
	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

## Control de versiones

Versión	Fecha	Autor	Descripción
1.0	31-10-2024	Juan Pérez	Se analiza el problema, se encuentra causa raíz y se propone una solución.

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

## Descripción general

Se identifica una venta con folio **V25752100092966**, esta venta corresponde a la sucursal **2575** y a la caja **21**, la venta se realiza el **13 de octubre del año 2024**, se encuentra que la venta finaliza correctamente ya que es actualizada a su **status V (Válida)**, este **status** se alcanza una vez la venta se finaliza. 11 o

Esta venta se realiza utilizando dos formas de pago, la forma de pago de “**EFFECTIVO (CA)**” con **id 1** y la forma de pago “**TARJETA DE DEBITO ACI (AD)**” con **id 12**.

Se encuentra que la forma de pago “**EFFECTIVO (CA)**” permanece en status **P** aún cuando la venta se finaliza, es decir, la venta llega a **status V**, al llegar a este estado, las formas de pago deberían estar en **status A** si se procesan de manera correcta o **status C** si ocurre algún problema con la forma de pago específica.

Esto se evidencia utilizando la siguiente consulta:


```
DECLARE @folio NVARCHAR(15) = N'V25752100092966';

-- Venta - Cabecero
SELECT
    trn.folioTransaccion,
    trn.status AS statusVenta,
    trn.codigoMovimientoCaja,
    formaPago.descripcion,
    formaPago.codigoFormaPago,
    movDet.estatusFormaPago,
    movDet.importePago
FROM dbo.trnTransaccionesCab trn
    INNER JOIN genMovimientosCajasDet movDet
        ON trn.codigoMovimientoCaja = movDet.codigoMovimientoCaja
    INNER JOIN genFormasDePagoCat formaPago
        ON movDet.codigoFormaPago = formaPago.codigoFormaPago
WHERE trn.folioTransaccion = @folio;
```

[Ver consulta](#)

folioTransaccion	statusVenta	codigoMovimientoCaja	descripcion	codigoFormaPago	estatusFormaPago	importePago
V25752100092966	V	AS219A43-A489-EF11-B7AA-10E7C6374C1A	EFFECTIVO (CA)	1	P	300.0000
V25752100092966	V	AS219A43-A489-EF11-B7AA-10E7C6374C1A	TARJETA DE DEBITO ACI (AD)	12	A	180.0000

[Ver figura](#)

		Incidencia – Venta finalizada con forma de pago pendiente	
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

## Diagnóstico

### Revisión de logs

#### Flujo de venta

Se revisa el flujo que tuvo la venta al momento de ser procesada, para ello se revisa el log llamado [20241013\\_FARMAX\\_TILL\\_SERVER\\_SALE.XLOG](#), este log está ubicado en **%ProgramData%/FAhorro/FARMAX\_TILL\_SERVER** y contiene la traza de los distintos cambios de estado que puede sufrir una venta, aquí se puede observar que la venta sigue un flujo normal, los “...” significan que se omiten algunas líneas hasta llegar a la que se presenta después, se tiene a partir de la **línea 989**:

```
[2024-10-13      14:44:46:759],FARMAX_TILL_SERVER,SALE,INFO,,Venta      ::      Metodo:AddSale
FolioVenta:V25752100092966 Estado:P
```

...

```
[2024-10-13   14:46:09:324],FARMAX_TILL_SERVER,SALE,INFO,,Venta   ::  Metodo:UpdateSaleStatus
FolioVenta:V25752100092966 Estado:N
```

...


```
[2024-10-13  14:46:48:364],FARMAX_TILL_SERVER,SALE,INFO,,Donativo :: Metodo:UpdateSaleHeader
FolioVenta:V25752100092966 Estado:R Monto: IdBeneficiencia: FolioDonativo:
```

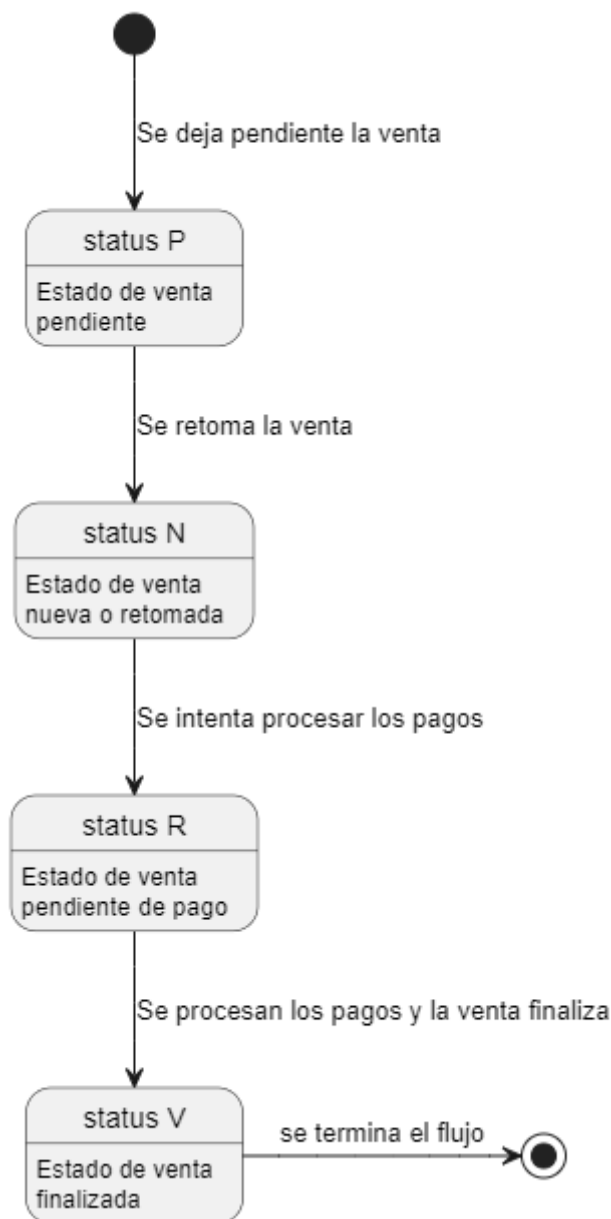
...

```
[2024-10-13
14:47:43:965],FARMAX_TILL_SERVER,SALE,ERROR,[REMOVEPENDINGSALEINFORMATIONEWALLET],Excepción
SaleManager-RemovePendingSaleInformationEwallet. Excepción:El valor no puede ser nulo.
      Nombre      del      parámetro:      entity      Stacktrace:      en
System.Data.Entity.Utilities.Check.NotNull[T] (T value, String parameterName)
      en System.Data.Entity.DbSet`1.Remove(TEntity entity)
      en
FAhorro.POS.Data.Sale.DAO.SalesDao.RemovePendingSaleInformationEwallet(trnTransaccionVentaP
endiente pendingSale, Guid idTransaccion)
      en
FAhorro.POS.BusinessLogic.Sale.SaleManager.RemovePendingSaleInformationEwallet(trnTransacci
onVentaPendiente id, Guid idTransaccion)
[2024-10-13  14:47:43:969],FARMAX_TILL_SERVER,SALE,INFO,,Donativo :: Metodo:UpdateSaleHeader
FolioVenta:V25752100092966 Estado:V Monto: IdBeneficiencia: FolioDonativo:
```

La excepción encontrada en medio del flujo de venta se explica más adelante, después del flujo de venta.


Aquí se observa que el flujo de la venta fue el siguiente:

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



[Ver diagrama](#)

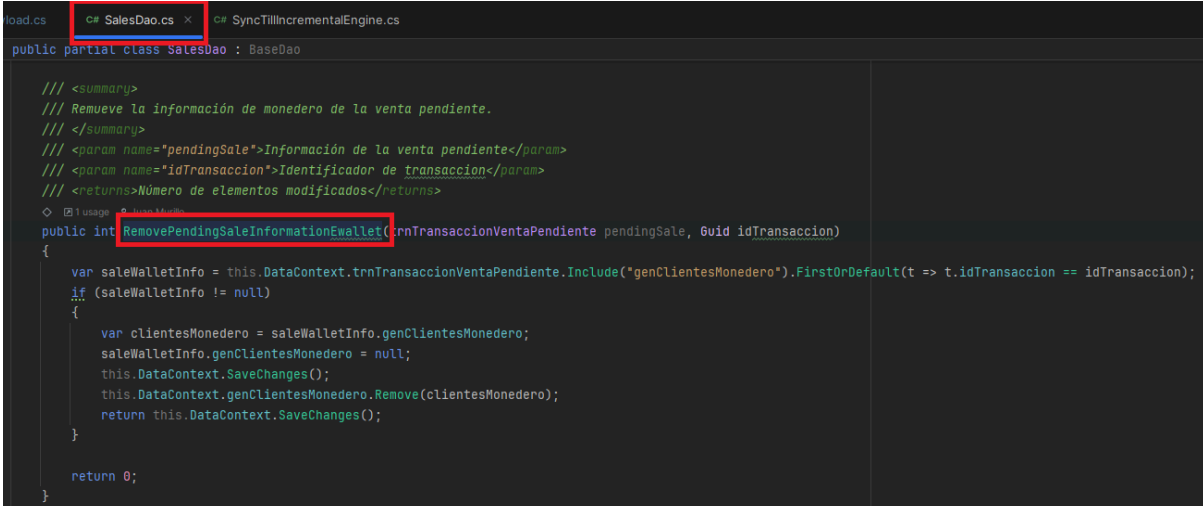
El flujo de la venta **no presenta nada fuera de lo habitual**, es el flujo que normalmente siguen las ventas pendientes al ser retomadas para su finalización, por lo que **no se detecta ninguna anomalía**, al menos con respecto a los cambios de estado de la venta.

