



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.

💡 Detalles de los datos ficticios:

👤 Usuarios (6 personas):

- **Ana García** - Tech Lead Senior (React, Node.js, Python)
- **Carlos Rodríguez** - Developer Mid (Vue.js, Laravel, MySQL)
- **María López** - Developer Junior (React Native, Flutter)
- **Diego Martínez** - Developer Senior (Angular, Spring Boot)
- **Sofia Chen** - Designer Mid (Figma, UI/UX)
- **Roberto Silva** - Ex-empleado (inactivo)

🚀 Proyectos (5 proyectos):

- **E-commerce Mobile** - \$85K, React Native (activo)
- **Sistema Escolar** - \$120K, Angular + Spring (activo)
- **Dashboard Analytics** - \$95K, React + Python (planificación)
- **API Facturación** - \$45K, Laravel (completado)
- **App Delivery** - \$135K, Flutter (planificación)

📅 Tipos de solicitudes incluidas:

- **Días libres** - Vacaciones y conferencias
- **Refuerzos** - Solicitudes de más desarrolladores
- **Cambios** - Modificaciones de asignación
- **Incorporaciones** - Nuevos miembros al proyecto
- **Extensiones** - Ampliar participación



¡Instalación super simple!

Sin base de datos externa, sin Docker, sin complicaciones. Solo Python y listo para funcionar.

💡 ¿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 de datos en archivos JSON** - sin necesidad de base de datos externa

✨ Ventajas de usar JSON como base de datos:

- 🚀 **Instalación instantánea:** No necesitas instalar MySQL, PostgreSQL o Docker
- 🎯 **Ideal para desarrollo:** Perfecto para prototipos, demos y desarrollo local
- 📁 **Portable:** Toda la información en archivos que puedes respaldar fácilmente
- 🗣️ **Visible:** Puedes abrir y editar los datos directamente en cualquier editor
- 🔄 **Versionable:** Los archivos JSON se pueden versionar con Git
- ⚡ **Rápido setup:** De cero a funcionando en menos de 5 minutos

⚙️ Requisitos previos

🐍 Python (SOLO esto necesitas!)

🖥️ Windows:

1. Descargar Python desde <https://python.org/downloads/>
2. Durante la instalación, marcar **"Add Python to PATH"**
3. Verificar instalación en CMD: `python --version`
4. **Verificar pip:** `python -m pip --version`

Si pip no está instalado:

```
# Descargar get-pip.py
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

```
# Instalar pip
python get-pip.py
```

macOS:

1. **Opción 1 - Homebrew (recomendado):**

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

2. **Opción 2:** Descargar desde <https://python.org/downloads/>

3. Verificar instalación en Terminal: `python3 --version`

4. **Verificar pip:** `python3 -m pip --version`

Si pip no está instalado:

```
# Descargar get-pip.py
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

# Instalar pip
python3 get-pip.py
```

Git (opcional, para clonar el repo)

Windows:

1. Descargar Git desde <https://git-scm.com/>
2. Instalar con configuración por defecto
3. Verificar en CMD: `git --version`

macOS:

1. Con Homebrew: `brew install git`

2. O instalar Xcode Command Line Tools: `xcode-select --install`

3. Verificar en Terminal: `git --version`



Instalación súper simple (3 pasos)

1. 📦 Obtener el proyecto

⚠️ Si ya tienes el proyecto con MySQL:

El código actual usa SQLAlchemy. Necesitas crear la versión JSON desde cero o reemplazar `app.py`



Windows (PowerShell o CMD):

```
# Opción A: Proyecto nuevo (recomendado)
mkdir flask-proyecto-bench-json
cd flask-proyecto-bench-json

# Opción B: Si ya tienes el proyecto
cd flask-proyecto-bench

# Necesitarás reemplazar app.py (ver sección de problemas comunes)
```



macOS (Terminal):

```
# Opción A: Proyecto nuevo (recomendado)
mkdir flask-proyecto-bench-json
cd flask-proyecto-bench-json

# Opción B: Si ya tienes el proyecto
cd flask-proyecto-bench

# Necesitarás reemplazar app.py (ver sección de problemas comunes)
```

2. 🌐 Crear entorno virtual e instalar dependencias



Windows:

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

# Activar entorno virtual
venv\Scripts\activate

# Instalar dependencias
pip install flask flask-login wtforms
```

macOS:

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

# Activar entorno virtual
source venv/bin/activate

# Instalar dependencias
pip install flask flask-login wtforms
```

Nota: Verás `(venv)` al inicio de la línea cuando el entorno virtual esté activado.

3. Ejecutar la aplicación

 **Si tienes error "No module named 'flask_sqlalchemy'" :**

Crea un nuevo archivo `app.py` con el código de la sección "Solución de problemas comunes"

Windows:

```
python app.py
```

macOS:

```
python3 app.py
```

¡Listo! Abrir en navegador:

http://localhost:5000

Los archivos JSON se crearán automáticamente en la carpeta `data/`

✓ Deberías ver:

- Mensaje "🚀 Proyecto Bench - Sistema JSON"
- Enlaces a /proyectos y /usuarios
- Carpeta `data/` creada automáticamente
- 4 archivos JSON con datos de ejemplo



Estructura de datos JSON con ejemplos ricos

Datos ficticios incluidos:

- **6 usuarios** - Tech leads, developers, diseñador (con skills y seniority)
- **5 proyectos** - E-commerce, gestión escolar, analytics, API, delivery
- **6 asignaciones** - Core y bench con porcentajes reales
- **6 solicitudes** - Vacaciones, refuerzos, cambios, incorporaciones

✨ **Todo interconectado** - Los IDs y relaciones son coherentes entre archivos

El sistema creará automáticamente estos archivos en la carpeta `data/` con **datos realistas y completos**:



data/usuarios.json

```
{
  "usuarios": [
    {
      "id": 1,
      "nombre": "Ana García",
      "email": "ana.garcia@empresa.com",
      "rol": "tech_lead",
      "activo": true,
      "fecha_registro": "2023-12-01T09:00:00",
```



```
      "skills": ["PHP", "WordPress", "jQuery"],
      "seniority": "mid",
      "fecha_salida": "2024-06-30"
    }
  ]
}
```

data/proyectos.json

```
{
  "proyectos": [
    {
      "id": 1,
      "nombre": "E-commerce Mobile App",
      "descripcion": "Aplicación móvil para tienda online con carrito",
      "fecha_inicio": "2024-02-01",
      "fecha_fin": "2024-06-30",
      "estado": "activo",
      "core_developers": [1, 3],
      "bench_developers": [2],
      "tecnologias": ["React Native", "Node.js", "MongoDB", "Stripe"],
      "cliente": "TechStore S.A.",
      "presupuesto": 85000,
      "prioridad": "alta"
    },
    {
      "id": 2,
      "nombre": "Sistema de Gestión Escolar",
      "descripcion": "Plataforma web para administrar estudiantes, profesores",
      "fecha_inicio": "2024-01-15",
      "fecha_fin": "2024-08-15",
      "estado": "activo",
      "core_developers": [4],
      "bench_developers": [2, 5],
      "tecnologias": ["Angular", "Spring Boot", "PostgreSQL", "Docker"],
      "cliente": "Colegio San Martín",
      "presupuesto": 120000,
      "prioridad": "media"
    },
    {
      "id": 3,
      "nombre": "Dashboard Analytics",
      "descripcion": "Tablero de control con métricas en tiempo real",
      "fecha_inicio": "2024-03-01",
      "fecha_fin": "2024-05-31",
      "estado": "planificacion",
      "tecnologias": ["Tableau", "Python", "AWS", "Google Analytics"],
      "cliente": "DataCorp",
      "presupuesto": 60000,
      "prioridad": "media"
    }
  ]
}
```



```

        "core_developers": [1],
        "bench_developers": [4, 5],
        "tecnologias": ["React", "D3.js", "Python", "Redis"],
        "cliente": "DataCorp Inc.",
        "presupuesto": 95000,
        "prioridad": "alta"
    },
    {
        "id": 4,
        "nombre": "API de Facturación",
        "descripcion": "Microservicio para generar y gestionar facturas",
        "fecha_inicio": "2023-11-01",
        "fecha_fin": "2024-02-29",
        "estado": "completado",
        "core_developers": [2, 4],
        "bench_developers": [],
        "tecnologias": ["Laravel", "MySQL", "PDF", "SOAP"],
        "cliente": "FactuFácil LTDA",
        "presupuesto": 45000,
        "prioridad": "baja"
    },
    {
        "id": 5,
        "nombre": "App de Delivery",
        "descripcion": "Aplicación para pedidos de comida con tracking",
        "fecha_inicio": "2024-04-01",
        "fecha_fin": "2024-09-30",
        "estado": "planificacion",
        "core_developers": [3],
        "bench_developers": [1, 2],
        "tecnologias": ["Flutter", "Node.js", "Socket.io", "Maps API"],
        "cliente": "FoodExpress",
        "presupuesto": 135000,
        "prioridad": "alta"
    }
]
}

```



data/asignaciones.json

