"Año de la Unidad, la Paz y el Desarrollo"

Implementación de Aplicaciones Multiplataforma para el correcto Funcionamiento y Gestión de la Pastelería y Dulcería Velazco

Curso: Diseño e implementación de arquitectura empresarial

Profesor:
Mg. Gonzales Castilla, Ángel Ernesto

Integrantes:

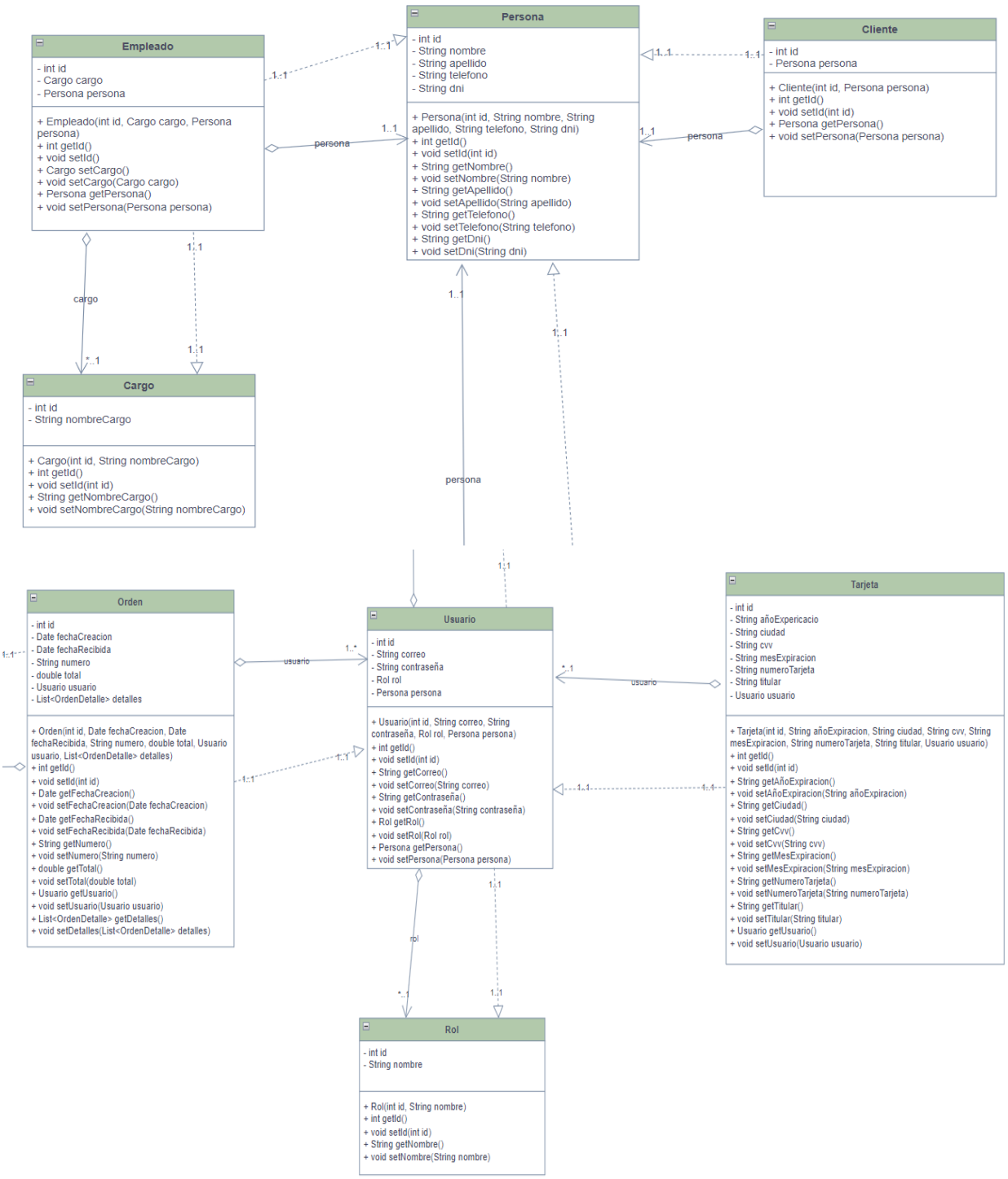
Bonifaz Campos, Jorge Armando	U20206004
Huallcca Anyosa, Brandon Mark	U20209633
Perez de la Borda, Jack Aymar	U20217506
Rodas Flores, Carlos Esteban	U20228352
Gutierrez Redolfo, Cristopher Walken	U20217372

