UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERIA ELECTRICA, ELECTRONICA, INFORMATICA Y MECANICA
DEPARTAMENTO ACADEMICO DE INFORMÁTICA

INFORME: SISTEMA DE GESTIÓN DE PORTAFOLIOS

CURSO: Fundamentos de Base de Datos
DOCENTE: M.Sc. Maritza Katherine Irpanocca Cusimayta

INTEGRANTES:

Héctor Paolo Barazorda Cuellar

Informe de Requerimientos: Sistema de Gestión de Portafolios de Cursos

Proyecto de software en tecnologia javascript

20 de julio de 2025

Índice

1. Resumen del Proyecto

El presente informe detalla los requerimientos para el desarrollo de un **Sistema de Gestión de Portafolios de Cursos** para la Escuela Profesional de Ingeniería Informática y de Sistemas de la Universidad Nacional de San Antonio Abad del Cusco. El objetivo principal es proporcionar una plataforma donde los docentes puedan organizar y almacenar digitalmente el material de sus cursos, incluyendo sílabos, materiales de enseñanza, ejemplos de asignaciones y exámenes, y muestras de trabajos estudiantiles. La metodología de desarrollo a seguir es **Scrum**.

2. Planteamiento General del Problema

Actualmente, muchas instituciones educativas gestionan los portafolios docentes de forma manual o a través de archivos dispersos en Google Drive, lo que genera problemas como:

- **Pérdida de información**: Los documentos importantes pueden extraviarse o dañarse fácilmente.
- **Dificultad para realizar seguimiento**: Resulta complicado monitorear el estado y la evolución de los portafolios a lo largo del tiempo.
- **Procesos administrativos lentos e ineficientes**: La gestión manual consume mucho tiempo y recursos.
- **Limitada transparencia y trazabilidad**: Es difícil verificar quién accedió a qué documento y cuándo, quién lo creó o quién lo eliminó.

Problema Central: ¿Cómo mejorar la gestión, consulta y actualización de portafolios docentes mediante el uso de un sistema de información eficiente, accesible y escalable?

3. Requerimientos Funcionales

El sistema debe cumplir con las siguientes funcionalidades clave:

3.1. Gestión de Usuarios y Roles

El sistema soportará tres roles de usuario principales:

- *Administradores*: Podrán gestionar portafolios para diferentes semestres.Eliminar Docentes,filtar documentos.
- *Docentes*: Podrán subir todo el material necesario para los portafolios de sus cursos.
- *Evaluadores*: Podrán revisar los portafolios y enviar retroalimentación directamente en el dashboard del docente.

Los permisos de acceso y las funciones serán específicos para cada rol.

3.2. Gestión de Portafolios y Documentos

- Cada portafolio incluirá: carátula, carga lectiva, filosofía, currículo y la sección de cursos.
- Cada *Curso* incluirá: sílabo y avance de curso.
- El sistema permitirá la carga y gestión de archivos.
- Habrá un límite de tamaño de **5 MB por documento**.
- Los documentos deben estar correctamente organizados y ser accesibles.

3.3. Visualización de Documentos

- El sistema integrará un **visor de documentos** que permitirá la previsualización de archivos sin necesidad de descargarlos.

4. Requerimientos No Funcionales

- **Seguridad**: La plataforma debe garantizar un alto nivel de seguridad mediante el uso de autenticación basada en **tokens** (por ejemplo, JWT) y la protección de acceso basada en **roles definidos**. Cada usuario accederá únicamente a las funcionalidades que su rol le permita, reforzando el control y trazabilidad del sistema.
- **Escalabilidad**: El sistema debe ser capaz de crecer y adaptarse a un mayor volumen de usuarios y datos.
- **Facilidad de Uso**: La interfaz debe ser intuitiva y fácil de manejar para todos los tipos de usuarios.
- **Rendimiento**: El acceso a los documentos a través del visor integrado debe ser rápido y eficiente.
- **Adaptabilidad**: Tendrá un diseño responsivo que permitirá una visualización óptima en todos los dispositivos.

5. Modelado de Base de Datos

5.1. Diseño Conceptual

El diseño conceptual de la base de datos se centra en la identificación de las entidades principales del sistema y las relaciones entre ellas, sin entrar en detalles de implementación. Representa una visión de alto nivel de la información a almacenar. Las entidades clave identificadas son:

- *Usuarios*: Representando a administradores, docentes y evaluadores.
- *Portafolios*: Que agrupan el material de un curso específico por un docente.
- *Cursos*: Las asignaturas que forman parte de un portafolio.
- *Documentos*: Los archivos específicos dentro de un portafolio o curso (carátulas, sílabos, etc.).
- *Retroalimentación*: Comentarios y evaluaciones sobre los portafolios.

