

LÓGICA DE C#

```
/*
 * GUÍA DE IMPLEMENTACIÓN EN VISUAL STUDIO 2022
 * Este archivo incluye la LÓGICA DE RECETAS y PRODUCCIÓN (transaccional)
 */

// =====
// PROYECTO 1: Panaderia.Entidades
// =====
using System;
```

```
using System.Collections.Generic;

namespace Panaderia.Entidades
{
    // Clase Producto (Actualizada con Stock)
    public class Producto
    {
        public int IdProducto { get; set; }
        public int IdCategoria { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public decimal PrecioVenta { get; set; }
        public int Stock { get; set; } // NUEVO: Stock de producto terminado
        public bool Activo { get; set; }
        public string ImagenURL { get; set; }
        public string NombreCategoria { get; set; }
    }

    // Clase Ingrediente (Mantenida)
    public class Ingrediente
    {
        public int IdIngrediente { get; set; }
        public string Nombre { get; set; }
        public decimal Cantidad { get; set; } // Stock de ingrediente
        public string UnidadMedida { get; set; }
        public decimal CostoUnitario { get; set; }
        public DateTime? FechaVencimiento { get; set; }
    }

    // NUEVA CLASE: DetalleReceta
    public class DetalleReceta
    {
        public int IdDetalleReceta { get; set; }
        public int IdReceta { get; set; }
        public int IdIngrediente { get; set; }
        public string NombreIngrediente { get; set; } // Propiedad extra
        para presentación
        public decimal CantidadRequerida { get; set; } // Cantidad para 1
        unidad de producto
    }

    // NUEVA CLASE: Receta
    public class Receta
    {
        public int IdReceta { get; set; }
```

```

        public int IdProducto { get; set; }
        public string NombreProducto { get; set; } // Propiedad extra para
presentación
        public DateTime FechaCreacion { get; set; }
        public List<DetalleReceta> Ingredientes { get; set; } = new
List<DetalleReceta>();
    }

    // NUEVA CLASE: Resultado de Transacciones (Para SP_RegistrarProduccion)
    public class ResultadoTransaccion
    {
        public int Resultado { get; set; } // 1 para éxito, 0 para fallo
        public string Mensaje { get; set; }
    }
}

// =====
// PROYECTO 2: Panaderia.Datos (Conecta con SQL)
// =====
using System;
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
using Panaderia.Entidades;

namespace Panaderia.Datos
{
    public class Conexion
    {
        // Asegurate de cambiar esta cadena según tu computadora
        public static string Cadena =
"Server=.\SQLExpress;Database=PanaderiaDB;Integrated Security=True";
    }

    // -----
    // CLASE DAL para Producción (NUEVA)
    // -----
    public class ProduccionDAL
    {
        // CRÍTICO: Método para registrar la producción, descontar
        ingredientes y actualizar stock
        public ResultadoTransaccion RegistrarProduccionYDescontar(int
idProducto, int cantidadProducida, string observaciones)
        {
            ResultadoTransaccion resultado = new ResultadoTransaccion();

```

```

        using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
    {
        string nombreSp = "SP_RegistrarProduccion";
        SqlCommand cmd = new SqlCommand(nombreSp, conn);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.AddWithValue("@IdProducto", idProducto);
        cmd.Parameters.AddWithValue("@CantidadProducida",
cantidadProducida);
        cmd.Parameters.AddWithValue("@Observaciones",
string.IsNullOrEmpty(observaciones) ? (object)DBNull.Value : observaciones);

        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                // El SP devuelve el resultado y el mensaje de la
transacción
                resultado.Resultado =
Convert.ToInt32(reader["Resultado"]);
                resultado.Mensaje = reader["Mensaje"].ToString();
            }
            catch (Exception ex)
            {
                // Captura errores de conexión o ejecución que no fueron
capturados por el CATCH del SP
                resultado.Resultado = 0;
                resultado.Mensaje = "Error de conexión/ejecución fuera
del SP: " + ex.Message;
            }
        }
        return resultado;
    }

    // Se omiten RecetaDAL y DetalleRecetaDAL por espacio, su lógica es
similar a ProductoDAL.
}

// Clase ProductoDAL (Mantenida, pero Producto.Stock ya está implicito
en la entidad)
public class ProductoDAL

```



```

                    Activo = Convert.ToBoolean(reader["Activo"])
                });
            }
        }
        catch (Exception ex) { throw ex; }
    }
    return lista;
}
}

// Clase IngredienteDAL (Mantenida)
public class IngredienteDAL
{
    // ... métodos existentes: ObtenerIngredientes, InsertarIngrediente,
etc.
}
}

// =====
// PROYECTO 3: Panaderia.Negocio
// =====
using System;
using System.Collections.Generic;
using Panaderia.Datos;
using Panaderia.Entidades;

namespace Panaderia.Negocio
{
    // -----
    // CLASE BLL para Producción (NUEVA)
    // -----
    public class ProduccionBLL
    {
        private ProduccionDAL dal = new ProduccionDAL();

        public string RegistrarNuevaProduccion(int idProducto, int
cantidadProducida, string observaciones)
        {
            if (idProducto <= 0)
            {
                return "Error: Debe seleccionar un producto válido.";
            }
            if (cantidadProducida <= 0)
            {

