SEMANA 6 - MÓDULO DE DATOS ACADÉMICOS (PARTE 1)

1. INTRODUCCIÓN CONTEXTUAL DEL PROYECTO

La plataforma de "Comunicación y Seguimiento Académico" para la I.E.P. Las Orquídeas representa una solución integral para mejorar la interacción entre la institución educativa, docentes, estudiantes y padres de familia. Durante la Semana 6 del cronograma de desarrollo, nos enfocamos en la primera parte del Módulo de Datos Académicos, componente crítico que permitirá la gestión eficiente de calificaciones y asistencia.

Este módulo aborda uno de los problemas centrales identificados en la institución: la dificultad para registrar, procesar y comunicar información académica de manera ágil y precisa. El sistema actual requiere procesos manuales que consumen tiempo significativo del personal docente y generan retrasos en la comunicación con los padres.

El desarrollo de este módulo se enmarca dentro de una metodología ágil incremental, donde cada semana representa un ciclo completo de análisis, diseño, implementación y prueba. Esta aproximación nos permite obtener retroalimentación constante y asegurar que el producto final se alinee con las necesidades reales de los usuarios.

2. Entregables en el modulo

2.1 Documentación Técnica

La documentación fue un componente fundamental del desarrollo:

- **Historias de Usuario Detalladas**: Elaboración de especificaciones funcionales completas en [`doc/Semana 5/HUDetalladas_datos1.md`](doc/Semana 5/HUDetalladas_datos1.md:1)
- **Documentación API**: Creación de una referencia completa de endpoints en [`doc/Semana 5/DocumentacionAPI_datos1.md`](doc/Semana 5/DocumentacionAPI_datos1.md:1)
- **Comentarios en el código**: Inclusión de explicaciones claras en funciones complejas y lógica de negocio

Esta documentación no solo sirve como referencia técnica, sino que facilita la colaboración y el mantenimiento futuro del sistema.

2.2 Implementación de API REST

El diseño de la API REST siguió principios de arquitectura limpia y buenas prácticas:

- **Endpoints bien definidos**: Cada ruta tiene un propósito claro y utiliza métodos HTTP apropiados (GET, POST)
- **Formato estandarizado de respuestas**: Todas las respuestas siguen una estructura consistente con campos `success`, `data` y `error` cuando aplica
- **Códigos de estado HTTP**: Utilización apropiada de códigos para indicar éxito (200), errores de cliente (400, 403, 404) y errores del servidor (500)

Por ejemplo, en el servicio de calificaciones (`services/gradesService.js`), implementamos un manejo robusto de errores con códigos específicos que facilitan la depuración y proporcionan información clara al cliente.

2.3 Gestión de Bases de Datos con Prisma

La implementación del módulo académico requirió un manejo sofisticado de la base de datos:

- **Consultas optimizadas**
- **Transacciones atómicas**: Implementación de operaciones complejas que deben completarse en su totalidad o revertirse completamente
- **Validaciones a nivel de base de datos**: Aprovechamiento de las capacidades de Prisma para asegurar integridad referencial

2.4 Desarrollo Backend con Node.js y Express

Durante esta semana, consolidé mi comprensión y aplicación del patrón MVC (Modelo-Vista-Controlador) utilizando Node.js con el framework Express. Esta arquitectura permitió una separación clara de responsabilidades:

- **Modelo**: Representado por Prisma ORM, gestionando la interacción con la base de datos PostgreSQL
- **Vista**: Aunque el desarrollo frontend corresponde a la Semana 7, diseñamos endpoints que retornan datos estructurados en formato JSON
- **Controlador**: Implementando la lógica de negocio y coordinando las interacciones entre el modelo y las respuestas HTTP

2.5 Pruebas y Validación

Aunque las pruebas unitarias e integración se profundizarán en semanas posteriores, durante esta semana implementamos validaciones robustas:

- **Validación de entrada**: Uso de Zod para validar parámetros de entrada y asegurar integridad de datos
- **Validación de archivos**: Implementación de lógica compleja para procesar archivos Excel, verificando estructura y contenido
- **Manejo de errores**: Creación de un sistema estandarizado para gestionar diferentes tipos de errores

3. DESARROLLO DE COMPETENCIAS GENERALES

3.1 Trabajo en Equipo y Metodología Scrum

Aunque el desarrollo individual fue predominante esta semana, seguí los principios de Scrum definidos en el cronograma:

- **Semanas como sprints**: Cada semana funcionó como un sprint con objetivos claros y entregables definidos
- **Reuniones diarias de seguimiento**: Coordinación con el supervisor para alinear expectativas y resolver bloqueos
- **Retrospectivas al final de cada semana**: Evaluación del progreso y ajuste de planes para la siguiente semana

Esta metodología permitió un avance constante y la detección temprana de desviaciones respecto al plan original.

3.2 Comunicación Efectiva

La comunicación fue clave durante el desarrollo del módulo:

- **Documentación clara**: Elaboración de documentos técnicos comprensibles para diferentes audiencias (desarrolladores, analistas, usuarios finales)
- **Presentación de avances**: Comunicación periódica de resultados al supervisor, adaptando el lenguaje técnico según el público

- **Feedback constructivo**: Recepción e incorporación activa de sugerencias para mejorar la calidad del producto

3.3 Resolución de Problemas

El desarrollo del módulo académico presentó desafíos significativos que requirieron soluciones creativas:

- **Procesamiento de archivos Excel**: Implementación de una solución robusta para validar y procesar archivos complejos con múltiples formatos y estructuras
- **Manejo de errores masivos**: Creación de un sistema que puede procesar cientos de registros, identificando errores específicos sin detener el proceso completo
- **Optimización de consultas**: Mejora progresiva del rendimiento de las consultas a la base de datos para manejar volúmenes crecientes de información

4. EVIDENCIA CONCRETA DE CADA COMPETENCIA

4.1 Competencia CT1.1 - Aplicación de Metodologías y Estándares de Calidad

Evidencia: Implementación del patrón MVC con separación clara de responsabilidades en los módulos de asistencia y calificaciones.

