“Año de la recuperación y consolidación de la economía peruana”

HERRAMIENTAS DE DESARROLLO

AVANCE PROYECTO FINAL 1 (APF1)

SISTEMA DE GESTIÓN DE RECURSOS EDUCATIVOS

ASESOR DEL CURSO: HUARCAYA ALMEYDA, REMIGIO SANTOS

SECCIÓN: 25083

AUTORES:

CHIQUINTA VERAMENDI, SERGIO ANDRÉ (U22201712)

CHOCCE CALDERÓN, OSCAR MANUEL (U22230523)

PALACIOS VALENCIA, SEBASTIAN MATIAS (U22235341)

INGENIERÍA DE SOFTWARE / SISTEMAS E INFORMÁTICA

LIMA, ABRIL DE 2025

ÍNDICE

1. INTRODUCCIÓN 3
1.1. Objetivos del proyecto 3
1.2. Tecnologías utilizadas 3
2. DESCRIPCIÓN DEL PROYECTO 3
2.1. Resumen 3
2.2. Requerimientos 4
2.2.1. Funcionales 4
2.2.1. No Funcionales 4
2.3. Flujo de Trabajo Colaborativo 4
3. PROCEDIMIENTOS Y CONFIGURACIONES DEL PROYECTO 5
3.1. Configuración Inicial del Repositorio 5
3.1.1. Creación del Repositorio de manera local 5
3.1.2. Creación del README.md 5
3.1.3. Creación del .gitignore 5
3.1.4. Creación del Repositorio remoto: 5
3.1.5 Añadir colaboradores al Repositorio remoto: 6
3.2. Estructura del proyecto 7
3.3. Flujo de trabajo de ramas (branching) 9
3.4. Código del proyecto 11
3.4.1. Enlace al Repositorio GitHub 11
3.4.2. Base de Datos 11
4. PROCEDIMIENTOS DE CONTROL DE VERSIONES 12
4.1. Realización de Commits 12
4.1.1. Commits realizados en la rama de Sergio 12
4.1.2. Commits realizados en la rama de Sebastian 12
4.1.3. Commits realizados en la rama de Oscar 13
4.2. Pull Request y revisión de código 13
4.2.1. Creación y Gestión de Pull Requests 13
4.2.2. Proceso de Revisión de Código 14
4.2.3. Ejemplo de un Pull Request en nuestro proyecto 14
4.3. Issues y Control del proyecto usando Trello 15
4.3.1. Gestión estructurada de Issues 15
4.3.2. Construir el tablero de Proyecto 16

1. INTRODUCCIÓN

1.1. Objetivos del proyecto

El objetivo principal de este proyecto es diseñar y construir una aplicación web que funcione como un sistema de gestión de recursos educativos digitales. Esta plataforma facilitará la creación, organización y distribución de materiales académicos por parte de los docentes de la UTP, permitiendo a los estudiantes acceder y descargar dichos contenidos.

Además, se busca fomentar el trabajo colaborativo mediante el uso de herramientas modernas de desarrollo, control de versiones con Git/GitHub y metodologías de desarrollo web actuales.

1.2. Tecnologías utilizadas

Categoría del Proyecto	Herramienta / Lenguaje
Frontend	React, Bootstrap
Backend	Node.js (LTS) v20.15.0, Express
Base de Datos	MySQL (XAMPP)
Control de versiones	Git, Github
Lenguajes	HTML5, CSS3, JavaScript (JSX), TypeScript (TSX), SQL
Otros	JWT para autenticación, herramientas de desarrollo accesible y validación de formularios.

2. DESCRIPCIÓN DEL PROYECTO

2.1. Resumen

El proyecto consiste en desarrollar una plataforma web educativa que actúe como repositorio centralizado de recursos académicos. Esta plataforma estará dirigida a los docentes de la UTP, quienes podrán subir, organizar y compartir distintos tipos de materiales (documentos, videos, enlaces, ejercicios) clasificados por curso o categoría. Los estudiantes podrán acceder a estos recursos, visualizarlos o descargarlos según su rol de usuario.

El sistema contará con funcionalidades diferenciadas para administradores, docentes y estudiantes, promoviendo así una gestión eficiente y segura de los contenidos educativos.

2.2. Requerimientos

2.2.1. Funcionales

- **RF01:** Registro y autenticación de usuarios con roles (administrador, docente, estudiante).
- **RF02:** Gestión de perfiles de usuario: visualización y edición de datos personales.
- **RF03:** Subida y gestión de recursos educativos por parte de los docentes.
- **RF04:** Organización de recursos por categorías y cursos.
- **RF05:** Búsqueda avanzada de materiales mediante filtros y búsqueda textual.
- **RF06:** Descarga de recursos disponibles para los estudiantes.
- **RF07:** Panel administrativo con: estadísticas de uso de los recursos, moderación de contenido subido, gestión de usuarios y roles.

