“SISTEMA DE GESTIÓN DE INVENTARIO Y VENTAS”

Ferreteria DSA

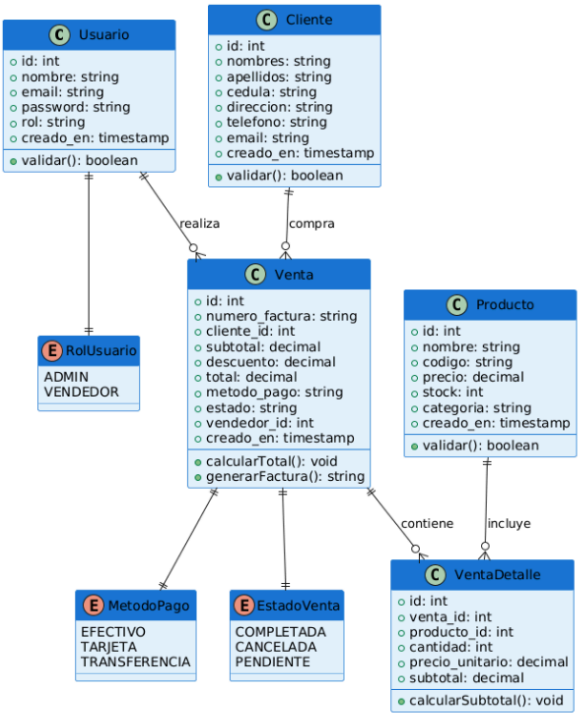
Diagrama de Clases

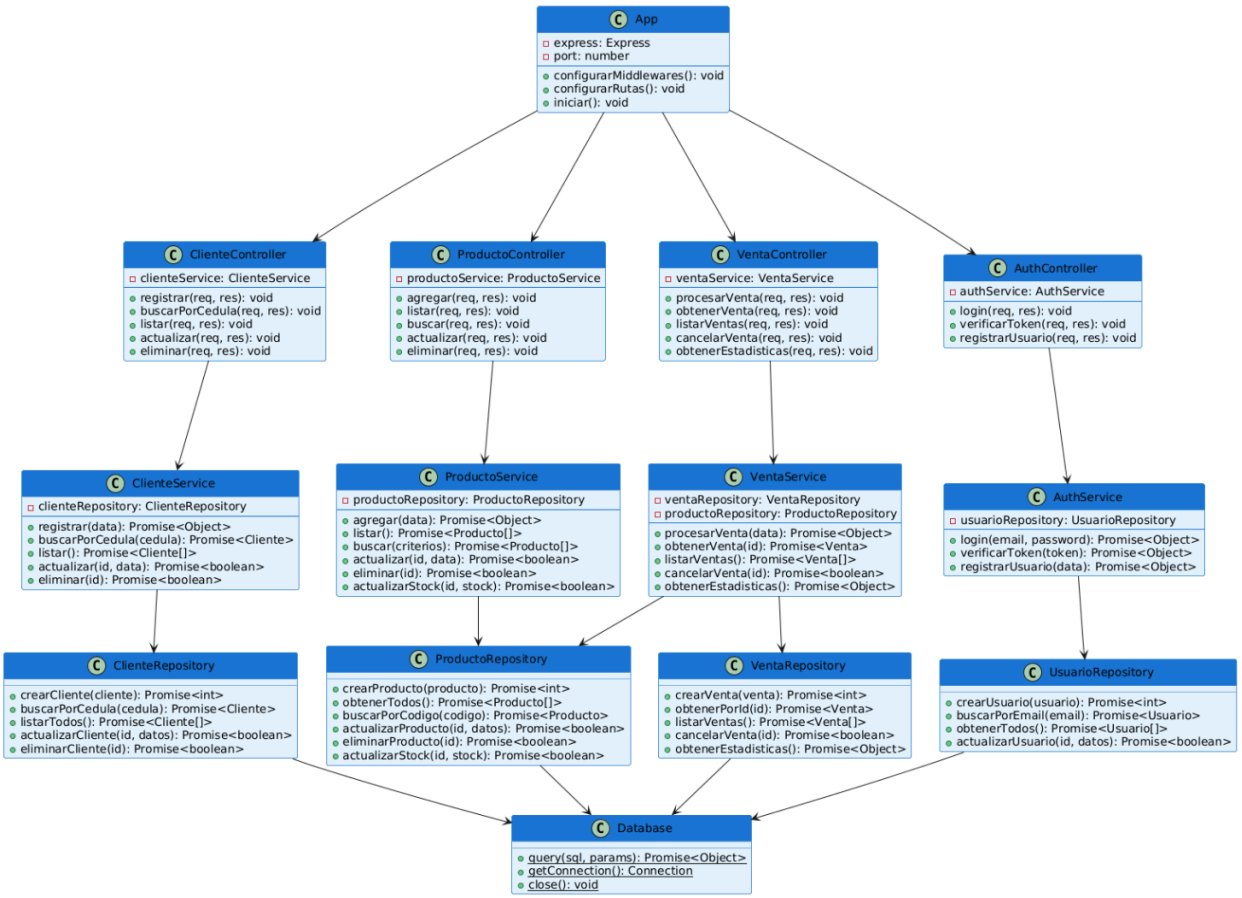
Versión 1.0

Presentado por: Jaya Carlos, Chipe Pamela, Pabon Elkin

Director: Ing. Jenny Ruiz

Diagrama de Clases:





El diagrama se divide en dos partes principales para garantizar una arquitectura limpia:

- **Modelos de dominio:** Representan las entidades del negocio como Cliente, Producto y Venta, junto con sus atributos, métodos y relaciones.
- **Backend funcional:** Define cómo interactúa el sistema con los modelos mediante controladores, servicios y repositorios, separando la lógica del negocio de la infraestructura.

Esta división mejora la organización, facilita el mantenimiento y permite una escalabilidad adecuada del sistema.

