


- ARRIBASPLATA CHAVARRI, FABIO JOAQUIN
- AGUIRRE RODRIGUEZ, SHEYLA MARITA

# MODELAMIENTO DE NEGOCIO

# **TIENDAS MASS**

UNIVERSIDAD PRIVADA DEL  
NORTE

- 
- 01 EMPRESA Y PROBLEMÁTICA**
  - 02 OBJETIVOS DEL PROYECTO**
  - 03 PROCESO DE NEGOCIO**
  - 04 W. Y ACTORES DEL NEGOCIO**
  - 05 DIAGRAMA DE CASO DE USO DE NEGOCIO**
  - 06 DIAGRAMA DE ACTIVIDADES DEL NEGOCIO**
  - 07 MODELO DE DOMINIO**
  - 08 NECESIDADES Y CARACTERÍSTICAS**
  - 09 DIAGRAMA DE CASOS DE USO**
  - 10 DIAGRAMA LÓGICO**
  - 11 SOFTWARE Y BASE DE DATOS**

# EMPRESA MASS

1

## ¿QUÉ ES Y DE DONDE ES?

Mass es una cadena de establecimientos de descuento perteneciente al grupo peruano Intercorp, el cual incluye otras reconocidas marcas como Plaza Veja, Vivanda y Makro.

2

## ENFOQUE

Su enfoque se centra en ofrecer a los clientes la posibilidad de realizar compras rápidas y puntuales a precios bajos, con un catálogo limitado de productos.

3

## ESTRATEGIA

Su estrategia de negocio la sitúa en competencia directa con bodegas y mercados locales, destacando por su conveniencia y asequibilidad.



## PROBLEMÁTICA

La tienda MASS enfrenta un desafío logístico significativo relacionado con el exceso de inventario y la falta de espacio de almacenamiento adecuado, lo que impacta negativamente en la eficiencia operativa y la satisfacción del cliente.

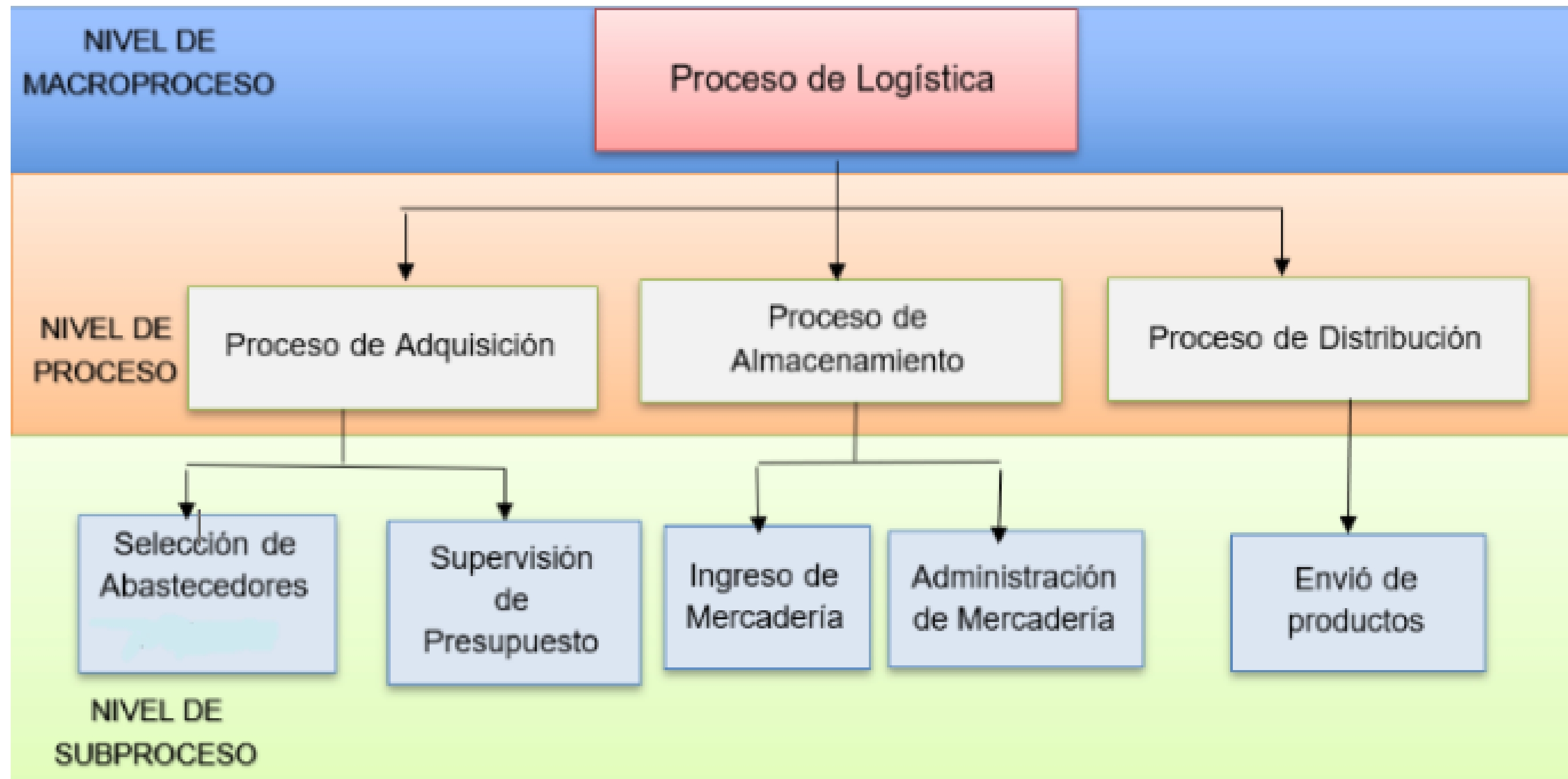
Esta problemática se manifiesta en demoras en la reposición de estantes y una atención menos eficiente. La falta de un sistema de gestión de inventario eficiente y la omisión de ciertos factores contribuyen a esta situación.

# OBJETIVOS DEL PROYECTO

- **Optimizar** la gestión de compras de productos
- **Implementar** un sistema de control de calidad.
- **Mejorar** la eficiencia de compra.
- **Reducir** el espacio de almacenamiento.
- **Optimizar** la rotación de inventario.
- **Mejorar** la calidad de los productos.
- **Asegurar** la presencia de productos.
- **Optimizar** la cadena de suministro