2.2.1. No Funcionales

- **RNF01:** Interfaces web responsivas y accesibles (según los estándares WCAG).
- **RNF02:** Validación de formularios en el cliente (frontend) y en el servidor (backend).
- **RNF03:** Seguridad: uso de autenticación con JWT, prevención de inyecciones SQL y XSS.
- **RNF04:** Escalabilidad: diseño pensado para soportar miles de recursos y usuarios.
- **RNF05:** Documentación clara del código fuente mediante estándares JSDoc.
- **RNF06:** Uso de procedimientos almacenados en MySQL para mejorar el rendimiento de operaciones frecuentes.

2.3. Flujo de Trabajo Colaborativo

El desarrollo del proyecto se realizó utilizando un enfoque colaborativo mediante la plataforma GitHub, lo cual permitió una gestión eficiente del código fuente y un trabajo en equipo organizado.

Herramientas empleadas:

- Git: Control de versiones distribuido para llevar seguimiento de los cambios en el proyecto.
- GitHub: Repositorio remoto donde se almacenó el código y se coordinaron las tareas colaborativas.

Metodología aplicada:

Se creó un repositorio principal en GitHub desde el cual se clonó el proyecto de manera local por cada integrante.

Se utilizó una rama principal (master) y ramas secundarias específicas por funcionalidad, como por ejemplo:

- feature/login-usuario-sergio
- feature/gestión-recursos-óscar
- feature/descarga-estudiantes-sebastián

Cada integrante trabajó en su respectiva rama para evitar conflictos y mantener el código modular.

Se aplicaron comandos como git add, git commit, git push y git pull para gestionar el avance del proyecto.

Una vez completada una funcionalidad, se realizaban pull requests (PR) para revisión y fusión al repositorio principal.

Se fomento el uso de mensajes de commit descriptivos para mejorar la trazabilidad de los cambios.

Beneficios del flujo colaborativo:

- Prevención de pérdida de información o sobrescritura de código.
- Permite trabajar simultáneamente sin interferencias entre compañeros.
- Fomenta buenas prácticas de desarrollo profesional y trabajo en equipo.

3. PROCEDIMIENTOS Y CONFIGURACIONES DEL PROYECTO

3.1. Configuración Inicial del Repositorio

Para iniciar el desarrollo del proyecto se creó un entorno de trabajo organizado y controlado con Git y GitHub, siguiendo los siguientes pasos:

3.1.1. Creación del Repositorio de manera local

Se creó una carpeta del proyecto (Educa-UTP) y se inicializó como repositorio de Git desde la terminal con:

- git init

3.1.2. Creación del README.md

Se creó un archivo README.md para incluir información básica del primer avance del proyecto, desde la terminal con:

- New-Item README.md -ItemType File

3.1.3. Creación del .gitignore

Se configuró un archivo .gitignore para excluir archivos sensibles y carpetas innecesarias como:

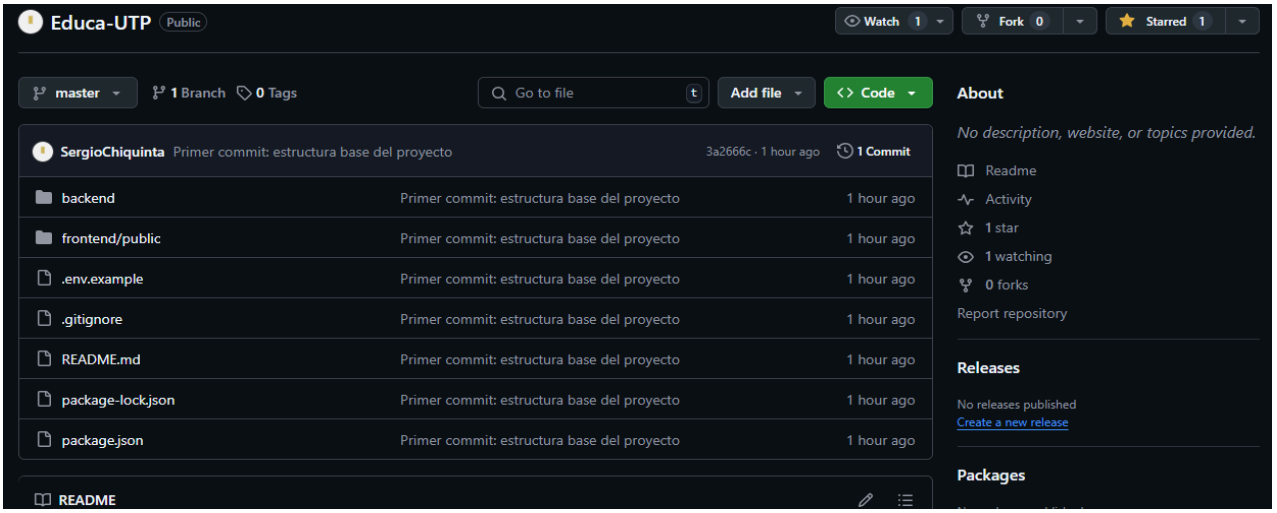
- node_modules
- .env
- DS_store

3.1.4. Creación del Repositorio remoto:

Se creó un repositorio remoto en GitHub y se enlazó con el repositorio local:

- git add.
- git commit “Primer commit: estructura base del proyecto”
- git remote add origin https://github.com/SergioChiquinta/Educa-UTP.git