```

{
  "asignaciones": [
    {
      "id": 1,
      "proyecto_id": 1,
      "usuario_id": 1,

```



```
},
{
  "id": 6,
  "proyecto_id": 2,
  "usuario_id": 5,
  "tipo": "bench",
  "fecha_asignacion": "2024-01-20",
  "fecha_fin": "2024-08-15",
  "porcentaje_dedicacion": 50,
  "activa": true,
  "rol_proyecto": "UI/UX Designer"
}
]
```

data/solicitudes.json

```
{
  "solicitudes": [
    {
      "id": 1,
      "usuario_id": 3,
      "proyecto_id": 1,
      "tipo": "dias_libres",
      "fecha_inicio": "2024-03-15",
      "fecha_fin": "2024-03-18",
      "motivo": "Vacaciones de Semana Santa",
      "estado": "aprobada",
      "fecha_solicitud": "2024-03-01T09:00:00",
      "aprobado_por": 1,
      "fecha_aprobacion": "2024-03-02T14:30:00"
    },
    {
      "id": 2,
      "usuario_id": 2,
      "proyecto_id": 2,
      "tipo": "refuerzo",
      "fecha_inicio": "2024-04-01",
      "fecha_fin": "2024-04-30",
      "motivo": "Necesitamos acelerar el desarrollo del módulo de r",
      "estado": "pendiente",
      "fecha_solicitud": "2024-03-25T16:45:00",
      "desarrolladores_solicitados": [4, 5],
      "horas_adicionales": 40
    },
    {
```



```
"estado": "pendiente",
"fecha_solicitud": "2024-04-28T17:00:00",
"nuevo_porcentaje": 70
}
]
```

Configuración opcional

Variables de entorno (.env)

Puedes crear un archivo `.env` para configuraciones personalizadas:

```
# Puerto del servidor (por defecto: 5000)
PORT=5000

# Clave secreta para sesiones
SECRET_KEY=tu_clave_secreta_super_segura

# Carpeta de datos (por defecto: data/)
DATA_FOLDER=data

# Modo debug (por defecto: True)
DEBUG=True
```

Backup automático

El sistema puede crear backups automáticos de tus datos JSON:

```
# En el archivo config.py
BACKUP_ENABLED = True
BACKUP_INTERVAL = 24 # horas
BACKUP_FOLDER = "backups/"
```

Solución de problemas comunes

✗ Error: "No module named 'flask_sqlalchemy'"

Problema más común:

Tu archivo `app.py` actual está configurado para MySQL/SQLAlchemy, pero necesitas la versión JSON.

Solución: Reemplazar app.py con la versión JSON



app.py correcto para JSON:

```
from flask import Flask, render_template, jsonify, request, redirect
import json
import os
from datetime import datetime

app = Flask(__name__)
app.secret_key = 'tu_clave_secreta_aqui'

# Configuración para JSON
DATA_FOLDER = 'data'

def ensure_data_folder():
    """Crear carpeta data si no existe"""
    if not os.path.exists(DATA_FOLDER):
        os.makedirs(DATA_FOLDER)

def load_json_data(filename):
    """Cargar datos desde archivo JSON"""
    filepath = os.path.join(DATA_FOLDER, filename)
    if os.path.exists(filepath):
        with open(filepath, 'r', encoding='utf-8') as f:
            return json.load(f)
    return {}

def save_json_data(filename, data):
    """Guardar datos en archivo JSON"""
    ensure_data_folder()
    filepath = os.path.join(DATA_FOLDER, filename)
    with open(filepath, 'w', encoding='utf-8') as f:
        json.dump(data, f, indent=2, ensure_ascii=False)

def init_default_data():
    """Inicializar datos por defecto si no existen"""
    # Usuarios completos con datos ficticios
    usuarios_default = {
        "usuarios": [
            {
                "id": 1,
                "nombre": "Ana García",
                "email": "ana.garcia@empresa.com",
                "rol": "tech_lead",
                "activo": True,
                "fecha_registro": "2023-12-01T09:00:00",
                "skills": ["React", "Node.js", "Python", "Leadership"],
                "seniority": "senior"
            }
        ],
    },
```