# PROCESOS DEL NEGOCIO



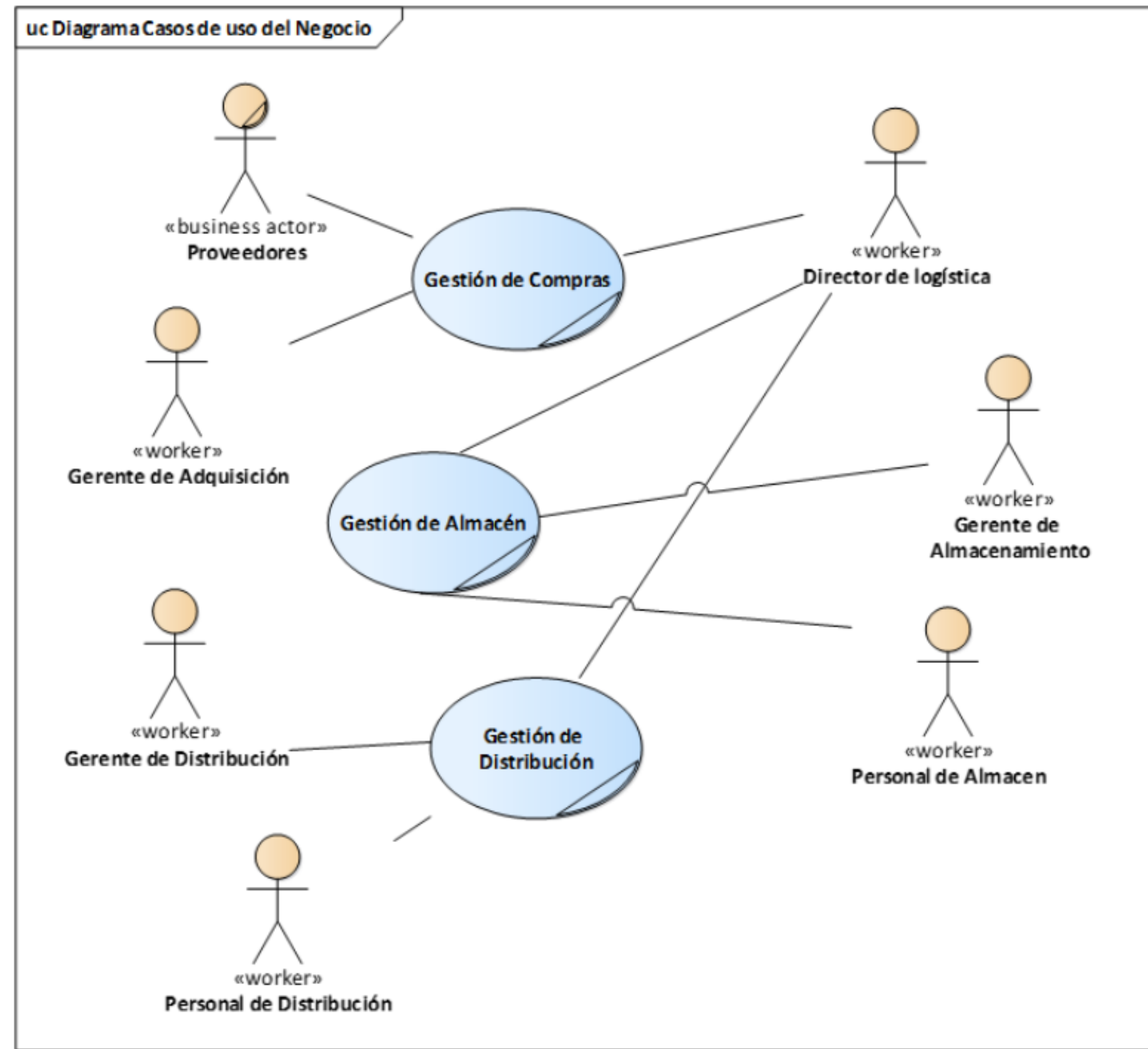
## TRABAJADORES



## ACTORES



# DIAGRAMA DE CASOS DE USO DEL NEGOCIO

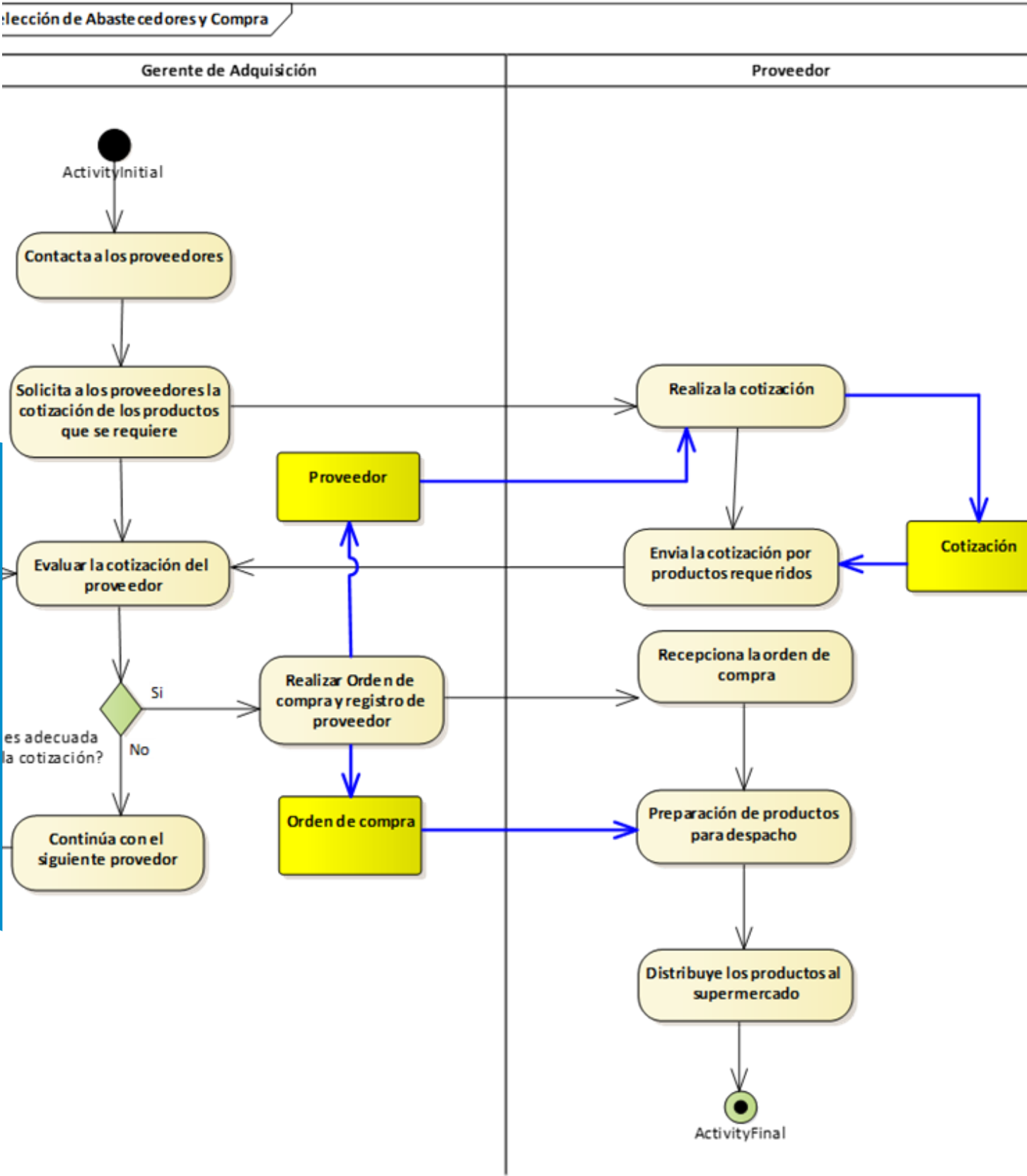




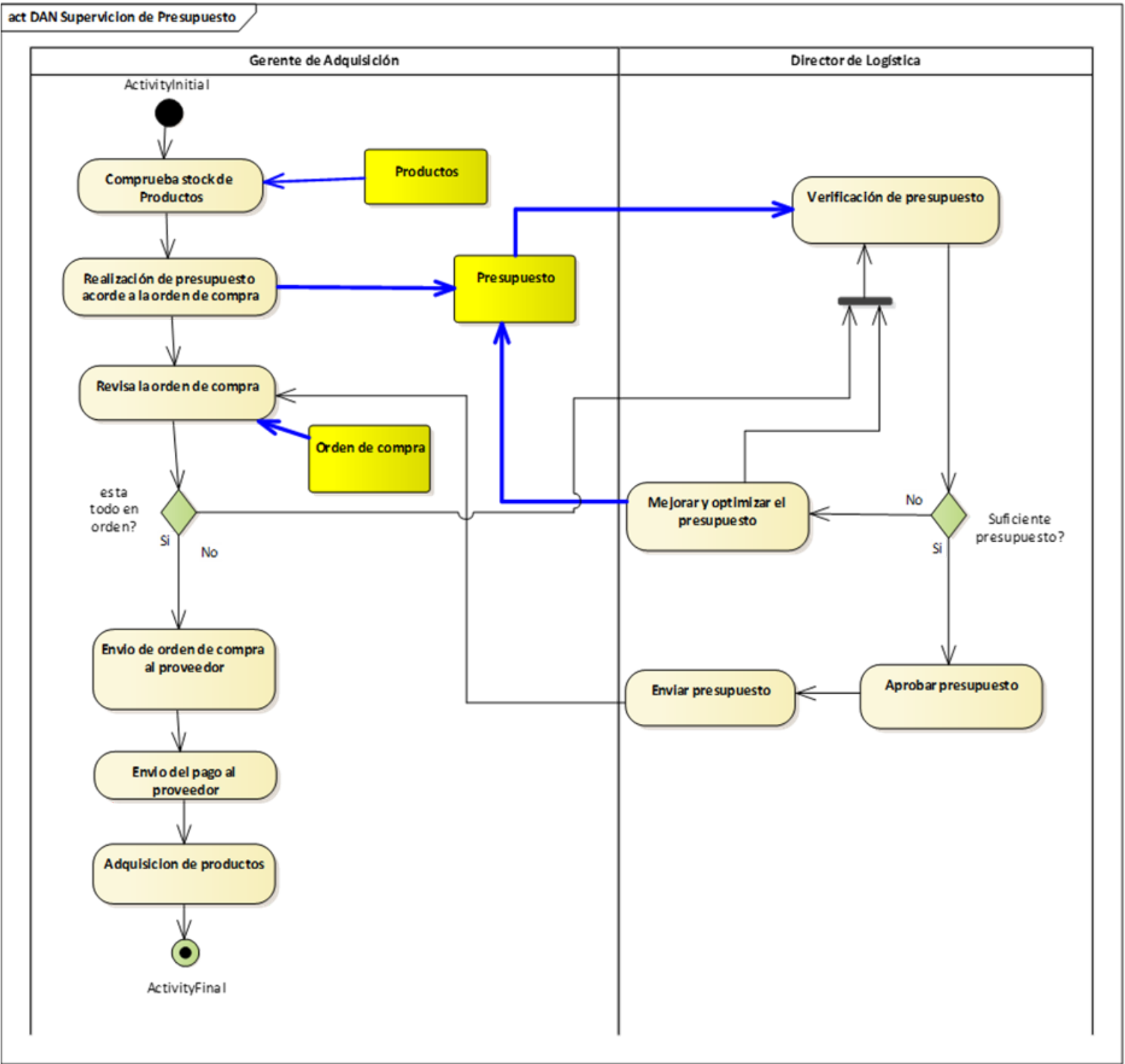
# DIAGRAMAS DE ACTIVIDAD DEL NEGOCIO

- 01 SELECCION DE ABASTECEDORES Y COMPRAS**
- 02 SUPERVISION DE PRESUPUESTO**
- 03 INGRESO DE MERCADERIA**
- 04 ADMINISTRACION DE MERCADERIA**
- 05 ENVIO DE MERCADERIA**