Codigo de PlantUML:

```
@startuml
!define CLASS_COLOR #E3F2FD
!define ENUM_COLOR #F3E5F5

skinparam class {
  BackgroundColor CLASS_COLOR
  BorderColor #1976D2
  HeaderBackgroundColor #1976D2
  HeaderFontColor white
}

' ===== MODELOS =====
class Usuario {
  + id: int
  + nombre: string
  + email: string
  + password: string
  + rol: string
  + creado_en: timestamp
  --
  + validar(): boolean
}

class Cliente {
  + id: int
  + nombres: string
  + apellidos: string
  + cedula: string
  + direccion: string
  + telefono: string
  + email: string
  + creado_en: timestamp
  --
  + validar(): boolean
}

class Producto {
  + id: int
  + nombre: string
  + codigo: string
  + precio: decimal
  + stock: int
  + categoria: string
  + creado_en: timestamp
  --
  + validar(): boolean
}
```

```
class Venta {
  + id: int
  + numero_factura: string
  + cliente_id: int
  + subtotal: decimal
  + descuento: decimal
  + total: decimal
  + metodo_pago: string
  + estado: string
  + vendedor_id: int
  + creado_en: timestamp
  --
  + calcularTotal(): void
  + generarFactura(): string
}

class VentaDetalle {
  + id: int
  + venta_id: int
  + producto_id: int
  + cantidad: int
  + precio_unitario: decimal
  + subtotal: decimal
  --
  + calcularSubtotal(): void
}

' ===== ENUMS =====
enum RolUsuario {
  ADMIN
  VENDEDOR
}

enum MetodoPago {
  EFECTIVO
  TARJETA
  TRANSFERENCIA
}

enum EstadoVenta {
  COMPLETADA
  CANCELADA
  PENDIENTE
}

' ===== REPOSITORIES =====
class ClienteRepository {
  + crearCliente(cliente): Promise<int>
  + buscarPorCedula(cedula): Promise<Cliente>
  + listarTodos(): Promise<Cliente[]>
  + actualizarCliente(id, datos): Promise<boolean>
  + eliminarCliente(id): Promise<boolean>
}

class ProductoRepository {
  + crearProducto(producto): Promise<int>
  + obtenerTodos(): Promise<Producto[]>
  + buscarPorCodigo(codigo): Promise<Producto>
  + actualizarProducto(id, datos): Promise<boolean>
  + eliminarProducto(id): Promise<boolean>
  + actualizarStock(id, stock): Promise<boolean>
}

class VentaRepository {
  + crearVenta(venta): Promise<int>
```

```
  + obtenerPorId(id): Promise<Venta>
  + listarVentas(): Promise<Venta[]>
  + cancelarVenta(id): Promise<boolean>
  + obtenerEstadisticas(): Promise<Object>
}

class UsuarioRepository {
  + crearUsuario(usuario): Promise<int>
  + buscarPorEmail(email): Promise<Usuario>
  + obtenerTodos(): Promise<Usuario[]>
  + actualizarUsuario(id, datos): Promise<boolean>
}

' ===== SERVICES =====
class ClienteService {
  - clienteRepository: ClienteRepository
  --
  + registrar(data): Promise<Object>
  + buscarPorCedula(cedula): Promise<Cliente>
  + listar(): Promise<Cliente[]>
  + actualizar(id, data): Promise<boolean>
  + eliminar(id): Promise<boolean>
}

class ProductoService {
  - productoRepository: ProductoRepository
  --
  + agregar(data): Promise<Object>
