PANADERIA

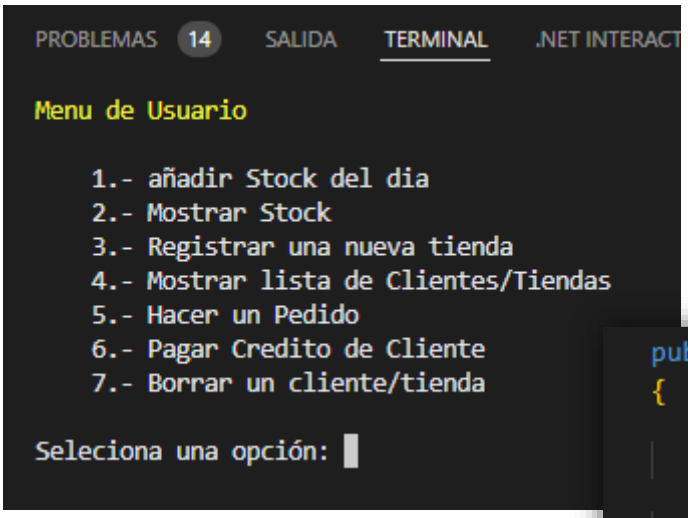
DESCRIPCION:

Un panadero necesita un programa que le puede ayudar en gestionar la panadería mejor, entonces tenemos que crear un programa que puede ayudar le máximum posible en la tienda. las cosas más importantes en una panadería son los pedidos y los pagos.

He Creado esta Programa que ayuda el panadero en gestionar los clientes de la panadería y sobre todo los pagos o los créditos que están registrados en un fichero CSV y en el mismo tiempo gestiona el stock de cada día.

A la hora de arrancar el programa se muestra una lista con varias opciones o si queremos decir (acciones).

Programa:



la primera opción es para añadir el stock de pan de cada por ejemplo (Se puede modificar en cualquier tiempo).

se guarda el stock en el objeto

Panadería que tiene Stock y precio

como parámetros.

```
public class Panderia
{
    5 references
    public int StockDelDia { get; set; }
    2 references
    public double PrecioDepan { get; set; }

    1 reference
    public Panderia(int stockDelDia)
    {
        StockDelDia = stockDelDia;
        PrecioDepan = 1.20;
    }
```

```
    1.- añadir Stock del dia
    2.- Mostrar Stock
    3.- Registrar una nueva tienda
    4.- Mostrar lista de Clientes/Tiendas
    5.- Hacer un Pedido
    6.- Pagar Credito de Cliente
    7.- Borrar un cliente/tienda

Seleciona una opción: 2

Stock actual: 200 UNIDAD
Pulsa <Return> para continuar
```

la segunda opción es solo para mostrar el stock

la tercera opción es para registrar un Cliente(tienda) en nuestro base de datos lo que es en nuestro caso un fichero CSV.

Se guarda a tienda con un credito de 0 hasta que realiza un pedido

Porque cada vez se crea un nuevo cliente se crea en la clase gestor con un Crédito de 0.

```
Menu de Usuario

    1.- añadir Stock del dia
    2.- Mostrar Stock
    3.- Registrar una nueva tienda
    4.- Mostrar lista de Clientes/Tiendas
    5.- Hacer un Pedido
    6.- Pagar Credito de Cliente
    7.- Borrar un cliente/tienda

Seleciona una opción: 3

Nombre de la tienda: tien23
Nombre del dueño: anass
```

```
4 references
public Panderia panderia { get; set; } = new(0);
```

La cuarta opción muestra la lista de tiendas que esta sacada del fichero CSV

```
Seleciona una opción: 4

Tiendas/Clientes

    1.- Nombre de la tienda: 3  Nombre del dueño: ANASS    Su Credito: 276 EURO
    2.- Nombre de la tienda: ANASS      Nombre del dueño: KBIR     Su Credito: 104 EURO
    3.- Nombre de la tienda: TN2        Nombre del dueño: ANASS    Su Credito: 180 EURO
    4.- Nombre de la tienda: TIEN       Nombre del dueño: MAROUAN          Su Credito: 144 EURO
    5.- Nombre de la tienda: TIE_NDA    Nombre del dueño: ALEX     Su Credito: 0 EURO
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO

Pulsa <Return> para continuar
```

El Controlador llama a la función listar Tiendas.

```
//Mostrar lista de clientes/tiendas
4 references
public void ListarTiendas()
{
    _vista.MostrarListaEnumerada<Tienda>("Tiendas/Clientes", _sistema.clientes);
}
```