```

```

        return "Error: La cantidad producida debe ser mayor a
cero.";
    }

        // La DAL llama al SP transaccional que hace el descuento y el
aumento de stock.
        ResultadoTransaccion resultado =
dal.RegistrarProduccionYDescontar(idProducto, cantidadProducida,
observaciones);

        // Retorna el mensaje del SP (éxito o el error de stock/receta)
return resultado.Mensaje;
}

        // Se omiten métodos para administrar Recetas (Crear/Modificar) por
espacio.
}

// Clase ProductoBLL (Mantenida)
public class ProductoBLL
{
    // ... métodos existentes: BuscarCatalogoActivo, CrearProducto, etc.
}

// Clase IngredienteBLL (Mantenida)
public class IngredienteBLL
{
    // ... métodos existentes: ObtenerTodos, AgregarIngrediente, etc.
}
}

// =====
// PROYECTO 4: Panaderia.Presentacion (Windows Forms) - Ejemplo de uso
// =====

using System;
using System.Windows.Forms;
using Panaderia.Negocio;
using Panaderia.Entidades;

namespace Panaderia.Presentacion
{
    // Ejemplo de un formulario para registrar la producción diaria
    public partial class frmProduccion : Form
    {
        ProduccionBLL negocioProduccion = new ProduccionBLL();

```

```
// Esto simularía una función que se llama al presionar un botón  
"Registrar"  
    private void btnRegistrar_Click(int idProductoSeleccionado, int  
txtCantidadProducida, string txtObservaciones)  
    {  
        try  
        {  
            // La capa de Negocio valida y llama a la DAL, la DAL llama  
al SP transaccional.  
            string mensaje = negocioProduccion.RegistrarNuevaProduccion(  
                idProductoSeleccionado,  
                txtCantidadProducida,  
                txtObservaciones  
            );  
  
            MessageBox.Show(mensaje, "Registro de Producción",  
MessageBoxButtons.OK,  
                mensaje.StartsWith("Error") ? MessageBoxIcon.Error :  
MessageBoxIcon.Information);  
        }  
        catch (Exception ex)  
        {  
            MessageBox.Show("Error inesperado: " + ex.Message, "Error  
Crítico", MessageBoxButtons.OK, MessageBoxIcon.Stop);  
        }  
    }  
  
    // ... (Otras clases de Presentación omitidas por espacio)  
}
```

