PROYECTO No. 0: NIVELACIÓN

OBJETIVOS

- Colocar en práctica los conocimientos necesarios para desarrollar una aplicación web, para reforzar sus habilidades y competencias en el desarrollo de software que son prerrequisitos del curso.
- Desarrollar una aplicación web en un framework de desarrollo ágil, en este caso particular Flask o FastAPI para Python y estándares de desarrollo web para el desarrollo de la capa de presentación.

RECOMENDACIONES, CONSIDERACIONES Y LECTURAS PREVIAS

En este taller, se deberá desarrollar una aplicación web. El backend será construido como una API REST, utilizando Python como lenguaje de programación y un framework de desarrollo web como Flask o FastAPI. La API REST debe ser consumida por el frontend de la aplicación. En cuanto al frontend, tienes libertad para utilizar las tecnologías de tu preferencia.

A continuación, se relacionan algunos enlaces y material de referencia para aprender cada framework.

- **Flask.** Para aprender este framework se recomienda visitar, revisar y utilizar los siguientes materiales.
  - Sitio oficial: https://flask.palletsprojects.com/
- **FastAPI.** Para aprender este framework se recomienda visitar, revisar y utilizar los siguientes materiales.
  - Sitio oficial: https://fastapi.tiangolo.com/es/tutorial/first-steps/

Adicionalmente, pueden incorporar frameworks MVC para el desarrollo de la capa de presentación (frontend), como es el caso de Angular o Vue, entre muchos otros. El diseño de las interfaces de usuario es de libre elección. Si no tienen experiencia con temas de frontend y no desea salir de Python como lenguaje de programación, pueden utilizar frameworks como Flet o Reflex.

- **Flet.** Para aprender este framework se recomienda visitar, revisar y utilizar los siguientes materiales.
  - Sitio oficial: https://flet.dev/docs/
- **Reflex.** Para aprender este framework se recomienda visitar, revisar y utilizar los siguientes materiales.

- Sitio oficial: https://reflex.dev/

La aplicación deberá ser empaquetada en contenedores Docker y ejecutada en una máquina virtual asignada a cada estudiante. En dicha máquina, se desplegarán todos los componentes de la aplicación.

SOLICITUD DE LA MÁQUINA VIRTUAL

Cada estudiante del curso tendrá acceso al laboratorio de aprendizaje de AWS Academy para aprovisionar una máquina virtual con el sistema operativo Ubuntu GNU/Linux y en ella ejecutar su aplicación web. En Bloqueneon, encontrará la información sobre cómo utilizar AWS Academy.

