

Fábrica de Helados y Tortas Heladas

Aguilera.Kevin.2A

Fabrica de Helados y Tortas Heladas

Proceso del Fabricado

PEDIDOS EN PREPARACIÓN

	Numero de pedido	empleado	Monto Total
*			

PEDIDOS FINALIZADOS

	Numero de pedido	empleado	Monto Total
*			

NUEVO PEDIDO

Nombre Empleado:

Elegir producto:

Tamaño:

Seleccionar gustos:

- ☐ Granizado
- ☐ Americana
- ☐ Chocolate
- ☒ Tramontana
- ☐ Mascarpone
- ☐ Frutilla
- ☐ Vainilla

CARRITO DE PRODUCCIÓN

	Numero Pedido	Tamaño	Tipo	Precio	Sabor1
*					

Total 0

IMPORTANTE: Conexión base de datos: @"Data Source=localhost\SQLEXPRESS.
(Entidades - ControlSql).

Mi fábrica consiste en la fabricación de potes de helado y/o tortas heladas.

El **proceso del fabricado** comienza cuando se efectúa el pedido por un empleado encargado de añadir lo que se va a fabricar, el carrito con los datos se envía al datagrid "Pedidos En Preparación" y luego de unos segundos(**cinco**) pasa del estado "En Preparación" a "Pedido Finalizado".

Para efectuar un pedido e iniciar su fabricación, todos los campos son obligatorios, caso contrario, el sistema no permitirá generarlo.

Se debe ingresar en orden de la siguiente manera:

Nombre Empleado: Persona encargada de añadir lo que se va a fabricar. (quien maneja el sistema)

Elegir producto: Se podrá elegir entre HELADO o TORTA HELADA.

Tamaño: Si es TORTAHELADA, se podrá elegir el tamaño individual o entero, si es HELADO, puede elegirse Pote20kilos, Pote30kilos, Pote45kilos o Pote60kilos.

Selección de gustos:

Torta Individual – 1 gusto máximo.

Torta Entera – 2 gustos máximo.

Pote 20 Kilos – 2 gustos máximo.

Pote 30 Kilos y 45 Kilos – 3 gustos máximo.

Pote 60 Kilos – 4 gustos máximo.

Botón “Agregar al Carrito”: Al hacer click, se cargará el producto al carrito.

Botón “Cancelar”: Si el empleado se arrepiente, con este botón se eliminará y se limpiarán los campos del pedido.

• TEMA 15 (Excepciones)

Seleccionar gustos: Si la cantidad de gustos supera a la permitida, se lanzará una excepción (Clase - Excepciones), pasará lo mismo si ocurre algún error en generar documentos TXT/XML en la sección (Clase – Archivos).

```
public class ArchivosException : Exception
{
    6 references
    public ArchivosException(Exception innerException) : base("ocurrio un error", innerException)
    {
    }

    0 references
    public ArchivosException() : base("ocurrio un error")
    {
    }

    0 references
    public ArchivosException(string message) : base(message)
    {
    }
}
```

```
public class GustosException : Exception
{
    2 references
    public GustosException() :base("Demasiados gustos para el tipo de producto")
    {
    }

    2 references
    public GustosException(string message) :base(message)
    {
    }
}
```

• TEMA 16 (Test Unitarios)