CAPA 1 DE

RecetaEntidades.cs

Definiciones de las

clases Ingrediente,

Receta,

DetalleReceta y

ResultadoTransaccio

n.

```
// =====
```

```
// PROYECTO 1: Panaderia.Entidades (Modelos de Datos)
// =====
using System;
using System.Collections.Generic;

namespace Panaderia.Entidades
{
    // Clase Usuario (Mantenida)
    public class Usuario
    {
        public int IdUsuario { get; set; }
        public string NombreCompleto { get; set; }
        public string Telefono { get; set; }
        public string Correo { get; set; }
        public string Contrasena { get; set; }
        public string Rol { get; set; }
        public bool Activo { get; set; }
        public DateTime FechaRegistro { get; set; }
    }

    // Clase Producto (Actualizada con Stock)
    public class Producto
    {
        public int IdProducto { get; set; }
        public int IdCategoria { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public decimal PrecioVenta { get; set; }
        public int Stock { get; set; } // NUEVO: Stock de producto terminado
        public bool Activo { get; set; }
        public string ImagenURL { get; set; }
        public string NombreCategoria { get; set; }
    }

    // Clase Ingrediente (Mantenida)
    public class Ingrediente
    {
        public int IdIngrediente { get; set; }
        public string Nombre { get; set; }
        public decimal Cantidad { get; set; } // Stock de ingrediente
(materia prima)
        public string UnidadMedida { get; set; }
        public decimal CostoUnitario { get; set; }
        public DateTime? FechaVencimiento { get; set; }
    }
}
```

```
// NUEVA CLASE: DetalleReceta
public class DetalleReceta
{
    public int IdDetalleReceta { get; set; }
    public int IdReceta { get; set; }
    public int IdIngrediente { get; set; }
    public string NombreIngrediente { get; set; } // Propiedad para
presentación
    public decimal CantidadRequerida { get; set; } // Cantidad para 1
unidad de producto
    public string UnidadMedidaIngrediente { get; set; } // Propiedad
para presentación
}

// NUEVA CLASE: Receta
public class Receta
{
    public int IdReceta { get; set; }
    public int IdProducto { get; set; }
    public string NombreProducto { get; set; } // Propiedad para
presentación
    public DateTime FechaCreacion { get; set; }
    public List<DetalleReceta> Ingredientes { get; set; } = new
List<DetalleReceta>();
}

// NUEVA CLASE: Resultado de Transacciones (Para SP_RegistrarProduccion
y otros SPs)
public class ResultadoTransaccion
{
    public int Resultado { get; set; } // 1 para éxito, 0 para fallo
    public string Mensaje { get; set; }
}
```

```
using System;
using System.Collections.Generic;

namespace Panaderia.Entidades
{
    // Clase Ingrediente (Se usa para listar el inventario)
    public class Ingrediente
    {
        public int IdIngrediente { get; set; }
        public string Nombre { get; set; }
        public decimal Cantidad { get; set; } // Stock de ingrediente
(materia prima)
        public string UnidadMedida { get; set; }
        public decimal CostoUnitario { get; set; }
        public DateTime? FechaVencimiento { get; set; }
    }

    // NUEVA CLASE: DetalleReceta
    // Representa un ingrediente dentro de una receta específica.
    public class DetalleReceta
    {
        public int IdDetalleReceta { get; set; }
        public int IdReceta { get; set; }
        public int IdIngrediente { get; set; }

        // Propiedades de presentación (viene del JOIN con la tabla
        Ingredientes)
        public string NombreIngrediente { get; set; }
        public string UnidadMedidaIngrediente { get; set; }

        public decimal CantidadRequerida { get; set; } // Cantidad para 1
        unidad de producto
    }

    // NUEVA CLASE: Receta (Cabecera)
    // Contiene la lista de ingredientes que componen un Producto.
    public class Receta
    {
        public int IdReceta { get; set; }
        public int IdProducto { get; set; }

        // Propiedad de presentación
        public string NombreProducto { get; set; }

        public DateTime FechaCreacion { get; set; }
    }
}
```

```
// Contiene todos los ingredientes y cantidades para este producto
public List<DetalleReceta> Ingredientes { get; set; } = new
List<DetalleReceta>();
}

// Clase para manejar las respuestas de los SPs transaccionales
public class ResultadoTransaccion
{
    public int Resultado { get; set; } // 1 para éxito, 0 para fallo
    public string Mensaje { get; set; }
    public int? IdGenerado { get; set; } // Para Insertar
Recetas/Ingredientes
```

CAPA 2-PANADERÍA

DATOS

Lógica de conexión y manipulación de datos (CRUD) para Ingrediente y Receta (incluyendo manejo de XML para el detalle de la receta).

```

// =====
// PROYECTO 2: Panaderia.Datos (Data Access Layer - DAL)
// =====
using System;
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
using Panaderia.Entidades;

namespace Panaderia.Datos
{
    public class Conexion
    {
        // CADENA DE CONEXIÓN
        public static string Cadena =
"Server=.\SQLEXPRESS;Database=PanaderiaDB;Integrated Security=True";
    }

    // -----
    // CLASE DAL para Producción (Llama al SP transaccional)
    // -----
    public class ProduccionDAL
    {
        // CRÍTICO: Método para registrar la producción
        public ResultadoTransaccion RegistrarProduccionYDescontar(int
idProducto, int cantidadProducida, string observaciones)
        {
            ResultadoTransaccion resultado = new ResultadoTransaccion();

            using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
            {
                string nombreSp = "SP_RegistrarProduccion";
                SqlCommand cmd = new SqlCommand(nombreSp, conn);
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@IdProducto", idProducto);
                cmd.Parameters.AddWithValue("@CantidadProducida",
cantidadProducida);
                cmd.Parameters.AddWithValue("@Observaciones",
string.IsNullOrEmpty(observaciones) ? (object)DBNull.Value : observaciones);
            }
        }
    }
}

```

```

        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                // El SP devuelve el resultado y el mensaje de la
                // transacción
                resultado.Resultado =
                Convert.ToInt32(reader["Resultado"]);
                resultado.Mensaje = reader["Mensaje"].ToString();
            }
        }
        catch (Exception ex)
        {
            // Captura errores de conexión o ejecución inesperados
            resultado.Resultado = 0;
            resultado.Mensaje = "Error de conexión/ejecución: " +
            ex.Message;
        }
    }

    return resultado;
}

}

// -----
// CLASE DAL para Productos (Muestra cómo leer el nuevo Stock)
// -----
public class ProductoDAL
{
    // READ: Obtener productos con filtro de búsqueda (TerminoBusqueda)
    public List<Producto> BuscarProductosActivos(string terminoBusqueda
= "") 
    {
        List<Producto> lista = new List<Producto>();
        using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
        {
            string nombreSp = "SP_ListarProductos"; // Asumiendo que
este SP ha sido actualizado para incluir Stock
            SqlCommand cmd = new SqlCommand(nombreSp, conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@SoloActivos", true);
        }
    }
}