- git push -u origin master



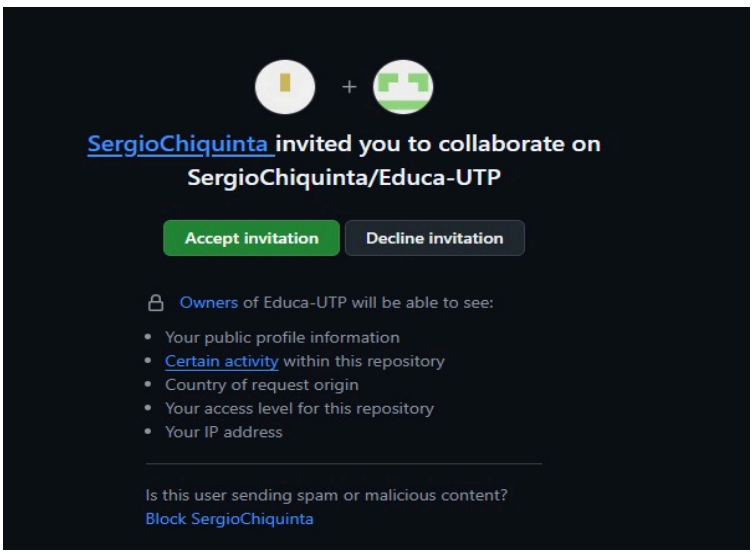
3.1.5 Añadir colaboradores al Repositorio remoto:

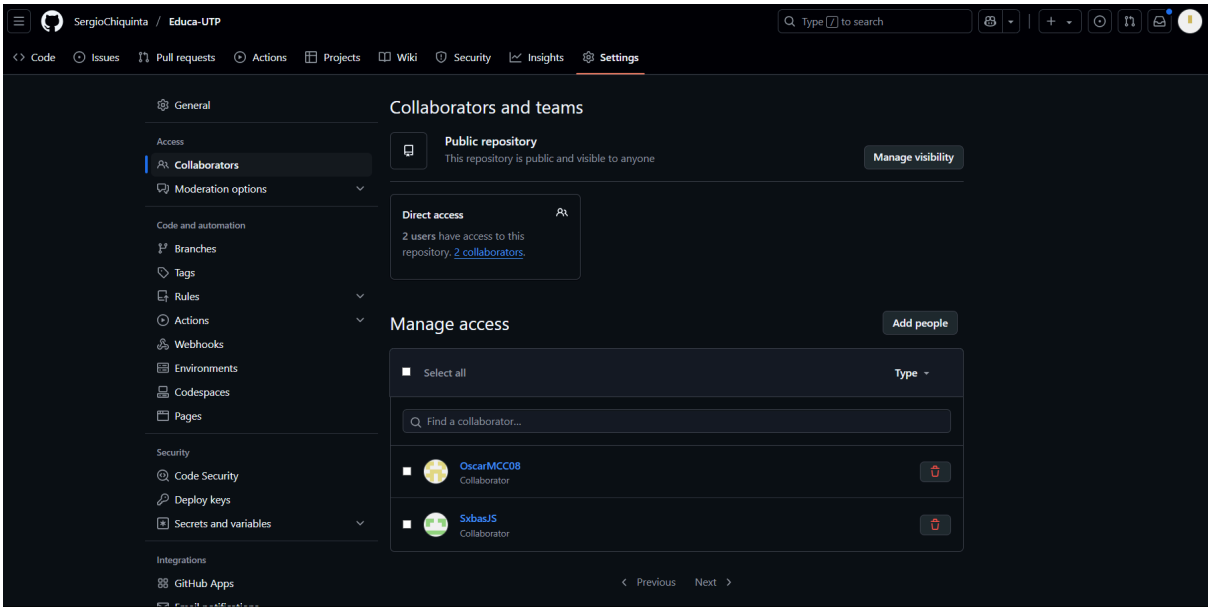
Para fomentar el trabajo colaborativo y el control de versiones entre los integrantes del equipo, se añadieron colaboradores directamente desde la plataforma de GitHub. Los pasos seguidos fueron:

1. Ingresar al Repositorio (https://github.com/SergioChiquinta/Educa-UTP.git).
2. Ir a la pestaña “Settings”.
3. Seleccionar “Collaborators” en la sección de “Access”.
4. Invitar a los integrantes del equipo.

   Nota: Se añadió a cada colaborador usando su nombre de usuario de GitHub, otorgándoles permisos de escritura (Write) para que puedan clonar, crear ramas, subir cambios y colaborar activamente en el proyecto.