Y se muestra con la ayuda de la clase vista la lista de tiendas registradas en el fichero CSV

```
public Gestor(RepoPanaderiaCSV repoPanaderia)
{
    _repoPanaderia = repoPanaderia;
    clientes = _repoPanaderia.Leer();
}
6 references
RepoPanaderiaCSV _repoPanaderia;
14 references
public List<Tienda> clientes { get; set; } = new();
4 references
public Panderia panderia { get; set; } = new(0);

// === Gestion de tiendas ===
1 reference
```

La quinta opción es para hacer un pedido

Se muestra dos opciones Persona y Tienda

Para que podemos hacer los pedidos a un Cliente registrado o una persona individual (No registrado)

```
    1.- añadir Stock del dia
    2.- Mostrar Stock
    3.- Registrar una nueva tienda
    4.- Mostrar lista de Clientes/Tiendas
    5.- Hacer un Pedido
    6.- Pagar Credito de Cliente
    7.- Borrar un cliente/tienda

Seleciona una opción: 5

1- Persona
2- Tienda
Selecciona una opción:
```

```
Selecciona una opción: 2
Mete la Cantidad de pan que va pedir (UNIDAD): 50
Tiendas/Clientes

    1.- Nombre de la tienda: 3  Nombre del dueño: ANASS    Su Credito: 276 EURO
    2.- Nombre de la tienda: ANASS      Nombre del dueño: KBIR     Su Credito: 104 EURO
    3.- Nombre de la tienda: TN2        Nombre del dueño: ANASS    Su Credito: 180 EURO
    4.- Nombre de la tienda: TIEN       Nombre del dueño: MAROUAN          Su Credito: 144 EURO
    5.- Nombre de la tienda: TIE_NDA    Nombre del dueño: ALEX     Su Credito: 0 EURO
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO

Seleccione una tienda: 4
```

Si elegimos Tiendas/Clientes sale el mensaje de cantidad y después la lista de clientes (en este caso es el cuatro)

Se muestra directamente la modificación del crédito de tienda y el stock

```
2- Tienda
Selecciona una opción: 2
Mete la Cantidad de pan que va pedir (UNIDAD): 50
Tiendas/Clientes

    1.- Nombre de la tienda: 3  Nombre del dueño: ANASS    Su Credito: 276 EURO
    2.- Nombre de la tienda: ANASS      Nombre del dueño: KBIR     Su Credito: 104 EURO
    3.- Nombre de la tienda: TN2        Nombre del dueño: ANASS    Su Credito: 180 EURO
    4.- Nombre de la tienda: TIEN       Nombre del dueño: MAROUAN          Su Credito: 144 EURO
    5.- Nombre de la tienda: TIE_NDA    Nombre del dueño: ALEX     Su Credito: 0 EURO
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO

Seleccione una tienda: 4
Credito actual: 204 EURO
Pedido realizado
Stock actual: 150
Pulsa <Return> para continuar
```

Si el stock suficiente sino sale un mensaje de error.

```
try
{   _vista.Mostrar("1- Persona");
    _vista.Mostrar("2- Tienda");
    var opcion = _vista.TryObtenerDatoDeTipo<int>("Selecciona una opción"
    if(opcion == 1 || opcion == 2)
    {
        var cp = _vista.TryObtenerDatoDeTipo<int>("Mete la Cantidad de pa
        if(cp > _sistema.MostrarStock())
        {
            _vista.Mostrar("No hay stock suficiente");
            _vista.Mostrar("Stock actual: " + _sistema.MostrarStock());
            return;
```

```
Selecciona una opción: 1
Mete la Cantidad de pan que va pedir (UNIDAD): 300
No hay stock suficiente
Stock actual: 150
```

```
opcion == 1)
```

Si elegimos la primer opción persona si muestra el mensaje de cantidad querida y se quita directamente del stock

```
if(opcion == 1)
{
    _sistema.RestarCantidad(cp);
}
```

el controlador llama la función Restarcantidad() en la clase sistema.

```
2 references
public void RestarCantidad(int cantidad)
{
    panderia.StockDelDia -= cantidad;
}
```

y se modifica directamente el stock del objeto/modelo panadería

la sexta opción es de pagar Crédito de Cliente

```
Seleciona una opción: 6

Tiendas/Clientes

    1.- Nombre de la tienda: 3  Nombre del dueño: ANASS    Su Credito: 276 EURO
    2.- Nombre de la tienda: ANASS      Nombre del dueño: KBIR     Su Credito: 104 EURO
    3.- Nombre de la tienda: TN2        Nombre del dueño: ANASS    Su Credito: 180 EURO
    4.- Nombre de la tienda: TIEN       Nombre del dueño: MAROUAN          Su Credito: 204 EURO
    5.- Nombre de la tienda: TIE_NDA    Nombre del dueño: ALEX     Su Credito: 0 EURO
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO

Seleccione una tienda: 4
Cuanto va a pagar?: 100
Credito actual: 104 EURO
Pulsa <Return> para continuar
```

