UNIVERSIDAD POLITÉCNICA DE PUEBLA
Ingeniería en Informática

Proyecto de Estadía Profesional

“Centralización y respaldo de folios fiscales”

Área temática del CONACYT: VII
Ingenierías y tecnologías

Presenta:
José Antonio Xochimitl Tlamani

Asesor técnico
Ing. Modesto Romero Martínez

Asesor académico
M. C. Rebeca Rodríguez Huesca

Juan C. Bonilla, Puebla, México. 19 de Diciembre 2018

Resumen

El presente documento muestra la información obtenida a lo largo de la estadía profesional en la cual se llevó a cabo una recopilación de elementos que fueron clave para generar los procedimientos, algunos de éstos son las herramientas informáticas y servicios, también se generó una explicación de la metodología que sirvió de apoyo para poder concluir con el desarrollo del proyecto titulado *centralización y respaldo de folios fiscales*, y así poder alcanzar los objetivos que fueron descritos dentro del mismo. Cabe mencionar que el funcionamiento de lo desarrollado optimiza el proceso y mantiene una mejor administración de los datos.

Índice

Índice de figuras

Índice de Tablas

1. Introducción

Para crear software es necesario plantearse algunas interrogativas que ayuden a conocer el alcance del desarrollo, así como tener un fin o una meta, es por ello que a lo largo de este capítulo se describen las problemáticas o necesidades que se pretenden abordar con este proyecto, también, se explica el por qué implementar esta solución, el objetivo general y específicos para poder llegar al resultado deseado.

1.1. Descripción del problema o necesidad

En una cadena de tiendas se genera una gran cantidad de información desde proveedores hasta folios fiscales (tickets de venta) todos los días, estos datos se encuentran guardados en un centro de almacenamiento, también cada tienda mantiene los folios fiscales en archivos de texto de esta forma la información no sería segura y no podrían ser consultados a nivel general por algún interesado, por otro lado ya que se generan todos los días más folios y su conservación es limitada, de manera que se desea consultar los folios de años anteriores es prácticamente imposible.

1.2. Justificación

De acuerdo a la problemática descrita anteriormente para solucionar dicho problema es necesario desarrollar un conjunto de procedimientos que se ejecuten automáticamente para realizar el envío de datos de folios fiscales hacia un centro de almacenamiento de información donde posteriormente puedan ser consultado mediante un sistema que los recupere en formato de folio fiscal.
Las herramientas y recursos que se encuentran disponibles para el desarrollo de estos procedimientos son: un espacio de trabajo, el software necesario y bases de datos de pruebas.

1.3. Objetivo General y Específicos

Objetivo general

Desarrollar procedimientos para el respaldo de folios fiscales de una cadena de tiendas comerciales en un repositorio a través del uso de herramientas informáticas de desarrollo.

Objetivos específicos

1. Analizar la especificación funcional de los procedimientos de respaldo de información.
2. Elaborar la especificación técnica para el desarrollo.
3. Diseñar las interfaces en la herramienta de webMethods.
4. Realizar pruebas de funcionamiento.

2. Metodología y herramientas

En este capítulo se presentará la fundamentación teórica en que se basa el desarrollo del proyecto, la cual incluirá la descripción de la metodología empleada, las características, ventajas y desventajas de las herramientas tecnológicas utilizadas, así como las razones por las cuales se emplearon estas. La metodología RAD (Rapid application development o desarrollo rápido de aplicaciones) se aplicará para realizar este proyecto fue seleccionada de acuerdo a la manera en que se desarrollan los proyectos dentro de la empresa.

2.1. Descripción de la metodología empleada

El desarrollo rápido de aplicaciones (DRA) es un método para el desarrollo de sistemas de información, creado por James Martin en 1980; se basa en la creación de prototipos.

El desarrollo de ***prototipos*** está pensado para sistemas de información de tamaño pequeño o mediano. Un ***prototipo*** es un sistema de información a pequeña escala que permite descubrir cuáles son las necesidades de los usuarios. [1]

La metodología se divide en 5 fases o procesos, *véase la Figura 1.*

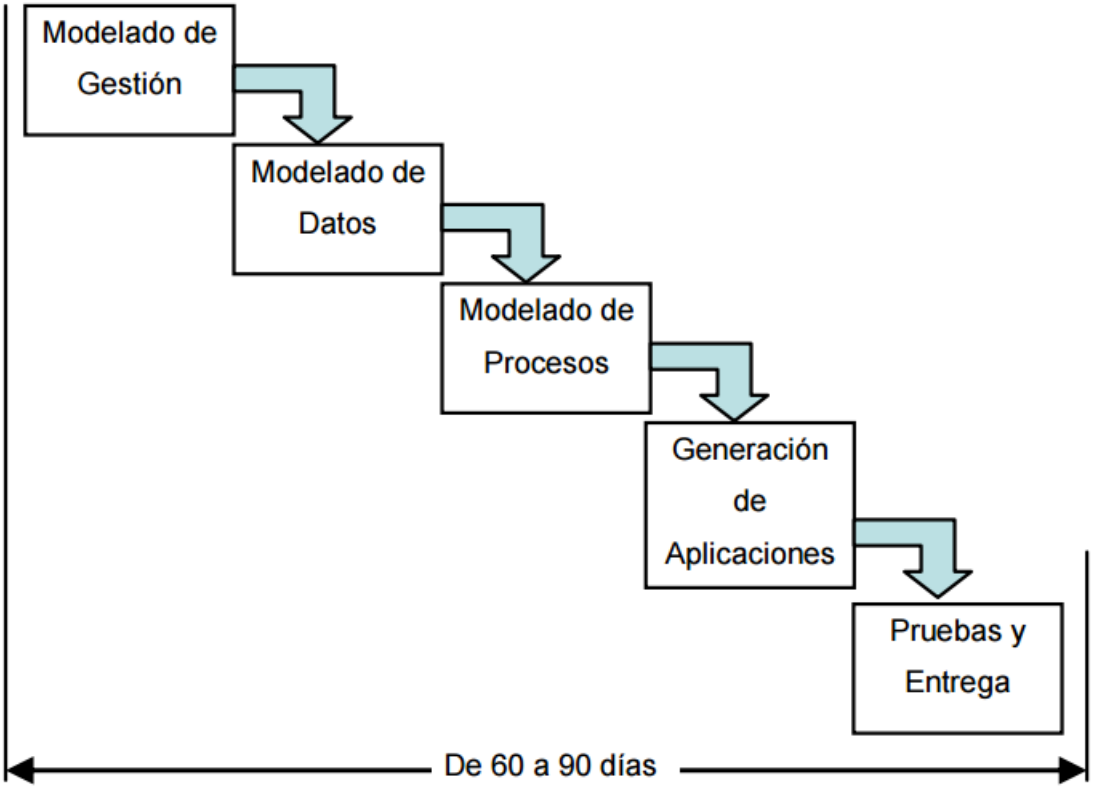


Figura 1 Modelo DRA

El Modelo DRA comprende las siguientes etapas:

Modelado de gestión. El flujo de información entre las funciones de gestión se modela en base a las siguientes preguntas: – ¿Qué información conduce el

proceso de gestión? – ¿Qué información genera? – ¿A dónde va la información? – ¿Quién la procesa?

Modelado de datos. el flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.

Modelado del proceso. los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.

Generación de aplicaciones. El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

Pruebas y entrega. Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo. [2]

Ventajas

- Los entregables son más fáciles de trasladas a otras plataformas.
- Visibilidad de resultados temprana.
- Permite menor codificación manual a nivel programador.
- Requiere un involucramiento de los usuarios dentro del desarrollo.
- Los ciclos de desarrollos son cortos.
- Mayor flexibilidad.

Desventajas

- Costo de herramientas integradas y equipo necesario.
- Prototipo no puede escalar.
- Dependencia en componentes de terceros.

2.2. Herramientas tecnológicas

Para llevar a cabo el desarrollo de este proyecto se utilizó el sistema operativo Windows 10, webMethods como integrador de procesos, software AG designer como ***IDE*** para el desarrollo de las interfaces, Oracle ***SQL*** developer para realizar consultas a la base de datos, editores de archivos para realizar la documentación, Integration Server que ayuda a la comunicación con las interfaces y Database component configurator para la creación de tablas.

A continuación, se describe detalladamente cada herramienta que se utilizó para la culminación del proyecto.

2.2.1. WebMethods

WebMethods es una ***suite*** de integración de procesos de negocio a nivel empresarial hecha en ***java***, también conocidas como Enterprise Application Integration (EAI). Fue desarrollado por la compañía del mismo nombre fundada en el año 1996 en Fairfax, Virginia. Más tarde adquirida por la empresa alemana Software AG. [3]

Ventajas

- Da una plataforma para poder gestionar por completo el Gobierno SOA
- Modernización de Sistemas Heredados Gobierno SOA Produce políticas, procesos y procedimientos para controlar el desarrollo, despliegue y gestión de servicios.
- Gobierno uniforme de los activos de TI

Desventajas

- Contiene complementos que son de paga.

2.2.2. Software AG Designer

Para el desarrollo en webMethods se usa un entorno de desarrollo integrado (***IDE***) basado en eclipse, llamado Software AG Designer, el cual cuenta con diferentes perspectivas optimizadas para cada actividad que se realice.
En el designer se configura la conexión a ambos servidores. Los ***portlets*** y modelos son desarrollados en la carpeta local del workspace[1], para una vez que se encuentran terminados desplegarlos a MWS[2]. Mientras que lo servicios son desarrollados y editados directamente en el Integration Server. [3] *Véase La figura 2. Entorno de desarrollo de Software AG Designer*.

[1] Espacio de trabajo
[2] Servidor con interfaz para la administración de páginas y su acceso a estas.

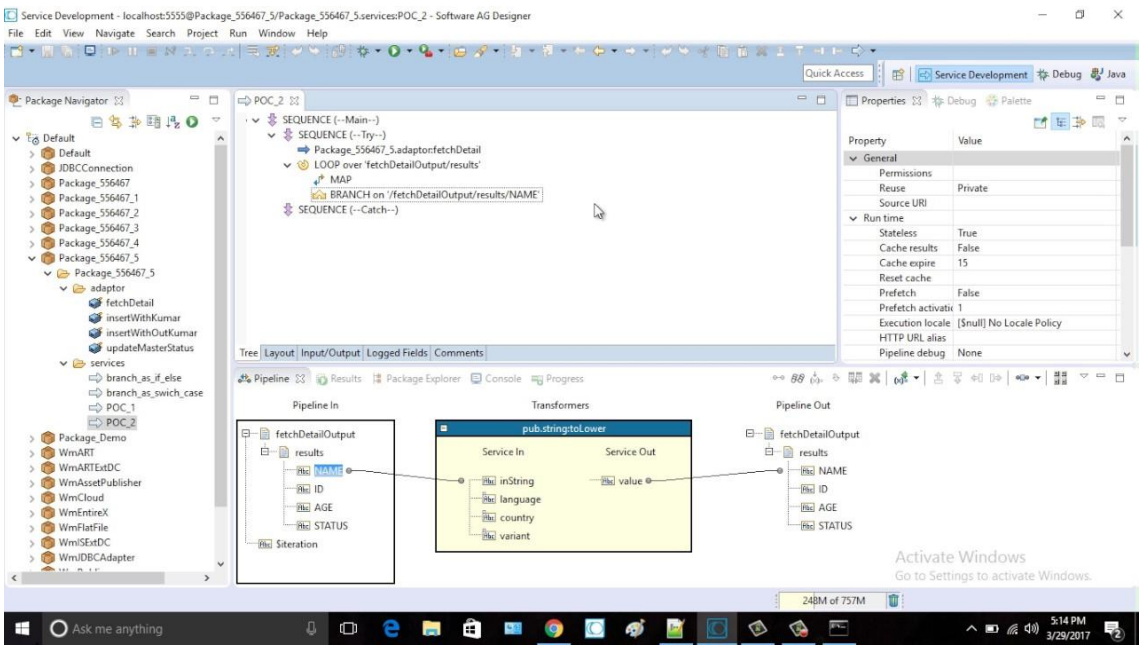


Figura 2 Entorno de desarrollo

Ventajas

- Extienda rápidamente las aplicaciones y procesos de negocio a dispositivos móviles
- Desarrolla aplicaciones móviles una vez, ejecuta de forma nativa en múltiples plataformas
- Integre de forma segura las aplicaciones móviles con sus sistemas back-end.

Desventajas

- Los desarrollos son más robustos por el uso de paquetes.

2.2.3. Integration Server

Integration server es uno de sus dos principales servidores de aplicación con los que cuenta la plataforma de webMethods, es encargado de almacenar e interpretar los servicios creados, crear las conexiones de a bases de datos que son utilizados por los servicios, hacer migraciones entre servidores, administrar paquetes entre otras cosas.

Se usa el concepto de adaptadores algunos incluidos dentro del entorno y otros que pueden ser personalizados, estos ayudan a la integración de sistemas de terceros, como lo son JDBC[3], JMS[4], ***SAP*** entre otros.[3] *Véase la figura 3. Inicio de Integration Server*.

[3] Conectividad a bases de datos de Java.
[4] Servicios de mensajería de Java.

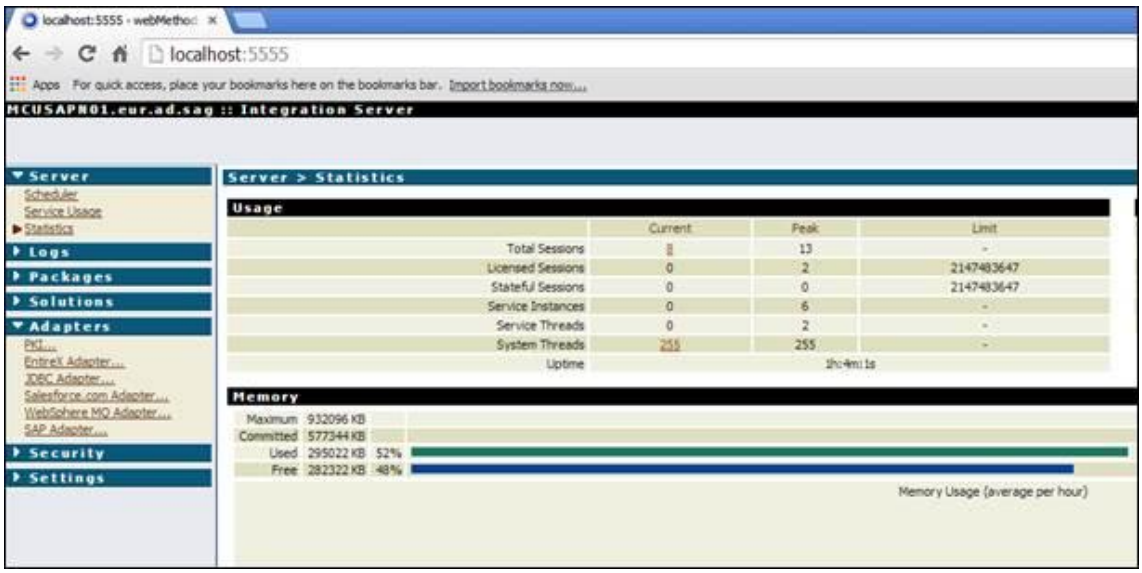


Figura 3 Inicio de Integration Server

2.2.4. Oracle SQL Developer

Oracle ***SQL*** Developer es un entorno de desarrollo integrado (***IDE***) para trabajar con ***SQL*** en bases de datos Oracle. Oracle Corporation proporciona este producto gratis; utiliza el kit de desarrollo de ***Java***.[4] *Véase la Figura 4. IDE de Oracle **SQL** Developer.*

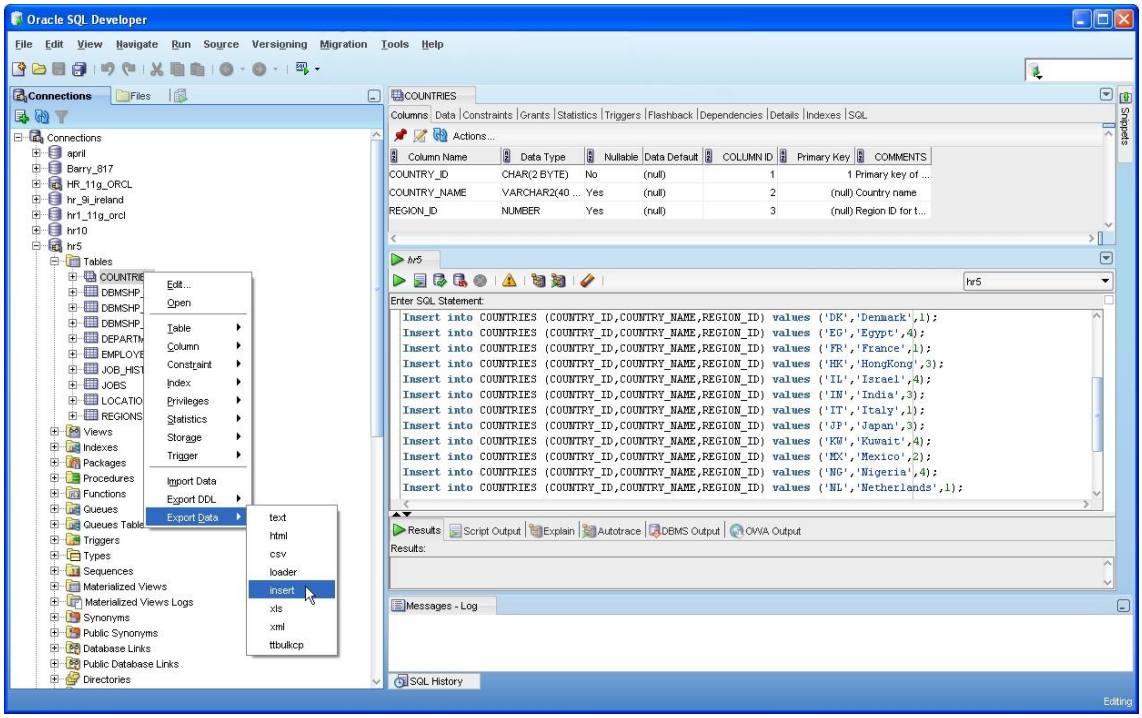


Figura 4 IDE de Oracle SQL Developer

Ventajas

- Sistema de gestión y control centralizado.
- Estandarización.
- Motor de base de datos objeto-relación.