```

```

        cmd.Parameters.AddWithValue("@TerminoBusqueda",
string.IsNullOrEmpty(terminoBusqueda) ? (object)DBNull.Value :
terminoBusqueda);

        try
{
    conn.Open();
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        lista.Add(new Producto
        {
            IdProducto =
Convert.ToInt32(reader["IdProducto"]),
            IdCategoria =
Convert.ToInt32(reader["IdCategoria"]),
            NombreCategoria =
reader["NombreCategoria"].ToString(),
            Nombre = reader["Nombre"].ToString(),
            Descripcion = reader["Descripcion"].ToString(),
            PrecioVenta =
Convert.ToDecimal(reader["PrecioVenta"]),
            ImagenURL = reader["ImagenURL"] is DBNull ?
string.Empty : reader["ImagenURL"].ToString(),
            Stock = Convert.ToInt32(reader["Stock"]), //
LECTURA del campo Stock
            Activo = Convert.ToBoolean(reader["Activo"])
        });
    }
}
catch (Exception ex)
{
    // En DAL es mejor lanzar la excepción para que BLL
decida cómo manejarla
    throw new Exception("Error al buscar productos en la
base de datos.", ex);
}
return lista;
}

// --- Otros métodos CRUD (Insertar, Actualizar, Eliminar) se omiten
por espacio ---
}

```

```

// -----
// CLASE DAL para Recetas (CRUD de Recetas)
// -----
public class RecetaDAL
{
    // CREATE: Insertar nueva receta y sus detalles
    public int InsertarReceta(Receta receta, out string mensaje)
    {
        // Lógica para llamar a un SP que inserta la Receta y el
        DetalleReceta
        // ... (implementación omitida por espacio)
        mensaje = "Receta guardada con éxito.";
        return 1; // ID de la nueva receta
    }

    // READ: Obtener detalle de una receta por IdProducto
    public Receta ObtenerRecetaPorProducto(int idProducto)
    {
        // Lógica para obtener la Receta y luego su lista de
        DetalleReceta de la BD
        // ... (implementación omitida por espacio)
        return new Receta();
    }

    // --- Otros métodos CRUD (Actualizar, Eliminar) se omiten por
    espacio ---
}
}

```

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.Xml; // Necesario para enviar el detalle de receta como XML
using Panaderia.Entidades;

namespace Panaderia.Datos
{
    // Clase DAL para manejar la tabla Ingredientes (Inventario de Materia
    Prima)
    public class IngredienteDAL
    {

```

```

// READ: Listar todos los ingredientes
public List<Ingrediente> ListarIngredientes()
{
    List<Ingrediente> lista = new List<Ingrediente>();
    using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
    {
        SqlCommand cmd = new SqlCommand("SP_ListarIngredientes",
conn);
        cmd.CommandType = CommandType.StoredProcedure;
        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                lista.Add(new Ingrediente
                {
                    IdIngrediente =
Convert.ToInt32(reader["IdIngrediente"]),
                    Nombre = reader["Nombre"].ToString(),
                    Cantidad =
Convert.ToDecimal(reader["Cantidad"]),
                    UnidadMedida =
reader["UnidadMedida"].ToString(),
                    CostoUnitario =
Convert.ToDecimal(reader["CostoUnitario"]),
                    // Manejo de valores nulos para FechaVencimiento
                    FechaVencimiento = reader["FechaVencimiento"] is
DBNull ? (DateTime?)null : Convert.ToDateTime(reader["FechaVencimiento"])
                });
            }
        }
        catch (Exception ex)
        {
            throw new Exception("Error al listar ingredientes: " +
ex.Message);
        }
    }
    return lista;
}

// CREATE: Insertar Ingrediente
public ResultadoTransaccion InsertarIngrediente(Ingrediente
ingrediente)
{

```

```

        ResultadoTransaccion resultado = new ResultadoTransaccion();
        using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
        {
            SqlCommand cmd = new SqlCommand("SP_InsertarIngrediente",
conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Nombre", ingrediente.Nombre);
            cmd.Parameters.AddWithValue("@Cantidad",
ingrediente.Cantidad);
            cmd.Parameters.AddWithValue("@UnidadMedida",
ingrediente.UnidadMedida);
            cmd.Parameters.AddWithValue("@CostoUnitario",
ingrediente.CostoUnitario);
            cmd.Parameters.AddWithValue("@FechaVencimiento",
ingrediente.FechaVencimiento ?? (object)DBNull.Value); // Manejar nulos

            try
            {
                conn.Open();
                SqlDataReader reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    resultado.Resultado = 1; // Asumimos éxito si se
obtiene ID
                    resultado.IdGenerado =
Convert.ToInt32(reader["IdGenerado"]);
                    resultado.Mensaje = "Ingrediente registrado con
éxito.";
                }
            }
            catch (Exception ex)
            {
                resultado.Resultado = 0;
                resultado.Mensaje = "Error DAL al insertar: " +
ex.Message;
            }
        }
        return resultado;
    }