<div>  <div> Incidencia – Venta finalizada con forma de pago pendiente </div> </div>			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

Se observa también en el mismo log, a partir de la **línea 1023** que ocurre una excepción en medio del flujo de la venta:

```
[2024-10-13
14:47:43:965], FARMAX_TILL_SERVER, SALE, ERROR, [REMOVEPENDINGSALEINFORMATIONEWALLET], Excepción
SaleManager-RemovePendingSaleInformationEwallet. Excepción:El valor no puede ser
nulo.
Nombre del parámetro: entity Stacktrace: en
System.Data.Entity.Utilities.Check.NotNull[T](T value, String parameterName)
en System.Data.Entity.DbSet`1.Remove(TEntity entity)
en
FAhorro.POS.Data.Sale.DAO.SalesDao.RemovePendingSaleInformationEwallet(trnTransaccionVentaP
endiente pendingSale, Guid idTransaccion)
en
FAhorro.POS.BusinessLogic.Sale.SaleManager.RemovePendingSaleInformationEwallet(trnTransacci
onVentaPendiente id, Guid idTransaccion)
```

Esta excepción es manejada dentro del flujo y no causa una interrupción, el error se da en la solución **POS.Services**, en el proyecto **Win.Services.TillHost**, en el archivo **SalesDao.cs**:



```


// <summary>
// Remueve la información de monedero de la venta pendiente.
// </summary>
// <param name="pendingSale">Información de la venta pendiente</param>
// <param name="idTransaccion">Identificador de transacción</param>
// <returns>Número de elementos modificados</returns>
// </param>
// </returns>
public int RemovePendingSaleInformationEwallet(trnTransaccionVentaPendiente pendingSale, Guid idTransaccion)
{
    var saleWalletInfo = this.DataContext.trnTransaccionVentaPendiente.Include("genClientesMonedero").FirstOrDefault(t => t.idTransaccion == idTransaccion);
    if (saleWalletInfo != null)
    {
        var clientesMonedero = saleWalletInfo.genClientesMonedero;
        saleWalletInfo.genClientesMonedero = null;
        this.DataContext.SaveChanges();
        this.DataContext.genClientesMonedero.Remove(clientesMonedero);
        return this.DataContext.SaveChanges();
    }

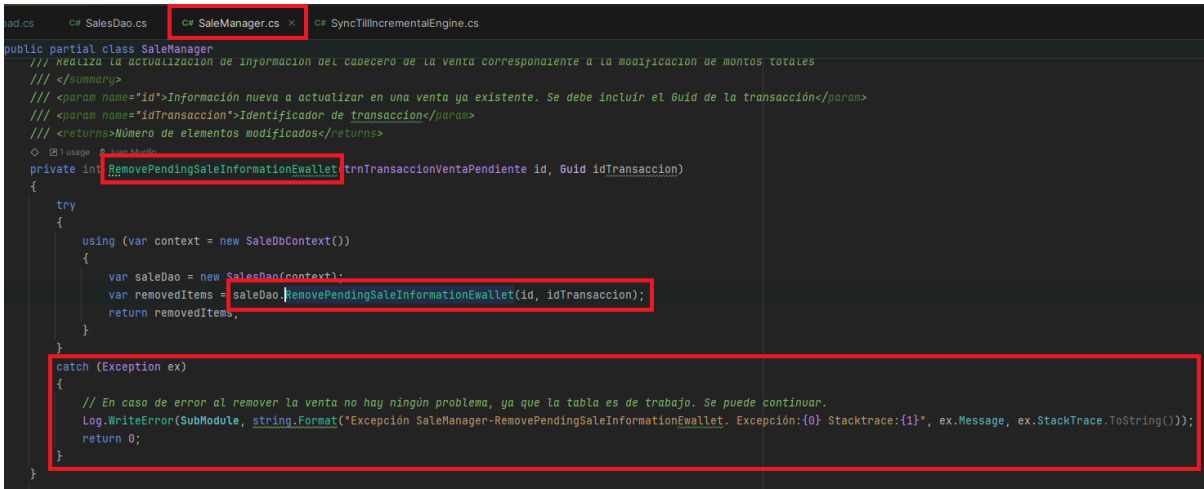
    return 0;
}

```

[Ver figura](#)

Lo que hace este método es borrar el registro de **genClientesMonedero** relacionado a una venta pendiente, este solamente se llama en **SaleManager.cs**, en el método **RemovePendingSaleInformationEwallet(...)**, las excepciones que puedan ocurrir son manejadas dentro del mismo método por lo que cualquier flujo externo que lo utilice no se ve interrumpido por las posibles excepciones:

		Incidencia – Venta finalizada con forma de pago pendiente	
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



```

public partial class SaleManager
{
    /// 
    /// Realiza la actualización de información del cabecero de la venta correspondiente a la modificación de montos totales
    /// 
    /// Información nueva a actualizar en una venta ya existente. Se debe incluir el Guid de la transacción
    /// Identificador de transacción
    /// Número de elementos modificados
    private int RemovePendingSaleInformationEwallet(trnTransaccionVentaPendiente id, Guid idTransaccion)
    {
        try
        {
            using (var context = new SaleDbContext())
            {
                var saleDao = new SaleDao(context);
                var removedItems = saleDao.RemovePendingSaleInformationEwallet(id, idTransaccion);
                return removedItems;
            }
        }
        catch (Exception ex)
        {
            // En caso de error al remover la venta no hay ningún problema, ya que la tabla es de trabajo. Se puede continuar.
            Log.WriteLine(SubModule, string.Format("Excepción SaleManager-RemovePendingSaleInformationEwallet. Excepción:{0} Stacktrace:{1}", ex.Message, ex.StackTrace.ToString()));
            return 0;
        }
    }
}

```

[Ver figura](#)

Se determina que la excepción no tiene un impacto en el flujo.

Ahora es necesario revisar el procesamiento de las formas de pago, se revisa el log llamado [20241013\\_FARMAX\\_POS\\_CORE.XLOG](#), este se ubica en **%ProgramData%/FAhorro/FARMAX\_POS/**, este log puede contener las operaciones realizadas con las formas de pago, no es muy detallado pero se presentan las excepciones que pueden ocurrir al intentar procesar una forma de pago, se tiene a partir de la **línea 750**:


```

[2024-10-13 14:47:08:631],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(3)
QPSLimitAmount: '250', PaymentType: 'TDEB' payment.IsQPS='True'
[2024-10-13 14:47:08:631],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(4)
QPSLimitAmount: '250', PaymentType: 'TDEB' payment.IsQPS='True'
[2024-10-13 14:47:08:644],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],ApplyPayments - Message: La
forma de pago TARJETA DE DEBITO ACI (AD) no pudo ser aplicado., StackTrace: en
FAhorro.POS.Win.Tender.TenderManager.ApplyPayments()
[2024-10-13 14:47:43:093],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(3)
QPSLimitAmount: '250', PaymentType: 'TDEB' payment.IsQPS='True'
[2024-10-13 14:47:43:093],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(4)
QPSLimitAmount: '250', PaymentType: 'TDEB' payment.IsQPS='True'
[2024-10-13 14:47:43:093],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(3)
QPSLimitAmount: '250', PaymentType: 'EFEC' payment.IsQPS='False'
[2024-10-13 14:47:43:094],FARMAX_POS,CORE,ERROR,[APPLYPAYMENTS],#####(4)
QPSLimitAmount: '250', PaymentType: 'EFEC' payment.IsQPS='False'
[2024-10-13 14:49:07:556],FARMAX_POS,CORE,ERROR,[TRYGETBUSINESSERRORMESSAGE],Usuario y/o
contraseña inválidos.

```

Se observa que la primera forma de pago que se aplica es la tarjeta de débito, se observa también que ocurrió un error en el primer intento de procesamiento para la forma de pago de tarjeta de débito, sin embargo, la tarjeta luego se procesa correctamente junto con el pago de efectivo, se observa que **no ocurrió una excepción al procesar el efectivo**.

