GESTION DE MALLA CURRICULAR

MIGUEL CASSERES

KEVIN COLORADO VERA

SAMUEL MEJIA

JUAN SEBASTIAN OSPINA

DISEÑO DE SISTEMAS DE INFORMACION

580304012-3

ALEXANDRA GUERRERO BOCANEGRA

INSTITUTO TECNOLÓGICO METROPOLITANO

FACULTAD DE INGENIERÍA Y SISTEMAS

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

MEDELLÍN

2025

Refinamiento:

- Se corrigió el User Story Mapping en ambos Release.
- Se agregó el Diagrama Causa Efecto que hizo falta en el entregable anterior.
- Se incluye paleta de colores en el frontend.

1. Descripción General

El sistema de gestión de la malla curricular del ITM tiene como objetivo optimizar la gestión académica relacionada con la consulta, actualización y planificación de los planes de estudio de las diferentes carreras ofrecidas por la institución. Actualmente, la administración de la malla curricular se realiza de forma manual a través de documentos y hojas de cálculo, lo que genera problemas como inconsistencias en la información, retrasos en la actualización de los programas y dificultad en la visualización de requisitos y prerrequisitos para los estudiantes y docentes.

El sistema busca resolver estos problemas mediante una plataforma digital centralizada que permita a los actores involucrados acceder y gestionar la información de manera eficiente y precisa. Con esta automatización, se reducirán errores en la planificación académica, se mejorará la experiencia de los estudiantes al organizar su carga académica y se facilitará la toma de decisiones por parte de los administradores.

2. Actores Principales

Jefe de programa: Responsable de gestionar, diseñar y actualizar la malla curricular, agregar nuevas materias, modificar prerrequisitos y aprobar cambios en los planes de estudio.

Docente: Puede consultar la malla curricular, revisar los requisitos de las asignaturas y sugerir modificaciones en el plan de estudio.

Estudiante: Accede a la malla curricular para planificar su carga académica, verificar requisitos de materias y permitir obtener una mejor secuencia de la carrera.

3. Funcionalidades Principales

Autenticación y gestión de usuarios: Permite el acceso seguro a la plataforma mediante credenciales de usuario (correo y contraseña) con diferentes roles de acceso.

Gestión de malla curricular: Creación, modificación y visualización de las asignaturas, incluyendo información como código, nombre, créditos, prerrequisitos y correquisitos.

Visualización interactiva de la malla: Representación gráfica de la estructura curricular con opciones de filtrado por semestre y dependencias entre materias.

Historial de modificaciones: Registro de cambios realizados en la malla curricular con fecha y usuario responsable.

Exportación e informes: Generación de reportes en formatos PDF y Excel con la estructura actualizada de la malla curricular.

4. Flujos Básicos

Consulta de malla curricular: Un estudiante accede a la plataforma, selecciona su programa académico y visualiza la estructura curricular, verificando los requisitos de las materias que desea inscribir.

Modificación de la malla curricular: Un administrador inicia sesión, edita una asignatura para modificar su número de créditos y guarda los cambios, los cuales quedan registrados en el historial.

Generación de recomendaciones: Un estudiante visualiza su progreso y recibe una sugerencia de inscripción con base en las materias aprobadas y disponibles para su próximo semestre.

Supervisión de cambios: Un coordinador revisa una modificación propuesta por un docente y la aprueba o rechaza según la normativa institucional.

5. Restricciones o Reglas de Negocio

Solo los jefes de programa pueden realizar cambios en la malla curricular.

Los estudiantes solo pueden visualizar la información, sin permisos para modificar la estructura curricular.

Las recomendaciones de inscripción deben basarse en reglas predefinidas, como el cumplimiento de prerrequisitos y el avance del estudiante.

La plataforma debe garantizar la seguridad de los datos mediante encriptación y copias de seguridad periódicas.

Diagrama de Causa y Efecto

User Story Mapping

Historias de Usuario

HU01: Inicio de sesión

Como estudiante, docente o jefe de programa,
quiero iniciar sesión en la plataforma con mi correo y contraseña
para acceder a mis funcionalidades según mi rol.