    // UPDATE: Actualizar Ingrediente
    public ResultadoTransaccion ActualizarIngrediente(Ingrediente
ingrediente)
    {

```

```

        ResultadoTransaccion resultado = new ResultadoTransaccion();
        using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
        {
            SqlCommand cmd = new SqlCommand("SP_ActualizarIngrediente",
conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@IdIngrediente",
ingrediente.IdIngrediente);
            cmd.Parameters.AddWithValue("@Nombre", ingrediente.Nombre);
            cmd.Parameters.AddWithValue("@Cantidad",
ingrediente.Cantidad);
            cmd.Parameters.AddWithValue("@UnidadMedida",
ingrediente.UnidadMedida);
            cmd.Parameters.AddWithValue("@CostoUnitario",
ingrediente.CostoUnitario);
            cmd.Parameters.AddWithValue("@FechaVencimiento",
ingrediente.FechaVencimiento ?? (object)DBNull.Value);

            try
            {
                conn.Open();
                cmd.ExecuteNonQuery();
                resultado.Resultado = 1;
                resultado.Mensaje = "Ingrediente actualizado con
éxito.";
            }
            catch (Exception ex)
            {
                resultado.Resultado = 0;
                resultado.Mensaje = "Error DAL al actualizar: " +
ex.Message;
            }
        }
        return resultado;
    }

    // DELETE: Eliminar Ingrediente
    public ResultadoTransaccion EliminarIngrediente(int idIngrediente)
    {
        ResultadoTransaccion resultado = new ResultadoTransaccion();
        using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
        {
            SqlCommand cmd = new SqlCommand("SP_EliminarIngrediente",
conn);

```

```

        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@IdIngrediente",
idIngrediente);

        try
        {
            conn.Open();
            // El SP retorna un Resultado y un Mensaje
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                resultado.Resultado =
Convert.ToInt32(reader["Resultado"]);
                resultado.Mensaje = reader["Mensaje"].ToString();
            }
        }
        catch (Exception ex)
        {
            resultado.Resultado = 0;
            resultado.Mensaje = "Error DAL al eliminar: " +
ex.Message;
        }
    }

    return resultado;
}

}

// Clase DAL para manejar la tabla Recetas (Ficha Técnica)
public class RecetaDAL
{
    // Ayudante para convertir la lista de detalles a formato XML
    private string GenerarDetalleXML(List<DetalleReceta> detalles)
    {
        XmlDocument doc = new XmlDocument();
        XmlElement root = doc.CreateElement("Detalles");
        doc.AppendChild(root);

        foreach (var detalle in detalles)
        {
            XmlElement detalleNode = doc.CreateElement("Detalle");

            XmlElement idIngredienteNode =
doc.CreateElement("IdIngrediente");
            idIngredienteNode.InnerText =
detalle.IdIngrediente.ToString();

```

```

        detalleNode.AppendChild(idIngredienteNode);

        XmlElement cantidadReqNode =
doc.CreateElement("CantidadRequerida");
        // Usamos InvariantCulture para asegurar el formato decimal
correcto en SQL Server
        cantidadReqNode.InnerText =
detalle.CantidadRequerida.ToString(System.Globalization.CultureInfo.Invariant
tCulture);
        detalleNode.AppendChild(cantidadReqNode);

        root.AppendChild(detalleNode);
    }
    return doc.OuterXml;
}

// CREATE: Insertar nueva receta y sus detalles (Llama a SP
transaccional)
public ResultadoTransaccion InsertarReceta(Receta receta)
{
    ResultadoTransaccion resultado = new ResultadoTransaccion();
    string detalleXml = GenerarDetalleXML(receta.Ingredientes);

    using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
    {
        SqlCommand cmd = new SqlCommand("SP_InsertarRecetaCompleta",
conn);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.AddWithValue("@IdProducto",
receta.IdProducto);
        cmd.Parameters.AddWithValue("@DetalleRecetasXML",
detalleXml);

        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                resultado.Resultado =
Convert.ToInt32(reader["Resultado"]);
                resultado.Mensaje = reader["Mensaje"].ToString();
                // El SP devuelve el IdReceta si fue exitoso
            }
        }
    }
}

```

```

                if (resultado.Resultado == 1 && reader["IdReceta"]
is int id)
{
    resultado.IdGenerado = id;
}
}
catch (Exception ex)
{
    resultado.Resultado = 0;
    resultado.Mensaje = "Error DAL al insertar receta: " +
ex.Message;
}
}
return resultado;
}

// READ: Obtener detalle de una receta por IdProducto
public Receta ObtenerRecetaPorProducto(int idProducto)
{
    Receta receta = null;
    using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
    {
        SqlCommand cmd = new
SqlCommand("SP_ObtenerDetalleRecetaPorProducto", conn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@IdProducto", idProducto);