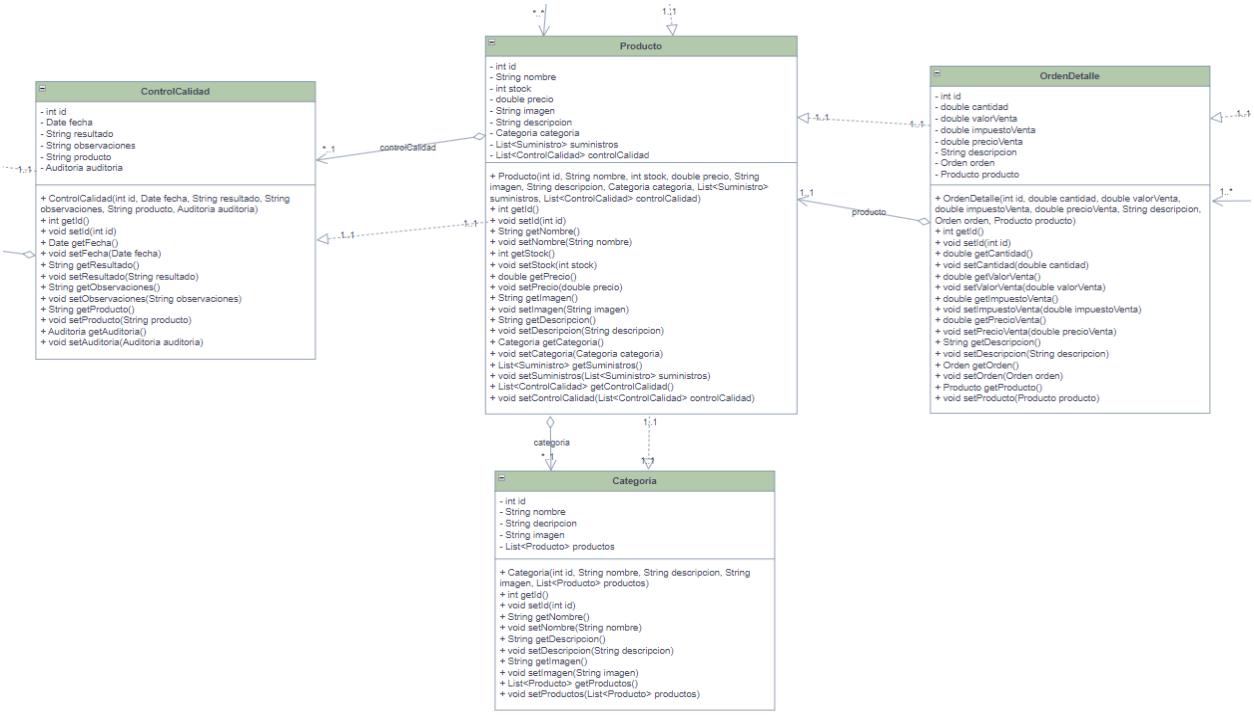
Ciclo: VII

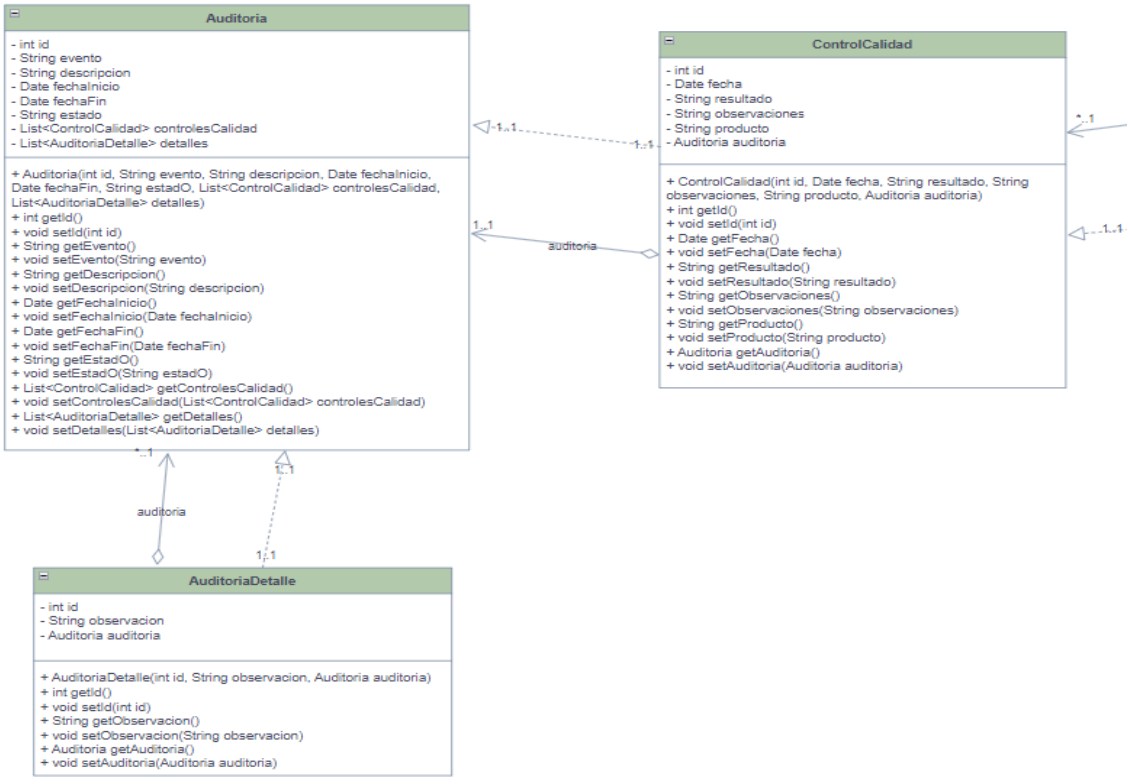
