UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3088 – Base de datos I
Sección 20
Ing. Bacilio Bolaños

Laboratorio #3 - analisis

Diego Patzán - 23525
Ihan Marroquín - 23108

GUATEMALA, 31 de mayo de 2025

Repositorio de Github con el laboratorio completo: Repositorio

1. ¿Por qué eligieron ese ORM y qué beneficios o dificultades encontraron?

Escogimos Prisma porque nos permite trabajar con la base de datos de forma sencilla y segura, lo mejor es que genera automáticamente código con tipos de datos, lo que evita errores comunes, y sus consultas son fáciles de leer, pero al principio nos costó configurarlo correctamente con Docker y tuvimos que acostumbrarnos a regenerar el cliente cada vez que hacíamos cambios en la estructura de la base.

2. ¿Cómo implementaron la lógica master-detail dentro del mismo formulario?

Implementamos un formulario unificado que maneja tanto los datos principales del estudiante como sus cursos inscritos, para esto samos estados en React para mantener la coherencia entre las secciones y validamos los datos antes de enviarlos, asegurándonos de que todo esté completo y correcto.

3. ¿Qué validaciones implementaron en la base de datos y cuáles en el código?

En la base de datos definimos reglas estrictas como campos obligatorios y valores únicos para evitar duplicados, como en el código de la aplicación añadimos validaciones adicionales, como verificar que las notas estén en el rango correcto o que un estudiante esté activo antes de inscribirlo, dando más seguridad a la aplicación.

4. ¿Qué beneficios encontraron al usar tipos de datos personalizados?

Crear tipos específicos como "NivelCurso" (Básico, Intermedio, Avanzado) nos ayudó mucho, ccon esto evitamos que se ingresen valores incorrectos, en el editor nos sugiere automáticamente las opciones disponibles, haciendo el desarrollo más rápido y con menos errores.

5. ¿Qué ventajas ofrece usar una VIEW como base del índice en vez de una consulta directa?

La vista nos simplificó mucho el trabajo porque combina datos de varias tablas en una sola estructura, esto hace que las consultas sean más sencillas y eficientes especialmente para generar reportes, tambien al tener la lógica centralizada en la vista, cualquier cambio se propaga automáticamente a todas las partes que la usan.

6. ¿Qué escenarios podrían romper la lógica actual si no existieran las restricciones?

Si no tuviéramos las restricciones de unicidad, podrían darse casos tales como un mismo estudiante inscrito múltiples veces en un curso durante el mismo período, esto generaría datos inconsistentes y afectaría la confiabilidad de los reportes, por lo tanto, las restricciones actúan como red de seguridad.

7. ¿Qué aprendieron sobre la separación entre lógica de aplicación y lógica de persistencia?

Entendimos que es crucial dividir responsabilidades: la base debe encargarse de la integridad básica de los datos, mientras que la aplicación maneja la lógica más compleja y la presentación al usuario, con esto logramos hacer que el sistema sea más mantenible y flexible ante cambios.

8. ¿Cómo escalaría este diseño en una base de datos de gran tamaño?

Para manejar más datos, implementaríamos paginación en las consultas y añadiríamos más índices a las tablas, como también consideraríamos almacenar menos datos históricos en la base principal y mover algunos a archivos o bases separadas para optimizar el rendimiento.

9. ¿Consideran que este diseño es adecuado para una arquitectura con microservicios?

El diseño actual funciona bien como monolito, pero para microservicios tendríamos que dividirlo, ya sea crear un servicio solo para estudiantes y otro para cursos, con sus propias bases de datos, la comunicación entre ellos se haría mediante APIs bien definidas.

10. ¿Cómo reutilizarían la vista en otros contextos como reportes o APIs?

La vista no solo sirve para mostrar datos en pantalla, sino que también la usamos como base para generar reportes en PDF y como fuente de datos para APIs que consumen otras aplicaciones. Esto evita tener que repetir la misma lógica en múltiples lugares.

11. ¿Qué decisiones tomaron para estructurar su modelo de datos y por qué?

Optamos por una tabla intermedia explícita para las inscripciones porque necesitábamos almacenar atributos adicionales como la nota y asistencia, los enums personalizados nos ayudaron a mantener la consistencia donde solo ciertos valores son permitidos.

12. ¿Cómo documentaron su modelo para facilitar su comprensión por otros desarrolladores?

Además de comentarios en el código, creamos un README claro con ejemplos de uso, diagramamos las relaciones entre tablas para que nuevos desarrolladores entiendan rápido cómo funciona todo.

13. ¿Cómo evitaron la duplicación de registros o errores de asignación en la tabla intermedia?

Combinamos restricciones en la base de datos (como claves únicas compuestas) con validaciones en la aplicación, previo a crear una nueva inscripción, verificamos si ya existe una igual, esto nos da doble protección contra datos duplicados.