# PRÁCTICA 2 Desarrollo Web Integral

## Objetivo de la práctica

Establecer estrategias eficientes y profesionales para el manejo del control de versiones en un proyecto de software, definiendo normas de trabajo colaborativo que garanticen trazabilidad, organización, seguridad y calidad del código fuente.

## Temas a aplicar

- Nomenclatura de ramas.
- Acceso a usuarios.
- Políticas de combinación de ramas (merge).
- Políticas de Pull Requests.
- Control del versionamiento del código fuente.

## Instrucciones

### Parte 1: Configuración del entorno

1. Crea un repositorio en GitHub (o GitLab o Bitbucket).
2. Define un archivo `README.md` con el nombre del proyecto y una breve descripción.

### Parte 2: Estrategia de trabajo colaborativo

#### 1. Establecer una convención de nomenclatura de ramas

- Define una guía clara para nombrar ramas, por ejemplo:
  - `feature/` para nuevas funcionalidades.
  - `bugfix/` para correcciones.
  - `hotfix/` para emergencias.
  - `release/` para preparar una nueva versión.
- Escribe esta convención en un archivo `docs/nomenclatura.md`.

#### 2. Definir políticas de acceso a usuarios

- Simula un equipo de 3 roles: líder del proyecto, desarrollador y colaborador externo.
- Especifica en un documento:
  - Quién puede hacer *push* directamente a ramas protegidas.
  - Quién debe trabajar con Pull Requests.
  - Qué permisos tendrá cada rol.
- Guardar como `docs/acceso_usuarios.md`.

**3. Establecer políticas de combinación de ramas**

- Indica si se permite `merge`, `squash merge` o `rebase merge`.
- Define si es obligatorio pasar pruebas automáticas antes del *merge*.
- Establece la necesidad de revisiones de código.
- Guardar como `docs/politicas_merge.md`.

**4. Definir políticas de Pull Request**

- Especifica:
    - Número mínimo de revisores.
    - Proceso de validación (pruebas, revisión de estilo, documentación).
    - Tiempo máximo de respuesta a una PR.
- Guardar como `docs/politicas_pr.md`.

### Parte 3: Aplicación práctica del control de versiones

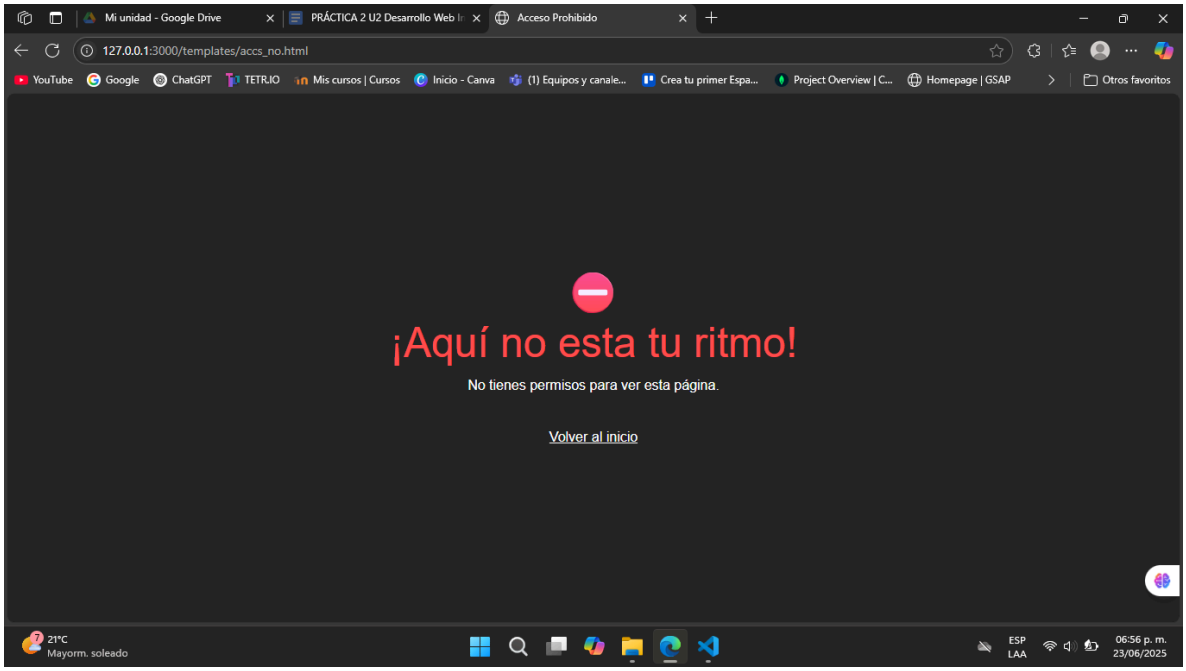
1. Crea una rama `feature/mi-funcionalidad` y desarrolla una función ficticia (puede ser un script simple en JS, Python, etc.).
2. Sube tu código a la nueva rama.
3. Genera un Pull Request siguiendo las políticas definidas.
4. Simula una revisión con otro compañero o desde otra cuenta, agregando comentarios o aprobaciones.
5. Realiza el merge solo si cumple las condiciones establecidas.