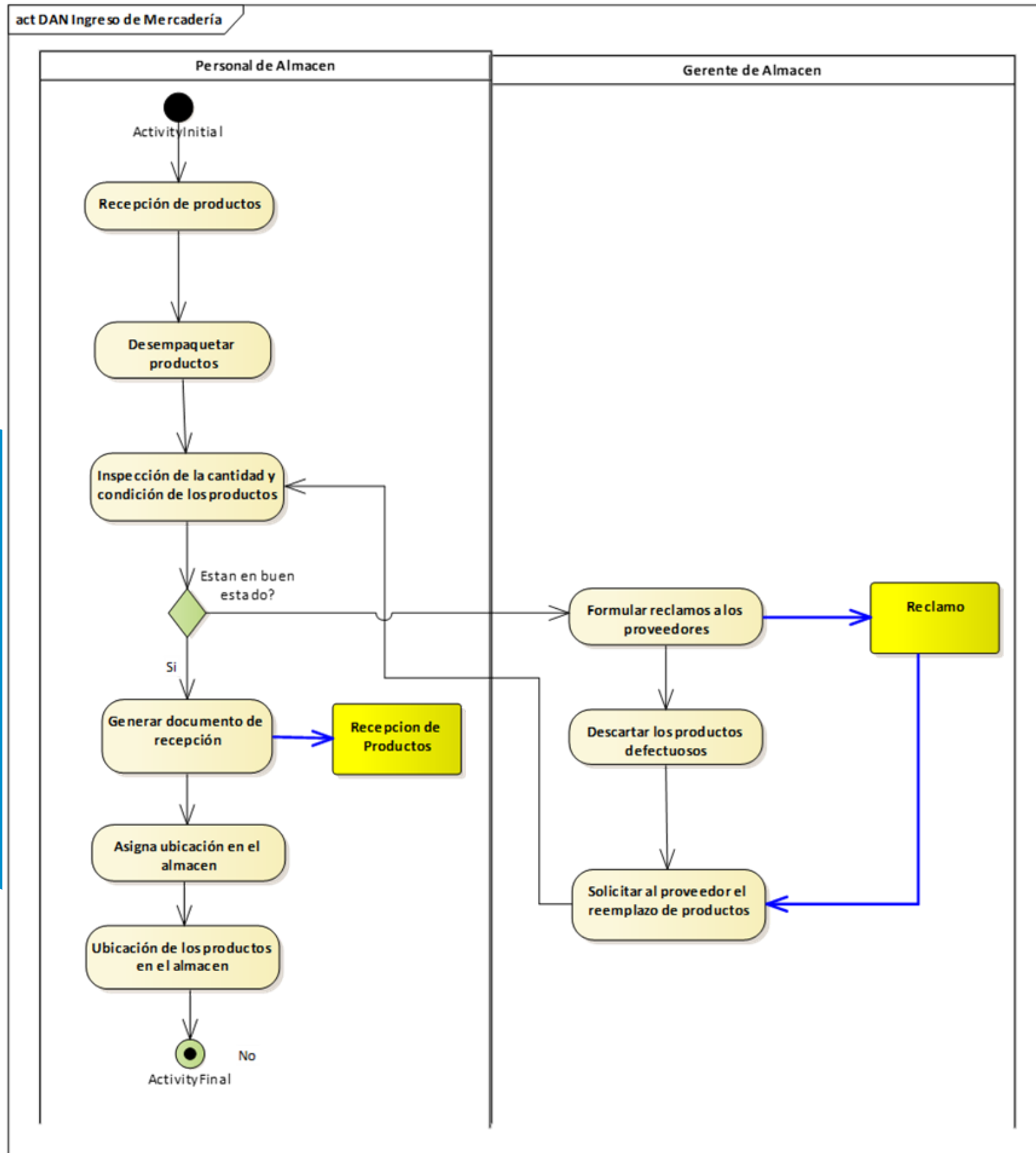
# SELECCION DE ABASTECEDORES Y COMPRAS



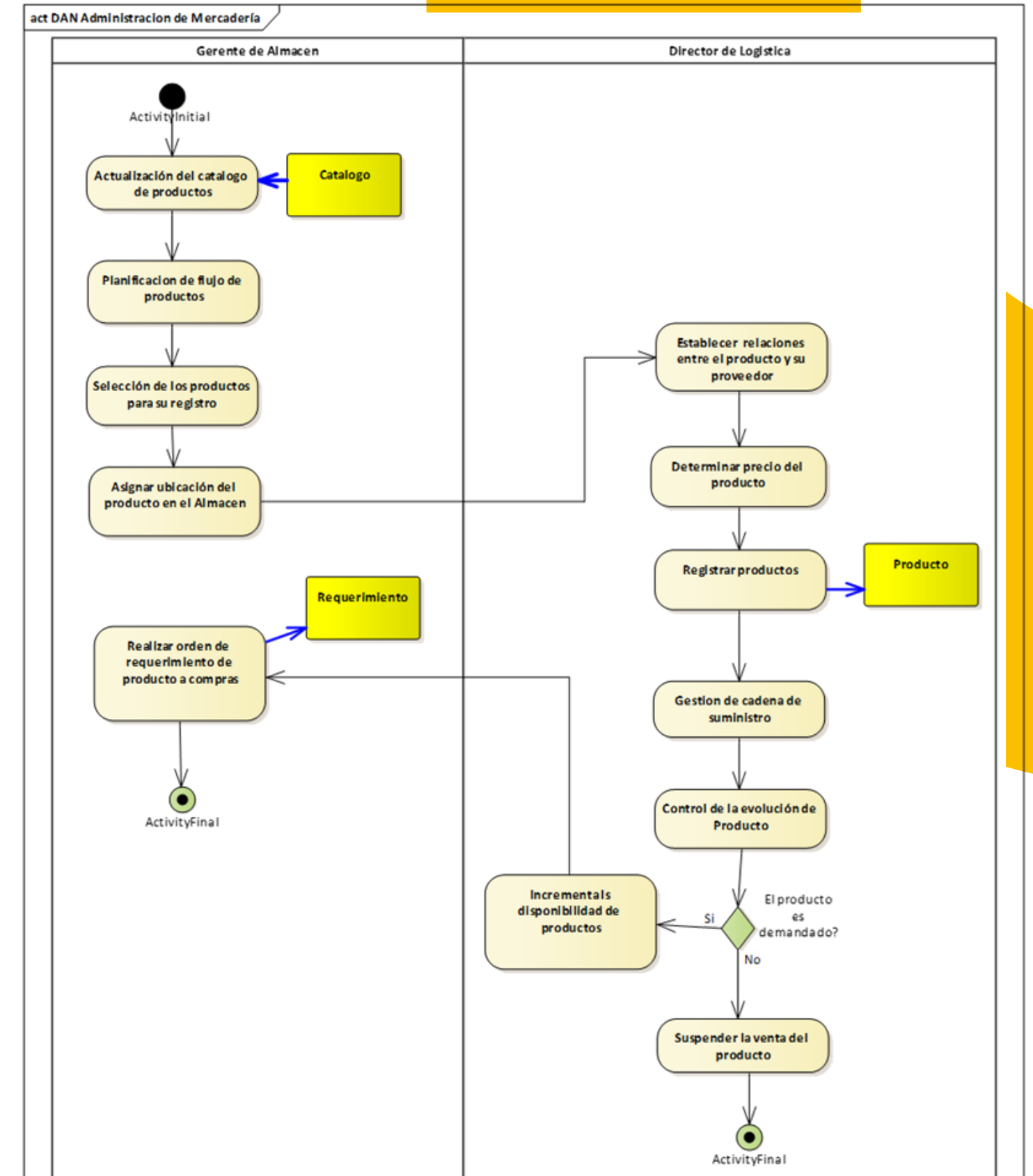
# SELECCION DE ABASTECEDORES Y COMPRAS



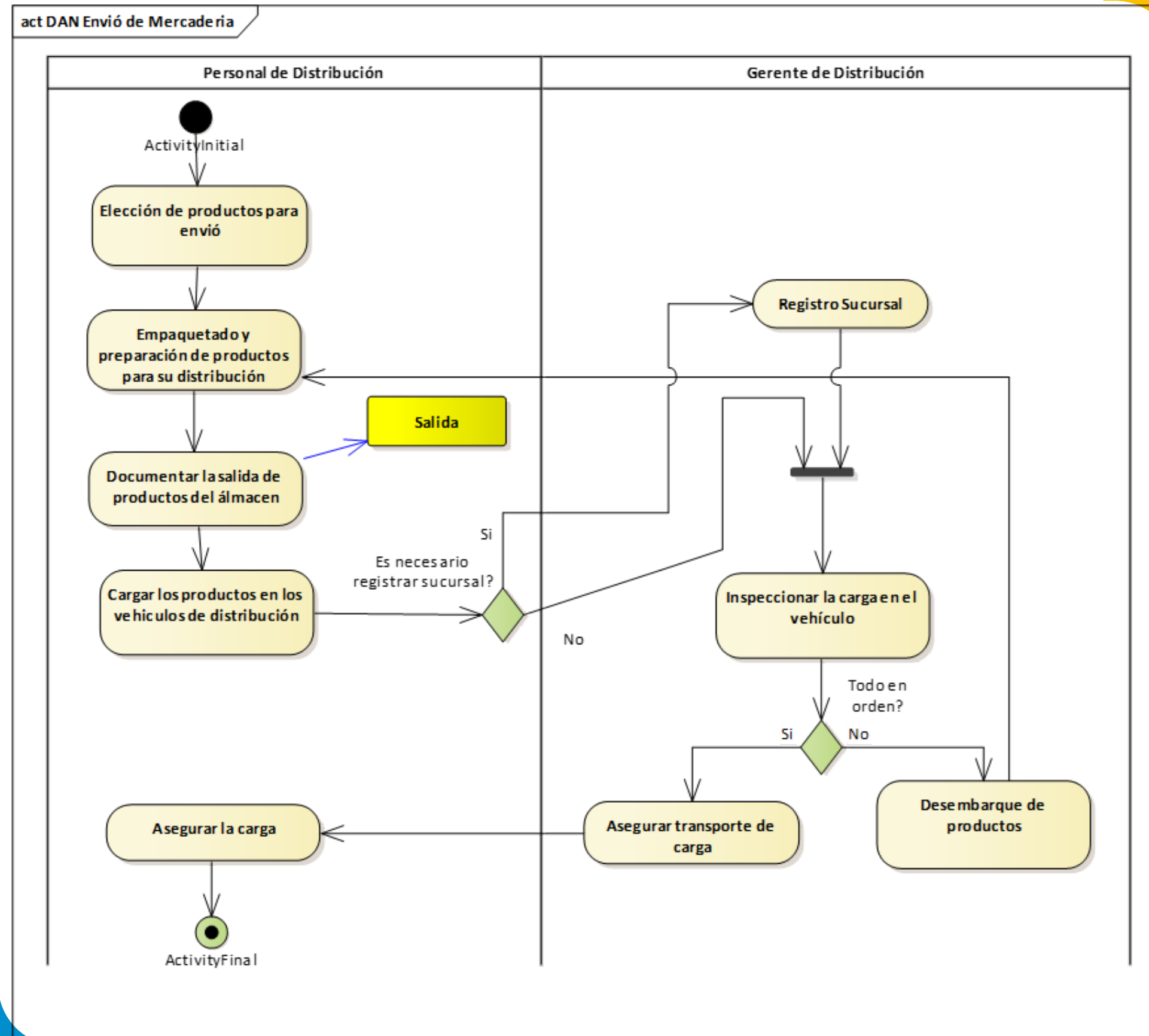
# INGRESO DE MERCADERIA



# ADMINISTRACION DE MERCADERIA



# ENVIO DE MERCADERIA



## NECESIDADES:

ID - NEC	DESCRIPCION	TIPO
01 - NEC	El sistema deberá permitir realizar ordenes de compra	CORE
2 - NEC	El sistema deberá permitir realizar salida	CORE
3 - NEC	El sistema debera permitir realizar requerimientos	CORE
4 - NEC	El sistema deberá permitir realizar presupuestos de productos	CORE
5 - NEC	El sistema deberá permitir realizar recepcion de productos	CORE
6 - NEC	El sistema deberá actualizar, crear, mostrar y eliminar un proveedor	CRUD
7 - NEC	El sistema deberá actualizar, crear, mostrar y eliminar productos	CRUD
8 - NEC	El sistema deberá permitir crear, mostrar, eliminar y actualizar un sucursal	CRUD
