**MANUAL DE TÉCNICO**

**SOFTWARE PARA UN SISTEMA DE CONTROL** **DE PERNOS Y TUERCAS.**

**NAVARRETE LUIS, SANCHEZ STEVEN, CUMBICOS LEYDER, REYES DAYSI, URDANIGO DAYANA, ZHUNIO GIA.**

**UNIVERSIDAD DE GUAYAQUIL**

**FACULTAD DE CIENCIAS MATEMATICS Y FISICAS**

**INGENIERIA EN SOFTWARE**

**ECUADOR – GYE, ENERO 2024**

**OBJETIVOS**

Identificar y destacar las implicaciones prácticas de las decisiones de arquitectura en términos de rendimiento, escalabilidad y mantenimiento.

Específicos

* Describir las decisiones de arquitectura más relevantes tomadas durante el proyecto, proporcionando justificaciones claras y detalladas para cada elección.
* Documentar las vistas necesarias de arquitectura, asegurando que abarquen de manera completa y precisa los diferentes aspectos del sistema.
* Garantizar la claridad y accesibilidad del documento para que pueda ser utilizado como referencia por miembros del equipo de desarrollo, stakeholders y otros interesados.

**INTRODUCCIÓN**

Este manual describe los pasos necesarios para que una persona con una base de sistema específica instale una aplicación diseñada para ejecutarse en el sistema.

Es importante recordar que este documento menciona los requisitos mínimos de hardware y software para la correcta instalación de la aplicación.

**REQUERIMIENTOS TÉCNICOS**

**REQUERIMIENTOS MÍNIMOS DE HARDWARE**

* Procesador: Core i5
* Memoria RAM: 8 GB o superior
* Disco Duro: 64 o 256 GB

**REQUERIMIENTOS MÍNIMOS DE SOFTWARE**

* Privilegios de administrador
* Sistema Operativo: Windows 10
* Base de datos: Microsoft SQL Server 2016 o superior
* JDK: Versión 8
* Visual studio code: IDE 2022

**HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO**

1. **Microsoft SQL Server 2016**

Es una potente plataforma de gestión de bases de datos diseñada para abordar las complejas necesidades de almacenamiento, administración y recuperación de datos en entornos empresariales.

Ofrece capacidades de procesamiento analítico avanzado con su motor de análisis, permitiendo la implementación de modelos predictivos y la realización de consultas complejas.

Además, se destaca por su enfoque en la seguridad, con características como el cifrado de datos dinámico y la auditoría avanzada.

1. **VISUAL STUDIO CODE**

Es un entorno de desarrollo muy popular y eficiente desarrollado por Microsoft. Destaca por su diseño liviano, flexibilidad y amplia compatibilidad con muchos lenguajes de programación.

VS Code ofrece una interfaz de usuario intuitiva y personalizable, que nos permitió a nosotros los desarrolladores personalizar nuestro entorno de trabajo. Además, es un editor de código fuente tiene potentes funciones como resaltado de sintaxis, autocompletado inteligente, integración con sistemas de control de versiones y varias extensiones para ampliar su funcionalidad.

1. **GITHUT**

Es la plataforma líder de desarrollo de software que facilita la gestión de proyectos y la colaboración entre equipos de desarrolladores.

Se utilizo Git como repositorio de control de versiones basado en la nube, lo que nos permitió realizar un seguimiento de los cambios en el código fuente, coordinar tareas y colaborar.

**INSTALACION**

Descargar las aplicaciones de Microsoft SQL Server 2016 y Visual Studio Code para la ejecución.

Una vez que hayas descargado el instalador de Microsoft SQL Server 2016, ejecútalo. Se abrirá un asistente de instalación que guiará a través del proceso. La instalación básica instalará la configuración predeterminada.