Desventajas

- Incompatibilidad y complejidad.
- Funcionalidad limitada.

2.2.5. Database Component Configurator

Utilizado para instalar los componentes de base de datos necesarios, es decir cada función requiere un conjunto especifico de objetos de base de datos para el esquema, véase la *Figura 5. Entorno de database component configurator.*

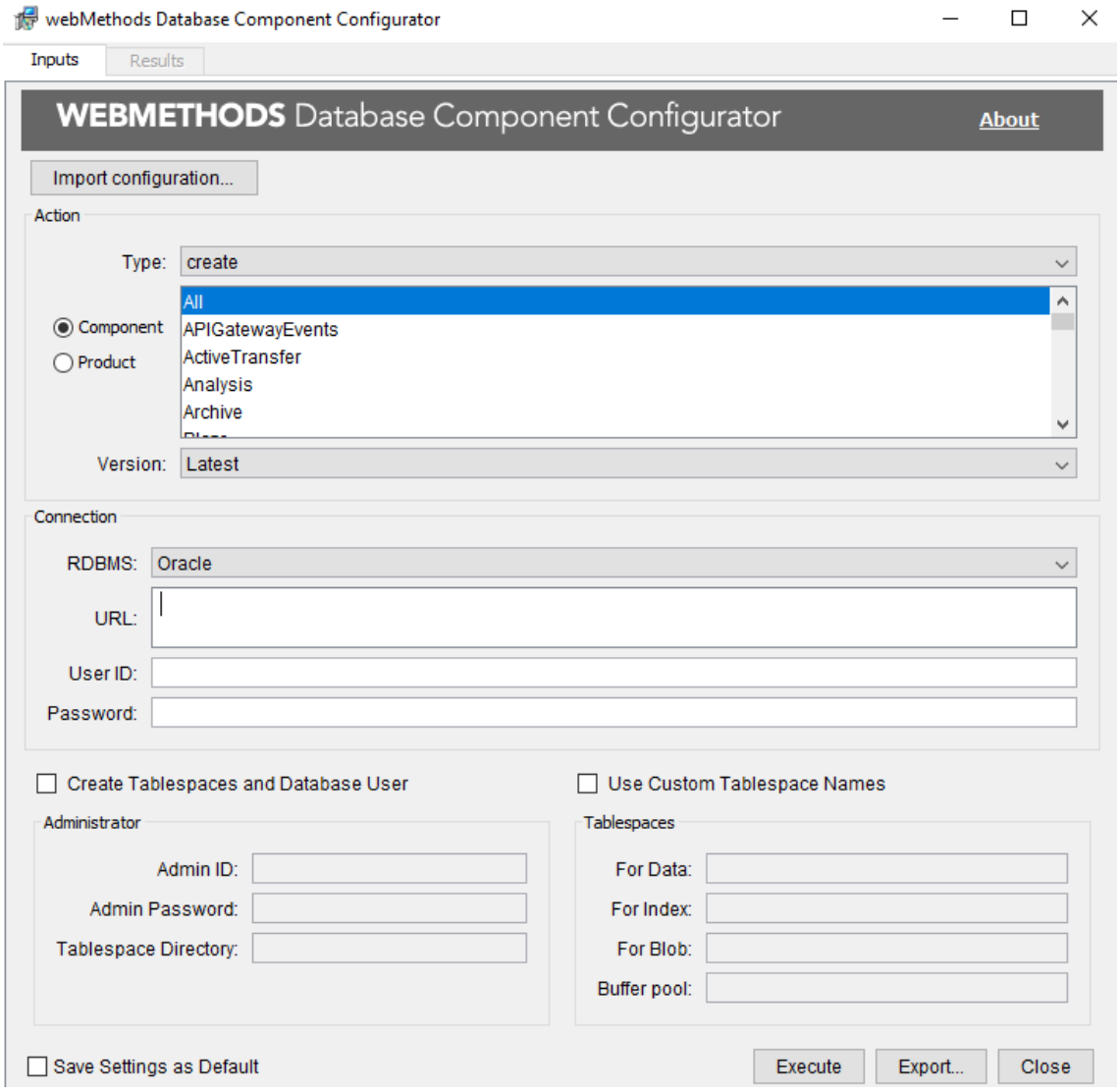


Figura 5 Entorno de Database component configurator

2.2.6. DIA

DIA es una aplicación informática para la creación de diferentes tipos de diagramas, como pueden ser, entidad-relación, UML[5], de flujo, de redes, de circuitos eléctricos, etc. Permite la exportación de los diagramas en formato EPS, SVG, PNG, DXF, CGM, WMF, JPEG y VDX.[5] *Veáse la Figura 6 Entorno de DIA.*

[5] Lenguaje de modelado unificado

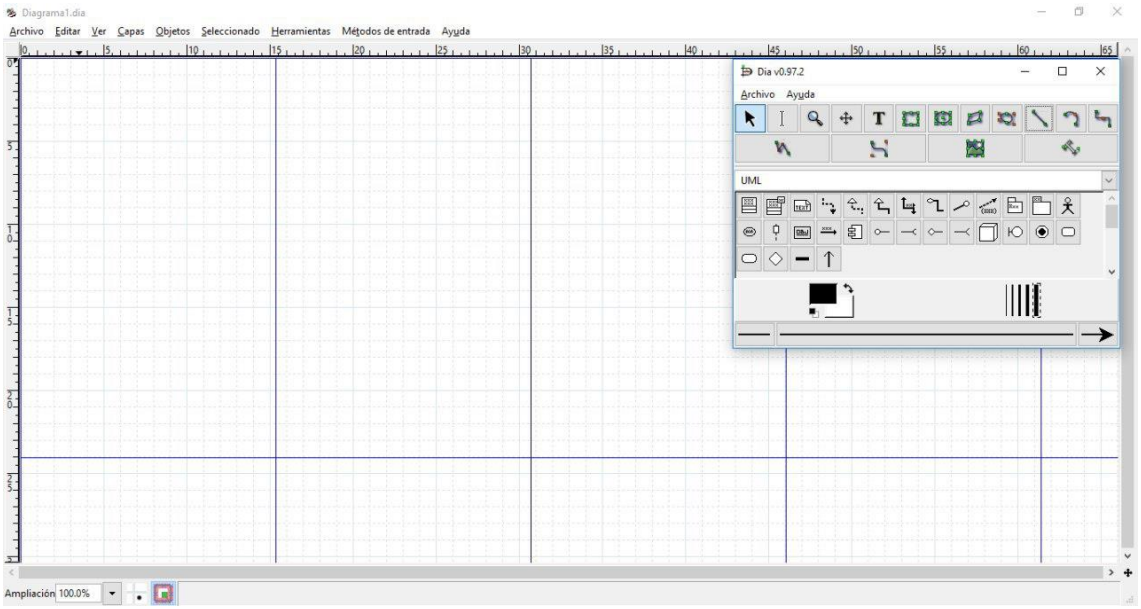


Figura 6 Entorno de DIA

3. Resultados

A lo largo de este capítulo se muestran los resultados obtenidos durante el desarrollo del proyecto, estos son mostrados en orden cronológico, en el cual se plantea en la metodología que se está usando.

3.1. Modelado de gestión

Para facilitar el mantenimiento del proceso de respaldo de folios fiscales fue necesario dividir el desarrollo en cinco interfaces o procedimientos, de los cuales estarán nombrados de la siguiente manera, PB7, PB8, PB9, PB10, PB11.

3.1.1. Definiciones, acrónimos y abreviaciones

Para un mejor entendimiento de este capítulo se agregó una tabla con algunas abreviaciones que son usadas repetidamente, *véase la Tabla 1 definiciones, acrónimos y abreviaturas.*

Abreviación	Descripción
WM	Web Methods
POSREP	Sistema de consulta de POS, para reportes.
WEB	Aplicaciones WEB
POS	Punto de venta

Tabla 1 Definiciones, acrónimos y abreviaturas

3.1.2. Arquitectura del proceso de negocio

Como antes se mencionaba, fue necesario dividir el proceso en cinco interfaces de las cuales realizarán una acción diferente a las vistas o tablas en cada una de las bases de datos, *véase la Figura 7 Arquitectura del proceso de negocio para el respaldo de folios fiscales* para un mejor entendimiento.

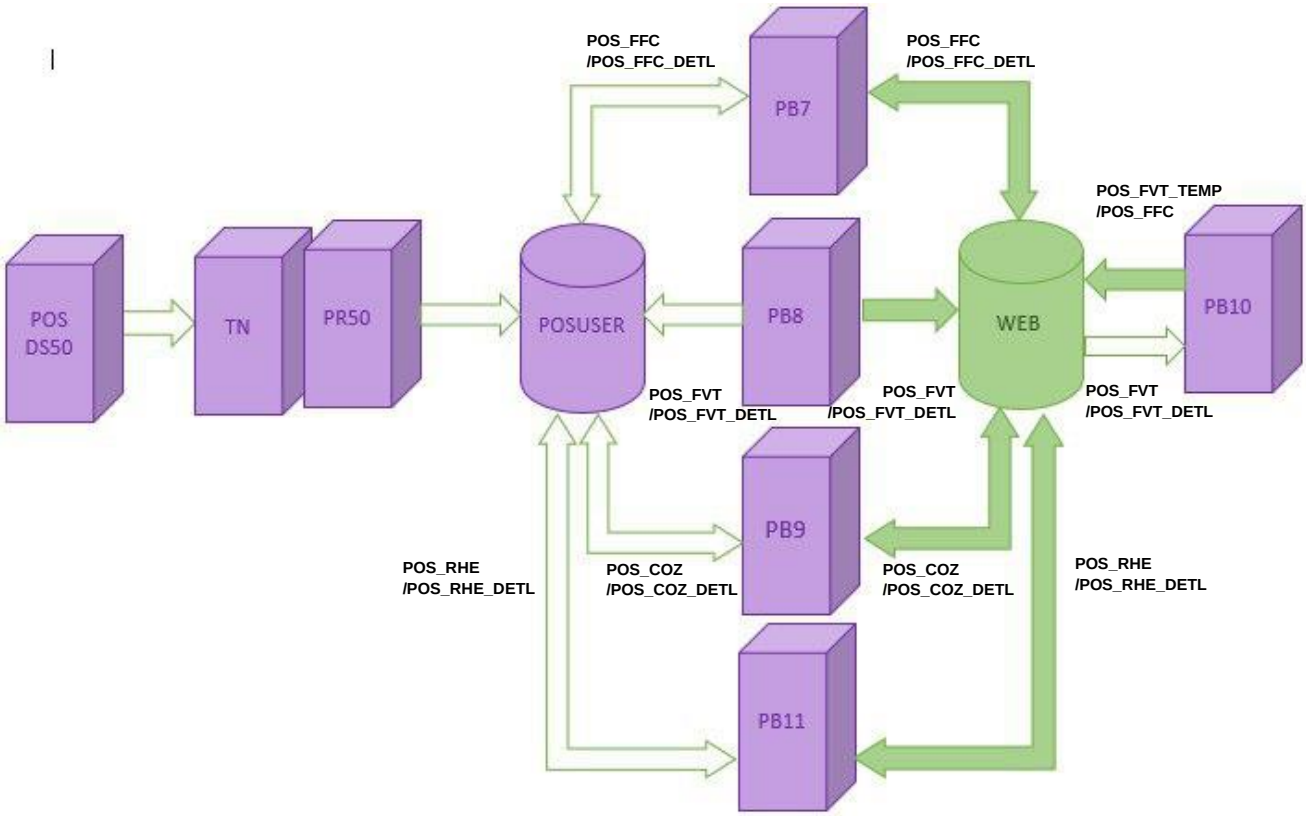


Figura 7 Arquitectura del proceso de negocio para el respaldo de folios fiscales

3.1.3. Requerimientos funcionales

De acuerdo a la función de cada interfaz es necesario listas los requerimientos funcionales de los cuales fueron asignados a cada una de las interfaces de acuerdo a las tablas o vistas que ocupa cada una, *véase la Tabla 2 requerimientos funcionales específicos por interfaz.*

ID	*Descripción*	*Interfaz*
RF1	Obtener los datos de folios fiscales pendientes de procesar en POSUSER y resguardarlo en POSREP	PB7
RF2	Agrupar los datos por: número de factura, fecha, número de resolución, folio fiscal.	PB7
RF3	Obtener los datos de Folios de ventas pendientes de procesar en POSUSER y resguardarlo en POSREP	PB8
RF4	Obtener los datos de cortes Z pendientes de procesar en POSUSER y resguardarlo en POSREP	PB9
RF5	Separación de archivo de folios fiscales por folios individuales.	PB10
RF6	Almacenar cada folio individual en base de datos identicados por su folio de venta	PB10

RF7	Asignación de cada folio individual a su contraparte en base de datos de POSREP, con su correspondiente folio de venta tal como fue emitido en la tienda.	PB10

Tabla 2 Requerimientos funcionales específicos por interfaz

Es necesario conocer que información de la base de datos se requiere para los tres tipos de formatos para respaldar, es decir para los tickets de venta, corte Z y hojas de entrega final, *véase la Tabla 3 Requerimiento funcial genera* que muestra dicha información por recuperar y respaldar.

RF-0001	**Generar Información de los folios de venta en POS**

La información de los folios fiscales, tales como Tickets de Venta, Cortes Z y Hojas de Entrega Final, deberá generarse desde el punto de venta, la información deberá enviarse todos los días al final del día.