Se revisa el log del pin pad [20241013\\_FARMAX\\_POS\\_ANXOPINPAD.XLOG](#), puesto que ocurrió una excepción en el primer intento de procesar el método de pago de tarjeta de

		Incidencia – Venta finalizada con forma de pago pendiente	
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

débito, el log se encuentra en %ProgramData%/FAhorro/FARMAX\_POS, este contiene las operaciones realizadas en el pin pad, se observa a partir de la línea 703:

```
[2024-10-13 14:46:50:044],FARMAX_POS,ANKOPINPAD,INFO,,
-----
CC1 request - card processing start
-----
[2024-10-13 14:46:50:044],FARMAX_POS,ANKOPINPAD,INFO,,
Amount: $200
Counter: 21
Payment Type: Credit/debit tender ID: 12
Transaction Code: Q QPS

[2024-10-13 14:46:59:673],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,ObtieneTarjeta,00:00:09.63
[2024-10-13 14:46:59:673],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,CierraVentana,00:00:00.00
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,,
*****
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,,Loaded configuration shown
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Implementation:      AnxoImplementation
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port:      9
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port PRESENT:      True
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port baud rate: 19200
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port parity:      None
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port data bits: 8
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Serial port stop bits: One
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad ACK:      5
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad Q4:      20
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad CC0:      30
[2024-10-13 14:46:59:674],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad CC1:      45
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad CC4 chip 45
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Timeout PINpad CP0      30
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Auth server:      192.168.152.17
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Auth server port:      7000
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Timeout auth:      30
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, Credit/debit tender ID: 12
[2024-10-13 14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,, AMEX tender ID:      14
[2024-10-13
14:46:59:675],FARMAX_POS,ANKOPINPAD,INFO,,*****
[2024-10-13 14:47:00:032],FARMAX_POS,ANKOPINPAD,INFO,,TIMING2,SendAuthMessage,00:00:00.35,OK
[2024-10-13
14:47:00:049],FARMAX_POS,ANKOPINPAD,WARNING,,SUC:2575|CAJA:00000021|FOL:092966|TARJ:*****8414|MTO:20000|
ARQC:A1D8B43D3DC0614C|RC:51|AUT:000000


[2024-10-13 14:47:00:396],FARMAX_POS,ANKOPINPAD,WARNING,,AUTH NOT approved
[2024-10-13 14:47:03:675],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,EnviaPeticiónVenta,00:00:03.99
[2024-10-13 14:47:03:675],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,CierraVentana,00:00:00.00
```

Se observa que el primer intento fue por el monto de **MX\$200**, pero que este no fue aprobado.

Se revisa el siguiente intento, a partir de la línea 764:

```
[2024-10-13 14:47:28:323],FARMAX_POS,ANKOPINPAD,INFO,,
-----
CC1 request - card processing start
-----
[2024-10-13 14:47:28:323],FARMAX_POS,ANKOPINPAD,INFO,,
Amount: $180
Counter: 21
Payment Type: Credit/debit tender ID: 12
Transaction Code: Q QPS

[2024-10-13 14:47:38:020],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,ObtieneTarjeta,00:00:09.69
[2024-10-13 14:47:38:020],FARMAX_POS,ANKOPINPAD,INFO,,TIMING1,CierraVentana,00:00:00.00
[2024-10-13 14:47:38:021],FARMAX_POS,ANKOPINPAD,INFO,,
*****
```


	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

```
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,,Loaded configuration shown
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Implementation:      AnxoImplementation
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port:          9
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port PRESENT:      True
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port baud rate:   19200
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port parity:      None
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port data bits:   8
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Serial port stop bits:   One
[2024-10-13 14:47:38:021],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad ACK:      5
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad Q4:       20
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad CC0:      30
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad CC1:      45
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad CC4 chip 45
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout PINpad CP0       30
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Auth server:            192.168.152.17
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Auth server port:       7000
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Timeout auth:           30
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, Credit/debit tender ID: 12
[2024-10-13 14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,, AMEX tender ID:        14
[2024-10-13
14:47:38:022],FARMAX_POS,ANXOPINPAD,INFO,,*****
[2024-10-13 14:47:38:295],FARMAX_POS,ANXOPINPAD,INFO,,TIMING2,SendAuthMessage,00:00:00.27,OK
[2024-10-13
14:47:38:295],FARMAX_POS,ANXOPINPAD,WARNING,,SUC:2575|CAJA:00000021|FOL:092966|TARJ:*****8414|MTO:18000|
ARQC:C2E7C7EBA2C1261E|RC:00|AUT:430575
[2024-10-13 14:47:38:641],FARMAX_POS,ANXOPINPAD,INFO,,AUTH approved
[2024-10-13 14:47:41:920],FARMAX_POS,ANXOPINPAD,INFO,,TIMING1,EnviaPeticiónVenta,00:00:03.89
[2024-10-13 14:47:41:920],FARMAX_POS,ANXOPINPAD,INFO,,TIMING1,CierraVentana,00:00:00.00
```

Se observa que esta vez fue procesado por el monto de **MX\$180** y esta vez fue aprobado, por lo que la excepción anterior se da porque el primer intento no fue aprobado.

Se determina que las formas de pago fueron procesadas correctamente y que la excepción encontrada no interrumpe ni corrompe el flujo de ninguna manera.



	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

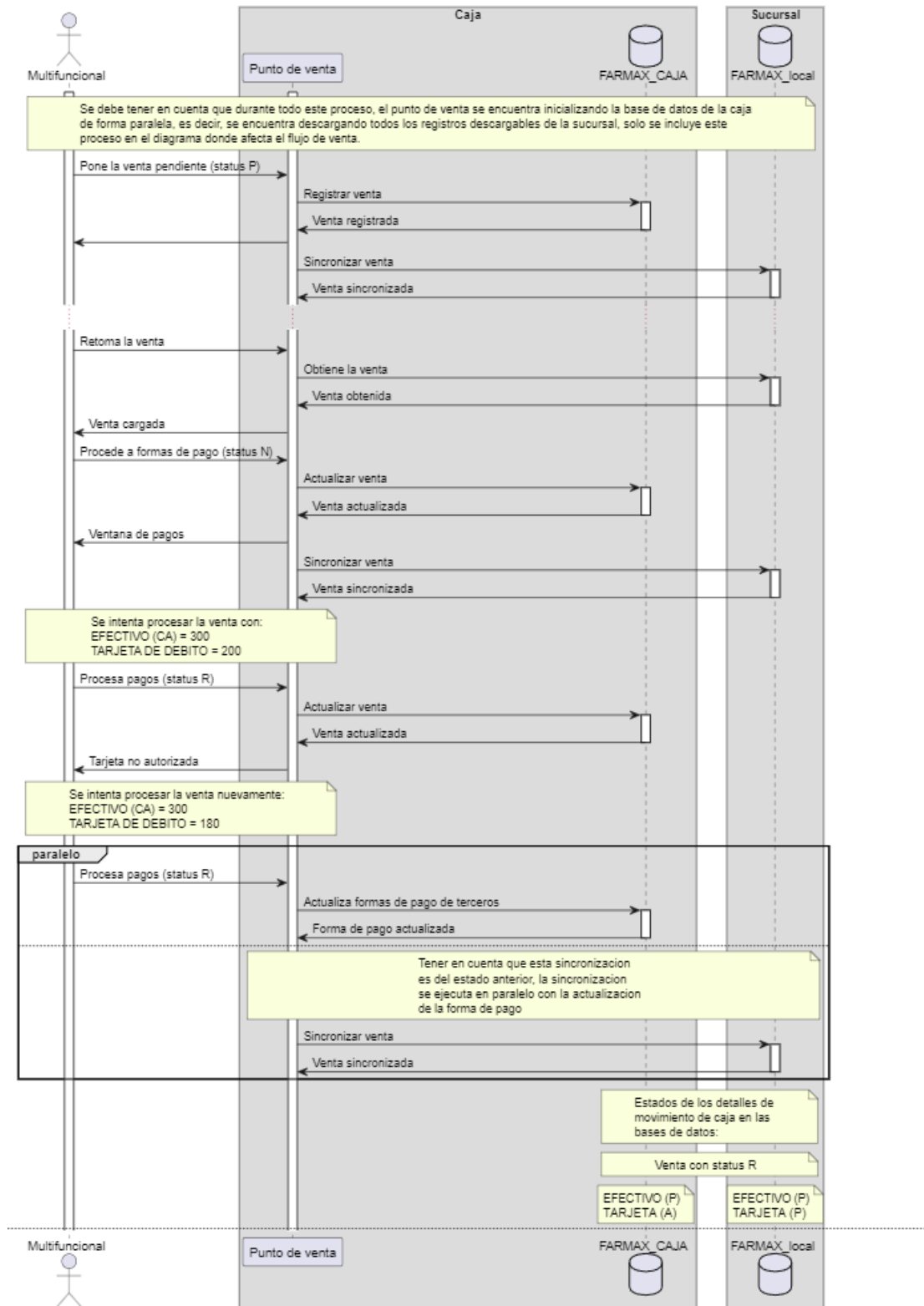
## Sincronización de bases de datos

Se determina que el flujo de venta no sufre ningún tipo de anomalía, por lo que se procede a revisar el log de sincronización, este log se llama [20241013 FARMAX TILL SERVER SYNCLIBRARY.XLOG](#) y se ubica en `%ProgramData%/FAhorro/FARMAX_TILL_SERVER`, en este log se presentan las operaciones de sincronización realizadas tanto para subida y como para descarga de datos entre la caja y la sucursal, se observa que al momento de procesar las formas de pago de la venta, al mismo tiempo, se está inicializando la base de datos de la caja, a partir de la **línea 3628**:

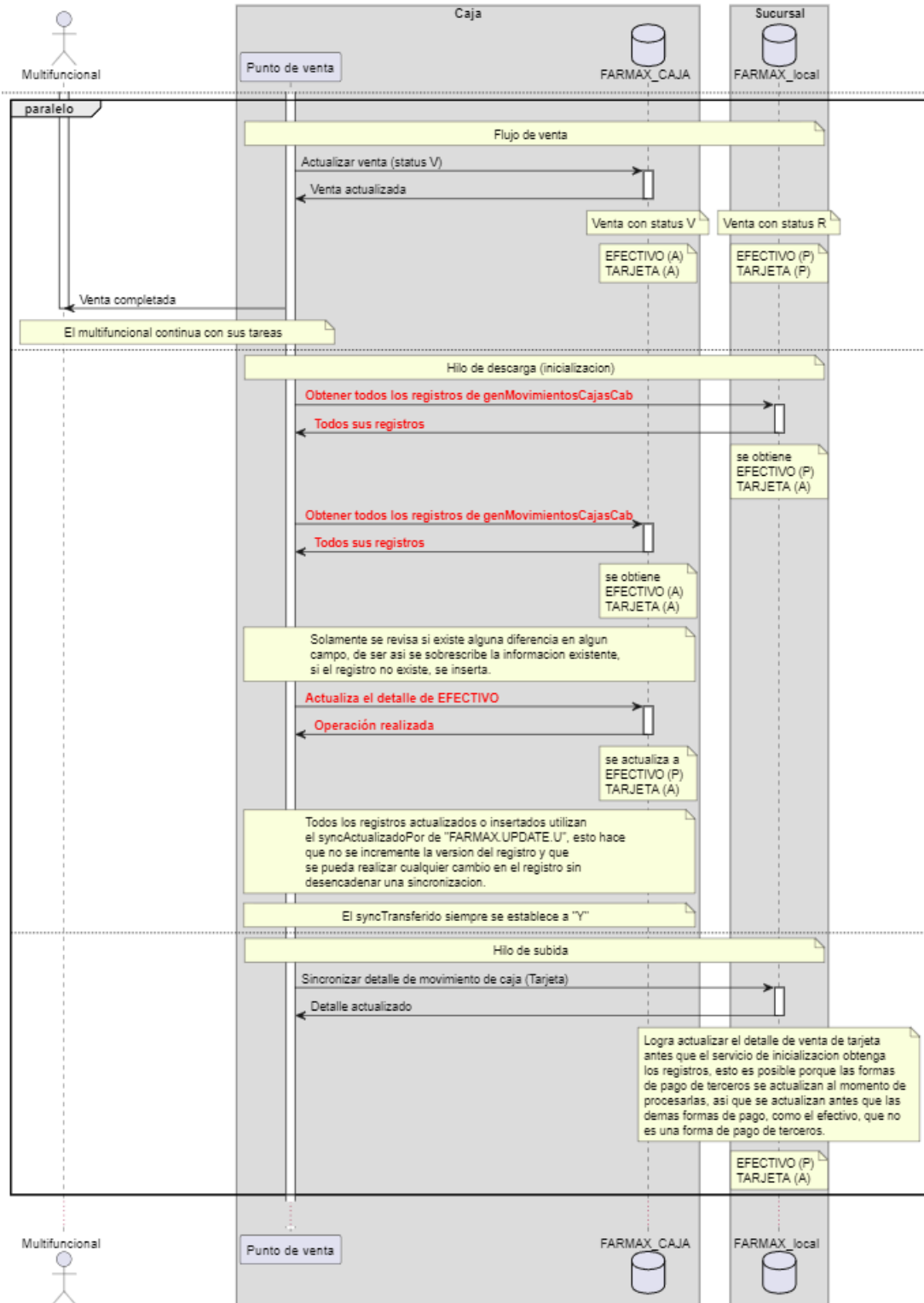
```
[2024-10-13 14:47:24:502],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_INIT_DATA]
Se descargara la tabla: 'genMovimientosCajasDet' con el servidor de sucursal
[2024-10-13 14:47:27:925],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_INIT_DATA]
Se recibieron 117985 registro(s) de la tabla: 'genMovimientosCajasDet'
[2024-10-13 14:47:42:776],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_TX_DATA] Se
guardaran 1 registro(s) de la tabla: 'genFormasPagosElectronicas' en la base de datos de
la sucursal
[2024-10-13 14:47:43:031],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_TX_DATA] Se
guardaran 1 registro(s) de la tabla: 'genMovimientosCajasDet' en la base de datos de la
sucursal
[2024-10-13 14:47:43:087],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_TX_DATA] Se
guardaran 1 registro(s) de la tabla: 'trnTransaccionesCab' en la base de datos de la
sucursal
[2024-10-13 14:48:05:150],FARMAX_TILL_SERVER,SYNCLIBRARY,INFO,,[SYNC_TILL_TX_DATA] Se
guardaran 1 registro(s) de la tabla: 'trnTransaccionesCab' en la base de datos de la
sucursal
```


Se determina que la caja se encontraba descargando todos los registros de `genMovimientosCajasDet` antes de mandar la última versión de la venta a la base de datos de la sucursal, se debe considerar también que esta descarga tarda alrededor de 30 segundos, en estos 30 segundos se insertan nuevos registros y se actualizan los registros existentes, con esta información se puede representar el escenario de lo que ocurrió utilizando un diagrama de secuencia de alto nivel:

Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

Todos estos procesos se ejecutan en paralelo, causando el problema de sobreescritura de datos.

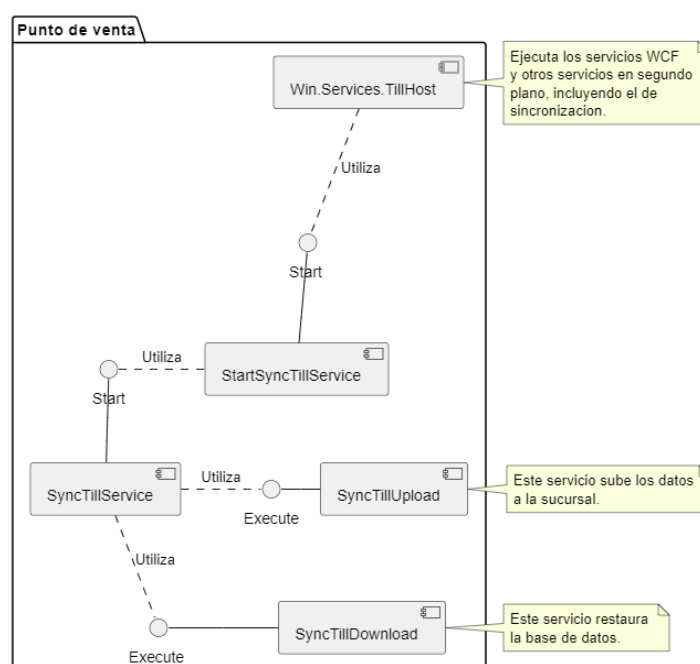
## Restauración de base de datos de caja

Se explica brevemente qué es y donde ocurre la inicialización de base de datos.


En la solución **POS.Services**, en el proyecto **Farmax.SyncTillLibrary** se encuentra el servicio **SyncTillService.cs**, este servicio ejecuta dos servicios llamados:

- **SyncTillDownload**
  - Se encarga de descargar datos desde la base de datos de la sucursal a la base de datos de la caja.
  - Otra de sus responsabilidades es inicializar la base de datos en caso de que sea necesario, este proceso también es conocido como modo o proceso de restauración y consiste en descargar todos los registros de todas las tablas descargables desde de la base de datos de la sucursal a la base de datos de la caja, incluso si estos ya existen en la base de datos de la caja.
- **SyncTillUpload**
  - Se encarga de subir datos desde la base de datos de la caja hacia la base de datos de la sucursal.

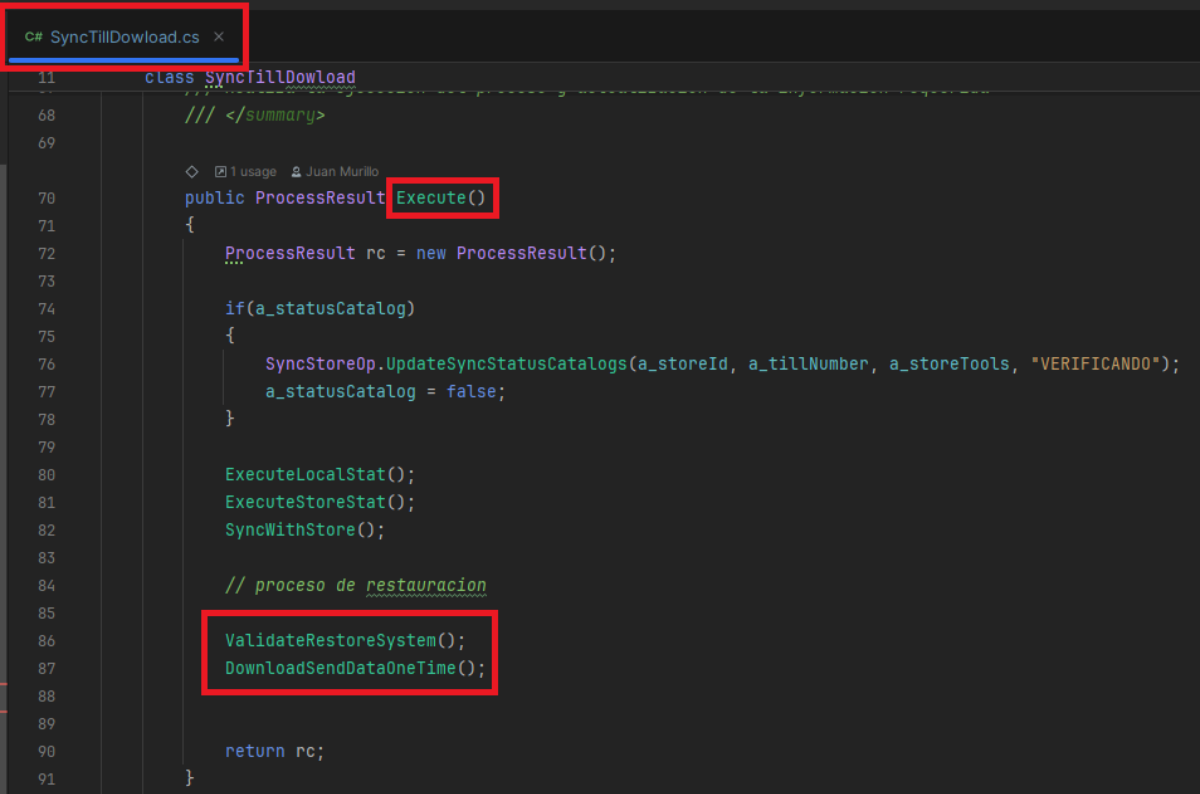
Se presenta en el siguiente diagrama de componentes la arquitectura de estos servicios:



[Ver diagrama](#)

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

En el servicio **SyncTillDownload**, en el método **Execute()**, se realiza la validación para saber si se debe realizar la restauración de base de datos de caja y posteriormente se realiza la restauración en caso de ser necesario, esta validación ocurre en **ValidateRestoreSystem()** y la ejecución de restauración ocurre en **DownloadSendDataOneTime()**:




```

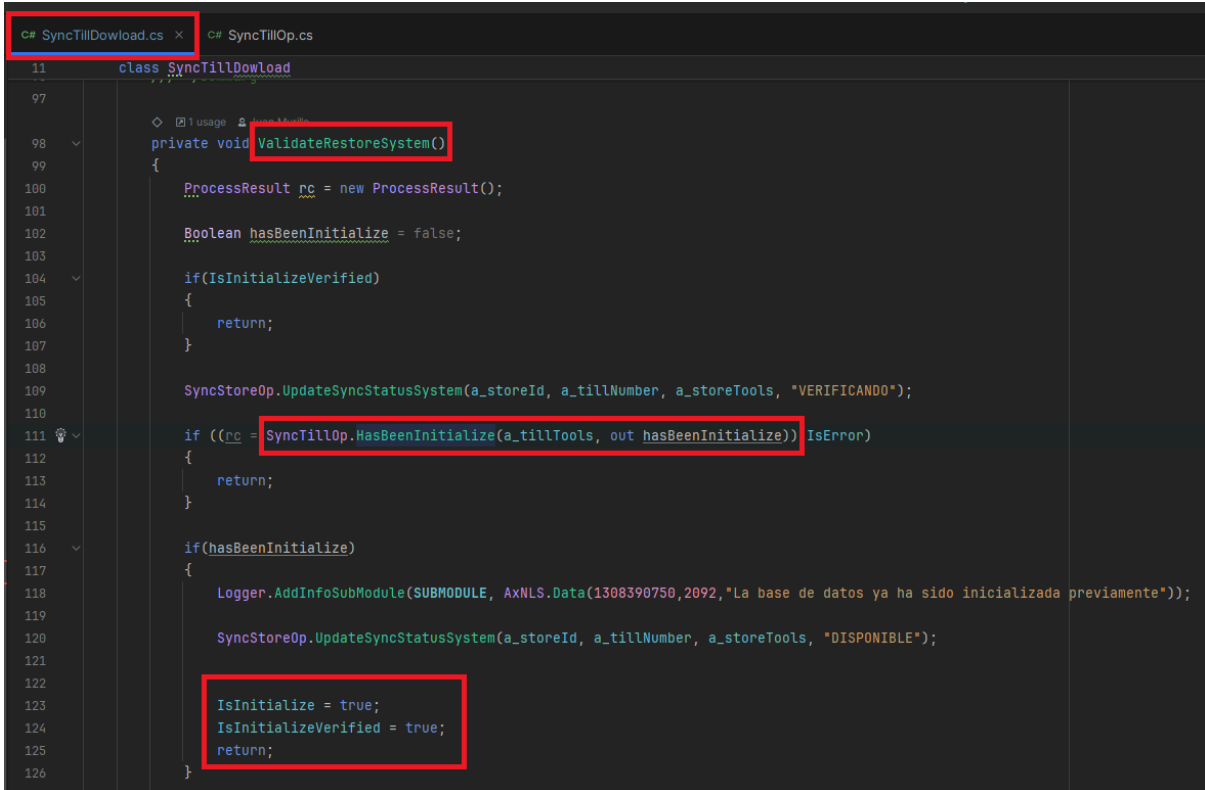
11 class SyncTillDownload
12 {
13     /// </summary>
14
15     public ProcessResult Execute()
16     {
17         ProcessResult rc = new ProcessResult();
18
19         if(a_statusCatalog)
20         {
21             SyncStoreOp.UpdateSyncStatusCatalogs(a_storeId, a_tillNumber, a_storeTools, "VERIFICANDO");
22             a_statusCatalog = false;
23         }
24
25         ExecuteLocalStat();
26         ExecuteStoreStat();
27         SyncWithStore();
28
29         // proceso de restauracion
30
31         ValidateRestoreSystem();
32         DownloadSendDataOneTime();
33
34         return rc;
35     }
36 }

```

[Ver figura](#)

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

Lo que se realiza en **ValidateRestoreSystem()** es revisar los valores en la tabla **AX\_PARAMETERS**, para el registro con el campo **PARAMETER** con valor de **40**:




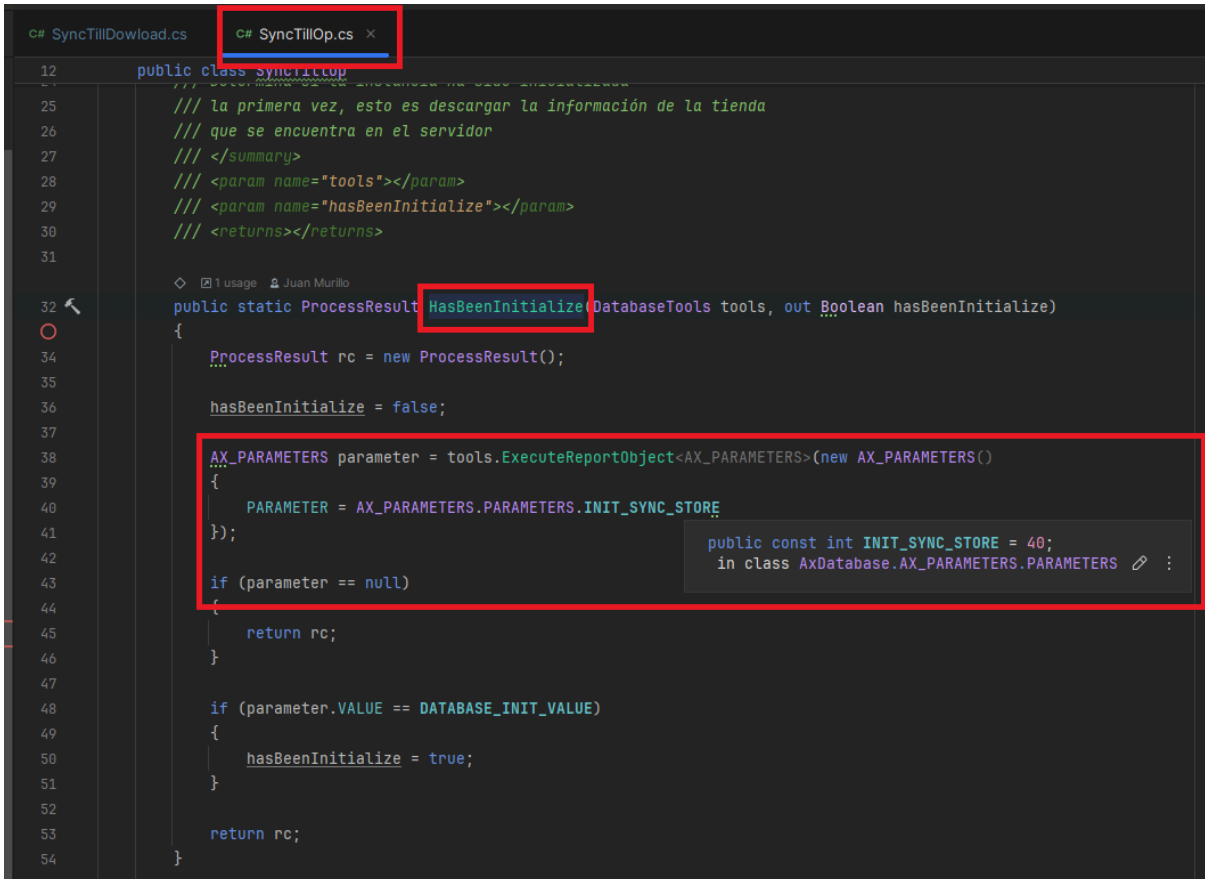
```

11  class SyncTillDownload
97
98  private void ValidateRestoreSystem()
99  {
100     ProcessResult rc = new ProcessResult();
101
102     Boolean hasBeenInitialize = false;
103
104     if(IsInitializeVerified)
105     {
106         return;
107     }
108
109     SyncStoreOp.UpdateSyncStatusSystem(a_storeId, a_tillNumber, a_storeTools, "VERIFICANDO");
110
111     if ((rc = SyncTillOp.HasBeenInitialize(a_tillTools, out hasBeenInitialize)) IsError)
112     {
113         return;
114     }
115
116     if(hasBeenInitialize)
117     {
118         Logger.AddInfoSubModule(SUBMODULE, AxNLS.Data(1308390750,2092,"La base de datos ya ha sido inicializada previamente"));
119
120         SyncStoreOp.UpdateSyncStatusSystem(a_storeId, a_tillNumber, a_storeTools, "DISPONIBLE");
121
122         IsInitialize = true;
123         IsInitializeVerified = true;
124         return;
125     }
126
127

