Mejores Prácticas - Sistema de Carrito de Compras

Diseño de Base de Datos

Normalización

- Seguir las reglas de normalización hasta 3NF (Tercera Forma Normal)
- Evitar la redundancia de datos
- Mantener la integridad referencial
- Utilizar claves foráneas apropiadamente

Índices

- Crear índices para campos frecuentemente consultados
- Índices para claves foráneas
- Índices para campos de búsqueda
- Monitorear el rendimiento de los índices

Convenciones de Nombrado

- Usar nombres descriptivos y en español
- Utilizar PascalCase para nombres de tablas
- Utilizar camelCase para nombres de columnas
- Prefijos consistentes para claves foráneas (ej: `cliente_id`)

Calidad de Código

Legibilidad

- Seguir un estilo de código consistente
- Documentar funciones y clases
- Usar nombres descriptivos para variables y funciones
- Mantener funciones pequeñas y enfocadas

Modularidad

- Seguir el principio de responsabilidad única
- Crear componentes reutilizables
- Implementar patrones de diseño apropiados
- Mantener el código DRY (Don't Repeat Yourself)

Manejo de Errores

- Implementar logging apropiado
- Usar try-catch para manejar excepciones
- Proporcionar mensajes de error claros
- Implementar rollback en transacciones

Seguridad

Autenticación y Autorización

- Implementar autenticación segura
- Usar hash para contraseñas (bcrypt)
- Implementar JWT para sesiones
- Control de acceso basado en roles (RBAC)

Protección de Datos

- Validar todas las entradas de usuario
- Implementar CSRF tokens
- Usar HTTPS
- Sanitizar datos de salida

Seguridad de Base de Datos

- Usar consultas preparadas
- Implementar el principio de mínimo privilegio
- Encriptar datos sensibles
- Mantener backups regulares

Escalabilidad

Arquitectura

- Diseñar para escalar horizontalmente
- Implementar caché donde sea apropiado
- Usar balanceo de carga
- Considerar microservicios para componentes críticos

Rendimiento

- Optimizar consultas de base de datos
- Implementar paginación
- Usar lazy loading
- Minimizar llamadas a la base de datos

Monitoreo

- Implementar logging centralizado
- Monitorear métricas de rendimiento
- Configurar alertas
- Mantener documentación actualizada

Testing

Pruebas Unitarias

- Escribir pruebas para cada componente
- Mantener alta cobertura de código
- Automatizar pruebas
- Seguir el patrón AAA (Arrange-Act-Assert)

Pruebas de Integración

- Probar interacciones entre componentes
- Verificar flujos de trabajo completos
- Probar casos de error
- Simular dependencias externas

Pruebas de Rendimiento

- Realizar pruebas de carga
- Monitorear tiempos de respuesta
- Identificar cuellos de botella
- Optimizar basado en resultados

Documentación

Código

- Documentar APIs
- Mantener README actualizado
- Documentar decisiones de arquitectura
- Incluir ejemplos de uso

Usuario

- Proporcionar guías de usuario
- Documentar procesos de instalación
- Mantener documentación de API actualizada
- Incluir ejemplos y casos de uso

Control de Versiones

Git

- Usar ramas feature
- Mantener commits atómicos
- Escribir mensajes de commit descriptivos
- Revisar código antes de merge

CI/CD

- Automatizar builds
- Implementar pruebas automáticas
- Automatizar despliegues
- Mantener pipeline de CI/CD