Ica – Perú
2023

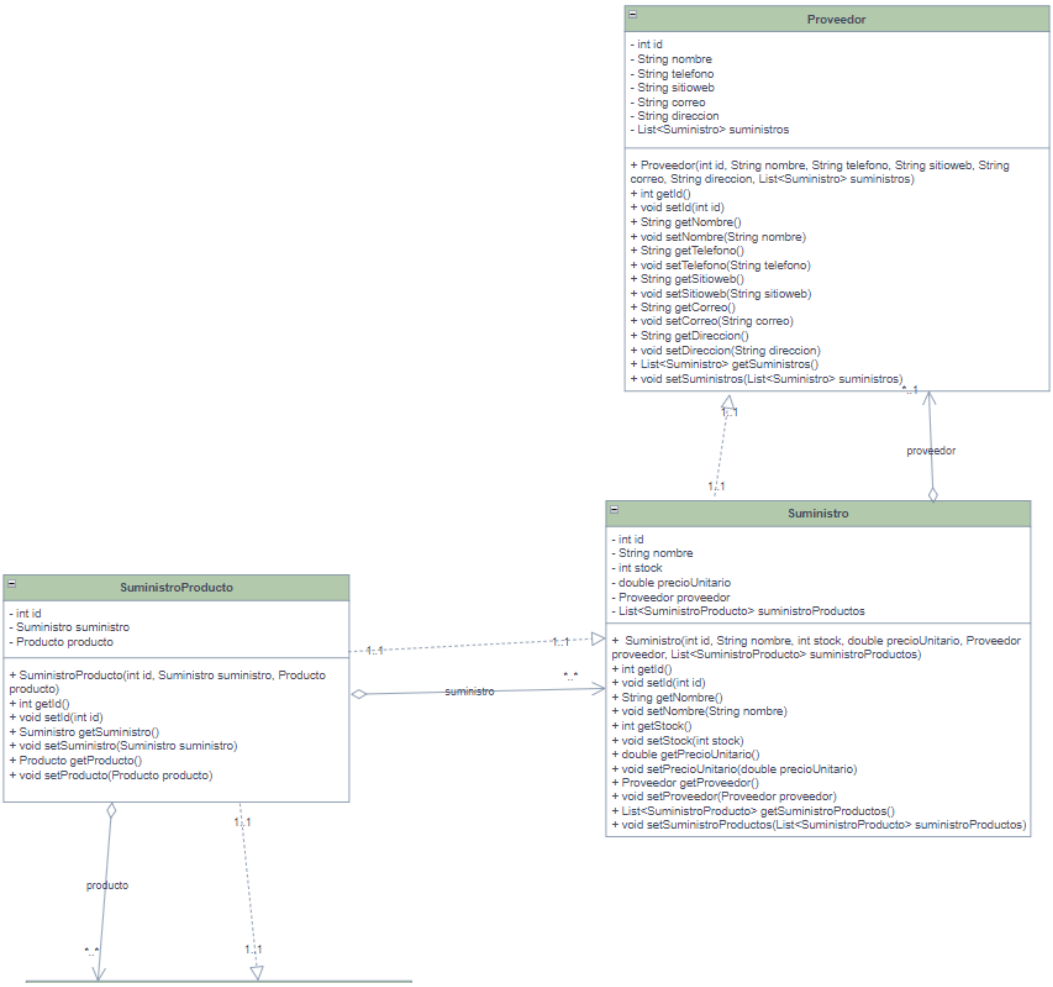
ÍNDICE:

18. DIAGRAMA DE CLASES:

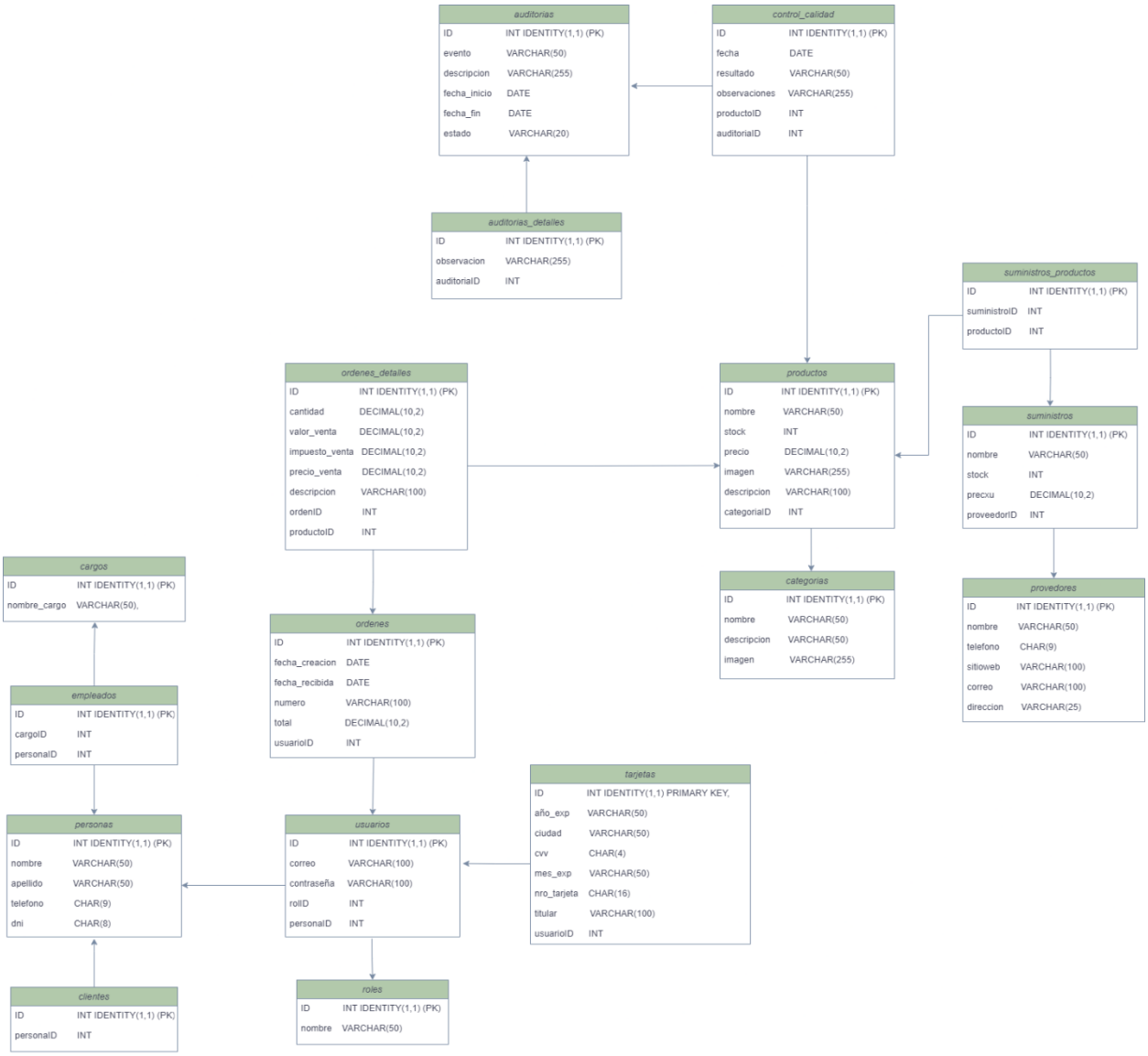



Producto
- int id
- String nombre
- int stock
- double precio
- String imagen
- String descripcion
- Categoria categoria
- List<Suministro> suministros
- List<ControlCalidad> controlCalidad
+ Producto(int id, String nombre, int stock, double precio, String imagen, String descripcion, Categoria categoria, List<Suministro> suministros, List<ControlCalidad> controlCalidad)
+ int getId()
+ void setId(int id)
+ String getNombre()
+ void setNombre(String nombre)
+ int getStock()
+ void setStock(int stock)
+ double getPrecio()
+ void setPrecio(double precio)
+ String getImagen()
+ void setImagen(String imagen)
+ String getDescripcion()
+ void setDescripcion(String descripcion)
+ Categoria getCategoria()
+ void setCategoria(Categoria categoria)
+ List<Suministro> getSuministros()
+ void setSuministros(List<Suministro> suministros)
+ List<ControlCalidad> getControlCalidad()
+ void setControlCalidad(List<ControlCalidad> controlCalidad)
ControlCalidad
- int id
- Date fecha
- String resultado
- String observaciones
- String producto
- Auditoria auditoria
+ ControlCalidad(int id, Date fecha, String resultado, String observaciones, String producto, Auditoria auditoria)
+ int getId()
+ void setId(int id)
+ Date getFecha()
+ void setFecha(Date fecha)
+ String getResultado()
+ void setResultado(String resultado)
+ String getObservaciones()
+ void setObservaciones(String observaciones)
+ String getProducto()
+ void setProducto(String producto)
+ Auditoria getAuditoria()
+ void setAuditoria(Auditoria auditoria)
controlCalidad
OrdenDetalle
- int id
- double cantidad
- double valorVenta
- double impuestoVenta
- double precioVenta
- String descripcion
- Orden orden
- Producto producto
+ OrdenDetalle(int id, double cantidad, double valorVenta, double impuestoVenta, double precioVenta, String descripcion, Orden orden, Producto producto)
+ int getId()
+ void setId(int id)
+ double getCantidad()
+ void setCantidad(double cantidad)
+ double getValorVenta()
+ void setValorVenta(double valorVenta)
+ double getImpuestoVenta()
+ void setImpuestoVenta(double impuestoVenta)
+ double getPrecioVenta()
+ void setPrecioVenta(double precioVenta)
+ String getDescripcion()
+ void setDescripcion(String descripcion)
+ Orden getOrden()
+ void setOrden(Orden orden)
+ Producto getProducto()