## Entregables

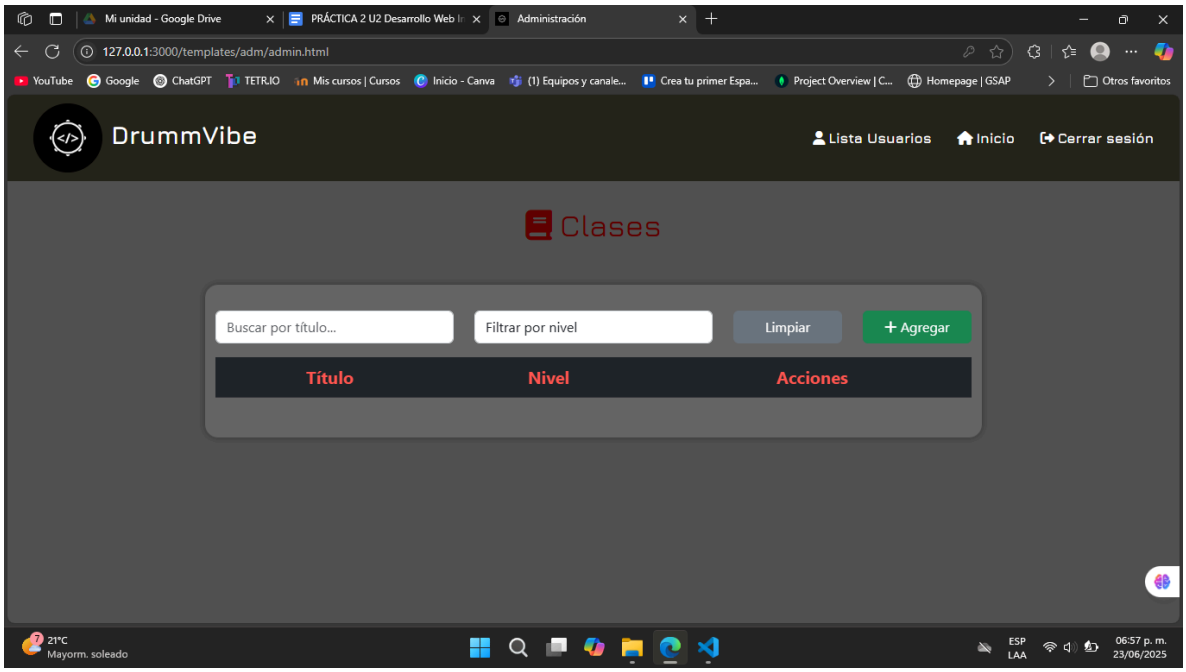
- Repositorio con:
    - Archivos de políticas (`docs/*.md`).
    - Pull Request documentada.
    - Historial de ramas y merges conforme a las estrategias.

Repositorio: Jose-Hernandez-45/proyecto-control-versiones at feature/mi-funcionalidad