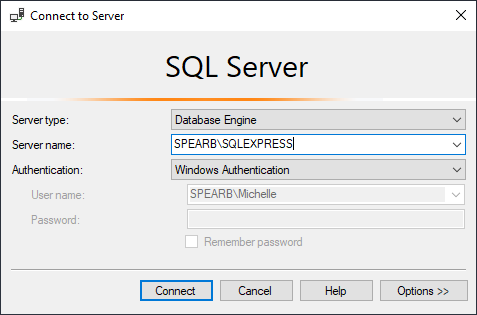
Revisa la configuración que has seleccionado y haz clic en "Instalar" para comenzar el proceso de instalación.

Por último, descargamos el sitio web oficial de Visual Studio Code: Visual Studio Code. Hacemos clic en el botón "Download" para descargar el instalador.

Ejecutar el archivo descargado (generalmente llamado VSCodeSetup.exe).

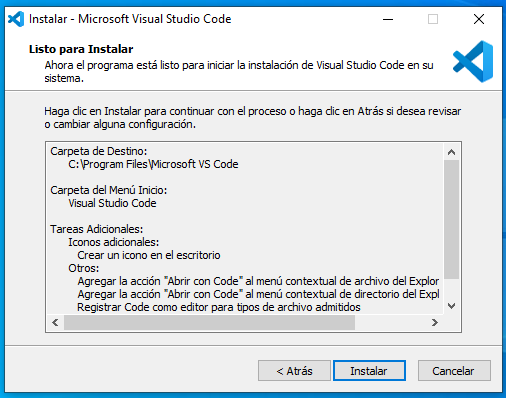
Sigue las instrucciones del instalador. Puedes dejar las opciones predeterminadas o personalizarlas según tus preferencias.

Una vez completada la instalación, ejecuta Visual Studio Code desde el menú de inicio.



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



**INGRESO DE DATOS A MICROSOFT SQL SERVER**

1. **TABLAS**

-----------------------------Tablas-----------------------------

-- Tabla para almacenar información de usuarios

CREATE TABLE [dbo].[Administrador] (

[id\_administrador] INT IDENTITY (1, 1) NOT NULL,

[usuario\_id] INT NOT NULL,

[nombre\_administrador] VARCHAR (255) NOT NULL,

[correo\_electronico] VARCHAR (100) NOT NULL,

[telefono] VARCHAR (20) NULL,

[id\_inventario] VARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([id\_administrador] ASC),

UNIQUE NONCLUSTERED ([usuario\_id] ASC),

FOREIGN KEY ([usuario\_id]) REFERENCES [dbo].[Usuario] ([id\_usuario])

);

-- Tabla para almacenar información de administradores

CREATE TABLE [dbo].[CatalogoProducto] (

[id\_producto] INT IDENTITY (1, 1) NOT NULL,

[foto\_producto] VARBINARY (MAX) NULL,

[nombre\_producto] VARCHAR (255) NOT NULL,

[precio\_producto] DECIMAL (10, 2) NOT NULL,

[id\_proveedor] INT NOT NULL,

[id\_modelo] INT NOT NULL,

[id\_marca] INT NOT NULL,

[id\_tipo\_producto] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id\_producto] ASC),

FOREIGN KEY ([id\_modelo]) REFERENCES [dbo].[Modelo] ([id\_modelo]),

FOREIGN KEY ([id\_tipo\_producto]) REFERENCES [dbo].[TipoProducto] ([id\_tipo\_producto]),

FOREIGN KEY ([id\_marca]) REFERENCES [dbo].[Marca] ([id\_marca]),

FOREIGN KEY ([id\_proveedor]) REFERENCES [dbo].[Proveedor] ([id\_Proveedor])

);

-- Tabla para almacenar información de clientes

CREATE TABLE [dbo].[Cliente] (

[id\_cliente] INT IDENTITY (1, 1) NOT NULL,

[usuario\_id] INT NOT NULL,

[direccion] VARCHAR (255) NOT NULL,

[correo\_electronico] VARCHAR (100) NOT NULL,

[telefono] VARCHAR (10) NOT NULL,

PRIMARY KEY CLUSTERED ([id\_cliente] ASC),

UNIQUE NONCLUSTERED ([usuario\_id] ASC),

FOREIGN KEY ([usuario\_id]) REFERENCES [dbo].[Usuario] ([id\_usuario])