Criterios de aceptación:

- El sistema debe contar con un formulario de inicio de sesión que incluya correo y contraseña.
- El sistema debe validar las credenciales ingresadas y mostrar un mensaje si son incorrectas.
- El usuario debe ser redirigido a su panel correspondiente según el rol.
- El inicio de sesión exitoso debe registrar la hora y el usuario que accedió.

HU02: Gestión de roles de usuario

Como jefe de programa,
quiero gestionar los usuarios y sus roles
para asegurar que cada actor tenga los permisos adecuados.

Criterios de aceptación:

- El jefe de programa debe poder consultar una lista de usuarios con sus roles actuales.
- El sistema debe permitir cambiar el rol de un usuario y guardar la modificación.
- Los cambios de rol deben quedar registrados con fecha y responsable del cambio.
- El sistema debe restringir las acciones de cada usuario según su rol asignado.

HU03: Agregar nueva materia

Como jefe de programa,
quiero agregar nuevas materias con su información correspondiente (código, nombre, créditos, prerrequisitos y correquisitos)
para mantener la malla curricular actualizada.

Criterios de aceptación:

- El sistema debe permitir registrar una nueva materia con todos los campos obligatorios.
- Al guardar, la nueva materia debe mostrarse en la malla curricular.
- La acción debe quedar registrada en el historial del sistema.
- No debe permitirse agregar materias con código duplicado.

HU04: Modificar materia existente

Como jefe de programa,
quiero modificar la información de una materia existente
para actualizar los cambios en la malla curricular.

Criterios de aceptación:

- El sistema debe mostrar los datos actuales de la materia para su edición.
- Los cambios realizados deben reflejarse inmediatamente en la malla curricular.
- La modificación debe registrarse con fecha, hora y usuario que la realizó.
- Si se modifican requisitos, el sistema debe actualizar las relaciones afectadas.

HU05: Eliminar materia

Como jefe de programa,
quiero eliminar una materia de la malla curricular
para retirar asignaturas obsoletas.

Criterios de aceptación:

- El sistema debe solicitar confirmación antes de eliminar la materia.
- No debe permitirse eliminar materias que sean requisito de otras sin una advertencia previa.
- La materia debe desaparecer de la malla una vez eliminada.
- La acción debe registrarse en el historial.

HU06: Visualizar malla como estudiante

Como estudiante,
quiero visualizar la malla curricular de mi programa académico
para planificar mi carga académica.

Criterios de aceptación:

- El estudiante debe poder ver todas las materias del programa organizadas por semestre.
- La interfaz debe mostrar claramente los créditos, requisitos y relaciones entre materias.
- Las materias cursadas o aprobadas deben estar diferenciadas visualmente.
- Debe ser posible consultar información detallada de cada asignatura.

HU07: Consultar malla como docente

Como docente,
quiero consultar la malla curricular
para conocer los requisitos y dependencias entre materias.

Criterios de aceptación:

- El docente debe poder acceder a la malla completa de los programas asignados.
- La malla debe incluir detalles como prerrequisitos, correquisitos y créditos.
- La visualización debe ser de solo lectura.
- El sistema debe mostrar la versión vigente del plan de estudios.

HU08: Filtro por semestre y dependencias

Como estudiante,
quiero filtrar las materias por semestre y dependencias
para comprender mejor la estructura del plan de estudios.

Criterios de aceptación:

- El sistema debe permitir aplicar filtros por semestre específico.
- Debe permitir visualizar solo materias con determinadas relaciones (prerrequisitos o correquisitos).
- Al quitar los filtros, la malla debe volver a mostrarse completamente.
- La interfaz debe ser intuitiva para aplicar y quitar filtros.

HU09: Consultar historial de cambios

Como jefe de programa,
quiero consultar el historial de modificaciones de la malla curricular
para revisar los cambios realizados y su justificación.

Criterios de aceptación:

- El historial debe mostrar una lista cronológica de modificaciones.
- Cada entrada debe incluir fecha, tipo de cambio, usuario y materia afectada.
- Debe poder filtrarse por rango de fechas o por materia.
- La consulta debe estar restringida a usuarios con permisos de jefe de programa.

HU10: Registro automático de cambios

Como jefe de programa,
quiero registrar automáticamente cualquier modificación en la malla curricular con la fecha y usuario responsable
para garantizar trazabilidad.