Se muestra la lista de tiendas , metemos la cantidad que va pagar

Y muestreamos el nuevo Crédito

```
private void PagarCredito()
{
    try
    {
        ListarTiendas();
        var idc = _vista.TryObtenerValorEnRangoInt(1, _sistema.clientes.Count, "Seleccione una tienda");
        var cliente = _sistema.clientes[idc - 1];

        var cp = _vista.TryObtenerDatoDeTipo<int>("Cuanto va a pagar?");
        _sistema.PagarCredito(cliente, cp);
        _vista.Mostrar("Credito actual: " + _sistema.MostrarCredito(cliente) + " EURO");
    }
    catch (Exception e)
    {
        _vista.Mostrar($"UC: {e.Message}");
    }
}
```

La clase controlador llama a la función pagarcredito() de la clase sistema.

```
// === Pagar credito ===
1 reference
public void PagarCredito(Tienda t, double pago)
{
    t.Credito -= pago;
    _repoPanaderia.Guardar(clientes);
}
```

La última opción es de borrar una tienda

```
Seleciona una opción: 7

Tiendas/Clientes

    1.- Nombre de la tienda: 3  Nombre del dueño: ANASS    Su Credito: 276 EURO
    2.- Nombre de la tienda: ANASS      Nombre del dueño: KBIR     Su Credito: 104 EURO
    3.- Nombre de la tienda: TN2        Nombre del dueño: ANASS    Su Credito: 180 EURO
    4.- Nombre de la tienda: TIEN       Nombre del dueño: MAROUAN          Su Credito: 54 EURO
    5.- Nombre de la tienda: TIE_NDA    Nombre del dueño: ALEX     Su Credito: 0 EURO
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO
```

Si no tiene Crédito se borra sino s muestra un mensaje de error

```
Seleccione una tienda: 4
No se puede borrar tienda con credito
Pulsa <Return> para continuar
```

```
    6.- Nombre de la tienda: tien23     Nombre del dueño: anass    Su Credito: 0 EURO

Seleccione una tienda: 6
se va borrar todos los datos de la tiendaTienda: tien23
```

Diagramas:

Diagrama de casos de usos de Negocio.

