	Backup Preventivo		
Autor	Versión	Fecha de elaboración	Última edición
Eduardo Ochoa	1.0	12-06-2025	12-06-2025

Control de versiones

Versión	Fecha	Autor	Descripción
1.0	12-06-2025	Eduardo Ochoa	Propuesta de backup preventivo

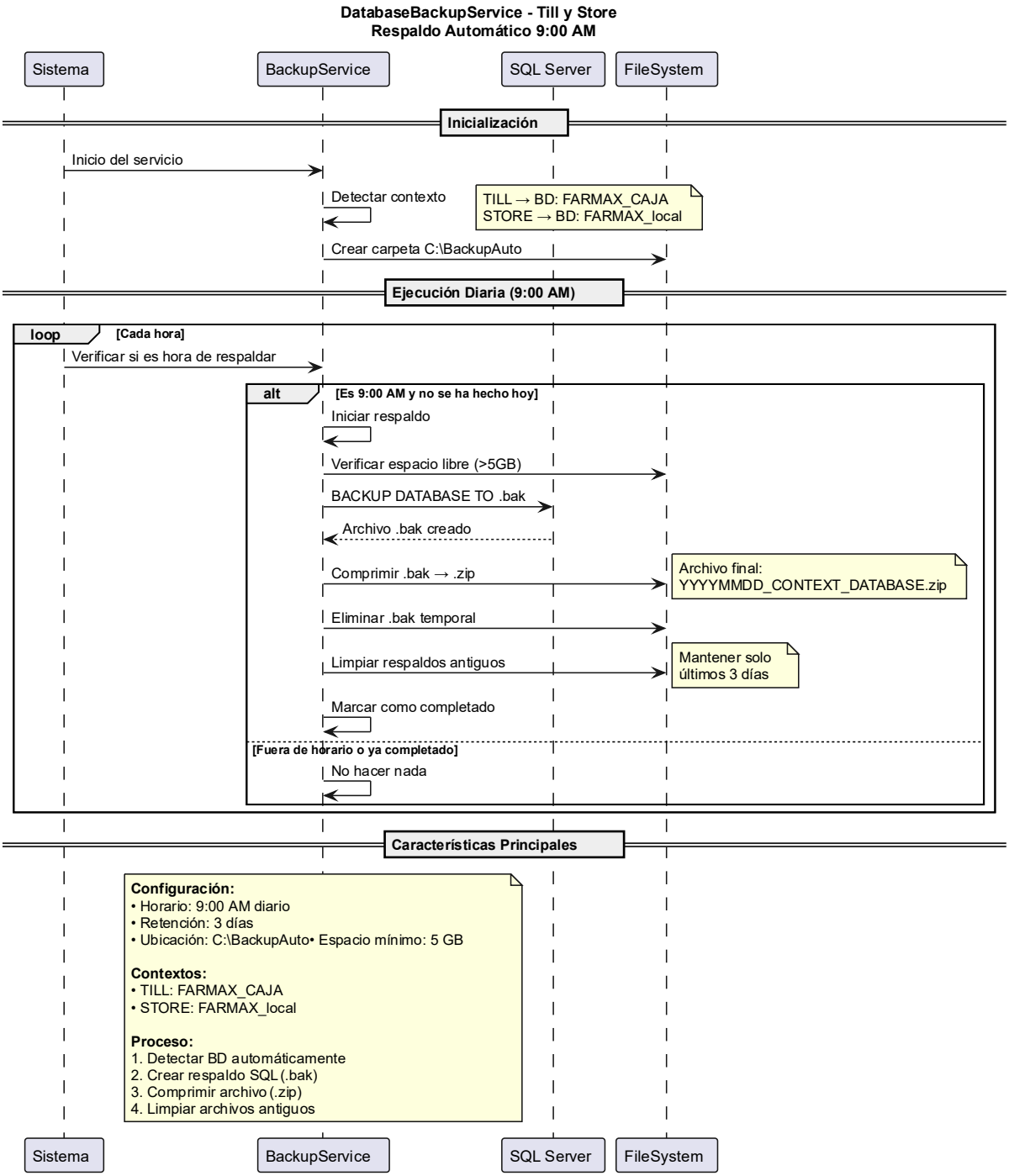
Sucursal	Todas
Caja	Todas
Venta	NA
Version	10.5.60702

