# Laboratorio 1: Desarrollo de una API

Cátedra de Redes y Sistemas Distribuidos

## Objetivos

---

- Utilizar y diseñar Apis, que corresponden a la programación en la capa de aplicación de redes según el modelo OSI de redes
- Incorporar fluidez y buenos hábitos en la programación con python.
- Familiarizarse con herramientas propias de la industria (git, python, flask, postman, curl)

## Api para gestión de información

En esta actividad, vamos a crear una API robusta utilizando el lenguaje de programación Python y el framework Flask. El proyecto se divide en cuatro partes, comenzando con la configuración del entorno de programación en el que se desarrollará este laboratorio mediante la creación de un entorno virtual con virtualenv.

A continuación, construimos la API principal, destinada a administrar una base de datos de películas que incluirá un campo adicional para el género de cada obra cinematográfica.

La segunda parte del proyecto implica la implementación de la funcionalidad para consultar las películas por género, aprovechando las capacidades flexibles de Flask.

La tercera fase presenta una integración interesante al conectar nuestra API cinematográfica con una API externa de feriados, utilizando la información sobre las próximas fechas festivas para recomendar películas que se ajusten al género solicitado para ese día en particular.

Finalmente, la actividad concluye con la evaluación de la API, donde se considerarán aspectos cruciales como la respuesta de la API, códigos de estado HTTP, escalabilidad y seguridad. Se espera que los participantes realicen pruebas exhaustivas, asegurándose de que la API funcione eficientemente en diversas situaciones. La entrega del proyecto incluirá un informe detallado documentando el proceso de creación, decisiones de diseño, tecnologías utilizadas, resultados de la evaluación y posibles mejoras o expansiones para la API.

# Parte 1: Configuración del entorno e instalación de librerías

1. Antes que nada descarguen el repositorio de git o creen un repositorio de git y creen un archivo gitignore como este

2. Crea un entorno virtual de Python utilizando `venv`.

   ```
   python3 -m venv .venv
   ```

   En lugar de .venv puede ser cualquier nombre, pero es un estándar
3. Activa el entorno virtual

   ```
   source .venv/bin/activate
   ```

4. cuando lo quieras dejar de usar solo escribe

   ```
   deactivate
   ```

5. con el venv activo vamos a instalar las librerías de python necesarias.

   ```
   pip install -r requirements.txt
   ```

# Parte 2: Creación de una API con Flask

De los archivos del kickstarter revise el archivo main.py y proximo_feriado.py

1. Complete las funcionalidades de la api que se les entrega en main.py
   a. Lógica para buscar la película por su ID y devolver sus detalles
   b. Lógica para buscar la película por su ID y actualizar sus detalles
   c. Lógica para buscar la película por su ID y eliminarla.
2. Implemente la funcionalidad para devolver el listado de películas de un género específico.
3. Implemente la funcionalidad de búsqueda de películas, devolviendo la lista de películas que tengan determinado string en el título.
4. Implemente la funcionalidad de sugerir una película aleatoria.
5. Implemente la funcionalidad de sugerir una película aleatoria según género.

Recomendaciones, para probar el funcionamiento de la api de manera manual pueden utilizar curl, una herramienta de terminal que puede ser útil para ver que contesta la api mientras la van desarrollando. Cómo probar apis con curl

# Parte 3: Consumo de una API externa

Integra tu API con una API externa de feriados. Puedes utilizar la siguiente API: API de Feriados, en este link pueden encontrar toda la documentación de la api.

Se les entrega una aplicación de Python que utiliza la API de nolaborables.com.ar para obtener información sobre los feriados en Argentina. La aplicación busca y muestra el próximo feriado disponible.

a) Modifica el código para que agregar la opción de buscar feriados por tipo:

```
Unset
inamovible | trasladable | nolaborable | puente
```

b) En la api de películas utiliza la API de feriados para agregar la siguiente funcionalidad:

Obtener la próxima fecha de feriado y recomendar una película que se ajuste al género solicitado para ese día.

El comportamiento debería ser el siguiente

Pregunta en lenguaje humano “Sugerime una película de DRAMA para ver el próximo feriado”
Respuesta en lenguaje humano: “El próximo feriado es el xxxx con motivo yyyy , Te sugiero ver la pelicula de DRAMA <titulo de pelicula>

Obviamente no tienen que entregar este texto, el ejemplo es solo de comportamiento, deberán hacer las consultas usando las url de la api y programar las respuestas en formato json con los campos correspondientes.

# Parte 4: Evaluación de la API

1. Usar el archivo test.py y test_pytest.py
    a. Teniendo levantado el servidor (main.py) deberán ejecutar “python test.py” y analizar el funcionamiento de este test.
    b. desde terminal ejecutar pytest tsts_pytest.py y analizar el funcionamiento de este test
2. En ambos archivos agregar tests para las funcionalidades nuevas siguiendo las formas de programar test de cada uno de los dos archivos.
3. Replicar los request de test.py en postman, y analizar las respuestas. Usar la versión desktop de postman, sino no van a poder acceder a localhost. Para esto Cómo probar apis con postman

# Entrega

Junto con el código deberá entregar una presentación (tipo powerpoint) y un video de 10 +/-1 minutos. Les damos una estructura de base como idea, pero pueden modificarla/ ampliarla.

