# Flask

- Ejecutar aplicaciones Flask en Nginx
- Uso de loginctl para procesos persistentes

# Ejecutar aplicaciones Flask en Nginx

La configuración del servidor Nginx en `alumnos.cbtis227.edu.mx` permite ejecutar aplicaciones Flask a través de la siguiente URL:

`https://alumnos.cbtis227.edu.mx/~‹usuario›/flask/‹proyecto›`

Donde:

- **`<usuario>`**: es el nombre de usuario en el servidor
- **`<proyecto>`**: es la carpeta que contiene la aplicación Flask

Existen dos entornos clave para visualizar la aplicación:

1. **Entorno de Desarrollo**
   - Para pruebas privadas donde podrás verificar avances y depurar el funcionamiento de tu proyecto.
2. **Entorno de Producción**
   - Donde se publica el trabajo final para que esté disponible para todos los usuarios.

# Configuración de Puertos para Desarrollo

Para evitar conflictos durante el desarrollo, cada alumno utilizará un puerto específico según su número de lista:

## ? Asignación de Puertos

- **Puerto base**: `5000`
- **Puerto personalizado**: `5000 + número_de_lista`

## ? Ejemplos de uso

| Alumno | N° Lista | Puerto | Comando |
|---|---|---|---|
| `jrobles` | 1 | **5001** | `python app.py --port 5001` |
| `mperez` | 2 | **5002** | `python app.py --port 5002` |

| Alumno | N° Lista | Puerto | Comando |
|---|---|---|---|
| `lgarcia` | 3 | **5003** | `python app.py --port 5003` |
| `ahernandez` | 4 | **5004** | `python app.py --port 5004` |

# ? Ejecutar la aplicación en entorno de desarollo

## Opción 1: Por argumento en línea de comandos

```
python app.py --port 5001 --debug
```

## Opción 2: Con variable de entorno

```
export FLASK_RUN_PORT=5001
export FLASK_DEBUG=1
python app.py
```

## Opción 3: En el mismo código

```
if __name__ == '__main__':
    app.run(debug=True, port=5001)  # Cambiar por tu puerto asignado
```

# ? Configuración de Gunicorn para el entorno de producción

## ? Requisitos previos

### Estructura de directorios requerida:

```
# Crear directorios necesarios
mkdir -p ~/flask_apps/.sockets
mkdir -p ~/flask_apps/mi_proyecto
```

## ? Ejecución con nohup (Solución temporal)

### Comando básico con nohup:

```
cd ~/flask_apps/mi_proyecto
nohup gunicorn --bind unix:/home/$USER/flask_apps/.sockets/mi_proyecto.sock app:app >
~/flask_apps/mi_proyecto.log 2>&1 &
```

### Comando completo con configuración optimizada:

```
cd ~/flask_apps/mi_proyecto
nohup gunicorn \
  --bind unix:/home/$USER/flask_apps/.sockets/mi_proyecto.sock \
  --workers 2 \
  --timeout 60 \
  --access-logfile ~/flask_apps/mi_proyecto_access.log \
  --error-logfile ~/flask_apps/mi_proyecto_error.log \
  app:app > ~/flask_apps/mi_proyecto.log 2>&1 &
```

### Verificar que está funcionando:

```
# Verificar el socket
ls -la ~/flask_apps/.sockets/mi_proyecto.sock

# Verificar procesos
ps aux | grep gunicorn

# Verificar logs
tail -f ~/flask_apps/mi_proyecto.log
```

### Detener la aplicación:

```
pkill -f "gunicorn.*mi_proyecto"
```

## ?? Configuración con systemd (Recomendado)

```
nano ~/.config/systemd/user/mi_proyecto.service
```

### Contenido del servicio systemd:

```
[Unit]
Description=Gunicorn instance para mi_proyecto Flask app
After=network.target

[Service]
WorkingDirectory=/home/%USER/flask_apps/mi_proyecto
ExecStart=/home/%USER/flask_apps/mi_proyecto/venv/bin/gunicorn \
          --bind unix:/home/%USER/flask_apps/.sockets/mi_proyecto.sock \
          --workers 2 \
          --timeout 60 \
```

```
          app:app
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
Restart=always

[Install]
WantedBy=default.target
```

## Comandos para gestionar el servicio:

```
# Recargar systemd
systemctl --user daemon-reload

# Habilitar inicio automático
systemctl --user enable mi_proyecto.service

# Iniciar servicio
systemctl --user start mi_proyecto.service

# Ver estado
systemctl --user status mi_proyecto.service

# Ver logs en tiempo real
journalctl --user-unit=mi_proyecto.service -f

# Reiniciar servicio
systemctl --user restart mi_proyecto.service

# Detener servicio
systemctl --user stop mi_proyecto.service
```

# Para observar errores en la aplicación

```
journalctl --user -u mi_proyecto.service -p 3
```

La opción -p 3 en journalctl significa "mostrar mensajes con nivel de prioridad 3 o superior

# ? Configuración específica para el servidor

```
# Dar permisos al directorio de sockets
chmod 755 ~/flask_apps
```

```
chmod 755 ~/flask_apps/.sockets

# Verificar que el socket se crea correctamente
ls -la ~/flask_apps/.sockets/
```

# Uso de loginctl para procesos persistentes

loginctl enable-linger permite a los procesos de un usuario permanecer corriendo a pesar de que ellos hayan cerrado su sesión.

Para habilitar el uso use el siguiente comando:

```
loginctl enable-linger <username>
```

Para deshabilitar su uso ejecute:

```
loginctl disable-linger <username>
```