);

-- Tabla para almacenar información de compra

CREATE TABLE [dbo].[Compra] (

[id\_compra] INT IDENTITY (1, 1) NOT NULL,

[fecha\_compra] DATE NOT NULL,

[total\_compra] DECIMAL (10, 2) NOT NULL,

[id\_proveedor] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id\_compra] ASC),

FOREIGN KEY ([id\_proveedor]) REFERENCES [dbo].[Proveedor] ([id\_Proveedor])

);

-- Tabla para detalles de compras

CREATE TABLE [dbo].[DetalleCompra] (

[id\_detalle\_compra] INT IDENTITY (1, 1) NOT NULL,

[cantidad] INT NOT NULL,

[precio\_unitario] DECIMAL (10, 2) NOT NULL,

[id\_compra] INT NULL,

[id\_producto] INT NULL,

PRIMARY KEY CLUSTERED ([id\_detalle\_compra] ASC),

FOREIGN KEY ([id\_compra]) REFERENCES [dbo].[Compra] ([id\_compra]),

FOREIGN KEY ([id\_producto]) REFERENCES [dbo].[CatalogoProducto] ([id\_producto])

);

-- Tabla para detalles de facturas

CREATE TABLE [dbo].[DetalleFactura] (

[id\_detalle\_factura] INT IDENTITY (1, 1) NOT NULL,

[cantidad] INT NOT NULL,

[precio\_unitario] DECIMAL (10, 2) NOT NULL,

[id\_factura\_venta] INT NOT NULL,

[id\_producto] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id\_detalle\_factura] ASC),

FOREIGN KEY ([id\_factura\_venta]) REFERENCES [dbo].[FacturaVenta] ([id\_factura\_venta]),

FOREIGN KEY ([id\_producto]) REFERENCES [dbo].[CatalogoProducto] ([id\_producto])

);

-- Tabla para ventas

CREATE TABLE [dbo].[FacturaVenta] (

[id\_factura\_venta] INT IDENTITY (1, 1) NOT NULL,

[fecha\_venta] DATE NOT NULL,

[total\_venta] DECIMAL (10, 2) NOT NULL,

[id\_cliente] INT NOT NULL,

[id\_forma\_pago] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id\_factura\_venta] ASC),

FOREIGN KEY ([id\_cliente]) REFERENCES [dbo].[Cliente] ([id\_cliente]),

FOREIGN KEY ([id\_forma\_pago]) REFERENCES [dbo].[FormaPago] ([id\_forma\_pago])

);

-- Tabla para métodos de pago aceptados

CREATE TABLE [dbo].[FormaPago] (

[id\_forma\_pago] INT IDENTITY (1, 1) NOT NULL,

[nombre\_forma\_pago] VARCHAR (255) NOT NULL,

PRIMARY KEY CLUSTERED ([id\_forma\_pago] ASC),

CHECK ([nombre\_forma\_pago]='Transferencia' OR [nombre\_forma\_pago]='PayPal' OR [nombre\_forma\_pago]='Tarjeta\_de\_credito')

);

-- Tabla para el inventario de productos

CREATE TABLE [dbo].[Inventario] (

[id\_inventario] INT IDENTITY (1, 1) NOT NULL,

[cantidad] INT NOT NULL,

[id\_producto] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id\_inventario] ASC),

FOREIGN KEY ([id\_producto]) REFERENCES [dbo].[CatalogoProducto] ([id\_producto])

);

-- Tabla para información de marcas de productos

CREATE TABLE [dbo].[Marca] (

[id\_marca] INT IDENTITY (1, 1) NOT NULL,

[foto\_marca] VARBINARY (MAX) NULL,

[nombre\_marca] VARCHAR (255) NOT NULL,

[pais\_origen] VARCHAR (255) NULL,

PRIMARY KEY CLUSTERED ([id\_marca] ASC)