1. **Introducción al proyecto**:
    a. Presenta brevemente el contexto del proyecto y sus objetivos.
    b. Explicar que es una API REST y cómo funciona
2. **Configuración del entorno e instalación de librerías**:
    a. Explica por qué es importante utilizar un entorno virtual de programación.
    b. Muestra cómo configurar un entorno virtual de Python usando `venv`.

    c. Explica cómo activar y desactivar el entorno virtual.
    d. Guía sobre la instalación de las librerías necesarias utilizando `pip`.
3. **Creación de una API con Flask**:
    a. Describir funcionamiento de la api proporcionada y como funciona FLASK
    b. Describe las funcionalidades básicas de la API a desarrollar.
    c. Muestra cómo completaron las funcionalidades proporcionadas en el archivo `main.py`.
    d. Mostrar el funcionamiento mediante llamadas de curl
    e. Explica cada una de las partes de la API y su propósito.
4. **Consumo de una API externa**:
    a. Introduce la integración con la API externa de feriados.
    b. Explica cómo modificar el código existente para agregar funcionalidades relacionadas con los feriados.
    c. Demuestra cómo utilizar la API de feriados para obtener información y recomendar películas.
5. **Evaluación de la API**:
    a. Presenta el archivo `test.py`
    b. `test_pytest.py` y muestra cómo agregar tests para las nuevas funcionalidades. (explicar para qué sirve pytest y los mocs)
    c. Reproduce los mismos tests en Postman para verificar la funcionalidad desde el punto de vista del cliente.
    d. Explicar las diferencias entre las pruebas de test.py test_pytest.py y postman.
6. **Conclusiones y próximos pasos**:
    a. Resalta los resultados de la evaluación de la API.
    b. Propone posibles mejoras o ampliaciones para la API.
    c. Mejoras hechas
    d. Concluye el video resumiendo los principales puntos y agradeciendo por la atención.

# Criterios de evaluación del Laboratorio

## Evaluación del código

| Criterio Código | Descripción |
|---|---|
| **Funcionalidad de la API** | - Implementación correcta de endpoints básicos (buscar, actualizar, eliminar).<br>- Funcionalidades de búsqueda y sugerencia (aleatoria y por género). |
| **Integración de la API Externa** | - Integración básica y funcional con la API de feriados.<br>- Flujo coherente que asocia la fecha de feriado con la sugerencia de película. |

| | |
|---|---|
| **Buenas Prácticas y Código** | - Uso adecuado de entornos virtuales y control de versiones (git). Commits equitativos<br>- Código limpio, con docstrings, tipado y comentarios que faciliten la comprensión. |
| **Pruebas y Validación** | - Implementación de tests automatizados (test.py, pytest).<br>- Pruebas manuales replicadas correctamente con Postman/curl. |

## Evaluación del video

| Criterio Video | Descripción |
|---|---|
| **Cobertura del Desarrollo** | El video debe abordar todos los puntos del laboratorio: configuración del entorno, desarrollo de la API, integración externa, pruebas y documentación. Se deben incluir ejemplos prácticos y demostraciones visuales. |
| **Participación Equitativa** | Todos los integrantes deben participar de forma balanceada, demostrando colaboración y aportes significativos por parte de cada uno. |
| **Estructura y Organización** | La presentación debe tener una introducción, desarrollo organizado y conclusión clara que sintetice los aprendizajes. |
| **Claridad y Comunicación** | Uso de lenguaje técnico accesible, buena dicción y ritmo. Se valorará el apoyo de recursos visuales que faciliten la comprensión. |
| **Calidad de Producción y Tiempo** | El video debe tener audio y video de buena calidad, con recursos visuales legibles y editados profesionalmente. Debe respetar la duración estipulada (9 a 11 minutos). |

| | |
|---|---|
| **Profesionalismo** | Se espera una actitud profesional, con presencia y confianza en la exposición, acorde a un entorno académico y profesional. |

# Recomendaciones para el Laboratorio

**Planificá tu trabajo**

- Dividí el laboratorio en hitos semanales (configuración, desarrollo de funcionalidades, integración, pruebas y documentación).
- Estimá el tiempo que vas a dedicar a cada fase y realizá autoevaluaciones periódicas.

**Aprovechá los recursos y ejemplos proporcionados**

- Usá el código base de `main.py` y los tutoriales breves sobre Postman y curl.
- No dudes en consultar dudas y buscar referencias adicionales sobre Flask y buenas prácticas de programación.

**Documentá todo el proceso**

- Comentá y documentá tu código adecuadamente.
- Redactá un informe claro que explique tus decisiones y resuma los desafíos y soluciones implementadas.

**Prepará tus entregables con anticipación**

- Asegurate de que tu código funcione correctamente y que las pruebas sean repetibles.
- Organizá la presentación y el video siguiendo la estructura sugerida para comunicar de manera efectiva lo realizado.

**Revisá la rúbrica de evaluación**

- Usá la rúbrica para autoevaluarte y asegurarte de cubrir todos los aspectos requeridos.

¡Buena suerte con el laboratorio!