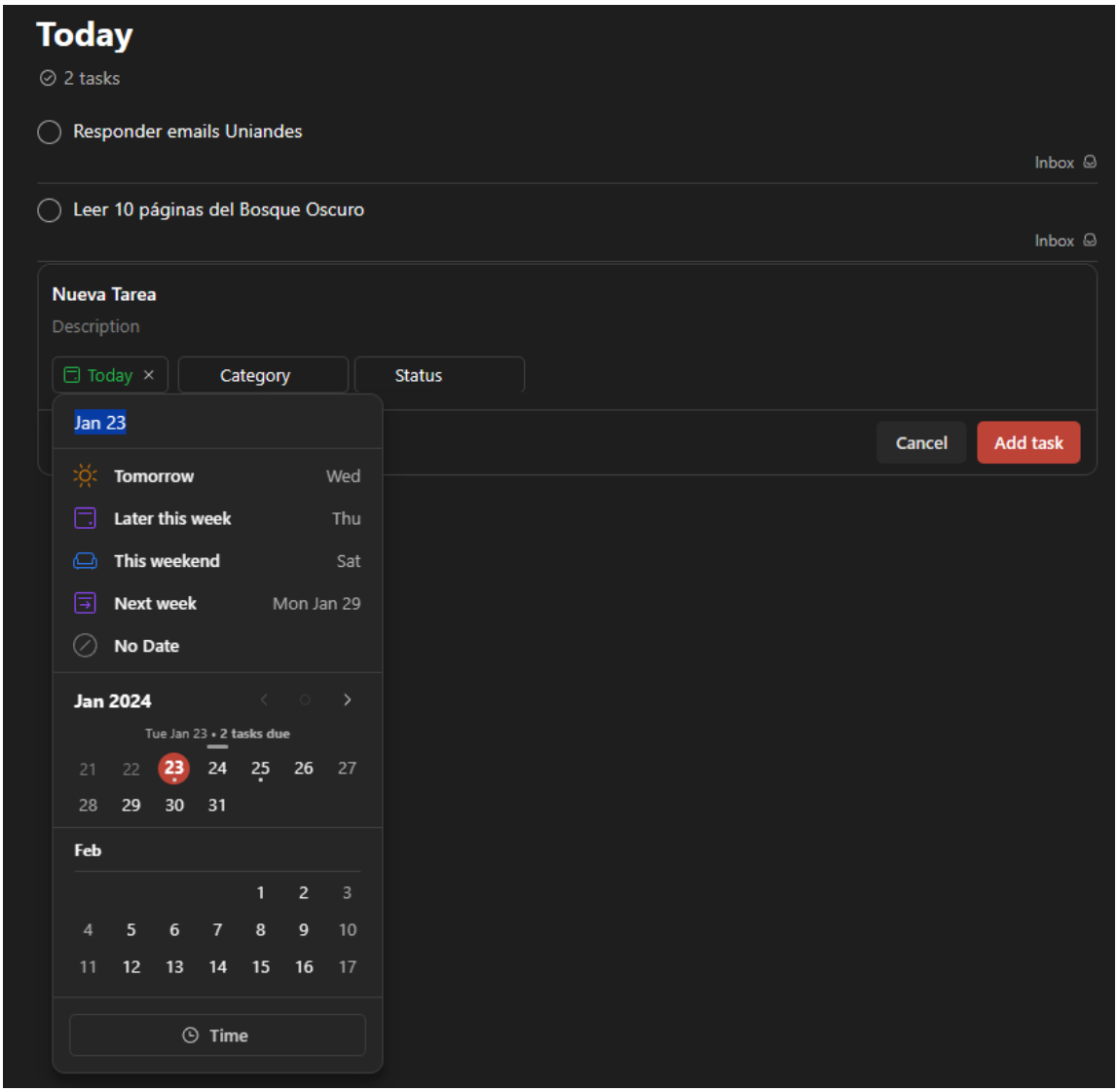
DESCRIPCIÓN DE LA APLICACIÓN WEB A DESARROLLAR

El objetivo de este proyecto es desarrollar una aplicación web de listas y gestión de tareas que permita a los usuarios crear, organizar y seguir el progreso de sus tareas diarias. La aplicación constará de tres componentes principales: una API REST para la lógica del negocio, una base de datos para almacenar la información de los usuarios y sus tareas, y una interfaz web para que los usuarios interactúen con la aplicación.

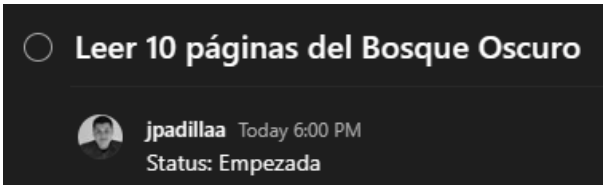
Funcionalidades Principales:

1. Autenticación de Usuarios:
    - ~~Creación de cuenta con usuario y contraseña.~~
    - Posibilidad de cargar una imagen de perfil.
    - ~~Si el usuario no carga una foto, el sistema colocará un icono por defecto.~~
    - ~~Inicio de sesión y cierre de sesión.~~
2. Gestión de Listas y Tareas:

- Captura de texto que representa una tarea.
- Organización de tareas en categorías (por ejemplo, Hogar, Trabajo, Urgente, entre otras).
- ~~Creación y eliminación de categorías.~~



3. Estados y Fechas:
    - Asignación de cada tarea a al menos un estado: Sin Empezar, Empezada, Finalizada.
    - Actualización de estados por parte del usuario.