+ void setProducto(Producto producto)
producto
categoria
Categoria
- int id
- String nombre
- String decripcion
- String imagen
- List<Producto> productos
+ Categoria(int id, String nombre, String descripcion, String imagen, List<Producto> productos)
+ int getId()
+ void setId(int id)
+ String getNombre()
+ void setNombre(String nombre)
+ String getDescripcion()
+ void setDescripcion(String descripcion)
+ String getImagen()
+ void setImagen(String imagen)
+ List<Producto> getProductos()
+ void setProductos(List<Producto> productos)


Auditoria
- int id
- String evento
- String descripcion
- Date fechaInicio
- Date fechaFin
- String estado
- List<ControlCalidad> controlesCalidad
- List<AuditoriaDetalle> detalles
+ Auditoria(int id, String evento, String descripcion, Date fechaInicio, Date fechaFin, String estadO, List<ControlCalidad> controlesCalidad, List<AuditoriaDetalle> detalles)
+ int getId()
+ void setId(int id)
+ String getEvento()
+ void setEvento(String evento)
+ String getDescripcion()
+ void setDescripcion(String descripcion)
+ Date getFechaInicio()
+ void setFechaInicio(Date fechaInicio)
+ Date getFechaFin()
+ void setFechaFin(Date fechaFin)
+ String getEstadO()
+ void setEstadO(String estadO)
+ List<ControlCalidad> getControlesCalidad()
+ void setControlesCalidad(List<ControlCalidad> controlesCalidad)
+ List<AuditoriaDetalle> getDetalles()
+ void setDetalles(List<AuditoriaDetalle> detalles)
ControlCalidad
- int id
- Date fecha
- String resultado
- String observaciones
- String producto
- Auditoria auditoria
+ ControlCalidad(int id, Date fecha, String resultado, String observaciones, String producto, Auditoria auditoria)
+ int getId()
+ void setId(int id)
+ Date getFecha()
+ void setFecha(Date fecha)
+ String getResultado()
+ void setResultado(String resultado)
+ String getObservaciones()
+ void setObservaciones(String observaciones)
+ String getProducto()
+ void setProducto(String producto)
+ Auditoria getAuditoria()
+ void setAuditoria(Auditoria auditoria)
auditoria
auditoria
AuditoriaDetalle
- int id
- String observacion
- Auditoria auditoria
+ AuditoriaDetalle(int id, String observacion, Auditoria auditoria)
+ int getId()
+ void setId(int id)
+ String getObservacion()
+ void setObservacion(String observacion)
+ Auditoria getAuditoria()
+ void setAuditoria(Auditoria auditoria)