Se testea el funcionamiento de GustosException, verificación del guardado TXT y validación de la lista de productos (Clase – Test Unitarios).

```
public class Excepciones
{
    [TestMethod]
    [ExpectedException(typeof(GustosException))]
    public void GustosException()
    {
        //arrange
        Carrito miCarrito = new Carrito("Luis");

        //act
        List<Producto.GustoHelado> gustos = new List<Producto.GustoHelado>();
        gustos.Add(Producto.GustoHelado.frutilla);
        gustos.Add(Producto.GustoHelado.Granizado);

        //assert
        Torta productoTorta = new Torta();
        productoTorta.TamanoElegido = Torta.Tamano.Individual;
        productoTorta += Producto.GustoHelado.Americana;
        productoTorta += Producto.GustoHelado.Chocolate;
        productoTorta += Producto.GustoHelado.frutilla;

        miCarrito += productoTorta;
        miCarrito.CalcularTotal();

        Carrito.Guardar(miCarrito);
    }
}
```

```
[TestMethod]
References
public void VerificarTxt()
{
    Carrito v = new Carrito("Jorge");
    bool ret = Carrito.Guardar(v);

    Assert.IsTrue(ret);
}

[TestClass]
References
public class TestPruebasEntidades
{
    [TestMethod]
    References
    public void ValidarListaProducto()
    {
        Pedido pedido = new Pedido();
        Assert.IsNotNull(pedido.PedidosEnPreparacion);
    }
}
```

• **TEMA 17 (Tipos Genéricos), TEMA 18 (Interfaces), TEMA 19 (Archivos y Serialización)**

Botón “Iniciar Fabricación”: Se guardarán todos los pedidos con todos sus datos en Carrito.txt (en el cual se aplica tipos genéricos e interfaces) (Clase – Archivos – IArchivo/Texto). También se registra el nombre de empleado y numero de pedido del último en usar el sistema en Carrito.xml.

Locación: FabricaDeHeladosyTortasHeladas/bin/Debug.

```
public interface IArchivo <T>
{
    /// <summary>
    /// Metodo de tipo Interfaz para guardar un archivo
    /// </summary>
    /// <param name="archivo">ruta del archivo</param>
    /// <param name="datos">datos del archivo</param>
    /// <returns></returns>
    4 references
    bool Guardar(string archivo, T datos);

    /// <summary>
    /// Metodo de tipo Interfaz para leer un archivo
    /// </summary>
    /// <param name="archivo">ruta del archivo</param>
    /// <param name="datos">se pasaran por referencia los datos del archivo</param>
    /// <returns></returns>
    3 references
    bool Leer(string archivo, out T datos);
}
```

```
public class Texto : IArchivo<string>
{
    /// <summary>
    /// Guarda un archivo con extension .txt
    /// </summary>
    /// <param name="archivo">ruta del archivo</param>
    /// <param name="datos">datos a guardar</param>
    /// <returns>retorna true si se pudo guardar, caso contrario retorna false</returns>
    2 references
    public bool Guardar(string archivo, string datos)
    {
        bool guardado = false;
        StreamWriter writer = null;
        try
        {
            writer = new StreamWriter($"{archivo}.txt",true);
            writer.WriteLine(datos);
            guardado = true;
        }
        catch (Exception ex)
        {
            throw new ArchivosException(ex);
        }
        finally
        {
            if(writer!=null)
            {
                writer.Close();
            }
        }
        return guardado;
    }
}
```

```

public class Xml<T> : Iarchivo<T>
{
    /// <summary>
    /// Guarda un archivo con extension .xml
    /// </summary>
    /// <param name="archivo">ruta del archivo</param>
    /// <param name="datos">se devolverán los datos leídos del archivo</param>
    /// <returns>retorna true si se pudo guardar, caso contrario retorna false</returns>
    2 references
    public bool Guardar (string archivo, T datos)
    {
        bool guardado = false;
        XmlTextWriter textWriter=null;
        XmlSerializer serializer;
        try
        {
            textWriter = new XmlTextWriter($"{archivo}.xml", Encoding.UTF8);
            serializer = new XmlSerializer(typeof(T));
            serializer.Serialize(textWriter, datos);
            guardado = true;
        }
        catch (Exception ex)
        {
            throw new ArchivosException(ex);
        }
        finally
        {
            if(textWriter!=null)
            {
                textWriter.Close();
            }
        }

        return guardado;
    }
}