5. Los integrantes deben aceptar la invitación.

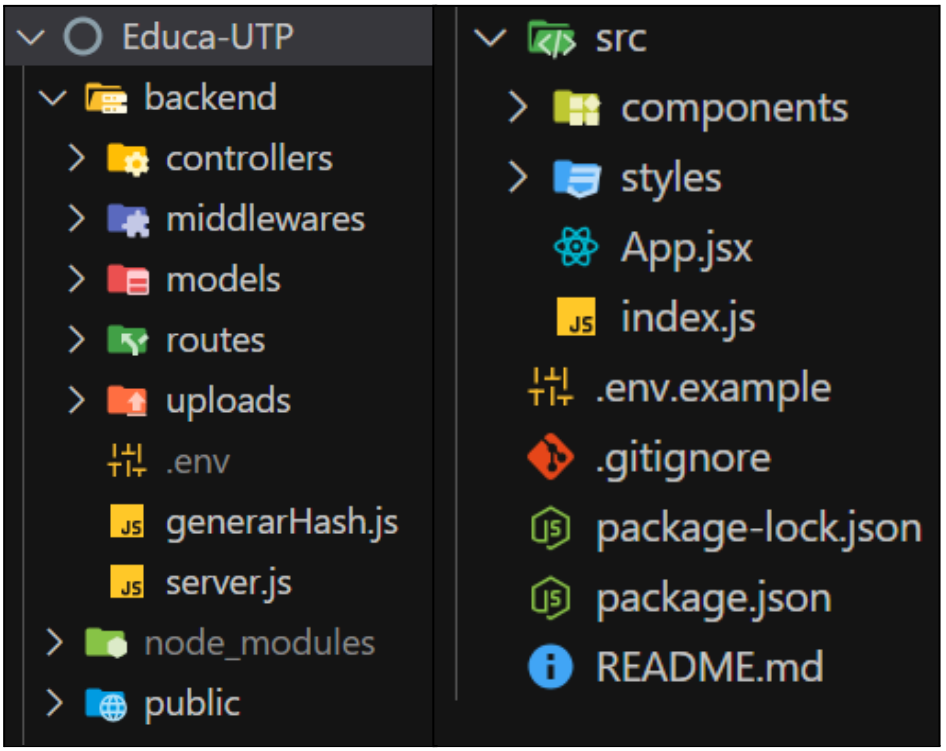




Este procedimiento permite mantener la trazabilidad de los aportes individuales y fomenta un flujo de trabajo ágil con control de versiones y colaboración distribuida.

3.2. Estructura del proyecto

La organización de archivos y carpetas en el proyecto Educa-UTP sigue una estructura clara y modular que permite separar de forma lógica el Backend y el Frontend, facilitando el mantenimiento y la escalabilidad de la aplicación:



- **backend/**

  Carpeta donde se desarrolla toda la lógica del servidor con Node.js y Express, encargándose de procesar peticiones, conectarse a la base de datos, validar datos y gestionar la seguridad de la aplicación:

  - **controllers/:** Lógica de negocio para responder a las diferentes solicitudes de la API.
  - **middlewares/:** Funciones intermedias que procesan y validan solicitudes antes de llegar a los controladores.
  - **models/:** Modelos que representan las tablas de MySQL, facilitando el acceso y manipulación de los datos.
  - **routes/:** Definición de las rutas de la API RESTful, que conectan las solicitudes HTTP con los controladores correspondientes.
  - **uploads/:** Carpeta donde se guardarán archivos multimedia de la base de datos (fotos de perfil, imágenes de recursos estudiantiles, etc.).
  - **server.js:** Archivo principal del servidor que inicia Node.js, configura los middlewares y carga las rutas de la API.
  - **.env:** Archivo que contiene los nombres de las variables de entorno necesarias (como URL de conexión a MySQL, puertos, claves JWT, etc.), para facilitar la configuración en distintos entornos de desarrollo.
  - **generarHash.js:** Archivo que nos permite encriptar una contraseña que nosotros indiquemos, Sólo para cuando necesitemos agregar nuevas cuentas en la base de datos, no está afiliado al programa como tal

- **public/**

  Contiene archivos estáticos accesibles directamente como index.html.

- **src/**

  Será la carpeta principal de React, donde se alojarán los componentes, páginas, rutas de Frontend, servicios de conexión con el Backend, y toda la lógica del cliente.

  - components/: Carpeta donde se distribuirán los componentes.
  - styles/: Carpeta donde se almacenarán los archivos .css de estilos.
  - App.jsx: Archivo filtrador que obtiene los componentes alineados.
  - index.js: Archivo principal que incluye todo lo introducido en App.jsx.

- **.env.example**

  Archivo que contiene los nombres de las variables de entorno necesarias (como URL de conexión a MySQL, puertos, claves JWT, etc.), para facilitar la configuración en distintos entornos de desarrollo.

- **.gitignore**

  Define qué archivos no deben subirse al repositorio remoto (por ejemplo, node_modules/, archivos .env, carpetas de builds, etc.).

- **README.md**

Archivo simplificado de documentación que explica los objetivos del proyecto, las instrucciones de instalación, ejecución y cualquier otra información relevante para usuarios o desarrolladores.

3.3. Flujo de trabajo de ramas (branching)

Para mantener un desarrollo ordenado y colaborativo en el proyecto Educa-UTP, se adopta una estrategia de ramas basada en GitFlow, adaptada de acuerdo con las necesidades del equipo y del proyecto.