```

[Ver figura](#)

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



```


12 public class SyncTillOp
13 {
14     /// 
15     /// la primera vez, esto es descargar la información de la tienda
16     /// que se encuentra en el servidor
17     /// 
18     /// </param>
19     /// 

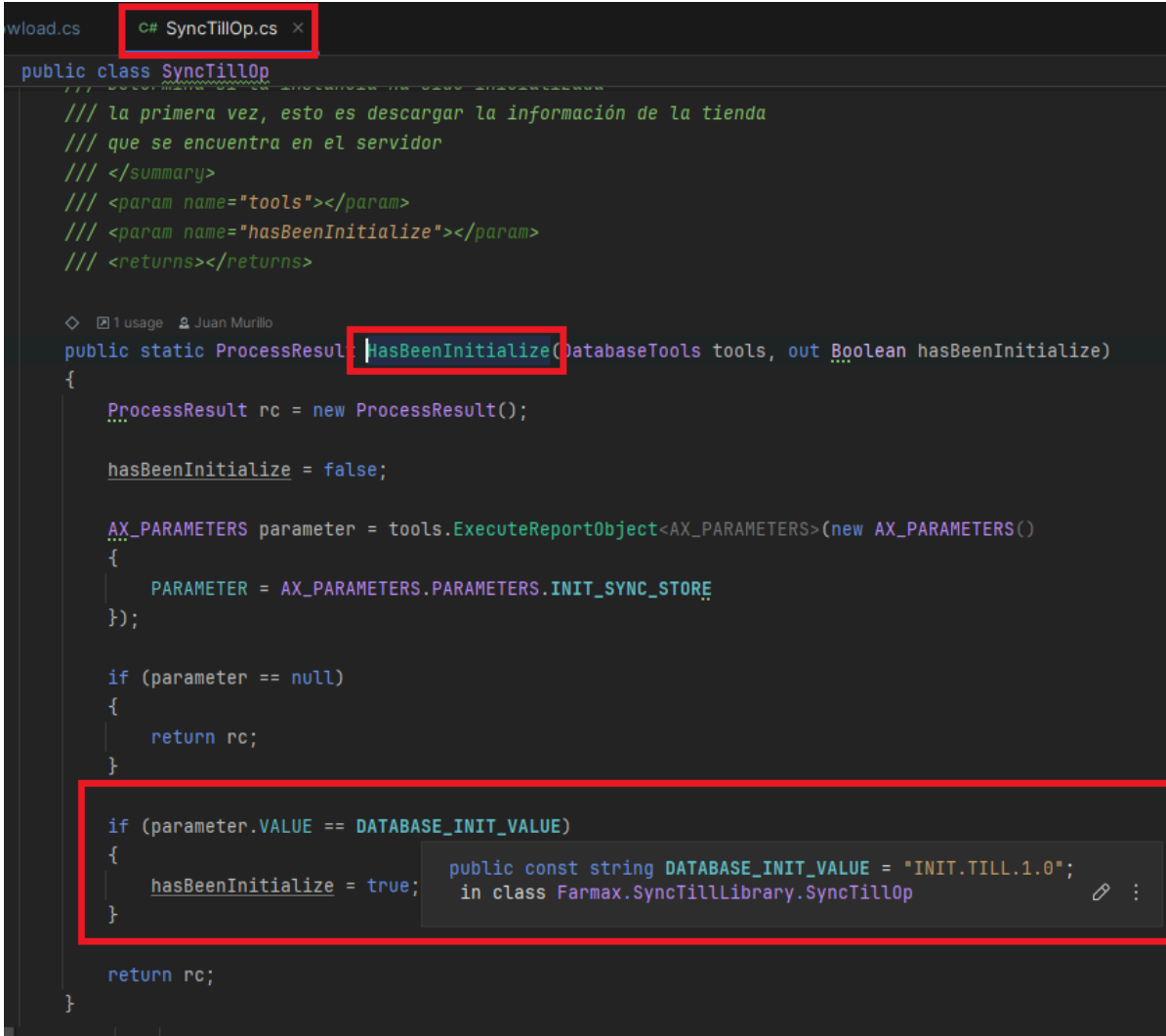
public const int INIT_SYNC_STORE = 40;  

    in class AxDatabase.AX_PARAMETERS.PARAMETERS


```

[Ver figura](#)

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024



```

wload.cs C# SyncTillOp.cs x
public class SyncTillOp
{
    /// 
    /// la primera vez, esto es descargar la información de la tienda
    /// que se encuentra en el servidor
    /// 
    /// </param>
    /// 

public const string DATABASE_INIT_VALUE = "INIT.TILL.1.0";  
in class Farmax.SyncTillLibrary.SyncTillOp


```

[Ver figura](#)

Esta información se puede consultar de la siguiente manera:

```


SELECT
    *
FROM AX_PARAMETERS
WHERE PARAMETER = 40;

```

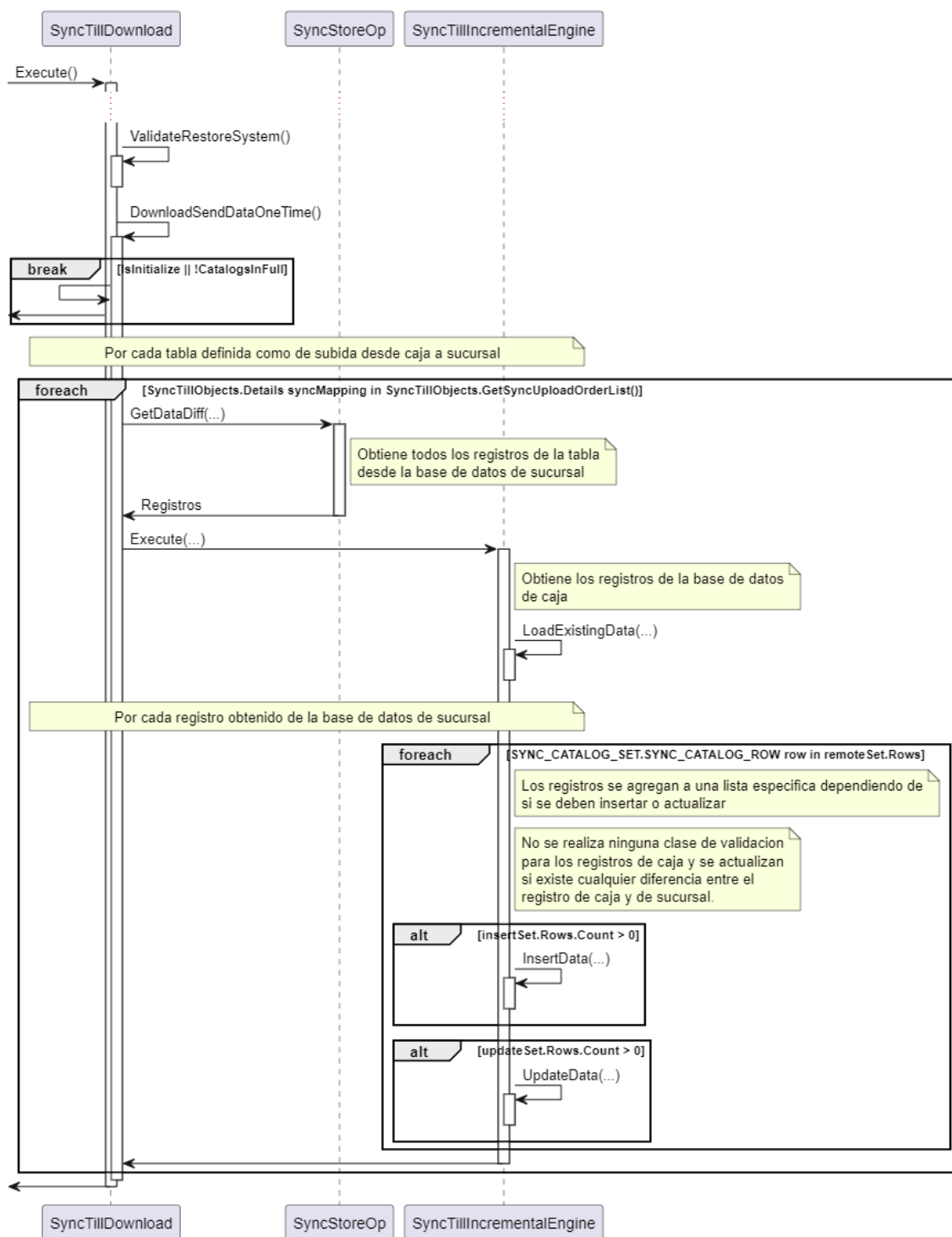
[Ver consulta](#)

Si el registro no existe o su campo **VALUE** es diferente a la cadena “INIT.TILL.1.0” se considera que la base de datos necesita ser restaurada.




