UNIVERSIDAD DEL VALLE DE GUATEMALA

Departamento de ingeniería

Proyecto 4

Base de Datos 1

Instructor: Erick Francisco Marroquín Rodríguez

Nils Muralles Morales / 23727

Diego Oswaldo Flores Rivas / 23714

Luis Francisco Padilla Juárez / 23663

José Antonio Mérida Castejón / 201105

Isabella Recinos Rodríguez / 23003

Sección 10

Nueva Guatemala de la Asunción

Domingo 7 de Junio del año 2025

1. **¿Cuál fue el aporte técnico de cada miembro del equipo?**
    a. El trabajo se dividió equitativamente entre backend y frontend para aprovechar las fortalezas de cada integrante y asegurar un desarrollo eficiente y coordinado. La mitad del equipo se enfocó en el backend, donde se encargaron de diseñar y estructurar la base de datos, crear las migraciones, definir las funciones y vistas SQL, así como implementar la lógica de negocio y las API RESTful utilizando Laravel. Estos miembros también se responsabilizaron de la integración de reglas de negocio, validaciones, triggers y de garantizar la integridad y seguridad de los datos. Por otro lado, la otra mitad del equipo se dedicó al desarrollo del frontend utilizando React y TypeScript. Su aporte incluyó la creación de interfaces de usuario intuitivas y responsivas, la definición de los tipos personalizados para el manejo seguro de datos, y la integración con la API del backend para mostrar y manipular la información en tiempo real. Además, se encargaron de la gestión de rutas, la validación de formularios y la experiencia de usuario, asegurando que la aplicación fuera fácil de usar y visualmente atractiva.
2. **¿Qué decisiones estructurales se tomaron en el modelo de datos y por qué?**
    a. Separación de entidades principales: Se crearon tablas independientes para usuarios, productos, marcas, categorías, proveedores, almacenes, direcciones, carritos, órdenes y roles. Esto permite una gestión clara y escalable de cada entidad.
    b. Tablas de cruce para relaciones N:M: Por ejemplo, category_product, product_supplier, role_user, y address_user permiten relaciones muchos a muchos, facilitando la flexibilidad y evitando redundancia.
    c. Soporte para múltiples categorías y proveedores: Un producto puede tener varias categorías y proveedores, lo que refleja escenarios reales de e-commerce.
    d. Campos de control y estado: Se agregaron campos como is_active, status, y timestamps para controlar la vigencia y trazabilidad de los registros.
    e. Normalización y claves foráneas: Uso extensivo de claves foráneas y restricciones para mantener la integridad referencial.
3. **¿Qué criterios siguieron para aplicar la normalización?**
    a. Eliminación de redundancia: Cada entidad almacena solo sus propios datos, evitando duplicidad..
    b. Dependencias funcionales claras: Los atributos dependen completamente de la clave primaria de su tabla.
    c. Tablas de cruce para relaciones complejas: Se usaron tablas intermedias para relaciones muchos a muchos, cumpliendo con la 3FN.
    d. Separación de datos estáticos y dinámicos: Por ejemplo, roles y categorías se gestionan en tablas separadas y se relacionan mediante pivotes.
4. **¿Cómo estructuraron los tipos personalizados y para qué los usaron?**
    a. Los tipos personalizados se estructuraron principalmente mediante interfaces en TypeScript en el frontend. Estas interfaces definen la forma de los datos que se manejan en la aplicación, como productos, categorías y usuarios,

asegurando que la comunicación entre el frontend y el backend sea consistente y segura.

5. **¿Qué beneficios encontraron al usar vistas para el índice?**
    a. El uso de vistas en la base de datos trajo varios beneficios, especialmente para la generación de índices y reportes. Las vistas permiten obtener información agregada y relevante de manera eficiente, simplificando las consultas tanto para el backend como para el frontend. Esto reduce la complejidad de las consultas y mejora el rendimiento, ya que la lógica de agregación se centraliza en la base de datos y no en la aplicación.
6. **¿Cómo se aseguraron de evitar duplicidad de datos?**
    a. Para evitar la duplicidad de datos, se implementaron restricciones únicas en campos clave, como correos electrónicos y combinaciones de claves foráneas en tablas de cruce. Además, se realizaron validaciones tanto en el backend como en la base de datos para asegurar que no se puedan insertar registros duplicados. El uso de triggers también ayudó a mantener la integridad de los datos en operaciones críticas.
7. **¿Qué reglas de negocio implementaron como restricciones y por qué?**
    a. Se implementaron reglas de negocio para asegurar la validez y coherencia de los datos en el sistema. Por ejemplo, se establecieron restricciones para que los campos obligatorios, como nombre, correo electrónico y contraseña, deban estar completos y cumplir con un formato específico antes de permitir su registro. También se validó que los valores numéricos, como cantidades o precios, sean positivos y estén dentro de rangos permitidos.
8. **¿Qué trigger resultó más útil en el sistema? Justifica.**
    a. El trigger más útil en el sistema fue el encargado de actualizar el inventario reservado cuando se agregan, modifican o eliminan productos del carrito. Este trigger es fundamental porque garantiza que el stock refleje en tiempo real la disponibilidad de productos, evitando sobreventas y asegurando una experiencia de usuario confiable.
9. **¿Cuáles fueron las validaciones más complejas y cómo las resolvieron?**
    a. Las validaciones más complejas estuvieron relacionadas con asegurar la unicidad y coherencia de los datos en las relaciones entre entidades. Por ejemplo, fue necesario validar que no se asignaran categorías duplicadas a un mismo producto o que un usuario no tuviera roles repetidos. Para resolver esto, se implementaron restricciones únicas a nivel de base de datos en las tablas de cruce, lo que impide la inserción de registros duplicados. Además, se realizaron validaciones adicionales en la lógica de la aplicación para verificar que los datos cumplieran con los formatos requeridos y que los valores numéricos, como cantidades o precios, fueran positivos y estuvieran dentro de los rangos permitidos.
10. **¿Qué compromisos hicieron entre diseño ideal y rendimiento?**
    a. Aunque la normalización completa es ideal para evitar redundancias, en algunos casos se utilizaron vistas y funciones para agilizar consultas

agregadas. También se optimizó la carga de relaciones en las consultas para evitar traer datos innecesarios y mejorar la eficiencia general del sistema.