9 - NEC	El sistema deberá permitir consultar una orden de compra	CONSULTA
10 - NEC	El sistema deberá permitir consultar los productos	CONSULTA
11 - NEC	El sistema deberá permitir consultar los presupuestos	CONSULTA

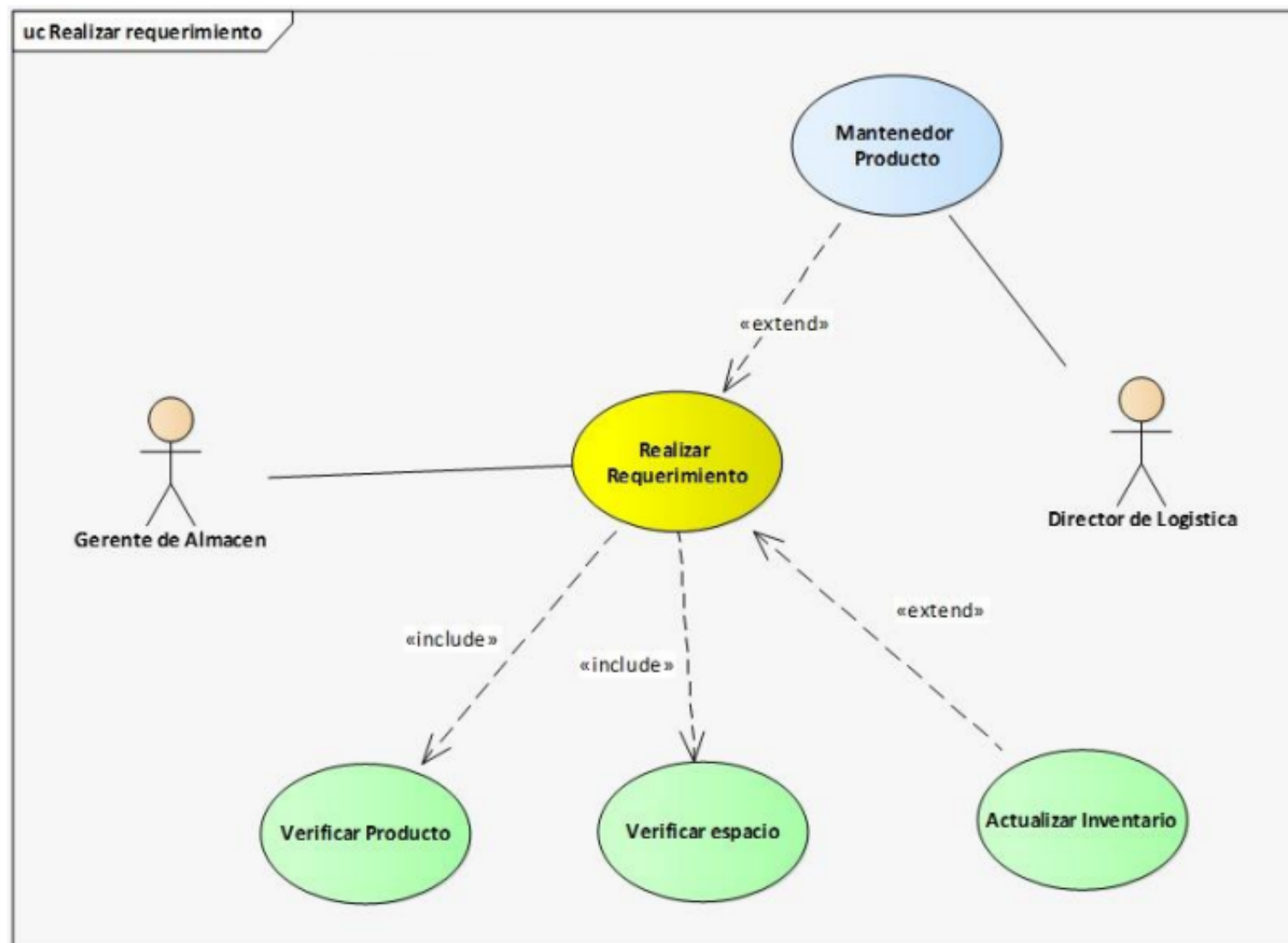
# CARACTERISTICAS:

CAR - ID	NEC - ID	DESCRIPCION
CAR - 1	NEC - 1	El sistema deberá permitir registrar ordenes de compra
CAR - 2	NEC - 2	El sistema deberá permitir registrar salidas
CAR - 3	NEC - 3	El sistema deberá permitir registrar requerimientos
CAR - 4	NEC - 4	El sistema deberá permitir registrar presupuestos de productos
CAR - 5	NEC - 5	El sistema deberá permitir registrar recepciones de productos
CAR - 6	NEC - 6	El sistema deberá permitir registrar un proveedor
CAR - 7	NEC - 6	El sistema deberá permitir eliminar un proveedor
CAR - 8	NEC - 6	El sistema deberá permitir mostrar un proveedor
CAR - 9	NEC - 6	El sistema deberá permitir actualizar un proveedor
CAR - 10	NEC - 7	El sistema deberá permitir registrar un producto