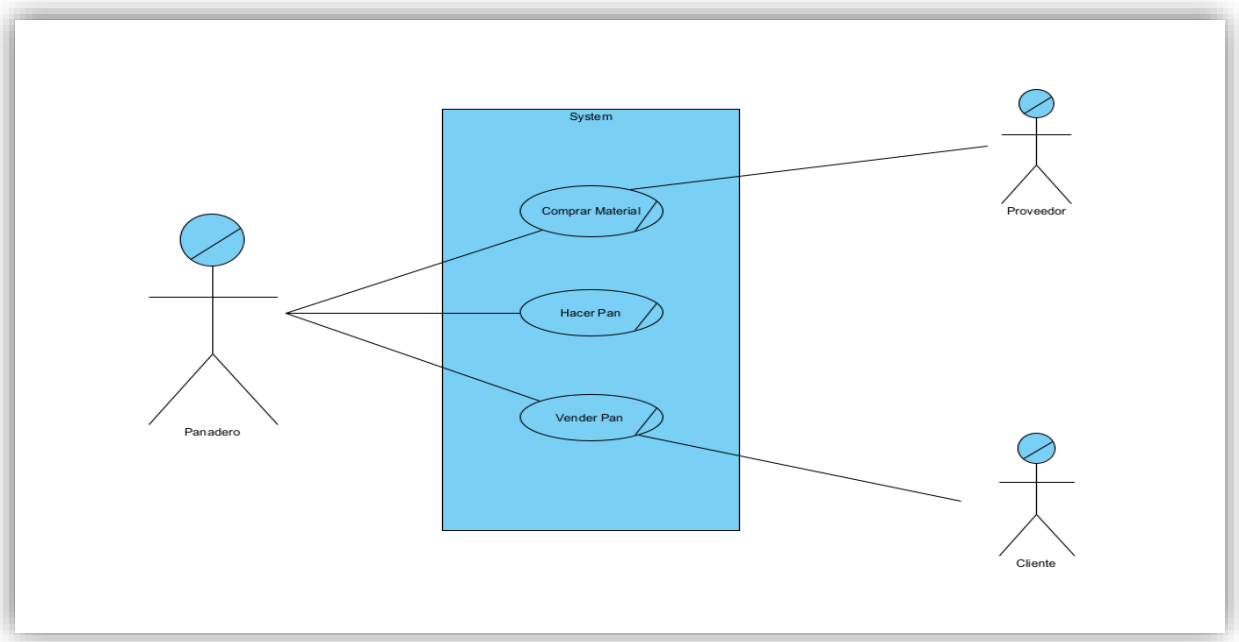
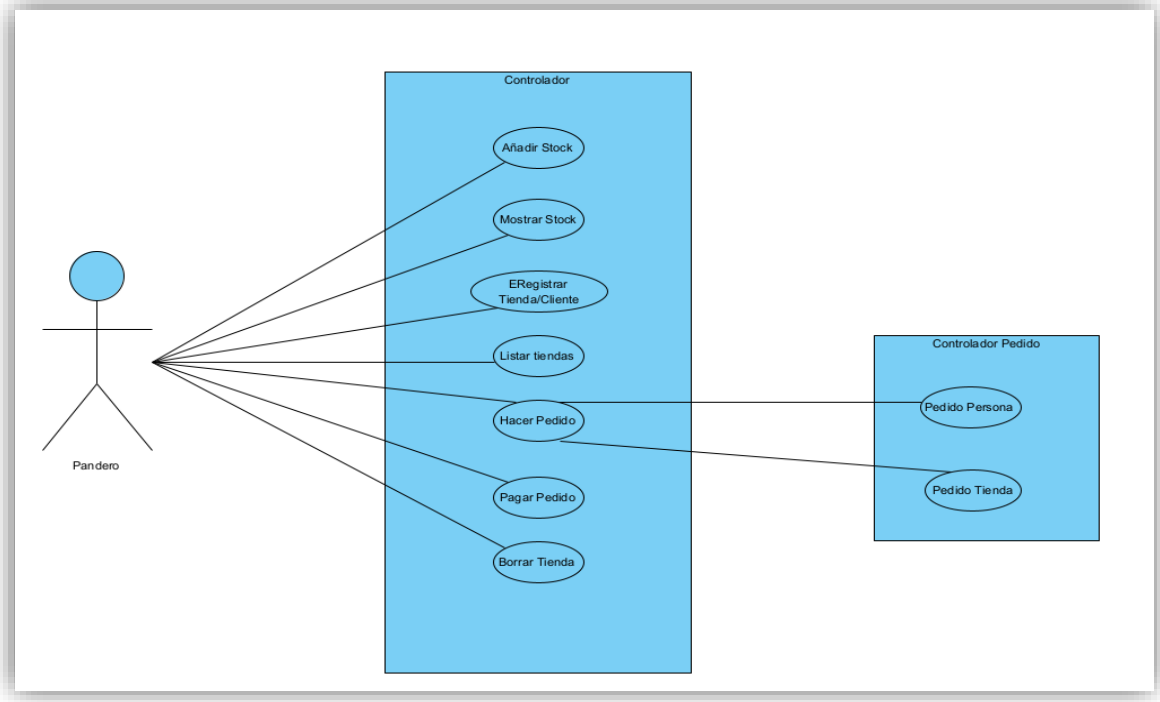


Diagrama de casos de usos de nuestro sistema (muestra el menu principal o las acciones de nuestro programa).



Diagramara de clases de modelos (muestra los modelos y el repositorio que tenemos en nuestro programa).