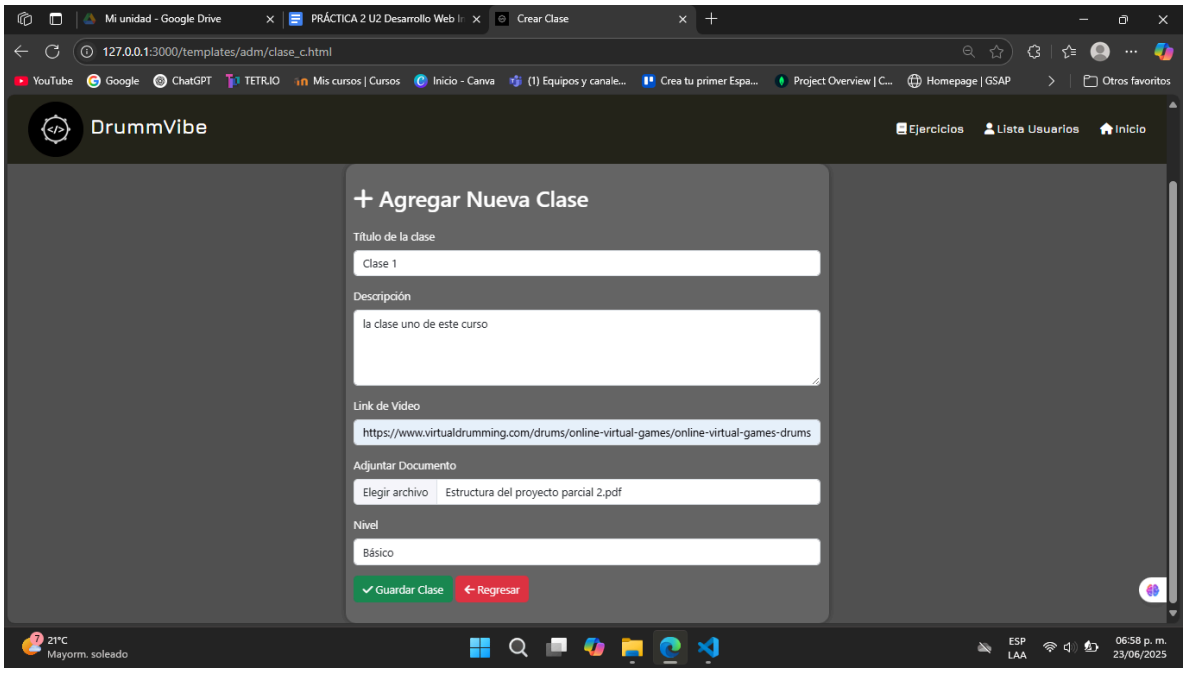
Avances de proyecto



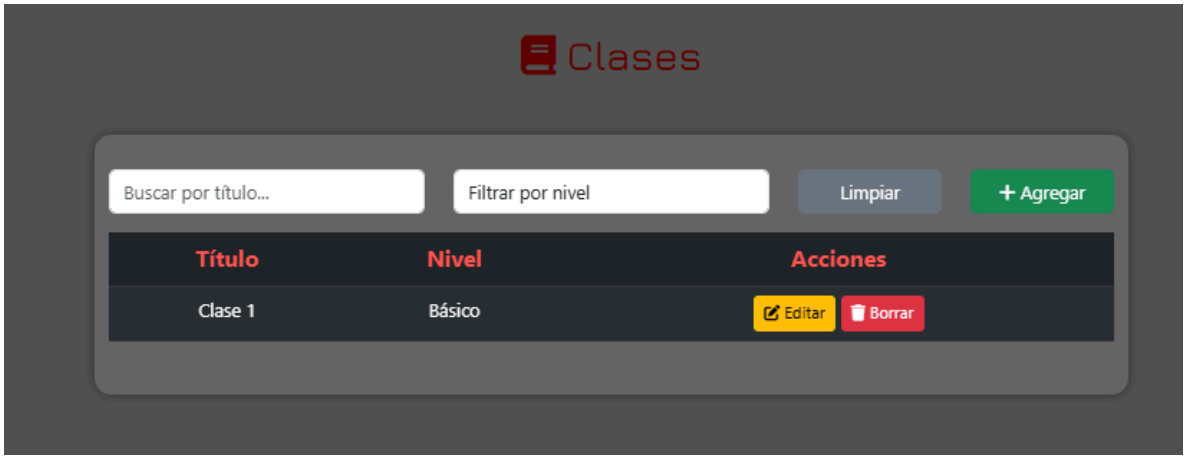

se agregó una pantalla para usuarios que no son administradores e intentan ingresar al administrador.