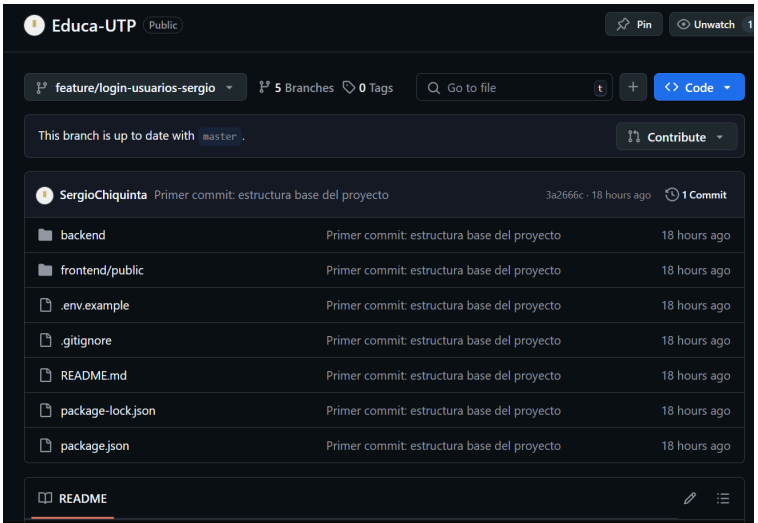
Estrategia de Branching y Buenas Prácticas:

- **master:** Rama principal donde se mantiene el código en producción. Solo se actualiza tras pasar todas las pruebas necesarias.
    - Se crea automáticamente con el proyecto (git init), puede ser main o master.
- **develop:** Rama de integración donde se combinan las nuevas funcionalidades y se realizan pruebas antes de subir los cambios a master.
    - git checkout -b develop (Crear en repositorio local)
    - git push -u origin develop (Vincular en repositorio remoto)
- **feature/:** Ramas creadas para el desarrollo de nuevas funcionalidades. Cada funcionalidad debe crearse en su propia rama específica:
    - git checkout feature/nueva-funcionalidad (Crear en repositorio local)
    - git push -u origin feature/nueva-funcionalidad (Vincular en repositorio remoto)

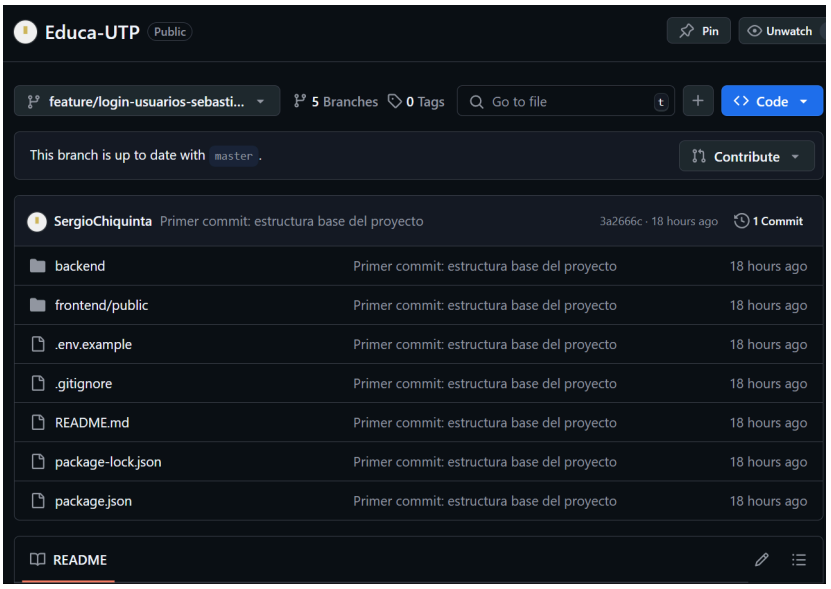
  Nota: Después de hacer “git push -u origin **rama**”, ya no tendrás que escribir todo eso otra vez, solo harás en el futuro dentro de la rama:
    - git push
    - git pull

  Ejemplos en Educa-UTP:

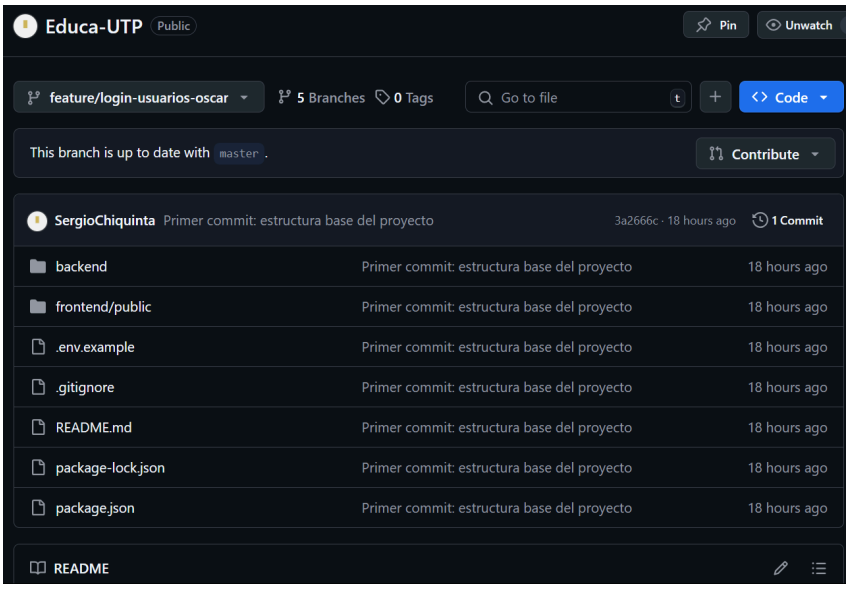
    - feature/login-usuarios-sergio



- feature/login-usuarios-sebastian



- feature/login-usuarios-oscar



Nota: Estas ramas se crean a partir de develop y se deben fusionar nuevamente a develop una vez terminadas.