```
{
  {
    "id": 2,
    "nombre": "Carlos Rodríguez",
    "email": "carlos.rodriguez@empresa.com",
    "rol": "developer",
    "activo": True,
    "fecha_registro": "2024-01-15T10:30:00",
    "skills": ["Vue.js", "Laravel", "MySQL", "Docker"],
    "seniority": "mid"
  },
  {
    "id": 3,
    "nombre": "María López",
    "email": "maria.lopez@empresa.com",
    "rol": "developer",
    "activo": True,
    "fecha_registro": "2024-01-20T11:00:00",
    "skills": ["React Native", "Flutter", "Firebase"],
    "seniority": "junior"
  },
  {
    "id": 4,
    "nombre": "Diego Martínez",
    "email": "diego.martinez@empresa.com",
    "rol": "developer",
    "activo": True,
    "fecha_registro": "2023-11-10T14:15:00",
    "skills": ["Angular", "Spring Boot", "PostgreSQL"],
    "seniority": "senior"
  },
  {
    "id": 5,
    "nombre": "Sofia Chen",
    "email": "sofia.chen@empresa.com",
    "rol": "designer",
    "activo": True,
    "fecha_registro": "2024-02-01T08:30:00",
    "skills": ["Figma", "UI/UX", "Frontend", "Design Sys"],
    "seniority": "mid"
  }
]
}
```

Proyectos realistas con datos completos

```
proyectos_default = {
  "proyectos": [
    {
      "id": 1,
      "nombre": "E-commerce Mobile App",
      "descripcion": "Aplicación móvil para tienda online
```

```
        "fecha_inicio": "2024-02-01",
        "fecha_fin": "2024-06-30",
        "estado": "activo",
        "core_developers": [1, 3],
        "bench_developers": [2],
        "tecnologias": ["React Native", "Node.js", "MongoDB"],
        "cliente": "TechStore S.A.",
        "presupuesto": 85000,
        "prioridad": "alta"
    },
    {
        "id": 2,
        "nombre": "Sistema de Gestión Escolar",
        "descripcion": "Plataforma web para administrar estudiantes",
        "fecha_inicio": "2024-01-15",
        "fecha_fin": "2024-08-15",
        "estado": "activo",
        "core_developers": [4],
        "bench_developers": [2, 5],
        "tecnologias": ["Angular", "Spring Boot", "PostgreSQL"],
        "cliente": "Colegio San Martín",
        "presupuesto": 120000,
        "prioridad": "media"
    },
    {
        "id": 3,
        "nombre": "Dashboard Analytics",
        "descripcion": "Tablero de control con métricas en tiempo real",
        "fecha_inicio": "2024-03-01",
        "fecha_fin": "2024-05-31",
        "estado": "planificacion",
        "core_developers": [1],
        "bench_developers": [4, 5],
        "tecnologias": ["React", "D3.js", "Python", "Redis"],
        "cliente": "DataCorp Inc.",
        "presupuesto": 95000,
        "prioridad": "alta"
    }
]

# Asignaciones realistas
asignaciones_default = {
    "asignaciones": [
        {
            "id": 1,
            "proyecto_id": 1,
            "usuario_id": 1,
            "tipo": "core",
            "fecha_asignacion": "2024-02-01",
```



```
        "fecha_fin": "2024-06-30",
        "porcentaje_dedicacion": 80,
        "activa": True,
        "rol_proyecto": "Tech Lead"
    },
    {
        "id": 2,
        "proyecto_id": 1,
        "usuario_id": 3,
        "tipo": "core",
        "fecha_asignacion": "2024-02-15",
        "fecha_fin": "2024-06-30",
        "porcentaje_dedicacion": 100,
        "activa": True,
        "rol_proyecto": "Mobile Developer"
    },
    {
        "id": 3,
        "proyecto_id": 2,
        "usuario_id": 4,
        "tipo": "core",
        "fecha_asignacion": "2024-01-15",
        "fecha_fin": "2024-08-15",
        "porcentaje_dedicacion": 90,
        "activa": True,
        "rol_proyecto": "Full Stack Lead"
    }
]

# Solicitudes variadas
solicitudes_default = {
    "solicitudes": [
        {
            "id": 1,
            "usuario_id": 3,
            "proyecto_id": 1,
            "tipo": "dias_libres",
            "fecha_inicio": "2024-03-15",
            "fecha_fin": "2024-03-18",
            "motivo": "Vacaciones de Semana Santa",
            "estado": "aprobada",
            "fecha_solicitud": "2024-03-01T09:00:00",
            "aprobado_por": 1
        },
        {
            "id": 2,
            "usuario_id": 2,
            "proyecto_id": 2,
            "tipo": "refuerzo",
```