    - Indicación de una fecha tentativa de finalización para cada tarea.
    - ~~Registro de la fecha de creación de la tarea para futura trazabilidad y analítica.~~
4. Operaciones sobre Tareas:
    - Creación y actualización de tareas.
    - Eliminación de tareas por parte del usuario.

Componentes de la Aplicación:

1. API REST (Flask o FastAPI):
    - Encargada de la lógica del negocio.
    - Gestión de usuarios, autenticación y autorización.
    - Manipulación de listas, tareas, categorías y estados.
2. Base de Datos:
    - ~~Almacenamiento de información de usuarios (cuentas y preferencias).~~
    - Registro de listas de tareas, tareas individuales y estados.
3. Interfaz Web:
    - Permite a los usuarios interactuar de forma amigable con la aplicación.
    - Visualización, creación, actualización y eliminación de tareas.
    - Interfaz para la gestión de categorías y estados.

Modelo Entidad - Relación:

A continuación, se sugiere un modelo entidad-relación (ER) que representa las entidades mínimas necesarias para la aplicación de gestión de tareas. Siéntase libre de modificarlo o proponer uno diferente según su conveniencia y experiencia.

Este modelo incluye las entidades principales (Usuario, Categoría, Tarea) y establece relaciones entre ellas. Cada tarea está asociada a un usuario y a una categoría. Asimismo, un usuario puede tener varias tareas en su lista.

Entidades

1. Usuario
    - ID (Clave Primaria)
    - Nombre de usuario
    - Contraseña
    - Imagen de perfil
    - Lista de tareas
2. Categoría
    - ID (Clave Primaria)
    - Nombre
    - Descripción
3. Tarea
    - ID (Clave Primaria)
    - Texto de la tarea
    - Fecha de creación
    - Fecha tentativa de finalización
    - Estado (Sin Empezar, Empezada, Finalizada)
    - ID_Categoria (Clave Foránea)

- ID_Usuario (Clave Foránea)

Relaciones

- Usuario (1) - (N) Tarea: Un usuario puede asignarse a una o varias tareas.
- Categoría (1) - (N) Tarea: Una categoría puede tener una o varias tareas.

API REST:

A continuación, se presenta la definición de los endpoints necesarios para el backend REST de la aplicación de gestión de tareas. Estos son los endpoints básicos que cubren las funcionalidades requeridas para la aplicación de gestión de tareas, siéntase libre de modificarlo o proponer uno diferente según su conveniencia y experiencia.

***Usuarios*:**

1. ~~Crear Usuario (POST /usuarios)~~
    - Entrada: Nombre de usuario, Contraseña, (Opcional) Imagen de perfil
    - Salida: Usuario creado
2. ~~Iniciar Sesión (POST /usuarios/iniciar-sesion)~~
    - Entrada: Nombre de usuario, Contraseña
    - Salida: Token de autenticación

***Categorías*:**

1. Crear Categoría (POST /categorias)
    - Entrada: Nombre, Descripción
    - Salida: Categoría creada
2. ~~Eliminar Categoría (DELETE /categorias/{id})~~
    - Salida: Confirmación de eliminación
3. ~~Obtener Lista de Categorías (GET /categorias)~~
    - Salida: Lista de categorías disponibles

***Tareas*:**

1. Crear Tarea (POST /tareas)
    - Entrada: Texto de la tarea, Fecha tentativa de finalización, ID de categoría
    - Salida: Tarea creada
2. Actualizar Tarea (PUT /tareas/{id})
    - Entrada: (Opcional) Texto de la tarea, (Opcional) Fecha tentativa de finalización, (Opcional) Estado de la tarea
    - Salida: Tarea actualizada

3. Eliminar Tarea (DELETE /tareas/{id})
    - Salida: Confirmación de eliminación
4. Obtener Lista de Tareas por Usuario (GET /tareas/usuario)
    - Salida: Lista de tareas del usuario
5. Obtener Tarea por ID (GET /tareas/{id})
    - Salida: Detalles de la tarea

Nota: Los endpoints del API REST deben ser documentados con Postman.

