

# Proyecto Bench - Sistema de Gestión de Desarrolladores

---

Sistema Flask con **base de datos JSON en memoria** para gestionar desarrolladores core y bench en proyectos de desarrollo.

🎉 ¡Súper Simple!

**Sin MySQL, sin Docker, sin complicaciones.** Solo Python + Flask = ¡Funciona en 2 minutos!

💡 ¿Qué es?

Un sistema que permite:

- Asignar desarrolladores como **"Core"** (responsables principales) o **"Bench"** (soporte) en proyectos
- Solicitar días libres y refuerzos
- Visualizar calendario de asignaciones
- Gestionar usuarios y proyectos
- **Persistencia automática en archivos JSON** - sin necesidad de base de datos externa

## ✨ Ventajas de la versión JSON

- 🚀 **Instalación instantánea** - Sin instalaciones complejas
- 👁️ **Debug fácil** - Puedes ver y editar los datos directamente
- 📁 **100% portable** - Lleva el proyecto a cualquier lado
- 🔄 **Versionable** - Los archivos JSON se pueden versionar con Git
- 🇫🇷 **Ideal para prototipos** y equipos pequeños (hasta 1000 registros)

## ⚙️ Requisitos previos

🐍 Python (¡Solo esto necesitas!)

**macOS:**

```
# Opción 1: Homebrew (recomendado)
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install python

# Opción 2: Descargar desde https://python.org/downloads/
```

**Verificar instalación:**

```
python3 --version
# 0
python --version
```


### Verificar pip:

```
pip3 --version
# 0
pip --version
```

### Windows:

```
# Descargar desde https://python.org/downloads/
# Durante la instalación, marcar "Add Python to PATH"

# Verificar instalación
python --version
python -m pip --version
```

 Si pip no funciona

### macOS:

```
# Si pip3 no funciona, reinstalar
python3 -m ensurepip --upgrade

# O instalar manualmente
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py

# En macOS, casi siempre es pip3, no pip
pip3 install flask
```

### Windows:

```
# Si pip no funciona, reinstalar
python -m ensurepip --upgrade

# O instalar manualmente
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
```

---

## Instalación súper simple (3 pasos)

### 1. Obtener el proyecto

```
# Opción A: Clonar el repositorio
git clone [URL_DEL_REPOSITORIO]
cd flask-proyecto-bench

# Opción B: Crear carpeta nueva
mkdir flask-proyecto-bench
cd flask-proyecto-bench
# Luego copia el archivo app.py
```

### 2. Crear entorno virtual e instalar dependencias

#### macOS:

```
# Crear entorno virtual
python3 -m venv venv

# Activar entorno virtual
source venv/bin/activate

# Instalar dependencias (usa pip3 si pip no funciona)
pip3 install flask
# o
pip install flask
```

#### Windows:

```
# Crear entorno virtual
python -m venv venv

# Activar entorno virtual
venv\Scripts\activate

# Instalar dependencias
pip install flask
```

### 3. Ejecutar la aplicación

#### macOS:

```
python3 app.py
```

#### Windows:

```
python app.py
```

#### 4. 🌐 ¡Listo!

Abrir en navegador: **http://127.0.0.1:5000**

#### ✅ Verás:

- Página principal con datos de ejemplo
- Carpeta **data/** creada automáticamente
- 4 archivos JSON con datos ficticios completos



### Datos incluidos automáticamente

El sistema carga **datos ficticios realistas**:

- **6 usuarios** con skills, roles y seniority
- **5 proyectos** con presupuestos, clientes y tecnologías
- **6 asignaciones** core/bench con porcentajes reales
- **3 solicitudes** (vacaciones, refuerzos, cambios)



### Rutas disponibles

Páginas principales:

- **/** - Página principal con resumen
- **/proyectos** - Lista completa de proyectos
- **/usuarios** - Lista de usuarios activos
- **/proyecto/<id>** - Detalle de proyecto específico

APIs JSON:

- **/api/datos** - Todos los datos JSON
- **/api/proyectos** - Solo proyectos
- **/api/usuarios** - Solo usuarios
- **/dashboard** - Estadísticas del sistema
- **/solicitudes** - Lista de solicitudes

Pruebas:

- **/test** - Verificar que el servidor funciona
- **/test-json** - Verificar que JSON funciona