  + listar(): Promise<Producto[]>
  + buscar(criterios): Promise<Producto[]>
  + actualizar(id, data): Promise<boolean>
  + eliminar(id): Promise<boolean>
  + actualizarStock(id, stock): Promise<boolean>
}

class VentaService {
  - ventaRepository: VentaRepository
  - productoRepository: ProductoRepository
  --
  + procesarVenta(data): Promise<Object>
  + obtenerVenta(id): Promise<Venta>
  + listarVentas(): Promise<Venta[]>
  + cancelarVenta(id): Promise<boolean>
  + obtenerEstadisticas(): Promise<Object>
}

class AuthService {
  - usuarioRepository: UsuarioRepository
  --
  + login(email, password): Promise<Object>
  + verificarToken(token): Promise<Object>
  + registrarUsuario(data): Promise<Object>
}

' ===== CONTROLLERS =====
class ClienteController {
  - clienteService: ClienteService
  --
  + registrar(req, res): void
  + buscarPorCedula(req, res): void
  + listar(req, res): void
  + actualizar(req, res): void
  + eliminar(req, res): void
}

class ProductoController {
```

```
  - productoService: ProductoService
  --
  + agregar(req, res): void
  + listar(req, res): void
  + buscar(req, res): void
  + actualizar(req, res): void
  + eliminar(req, res): void
}

class VentaController {
  - ventaService: VentaService
  --
  + procesarVenta(req, res): void
  + obtenerVenta(req, res): void
  + listarVentas(req, res): void
  + cancelarVenta(req, res): void
  + obtenerEstadisticas(req, res): void
}

class AuthController {
  - authService: AuthService
  --
  + login(req, res): void
  + verificarToken(req, res): void
  + registrarUsuario(req, res): void
}

' ===== CONFIGURACION =====
class Database {
  + {static} query(sql, params): Promise<Object>
  + {static} getConnection(): Connection
  + {static} close(): void
}

class App {
  - express: Express
  - port: number
  --
  + configurarMiddlewares(): void
  + configurarRutas(): void
  + iniciar(): void
}

' ===== RELACIONES =====

' Modelos - Enums
Usuario ||--|| RolUsuario
Venta ||--|| MetodoPago
Venta ||--|| EstadoVenta

' Relaciones entre modelos
Usuario ||--o{ Venta : "realiza"
Cliente ||--o{ Venta : "compra"
Venta ||--o{ VentaDetalle : "contiene"
Producto ||--o{ VentaDetalle : "incluye"

' Services -> Repositories
ClienteService --> ClienteRepository
ProductoService --> ProductoRepository
VentaService --> VentaRepository
VentaService --> ProductoRepository
AuthService --> UsuarioRepository

' Controllers -> Services
ClienteController --> ClienteService
ProductoController --> ProductoService
```

```
VentaController --> VentaService
AuthController --> AuthService

' Repositories -> Database
ClienteRepository --> Database
ProductoRepository --> Database
VentaRepository --> Database
UsuarioRepository --> Database

' App -> Controllers
App --> ClienteController
App --> ProductoController
App --> VentaController
App --> AuthController
@enduml
```