```

```

public static bool Guardar(Carrito carrito)
{
    Texto texto = null;

    try
    {
        texto = new Texto();

        texto.Guardar("Carrito", carrito.ToString());
    }
    catch (Exception ex)
    {
        throw new ArchivosException(ex);
    }

    return true;
}
1 reference
public static bool GuardarXml(Carrito carrito)
{
    try
    {
        Xml<Carrito> xml = new Xml<Carrito>();
        xml.Guardar("Carrito", carrito);
    }
    catch (Exception ex)
    {
        throw new ArchivosException(ex);
    }

    return true;
}

```

• TEMA 21 y 22 (SQL y Bases de datos)

Se implementa en Entidades - ControlSql. Al iniciar la fabricación se almacenará en la base de datos (id, numero de pedido, nombre empleado y costo de fabricación)

Conexión: @"Data Source=localhost\SQLEXPRESS

```
/// <summary>
/// Conexión con la base de datos "TP4"
/// </summary>
References
static ControlSql()
{
    conexionBD = new SqlConnection(@"Data Source=localhost\SQLEXPRESS; Initial Catalog = TP4; integrated security = true");
}

/// <summary>
/// Guardo el carrito en mi base de datos
/// </summary>
/// <param name="carrito">carrito a guardar</param>
1 reference
public static void SetCarrito(Carrito carrito)
{
    SqlCommand command = new SqlCommand();
    command.Connection = conexionBD;

    StringBuilder sb = new StringBuilder();
    sb.AppendFormat("INSERT INTO Fabrica (numeroPedido, empleado, total) VALUES ('{0}', '{1}', '{2}']", carrito.NumeroDePedido, carrito.NombreEmpleado, carrito.Total);
    command.CommandText = sb.ToString();
    conexionBD.Open();

    command.ExecuteNonQuery();
    conexionBD.Close();
}
```

	id	numeroPedido	empleado	total
1	1160	1	pepita	4800
2	1161	1	jorge	120
3	1162	2	martin	2000
4	1163	3	kevin	6200
5	1164	1	pepe	400
6	1165	2	pepa	120
7	1166	1	AFAFAFA	4800
8	1167	1	melon	3200

• TEMA 23 (Hilos)

Con el **Botón “Iniciar Fabricación”** también se inicializarán los threads(Entidades - Pedido), el cual pasará el pedido del datagrid “En preparación” a “Finalizados”.

```
/// <summary>
/// Operador que agrega un carrito al pedido, ademas inicia un hilo TEMA 23
/// </summary>
/// <param name="pedido">pedido</param>
/// <param name="carrito">carrito</param>
/// <returns></returns>
1 reference
public static Pedido operator +(Pedido pedido, Carrito carrito)
{
    if (carrito != null)
    {
        Thread hilo = new Thread(pedido.CarritoAceptado);
        pedidosEnPreparacion.Enqueue(carrito);
        Pedido.listaHilos.Add(hilo);
        hilo.Start();
    }

    return pedido;
}
```

Al cerrar el Form se terminan los hilos abiertos.

```
/// <summary>
/// Cierra todos los hilos abiertos
/// </summary>
1 reference
public static void CerrarHilos()
{
    foreach (Thread hilo in listaHilos)
    {
        hilo.Abort();
    }
}
```

• TEMA 24 (Eventos)

Se implementa en Entidades – Pedido.

```
public event Preparacion PreparandoPedido;
```

```
/// <summary>
/// Declaracion del delegado
/// </summary>
/// <param name="carrito">carrito</param>
public delegate void Preparacion(Carrito carrito);
```

```

/// <summary>
/// Metodo que invoca el evento TEMA 24
/// </summary>
1 reference
public void CarritoAceptado()
{
    Carrito carritoAux = new Carrito();
    while ((this.PedidosEnPreparacion.Count >= 1) == true)
    {
        //espero 5 segundos
        Thread.Sleep(new Random().Next(5000, 7000));

        //elimino el pedido en preparacion y agrego el pedido a mi lista finalizada
        carritoAux = PedidosEnPreparacion.Peek();

        PedidosFinalizados.Add(carritoAux);
        PedidosEnPreparacion.Dequeue();

        //invoco a mi metodo para agregar el dato a dgv terminado
        this.PreparandoPedido.Invoke(carritoAux);
    }
}