Estas entidades se relacionarán para reflejar la estructura de la información académica y administrativa.

5.2. Diseño Lógico

El diseño lógico traduce el modelo conceptual a un modelo de datos relacional específico, detallando las tablas, columnas, tipos de datos y las relaciones con sus claves primarias y foráneas. Este diseño se basa directamente en el esquema de Prisma proporcionado.

5.3. Diseño Físico

El diseño físico se ocupa de la implementación real de la base de datos, considerando el sistema de gestión de base de datos (SGBD) específico, la optimización del rendimiento y la seguridad.

Consultas y Recuperación de Datos: El diseño de la base de datos, junto con las capacidades de Prisma, facilita operaciones eficientes:

- **Búsqueda y filtrado** de portafolios por docente (`teacherId`), semestre (`semester`) o curso (`curso.name`, `curso.code`).
- **Recuperación de todos los documentos** asociados a un portafolio específico mediante las relaciones definidas.
- **Agrupamiento de documentos**: Las consultas permitirán identificar y recuperar las URLs de los documentos del mismo tipo para que la lógica de la aplicación los agrupe o fusione.
- **Listado de retroalimentaciones** para un docente o portafolio.

Seguridad de Base de Datos: La seguridad a nivel de base de datos es crítica:

- Los roles definidos (`ADMINISTRADOR`, `DOCENTE`, `EVALUADOR`) serán la base para implementar **políticas de acceso y privilegios** a nivel de aplicación, y si el SGBD lo permite, a nivel de base de datos.
- El almacenamiento de contraseñas utilizará **hash seguro con bcrypt** antes de la persistencia.
- El **cifrado de datos en reposo y en tránsito** debe ser gestionado por el servicio de almacenamiento en la nube y la configuración de la conexión segura entre la aplicación y el SGBD.

Tecnologías de Base de Datos Sugeridas: Aunque el esquema de Prisma está configurado para `sqlite` para desarrollo, las tecnologías sugeridas para producción son **MySQL** o **PostgreSQL**, ambas robustas y escalables, adecuadas para este tipo de aplicación.