Descripción general

Se describe una propuesta de un servicio de respaldo automático de bases de datos tanto en caja como en store, su propósito es garantizar copias de seguridad periódicas de las base de datos FARMAX_CAJA y FARMAX_local según corresponda el contexto en el que se este ejecutando. El servicio mantiene únicamente los respaldos de los últimos 3 dias en .zip y se elimina el mas antiguo una vez se cumplen estos 3 dias para no consumir el espacio libre del disco duro, además, esta configurado para ejecutarse todos los días en un intervalo de 9 a 10 am considerando los turnos que se manejan actualmente.

A continuación, se muestra un diagrama general de las características de este servicio.

	Backup Preventivo		
Autor	Versión	Fecha de elaboración	Última edición
Eduardo Ochoa	1.0	12-06-2025	12-06-2025



Causa del problema

Posibles fallos o corrupción de la base de datos en las sucursales o cajas.

Posible solución

Como medida preventiva se construye el servicio DatabaseBackupService que proporciona respaldos automáticos diarios tanto para las bases de datos de caja como las de sucursal.
A continuación, se mencionan los puntos principales del código fuente para el funcionamiento del servicio:

1. instanciación: El servicio de respaldo para la caja es iniciado desde el proceso Windows Win.Services.TillHost. En la implementación de la clase ServiceImplementation.

```
DataCenter.DatabaseBackupService = new DatabaseBackupService(FAhorro.DataManagement.DataManagement.TillDbInstance.Tools);
if ((rc = DataCenter.DatabaseBackupService.Start()).IsError)
{
    return rc;
}
```

En el **contexto de sucursal (STORE)**, la instanciación es similar pero ocurre desde el servicio DEV_StoreService en la clase FEStoreServiceImpl.

```
StoreDataCenter.DatabaseBackupService = new DatabaseBackupService(StoreDataCenter.StoreTools);
if ((rc = StoreDataCenter.DatabaseBackupService.Start()).IsError)
{
    Stop();
    return rc.Log();
}
```

2. **Detección del entorno (TILL vs STORE) dentro de DatabaseBackupService:**
   El código relevante está en DetectDatabaseAndContext() aquí hay un bloque if/else que detecta si existe la base de datos FARMAX_CAJA si es así estamos en el contexto till si no es así busca si estamos en el contexto store revisando la existencia de FARMAX_local.

```
private void DetectDatabaseAndContext()
{
    if (DatabaseExists( dbName: "FARMAX_CAJA"))
    {
        _databaseName = "FARMAX_CAJA";
        _context = "TILL";
    }
    else if (DatabaseExists( dbName: "FARMAX_local"))
    {
        _databaseName = "FARMAX_local";
        _context = "STORE";
    }
    else
    {
        throw new Exception("No se encontró BD FARMAX_CAJA ni FARMAX_local");
    }
}
```