Para el Ticket de Venta se requiere enviar la información de:
- Plaza
- Tienda
- Fecha de movimiento
- Contenido del ticket (ticket file)
- Folio Fiscal (Numevo campo para la estructura actual)

Para el corte Z se requiere enviar información de:
- Plaza
- Tienda
- Caja
- Fecha de movimiento
- Contenido del corte Z (COZ file)
- N° de corte Z

Para la Hoja de Entrega Final se requiere enviar información de:
- Plaza (Capos nuevos)
- Tienda (Campos nuevos)
- Fecha de movimiento
- Contenido de la hoja de entrega final (RHE file)

Tabla 3 Requerimiento funcional general

3.1.4. Reglas y funciones de negocio

Las reglas o funciones de negocio son vitales para alcanzar el objetivo que se desea alcanzar es por eso que gracias a los requerimientos funcionales y a las tablas que son consultadas se asignan estas reglas, están nombradas con un ID, una descripción y a que interfaz pertenece, *véase la Tabla 4 Reglas y funciones de negocio.*

ID	*Descripción*	*Interfaz*
RN1	Los datos que serán copiados deben ser sólo los que se encuentren con estatus I y de tipo FFC	PB7
RN2	Una vez insertados los datos en POSREP se debe actualizar el estatus a E (Enviado) en POSUSER para evitar que sean reprocesados	PB7
RN3	Los datos que serán copiados deben ser sólo los que se encuentren con status I y de tipo FVT	PB8
RN4	Una vez insertados los datos en POSREP se debe actualizar el status a E (Enviado) en POSUSER para evitar que sean reprocesados	PB8
RN5	Los datos que serán copiados deben ser sólo los que se encuentren con status I y de tipo COZ	PB9
RN6	Una vez insertados los datos en POSREP se debe actualizar el status a E (Enviado) en POSUSER para evitar que sean reprocesados	PB9
RN7	Los tickets que serán separados deberán ser sólo los que se encuentren con status I, y se encuentren insertados en POSREP.	PB10
RN8	Se debe validar que para el ticket separado de la lista, se encuentre su detalle en POSREP en la tabla de Folios Fiscales.	PB10
RN9	Antes de actualizar el registro del Folio fiscal en POSREP, con el ticket emitido en la tienda, verificar que el estatus se encuentre en pendiente.	PB10
RN10	Se deben controlar el máximo de intentos para actualizar el registro de Folio Fiscal	PB10

Tabla 4 Reglas y funciones de negocio

3.2. Modelado de datos

Debido que la cadena de tiendas tiene definida una base de datos y que solamente se hará uso de los datos por medio de las interfaces, sólo se tomarán unas tablas para cumplir el objetivo del procedimiento.

3.2.1. Tablas de PB7

Para realizar el procedimiento de la interfaz PB7 es necesario conocer las tablas que se ocuparán y que acción se realizará en cada una, *véase la Tabla 5 Tipos de acceso a las tablas de la interfaz PB7* donde se enuncian las tablas usadas,

también *véase la Figura 8 Diagrama entidad Relacion de interfaz PB7* que muestra las tablas en formato ER.

BD Alias	Tipo de objeto	Nombre vista, tabla o secuencia	Tipo de acceso
DBS_POSUSER	TABLA	POS_INBOUND_DOCS	SELECT, UPDATE
DBS_POSUSER	TABLA	POS_FFC	SELECT
DBS_POSUSER	TABLA	POS_FFC_DETL	SELECT
DBS_WEB_XXRFCO	TABLA	XXRFCO_POS_FFC	INSERT
DBS_WEB_XXRFCO	TABLA	XXRFCO_POS_FFC_DETL	INSERT

Tabla 5 Tipos de acceso a las tablas de la interfaz PB7

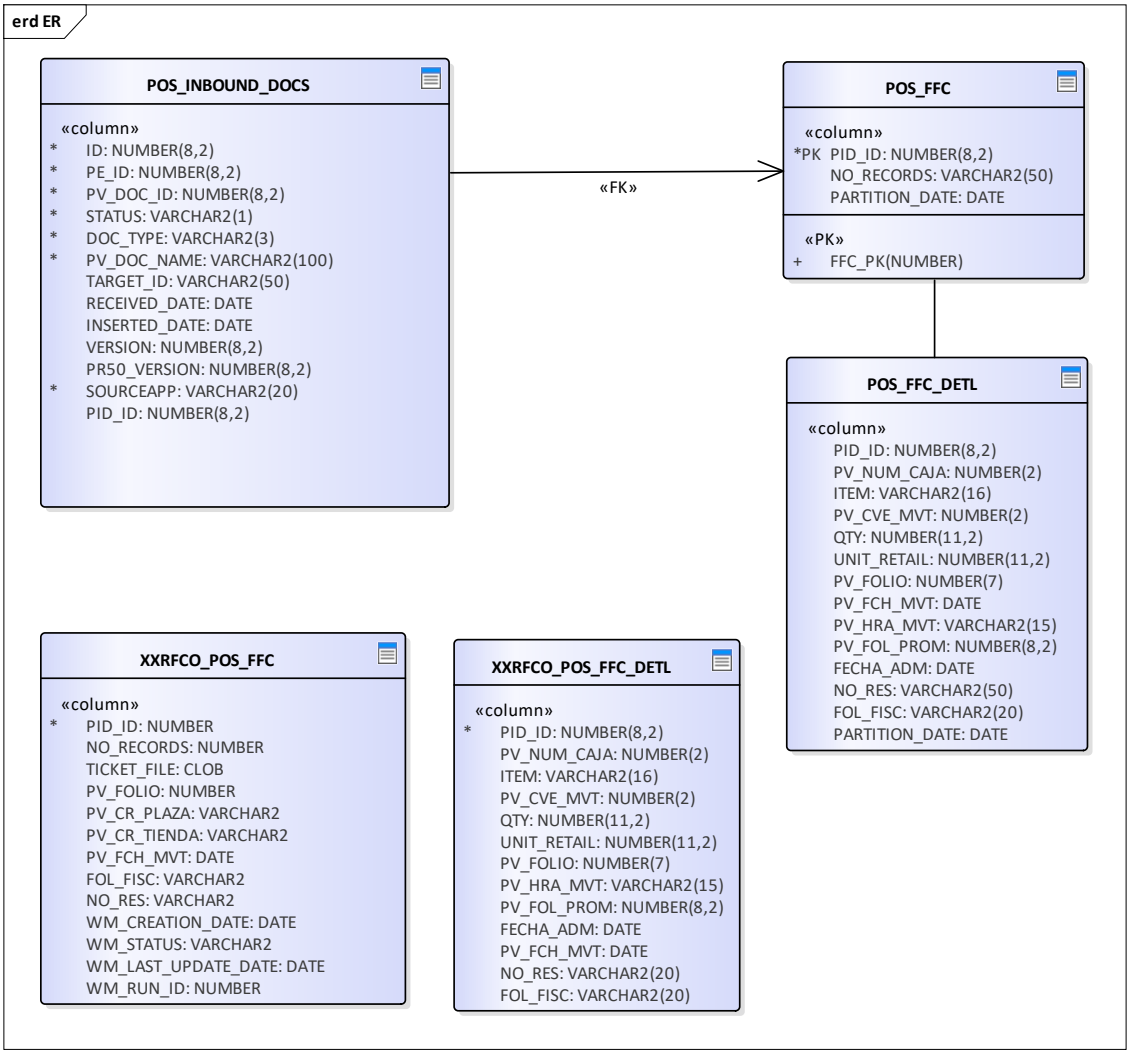


Figura 8 Diagrama entidad-relación de interfaz PB7

3.2.2. Tablas de PB8

Para realizar el procedimiento de la interfaz PB8 es necesario conocer las tablas que se ocuparán y que acción se realizará en cada una, *véase la Tabla 6 Tipos de acceso a las tablas de la interfaz PB8* donde se enuncian las tablas usadas,

también *véase la Figura 9 Diagrama entidad Relacion de interfaz PB8* que muestra las tablas en formato ER.

BD Alias	Tipo de objeto	Nombre vista, tabla o secuencia	Tipo de acceso
DBS_POSUSER	TABLA	POS_INBOUND_DOCS	SELECT, UPDATE
DBS_POSUSER	TABLA	POS_FVT	SELECT
DBS_POSUSER	TABLA	POS_FVT_DETL	SELECT
DBS_XXRFCO	TABLA	XXRFCO_POS_FVT	INSERT
DBS_XXRFCO	TABLA	XXRFCO_POS_FVT_DETL	INSERT

Tabla 6 Tipos de acceso a las tablas de la interfaz PB8

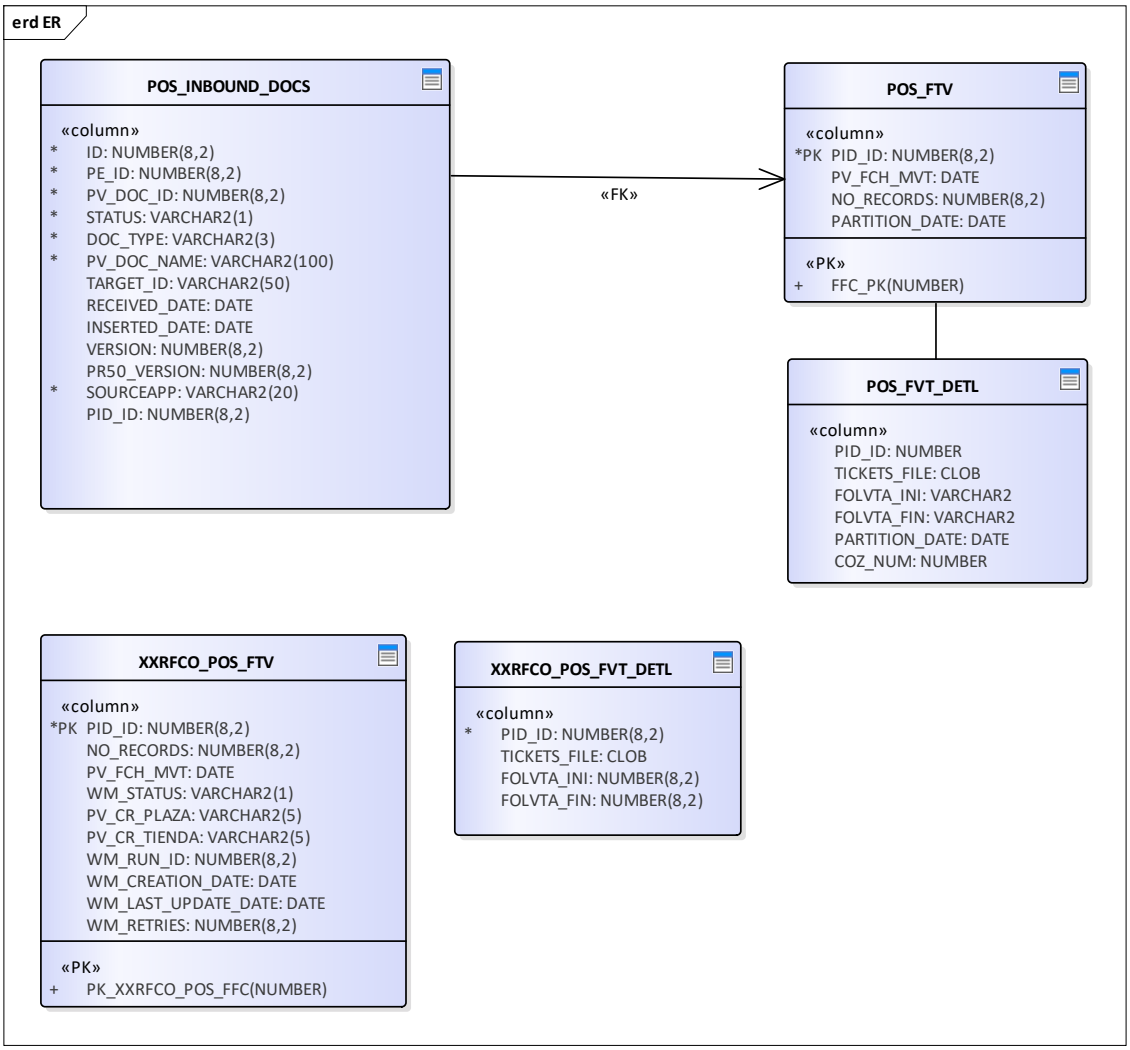


Figura 9 Diagrama entidad-relación de interfaz PB8

3.2.3. Tablas de PB9

Para realizar el procedimiento de la interfaz PB9 es necesario conocer las tablas que se ocuparán y que acción se realizará en cada una, *véase la Tabla 7 Tipos de acceso a las tablas de la interfaz PB9* donde se enuncian las tablas usadas,

también *véase la Figura 10 Diagrama entidad Relacion de interfaz PB9* que muestra las tablas en formato ER.

BD Alias	Tipo de objeto	Nombre vista, tabla o secuencia	Tipo de acceso
DBS_POSUSER	TABLA	DOS_INBOUND_DOCS	SELECT, UPDATE
DBS_POSUSER	TABLA	POS_COZ	SELECT
DBS_POSUSER	TABLA	POS_COZ_DETL	SELECT
DBS_XXRFCO	TABLA	XXRFCO_POS_COZ	INSERT
DBS_XXRFCO	TABLA	XXRFCO_POS_COZ_DETL	INSERT

Tabla 7 Tipos de acceso a las tablas de la interfaz PB9

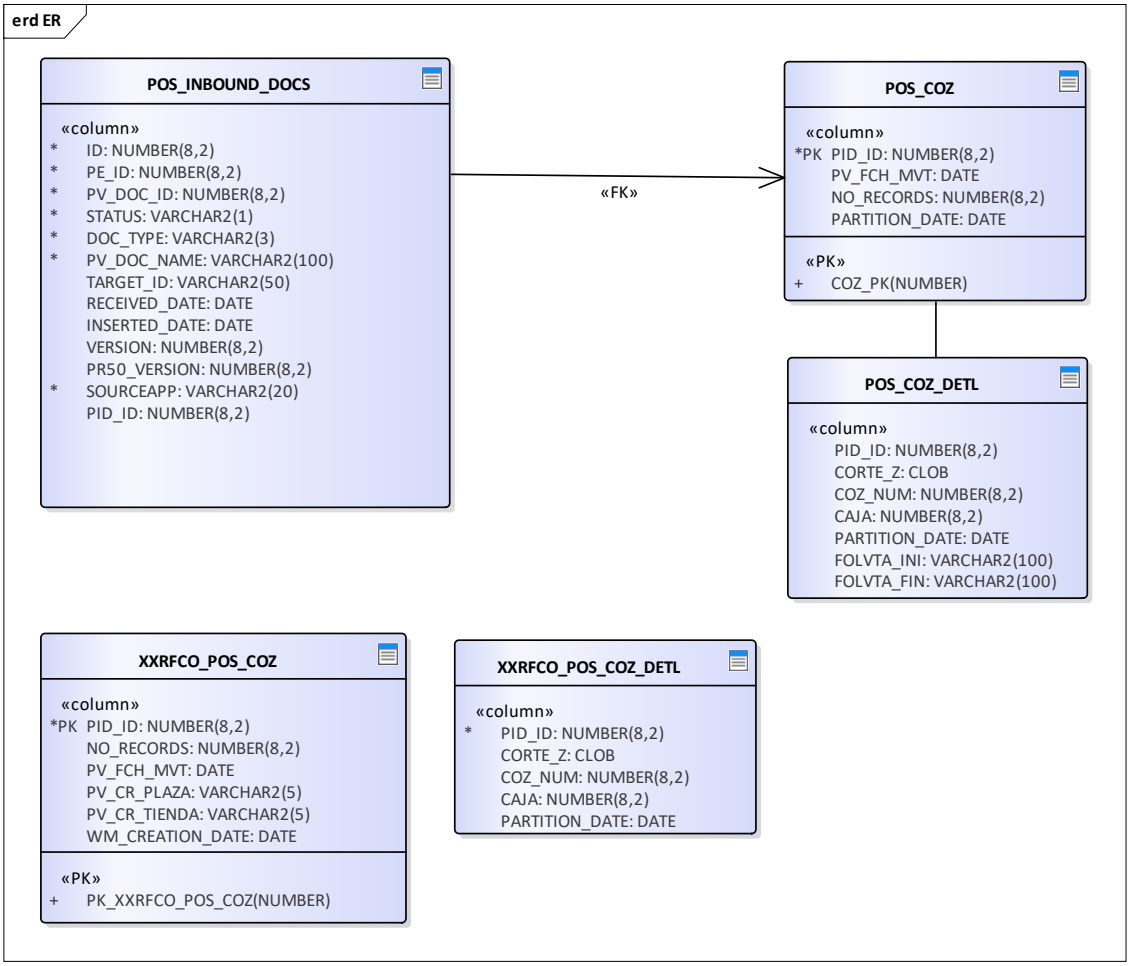


Figura 10 Diagrama entidad-relación de interfaz PB9

3.2.4. Tablas de PB10

Para realizar el procedimiento de la interfaz PB10 es necesario conocer las tablas que se ocuparán y que acción se realizará en cada una, *véase la Tabla 8*

Tipos de acceso a las tablas de la interfaz PB10 donde se enuncian las tablas usadas.