## Estructura del proyecto

```
flask-proyecto-bench/
├── app.py                # Aplicación principal (TODO EN UNO)
├── README.md             # Esta documentación
├── requirements.txt      # Dependencias (solo Flask)
├── data/                 # Base de datos JSON (se crea automáticamente)
│   ├── usuarios.json    # Usuarios con skills y roles
│   ├── proyectos.json   # Proyectos con presupuestos
│   ├── asignaciones.json # Asignaciones core/bench
│   └── solicitudes.json  # Solicitudes de cambios
├── app/                  # Carpeta opcional para templates
│   ├── templates/       # Si quieres usar templates HTML
│   └── static/           # CSS, JS, imágenes
└── venv/                 # Entorno virtual
```

## Tecnologías utilizadas

- **Backend:** Flask (solo esto!)
- **Base de datos:** JSON (archivos locales)
- **Frontend:** HTML generado automáticamente
- **Persistencia:** Sistema de archivos

## Requirements.txt minimalista

```
Flask==2.3.3
```

## Solución de problemas comunes

 Error: "No module named 'flask'"

**macOS:**

```
# Activar entorno virtual
source venv/bin/activate

# Usar pip3 en lugar de pip
pip3 install flask
```

**Windows:**

```
# Activar entorno virtual
venv\Scripts\activate
```

```
# Instalar Flask
pip install flask
```

✗ Error: "pip command not found"

**macOS:**

```
# Usar pip3 en lugar de pip
pip3 --version

# Si no funciona, reinstalar pip
python3 -m ensurepip --upgrade
```

**Windows:**

```
# Usar python -m pip
python -m pip --version

# Si no funciona, reinstalar pip
python -m ensurepip --upgrade
```

✗ Error: "Port 5000 already in use"

**macOS:**

```
# Ver qué usa el puerto
lsof -ti:5000

# Matar el proceso
kill -9 [PID]
```



**Windows:**

```
# Ver qué usa el puerto
netstat -ano | findstr :5000

# Matar el proceso
taskkill /PID [PID] /F
```

✗ Error: "localhost not working"

**Usar 127.0.0.1 en lugar de localhost:**

-  <http://127.0.0.1:5000/>
-  <http://localhost:5000/>

## Comandos útiles

Gestión del entorno virtual:

**macOS:**

```
# Activar
source venv/bin/activate

# Desactivar
deactivate
```

**Windows:**

```
# Activar
venv\Scripts\activate

# Desactivar
deactivate
```

Gestión de datos:






```
# Ver estructura de datos
python3 -m json.tool data/usuarios.json


# Respalidar datos
cp -r data/ backup_$(date +%Y%m%d)

# Limpiar datos (reiniciar)
rm -rf data/
```

## Roadmap de desarrollo

 Completado:

-  Sistema de archivos JSON como base de datos
-  Datos ficticios realistas
-  CRUD básico con persistencia
-  API REST completa
-  Interfaz HTML automática

 En progreso:

- ☐ Templates HTML personalizados
- ☐ Sistema de autenticación
- ☐ Formularios para CRUD
- ☐ Calendario interactivo

#### Futuro:

- ☐ PWA (Progressive Web App)
- ☐ Sistema de notificaciones
- ☐ Reportes automáticos
- ☐ Migración opcional a DB real

## Ventajas para diferentes casos de uso

### Para desarrolladores aprendiendo:

- Sin complejidad de base de datos
- Ve exactamente cómo se almacenan los datos
- Fácil de modificar y experimentar

### Para equipos pequeños:

- Setup instantáneo en cualquier máquina
- Backup simple: solo copia la carpeta `data/`
- Perfecto para hasta 1000 registros

### Para prototipos y demos:

- Funciona inmediatamente
- Datos realistas incluidos
- Fácil de mostrar y explicar

## Contribuir

1. Fork el proyecto
2. Crea una rama para tu feature: `git checkout -b feature/nueva-funcionalidad`
3. Commit tus cambios: `git commit -m 'Agrega nueva funcionalidad'`
4. Push a la rama: `git push origin feature/nueva-funcionalidad`
5. Abre un Pull Request

## Licencia

MIT License - ver archivo LICENSE para más detalles.

## Migración futura a base de datos real

Cuando tu proyecto crezca, puedes migrar fácilmente a MySQL/PostgreSQL:



1. La estructura de datos ya está definida
2. Los modelos están claros en los JSON
3. Solo cambias las funciones `load_json_data()` y `save_json_data()`
4. El resto del código permanece igual

**¡Empezar simple, crecer cuando sea necesario! 🚀**