);

-- Tabla para información de modelos de productos

CREATE TABLE [dbo].[Modelo] (

[id\_modelo] INT IDENTITY (1, 1) NOT NULL,

[nombre\_modelo] VARCHAR (255) NOT NULL,

[descripcion] VARCHAR (255) NULL,

PRIMARY KEY CLUSTERED ([id\_modelo] ASC)

);

-- Tabla para información de proveedores

CREATE TABLE [dbo].[Proveedor] (

[id\_Proveedor] INT IDENTITY (1, 1) NOT NULL,

[usuario\_id] INT NOT NULL,

[nombre\_empresa] VARCHAR (50) NOT NULL,

[contacto\_nombre] VARCHAR (30) NOT NULL,

[contacto\_telefono] VARCHAR (15) NOT NULL,

[contacto\_correo] VARCHAR (50) NOT NULL,

[direccion] VARCHAR (100) NOT NULL,

[ciudad] VARCHAR (30) NOT NULL,

[provincia] VARCHAR (30) NOT NULL,

[codigo\_postal] VARCHAR (10) NULL,

[pais] VARCHAR (30) NOT NULL,

PRIMARY KEY CLUSTERED ([id\_Proveedor] ASC),

UNIQUE NONCLUSTERED ([usuario\_id] ASC),

FOREIGN KEY ([usuario\_id]) REFERENCES [dbo].[Usuario] ([id\_usuario])

);

-- Tabla para tipos específicos de productos

CREATE TABLE [dbo].[TipoProducto] (

[id\_tipo\_producto] INT IDENTITY (1, 1) NOT NULL,

[nombre\_tipo\_producto] VARCHAR (255) NOT NULL,

PRIMARY KEY CLUSTERED ([id\_tipo\_producto] ASC)

);

-- Tabla para tipos específicos de Usuario

CREATE TABLE [dbo].[Usuario] (

[id\_usuario] INT IDENTITY (1, 1) NOT NULL,

[nombre\_usuario] VARCHAR (30) NOT NULL,

[contrasena] VARCHAR (50) NOT NULL,

[tipo] VARCHAR (15) NOT NULL,

PRIMARY KEY CLUSTERED ([id\_usuario] ASC),

CHECK ([tipo]='proveedor' OR [tipo]='admin' OR [tipo]='cliente')

);

1. **PROCESOS ALMCENADOS**

-----------------------------Procedimientos almacenados-----------------------------

CREATE PROCEDURE sp\_CrearCliente

@nombre\_usuario VARCHAR(30),

@contrasena VARCHAR(50),

@direccion VARCHAR(255),

@correo\_electronico VARCHAR(100),

@telefono VARCHAR(10)

AS

BEGIN

BEGIN TRY

DECLARE @usuario\_id INT;

-- Crear el usuario con tipo "cliente"

INSERT INTO [dbo].[Usuario] ([nombre\_usuario], [contrasena], [tipo])

VALUES (@nombre\_usuario, @contrasena, 'cliente');

-- Obtener el ID del usuario recién creado

SET @usuario\_id = SCOPE\_IDENTITY();

-- Crear el cliente asociado al usuario

INSERT INTO [dbo].[Cliente] ([usuario\_id], [direccion], [correo\_electronico], [telefono])

VALUES (@usuario\_id, @direccion, @correo\_electronico, @telefono);

END TRY

BEGIN CATCH

-- Mostrar mensaje de error

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH

END;

CREATE PROCEDURE sp\_CrearProveedor

@nombre\_usuario VARCHAR(30),

@contrasena VARCHAR(50),

@nombre\_empresa VARCHAR(50),

@contacto\_nombre VARCHAR(30),

@contacto\_telefono VARCHAR(15),

@contacto\_correo VARCHAR(50),

@direccion VARCHAR(100),

@ciudad VARCHAR(30),

@provincia VARCHAR(30),

@codigo\_postal VARCHAR(10),

@pais VARCHAR(30)