Proveedor
- int id
- String nombre
- String telefono
- String sitioweb
- String correo
- String direccion
- List<Suministro> suministros
+ Proveedor(int id, String nombre, String telefono, String sitioweb, String correo, String direccion, List<Suministro> suministros)
+ int getId()
+ void setId(int id)
+ String getNombre()
+ void setNombre(String nombre)
+ String getTelefono()
+ void setTelefono(String telefono)
+ String getSitioweb()
+ void setSitioweb(String sitioweb)
+ String getCorreo()
+ void setCorreo(String correo)
+ String getDireccion()
+ void setDireccion(String direccion)
+ List<Suministro> getSuministros()
+ void setSuministros(List<Suministro> suministros)
1..1
proveedor
Suministro
- int id
- String nombre
- int stock
- double precioUnitario
- Proveedor proveedor
- List<SuministroProducto> suministroProductos
+ Suministro(int id, String nombre, int stock, double precioUnitario, Proveedor proveedor, List<SuministroProducto> suministroProductos)
+ int getId()
+ void setId(int id)
+ String getNombre()
+ void setNombre(String nombre)
+ int getStock()
+ void setStock(int stock)
+ double getPrecioUnitario()
+ void setPrecioUnitario(double precioUnitario)
+ Proveedor getProveedor()
+ void setProveedor(Proveedor proveedor)
+ List<SuministroProducto> getSuministroProductos()
+ void setSuministroProductos(List<SuministroProducto> suministroProductos)
SuministroProducto
- int id
- Suministro suministro
- Producto producto
+ SuministroProducto(int id, Suministro suministro, Producto producto)
+ int getId()
+ void setId(int id)
+ Suministro getSuministro()
+ void setSuministro(Suministro suministro)
+ Producto getProducto()
+ void setProducto(Producto producto)
suministro
producto

19. ENTIDAD-RELACION:



20. CÓDIGO DE CREACIÓN DE BASE DE DATOS:

20.1.BD_VELAZCO:

```
CREATE DATABASE BD_Velazco
USE BD_Velazco

CREATE TABLE roles(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	nombre 			VARCHAR(50),
);

CREATE TABLE cargos(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	nombre_cargo 		VARCHAR(50),
);


CREATE TABLE personas(
    ID 			INT IDENTITY(1,1) PRIMARY KEY,
	nombre 			VARCHAR(50),
	apellido 		VARCHAR(50),
	telefono 		CHAR(9),
	dni 			CHAR(8),
);


CREATE TABLE empleados(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	cargoID 		INT,
	personaID 		INT,
);

CREATE TABLE clientes(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	personaID 		INT,
);

CREATE TABLE usuarios(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	correo 			VARCHAR(100),
	contraseña 		VARCHAR(100),
	rolID 			INT,
	personaID 		INT,
);

/* PRODUCTOS */

CREATE TABLE categorias(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	nombre 			VARCHAR(50),
	descripcion 		VARCHAR(50),
	imagen 			VARCHAR(255),
);

CREATE TABLE proveedores(
	ID 			INT IDENTITY(1,1) PRIMARY KEY,
	nombre 			VARCHAR(50),
```

```
	telefono		CHAR(9),
	sitioweb		VARCHAR(100),
	correo			VARCHAR(100),
	direccion		VARCHAR(25),
);

CREATE TABLE suministros(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	nombre			VARCHAR(50),
	stock			INT,
	precxu			DECIMAL(10,2),
	proveedorID		INT,
);

CREATE TABLE suministros_productos(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	suministroID		INT,
	productoID		INT,
);

CREATE TABLE productos(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	nombre			VARCHAR(50),
	stock			INT,
	precio			DECIMAL(10,2),
	imagen			VARCHAR(255),
	descripcion		VARCHAR(100),
	categoriaID		INT,
);


CREATE TABLE control_calidad(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	fecha			DATE,
	resultado		VARCHAR(50),
	observaciones		VARCHAR(255),
	productoID		INT,
	auditoriaID		INT,
);

CREATE TABLE auditorias(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	evento			VARCHAR(50),
	descripcion		VARCHAR(255),
	fecha_inicio		DATE,
	fecha_fin		DATE,
	estado			VARCHAR(20),

);

CREATE TABLE auditorias_detalles(
	ID			INT IDENTITY(1,1) PRIMARY KEY,
	observacion		VARCHAR(255),
	auditoriaID		INT,
);


/*TABLAS DE VENTAS*/
```

```sql
CREATE TABLE ordenes(
	ID 				INT IDENTITY(1,1) PRIMARY KEY,
	fecha_creacion 		DATE,
	fecha_recibida 		DATE,
	numero 			VARCHAR(100),
	total 			DECIMAL(10,2),
	usuarioID 			INT,
);

CREATE TABLE ordenes_detalles(
	ID 				INT IDENTITY(1,1) PRIMARY KEY,
	cantidad 			DECIMAL(10,2),
	valor_venta 		DECIMAL(10,2),
	impuesto_venta 		DECIMAL(10,2),
	precio_venta 		DECIMAL(10,2),
	descripcion 		VARCHAR(100),
	ordenID 			INT,
	productoID 			INT,
);

CREATE TABLE tarjetas(
	ID 				INT IDENTITY(1,1) PRIMARY KEY,
	año_exp 			VARCHAR(50),
	ciudad 			VARCHAR(50),
	cvv 				CHAR(4),
	mes_exp 			VARCHAR(50),
	nro_tarjeta 		CHAR(16),
	titular 			VARCHAR(100),
	usuarioID 			INT,
);

/*LLAVES FORANEAS*/

ALTER TABLE usuarios
ADD FOREIGN KEY (rolID) REFERENCES roles(ID);

ALTER TABLE empleados
ADD FOREIGN KEY (personaID) REFERENCES personas(ID);

ALTER TABLE clientes
ADD FOREIGN KEY (personaID) REFERENCES personas(ID);

ALTER TABLE empleados
ADD FOREIGN KEY (cargoID) REFERENCES cargos(ID);

ALTER TABLE usuarios
ADD FOREIGN KEY (personaID) REFERENCES personas(ID);


/*LLAVES PRODUCTOS*/

ALTER TABLE productos
ADD FOREIGN KEY (categoriaID) REFERENCES categorias(ID);

ALTER TABLE suministros
ADD FOREIGN KEY (proveedorID) REFERENCES proveedores(ID);
```

```
ALTER TABLE suministros_productos
ADD FOREIGN KEY (suministroID) REFERENCES suministros(ID);

ALTER TABLE suministros_productos
ADD FOREIGN KEY (productoID) REFERENCES productos(ID);

ALTER TABLE control_calidad
ADD FOREIGN KEY (productoID) REFERENCES productos(ID);

ALTER TABLE control_calidad
ADD FOREIGN KEY (auditoriaID) REFERENCES auditorias(ID);

ALTER TABLE auditorias_detalles
ADD FOREIGN KEY (auditoriaID) REFERENCES auditorias(ID);


/*LLAVES DE ORDENES*/

ALTER TABLE ordenes
ADD FOREIGN KEY (usuarioID) REFERENCES usuarios(ID);

ALTER TABLE ordenes_detalles
ADD FOREIGN KEY (productoID) REFERENCES productos(ID);

ALTER TABLE ordenes_detalles
ADD FOREIGN KEY (ordenID) REFERENCES ordenes(ID);

ALTER TABLE tarjetas
ADD FOREIGN KEY (usuarioID) REFERENCES usuarios(ID);
```