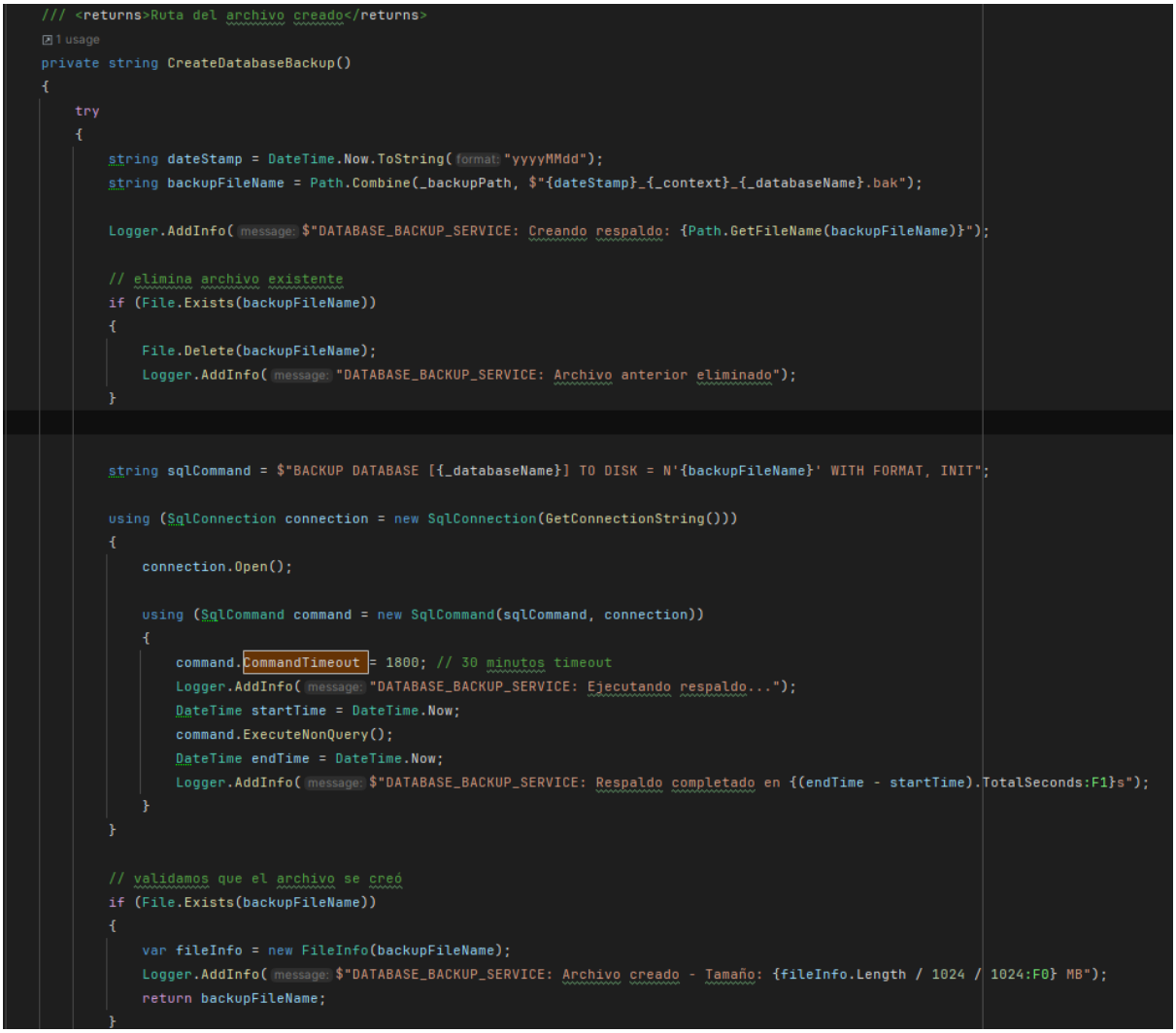
3. **Proceso principal de respaldo:** El método principal que ejecuta la tarea de respaldo es BackgroundExecutionTask() Su implementación básicamente comprueba si debe hacer el backup ahora (ShouldPerformBackup()), y si devuelve true, entonces escribe en log que iniciará el proceso y llama a PerformDailyBackup(). El método ShouldPerformBackup contiene las decisiones de timing y estado para la ventana horaria de 9:00am – 10:00 am no repitiendo el mismo día para evitar planchado.

```
protected override void BackgroundExecutionTask()
{
    try
    {
        if (ShouldPerformBackup())
        {
            Logger.AddInfo( message: "DATABASE_BACKUP_SERVICE: Iniciando proceso de respaldo programado");
            PerformDailyBackup();
        }
    }
    catch (Exception ex)
    {
        Logger.AddError( message: $"DATABASE_BACKUP_SERVICE: Error en tarea de respaldo: {ex.Message}");
    }
}
```

```
private bool ShouldPerformBackup()
{
    DateTime now = DateTime.Now;

    // solo entre las 9:00 y 10:00 AM
    if (now.Hour < BACKUP_HOUR || now.Hour >= BACKUP_HOUR + 1)
    {
        return false;
    }

    // no hacer el respaldo si este ya se hizo hoy
    if (_lastBackupDate.Date == now.Date)
    {
        return false;
    }

    // no hacer respaldo si otro respaldo ya está en progreso
    if (_backupInProgress)
    {
        Logger.AddWarning( message: "DATABASE_BACKUP_SERVICE: Respaldo en progreso, saltando ejecución");
        return false;
    }

    // validar que la ruta esté disponible
    if (string.IsNullOrEmpty(_backupPath))
    {
        Logger.AddError( message: "DATABASE_BACKUP_SERVICE: Ruta de respaldo no válida");
        return false;
    }

    return true;
}
```

```
private void PerformDailyBackup()
{
    _backupInProgress = true;
    string backupFileName = null;
    DateTime startTime = DateTime.Now;

    try
    {
        Logger.AddInfo( message: "DATABASE_BACKUP_SERVICE: ------- INICIO RESPALDO DIARIO ------");

        // 1. validar el espacio en disco
        if (!CheckDiskSpace())
        {
            Logger.AddError( message: "DATABASE_BACKUP_SERVICE: Espacio insuficiente en disco");
            return;
        }

        // 2. crear el respaldo de la base de datos
        backupFileName = CreateDatabaseBackup();
        if (string.IsNullOrEmpty(backupFileName))
        {
            Logger.AddError( message: "DATABASE_BACKUP_SERVICE: Error al crear respaldo de base de datos");
            return;
        }

        // 3. comprime archivo
        string zipFileName = CompressBackupFile(backupFileName);
        if (string.IsNullOrEmpty(zipFileName))
        {
            Logger.AddError( message: "DATABASE_BACKUP_SERVICE: Error al comprimir respaldo");
            return;
        }

        // 4. elimina archivo .bak original
        DeleteFile(backupFileName);

        // 5. limpia  respaldos antiguos
        CleanOldBackups();

        // 6. estado como completado
        _lastBackupDate = DateTime.Now;
        TimeSpan totalTime = DateTime.Now - startTime;
```