CONSIDERACIONES

El servidor web debe poder manejar solicitudes de forma concurrente. Se recomienda utilizar el siguiente stack tecnológico para el desarrollo de la entrega:

1. **Python 3.10 o superior**.
2. **Flask Framework 3.0.x o FastAPI 0.105.0**
3. **SQLAlchemy**: ORM, mapeador relacional de objetos que simplifica la interacción con una base de datos SQL.
4. **JWT**: Recomendamos utilizar JSON Web Tokens (Los tokens JWT son una opción popular para la autenticación en API REST).
5. **SQLite o PostgreSQL:** Motor open source de base de datos SQL.
6. **Gunicorn o Uvicorn:** servidor HTTP WSGI para Python y ambientes Unix.
7. **Postman**: Para documentar el API REST.
8. **Docker**: Para empaquetar la aplicación.
9. Para el **fronend** utilice la tecnología de su preferencia.

Recomendaciones adicionales:

- La página inicial (home) de la aplicación cuando el usuario no ha iniciado sesión es la página de inicio de sesión y cuando ya inició sesión es el listado de tareas.
- Tome las decisiones de análisis y diseño que considere conveniente.

EJECUCIÓN DE LA APLICACIÓN

Para la ejecución de la aplicación es necesario que cada estudiante configure el puerto y la IP por el que escucha el servidor, de la siguiente forma:

IP: 0.0.0.0 **Puerto:** 8080

- Al especificar la dirección IP 0.0.0.0 al exponer un servicio, estás indicando que el servicio debe escuchar en todas las interfaces de red disponibles en el sistema. En términos de configuración de servidores y redes, esto se interpreta como "escuchar en todas las direcciones IP disponibles".

- El 8080 es un puerto de ejemplo, puede utilizar el de su preferencia

Para comprobar que la aplicación puede ser accedida, abra el navegador, ingrese en el navegador la dirección ip de su máquina virtual (puede consultar la IP de su MV mediante el comando *ipconfig o ip a si está haciendo pruebas en local*) dos puntos el puerto 8080, por ejemplo:

http://192.168.0.1:8080

EQUIPO DE TRABAJO

Es un ejercicio de nivelación, por lo que este proyecto debe ser realizado de forma individual.

ESQUEMA DE EVALUACIÓN

La distribución de la calificación del taller está distribuida de la siguiente manera:

- Verificación funcional de los requerimientos de la aplicación en la sustentación: **100%**

Proyectos que no compilen o que no se puedan ejecutar durante la verificación tendrán como nota cero (0.0).

- Para el día de la entrega, la aplicación Web debe estar funcionando en una máquina virtual usando el Learner Lab de AWS Academy.

RUBRICA DE EVALUACIÓN

Criterios	Puntos
Backend (65 puntos)	
- Implementación de la API REST • Creación y Gestión de Categorías • Creación, Actualización y Eliminación de Tareas • Asignación de Estados y Fechas	50
- Integración con la Base de Datos	5
- Implementación de la Autenticación	10
Frontend (20 puntos)	
- Interfaz de Usuario	10
- Integración con Backend (Consumo de API)	5
- Autenticación de Usuarios (Frontend - Backend)	5
Documentación (7 puntos)	

- Documentación de la API con Postman	5
- Instrucciones de Despliegue y Uso	2
Despliegue (8 puntos)	
- Uso de Contenedores Docker	8
Total (100 puntos)	