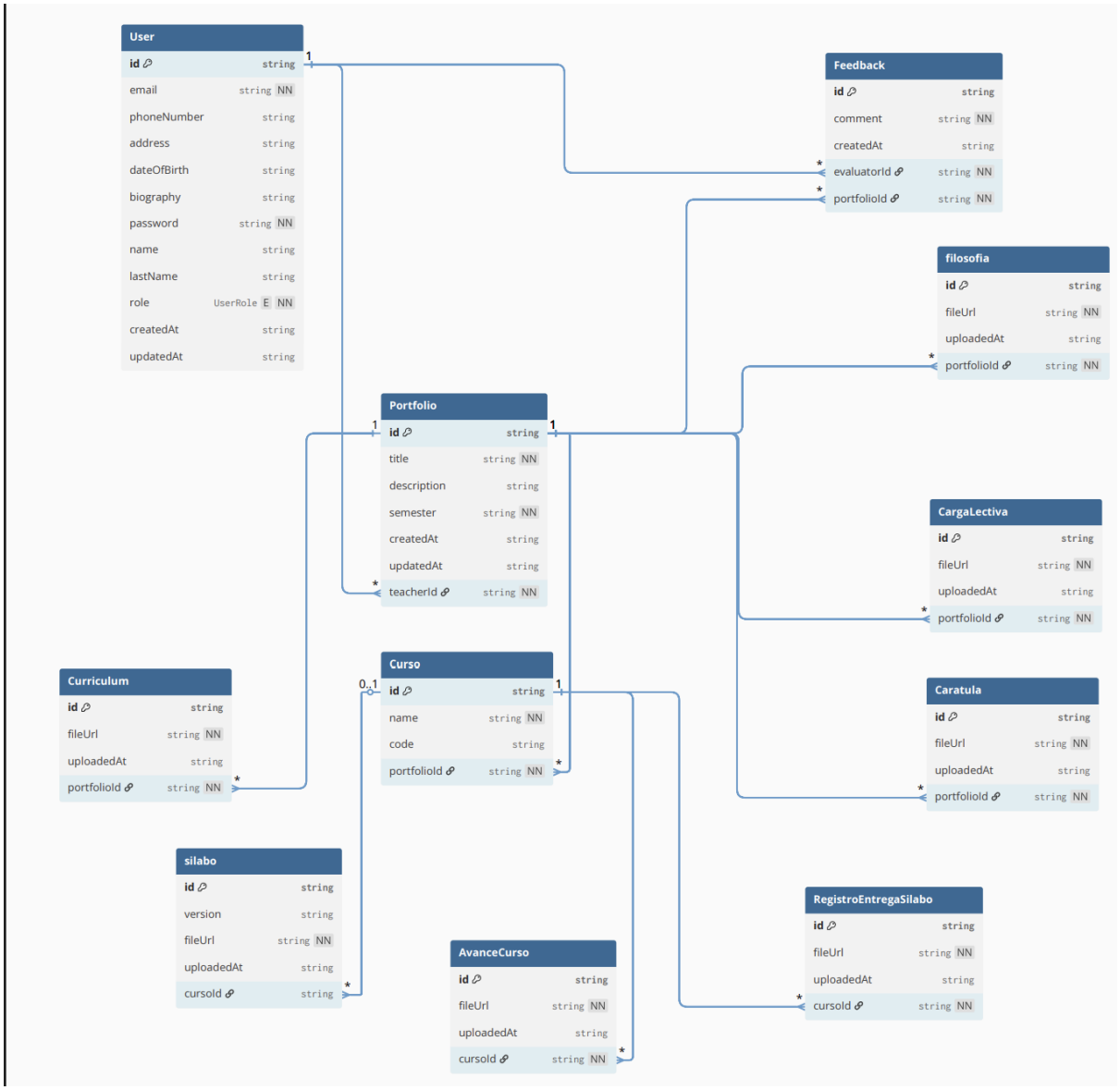

Figura 1: Diseño físico en SQLITE.

6. Conexión a la Base de Datos con Prisma y SQLite en Turso

Para la implementación de la base de datos, se utilizará **Prisma como ORM** para interactuar con una base de datos **SQLite hospedada en Turso**.

Prisma como ORM: Prisma simplifica la interacción con la base de datos. Su archivo `schema.prisma` define el modelo de datos. Prisma Client, generado a partir de este esquema, permite realizar operaciones CRUD de forma segura y tipada en el código de la aplicación.

SQLite y Turso:

- **SQLite**: Base de datos relacional ligera, autocontenida y sin servidor, ideal para desarrollo y pruebas.
- **Turso**: Proporciona una solución de base de datos SQLite distribuida y escalable en la nube, adecuada para aplicaciones web con usuarios en diferentes ubicaciones.

Configuración de Conexión: La conexión se gestionará a través de la variable de entorno `DATABASE_URL`, como se especifica en el `datasource` del `schema.prisma`:

```
datasource db {
  provider = "sqlite"
  url      = env("DATABASE_URL")
}
```

Para conectar con Turso, la `DATABASE_URL` contendrá la URL de conexión proporcionada por Turso, encapsulando la ubicación de la base de datos SQLite en la nube.

7. Tecnologías Utilizadas para el Desarrollo del Software

El desarrollo del **Sistema de Gestión de Portafolios de Cursos** se basa en las siguientes tecnologías:

- **Next.js**: Framework de JavaScript basado en React, utilizado para construir la interfaz de usuario del sistema, ofreciendo un diseño responsivo, renderizado del lado del servidor y una experiencia de usuario optimizada.
- **NestJS**: Framework de Node.js para el desarrollo del backend, que proporciona una arquitectura modular y escalable, facilitando la implementación de la lógica de negocio y la integración con la base de datos.
- **Prisma**: ORM (Object-Relational Mapping) que simplifica la interacción con la base de datos, permitiendo operaciones CRUD seguras y tipadas, y definiendo el esquema de datos mediante el archivo `schema.prisma`.
- **SQLite**: Base de datos relacional ligera y sin servidor, hospedada en Turso para entornos de producción, ideal para desarrollo y pruebas debido a su simplicidad y escalabilidad distribuida.

Estas tecnologías fueron seleccionadas por su robustez, facilidad de integración y capacidad para cumplir con los requerimientos funcionales y no funcionales del sistema.