4.2 Competencia CT2.2 - Construcción y Gestión de Repositorios de Datos

Evidencia: Diseño e implementación de un esquema de base de datos robusto para gestionar información académica sensible.

Ejemplo específico: Funciones que realizan validaciones exhaustivas antes de permitir operaciones, asegurando integridad y confidencialidad de los datos.

```
// Validación de contexto para proteger la integridad de los datos
if (fechaObj > hoy) {
  throw httpError('FUTURE_DATE_NOT_ALLOWED', 'No se puede verificar asistencia para fechas futu
}
```

4.3 Competencia CT2.3 - Fundamentos de Gestión de Calidad y Seguridad en Sistemas

Evidencia: Implementación de medidas de seguridad múltiples capas para proteger datos académicos.

Ejemplo específico: Implementaron funciona que utilizan transacciones atómicas para garantizar consistencia de datos y previene inserciones parciales.

```
// Uso de transacciones para asegurar atomicidad
await prisma.$transaction(async (tx) => {
  for (const reg of registros_validos) {
    // Procesamiento de cada registro
    await tx.evaluacion.create({
      data: {
        // Datos de la evaluación
      },
    });
  }
});
```

4.4 Competencia CT4.3 - Identificación y Análisis de Problemas de Investigación

Evidencia: Análisis detallado del problema de gestión de datos académicos y diseño de soluciones específicas.

Ejemplo específico: En [`doc/Semana 5/HUDetalladas_datos1.md`](doc/Semana 5/HUDetalladas_datos1.md:286), se identificaron y documentaron los problemas específicos del proceso actual de carga de calificaciones, diseñando soluciones tecnológicas precisas.

4.5 Competencia CT4.4 - Gestión de Proyectos de TI

Evidencia: Planificación y ejecución del módulo académico siguiendo el cronograma establecido.

Ejemplo específico: El cumplimiento de los entregables de la Semana 6 según lo planificado en `doc/CronogramaDesarrollo.md` , incluyendo la finalización de la API REST y la lógica de validación de archivos.

4.6 Competencia CG4 - Habilidad para la Comunicación Oral y Escrita

Evidencia: Elaboración de documentación técnica clara y comprensible.

Ejemplo específico: La documentación API en [`doc/Semana 5/DocumentacionAPI_datos1.md`] (doc/Semana 5/DocumentacionAPI_datos1.md:1) presenta ejemplos concretos de uso de cada endpoint, facilitando la comprensión para desarrolladores que consumirán la API.

5. REFLEXIÓN PERSONAL SOBRE DESAFÍOS Y SOLUCIONES

5.1 Desafíos Técnicos Enfrentados

- **Procesamiento de archivos Excel**: La implementación de un sistema robusto para validar y procesar archivos Excel con múltiples formatos y estructuras requirió investigación exhaustiva y experimentación con diferentes librerías.
- **Manejo de errores masivos**: Diseñar un sistema que pudiera procesar cientos de registros, identificando errores específicos sin detener el proceso completo, fue un desafío complejo que requirió un enfoque iterativo.
- **Optimización de consultas**: A medida que el volumen de datos crecía, fue necesario optimizar progresivamente las consultas a la base de datos para mantener un rendimiento aceptable.

5.2 Soluciones Implementadas

Para abordar estos desafíos, implementé varias soluciones:

- **Modularización del procesamiento de archivos**: Dividí el procesamiento de archivos en fases claras (lectura, validación estructural, validación de datos, procesamiento), lo que facilitó la depuración y mejora del sistema.
- **Sistema de reporte de errores granular**: Implementé un mecanismo que identifica errores específicos por fila y campo, generando reportes detallados que facilitan la corrección por parte de los usuarios.
- **Uso de transacciones y batch processing**: Para optimizar el rendimiento, utilicé transacciones para operaciones atómicas y procesamiento por lotes para manejar grandes volúmenes de datos.

5.3 Lecciones Aprendidas

Este proceso me enseñó varias lecciones valiosas:

- **La importancia de la validación temprana**: Detectar errores lo antes posible en el proceso de desarrollo ahorra tiempo significativo en etapas posteriores.
- **El valor de la documentación continua**: Documentar cada decisión y proceso facilita no solo la colaboración sino también el mantenimiento futuro del sistema.
- **La necesidad de pensar en escalabilidad**: Diseñar soluciones que funcionen no solo para los volúmenes actuales de datos sino también para crecimientos futuros.
- **La arquitectura limpia es fundamental**: La separación clara de responsabilidades y el diseño modular facilitan no solo el desarrollo inicial sino también el mantenimiento y la extensión del sistema.
- **La experiencia de usuario es clave**: Aunque esta semana nos enfocamos principalmente en el backend, tener presente cómo interactuarán los usuarios con el sistema influyó significativamente en el diseño de la API.
- **La seguridad no es un añadido**: Integrar consideraciones de seguridad desde el inicio del proceso de desarrollo es mucho más efectivo que intentar añadirlas posteriormente.