**Get Talent – Semana 3 – Conociendo sobre API con FastAPI.**

**Actividad Práctica**

**Desarrollo de una API con FastAPI para un Gestor de Tareas**

**Descripción**

Se requiere implementar una API utilizando **FastAPI** que permita gestionar una lista de tareas. La API debe manejar datos en memoria (sin base de datos) y proporcionar funcionalidades básicas de un sistema de gestión de tareas. Los datos se almacenarán como una lista de diccionarios en Python, y las operaciones se realizarán directamente sobre esta estructura en tiempo de ejecución.

**Objetivos**

Implementar los siguientes endpoints que permitan interactuar con las tareas:

1. **GET**:

* Obtener la lista completa de tareas.
* Obtener los detalles de una tarea específica utilizando su ID.

1. **POST**:

* Crear una nueva tarea completando todos los campos obligatorios y, opcionalmente, los campos adicionales.

1. **PUT**:

* Actualizar los datos de una tarea existente utilizando su ID (por ejemplo, marcarla como completada o modificar su contenido).

1. **DELETE**:

* Eliminar una tarea utilizando su ID.

**Modelo de Datos**

El modelo de datos de cada tarea es el siguiente:

{

  "id": 1,

  "title": "Plan a birthday party",

  "description": "Organize a surprise birthday party for a friend.",

  "completed": false,

  "priority": 4,

  "due\_date": "2023-12-20T15:00:00Z",

  "category": "Personal",

  "subtasks": [

    {

      "id": 101,

      "title": "Book a venue",

      "completed": false

    },

    {

      "id": 102,

      "title": "Send invitations",

      "completed": true

    },

    {

      "id": 103,

      "title": "Order a cake",

      "completed": false

    }

  ]

}

**Descripción de los campos**

* **id**: Identificador único de la tarea (entero).
* **title**: Título de la tarea (cadena de texto, obligatorio).
* **description**: Descripción detallada de la tarea (cadena de texto, obligatorio).
* **completed**: Estado de completado de la tarea (booleano, predeterminado: false).
* **priority**: Nivel de prioridad (entero entre 1 y 5, opcional).
* **due\_date**: Fecha límite para completar la tarea en formato ISO 8601 (opcional).
* **category**: Categoría de la tarea, como "Trabajo" o "Personal" (cadena de texto, opcional).
* **subtasks**: Lista de subtareas asociadas a la tarea principal, donde cada subtarea tiene:
* **id**: Identificador único de la subtarea (entero).
* **title**: Título de la subtarea (cadena de texto).
* **completed**: Estado de completado de la subtarea (booleano).

**Requisitos Técnicos**

1. Utilizar **FastAPI** para implementar la API.
2. Manejar los datos en memoria utilizando una lista de diccionarios en Python.
3. Crear rutas para las operaciones CRUD mencionadas.
4. Validar y manejar errores adecuadamente (por ejemplo, devolver un error 404 si se intenta acceder a una tarea inexistente).