```
        "fecha_inicio": "2024-04-01",
        "fecha_fin": "2024-04-30",
        "motivo": "Necesitamos acelerar el desarrollo del módulo",
        "estado": "pendiente",
        "fecha_solicitud": "2024-03-25T16:45:00"
    }
]

# Crear archivos si no existen
if not os.path.exists(os.path.join(DATA_FOLDER, 'usuarios.json')):
    save_json_data('usuarios.json', usuarios_default)

if not os.path.exists(os.path.join(DATA_FOLDER, 'proyectos.json')):
    save_json_data('proyectos.json', proyectos_default)

if not os.path.exists(os.path.join(DATA_FOLDER, 'asignaciones.json')):
    save_json_data('asignaciones.json', asignaciones_default)

if not os.path.exists(os.path.join(DATA_FOLDER, 'solicitudes.json')):
    save_json_data('solicitudes.json', solicitudes_default)

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



Proyecto Bench - Sistema

¡Funcionando correctamente!

- Ver Proyectos
- Ver Usuarios
- API - Todos los datos

Los datos se almacenan en la carpeta 'data/' como archivos JSON.

```
"""

@app.route('/proyectos')
def proyectos():
    data = load_json_data('proyectos.json')
    proyectos = data.get('proyectos', [])
    return jsonify({
        "mensaje": "Lista de proyectos",
        "total": len(proyectos),
        "proyectos": proyectos
    })

@app.route('/usuarios')
def usuarios():
    data = load_json_data('usuarios.json')
    usuarios = data.get('usuarios', [])
    return jsonify({
        "mensaje": "Lista de usuarios",
        "total": len(usuarios),
        "usuarios": usuarios
    })

@app.route('/api/datos')
def api_datos():
    return jsonify({
        "usuarios": load_json_data('usuarios.json'),
        "proyectos": load_json_data('proyectos.json'),
        "asignaciones": load_json_data('asignaciones.json'),
        "solicitudes": load_json_data('solicitudes.json')
    })

if __name__ == '__main__':
    print("🚀 Iniciando Proyecto Bench...")
    print("📁 Inicializando datos JSON...")
    init_default_data()
    print("✅ Servidor listo en http://localhost:5000")
    app.run(host='0.0.0.0', port=5000, debug=True)
```

✗ Error con pip

🖥 Windows:

Problema: "pip no se reconoce como comando"

```
# Verificar si pip está instalado
python -m pip --version

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

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

Problema: "Permission denied" al instalar paquetes

```
# Usar --user para instalar en tu usuario
python -m pip install --user flask

# O ejecutar CMD como administrador
```

macOS:

Problema: "command not found: pip"

```
# Verificar si pip está instalado
python3 -m pip --version

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

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

Problema: "Permission denied" al instalar paquetes

```
# Usar --user para instalar en tu usuario
python3 -m pip install --user flask

# O usar homebrew si instalaste Python con brew
```

```
brew install python
# Esto incluye pip automáticamente
```

✗ Error "Puerto 5000 ocupado"

Windows:

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

# Matar el proceso (reemplazar [PID] con el número real)
taskkill /PID [PID] /F

# O cambiar el puerto en app.py
app.run(host='0.0.0.0', port=8000)
```

macOS:

```
# Ver qué está usando el puerto
lsof -ti:5000

# Matar el proceso
kill -9 [PID]

# O cambiar el puerto en app.py
app.run(host='0.0.0.0', port=8000)
```

✗ Error de permisos al crear archivos

Solución:

- Asegúrate de ejecutar desde una carpeta donde tengas permisos de escritura
- En Windows: No ejecutes desde `C:\Program Files\`
- En macOS: No ejecutes desde carpetas del sistema
- Recomendado: Usar tu carpeta de usuario o Desktop

❌ Datos perdidos al reiniciar

Verificar:

- Que la carpeta `data/` se esté creando correctamente
- Que los archivos JSON tengan permisos de escritura
- Que no estés ejecutando en modo temporal/incognito

📁 Estructura del proyecto