- **bugfix/:** Ramas destinadas a corregir errores detectados en develop o en producción antes de una liberación.
- **release/:** Ramas que preparan nuevas versiones estables del sistema. Se usan para realizar pruebas finales, pequeñas correcciones, y tareas de documentación antes de pasar a master.

Comandos básicos y operaciones de ramas:

- Crear una nueva rama de funcionalidad
  - git checkout -b **feature/nombre-funcionalidad**
- Cambiar de rama
  - git checkout **nombre-rama**
- Actualizar rama actual con los últimos cambios en develop
  - git pull origin develop
- Visualizar las ramas disponibles
  - git branch -a
- Visualizar el historial de ramas (formato gráfico)
  - git log --graph --oneline --all

3.4. Código del proyecto

3.4.1. Enlace al Repositorio GitHub

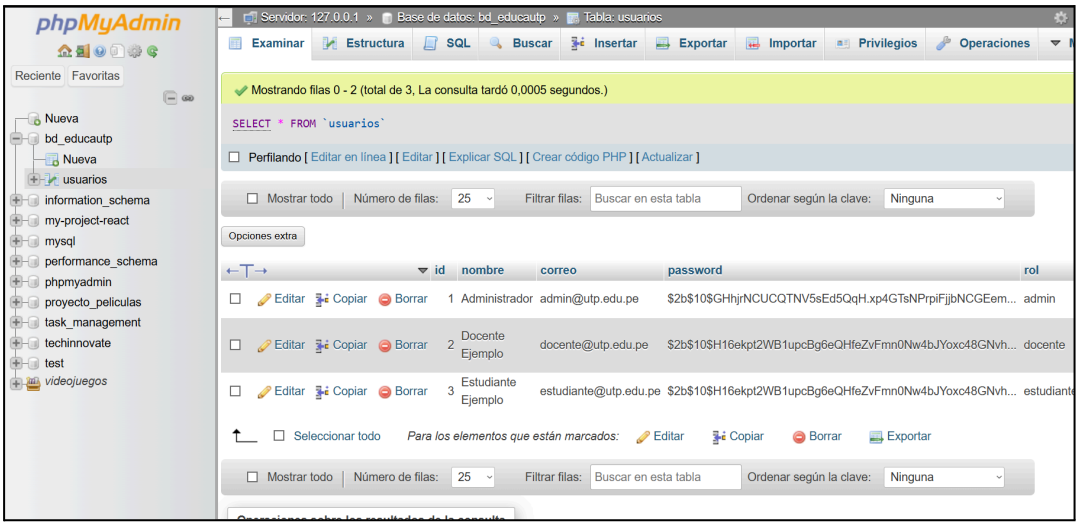
- **Enlace:** https://github.com/SergioChiquinta/Educa-UTP.git

3.4.2. Base de Datos

Para el primer avance se ha desarrollado una Base de Datos simple, únicamente para las funciones de Login. Esta cuenta con las siguientes características:

- Entidad usuarios:
  - nombre.
  - correo.
  - password.
  - rol.
  - foto_perfil.
  - area_interes.
  - descripcion..

Imagen de la Base de Datos en phpMyAdmin:



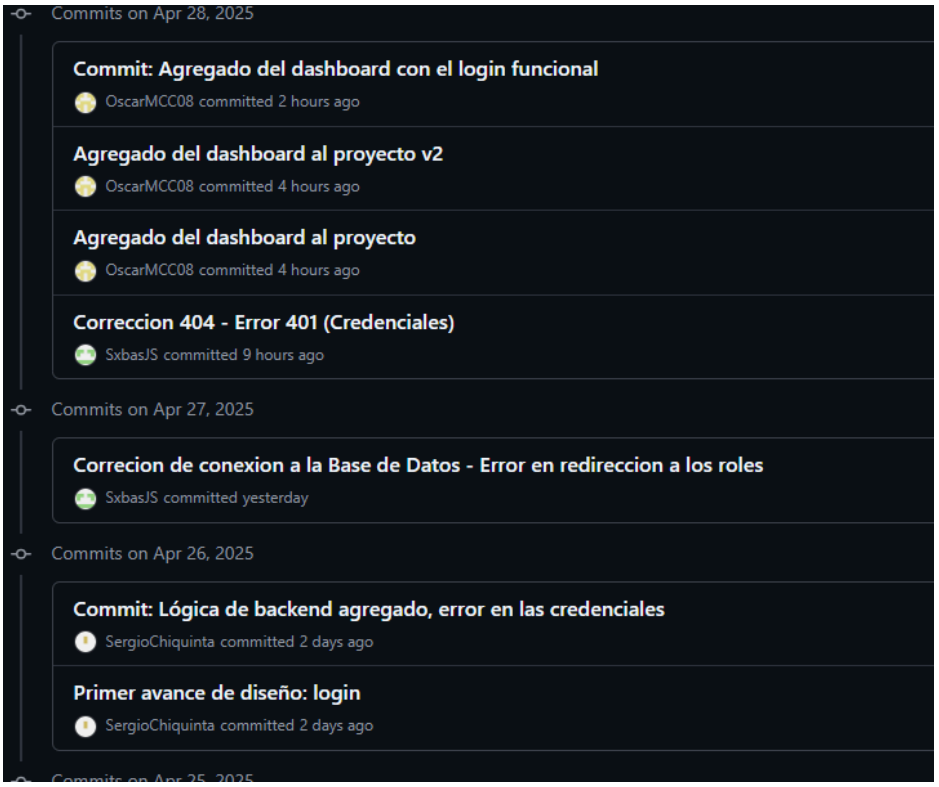
4. PROCEDIMIENTOS DE CONTROL DE VERSIONES