AS

BEGIN

BEGIN TRY

DECLARE @usuario\_id INT;

-- Crear el usuario con tipo "proveedor"

INSERT INTO [dbo].[Usuario] ([nombre\_usuario], [contrasena], [tipo])

VALUES (@nombre\_usuario, @contrasena, 'proveedor');

-- Obtener el ID del usuario recién creado

SET @usuario\_id = SCOPE\_IDENTITY();

-- Crear el proveedor asociado al usuario

INSERT INTO [dbo].[Proveedor] ([usuario\_id], [nombre\_empresa], [contacto\_nombre], [contacto\_telefono],

[contacto\_correo], [direccion], [ciudad], [provincia], [codigo\_postal], [pais])

VALUES (@usuario\_id, @nombre\_empresa, @contacto\_nombre, @contacto\_telefono,

@contacto\_correo, @direccion, @ciudad, @provincia, @codigo\_postal, @pais);

END TRY

BEGIN CATCH

-- Mostrar mensaje de error

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH

END;

CREATE PROCEDURE sp\_InsertarMarca

@foto\_marca VARBINARY(MAX),

@nombre\_marca VARCHAR(255),

@pais\_origen VARCHAR(255)

AS

BEGIN

INSERT INTO dbo.Marca (foto\_marca, nombre\_marca, pais\_origen)

VALUES (@foto\_marca, @nombre\_marca, @pais\_origen);

END;

CREATE PROCEDURE sp\_InsertarModelo

@nombre\_modelo VARCHAR(255),

@descripcion VARCHAR(255)

AS

BEGIN

INSERT INTO dbo.Modelo (nombre\_modelo, descripcion)

VALUES (@nombre\_modelo, @descripcion);

END;

CREATE PROCEDURE sp\_InsertarProducto

@foto\_producto VARBINARY(MAX),

@nombre\_producto VARCHAR(255),

@precio\_producto DECIMAL(10, 2),

@id\_proveedor INT,

@id\_modelo INT,

@id\_marca INT,

@id\_tipo\_producto INT

AS

BEGIN

BEGIN TRY

-- Insertar el nuevo producto en la tabla CatalogoProducto

INSERT INTO [dbo].[CatalogoProducto] (

[foto\_producto], [nombre\_producto], [precio\_producto],

[id\_proveedor], [id\_modelo], [id\_marca], [id\_tipo\_producto]

)

VALUES (

@foto\_producto, @nombre\_producto, @precio\_producto,

@id\_proveedor, @id\_modelo, @id\_marca, @id\_tipo\_producto

);

END TRY

BEGIN CATCH

-- Registra el error en una tabla de registro de errores

INSERT INTO [dbo].[ErrorLog] (ErrorMessage, ErrorProcedure)

VALUES (ERROR\_MESSAGE(), 'sp\_InsertarProducto');

-- Lanza el error para que la aplicación lo capture (si es necesario)

THROW;

END CATCH

END;

CREATE PROCEDURE sp\_InsertarTipoProducto

@nombre\_tipo\_producto VARCHAR(255)

AS

BEGIN

INSERT INTO dbo.TipoProducto (nombre\_tipo\_producto)

VALUES (@nombre\_tipo\_producto);

END;

CREATE PROCEDURE sp\_ObtenerTipoUsuario

@nombre\_usuario VARCHAR(30)

AS

BEGIN

BEGIN TRY

SELECT [tipo] FROM [dbo].[Usuario] WHERE [nombre\_usuario] = @nombre\_usuario;

END TRY

BEGIN CATCH

-- Manejar la excepción si es necesario

SELECT NULL AS [tipo]; -- Otra opción: RETURN NULL;

END CATCH

END;

CREATE PROCEDURE sp\_VerContraseñaCliente

@correo\_electronico VARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @contrasena\_cliente VARCHAR(50);

-- Buscar la contraseña del cliente por correo electrónico

SELECT @contrasena\_cliente = [contrasena]

FROM [dbo].[Usuario] AS U

INNER JOIN [dbo].[Cliente] AS C ON U.[id\_usuario] = C.[usuario\_id]