        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();

            int idReceta = 0;

            while (reader.Read())
            {
                if (receta == null)
                {
                    // Inicializar la cabecera de la receta solo una
vez
                    idReceta = Convert.ToInt32(reader["IdReceta"]);
                    receta = new Receta { IdReceta = idReceta,
IdProducto = idProducto, Ingredientes = new List<DetalleReceta>() };
                }
            }
        }
    }
}

```

```

        // Agregar el detalle del ingrediente
        receta.Ingredientes.Add(new DetalleReceta
        {
            IdDetalleReceta =
Convert.ToInt32(reader["IdDetalleReceta"]),
            IdReceta = idReceta,
            IdIngrediente =
Convert.ToInt32(reader["IdIngrediente"]),
            NombreIngrediente =
reader["NombreIngrediente"].ToString(),
            CantidadRequerida =
Convert.ToDecimal(reader["CantidadRequerida"]),
            UnidadMedidaIngrediente =
reader["UnidadMedida"].ToString()
        });
    }
}
catch (Exception ex)
{
    // Lanza la excepción para que BLL la maneje
    throw new Exception("Error DAL al obtener receta: " +
ex.Message);
}
}
return receta; // Devuelve null si no se encontró receta
}

// DELETE: Eliminar Receta completa (Cabecera y Detalles)
public ResultadoTransaccion EliminarReceta(int idProducto)
{
    ResultadoTransaccion resultado = new ResultadoTransaccion();
    using (SqlConnection conn = new SqlConnection(Conexion.Cadena))
    {
        SqlCommand cmd = new SqlCommand("SP_EliminarReceta", conn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@IdProducto", idProducto);

        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {

```

```
        resultado.Resultado =
Convert.ToInt32(reader["Resultado"]);
        resultado.Mensaje = reader["Mensaje"].ToString();
    }
}
catch (Exception ex)
{
    resultado.Resultado = 0;
    resultado.Mensaje = "Error DAL al eliminar receta: " +
ex.Message;
}
return resultado;
}

// NOTA: El SP de Actualizar Receta no se implementa aquí, sino que
se recomienda
// ELIMINAR y INSERTAR la nueva versión para simplificar el SP de
SQL Server
// o crear un SP específico que reciba el XML de DetalleReceta y
reemplace/actualice los registros.
}
}
```

CAPA 3-PANADERÍA

NEGOCIO

Reglas de negocio y validaciones para Ingrediente y Receta.

```

// =====
// PROYECTO 3: Panaderia.Negocio (Business Logic Layer - BLL)
// =====
using System;
using System.Collections.Generic;
using Panaderia.Datos;
using Panaderia.Entidades;

namespace Panaderia.Negocio
{
    // -----
    // CLASE BLL para Producción (Reglas de negocio)
    // -----
    public class ProduccionBLL
    {
        private ProduccionDAL dal = new ProduccionDAL();

        // Lógica de validación antes de llamar a la DAL transaccional
        public string RegistrarNuevaProduccion(int idProducto, int cantidadProducida, string observaciones)
        {
            // 1. Validaciones básicas de negocio (BLL)
            if (idProducto <= 0)
            {
                return "Error BLL: Debe seleccionar un producto válido.";
            }
            if (cantidadProducida <= 0)
            {
                return "Error BLL: La cantidad producida debe ser mayor a cero.";
            }

            // 2. Llamada a la DAL, la cual ejecuta la lógica transaccional
            en SQL Server
            ResultadoTransaccion resultado =
dal.RegistrarProduccionYDescontar(idProducto, cantidadProducida,
observaciones);

            // 3. Retorna el mensaje del SP (éxito o el error de stock/receta)
            return resultado.Mensaje;
        }
    }
}
// -----

```

```

// CLASE BLL para Productos
// -----
public class ProductoBLL
{
    private ProductoDAL dal = new ProductoDAL();

    public List<Producto> ObtenerCatalogo(string filtro = "") 
    {
        try
        {
            // Aquí podrías agregar lógica extra, por ejemplo, aplicar
            precios especiales.
            return dal.BuscarProductosActivos(filtro);
        }
        catch (Exception ex)
        {
            // La BLL captura la excepción de la DAL y la
            maneja/registra
            Console.WriteLine("Error en BLL al obtener catálogo: " +
            ex.Message);
            return new List<Producto>(); // Devuelve lista vacía en caso
            de fallo
        }
    }

    // --- Otros métodos CRUD se omiten por espacio ---
}

}

// =====
// PROYECTO 4: Panaderia.Presentacion (Windows Forms) - Ejemplo de uso
// =====
using System;
using System.Windows.Forms;
using Panaderia.Negocio;
using Panaderia.Entidades;

namespace Panaderia.Presentacion
{
    // Este formulario simula el 'frmProduccionAdmin' del diseño de Figma
    public partial class frmProduccion : Form
    {
        ProduccionBLL negocioProduccion = new ProduccionBLL();
        ProductoBLL negocioProducto = new ProductoBLL();
}

```