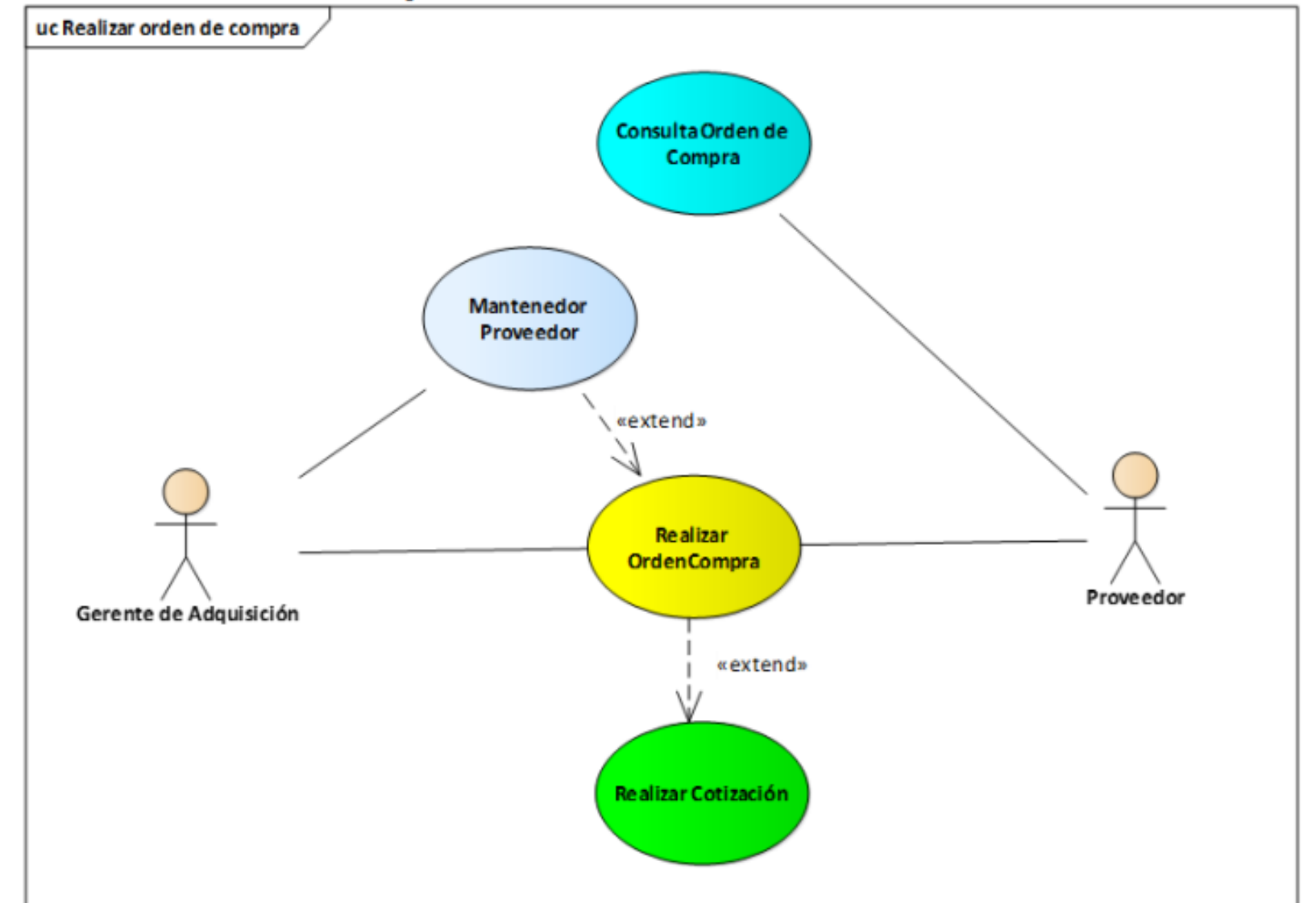
R - 11	NEC - 7	El sistema deberá permitir eliminar un producto
R - 12	NEC - 7	El sistema deberá permitir mostrar un producto
R - 13	NEC - 7	El sistema deberá permitir actualizar un producto
R - 14	NEC - 8	El sistema deberá permitir registrar una sucursal
R - 15	NEC - 8	El sistema deberá permitir eliminar un sucursal
R - 16	NEC - 8	El sistema deberá permitir mostrar un sucursal
R - 17	NEC - 8	El sistema deberá permitir actualizar un sucursal
R - 18	NEC - 9	El sistema debe permitir mostrar una orden de compra por su ID
R - 19	NEC - 10	El sistema debe permitir mostrar un producto por su ID
R - 20	NEC - 11	El sistema debe permitir mostrar un presupuesto por su ID