Podemos ver que aquí se llaman los métodos con las características que mencionamos. Por ejemplo, después de crear la base, si CreateDatabaseBackup() retorna null (indicando falla en la creación del .bak), se registra error y se retorna inmediatamente sin intentar comprimir ni limpiar.



```
/// <returns>Ruta del archivo creado</returns>
1 usage
private string CreateDatabaseBackup()
{
    try
    {
        string dateStamp = DateTime.Now.ToString( format: "yyyyMMdd");
        string backupFileName = Path.Combine(_backupPath, $"{dateStamp}_{_context}_{_databaseName}.bak");

        Logger.AddInfo( message: $"DATABASE_BACKUP_SERVICE: Creando respaldo: {Path.GetFileName(backupFileName)}");

        // elimina archivo existente
        if (File.Exists(backupFileName))
        {
            File.Delete(backupFileName);
            Logger.AddInfo( message: "DATABASE_BACKUP_SERVICE: Archivo anterior eliminado");
        }


        string sqlCommand = $"BACKUP DATABASE [{_databaseName}] TO DISK = N'{backupFileName}' WITH FORMAT, INIT";

        using (SqlConnection connection = new SqlConnection(GetConnectionString()))
        {
            connection.Open();

            using (SqlCommand command = new SqlCommand(sqlCommand, connection))
            {
                command.CommandTimeout = 1800; // 30 minutos timeout
                Logger.AddInfo( message: "DATABASE_BACKUP_SERVICE: Ejecutando respaldo...");
                DateTime startTime = DateTime.Now;
                command.ExecuteNonQuery();
                DateTime endTime = DateTime.Now;
                Logger.AddInfo( message: $"DATABASE_BACKUP_SERVICE: Respaldo completado en {(endTime - startTime).TotalSeconds:F1}s");
            }
        }

        // validamos que el archivo se creó
        if (File.Exists(backupFileName))
        {
            var fileInfo = new FileInfo(backupFileName);
            Logger.AddInfo( message: $"DATABASE_BACKUP_SERVICE: Archivo creado - Tamaño: {fileInfo.Length / 1024 / 1024:F0} MB");
            return backupFileName;
        }
```

Conclusión

El servicio DatabaseBackupService proporciona respaldos automáticos diarios tanto para las bases de datos de caja como las de sucursal, su funcionamiento es util en caso de fallos de hadware, corrupción de datos, disponer de copias de seguridad recientes entre otras situaciones donde se requiera de un respaldo reciente.