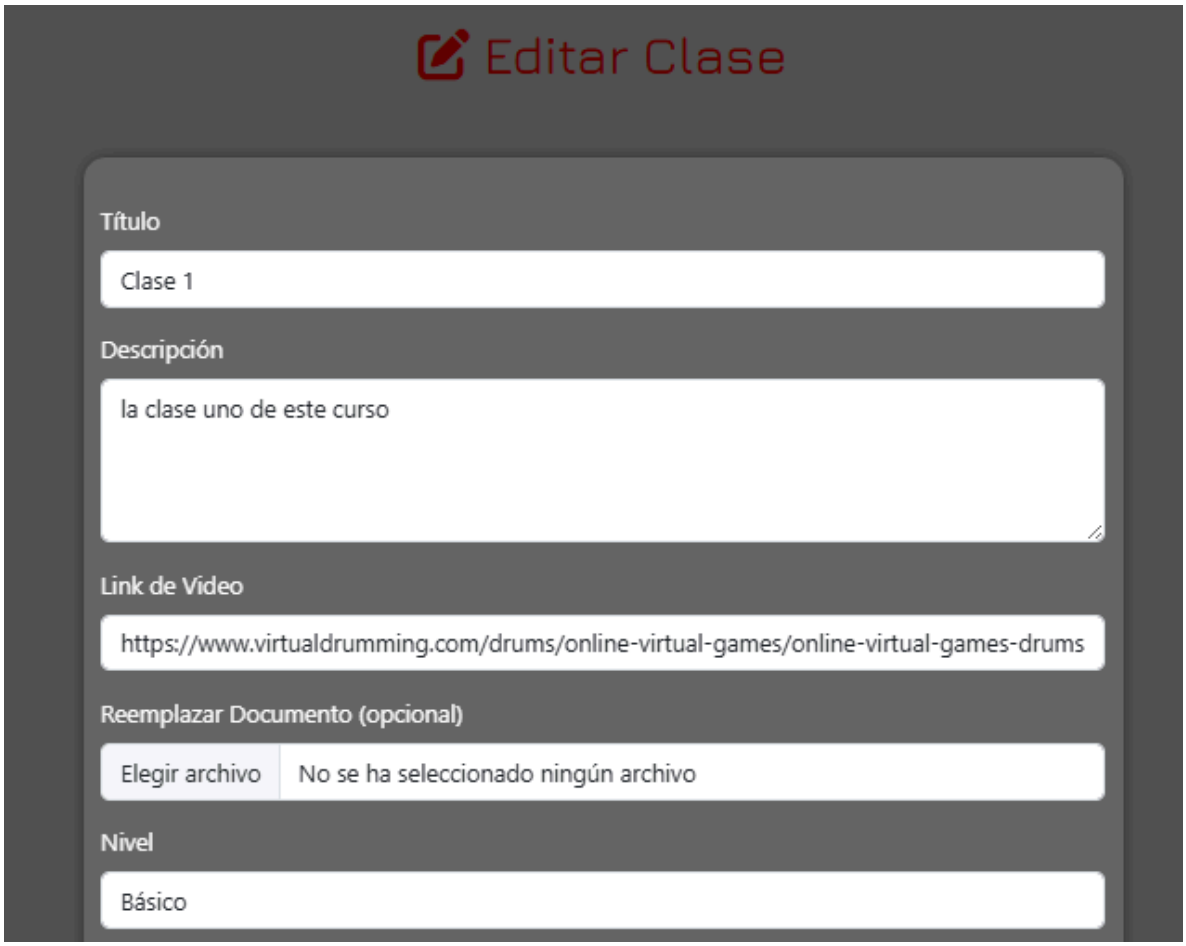



Se agregó el administrador de clases, para agregar, editar y eliminar clases.

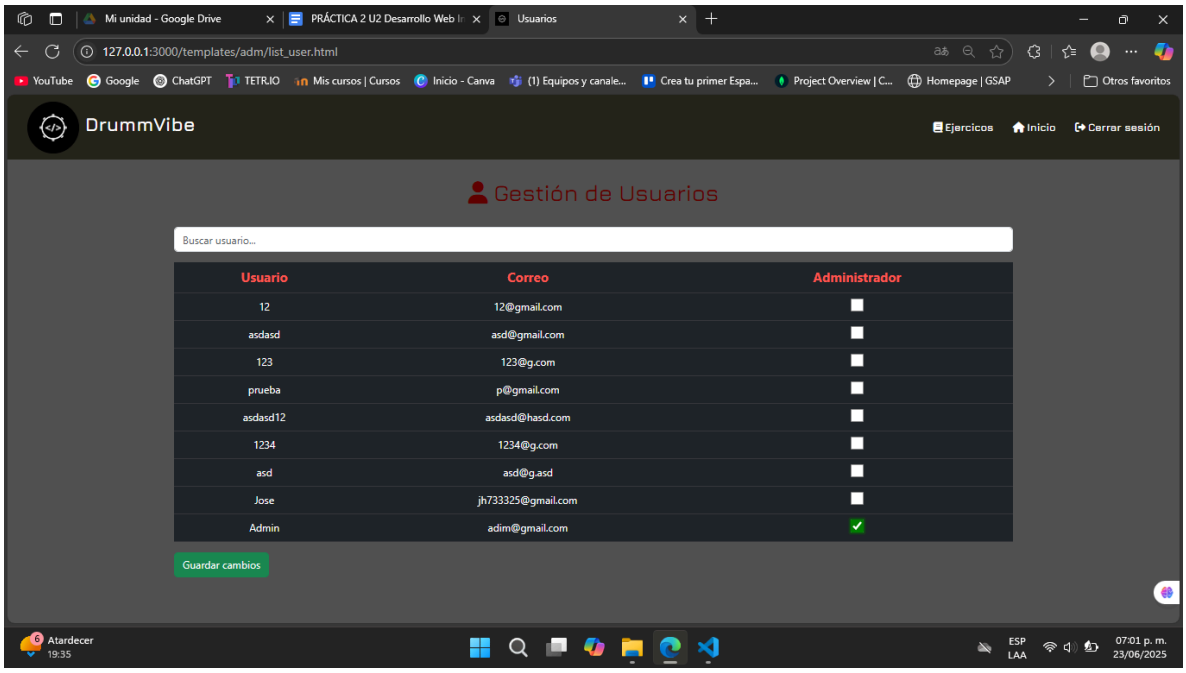



Apartado de agregar clases.






edición de clases



Gestión de usuarios, los usuarios marcados son administradores.