Grupo: Alex, Martín, Erick

EJERCICIO 3

Objetivo:
Implementar una aplicación móvil que permita a los clientes realizar compras en línea, programar entregas a domicilio, y recibir notificaciones sobre las ofertas especiales.

Requerimentos funcionales:
*Clientes
Buscar productos por categoría
Ver detalles del producto
Agregar productos al carrito de compras
Realizar pago a través de la aplicación
Programar entrega en franjas horarias específicas
Rastrear estado del pedido

*Administradores
Ofrecer funcionalidades para actualizar el inventario
Establecer precios
Gestionar promociones
Visualizar reportes de ventas y tendencias de consumo

Requerimentos no funcionales:
Usabilidad: interfaz intuitiva, fácil de usar y accesible en diversos dispositivos móviles
Seguridad: implementar autenticación segura para usuarios y cifrado de datos sensibles.
Disponibilidad: 24/7.
Rendimiento: constante incluso durante picos de alta demanda, tiempos de respuesta rápidos y sin interrupciones.
Escalabilidad: escalable para soportar un aumento en el número de usuarios y transacciones durante periodos de alta demanda, como festividades o primos.

Clases / Entidades
Clase: Persona

Atributos:
- nombre: String
- apellidos: String
- id: Int
- DNI: String
- correo: String
- telefono: Int
- contraseña: String
- fecha de nacimiento: Date
- rol: String
- fecha de creación: Date
- direccion: String

Métodos:
- iniciarSesion(String correo, String contraseñas): Boolean
+ getNombre(): String
+ setNombre(String nombre): void
- cerrarSesion(): void
+ verProducto(Producto productos): Boolean
+ rastrearEntregaProducto(): void

Subclase: Usuario hereda Persona

Atributos:
- comprasRealizadas: List<Producto>
- tarjeta: Tarjeta

Métodos: + agregarProductoCarrito(Producto productos): void
+ programarEntregaProducto (): void
+ verPromocion(Promocion promociones): Boolean

Subclase: Admin hereda Persona
Atributos: —

Métodos: + administrarProducto(Producto productos): Producto
+ agregarPromocion(): Promocion
+ administrarPromocion(Promocion promociones): Boolean

Clase: Producto
Atributos: - nombre: String
- id: Int
- categoria: String
- precio: Float
- cantidad: Int
- fecha de creación: Date
- fecha de caducidad: Date

Métodos: + crearProducto(): Boolean
+ despublicarProducto(Producto productos): void
- administrarProducto(Producto productos): Producto

Clase: Inventario
Atributos: - productos: List<Product>

Métodos: + crearInventario(): Boolean
+ buscarProducto(String categoria): List<Producto>

Clase: Carrito
Atributos: - productosAgregados: List<producto>

Métodos: + modificarProductoAgregado(Int cantidad): Boolean
+ eliminarProductoAgregado(): void

Clase: Pago
Atributos: - productosCarrito: List<carrito>

Métodos: + pagarProducto(Tarjeta tarjetas): Boolean

Clase: Promocion
Atributos: - precio descuento: Float
- texto: String
- id: Int
- categoria: String
- fecha inicio: Date
- fecha final: Date

Métodos: + crearPromocion(): Boolean
+ despublicarPromocion(Promocion promociones): void
- administrarPromocion(Promocion promociones): Promocion

Clase: Tarjeta

Atributos:
- nombre: String
- apellidos: String
- numero de tarjeta: Int
- clave: int
- fecha de caducidad: Date

Métodos: -verificarTarjeta (Tarjeta tarjetas): Boolean

Clase: Notificación:

Atributos:
- fecha: Date
- Texto: Srting
- id: Int

Métodos: + generarNotificacion(List<Producto>, String texto, Date fecha, Int id): String mensaje
+ enviarNotificacion(String mensaje, email): void

Clase: Reporte:

Atributos:
- nombre: String
- id: Int
- fecha: Date
- inventario: Inventario

Métodos:
+ generarReportePDF(): File
+ enviarReporte(File fichero): void

Relaciones:

Una persona registra a 1 usuario. Cardinalidad: 1

Un usuario puede registrar de 0 a múltiples tarjetas: Cardinalidad: 0..*

Un usuario puede hacer de 0 a múltiples compras: Cardinalidad: 0..*

Un usuario puede recibir de 0 a múltiples notificaciones: Cardinalidad: 0..*

Una notificación puede ser enviada de uno a múltiples usuarios: Cardinalidad: 1..*

Un inventario puede generar de 0 a múltiples notificaciones: Cardinalidad: 0..*

Un inventario puede generar de 1 a múltiples reportes: Cardinalidad: 1..*

Un producto puede tener de 0 a múltiples promociones: Cardinalidad: 0..*

Tipos de relaciones:

Persona - Usuario: Herencia.
Persona - Admin: Herencia.
Usuario - Admin: Asociación.

Producto - Inventario: Composición.
Usuario - Inventario: Asociación.

Admin - Inventario: Asociación

Producto - Notificación: Asociación.
Producto - Promoción: Asociación.
Producto - Carrito: Agregación.
Usuario - Producto: Asociación.

Inventario - Notificación: Asociación.
Inventario - Reporte: Asociación.

Notificación - Usuario: Asociación.
Notificación - Inventario: Asociación

Carrito - Pago: Agregación.
Usuario - Carrito: Asociación.

Pago - Tarjeta: Dependencia.
Usuario - Tarjeta: Asociación.

Recomendaciones/Ambiguedades:
Funcionales:
Contratar servidores externos

Crear factores de multiple autentificación

Crear FAQs

No funcionales:
-