BD Alias	Tipo de objeto	Nombre vista, tabla o secuencia	Tipo de acceso
DBS_POSUSER	TABLA	POS_FVT	SELECT, UPDATE
DBS_POSUSER	TABLA	POS_FVT_DETL	SELECT
DBS_POSUSER	TABLA	POS_FVT_TEMP	INSERT, DELETE, SELECT, UPDATE
DBS_POSREP	TABLA	POS_FFC	SELECT, UPDATE

Tabla 8 Tipos de acceso a las tablas de la interfaz PB10

3.2.5. Tablas de PB11

Para realizar el procedimiento de la interfaz PB11 es necesario conocer las tablas que se ocuparán y que acción se realizará en cada una, *véase la Tabla 9 Tipos de acceso a las tablas de la interfaz PB11* donde se enuncian las tablas usadas.

BD Alias	Tipo de objeto	Nombre vista, tabla o secuencia	Tipo de acceso
WM/POSUSER	TABLE	POS_INBOUND_DOCS	SELECT, UPDATE
WM/POSUSER	TABLE	POS_RHE_HEAD	SELECT
WM/POSUSER	TABLE	POS_RHE_DETL	SELECT
WM/POSREP	TABLE	POS_RHE_HEAD	SELECT

Tabla 9 Tipos de acceso a las tablas de la interfaz PB11

3.2.6. Simbología de estatus

Los datos dentro de la base de datos contienen un campo de el estatus en el que se encuentra, dentro de este es representado con una letra de las cuales tienen un significado particular, *véase la Tabla 10 Significado de letras del campo estatus* donde muestra los estatus o estados que puede tener un registro.

R	Recibido: el documento se esta procesando
W	En ciclo de reintentos
O	Deprecado: la versión del documento esta deprecada y la cantidad de reenvíos es inferior al máximo.
I	Insertado: el documento ha sido procesado con éxito.
F	Inválido: el formato del documento esta incorrecto o el documento es válido pero se ha excedido la cantidad máxima de reenvíos.
M	En error: el documento no se ha podido insertar con éxito y la cantidad de reenvíos es inferior al máximo.

Tabla 10 Significado de letras del campo estatus

3.3. Modelado de procesos

Ya que el desarrollo se encuentra divido en varias interfaces, fue necesario realizar un diagrama de flujo por cada una, a continuación se describe cada proceso.

En la figura 11 se muestra el proceso que seguirá la interfaz PB7 para realizar parte del procedimiento del desarrollo.

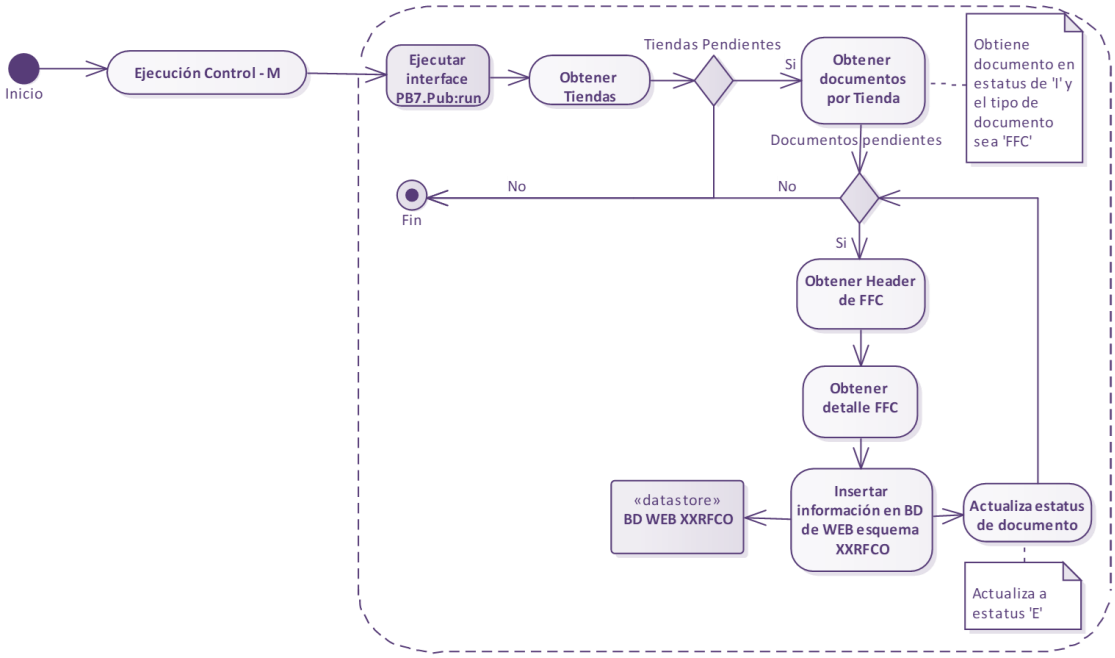


Figura 11 Diagrama de flujo de la interfaz PB7

Para un mejor entendimiento se agrega la descripción del funcionamiento del anterior diagrama.

Control-M ejecuta la interfaz PB7, por medio del job: PB7.Pub: run, una vez que se inicia la ejecución de la interfaz se obtiene el listado de las tiendas que tienen documentos ‘FFC’ pendientes por procesar en estatus de ‘I’. Una vez que obtiene el listado de tiendas pendientes, por cada una de ellas, se levanta un thread en el cual para esa tienda se obtienen los documentos pendientes de procesar que sean de tipo ‘FFC’ y en estatus de ‘I’. Ya que cuenta con los documentos faltantes de dicha tienda, se inicia un ciclo el cual por cada documento obtiene los datos del encabezado de la tabla POS_FFC y su respectivo detalle de la tabla POS_FFC_DETL. Una vez que cuenta con la información del documento, realiza la inserción de la información por medio de un Batch Insert a las tablas de XXRFCO_POS_FFC y XXRFCO_POS_FFC_DETL. Una vez que se guardó la información en la base de datos de WEB, realiza una actualización del estatus en la tabla POS_INBOUND_DOCS sobre el registro del documento procesado, cambiándolo a estatus de ‘E’. En vista que realizó la actualización del registro, se valida si cuenta con más documentos pendientes de procesar, de lo contrario, finaliza el flujo de la interfaz.

En la figura 12 se muestra el proceso que seguirá la interfaz PB8 para realizar parte del procedimiento del desarrollo.

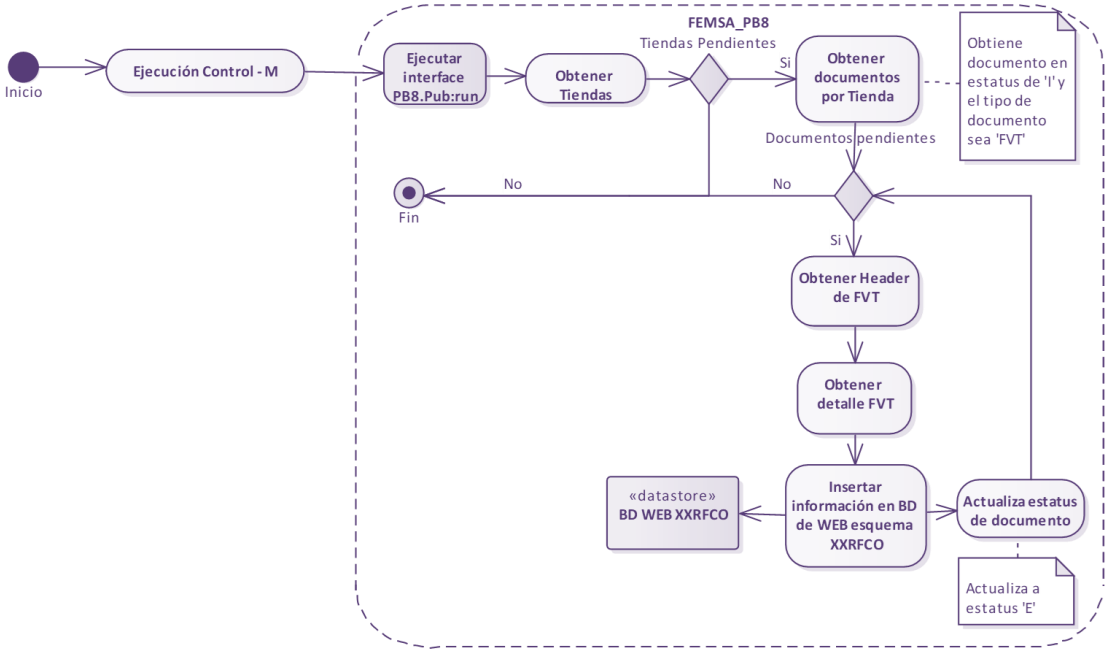


Figura 12 Diagrama de flujo de la interfaz PB8

Para un mejor entendimiento se agrega la descripción del funcionamiento del anterior diagrama.

Control-M ejecuta la interfaz PB8, por medio del job: PB8.Pub: run, una vez que se inicia la ejecución de la interfaz se obtiene el listado de las tiendas que tienen documentos ‘FVT’ pendientes por procesar en estatus de ‘I’. Una vez que obtiene el listado de tiendas pendientes, por cada una de ellas, se levanta un thread en el cual para esa tienda se obtienen los documentos pendientes de procesar que sean de tipo ‘FVT’ y en estatus de ‘I’. Ya que cuenta con los documentos pendientes de dicha tienda, se inicia un ciclo el cual por cada documento obtiene los datos del encabezado de la tabla POS_FVT y su respectivo detalle de la tabla POS_FVT_DETL. Una vez que cuenta con la información del documento, realiza la inserción de la información por medio de un Batch Insert a las tablas de XXRFCO_POS_FVT y XXRFCO_POS_FVT_DETL. Ya que la información se guardó en la base de datos de WEB, realiza una actualización del estatus en la tabla POS_INBOUND_DOCS sobre el registro del documento procesado, cambiándolo a estatus de ‘E’. En vista que se realizó la actualización del registro, se valida si cuenta con más documentos pendientes de procesar, de lo contrario, finaliza el flujo de la interfaz.

En la figura 13 se muestra el proceso que seguirá la interfaz PB9 para realizar parte del procedimiento del desarrollo.

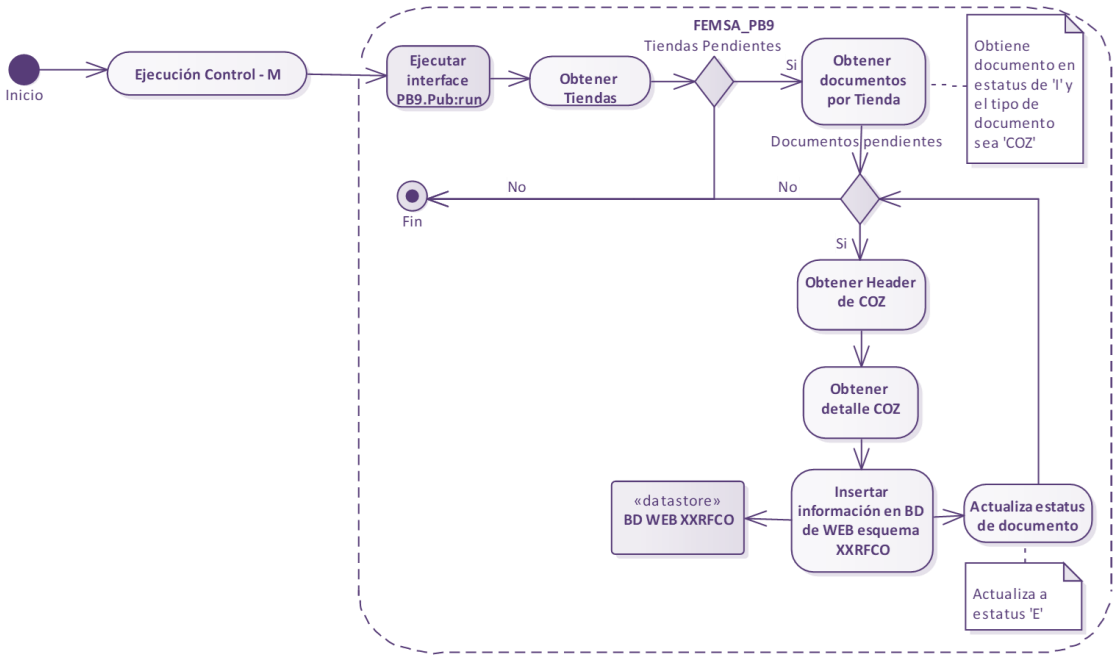


Figura 13 Diagrama de flujo de la interfaz PB9

Para un mejor entendimiento se agrega la descripción del funcionamiento del anterior diagrama.

Control-M ejecuta la interfaz PB9, por medio del job: PB9.Pub: run, una vez que se inicia la ejecución de la interfaz se obtiene el listado de las tiendas que tienen documentos ‘COZ’ pendientes por procesar en estatus de ‘I’. Una vez que obtiene el listado de tiendas pendientes, por cada una de ellas, se levanta un thread en el cual para esa tienda se obtienen los documentos pendientes de procesar que sean de tipo ‘COZ’ y en estatus de ‘I’. Ya que cuenta con los documentos pendientes de dicha tienda, se inicia un ciclo el cual por cada documento obtiene los datos del encabezado de la tabla POS_COZ y su respectivo detalle de la tabla POS_COZ_DETL. Una vez que cuenta con la información del documento, realiza la inserción de la información por medio de un Batch Insert a las tablas de XXRFCO_POS_COZ y XXRFCO_POS_COZ_DETL. Una vez que se guardó la información en la base de datos de WEB, realiza una actualización del estatus en la tabla POS_INBOUND_DOCS sobre el registro del documento procesado, cambiándo a estatus ‘E’. Ya que se realizó la actualización del registro, se valida si cuenta con más documentos pendientes, de lo contrario, finaliza el flujo de la interfaz.

En la figura 14 se muestra el proceso que seguirá la interfaz PB10 para realizar parte del procedimiento del desarrollo.

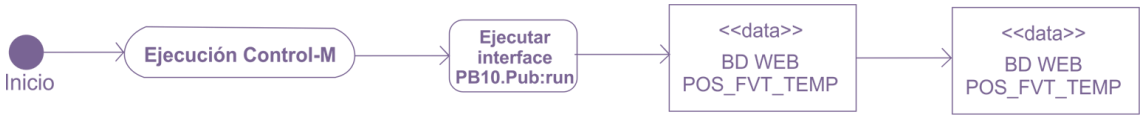


Figura 14 Diagrama de flujo de la interfaz PB10

Para un mejor entendimiento se agrega la descripción del funcionamiento del anterior diagrama.

La interface PB10 leera cada folio fiscal y buscará su correspondiente registro en la base de datos de WEB, una vez ubicado actualizará el registro del folio fiscal en WEB con su correspondiente folio de venta tal como fue emitido en la tienda.

En la figura 15 se muestra el proceso que seguirá la interfaz PB11 para realizar parte del procedimiento del desarrollo.

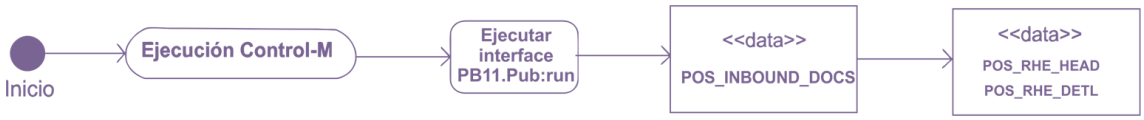


Figura 15 Diagrama de flujo de la interfaz PB11

Para un mejor entendimiento se agrega la descripción del funcionamiento del anterior diagrama.

Control-M ejecuta la interfaz PB11, por medio del job: PB11.Pub: run, una vez que se inicia la ejecución de la interfaz se obtiene la referencia de los registros de la POS_INBOUND_DOCS obtiene el encabezado de la RHE y el detalle de la RHE, la interfaz inserta la información en POS_RHE_HEAD y su detalle en POS_RHE_DETL y finaliza el flujo de la interfaz.