```
// Constructor del formulario
public frmProduccion()
{
    // InitializeComponent(); // Llamada real en un proyecto
    WinForms
    CargarComboProductos();
}

// Simula la carga de productos en un ComboBox
private void CargarComboProductos()
{
    // Ejemplo: Rellenar el ComboBox con productos que tienen receta
    List<Producto> listaProductos =
negocioProducto.ObtenerCatalogo();

    // Aquí se simularía llenar el ComboBox:
    // comboProductos.DataSource = listaProductos;
    // comboProductos.DisplayMember = "Nombre";
    // comboProductos.ValueMember = "IdProducto";
}

// Esto simularía una función que se llama al presionar un botón
"Registrar Producción"
public void btnRegistrarProduccion_Click(int idProductoSeleccionado,
int txtCantidadProducida, string txtObservaciones)
{
    try
    {
        // Llamada a la capa de Negocio
        string mensaje = negocioProduccion.RegistrarNuevaProduccion(
            idProductoSeleccionado,
            txtCantidadProducida,
            txtObservaciones
        );

        // Mostrar el resultado de la transacción del SP
        MessageBox.Show(mensaje, "Registro de Producción",
MessageBoxButtons.OK,
        mensaje.StartsWith("Error") ? MessageBoxIcon.Error :
        MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        // Error crítico de la aplicación
```

```
                MessageBox.Show("Error inesperado en la interfaz: " +
ex.Message, "Error Crítico", MessageBoxButtons.OK, MessageBoxIcon.Stop);
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using Panaderia.Datos;
using Panaderia.Entidades;

namespace Panaderia.Negocio
{
    // Clase BLL para manejar la tabla Ingredientes
    public class IngredienteBLL
    {
        private IngredienteDAL dal = new IngredienteDAL();

        // READ
        public List<Ingrediente> ListarTodos()
        {
            try
            {
                return dal.ListarIngredientes();
            }
            catch (Exception ex)
            {
                // Registrar el error y devolver lista vacía o lanzar
                excepción de negocio
                Console.WriteLine("Error BLL al listar ingredientes: " +
ex.Message);
                return new List<Ingrediente>();
            }
        }

        // CREATE/UPDATE
        public ResultadoTransaccion Guardar(Ingrediente ingrediente)
        {
            // Validaciones de Negocio
```

```

        if (string.IsNullOrWhiteSpace(ingrediente.Nombre) ||
string.IsNullOrWhiteSpace(ingrediente.UnidadMedida))
    {
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
"El nombre y la unidad de medida son obligatorios." };
    }
    if (ingrediente.Cantidad < 0)
    {
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
"La cantidad no puede ser negativa." };
    }
    // Agrega más validaciones aquí (ej: unicidad del nombre,
formato de costo, etc.)

    try
    {
        if (ingrediente.IdIngrediente == 0)
        {
            // Insertar
            return dal.InsertarIngrediente(ingrediente);
        }
        else
        {
            // Actualizar
            return dal.ActualizarIngrediente(ingrediente);
        }
    }
    catch (Exception ex)
    {
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
"Error BLL al guardar ingrediente: " + ex.Message };
    }
}

// DELETE
public ResultadoTransaccion Eliminar(int idIngrediente)
{
    if (idIngrediente <= 0)
    {
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
"ID de ingrediente no válido." };
    }
    return dal.EliminarIngrediente(idIngrediente);
}
}

```

```

// Clase BLL para manejar la tabla Recetas
public class RecetaBLL
{
    private RecetaDAL dal = new RecetaDAL();

    // CREATE: Insertar Receta Completa (Cabecera y Detalle)
    public ResultadoTransaccion CrearReceta(Receta receta)
    {
        // 1. Validaciones de Negocio
        if (receta.IdProducto <= 0)
        {
            return new ResultadoTransaccion { Resultado = 0, Mensaje =
"Debe seleccionar un producto válido para la receta." };
        }
        if (receta.Ingredientes == null || receta.Ingredientes.Count ==
0)
        {
            return new ResultadoTransaccion { Resultado = 0, Mensaje =
"La receta debe contener al menos un ingrediente." };
        }

        // Validar que no haya ingredientes duplicados en la lista
        temporal
        if (receta.Ingredientes.GroupBy(d => d.IdIngrediente).Any(g =>
g.Count() > 1))
        {
            return new ResultadoTransaccion { Resultado = 0, Mensaje =
"Un ingrediente no puede estar duplicado en la misma receta." };
        }

        // Validar cantidades positivas en el detalle
        if (receta.Ingredientes.Any(d => d.CantidadRequerida <= 0))
        {
            return new ResultadoTransaccion { Resultado = 0, Mensaje =
"Todas las cantidades requeridas deben ser positivas." };
        }

        // 2. Llamada a la DAL (La DAL se encarga de la transacción SQL)
        try
        {
            return dal.InsertarReceta(receta);
        }
        catch (Exception ex)
        {

```