Criterios de aceptación:

- Cada modificación realizada en la malla debe ser registrada automáticamente en un log.
- El registro debe incluir tipo de acción (crear, modificar, eliminar), fecha y usuario.
- El historial debe ser de solo lectura y no editable por los usuarios.
- El sistema debe permitir exportar el historial en formato PDF o Excel.

HU11: Recomendaciones de inscripción

Como estudiante,
quiero recibir sugerencias de inscripción basadas en las materias que ya aprobé
para optimizar mi avance en la carrera.

Criterios de aceptación:

- El sistema debe analizar las materias cursadas y sugerir las que estén habilitadas por requisitos cumplidos.
- La sugerencia debe incluir nombre, créditos y semestre recomendado de cada materia.
- Si no hay materias disponibles por inscribir, el sistema debe notificar al estudiante.
- Las recomendaciones deben actualizarse automáticamente al aprobar nuevas materias.

HU12: Sugerencias de modificación por parte del docente

Como docente,
quiero sugerir modificaciones en el plan de estudios
para mejorar la malla curricular según las necesidades académicas.

Criterios de aceptación:

- El sistema debe permitir al docente ingresar propuestas con título y justificación.
- Las sugerencias deben quedar registradas con fecha y nombre del docente.
- El estado de cada sugerencia debe actualizarse según el proceso de revisión (pendiente, aceptada, rechazada).
- El docente debe recibir una notificación con la decisión tomada sobre su sugerencia.

HU13: Aprobación o rechazo de cambios

Como coordinador de carrera,
quiero revisar y aprobar o rechazar modificaciones propuestas en la malla curricular
para garantizar que cumplan con las normativas.

Criterios de aceptación:

- El sistema debe mostrar una lista de propuestas pendientes de revisión.
- El coordinador debe poder aprobar o rechazar con comentarios.
- La decisión debe actualizar el estado de la sugerencia y notificar al proponente.
- El sistema debe registrar la acción en el historial.

HU14: Exportar malla curricular

Como jefe de programa,
quiero generar reportes de la malla curricular en PDF y Excel
para compartir la información con otros actores académicos.

Criterios de aceptación:

- El sistema debe permitir elegir entre los formatos PDF y Excel.
- El reporte debe contener los datos completos de las materias (código, nombre, créditos, requisitos).
- El archivo debe descargarse automáticamente.
- El nombre del archivo debe incluir el nombre del programa y la fecha de exportación.

HU15: Seguridad de los datos

Como administrador del sistema,
quiero asegurar que los datos sean encriptados y se realicen copias de seguridad periódicas
para proteger la información académica.

Criterios de aceptación:

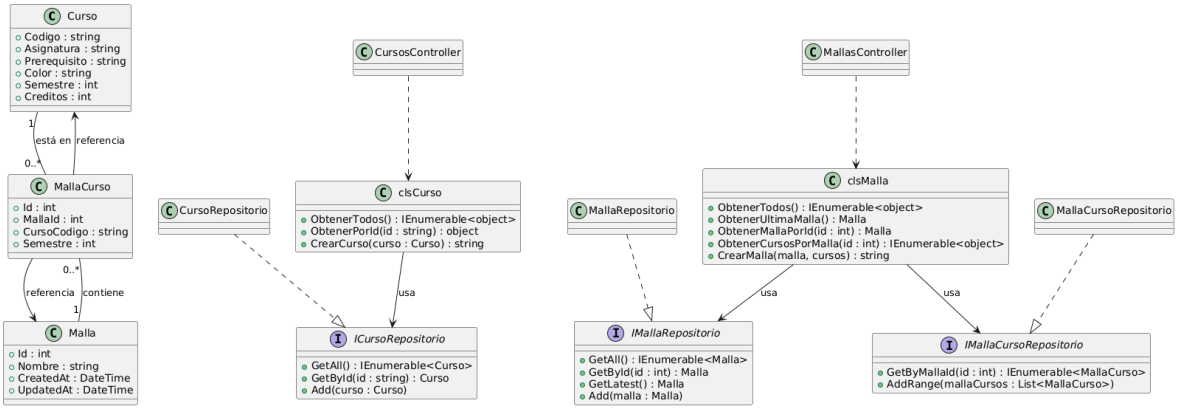
- Toda la información almacenada debe estar cifrada en la base de datos.
- El sistema debe generar respaldos automáticos en horarios establecidos.
- Debe haber una opción para restaurar información desde un respaldo.
- El administrador debe recibir alertas si un respaldo falla o no se realiza.