3.4. Generación de aplicación

A partir del análisis y el procedimiento realizado anteriormente se generaron las interfaces en el IDE de software AG, a continuación se muestran unas pantallas del proceso de creación de las interfaces.

Como se mencionaba anteriormente, el desarrollo se dividió en cinco interfaces, *véase la Figura 16 Paquetes creados por cada interfaz.*

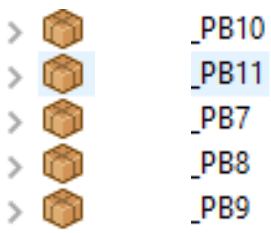


Figura 16 Paquetes creados por cada interfaz

A su vez, dentro del paquete se agrupan un conjunto de carpetas que a su vez contienen servicios de flujo marcados con una flecha, adaptadores marcados con una figura color azul, documentos marcados por una hoja y un disparador marcado marcado con una flecha azul para abajo, *véase la figura 17,18, 19 y 20* que contienen elementos parecidos.

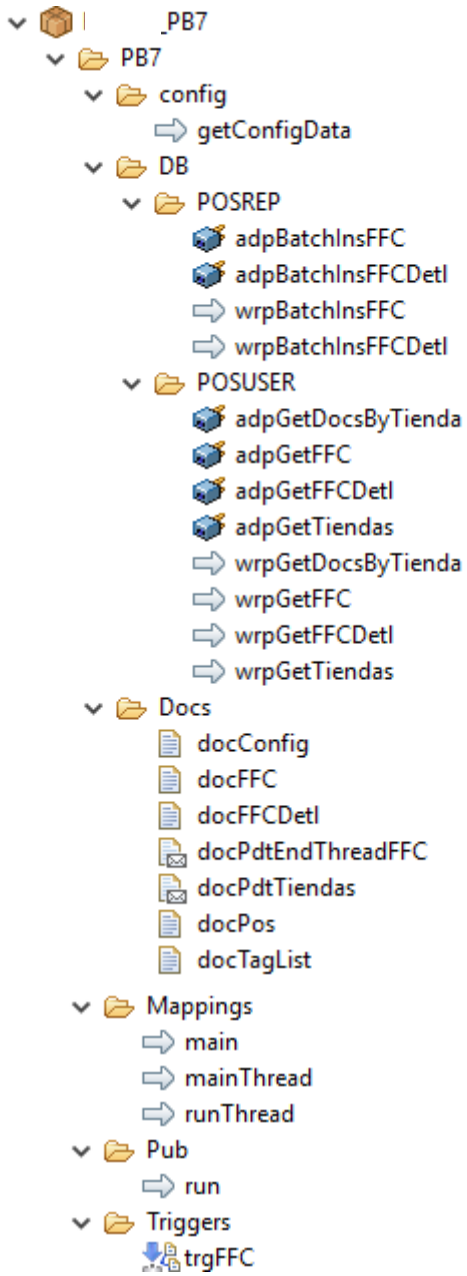


Figura 17 Estructura de carpetas de la interfaz PB7

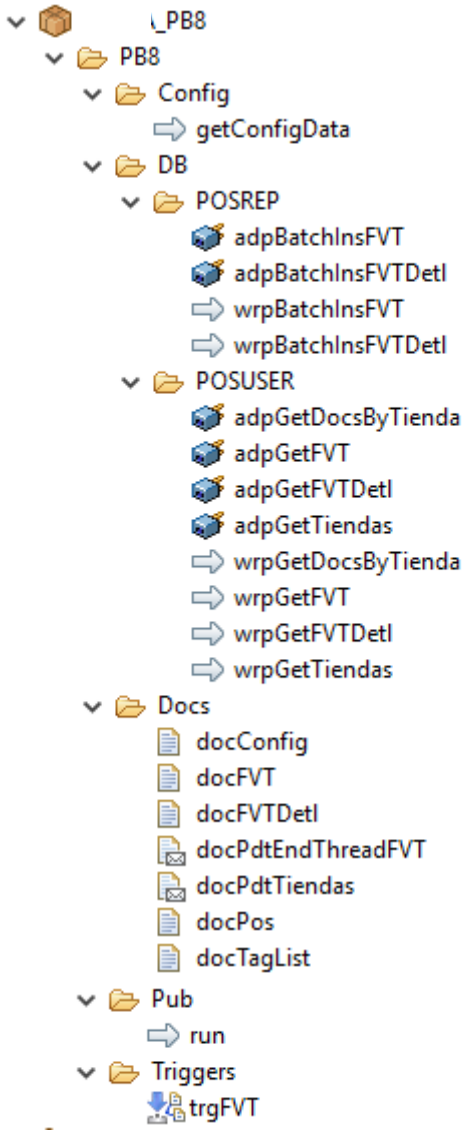


Figura 18 Estructura de carpetas de interfaz PB8

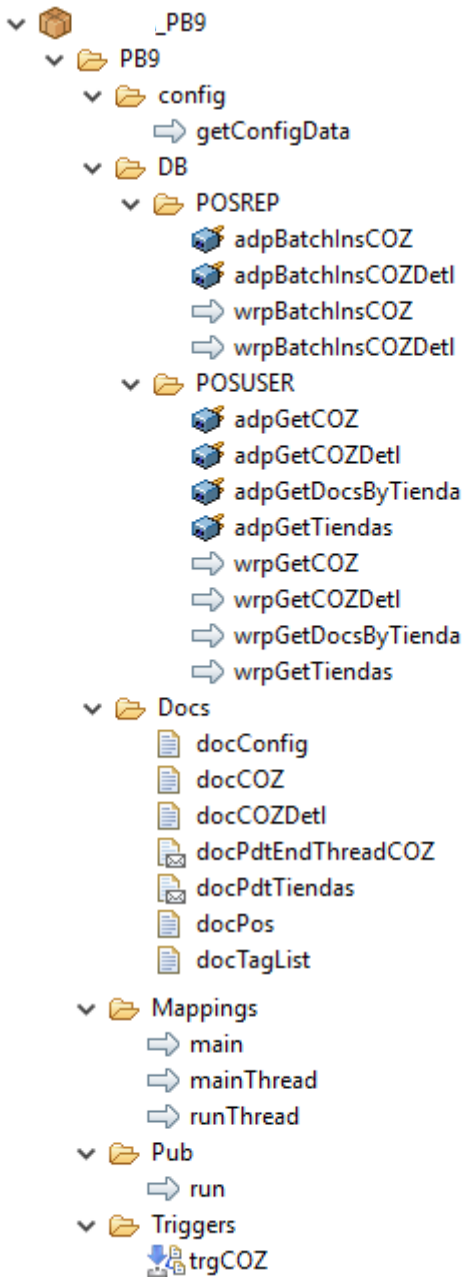


Figura 19 Estructura de carpetas de interfaz PB9

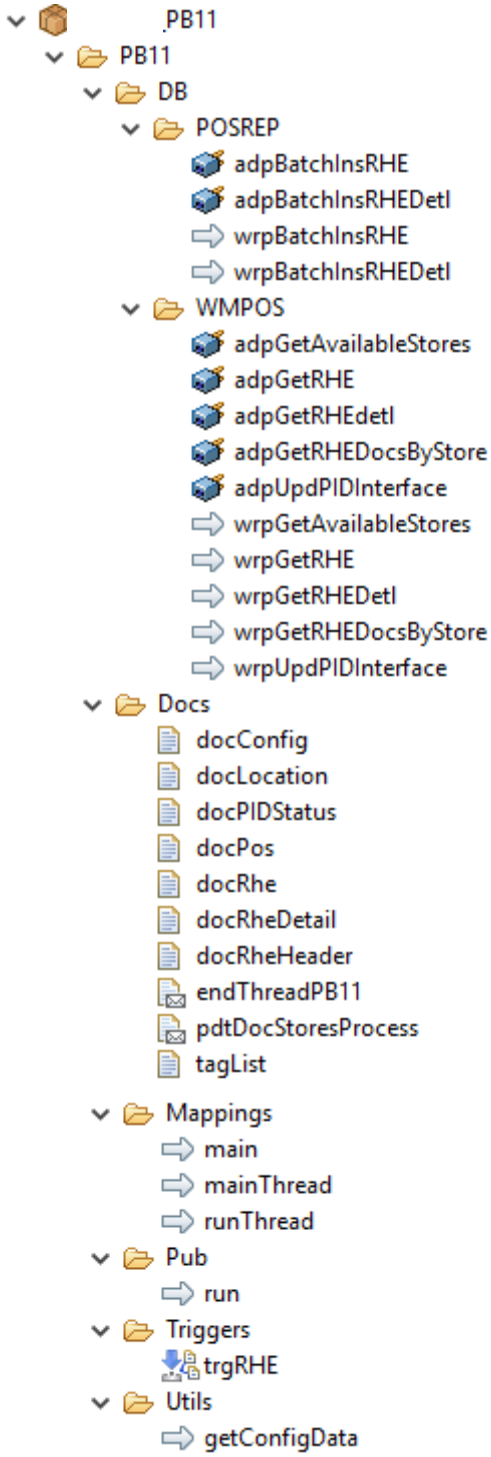


Figura 20 Estructura de carpetas de interfaz PB11

Dentro de los adaptadores se encuentra la estructura de como se mandará a traer la información, es decir la consulta SQL y recibirá una salida de información que se nombra en cada uno, *véase la Figura 21 Estructura de un adaptador* que muestra uno de todos los adaptadores que recibe información, estos adaptadores hacen uso de las reglas de negocio anteriormente definidas.

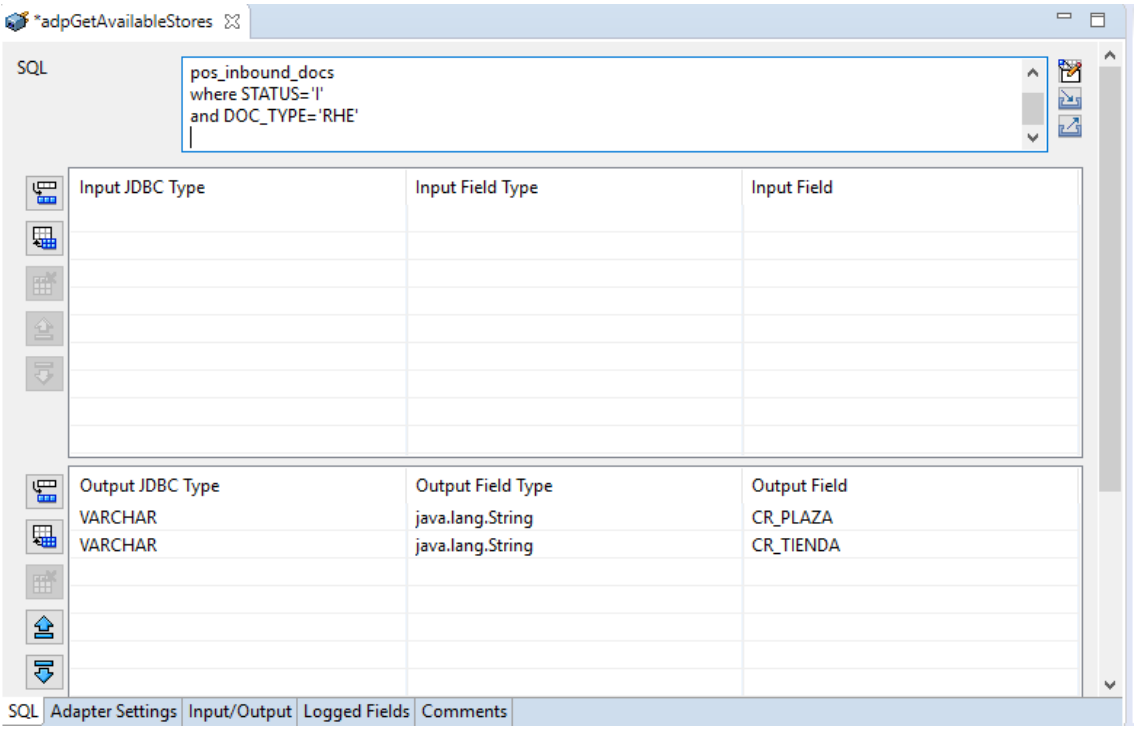


Figura 21 Estructura de un adaptador

Dentro de los servicios de flujo se realiza la programación de los bloques, haciendo uso de los adaptadores que son los que interactúan para recibir la información, también haciendo uso de otros servicios que la herramienta trae definidos, *véase la Figura 22 Estructura de un servicio de flujo.*

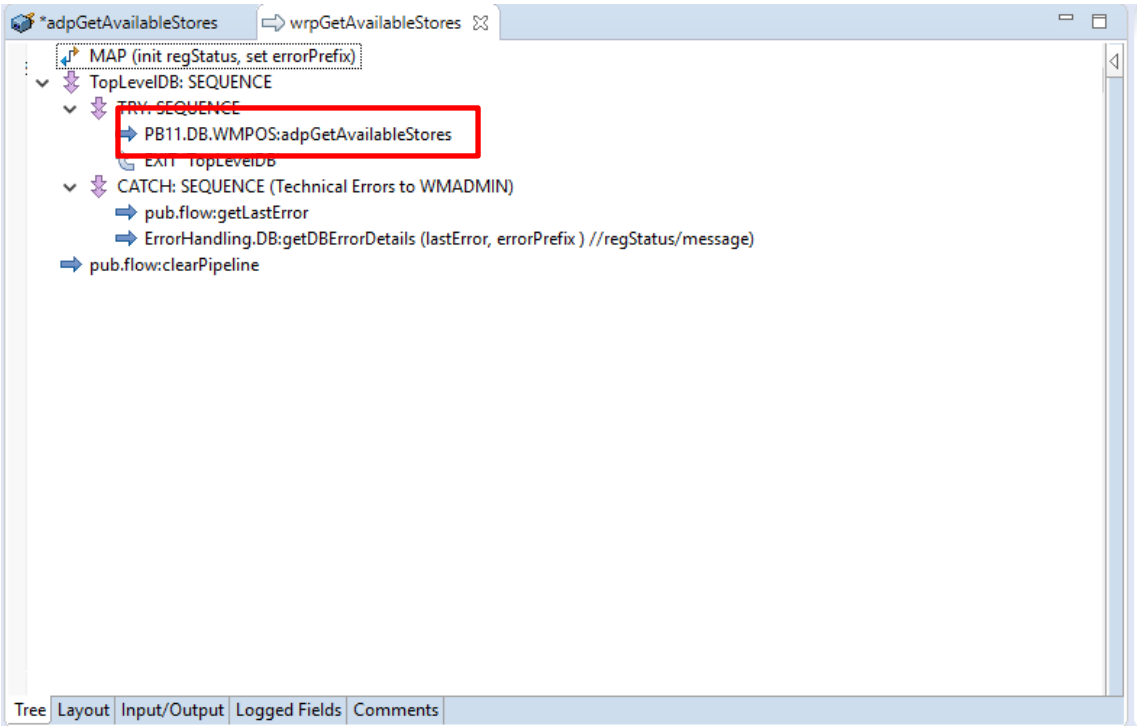


Figura 22 Estructura de un servicio de flujo

A su vez dentro de un bloque de un servicio de flujo se realiza el mapeo correspondiente es decir qué datos recibirá, cuál será su salida y hacia donde se mandará la salida, cabe resaltar que en esta parte es donde se mandan los errores a una salida en específica. *Véase la Figura 23 Mapeo en un bloque de código.*

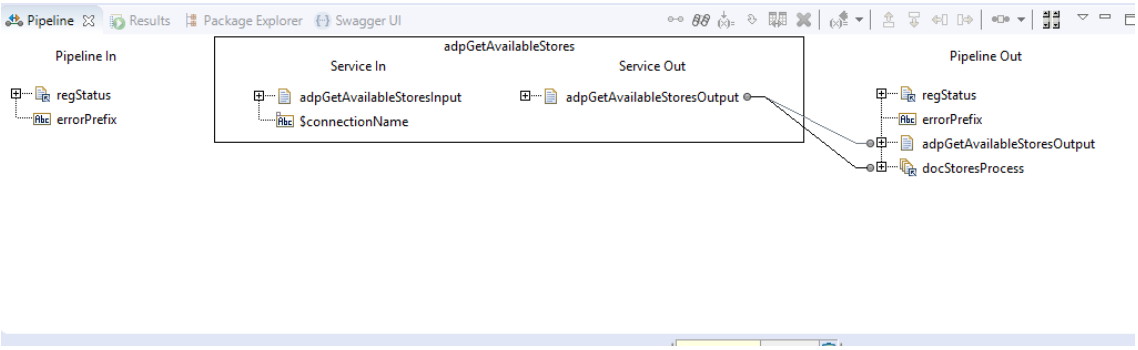


Figura 23 Mapeo en un bloque de código

3.5. Pruebas y entrega

3.5.1. Errores, advertencias y avisos

Al realizar las pruebas pertinentes se tiene que tomar en cuenta las salidas de errores que pueden existir, a continuación en la tabla 11 se muestra un listado con los errores y las acciones que se deberán tomar si existe una falla.