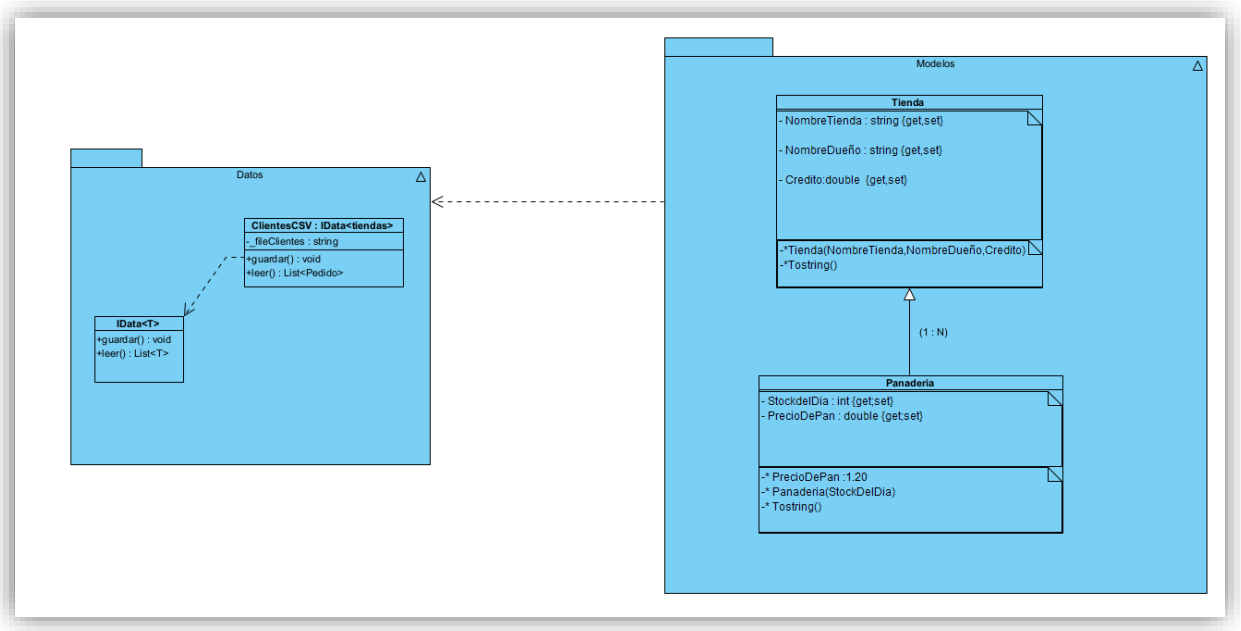
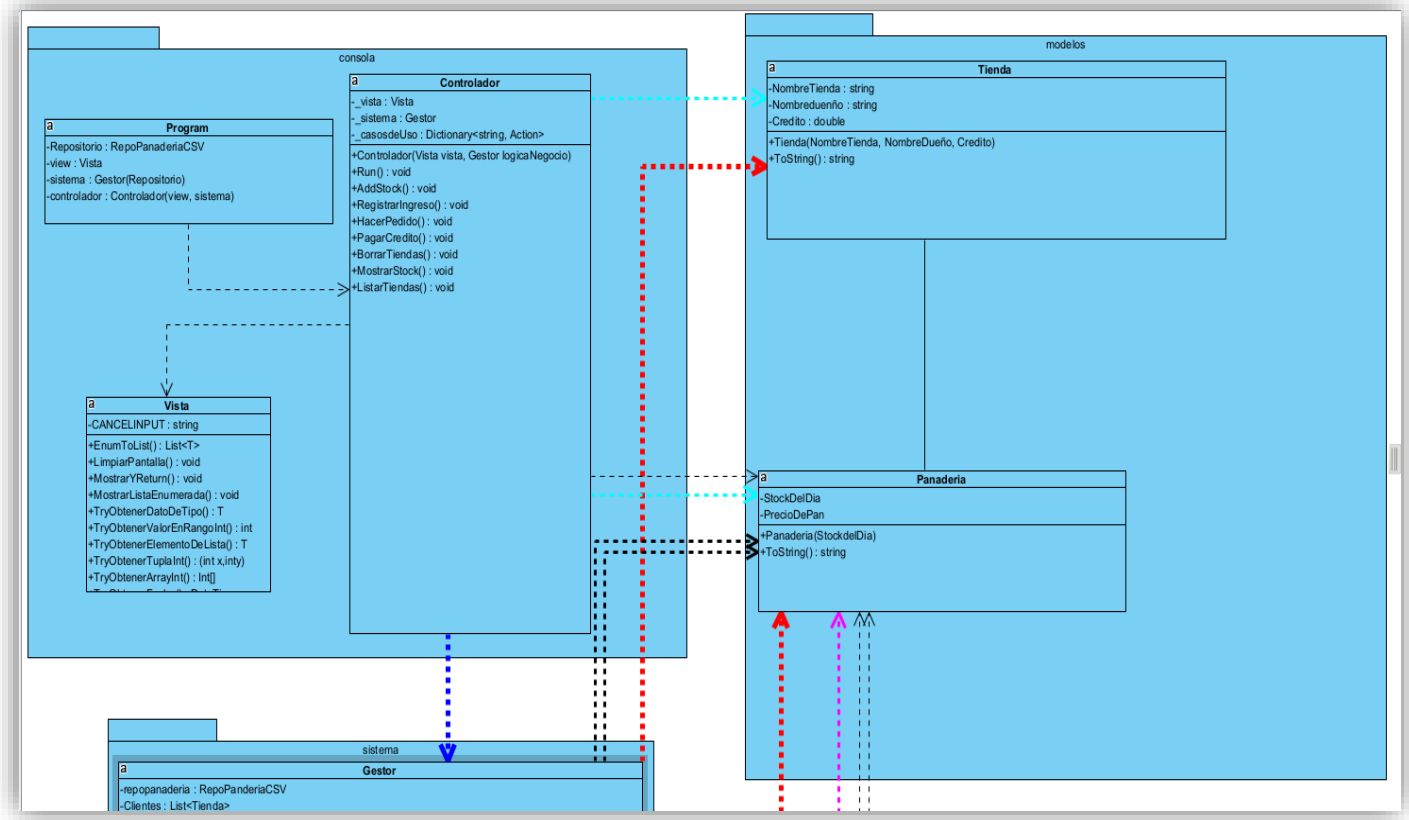


Diagrama de clases de nuestro Arquitectura (muestra todo las clases, modelos, métodos y funciones)