20.2.PROCEDIMIENTOS ALMACENADOS:

a) Procedimientos para la tabla Personas:

- ✓ CrearPersona
- ✓ ObtenerTodasLasPersonas
- ✓ ObtenerPersonaPorID
- ✓ ActualizarPersona
- ✓ EliminarPersona

b) Procedimientos para la tabla Cargos:

- ✓ CrearCargo
- ✓ ObtenerTodosLosCargos
- ✓ ObtenerCargoPorID
- ✓ ActualizarCargo
- ✓ EliminarCargo

c) Procedimientos para la tabla Empleados:
   - ✓ CrearEmpleado
   - ✓ ObtenerEmpleadoPorID
   - ✓ ObtenerTodosLosEmpleados
   - ✓ ActualizarEmpleado
   - ✓ EliminarEmpleado

d) Procedimientos para la tabla Clientes:
   - ✓ CrearCliente
   - ✓ ObtenerTodosLosClientes
   - ✓ ObtenerClientePorID
   - ✓ ActualizarCliente
   - ✓ EliminarCliente

e) Procedimientos para la tabla Roles:
   - ✓ CrearRol
   - ✓ ObtenerTodosLosRoles
   - ✓ ObtenerRolPorID
   - ✓ ActualizarRol
   - ✓ EliminarRol

f) Procedimientos para la tabla Usuarios:
   - ✓ CrearUsuario
   - ✓ ObtenerTodosLosUsuarios
   - ✓ ObtenerUsuarioPorID

g) Procedimientos para la tabla Tarjetas:
   - ✓ CrearTarjeta
   - ✓ EliminarTarjeta

h) Procedimientos para la tabla Categorias:
   - ✓ CrearCategoria
   - ✓ ObtenerTodasLasCategorias
   - ✓ ObtenerDetalleCategoria
   - ✓ ActualizarCategoria
   - ✓ EliminarCategoria

i) Procedimientos para la tabla Productos:
   - ✓ CrearProducto
   - ✓ ObtenerTodosLosProductos
   - ✓ ObtenerDetalleProducto
   - ✓ ActualizarProducto
   - ✓ EliminarProducto