ID	Descripción	Acción	Rol Receptor
1	Error de conexión en base de datos	Enviar un correo, registrar acción en WMLOG	WMADMIN
2	Error al actualizar estatus en PID	Enviar un correo, registrar error en WMLOG	WMADMIN
3	Error al insertar en BD	Enviar un correo, registrar error en WMLOG hace rollback a la transacción	WMADMIN

Tabla 11 Listado de errores

3.5.2. Pruebas unitarias

Para cada interfaz se realizaron una prueba unitaria por interfaz, primeramente se realizó una prueba a la interfaz PB7 y se definieron los parámetros que se tomarán, *véase la Tabla 12 Prueba unitaria PB7*.

Artefacto o módulo	**Interfaz PB7**
Tipo de prueba	Funcional
Descripción del objeto	Verificar proceso normal de ejecución de la interfaz PB7 para la plaza 10BGA y tienda 50UNK
Prerrequisitos	1. Validar que exista información en la tabla POS_INBOUND_DOCS de POSUSER para la plaza 10BGA y tienda 50UNK con tipos de documento FFC y STATUS I.
Querys	Validar existencia de información en POS_INBOUND_DOCS:

	SELECT substr(pv_doc_name,4,5) pv_plaza, substr (pv_doc_name,9,5) pv_tienda, status, doc_type FROM pos_inbound_docs WHERE STATUS = 'I' AND substr(pv_doc_name,4,5) = '10BGA' AND DOC_TYPE = 'FFC' AND substr(pv_doc_name,9,5)= '50UNK'; Verificar el estatus en WMLOG: SELECT RUN_ID, interface, start_dt, end_dt, status FROM wm_log_run where interface = 'PB7' and status = 'S' and TRUNC(end_dt) = TO_DATE('20/09/2016','DD/MM/YYYY') order by run_id desc; Verificar la información en POS_FFC y POS_FFC_DETL: SELECT distinct pos_ffc.pid_id, pv_cr_plaza, pv_cr_tienda FROM pos_ffc_detl,pos_ffc WHERE pos_ffc.pid_id = pos_ffc_detl.pid_id AND pv_cr_plaza = '10BGA' AND TRUNC (wm_creation_date) = TO_DATE('20/09/2016','DD/MM/YYYY') AND pv_cr_tienda = '50UNK'; Validar que el estatus sea 'E' en la tabla POS_INBOUND_DOCS: SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA, pos_inbound_docs.id, status FROM pos_inbound_docs WHERE status = 'E' AND substr(pv_doc_name,4,5)= '10BGA' AND DOC_TYPE = 'FFC' AND target_id = [wm_log.run_id] AND substr(pv_doc_name,9,5)= '50UNK';
Pasos a seguir	1. Ejecutar el servicio PB7.Pub:run. 2. Verificar que el estatus sea igual a 'S' en la tabla WM_LOG_RUN de la BD del WMLOG. 3. Verificar que la informacion sea insertada y la tienda sea "50UNK" en las tablas POS_FFC y POS_FFC_DETL en POSREP 4. Validar que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' y la tienda sea = "50UNK" en la Base de Datos del POSUSER.
Resultados esperados	La información del POS es insertada en las tablas de POS_FFC, y el estatus en la tabla POS_INBOUND_DOCS del POSUSER es actualizado a 'E'
Comentarios	La información se procesó correctamente y la informaión viajo a su destino de forma correcta.
Fecha	20/11/2018

Tabla 12 Prueba unitaria PB7

A continuación se muestran los pasos que se siguieron para realizar la prueba.

Prerrequisito. Se verificó que se cumpliera el requisito para que la interfaz se ejecutará correctamente, *véase la Figura 24 Prerrequisito PB7.*

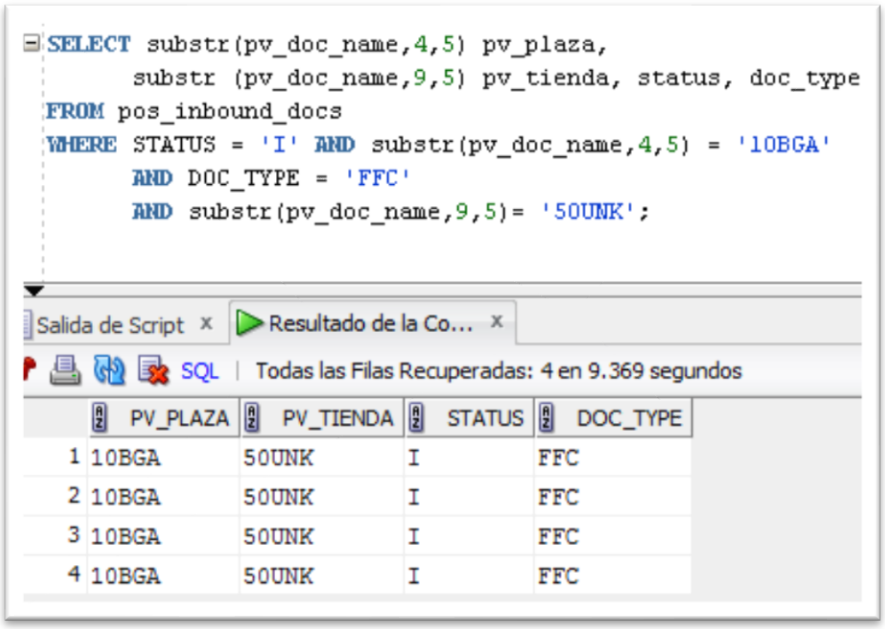


Figura 24 Prerrequisito PB7

Paso 1. Se ejecutó la interfaz en el IDE, *véase la Figura 25 Ejecución de interfaz PB7.*

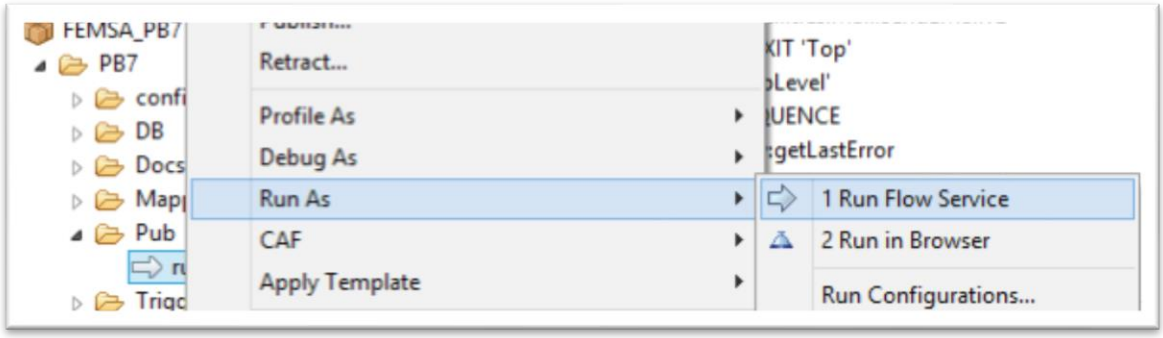


Figura 25 Ejecución de interfaz PB7

Paso 2. Se verifica que el estatus sea igual a S en la tabla WM_LOG_RUN de la interfaz que se esta ejecutando, *véase la Figura 26 Verificar estatus.*

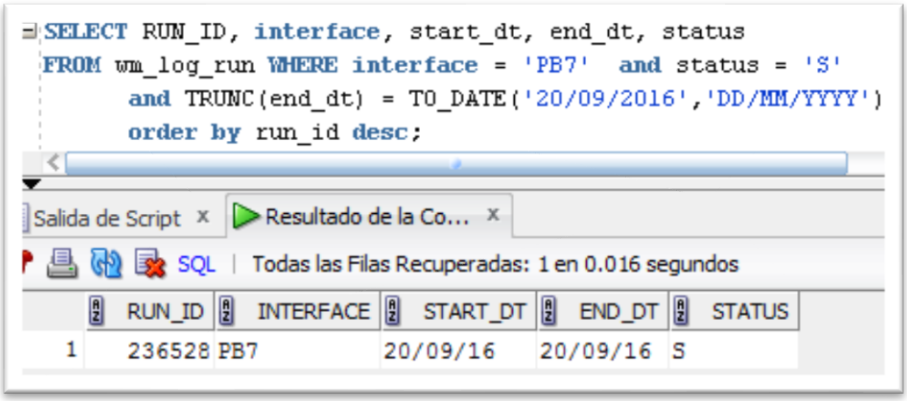


Figura 26 Verificar estatus

Paso 3. Se verifica que se haya insertado la información en la tabla POS_FFC_DETL y POS_FFC con la plaza 10BGA y la tienda 50UNK, *véase la Figura 27 Verificar información.*

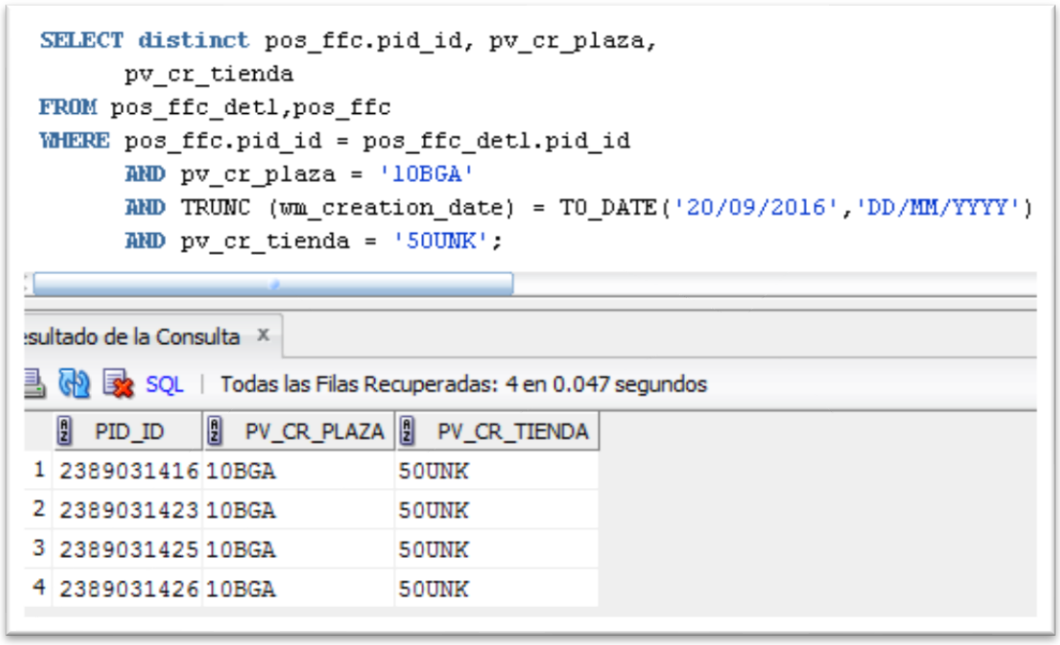


Figura 27 Verificar información

Paso 4. Se valida que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' y la tienda sea = "50UNK" en la Base de Datos del POSUSER, *véase la Figura 28 Verificar información insertada*.

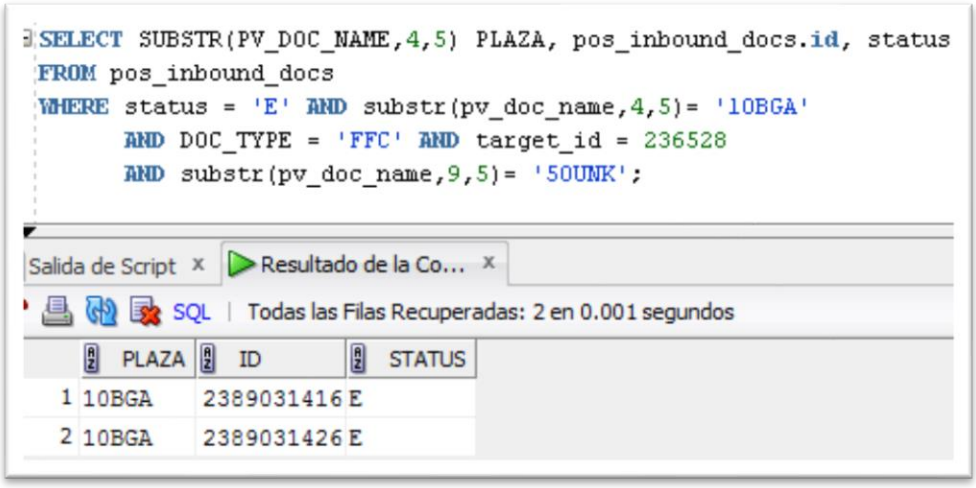


Figura 28 Verificar información insertada

La siguiente prueba se realizó a la interfaz PB8 de la cual se definieron los parámetros que se tomarán, *véase la Tabla 13 Prueba unitaria PB8*.

Artefacto o módulo	**Interfaz PB8**
Tipo de prueba	Funcional
Descripción del objeto	Verificar proceso normal de ejecución de la interfaz PB8 para la plaza 10BGA y tienda 50EKU
Prerrequisitos	1. Validar que exista información en la tabla POS_INBOUND_DOCS de POSUSER para la plaza 10BGA y tienda 50EKU con tipos de documento FVT y STATUS I.
Querys	Verificar si existe información en POS_INBOUND_DOCS: SELECT substr(pv_doc_name,4,5) pv_plaza,substr(pv_doc_name,9,5) pv_tienda, status, doc_type FROM pos_inbound_docs WHERE STATUS = 'I' AND substr(pv_doc_name,4,5) = '10BGA' AND DOC_TYPE = 'FVT' AND substr(pv_doc_name,9,5)= '50EKU'; Verificar el estatus en WMLOG: SELECT interface, start_dt, end_dt, status FROM wm_log_run WHERE interface = 'PB8' and status = 'S' AND TRUNC(end_dt) = TO_DATE('20/09/2016','DD/MM/YYYY') order by run_id desc; Verificar la información en POS_FVT y POS_FVT_DETL: SELECT * FROM pos_fvt_detl, pos_fvt WHERE pos_fvt.pid_id = pos_fvt_detl.pid_id AND pv_cr_plaza = '10BGA' AND TRUNC (wm_creation_date) = TO_DATE('20/09/2016','DD/MM/YYYY') AND pv_cr_tienda = '50EKU'; Validar que el estatus sea 'E' en la tabla POS_INBOUND_DOCS:

	SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA, pos_inbound_docs.id, status FROM pos_inbound_docs WHERE status = 'E' AND substr(pv_doc_name,4,5) = '10BGA' AND DOC_TYPE = 'FVT' AND substr(pv_doc_name,9,5) = '50EKU';
Pasos a seguir	1. Ejecutar el servicio PB8.Pub:run. 2. Verificar que el estatus sea igual a 'S' para la interface PB8 en la tabla WM_LOG_RUN de la BD del WMLOG. 3. Verificar que la informacion de la plaza 10BGA y la tienda 50EKU sea insertada en las tablas POS_FVT y POS_FVT_DETL en POSREP 4. Validar que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BGA y la tienda 50EKU con tipos de documento FVT Base de Datos del POSUSER.
Resultados esperados	La información del POS es insertada en las tablas de POS_FVT, y el estatus en la tabla POS_INBOUND_DOCS del POSUSER es actualizado a 'E'
Comentarios	La información se procesó correctamente y la informaión viajo a su destino de forma correcta.
Fecha	21/11/2018

Tabla 13 Prueba unitaria PB8

A continuación se muestran los pasos que se siguieron para realizar la prueba.