4. Diagrama de clases

5. FIGMA

6. Modelo C4

6.1. Nivel 4(Diagrama de clases) link



7. Diagrama de actividades

Patrón de Diseño: Facade

¿Qué es el patrón Facade?

El patrón Facade se utiliza para simplificar la interacción con sistemas complejos. En este caso, se aplica al encapsular la lógica de varias clases (como repositorios, validaciones y mapeos) dentro de una clase de dominio que ofrece una interfaz sencilla al controlador.

Ejemplo aplicado: clsCurso

A continuación, se muestra un fragmento de código que implementa el patrón Facade:

```
public class clsCurso
{
    private readonly CursoRepositorio _repositorio;
    private readonly IMapper _mapper;

    public clsCurso(CursoRepositorio repositorio, IMapper mapper)
    {
        _repositorio = repositorio;
        _mapper = mapper;
    }

    public async Task<List<CursoDto>> ObtenerCursosAsync()
    {
        var cursos = await _repositorio.ObtenerTodosAsync();
        return _mapper.Map<List<CursoDto>>(cursos);
    }

    public async Task<CursoDto> CrearCursoAsync(CursoDto cursoDto)
    {
        var curso = _mapper.Map<Curso>(cursoDto);
        await _repositorio.AgregarAsync(curso);
        return _mapper.Map<CursoDto>(curso);
    }
```

Simplifica la interacción: El controlador no necesita preocuparse por los detalles internos de cómo se accede o manipulan los datos.

Encapsula la complejidad: Validaciones, mapeos y operaciones con el repositorio se manejan internamente en la clase.

Proporciona una interfaz clara: Exponen solo los métodos necesarios, ocultando la implementación interna.

Patrón de Diseño: Repository

¿Dónde se aplica?

- Interfaces como ICursoRepositorio
- Clases concretas como CursoRepositorio.cs

¿Qué hace este patrón?

El patrón **Repository** aísla la lógica de acceso a datos. Permite trabajar con una **interfaz** (ICursoRepositorio) en lugar de acceder directamente al ORM (por ejemplo, Entity Framework). Esto hace que el código sea más limpio, testeable y fácil de mantener.

PRINCIPIOS SOLID

- **S:** Cada clase debe tener una única responsabilidad.
  Ejemplos:

  CursoRepositorio.cs, MallaRepositorio.cs, etc., están separados por entidad, y cada uno maneja solo su propio conjunto de operaciones de base de datos.

  ClsMalla.cs y ClsCurso.cs Estan separados y cada uno tiene una lógica distinta

  CursosController.cs y MallasController.cs están separados también por entidad y responsabilidad, encargándose de la lógica de presentación/controlador.

- Open/Closed Principle (OCP)
  El código debe estar abierto para extensión, pero cerrado para modificación.
  Ejemplo:

  Se puede extender la lógica de los repositorios creando nuevos métodos sin modificar los existentes.
  Con interfaces como ICursoRepositorio, se puede cambiar la implementación sin cambiar el controlador.

- Liskov Substitution Principle (LSP)
  "Una clase hija debe poder sustituir a su clase padre sin romper el sistema."

  La interfaz IMallaRepositorio y cualquier clase que la implemente (como MallaRepositorio) debe poder ser usada sin que falle la lógica del sistema.

- I – Interface Segregation Principle (ISP)
  “Las interfaces deben ser específicas al cliente.”
  Ejemplo:
  La interfaz ICursoRepositorio que solo contiene métodos relacionados con cursos, no obliga a otras clases a implementar métodos que no necesitan.

- Dependency Inversion Principle (DIP)
  "Depender de abstracciones, no de implementaciones concretas."
  clsMalla depende de interfaces, no de clases concretas.

Github

Referencias

Brown, S. (2021). The C4 Model for Software Architecture. https://c4model.com/

UML Resource Center. https://uml-diagrams.org/

Diagrama de clases. https://diagramasuml.com/diagrama-de-clases/

Principios SOLID. https://www.freecodecamp.org/espanol/news/los-principios-solid-explicados-en-espanol/