j) Procedimientos para la tabla Auditorias:
   - ✓ CrearAuditoria
   - ✓ ObtenerTodasLasAuditorias
   - ✓ ObtenerDetalleAuditoria
   - ✓ ActualizarAuditoria
   - ✓ EliminarAuditoria

k) Procedimientos para la tabla Control de Calidad:
   - ✓ CrearControlCalidad
   - ✓ ObtenerTodosLosControlesCalidad
   - ✓ ObtenerDetalleControlCalidad
   - ✓ ActualizarControlCalidad
   - ✓ EliminarControlCalidad

l) Procedimientos para la tabla Proveedores:
   - ✓ CrearProveedor
   - ✓ ObtenerTodosLosProveedores
   - ✓ ObtenerDetalleProveedor
   - ✓ ActualizarProveedor
   - ✓ EliminarProveedor

m) Procedimientos para la tabla Suministros:
   - ✓ CrearSuministro
   - ✓ ObtenerTodosLosSuministros
   - ✓ ObtenerDetalleSuministro
   - ✓ ActualizarSuministro
   - ✓ EliminarSuministro

n) Procedimientos para la tabla Suministro-Productos:

- ✓ CrearSuministroProducto
- ✓ ObtenerTodosLosSuministrosProductos
- ✓ ObtenerDetalleSuministroProducto
- ✓ ActualizarSuministroProducto
- ✓ EliminarSuministroProducto

o) Procedimientos para la tabla Orden:

- ✓ CrearOrden
- ✓ ObtenerTodasLasOrdenes
- ✓ ObtenerDetalleOrden
- ✓ ActualizarOrden
- ✓ EliminarOrden

p) Procedimientos para la tabla Detalle Orden:

- ✓ CrearDetalleOrden
- ✓ ObtenerTodosLosDetallesDeOrdenes
- ✓ ObtenerDetalleOrdenPorID
- ✓ ActualizarDetalleOrden
- ✓ EliminarDetalleOrden

21. PATRÓN DE DISEÑO EMPLEADO:

ARQUITECTURA MVC.

La arquitectura MVC (modelo, vista, controlador) consiste en un patrón de diseño de software que se utiliza para separar en tres componentes los datos, la metodología y la interfaz gráfica de una aplicación. La gran ventaja que posee esta técnica de programación es que permite modificar cada uno de ellos sin necesidad de modificar los demás, lo que permite desarrollar aplicaciones modulares y escalables que se puedan actualizar fácilmente y añadir o eliminar nuevos módulos o funcionalidades de forma paquetizada, ya que cada "paquete" utiliza el mismo sistema con sus vistas, modelos y controladores. Los tres componentes de un sistema basado en arquitectura MVC son:

- **Modelo**: El Modelo se encarga de manipular, gestionar y actualizar los datos. Si se utiliza una base de datos aquí es donde se realizan las consultas, búsquedas, filtros y actualizaciones.

- **Vista**: La Vista sirve para mostrar le al usuario final la interfaz gráfica (pantallas, ventanas, páginas, formularios…) como resultado de una solicitud enviada a través del controlador. Desde la perspectiva del programador este componente es el que se encarga del front-end; la programación de la interfaz de usuario si se trata de una aplicación de escritorio, o bien, la visualización de las páginas web (CSS, HTML, HTML5 y JavaScript).
- **Controlador**: es el componente principal de la aplicación, donde se especifican los métodos y funcionalidades que una aplicación (o módulo de una aplicación) tienen que realizar. Se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. A través del controlador se realizan las consultas al modelo (una búsqueda, por ejemplo), y una vez se hayan obtenido dichos datos, se envía a la vista las instrucciones necesarias para poder mostrarlos de una forma legible para el usuario.

VENTAJAS:

Una de las razones por la que usamos esta arquitectura MVC, es porque tiene como objetivo optimizar la escalabilidad, esta técnica de programación es una excelente práctica que permite a los programadores dar una estructura eficaz a diferentes sistemas.

- La separación del Modelo y la Vista, lo cual logra separar los datos, de su representación visual.
- Facilita el manejo de errores.
- Permite que el sistema sea escalable si es requerido.
- Es posible agregar múltiples representaciones de los datos.
- La separación clara del lugar al que debe ir cada tipo de lógica.
- La simplicidad del mantenimiento y la optimización de la escalabilidad de la aplicación.
- La simplicidad de crear diferentes representaciones de los mismos datos.
- La facilidad de realizar pruebas unitarias de los componentes.
- La reutilización de los componentes.
- La adaptabilidad a tecnologías y cambios sin afectar necesariamente al otro componente.

Mejora en la testabilidad, que facilita la creación de pruebas unitarias, que podemos evaluar la lógica de negocio por separado del aspecto visual, por lo que lleva pruebas más efectivas.