Prerrequisitos. Se valida que exista información en las tablas, *véase la Figura 29 Validación de información para interfaz PB8.*

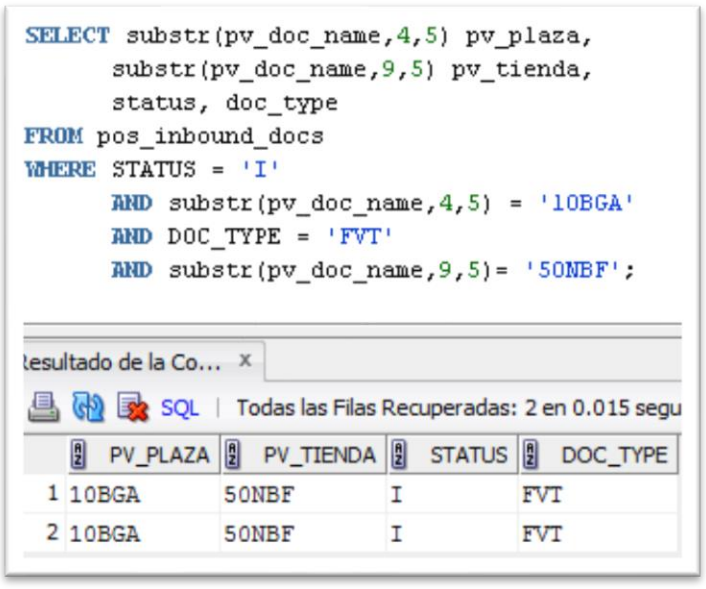


Figura 29 Validación de información para interfaz PB8

Paso 1. Se ejecutó la interfaz en el IDE, *véase la Figura 30 Ejecución de interfaz PB8.*

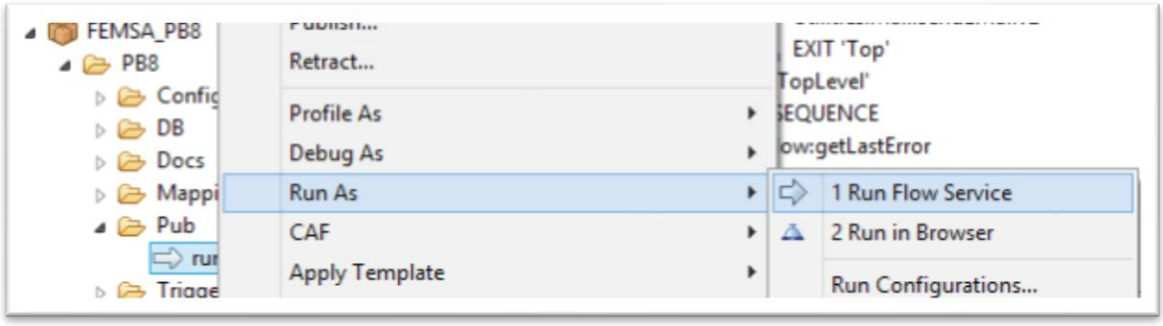
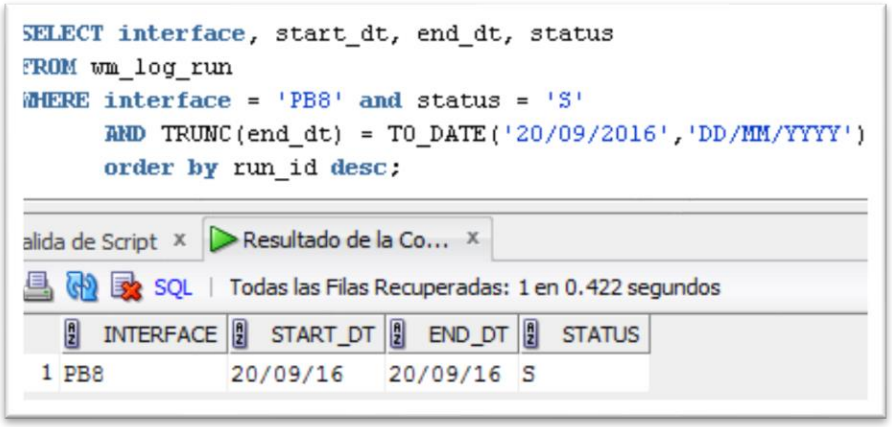


Figura 30 Ejecución de interfaz PB8

Paso 2. Se verificó que el estatus sea igual a 'S' en la tabla WM_LOG_RUN de la base de datos del WMLOG, *véase la Figura 31 Verificar estatus para interfaz PB8*.



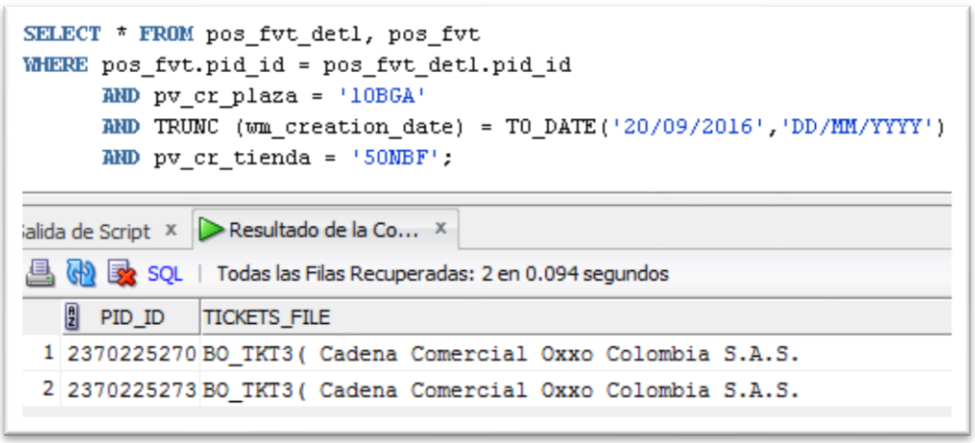
```
SELECT interface, start_dt, end_dt, status
FROM wm_log_run
WHERE interface = 'PB8' and status = 'S'
      AND TRUNC(end_dt) = TO_DATE('20/09/2016','DD/MM/YYYY')
      order by run_id desc;
```

alida de Script | Resultado de la Co... | SQL | Todas las Filas Recuperadas: 1 en 0.422 segundos

	INTERFACE	START_DT	END_DT	STATUS
1	PB8	20/09/16	20/09/16	S

Figura 31 Verificar estatus para interfaz PB8

Paso 3. Se verificó que la informacion de la plaza 10BGA y la tienda 50EKU sea insertada en las tablas POS_FVT y POS_FVT_DETL en POSREP, *véase la Figura 32 Verificar información de interfaz PB8.*



```
SELECT * FROM pos_fvt_detl, pos_fvt
WHERE pos_fvt.pid_id = pos_fvt_detl.pid_id
      AND pv_cr_plaza = '10BGA'
      AND TRUNC (wm_creation_date) = TO_DATE('20/09/2016','DD/MM/YYYY')
      AND pv_cr_tienda = '50NBF';
```

alida de Script | Resultado de la Co... | SQL | Todas las Filas Recuperadas: 2 en 0.094 segundos

	PID_ID	TICKETS_FILE
1	2370225270	BO_TKT3(Cadena Comercial Oxxo Colombia S.A.S.
2	2370225273	BO_TKT3(Cadena Comercial Oxxo Colombia S.A.S.

Figura 32 Verificar información de interfaz PB8

Paso 4. Se validó que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BGA y la tienda 50EKU con tipos de documento FVT Base de

Datos del POSUSER, *véase la Figura 33 Validación de estatus en la tabla POS_INBOUND_DOCS*.

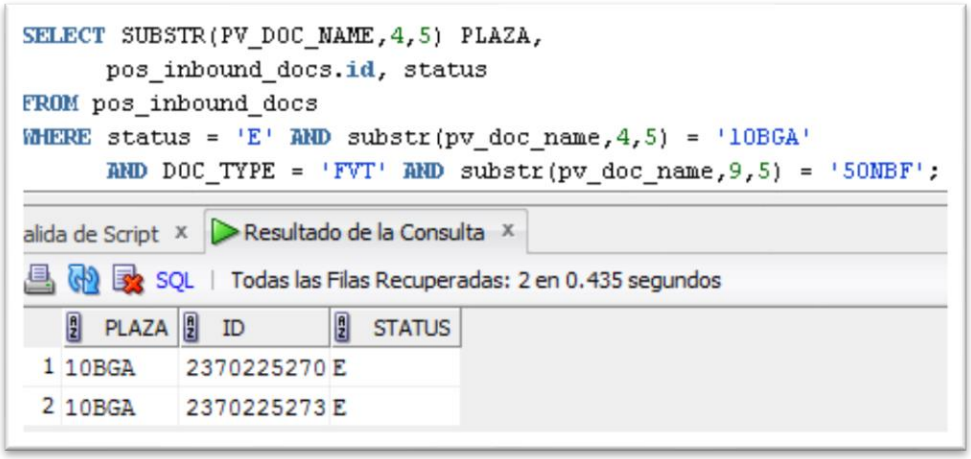


Figura 33 Validación de estatus en la tabla POS_INBOUND_DOCS

La siguiente prueba se realizó a la interfaz PB9 de la cual se definieron los parámetros que se tomarán, *véase la Tabla 14 Prueba unitaria PB9*.

Artefacto o módulo	**Interfaz PB9**
Tipo de prueba	Funcional
Descripción del objeto	Verificar proceso normal de ejecución de la interfaz PB9 para la plaza 10BGA y tienda 50UYL
Prerrequisitos	1. Validar que exista información en la tabla POS_INBOUND_DOCS de POSUSER para la plaza 10BGA y tienda 50UYL con tipos de documento COZ y STATUS I.
Querys	Verificar si existe información en POS_INBOUND_DOCS: SELECT DISTINCT substr(pv_doc_name,4,5) pv_plaza,substr (pv_doc_name,9,5) pv_tienda, status, doc_type FROM pos_inbound_docs WHERE STATUS = 'I' AND substr(pv_doc_name,4,5) = '10BGA' AND DOC_TYPE = 'COZ' AND substr(pv_doc_name,9,5)= '50UYL'; Verificar el estatus en WMLOG: SELECT RUN_ID, interface, start_dt, end_dt, status FROM wm_log_run where interface = 'PB9' and status = 'S' and TRUNC(end_dt) = TO_DATE('21/09/2016','DD/MM/YYYY') order by run_id desc; Verificar la información en POS_FVT y POS_FVT_DETL: SELECT * FROM pos_coz_detl, pos_coz WHERE pos_coz.pid_id = pos_coz_detl.pid_id AND pv_cr_plaza = '10BGA' and TRUNC (wm_creation_date) = TO_DATE('21/09/2016','DD/MM/YYYY') AND pv_cr_tienda = '50UYL';

	Validar que el estatus sea 'E' en la tabla POS_INBOUND_DOCS: SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA, pos_inbound_docs.id, status FROM pos_inbound_docs WHERE status = 'E' AND substr(pv_doc_name,4,5)= '10BGA' AND DOC_TYPE = 'COZ AND target_id = [wm_log.run_id] AND SUBSTR(PV_DOC_NAME,9,5) = '50UYL';
Pasos a seguir	1. Ejecutar el servicio PB9.Pub:run. 2. Verificar que el estatus sea igual a 'S' para la interface PB9 en la tabla WM_LOG_RUN de la BD del WMLOG. 3. Verificar que la informacion de la plaza 10BGA y la tienda 50UYL sea insertada en las tablas POS_COZ y POS_COZ_DETL en POSREP 4. Validar que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BGA y la tienda 50UYL con tipos de documento COZ Base de Datos del POSUSER.
Resultados esperados	La información del POS es insertada en las tablas de POS_COZ, y el estatus en la tabla POS_INBOUND_DOCS del POSUSER es actualizado a 'E
Comentarios	La información se procesó correctamente y la informaión viajo a su destino de forma correcta.
Fecha	22/11/2018

Tabla 14 Prueba unitaria PB9

A continuación se muestran los pasos que se siguieron para realizar la prueba.

Prerrequisitos. Se valida que exista información en las tablas, *véase la Figura 34 Validación de información para interfaz PB9.*

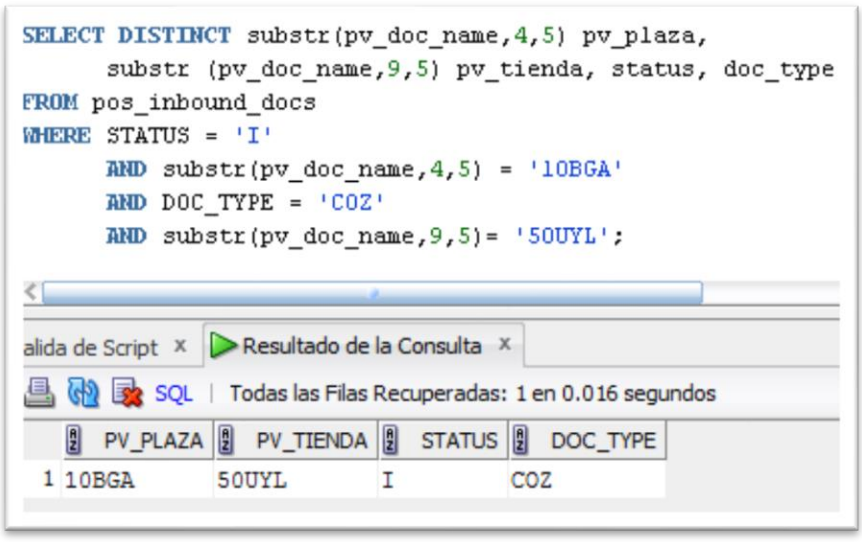


Figura 34 Validación de información para interfaz PB9

Paso 1. Se ejecutó la interfaz en el IDE, *véase la Figura 35 Ejecución de interfaz PB9.*

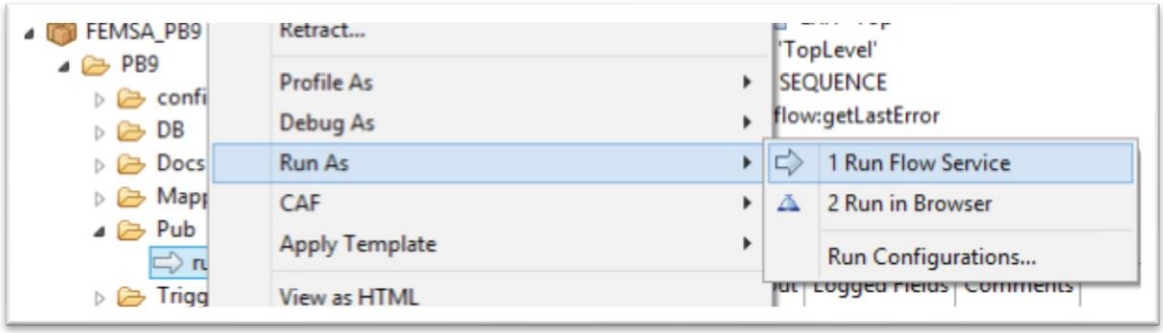
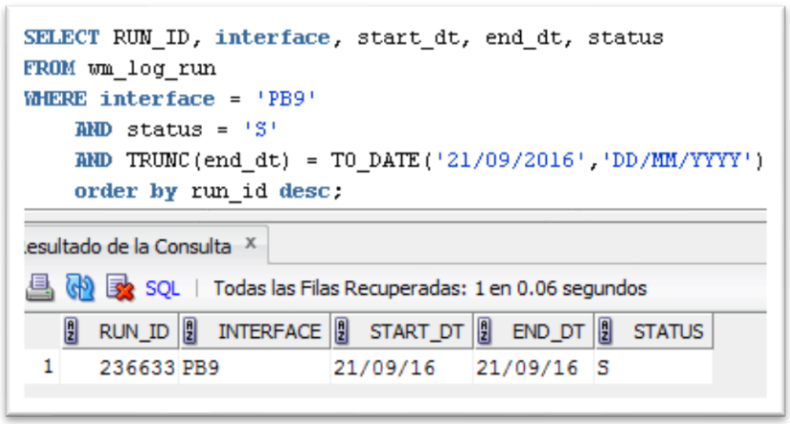


