



Distribución de Tareas del Proyecto

Objetivo

Desarrollar una aplicación web para gestionar talleres de formación profesional. La aplicación permitirá a estudiantes y administradores gestionar talleres como cursos técnicos, capacitaciones prácticas y programas de actualización profesional. Además, se implementará una API RESTful para interactuar con los datos de los talleres. El trabajo será realizado en grupos, y se debe entregar un prototipo funcional.

Requisitos funcionales

◇ Gestión de Talleres de Formación Profesional

- Ver lista de talleres
 - Mostrar una lista de talleres programados con nombre, fecha, hora, lugar, y tipo de actividad.
- Registrar un taller (solo administradores)
 - Crear un nuevo taller con nombre, descripción, fecha, hora, lugar y categoría.
- Modificar un taller (solo administradores)
 - Editar los detalles de un taller ya registrado.
- Cancelar un taller (solo administradores)
 - Eliminar un taller que ya no se realizará.
- Registrarse a un taller (solo estudiantes)
 - Los estudiantes deben poder inscribirse para participar en los talleres de su elección.

◇ API RESTful

- GET /workshops → Obtener todos los talleres disponibles
- GET /workshops/{id} → Obtener los detalles de un taller específico
- POST /workshops → Crear un nuevo taller (solo administradores)
- PUT /workshops/{id} → Modificar un taller existente (solo administradores)
- DELETE /workshops/{id} → Eliminar un taller (solo administradores)
- POST /workshops/{id}/register → Registrar a un estudiante en un taller

La API debe ser implementada usando Flask, devolver datos en JSON y manejar correctamente los códigos de estado HTTP.

◊ Interfaz Web

Interfaz para estudiantes: ver talleres y registrarse.



Panel para administradores: gestionar la creación, edición y cancelación de talleres.

Implementar formularios y tablas usando HTML, CSS y JavaScript. Se puede utilizar Bootstrap o React si se desea.

Trabajo en Grupo

- Cada grupo debe dividir tareas entre backend, frontend, base de datos y documentación.
- Usar Git para el control de versiones y colaboración.
- Se recomienda el uso de pull requests y revisión de código entre los miembros.

Extras Opcionales

-  Pruebas unitarias (+15 pts): pruebas que validen al menos los endpoints principales.
-  Documentación técnica detallada (+15 pts): descripción de arquitectura, base de datos y flujo de desarrollo.

Instrucciones adicionales

- Backend: Flask (con Flask-RESTful o similar)
- Frontend: HTML, CSS, JS (con Bootstrap o React si se desea)
- Base de datos: PostgreSQL, MongoDB o MySQL
- Autenticación: opcional con JWT o sesiones para admins
- README.md: instalación, ejecución y uso de la aplicación
- Documentación técnica: estructura del proyecto, diseño de la base de datos, arquitectura de la API, instrucciones de despliegue.



Distribución de Tareas

- Base de Datos (MongoDB):
 - Alec
 - Gil
 - Carlos 2
- Diagramas de base de datos:
 - Aaron
- Frontend:
 - Ana
 - Sofia
 - Daniela
- Backend:
 - Franklin
 - Euris
 - Stewart
- Docker:
 - Diego
- Documentación del Código:
 - Alonso
 - Abel
- Readme.md:
 - Veronica
 - Carlos Contreras
- Autenticación con JWT:
 - Daniel
 - Esteban
- Faltan asignar:
 - Raul
 - Emilio