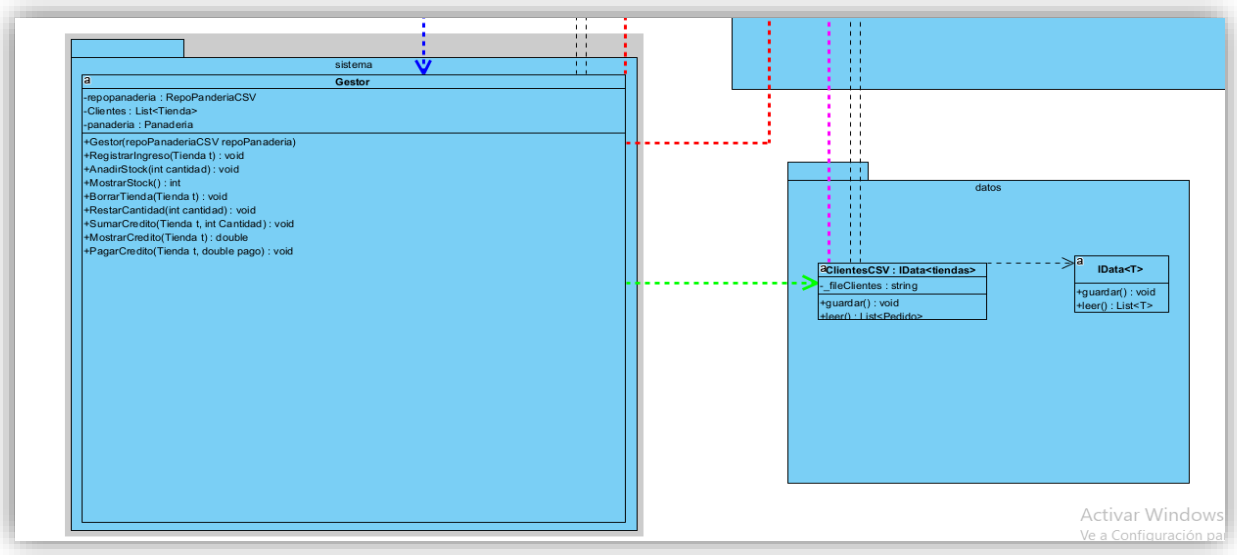


Diagrama de secuencia (presenta un caso de uso lo que es borrar tienda y como maneja los clases este caso desde la clase vista hasta el fichero CSV)