	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

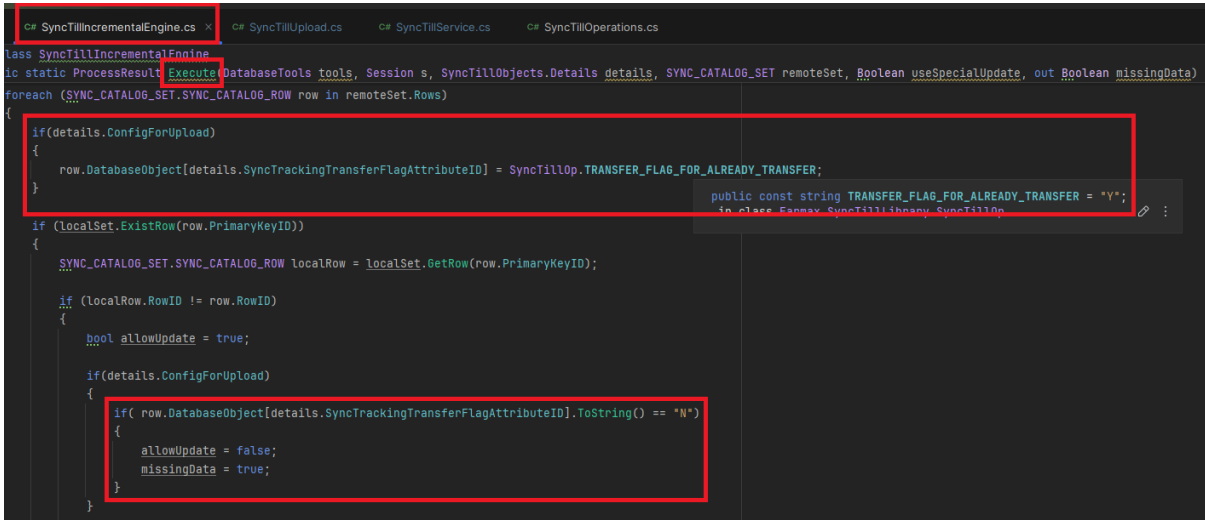
En el método **DownloadSendDataOneTime()** se descargan los datos desde la base de datos de la sucursal a la base de datos de la caja, primero se revisa si ya está inicializada la base de datos, si no, se procede a inicializar, esto se presenta a nivel técnico en el siguiente diagrama de secuencia:



[Ver diagrama](#)

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

Como se menciona en el diagrama de secuencia, no se realiza ninguna validación con respecto a los registros existentes en la base de datos de caja, aunque existen validaciones para los registros de base de datos de sucursal:



```

class SyncTillIncrementalEngine
{
    public static ProcessResult Execute(DatabaseTools tools, Session s, SyncTillObjects.Details details, SYNC_CATALOG_SET remoteSet, Boolean useSpecialUpdate, out Boolean missingData)
    {
        foreach (SYNC_CATALOG_SET.SYNC_CATALOG_ROW row in remoteSet.Rows)
        {
            if(details.ConfigForUpload)
            {
                row.DatabaseObject[details.SyncTrackingTransferFlagAttributeID] = SyncTillOp.TRANSFER_FLAG_FOR_ALREADY_TRANSFER;
            }

            if (localSet.ExistRow(row.PrimaryKeyID))
            {
                SYNC_CATALOG_SET.SYNC_CATALOG_ROW localRow = localSet.GetRow(row.PrimaryKeyID);

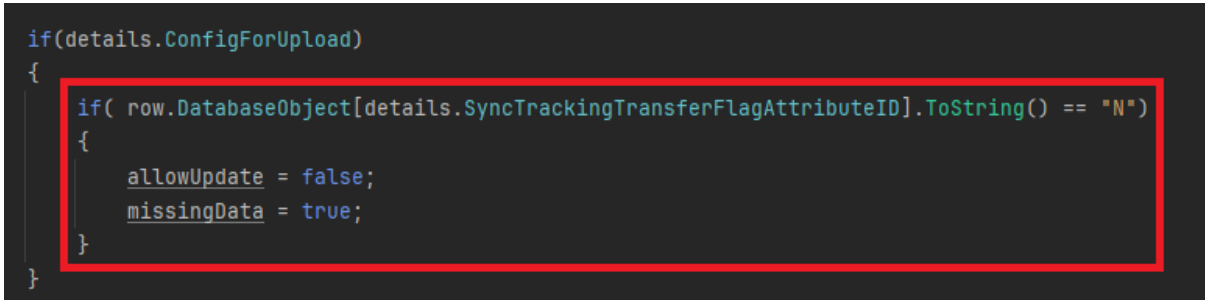
                if (localRow.RowID != row.RowID)
                {
                    bool allowUpdate = true;

                    if(details.ConfigForUpload)
                    {
                        if( row.DatabaseObject[details.SyncTrackingTransferFlagAttributeID].ToString() == "N")
                        {
                            allowUpdate = false;
                            missingData = true;
                        }
                    }
                }
            }
        }
    }
}

```

[Ver figura](#)

Aunque después hay una validación para saber si el registro de sucursal no se ha transferido:



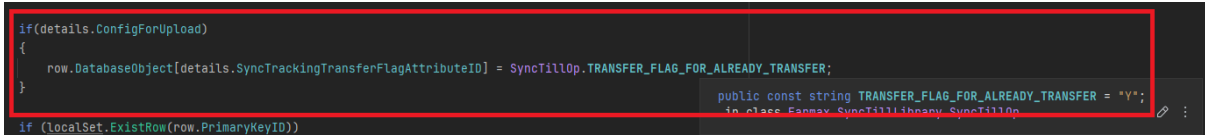
```

if(details.ConfigForUpload)
{
    if( row.DatabaseObject[details.SyncTrackingTransferFlagAttributeID].ToString() == "N")
    {
        allowUpdate = false;
        missingData = true;
    }
}

```

[Ver figura](#)

Esta validación **nunca se evalúa a true** puesto que unas líneas de código antes el valor se establece a “Y” (transferido):




```

if(details.ConfigForUpload)
{
    row.DatabaseObject[details.SyncTrackingTransferFlagAttributeID] = SyncTillOp.TRANSFER_FLAG_FOR_ALREADY_TRANSFER;
}

```

[Ver figura](#)

Es posible sincronizar el hilo de restauración de base de datos en **SyncTillDownload** con el hilo de subida de datos en **SyncTillUpload**, existe un **Mutex** en **SyncTillService** que se utiliza en **SyncTillUpload** para sincronizar el proceso de subida:

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

```
C# SyncTillDownload.cs C# SyncTillIncrementalEngine.cs C# SyncTillUpload.cs C# SyncTillService.cs x C# SyncTillOperations.cs
1 > using ...
7
8 namespace Farmax.SyncTillLibrary
9 {
10     [14 usages] [Juan Murillo *] [1 exposing API]
11     public class SyncTillService
12     {
13         private Boolean a_isrunning = false;
14         private AxThread a_mainthread = null;
15         private AxThread a_uploadthread = null;
16         private DatabaseTools a_tillTools = null;
17         private DatabaseTools a_storeTools = null;
18         private Int32 a_storeId = -1;
19         private Int32 a_tillNumber = -1;
20         private SyncTillDownload a_process = null;
21         private SyncTillUpload a_processupload = null;
22
23         internal const String SUBMODULE = FAhorro.POS.Application.Core.LogSubModules.SubModule_SyncLibrary;
24
25         private const Int32 WAITING_TIME_BEFORE_STARTING_PROCESS = 60000; // tiempo expresado en ms
26         private const Int32 WAITING_TIME_BETWEEN_UPLOADS = 15000; // tiempo expresado en ms
27         private const Int32 WAITING_TIME_BETWEEN_DOWNLOADS = 60000; // tiempo expresado en ms
28
29         [4 usages]
30         public static Mutex UploadLock { get; private set; } = new Mutex();
```