```
flask-proyecto-bench/ ├── app.py # Aplicación principal ├──
requirements.txt # Dependencias (simplificado) ├── README.md # Esta
documentación ├── .env # Variables de entorno (opcional) ├──
config.py # Configuración de la app ├── data/ # Base de datos JSON
(se crea automáticamente) | ├── usuarios.json | ├── proyectos.json |
| ├── asignaciones.json | └── solicitudes.json ├── backups/ #
RespalDOS automáticos (opcional) ├── app/ | ├── models/ | | ├──
__init__.py | | ├── usuario.py # Modelo Usuario | | ├── proyecto.py
# Modelo Proyecto | | └── database.py # Manejador JSON | ├── static/
| | ├── css/ | | | └── style.css # Estilos personalizados | | └──
js/ | | └── main.js # JavaScript principal | └── templates/ | ├──
base.html # Plantilla base | ├── index.html # Página principal | ├──
proyectos.html # Lista de proyectos | ├── proyecto_detalle.html #
Detalle de proyecto | ├── usuarios.html # Lista de usuarios | ├──
about.html # Acerca de | └── contact.html # Contacto
```

🛣️ Rutas disponibles

- `/` - Página principal con dashboard
- `/proyectos` - Lista de proyectos
- `/proyectos/nuevo` - Crear nuevo proyecto
- `/proyecto/<id>` - Detalle de proyecto específico
- `/usuarios` - Lista de usuarios
- `/usuarios/nuevo` - Registrar nuevo usuario
- `/asignaciones` - Gestión de asignaciones
- `/solicitudes` - Sistema de solicitudes
- `/calendario` - Vista de calendario
- `/api/datos` - API REST para datos JSON
- `/about` - Información del proyecto

- **/contact** - Formulario de contacto



Tecnologías utilizadas

Backend:

Flask, Flask-Login, WTForms

Base de datos:

JSON (archivos locales)

Frontend:

HTML5, CSS3, Bootstrap 5, JavaScript

Persistencia:

Sistema de archivos JSON



Comandos útiles para desarrollo



Activar/Desactivar entorno virtual



Windows:

```
# Activar
venv\Scripts\activate

# Desactivar
deactivate
```



macOS:

```
# Activar
source venv/bin/activate

# Desactivar
deactivate
```



Gestión de dependencias simplificada

requirements.txt minimalista:

```
Flask==2.3.3
Flask-Login==0.6.2
WTForms==3.0.1
```

Comandos útiles:

```
# Instalar dependencias
pip install -r requirements.txt

# Generar requirements actualizado
pip freeze > requirements.txt

# Verificar dependencias instaladas
pip list
```



Gestión de datos JSON

Comandos para manejo de datos:

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

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

# Importar datos de backup
cp -r backup_20241115/ data/

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



Próximos pasos de desarrollo

Sprint 1: Estructura base con JSON

- Sistema de archivos JSON como base de datos
- Modelos para Usuario, Proyecto, Asignación
- CRUD básico con persistencia en archivos

Sprint 2: Autenticación y permisos

- Sistema de login/registro con Flask-Login

- Roles y permisos almacenados en JSON
- Formularios con WTForms y validaciones

Sprint 3: Gestión completa de proyectos

- CRUD completo para proyectos
- Asignación de desarrolladores core y bench
- Validaciones de negocio y manejo de errores

Sprint 4: Calendario y solicitudes avanzadas

- Sistema completo de solicitudes
- Calendario interactivo de asignaciones
- Notificaciones y alertas
- Reportes y estadísticas

Sprint 5: Mejoras y optimización

- API REST completa
- Sistema de backup automático
- Migración opcional a base de datos real
- PWA y funcionalidades offline



Ventajas de esta arquitectura

✨ Para desarrollo y prototipos:

- 🔥 **Setup instantáneo:** Desde cero a funcionando en 2 minutos
- 🔍 **Debug fácil:** Puedes ver y editar los datos directamente
- 📱 **Portable:** Lleva tu proyecto a cualquier lado
- 🎯 **Ideal para demos:** Sin dependencias complejas



Para equipos pequeños:

- 🇮🇹 **Hasta 1000 registros:** Rendimiento excelente
- 🗝️ **Control total:** Sabes exactamente dónde están tus datos
- 💾 **Backup simple:** Solo copia la carpeta data/
- 🔄 **Versionado:** Los cambios se pueden trackear con Git



Para aprender:

- 📖 **Conceptos claros:** Ves cómo funcionan las bases de datos
- 🛠️ **Sin complejidad:** Enfócate en la lógica de negocio

- ⚡ **Migración futura:** Fácil evolución a DB tradicional



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.



Proyecto Bench - Sistema de Gestión de Desarrolladores

Creado con ❤️ para simplificar la gestión de equipos de desarrollo



Versión JSON - Sin complicaciones, máxima simplicidad