WHERE C.[correo\_electronico] = @correo\_electronico;

-- Mostrar la contraseña si se encontró el cliente

IF @contrasena\_cliente IS NOT NULL

BEGIN

SELECT @contrasena\_cliente AS 'ContrasenaCliente';

END

ELSE

BEGIN

SELECT 'Correo electrónico no correspondiente a un cliente registrado' AS 'Mensaje';

END;

END;

CREATE PROCEDURE sp\_VerficarLog

@nombre\_usuario VARCHAR(15),

@contrasena VARCHAR(50)

AS

SELECT COUNT(\*) FROM Usuario WHERE nombre\_usuario = @nombre\_usuario AND contrasena = @contrasena

CREATE PROCEDURE sp\_Verificar\_correo

@nombre\_usuario VARCHAR(30)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @CorreoElectronico VARCHAR(100);

-- Buscar el correo electrónico en la tabla Cliente

SELECT @CorreoElectronico = [correo\_electronico]

FROM [dbo].[Cliente]

WHERE [usuario\_id] = (SELECT [id\_usuario] FROM [dbo].[Usuario] WHERE [nombre\_usuario] = @nombre\_usuario)

IF @CorreoElectronico IS NOT NULL

BEGIN

SELECT 1 AS 'Exists'; -- Devuelve 1 si se encuentra el correo en Cliente

RETURN;

END

-- Buscar el correo electrónico en la tabla Proveedor

SELECT @CorreoElectronico = [contacto\_correo]

FROM [dbo].[Proveedor]

WHERE [usuario\_id] = (SELECT [id\_usuario] FROM [dbo].[Usuario] WHERE [nombre\_usuario] = @nombre\_usuario)

IF @CorreoElectronico IS NOT NULL

BEGIN

SELECT 1 AS 'Exists'; -- Devuelve 1 si se encuentra el correo en Proveedor

RETURN;

END

-- Buscar el correo electrónico en la tabla Administrador

SELECT @CorreoElectronico = [correo\_electronico]

FROM [dbo].[Administrador]

WHERE [usuario\_id] = (SELECT [id\_usuario] FROM [dbo].[Usuario] WHERE [nombre\_usuario] = @nombre\_usuario)

IF @CorreoElectronico IS NOT NULL

BEGIN

SELECT 1 AS 'Exists'; -- Devuelve 1 si se encuentra el correo en Administrador

RETURN;

END

SELECT 0 AS 'Exists'; -- Devuelve 0 si no se encuentra en ninguna tabla

END;

CREATE PROCEDURE sp\_VerificarLog

@nombre\_usuario VARCHAR(30),

@contrasena VARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

-- Verificar si las credenciales coinciden en la tabla Usuario

SELECT COUNT(\*) AS 'Count'

FROM [dbo].[Usuario]

WHERE [nombre\_usuario] = @nombre\_usuario AND [contrasena] = @contrasena;

END;

CREATE PROCEDURE sp\_VerificarNombreUsuarioDuplicado

@nombre\_usuario VARCHAR(30)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @UsuarioExistente INT;

-- Verificar si el nombre de usuario ya existe

SELECT @UsuarioExistente = COUNT(\*)

FROM [dbo].[Usuario]