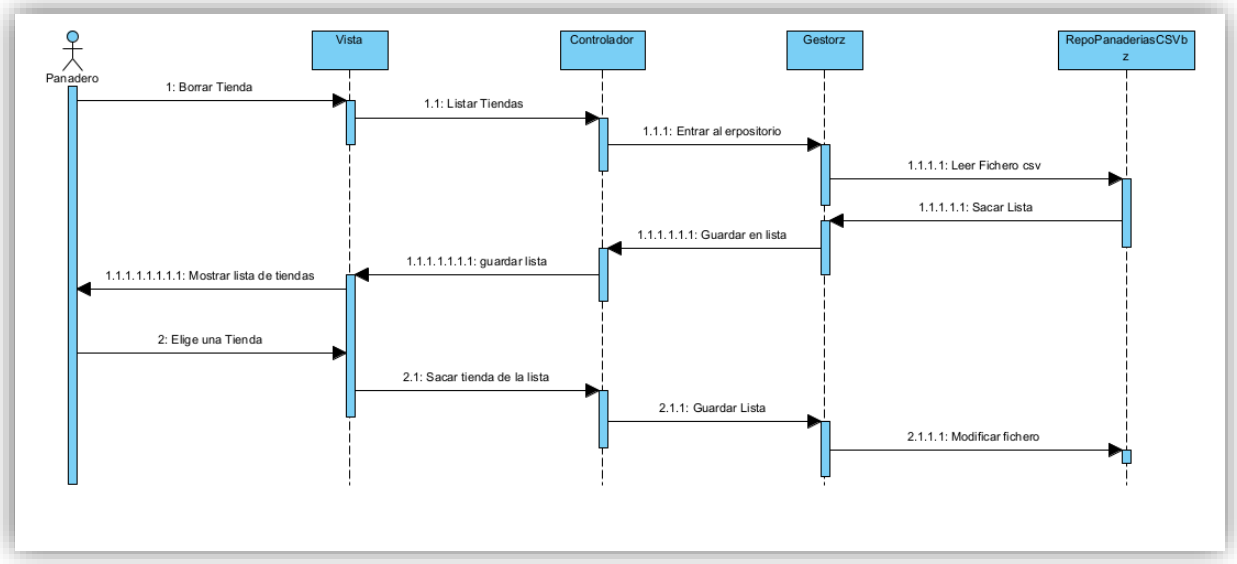


Diagrama de estado de borrar

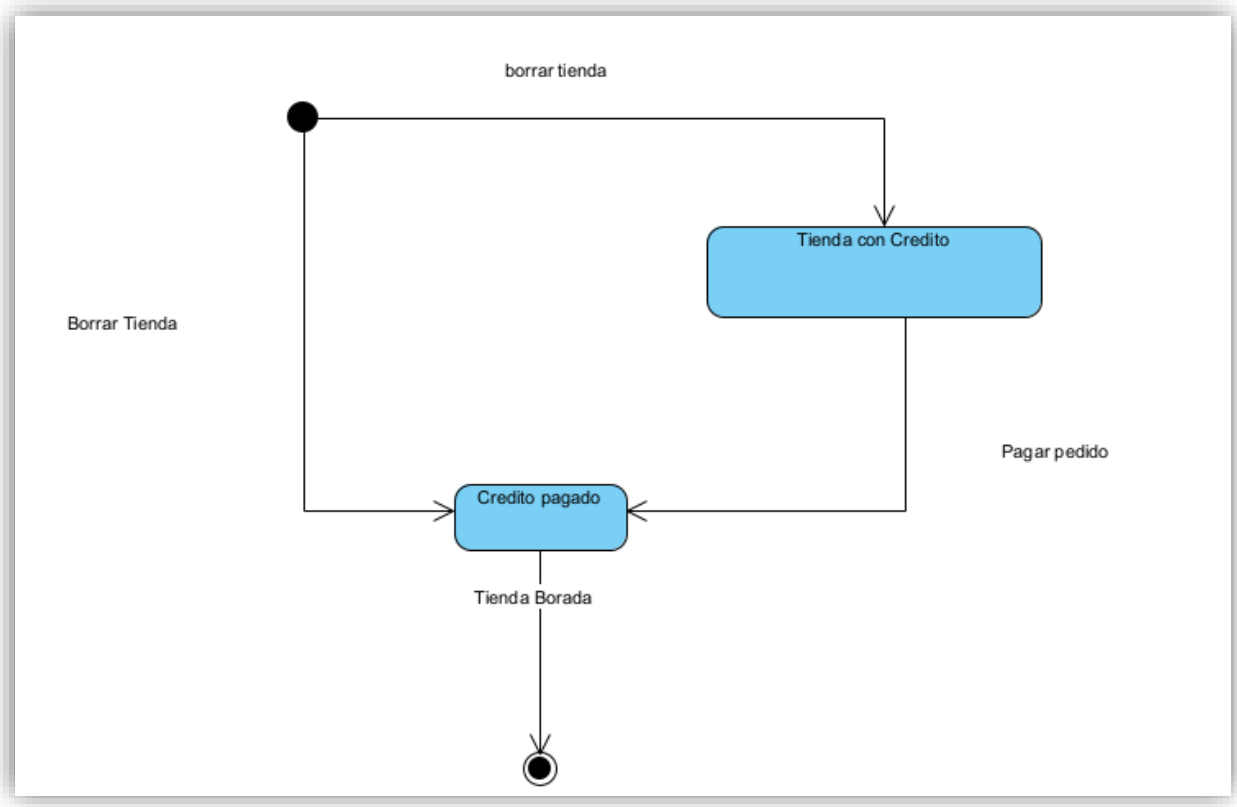
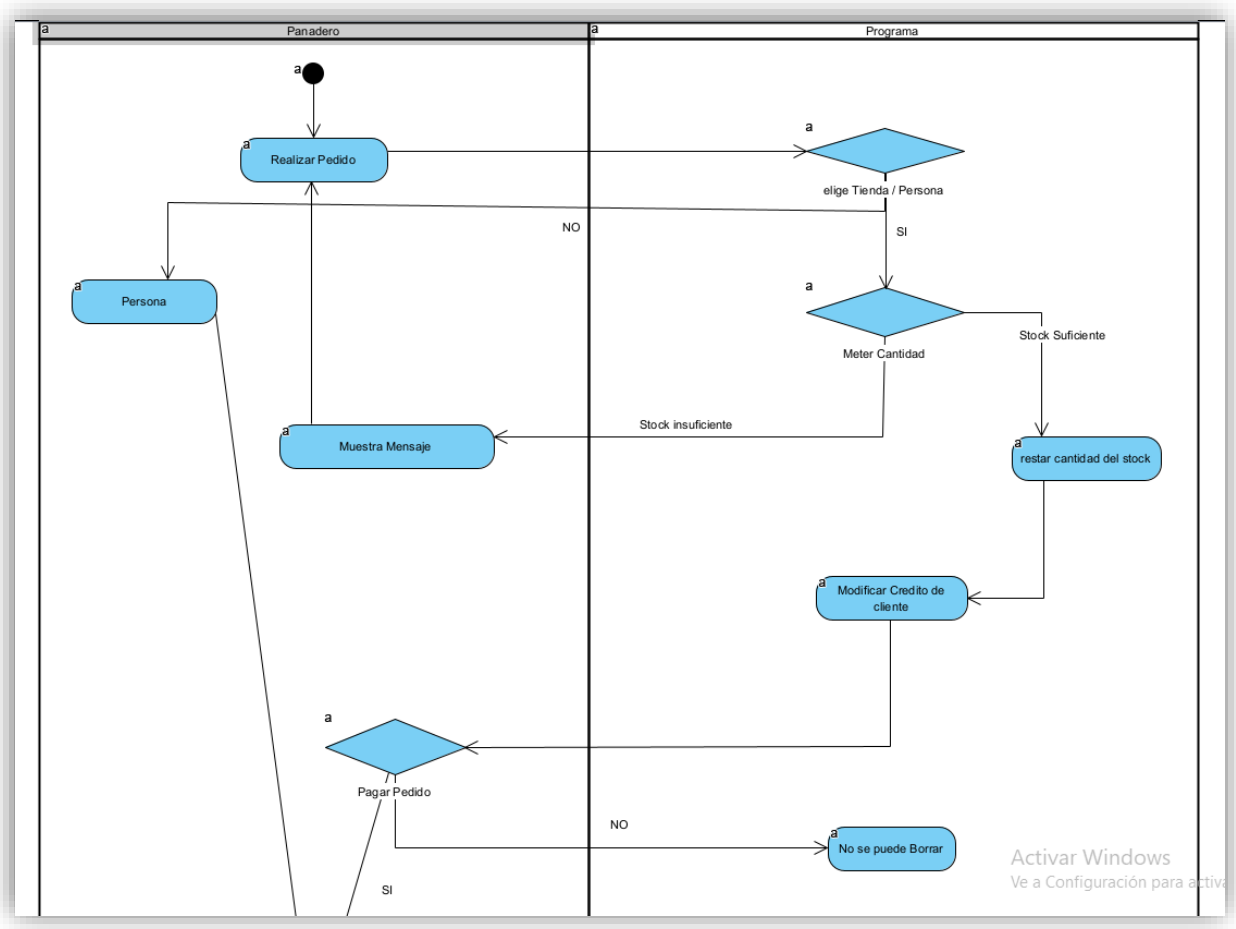