```

```

/// <summary>
/// Agrego el carrito a dgvEnPreparacion
/// </summary>
/// <param name="carrito">carrito que pasa a ser un pedido en preparacion</param>
1 reference
public void CargarPedido(Carrito carrito)
{
    if (dgvCarrito.InvokeRequired)
    {
        Preparacion delegado = new Preparacion(CargarPedido);
        this.Invoke(delegado, new object[] { carrito });
    }
    else
    {
        Pedido.CargoPedido(carrito);
        int posiciones = dgvEnPreparacion.Rows.Add();
        dgvEnPreparacion[0, posiciones].Value = carrito.NumeroDePedido.ToString();
        dgvEnPreparacion[1, posiciones].Value = carrito.NombreEmpleado;
        dgvEnPreparacion[2, posiciones].Value = carrito.Total;

        this.FinalizarPedido(carrito);
    }
}

```



```

/// <summary>
/// Elimina el pedido del gvEnPreparacion y lo pasa a dgvFinalizados
/// </summary>
/// <param name="carrito">pedido para finalizar</param>
3 references
public void FinalizarPedido(Carrito carrito)
{
    if (dgvCarrito.InvokeRequired)
    {
        Preparacion delegado = new Preparacion(FinalizarPedido);
        this.Invoke(delegado, new object[] { carrito });
    }
    else
    {
        if (dgvEnPreparacion.Rows.Count > 1)
        {
            dgvEnPreparacion.Rows.RemoveAt(0);
        }

        ActualizarPedidosFinalizados();
        MessageBox.Show("Fabricación finalizada! Muchas Gracias");
    }
}

```

• **TEMA 25 (Métodos de Extensión)**

En la selección de gustos se utilizan métodos de extensión (Entidades – Extensiones) para validar la cantidad de gustos máxima según el tipo de producto y su tamaño.

Torta Individual – 1 gusto máximo.

Torta Entera – 2 gustos máximo.

Pote 20 Kilos – 2 gustos máximo.

Pote 30 Kilos y 45 Kilos – 3 gustos máximo.

Pote 60 Kilos – 4 gustos máximo.

```

/// <summary>
/// Extension que devuelve la cantidad maxima de gustos para un tamaño de Torta
/// </summary>
/// <param name="tamano">tamaño de la torta a evaluar</param>
/// <returns>la cantidad maxima de gustos</returns>
2references
public static int CantidadDeGustosPermitidos(this Torta.Tamano tamano)
{
    int cantidad = 0;
    switch (tamano)
    {
        case Torta.Tamano.Individual:
            cantidad = 1;
            break;
        case Torta.Tamano.Entero:
            cantidad = 2;
            break;
        default:
            break;
    }

    return cantidad;
}

```

```

/// <summary>
/// Extension que devuelve la cantidad maxima de gustos para un tamaño de helado
/// </summary>
/// <param name="tamano">tamaño del helado a evaluar</param>
/// <returns>la cantidad maxima de gustos</returns>
2references
public static int CantidadDeGustosPermitidos(this Helado.Tamano tamano)
{
    int cantidad = 0;

    switch (tamano)
    {
        case Helado.Tamano.Pote20Kilos:
            cantidad = 2;
            break;
        case Helado.Tamano.Pote30Kilos:
        case Helado.Tamano.Pote45Kilos:
            cantidad = 3;
            break;
        case Helado.Tamano.Pote60Kilos:
            cantidad = 4;
            break;
        default:
            break;
    }

    return cantidad;
}

```