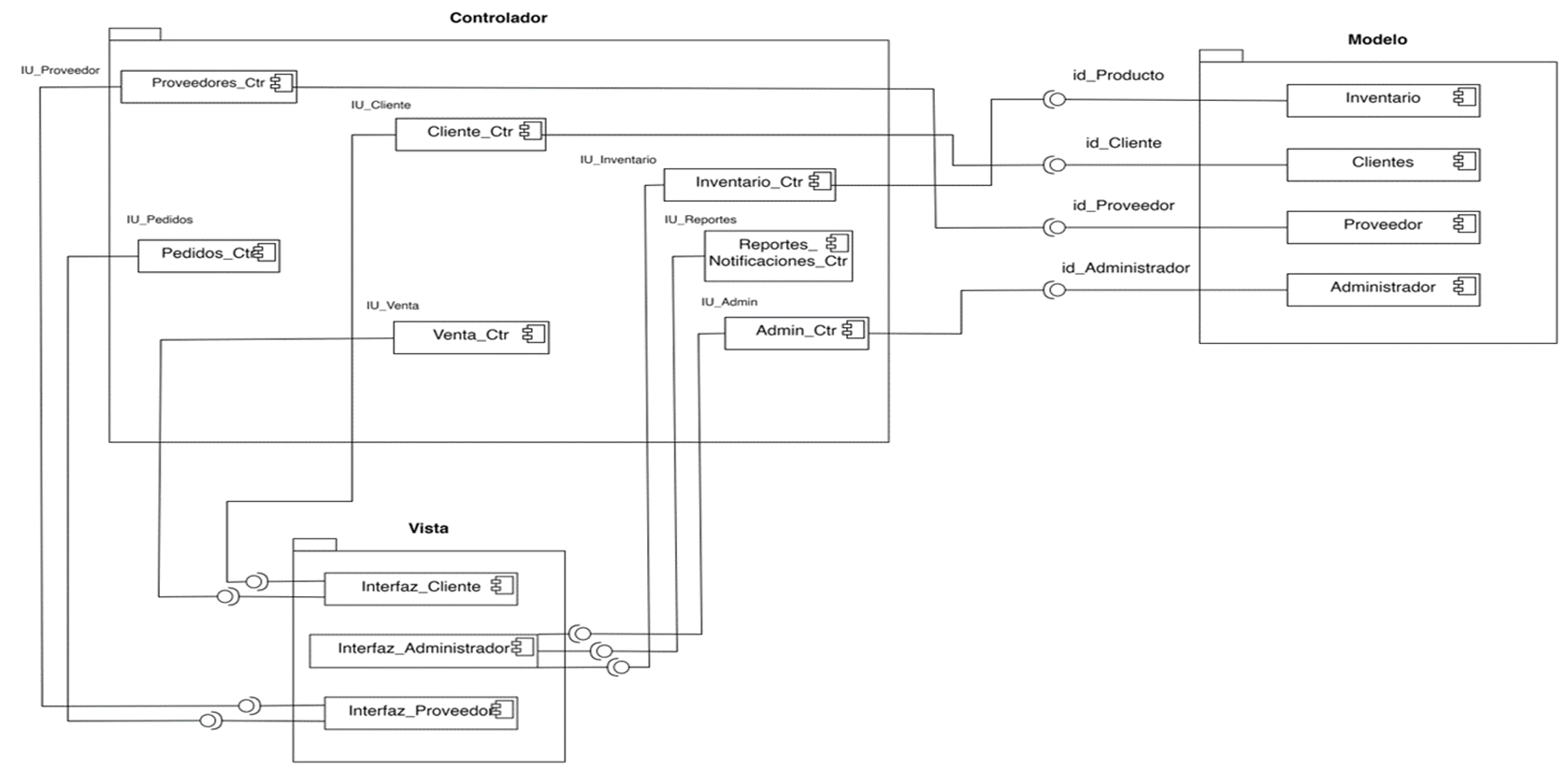
WHERE [nombre\_usuario] = @nombre\_usuario;

-- Devolver 1 si el nombre de usuario ya existe, 0 si no

SELECT CASE WHEN @UsuarioExistente > 0 THEN 1 ELSE 0 END AS 'UsuarioDuplicado';

END;

**COMPONENTES DEL APLICATIVO**



**PROTOTIPOS DE PANTALLAS DEL APLICATIVO**

A continuación, mostramos los prototipos planteados:



Figura Prototipo de pantalla Registrar Cuenta

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

Figura Prototipo Buscar Proveedor

Interfaz de usuario gráfica, PowerPoint

Descripción generada automáticamente con confianza media

Figura Prototipo Buscar Producto



Figura Modificar Proveedor