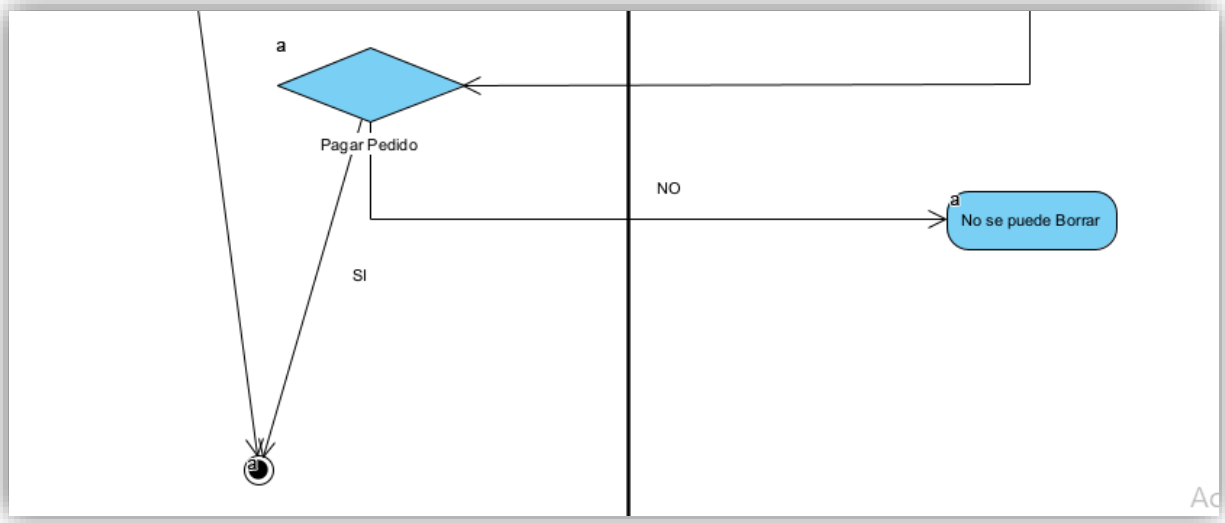


Diagrama de actividad (presenta como funciona el programa al caso de un pedido)




a
Pagar Pedido
NO
a
No se puede Borrar
SI