Ver figura


Se observa que se bloquea por cada tabla a subir:

```
C# SyncTillDownload.cs C# SyncTillIncrementalEngine.cs C# SyncTillUpload.cs x C# SyncTillService.cs C# SyncTillOperations.cs
class SyncTillUpload
{
    /// <summary>
    /// Transporta la informacion desde la CAJA a la base de datos de la SUCURSAL, la que este pendiente de transferir.
    /// </summary>
    private void SendDataToServer()
    {
        ProcessResult rc = new ProcessResult();

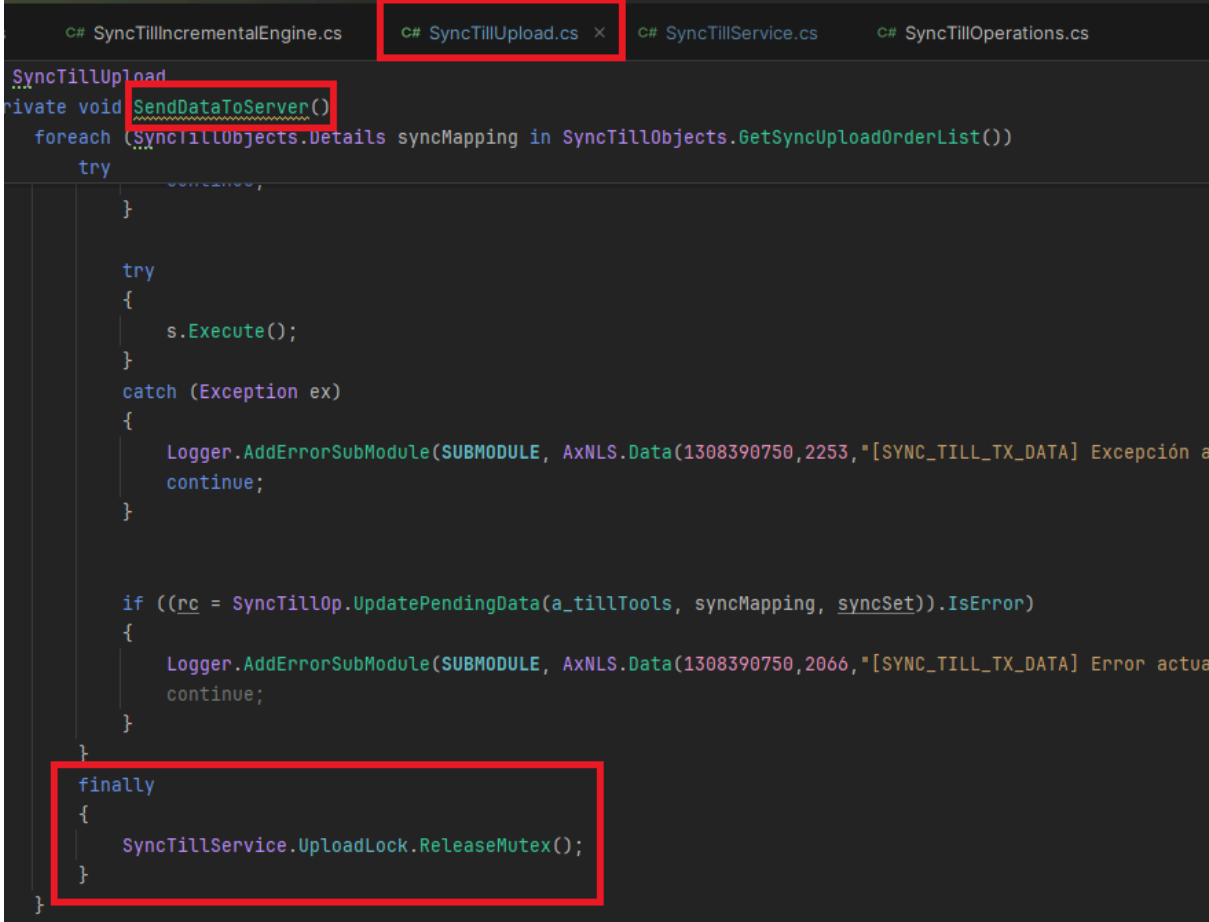
        SYNC_TXDATA_SET syncSet = null;

        foreach (SyncTillObjects.Details syncMapping in SyncTillObjects.GetSyncUploadOrderList())
        {
            SyncTillService.UploadLock.WaitOne();
```

Ver figura

	Incidencia – Venta finalizada con forma de pago pendiente		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

Luego de terminar con la tabla se libera:



```

C# SyncTillIncrementalEngine.cs  C# SyncTillUpload.cs x  C# SyncTillService.cs  C# SyncTillOperations.cs


SyncTillUpload
private void SendDataToServer()
{
    foreach (SyncTillObjects.Details syncMapping in SyncTillObjects.GetSyncUploadOrderList())
    {
        try
        {
            // ... code ...

            try
            {
                s.Execute();
            }
            catch (Exception ex)
            {
                Logger.AddErrorSubModule(SUBMODULE, AxNLS.Data(1308390750,2253,"[SYNC_TILL_TX_DATA] Excepción a
                continue;
            }

            if ((rc = SyncTillOp.UpdatePendingData(a_tillTools, syncMapping, syncSet)).IsError)
            {
                Logger.AddErrorSubModule(SUBMODULE, AxNLS.Data(1308390750,2066,"[SYNC_TILL_TX_DATA] Error actua
                continue;
            }
        }
        finally
        {
            SyncTillService.UploadLock.ReleaseMutex();
        }
    }
}

```

[Ver figura](#)


		Incidencia – Venta finalizada con forma de pago pendiente	
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

## Posibles causas

El estado del detalle de movimiento de caja pendiente en una venta finalizada **se debe a que el proceso de inicialización de base de datos de caja recibió información desactualizada** de la transacción realizada desde la **base de datos de la sucursal**, la información recibida es de **antes de poder sincronizar el detalle de movimiento de caja**, por lo que se **utiliza una versión en la que el detalle de movimiento de caja para la forma de pago en efectivo aún se encontraba pendiente**, pero desde la perspectiva del punto de venta, la venta fue realizada **correctamente** y la **venta fue actualizada en la base de datos de caja**, para luego ser **sobrescrita por el proceso de inicialización**, esto ocurre después de que la venta finaliza.


Los problemas solamente se dan con el detalle de movimiento de caja porque **genMovimientosCajaDet es la tabla que se estaba descargando en ese momento**, la incidencia ocurre en una situación muy excepcional donde distintos hilos y procesos se encontraban trabajando con la misma fuente de datos y compitiendo para realizar sus operaciones de actualización.

Si bien los motores de bases de datos implementan bloqueos a distintos niveles (filas, páginas, tablas, etc.) para las operaciones de lectura y escritura, los procesos de la incidencia que utilizan la base de datos no están al tanto uno del otro, no se sincronizan de ninguna manera lo que puede llevar a este comportamiento errático.

		Incidencia – Venta finalizada con forma de pago pendiente	
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	31-10-2024	31-10-2024

## Posible solución

Para sincronizar el servicio de **SyncTillDownload** con el de **SyncTillUpload**, se puede utilizar el **Mutex** en **SyncTillService** llamado **UploadLock**, se podría bloquear la subida antes de descargar una tabla:



```

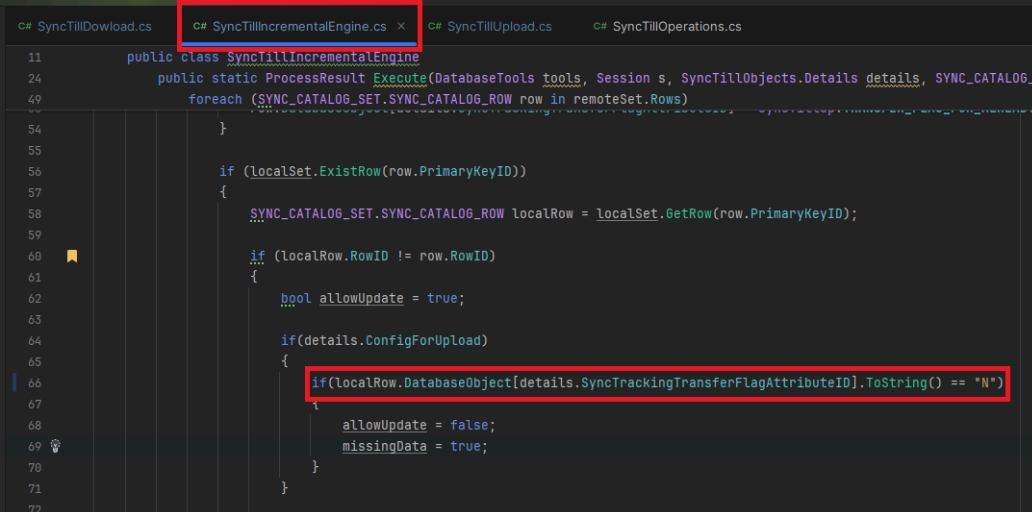
11      class SyncTillDownload
143         private void DownloadSendDataOneTime()
156
157             Boolean completeAll = true;
158
159             foreach (SyncTillObjects.Details syncMapping in SyncTillObjects.GetSyncUploadOrderList())
160             {
161                 SyncTillService.UploadLock.WaitOne();
162
163                 try
164                 {
228                     // ...
229                 }
230                 finally
231                 {
232                     SyncTillService.UploadLock.ReleaseMutex();
233                 }
234             }

```

[Ver figura](#)

Se encierra todo el código del foreach en un try-finally cuyo único propósito es asegurarse de que el bloqueo se libera después de finalizar de procesar una tabla, tenga éxito o no esta operación, para permitir al servicio de sincronización continuar con su procesamiento.

Bloquear la subida permite comparar los registros de la base de datos de caja con la confianza de que estos no van a sufrir un cambio realizado por el servicio de subida de datos a sucursal:



```

11      public class SyncTillIncrementalEngine
24      {
25          public static ProcessResult Execute(DatabaseTools tools, Session s, SyncTillObjects.Details details, SYNC_CATALOG_SET syncCatalogSet)
49          {
50              foreach (SYNC_CATALOG_SET.SYNC_CATALOG_ROW row in remoteSet.Rows)
51              {
52                  // ...
53              }
54
55              if (localSet.ExistRow(row.PrimaryKeyID))
56              {
57                  SYNC_CATALOG_SET.SYNC_CATALOG_ROW localRow = localSet.GetRow(row.PrimaryKeyID);
58
59                  if (localRow.RowID != row.RowID)
60                  {
61                      bool allowUpdate = true;
62
63                      if (details.ConfigForUpload)
64                      {
65                          if (localRow.DatabaseObject[details.SyncTrackingTransferFlagAttributeID].ToString() == "N")
66                          {
67                              allowUpdate = false;
68                              missingData = true;
69                          }
70                      }
71                  }
72              }

```

[Ver figura](#)