4.1. Realización de Commits

4.1.1. Commits realizados en la rama de Sergio

4.1.2. Commits realizados en la rama de Sebastian

4.1.3. Commits realizados en la rama de Oscar



4.2. Pull Request y revisión de código

4.2.1. Creación y Gestión de Pull Requests

- **Apertura de PR:** Se deben crear Pull Requests (PRs) desde ramas de características (feature branches) hacia la rama principal de desarrollo (develop). Esto asegura un flujo organizado de integración de nuevas funcionalidades.
- **Plantillas de PR:** Se implementarán plantillas prediseñadas para la creación de PRs, que incluyan campos obligatorios como:
    - Descripción del cambio realizado.
    - Tipo de cambio (nueva funcionalidad, corrección de error, mejora, etc.).
    - Detalle de las pruebas realizadas (testing).
- **Draft PRs (PRs en Borrador):** Se podrán utilizar PRs en modo borrador para trabajos en progreso, permitiendo visibilidad temprana y evitando fusiones prematuras.

4.2.2. Proceso de Revisión de Código

- Asignación de Revisores: Cada Pull Request debe tener asignado al menos un revisor obligatorio, preferentemente un compañero de equipo con conocimiento del módulo o funcionalidad involucrada.
- Comentarios en Línea: Durante la revisión, se debe proporcionar feedback específico directamente sobre el código fuente, utilizando los comentarios en línea de la plataforma de control de versiones.
- Discusión y Resolución: Las dudas, sugerencias o desacuerdos deben discutirse en el mismo hilo del PR, utilizando el sistema de conversaciones hasta lograr un consenso o resolución adecuada.
- Criterios de Aprobación y Rechazo: Los cambios sólo podrán ser fusionados a develop una vez que:
    - Se hayan recibido la aprobación de los revisores asignados.
    - Se hayan resuelto todos los comentarios críticos.
    - Se hayan realizado las pruebas necesarias y estén documentadas.

En caso de que un PR no cumpla con los requisitos, podrá ser rechazado o solicitado para cambios adicionales.

4.2.3. Ejemplo de un Pull Request en nuestro proyecto

- **Pull Request Generado**
    - **Origen:** Rama feature/login-usuarios-sergio
    - **Destino:** Rama develop
    - **Descripción del PR:** Se implementó el módulo de login de usuarios, incluyendo:
        - Gestión de autenticación de usuarios.
        - Subida junto a la actualización de datos de usuario y fotos de perfil.
        - Integración de la carga de imágenes desde el servidor.
        - Correcciones de comportamiento en la sesión de usuario.
    - **Testing realizado:**
        - Verificación de actualización de imagen de perfil.
        - Comprobación de persistencia de datos de sesión.

        - Validación de la correcta subida de imágenes a la carpeta /uploads.
        - Pruebas de inicio, actualización y cierre de sesión.
    - **Tipo de cambio:**
        - Nueva funcionalidad.
        - Corrección de errores.
        - Mejoras de código existentes.
    - **Estado del Pull Request:**
        - PR creado y marcado como Listo para revisión.
- **Proceso de Revisión aplicado**
    - **Asignación de revisor:** Se asignó a Sergio André Chiquinta Veramendi como revisor obligatorio para el código subido.
    - **Comentarios en línea:** El revisor proporcionó comentarios específicos en aquellas secciones que involucran el manejo de imágenes y la actualización de datos en tiempo real.
    - **Discusión y resolución:**
        - Ajuste en la estructura del proyecto.
        - Ajuste en el diseño del login.
        - Ajustes en la lógica del backend, como en el server.js para servir correctamente los archivos de imágenes.
        - Mejora en el frontend para reflejar los cambios de foto de perfil sin necesidad de cerrar sesión.
        - Entre otros.
    - **Discusión y resolución:** El Pull Request fue aprobado luego de las correcciones y se encuentra listo para ser fusionado en la rama develop.