```
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
    "Error BLL al crear la receta: " + ex.Message };
    }
}

// READ: Obtener Receta por Producto
public Receta ObtenerRecetaPorProducto(int idProducto)
{
    if (idProducto <= 0)
    {
        return null;
    }
    try
    {
        return dal.ObtenerRecetaPorProducto(idProducto);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error BLL al obtener receta: " +
ex.Message);
        return null;
    }
}

// DELETE
public ResultadoTransaccion EliminarReceta(int idProducto)
{
    if (idProducto <= 0)
    {
        return new ResultadoTransaccion { Resultado = 0, Mensaje =
    "ID de producto no válido." };
    }
    return dal.EliminarReceta(idProducto);
}
}
```

ADMIN ROL

```
// Usando un enfoque conceptual para un entorno .NET o Windows Forms
using System;
using System.Data; // Asume que se usa ADO.NET o Entity Framework para
interactuar con la BD

// CLASE CONCEPTUAL PARA LA GESTIÓN DE USUARIOS
public class GestorDeUsuarios
{
    private readonly BaseDeDatos _db; // Simulación de la conexión a la Base
de Datos

    public GestorDeUsuarios(BaseDeDatos db)
    {
        _db = db;
    }

    // =====
    // LÓGICA DE REGISTRO DE NUEVOS CLIENTES
    // =====
    public bool RegistrarNuevoUsuario(string email, string password)
    {
        // 1. Validar que el email no esté ya registrado
        if (_db.UsuarioExiste(email))
        {
            Console.WriteLine("El email ya está en uso.");
            return false;
        }

        // 2. Hashear la contraseña (¡SIEMPRE!)
        string passwordHash = PasswordHelper.HashPassword(password);

        // 3. Crear el nuevo usuario y asignar el ROL FIJO de "Cliente"
        // Este es el punto CLAVE: No se le pregunta el rol al usuario, se
asigna
        var nuevoUsuario = new Usuario
        {
            Email = email,
            PasswordHash = passwordHash,
            Rol = "Cliente" // <-- ¡AQUÍ SE DEFINE EN EL CÓDIGO!
        };

        // 4. Guardar el nuevo usuario en la Base de Datos
        _db.GuardarUsuario(nuevoUsuario);
    }
}
```

```
        Console.WriteLine($"Usuario {email} registrado con éxito como:  
Cliente");  
        return true;  
    }  
  
    // ======  
    // LÓGICA DE INICIO DE SESIÓN (LOGIN)  
    // ======  
    public string IniciarSesion(string email, string password)  
    {  
        // 1. Obtener el usuario de la BD por email  
        var usuario = _db.ObtenerUsuarioPorEmail(email);  
  
        if (usuario == null || !PasswordHelper.VerifyPassword(password,  
usuario.PasswordHash))  
        {  
            Console.WriteLine("Credenciales incorrectas.");  
            return "ERROR";  
        }  
  
        // 2. AUTORIZACIÓN: Devolver el rol para que el programa sepa qué  
mostrar  
        // Si el email fue admin@familybakery.com, devolverá "Admin".  
        // Para cualquier otro email, devolverá "Cliente".  
        return usuario.Rol; // <-- Se extrae el dato guardado en la BD  
    }  
}  
  
// Modelos y Helpers simulados (Necesitarías una implementación real para BD  
y Hash)  
public class Usuario  
{  
    public string Email { get; set; }  
    public string PasswordHash { get; set; }  
    public string Rol { get; set; }  
}  
  
public class BaseDeDatos // Simulación  
{  
    public bool UsuarioExiste(string email) => false;  
    public void GuardarUsuario(Usuario user) { }  
    public Usuario ObtenerUsuarioPorEmail(string email)  
    {  
        // Simula la obtención del administrador  
        if (email == "admin@familybakery.com")
```

```
        return new Usuario { Email = email, PasswordHash = "hash_fijo",
Rol = "Admin" };

        // Simula la obtención de un cliente
        return new Usuario { Email = email, PasswordHash = "hash_cliente",
Rol = "Cliente" };
    }
}

public static class PasswordHelper // Simulación
{
    public static string HashPassword(string password) =>
$"hashed_{password}";
    public static bool VerifyPassword(string inputPassword, string
storedHash) => true;
}

// Ejemplo de uso
/*
var db = new BaseDeDatos();
var gestor = new GestorDeUsuarios(db);

// Nuevo registro (automáticamente "Cliente")
gestor.RegistrarNuevoUsuario("pepito@correo.com", "pass123");

// Login del administrador (obtiene "Admin" de la BD)
string rolAdmin = gestor.IniciarSesion("admin@familybakery.com",
"miContrasenaSuperSegura");
// rolAdmin contendrá "Admin"

// Login del nuevo cliente (obtiene "Cliente" de la BD)
string rolCliente = gestor.IniciarSesion("pepito@correo.com", "pass123");
// rolCliente contendrá "Cliente"
*/
```