# DIAGRAMA DE CASOS DE USO

## • CUS Requerimiento

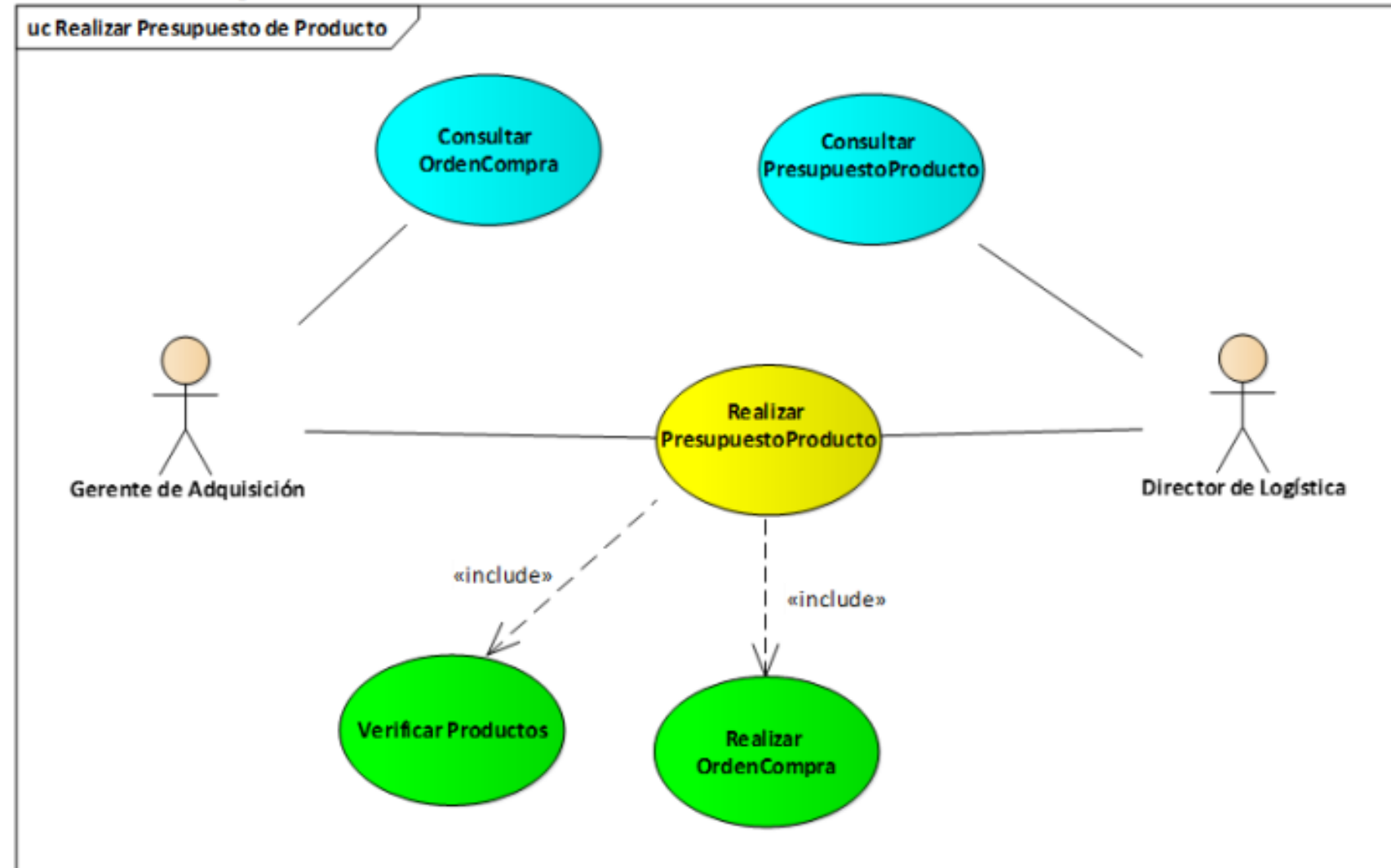


## • CUS Orden de Compra

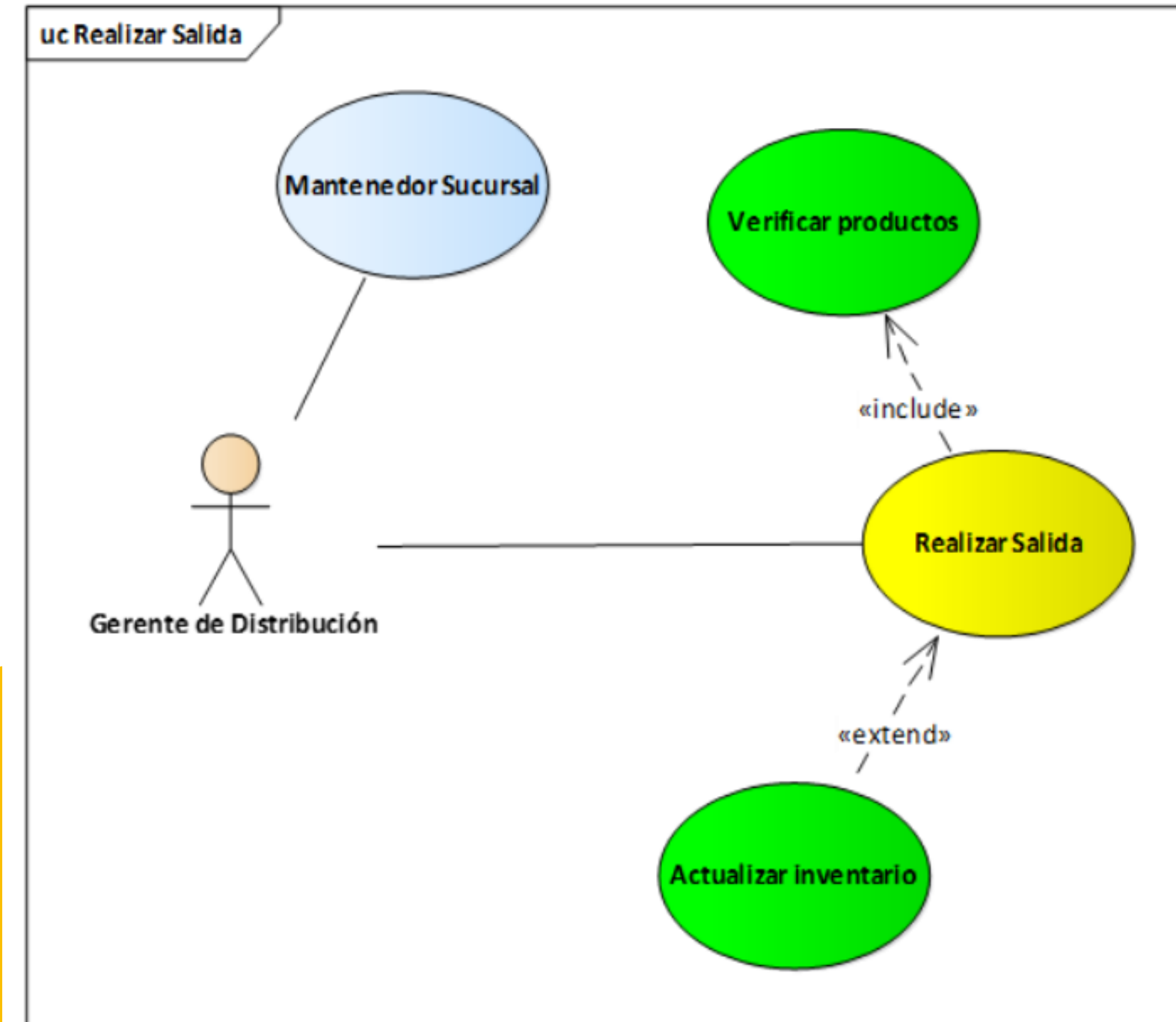




- **CUS Presupuesto Productos**

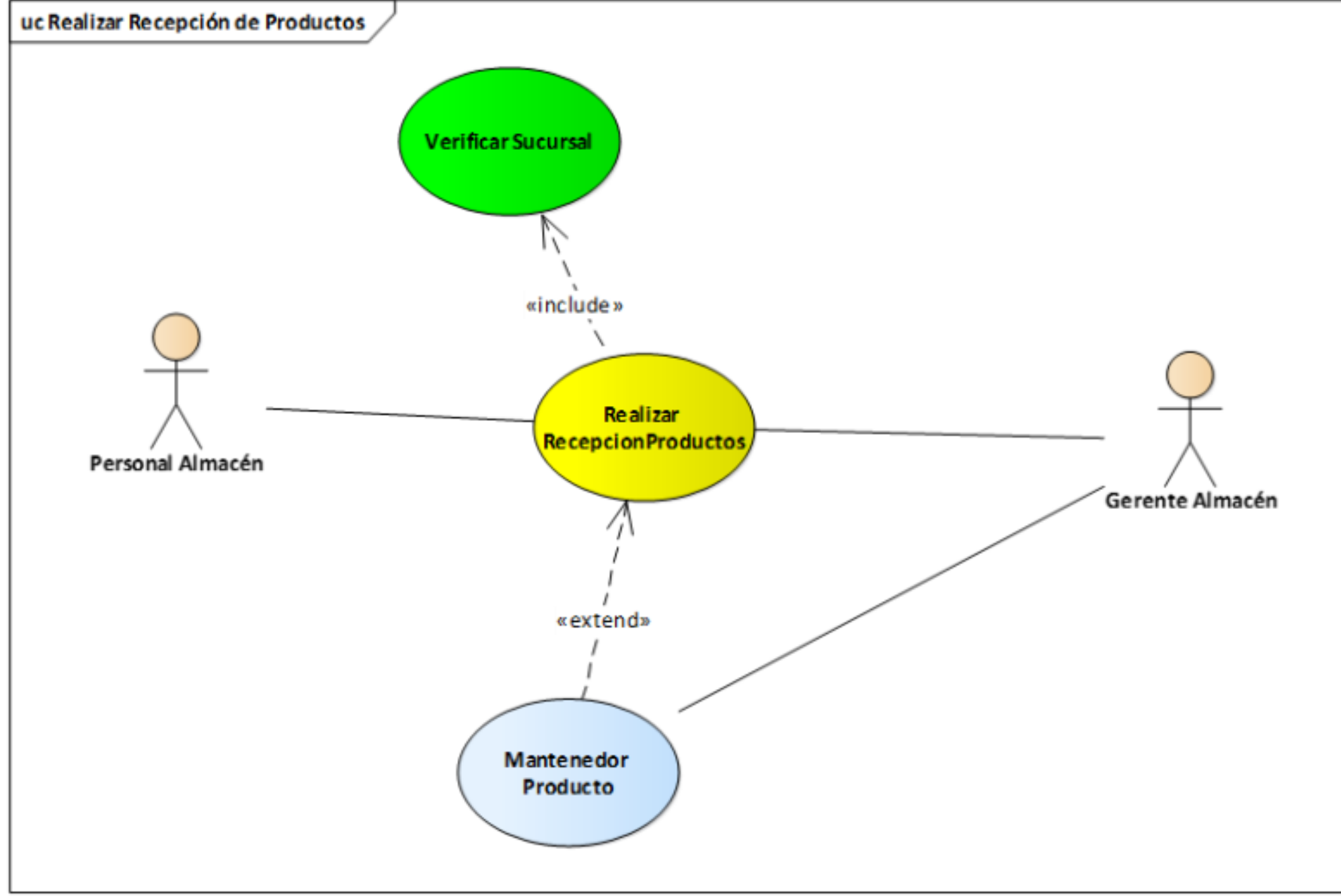


- **CUS Realizar Salida**

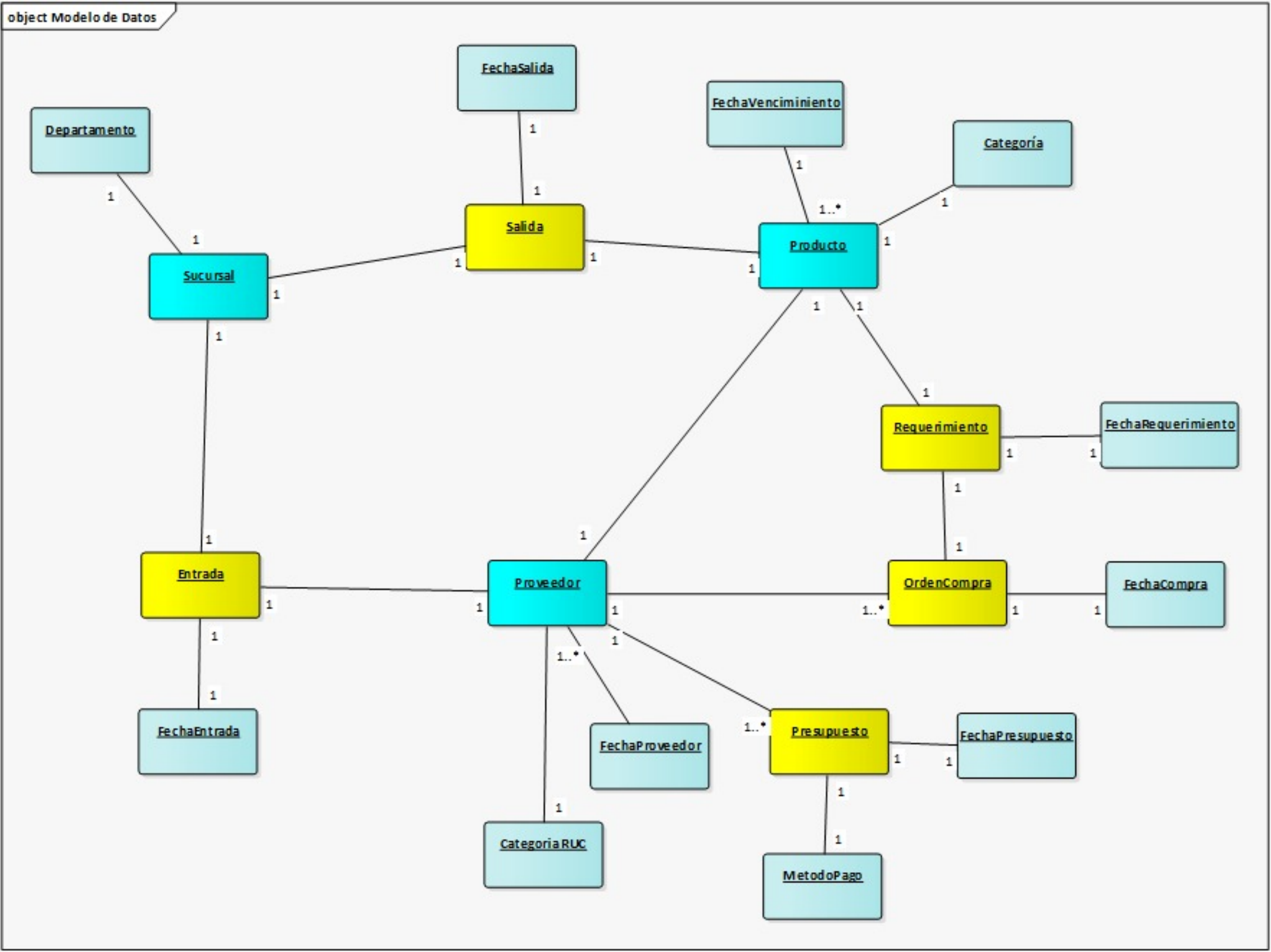




- **CUS Realizar Recepción de Productos**



# DIAGRAMA LOGICO:



# **SOFTWARE Y BASE DE DATOS**

En los próximos minutos, exploraremos en detalle las funcionalidades clave de este software y su integración sólida a una base de datos

# CONCEPTOS A TOMAR EN CUENTA

1

## PATRON SINGLETON

El patrón Singleton es un diseño de software que garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia.

2

## CLASES

Una clase es un diseño que se puede utilizar para crear varios objetos individuales. Los objetos contienen tanto funciones como datos.

3

## CAPAS

La programación por capas es un enfoque arquitectónico utilizado en el desarrollo de software para organizar y estructurar el código en capas o niveles lógicos.

# CAPA ENTIDAD:

```
namespace CapaEntidad
{
    18 referencias
    public class entProveedor
    {
        5 referencias
        public int IDProveedor { get; set; }

        5 referencias
        public string RUCProveedor { get; set; }

        5 referencias
        public string Categoria { get; set; }

        5 referencias
        public string NombreProveedor { get; set; }

        5 referencias
        public string FormaPago { get; set; }

        5 referencias
        public string DireccionProveedor { get; set; }

        5 referencias
        public string TelefonoProveedor { get; set; }

        5 referencias
        public string CorreoElectronicoProveedor { get; set; }

        5 referencias
        public DateTime FechaRegistro { get; set; }

        6 referencias
        public Boolean EstadoProveedor { get; set; }
    }
}
```

# CAPA DATOS:

```
CapaDatos.datProveedor

using CapaEntidad;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CapaDatos
{
    8 referencias
    public class datProveedor
    {
        #region singleton
        private static readonly datProveedor _instancia = new datProveedor();
        4 referencias
        public static datProveedor Instancia
        {
            get
            {
                return datProveedor._instancia;
            }
        }

        #endregion singleton

        metodos
    }
}
```

```
#region metodos
```

```
1 referencia
```

```
public List<entProveedor> ListarProveedor()
{
    SqlCommand cmd = null;
    List<entProveedor> lista = new List<entProveedor>();
    try
    {
        SqlConnection cn = Conexion.Instancia.Conectar();
        cmd = new SqlCommand("spListarProveedor", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            entProveedor Pro = new entProveedor();
            Pro.IDProveedor = Convert.ToInt32(dr["IDProveedor"]);
            Pro.RUCProveedor = dr["RUCProveedor"].ToString();
            Pro.Categoria = dr["Categoria"].ToString();
            Pro.NombreProveedor = dr["NombreProveedor"].ToString();
            Pro.FormaPago = dr["FormaPago"].ToString();
            Pro.DireccionProveedor = dr["DireccionProveedor"].ToString();
            Pro.TelefonoProveedor = dr["TelefonoProveedor"].ToString();
            Pro.CorreoElectronicoProveedor = dr["CorreoElectronicoProveedor"].ToString();
            Pro.FechaRegistro = Convert.ToDateTime(dr["FechaRegistro"]);
            Pro.EstadoProveedor = Convert.ToBoolean(dr["EstadoProveedor"]);
            lista.Add(Pro);
        }
    }
    catch (Exception e)
    {
        throw e;
    }
    finally
    {
        cmd.Connection.Close();
    }
    return lista;
}
```

```
1 referencia
```

```
public Boolean InsertarProveedor(entProveedor Pro)
{
    SqlCommand cmd = null;
    Boolean inserta = false;
    try
    {
        SqlConnection cn = Conexion.Instancia.Conectar();
        cmd = new SqlCommand("spInsertarProveedor", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@RUCProveedor", Pro.RUCProveedor);