4.3. Issues y Control del proyecto usando Trello

Para organizar el trabajo del equipo de forma efectiva, se utiliza Trello como herramienta principal de seguimiento. Se establecen buenas prácticas para el manejo de tareas, bugs y mejoras del sistema:

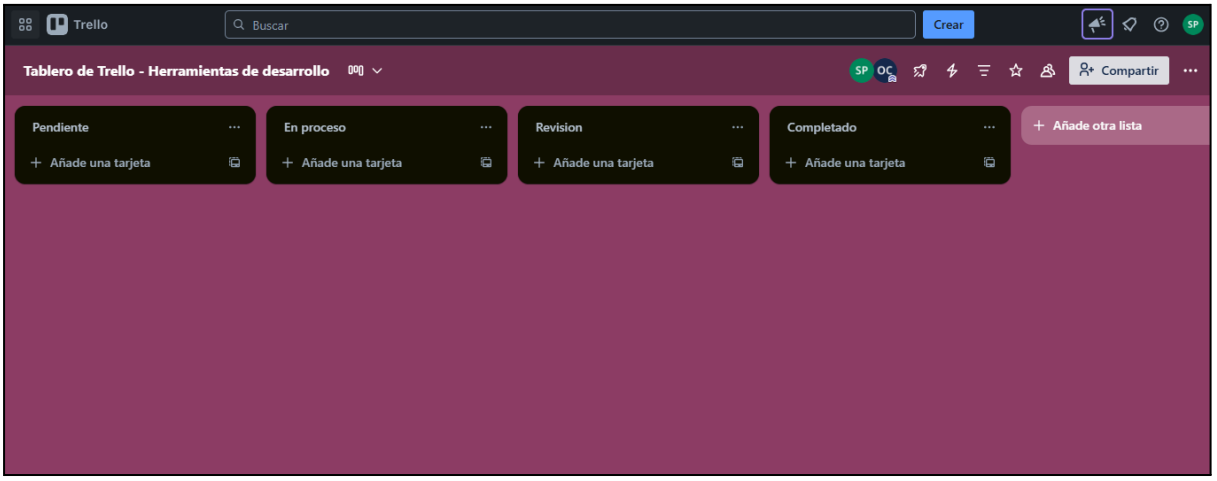
4.3.1. Gestión estructurada de Issues

- **Creación estructurada:** Usar plantillas para reportar bugs, solicitar funcionalidades o documentar tareas.

- **Etiquetas (labels):** Clasificar issues por tipo (bug, enhancement, documentation).
- **Asignaciones:** Distribuir tareas entre los miembros del equipo.

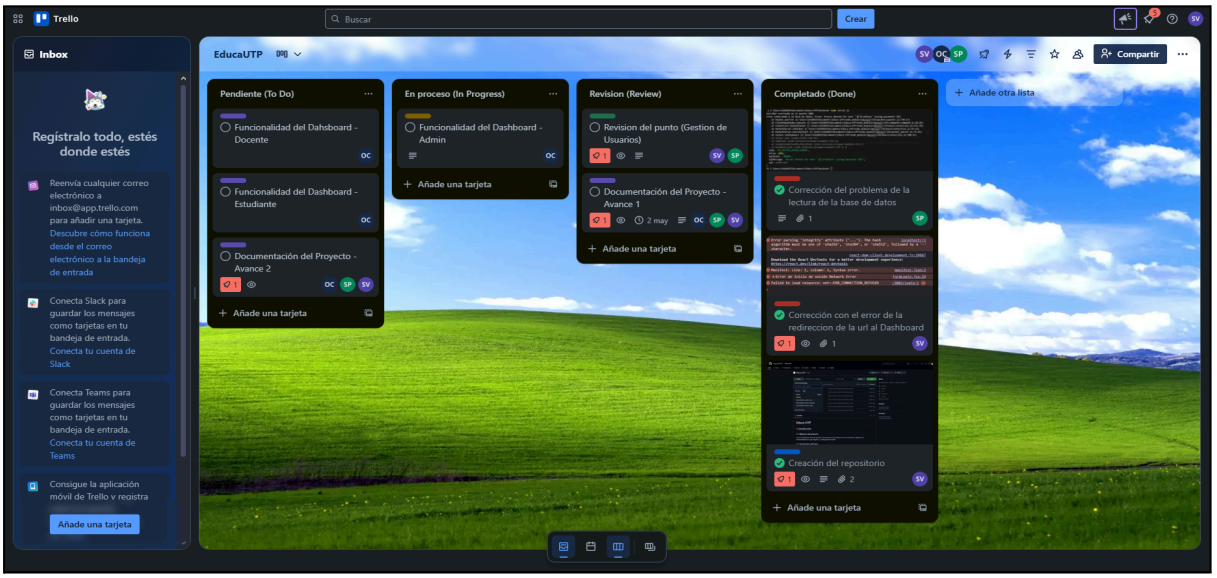
4.3.2. Construir el tablero de Proyecto

- **Configuración de proyecto:** Crear un tablero en Trello con columnas



(Pendiente, En Progreso, Revisión, Completado).

- **Automatizaciones:** Configurar movimiento automático de tarjetas según el estado de issues y PRs.
  - Pendiente.
- **Seguimiento de progreso:** Utilizar la vista de milestone para visualizar el avance del proyecto.
  - Pendiente.
- **Tablero Actual:** Así es como se vé nuestro tablero de proyecto para este 1er Avance.



- **Enlace del Tablero** Trello: Sólo para la vista, no editable.
    - **Enlace:** Requiere suscripción en Trello