /api/\* → Endpoints de API

**Debugging:**

# Verificar datos del cónyuge

import sys

sys.path.append('.')

from app.models import storage

for tablero in storage.get\_all\_tableros():

for lista in tablero.listas:

for tarjeta in lista.tarjetas:

if tarjeta.nombre\_conyuge:

print(f'{tarjeta.nombre} → {tarjeta.nombre\_conyuge}')

**🎉 RESUMEN FINAL**

**Aplicación Flask funcional y completa** para gestión de personas con información del cónyuge. Sistema modular, robusto, con importación/exportación CSV y visualización amigable.

**Todo funcionando correctamente** a partir de julio 2025.