        cmd.Parameters.AddWithValue("@Categoria", Pro.Categoria);
        cmd.Parameters.AddWithValue("@NombreProveedor", Pro.NombreProveedor);
        cmd.Parameters.AddWithValue("@FormaPago", Pro.FormaPago);
        cmd.Parameters.AddWithValue("@DireccionProveedor", Pro.DireccionProveedor);
        cmd.Parameters.AddWithValue("@TelefonoProveedor", Pro.TelefonoProveedor);
        cmd.Parameters.AddWithValue("@CorreoElectronicoProveedor", Pro.CorreoElectronicoProveedor);
        cmd.Parameters.AddWithValue("@FechaRegistro", Pro.FechaRegistro);
        cmd.Parameters.AddWithValue("@EstadoProveedor", Pro.EstadoProveedor);
        cn.Open();
        int i = cmd.ExecuteNonQuery();
        if (i > 0)
        {
            inserta = true;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
    finally { cmd.Connection.Close(); }
    return inserta;
}
```



1 referencia

```
public Boolean EditarProveedor(entProveedor Pro)
{
    SqlCommand cmd = null;
    Boolean edita = false;
    try
    {
        SqlConnection cn = Conexion.Instancia.Conectar();
        cmd = new SqlCommand("spEditarProveedor", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@IDProveedor", Pro.IDProveedor);
        cmd.Parameters.AddWithValue("@RUCProveedor", Pro.RUCProveedor);
        cmd.Parameters.AddWithValue("@Categoria", Pro.Categoria);
        cmd.Parameters.AddWithValue("@NombreProveedor", Pro.NombreProveedor);
        cmd.Parameters.AddWithValue("@FormaPago", Pro.FormaPago);
        cmd.Parameters.AddWithValue("@DireccionProveedor", Pro.DireccionProveedor);
        cmd.Parameters.AddWithValue("@TelefonoProveedor", Pro.TelefonoProveedor);
        cmd.Parameters.AddWithValue("@CorreoElectronicoProveedor", Pro.CorreoElectronicoProveedor);
        cmd.Parameters.AddWithValue("@FechaRegistro", Pro.FechaRegistro);
        cmd.Parameters.AddWithValue("@EstadoProveedor", Pro.EstadoProveedor);
        cn.Open();
        int i = cmd.ExecuteNonQuery();
        if (i > 0)
        {
            edita = true;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
    finally { cmd.Connection.Close(); }
    return edita;
}
```

1 referencia

```
public Boolean DeshabilitarProveedor(entProveedor Pro)
{
    SqlCommand cmd = null;
    Boolean delete = false;
    try
    {
        SqlConnection cn = Conexion.Instancia.Conectar();
        cmd = new SqlCommand("spDeshabilitarProveedor", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@IDProveedor", Pro.IDProveedor);
        cn.Open();
        int i = cmd.ExecuteNonQuery();
        if (i > 0)
        {
            delete = true;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
    finally { cmd.Connection.Close(); }
    return delete;
}
```

#endregion metodos



# CAPA LOGICA:

```
9 namespace CapaLogica
10 {
11     12 referencias
12     public class logProveedor
13     {
14         #region sigleton
15
16         private static readonly logProveedor _instancia = new logProveedor();
17
18         8 referencias
19         public static logProveedor Instancia
20         {
21             get
22             {
23                 return logProveedor._instancia;
24             }
25         }
26     #endregion singleton
27
28     [metodos]
29
30 }
31
32 }
```

```
#region metodos

5 referencias
public List<entProveedor> ListarProveedor()
{
    return datProveedor.Instancia.ListarProveedor();
}

1 referencia
public void InsertarProveedor(entProveedor Pro)
{
    datProveedor.Instancia.InsertarProveedor(Pro);
}

1 referencia
public void EditarProveedor(entProveedor Pro)
{
    datProveedor.Instancia.EditarProveedor(Pro);
}

1 referencia
public void DeshabilitarProveedor(entProveedor Pro)
{
    datProveedor.Instancia.DeshabilitarProveedor(Pro);
}

#endregion metodos
```

# CAPA PRESENTACION:

MantenedorProveedor

Mass

Datos del Proveedor

ID del Proveedor:

Nombre:

Fecha de Registro:

lunes . 20 de noviembre d

Ruc Proveedor:

Forma de pago:

Direccion:

Categoria RUC:

Correo Electronico:

Telefono:

☐ Estado Proveedor

Nuevo

Agregar

Editar

Eliminar

Salir

5 referencias

```
public partial class MantendorProveedor : Form
{
    1 referencia
    public MantendorProveedor()
    {
        InitializeComponent();
        listarProveedor();

        txtIdProveedor.Enabled = false;
        dgvProveedor.Enabled = false;
        gbProveedor.Enabled = false;
    }

    4 referencias
    public void listarProveedor()
    {
        dgvProveedor.DataSource = logProveedor.Instancia.ListarProveedor();
    }

    4 referencias
    private void LimpiarVariables()
    {
        txtIdProveedor.Text = "";
        txtRuc.Text = "";
        txtNombre.Text = "";
        txt_Direccion.Text = "";
        txtTelefono.Text = "";
        txtCorreo.Text = "";
        cbCategoria.Text = "";
        cbForma_Pago.Text = "";
        cbxEstProveedor.Checked = false;
    }

    1 referencia
    private void btnNuevo_Click(object sender, EventArgs e)
    {
        txtIdProveedor.Enabled = false;
        gbProveedor.Enabled = true;
        dgvProveedor.Enabled = true;
        LimpiarVariables();
    }
}
```

1 referencia

```
private void btnSalir_Click(object sender, EventArgs e)
{
    Close();
}
```

1 referencia

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    try
    {
        entProveedor p = new entProveedor();
        p.RUCProveedor = txtRuc.Text.Trim();
        p.Categoria = cbCategoria.Text.Trim();
        p.NombreProveedor = txtNombre.Text.Trim();
        p.FormaPago = cbForma_Pago.Text.Trim();
        p.DireccionProveedor = txt_Direccion.Text.Trim();
        p.TelefonoProveedor = txtTelefono.Text.Trim();
        p.CorreoElectronicoProveedor = txtCorreo.Text.Trim();
        p.FechaRegistro = dtpProveedor.Value;
        p.EstadoProveedor = cbxEstProveedor.Checked;
        logProveedor.Instancia.InsertarProveedor(p);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error.." + ex);
    }
    LimpiarVariables();
    listarProveedor();
    gbProveedor.Enabled = true;
}
```

```
private void dgvProveedor_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    txtIdProveedor.Enabled = true;
    gbProveedor.Enabled = true;
    DataGridViewRow filaActual = dgvProveedor.Rows[e.RowIndex];
    txtIdProveedor.Text = filaActual.Cells[0].Value.ToString();
    txtRuc.Text = filaActual.Cells[1].Value.ToString();
    cbCategoria.Text = filaActual.Cells[2].Value.ToString();
    txtNombre.Text = filaActual.Cells[3].Value.ToString();
    cbForma_Pago.Text = filaActual.Cells[4].Value.ToString();
    txt_Direccion.Text = filaActual.Cells[5].Value.ToString();
    txtTelefono.Text = filaActual.Cells[6].Value.ToString();
    txtCorreo.Text = filaActual.Cells[7].Value.ToString();
    dtpProveedor.Text = filaActual.Cells[8].Value.ToString();
    cbxEstProveedor.Checked = Convert.ToBoolean(filaActual.Cells[9].Value);
}
```

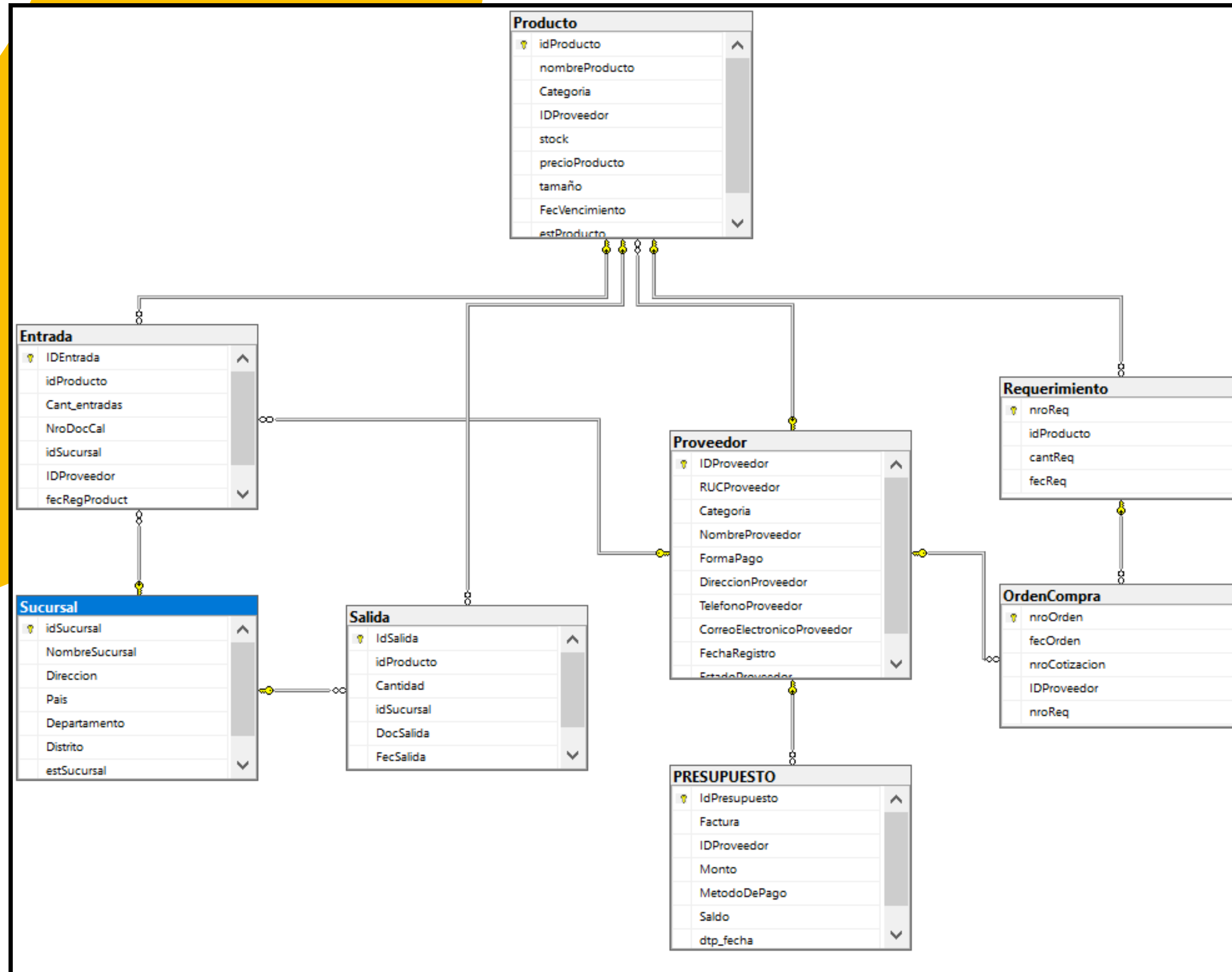
1 referencia

```
private void btnEditar_Click(object sender, EventArgs e)
{
    gbProveedor.Enabled = true;
    try
    {
        entProveedor p = new entProveedor();
        p.IDProveedor = int.Parse(txtIdProveedor.Text.Trim());
        p.RUCProveedor = txtRuc.Text.Trim();
        p.Categoria = cbCategoria.Text.Trim();
        p.NombreProveedor = txtNombre.Text.Trim();
        p.FormaPago = cbForma_Pago.Text.Trim();
        p.DireccionProveedor = txt_Direccion.Text.Trim();
        p.TelefonoProveedor = txtTelefono.Text.Trim();
        p.CorreoElectronicoProveedor = txtCorreo.Text.Trim();
        p.FechaRegistro = dtpProveedor.Value;
        p.EstadoProveedor = cbxEstProveedor.Checked;
        logProveedor.Instance.EditarProveedor(p);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error.." + ex);
    }
    LimpiarVariables();
    listarProveedor();
    gbProveedor.Enabled = true;
}
```

1 referencia

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    gbProveedor.Enabled = true;
    try
    {
        entProveedor p = new entProveedor();
        p.IDProveedor = int.Parse(txtIdProveedor.Text.Trim());
        cbxEstProveedor.Checked = false;
        p.EstadoProveedor = cbxEstProveedor.Checked;
        logProveedor.Instance.DeshabilitarProveedor(p);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error.." + ex);
    }
    LimpiarVariables();
    gbProveedor.Enabled = false;
    listarProveedor();
}
```

# BASE DE DATOS:





## REFERENCIAS BIBLIOGRAFICAS

- Tiendas Mass. (s.f.). Tiendas Mass. Recuperado de <https://www.tiendasmass.com.pe/>
- Patrón de diseño: Singleton. (2022, octubre 3). Java. <https://codigojava.online/patron-de-diseno-singleton/>
- Clases (programación). (s/f). Ecured.cu. Recuperado el 20 de noviembre de 2023, de [https://www.ecured.cu/Clases\\_%28programaci%C3%B3n%29](https://www.ecured.cu/Clases_%28programaci%C3%B3n%29)



# GRACIAS

UNIVERSIDAD PRIVADA DEL NORTE