Figura 35 Ejecución de interfaz PB9

Paso 2. Se verificó que el estatus sea igual a 'S' en la tabla WM_LOG_RUN de la base de datos del WMLOG, *véase la Figura 36 Verificar estatus para interfaz PB9*.



```
SELECT RUN_ID, interface, start_dt, end_dt, status
FROM wm_log_run
WHERE interface = 'PB9'
    AND status = 'S'
    AND TRUNC(end_dt) = TO_DATE('21/09/2016','DD/MM/YYYY')
    order by run_id desc;
```

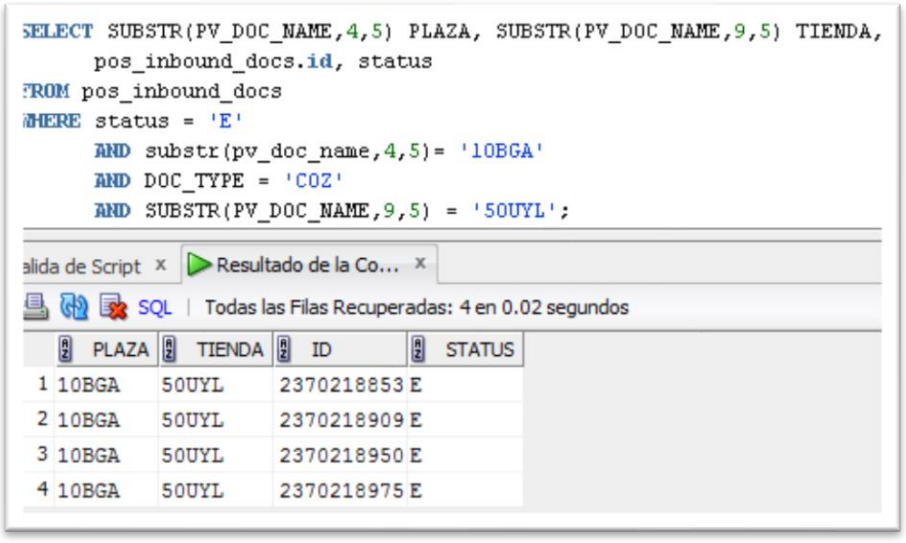
esultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0.06 segundos

	RUN_ID	INTERFACE	START_DT	END_DT	STATUS
1	236633	PB9	21/09/16	21/09/16	S

Figura 36 Verificar estatus para interfaz PB9

Paso 3. Se verificó que la informacion de la plaza 10BGA y la tienda 50EKU sea insertada en las tablas POS_FVT y POS_FVT_DETL en POSREP, *véase la Figura 37 Verificar información de interfaz PB9.*



```
SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA, SUBSTR(PV_DOC_NAME,9,5) TIENDA,
      pos_inbound_docs.id, status
FROM pos_inbound_docs
WHERE status = 'E'
      AND substr(pv_doc_name,4,5)= '10BGA'
      AND DOC_TYPE = 'COZ'
      AND SUBSTR(PV_DOC_NAME,9,5) = '50UYL';
```

alida de Script | Resultado de la Co...

SQL | Todas las Filas Recuperadas: 4 en 0.02 segundos

	PLAZA	TIENDA	ID	STATUS
1	10BGA	50UYL	2370218853	E
2	10BGA	50UYL	2370218909	E
3	10BGA	50UYL	2370218950	E
4	10BGA	50UYL	2370218975	E

Figura 37 Verificar información de interfaz PB9

Paso 4. Se validó que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BGA y la tienda 50UYL con tipos de documento COZ Base de Datos del POSUSER, *véase la Figura 38 Validación de estatus con tipo de documento COZ*.

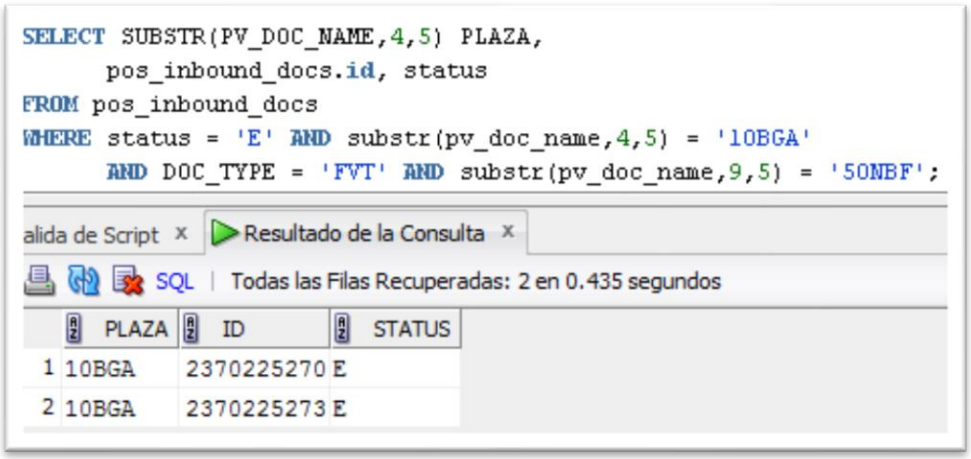


Figura 38 Validación de estatus con tipo de documento COZ

La siguiente prueba se realizó a la interfaz PB10 de la cual se definieron los parámetros que se tomarán, *véase la Tabla 15 Prueba unitaria PB10*.

Artefacto o módulo	**Interfaz PB10**
Tipo de prueba	Funcional
Descripción del objeto	Verificar la correcta ejecución de la interfaz PB10 (runLoad) para la plaza 10BGA y tienda 50RBT
Prerrequisitos	1. Validar que exista información en las tablas POS_FVT y POS_FVT_DETL de POSREP para la plaza 10BGA y tienda 50RBT con status I. 2. Validar que no existan los documentos que se van a procesar en la tabla POS_FVT_TEMP en POSREP para la plaza 10BGA y tienda 50RBT con status I.
Querys	SELECT COUNT(A.PID_ID) NUM_REG FROM POS_FVT_DETL B, POS_FVT A WHERE A.PID_ID = B.PID_ID AND PV_CR_PLAZA = '10BGA' AND PV_CR_TIENDA = '50RBT' AND STATUS = 'I'; Verificar que NO existan documentos a procesar en POS_FVT_TEMP en POSREP: SELECT COUNT(PID_ID) FROM POS_FVT_TEMP WHERE PV_CR_PLAZA = '10BGA' AND PV_CR_TIENDA = '50RBT' AND pid_id = [PID_ID] AND pv_folio = [PV_FOLIO]; Verificar el estatus en wm_log_run en WMLOG:

	SELECT RUN_ID, interface, start_dt, end_dt, status FROM wm_log_run WHERE interface = 'PB10Load' and status = 'S' and TRUNC(end_dt) = TO_DATE([Fecha de Creacion],'DD/MM/YYYY') order by run_id desc; Verificar que existan registros en WM_LOG_THREAD en WMLOG: SELECT * FROM wm_log_thread WHERE ATT1 = '10BGA' AND ATT2 = '50RBT' AND TRUNC(END_DT) = TO_DATE([Fecha_Ejecución], 'DD/MM/YYYY') AND PARENT_ID = [wm_log_run.run_id] AND STATUS = 'S' Verificar la información en POS_FVT_TEMP en POSREP: SELECT PID_ID, pv_cr_plaza, pv_cr_tienda FROM pos_fvt_temp WHERE pv_cr_plaza = '10BGA' AND pv_cr_tienda = '50RBT' AND pid_id = [PID_ID] AND TRUNC(CREATION_DATE) =TO_DATE([Fecha de Creacion],'DD/MM/YYYY'); Validar que el estatus sea 'P' en la tabla POS_FVT EN POSREP: SELECT PID_ID, pv_cr_plaza, pv_cr_tienda, STATUS, run_id FROM pos_fvt WHERE pv_cr_plaza = '10BGA' AND pv_cr_tienda = '50RBT' AND STATUS = 'P' and pid_id = [PID_ID] and run_id = [wmlog.wm_log_thread.thread_id];
Pasos a seguir	1. Ejecutar el servicio PB10.Pub:runLoad. 2. Verificar que el estatus sea igual a 'S' para la interface PB10Load en la tabla WM_LOG_RUN de la BD del WMLOG. 3.Verificar que existan registro en la tabla WM_LOG_THREAD para la plaza 10BGA y la tienda 50DDV con status S. 4. Verificar que la informacion de la plaza 10BGA y la tienda 50DDV sea insertada en la tabla POS_FVT_TEMP en POSREP 5. Validar que el estatus en la tabla POS_FVT sea igual a 'P' en la plaza 10BGA y la tienda 50RBT en POSREP.
Resultados esperados	La información de las tablas POS_FVT Y POS_FVT_DETL se debe procesar correctamente.
Comentarios	La información se procesó correctamente y la informaión viajo a su destino de forma correcta.
Fecha	23/11/2018

Tabla 15 Prueba unitaria PB10

A continuación se muestran los pasos que se siguieron para realizar la prueba.

Prerrequisitos 1 y 2. Se valida que exista información en las tablas, *véase la Figura 39 y 40.*

```
SELECT DISTINCT substr(pv_doc_name,4,5) pv_plaza,
      substr (pv_doc_name,9,5) pv_tienda, status, doc_type
FROM pos_inbound_docs
WHERE STATUS = 'I'
      AND substr(pv_doc_name,4,5) = '10BGA'
      AND DOC_TYPE = 'COZ'
      AND substr(pv_doc_name,9,5)= '50UYL';
```

alida de Script × Resultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 1 en 0.016 segundos

	PV_PLAZA	PV_TIENDA	STATUS	DOC_TYPE
1	10BGA	50UYL	I	COZ

Figura 39 Validación de información para interfaz PB10

```
SELECT COUNT(A.PID_ID) NUM_REG
FROM POS_FVT_DETL B, POS_FVT A
WHERE A.PID_ID = B.PID_ID
    AND PV_CR_PLAZA = '10BGA'
    AND PV_CR_TIENDA = '50DDV'
    AND STATUS = 'I';
```

alida de Script × Resultado de la Consu

SQL | Todas las Filas Recupera

	NUM_REG
1	1

Figura 40 Validación de información

Paso 1. Se ejecutó la interfaz en el IDE, *véase la Figura 41 Ejecución de interfaz PB10.*

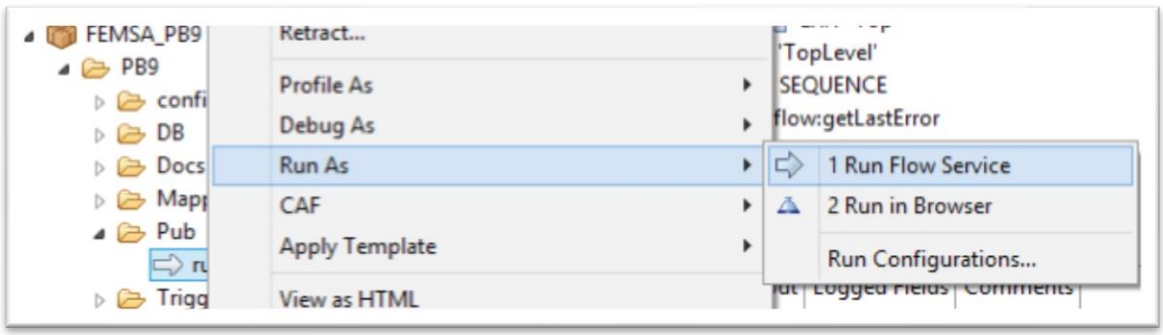


Figura 41 Ejecución de interfaz PB10

Paso 2. Se verificó que el estatus sea igual a 'S' en la tabla WM_LOG_RUN de la base de datos del WMLOG, *véase la Figura 42 Verificar estatus para interfaz PB10*.

```
SELECT RUN_ID, INTERFACE, START_DT, END_DT, STATUS
FROM WM_LOG_RUN WHERE INTERFACE = 'PB10Load' AND STATUS = 'S'
    AND TRUNC(end_dt) = TO_DATE('30/09/2016','DD/MM/YYYY')
    order by run_id desc;
```

esultado de la Co... ×

SQL | Todas las Filas Recuperadas: 2 en 0.016 segundos

	RUN_ID	INTERFACE	START_DT	END_DT	STATUS
1	237304	PB10Load	30/09/16	30/09/16	S
2	237269	PB10Load	30/09/16	30/09/16	S

Figura 42 Verificar estatus para interfaz PB10

Paso 3. Se verificó que existan registro en la tabla WM_LOG_THREAD para la plaza 10BGA y la tienda 50DDV con status S, *véase la Figura 43 Verificar información de interfaz PB10.*

```
SELECT * FROM WM_LOG_THREAD WHERE ATT1 = '10BGA' AND ATT2 = '50DDV'
    AND TRUNC(END_DT) = TO_DATE('30/09/2016','DD/MM/YYYY')
    AND PARENT_ID = 237304 AND STATUS = 'S';
```

esultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 1 en 0.016 segundos

	THREAD_ID	PARENT_ID	NAME	START_DT	END_DT	STATUS	ATT1
1	237305	237304	PB10Load_10BGA_50DDV	30/09/16	30/09/16	S	10BGA

Figura 43 Verificar información de interfaz PB10

Paso 4. Se verificó que la informacion de la plaza 10BGA y la tienda 50DDV sea insertada en la tabla POS_FVT_TEMP en POSREP, *véase la Figura 44 Validación información en POS_FVT_TEMP.*

```
SELECT PID_ID, PV_CR_PLAZA, PV_CR_TIENDA
FROM POS_FVT_TEMP
WHERE PV_CR_PLAZA = '10BGA' AND PV_CR_TIENDA = '50DDV'
      AND PID_ID = 2370226043 AND PV_FOLIO = 231379
      AND TRUNC(CREATION_DATE) =TO_DATE('30/09/2016','DD/MM/YYYY');
```

esultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 1 en 0.031 segundos

	PID_ID	PV_CR_PLAZA	PV_CR_TIENDA
1	2370226043	10BGA	50DDV

Figura 44 Validación de información en POS_FVT_TEMP

Paso 5. Se validó que los registros sean actualizados correctamente en la tabla POS_FFC con Plaza 10BGA y Tienda 50RBT y el campo TICKET_FILE no es nulo, *véase la Figura 45 Validaciónde estatus en tabla POS_FFC.*

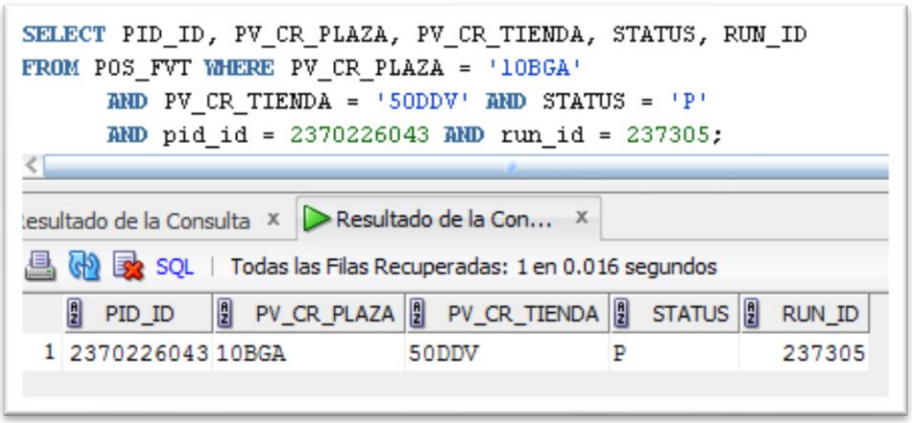


Figura 45 Validación de estatus en tabla POS_FFC

La siguiente prueba se realizó a la interfaz PB11 de la cual se definieron los parámetros que se tomarán, *véase la Tabla 16 Prueba unitaria PB11*.

Artefacto o módulo	**Interfaz PB11**
Tipo de prueba	Funcional
Descripción del objeto	Verificar proceso normal de ejecución de la interfaz PB11 para la plaza 10BCA y tienda 50BB1
Prerrequisitos	1. Validar que exista información en la tabla POS_INBOUND_DOCS de POSUSER para la plaza 10BCA y tienda 50BB1 con tipos de documento RHE y STATUS I.
Querys	Verificar si existe información en POS_INBOUND_DOCS: SELECT DISTINCT substr(pv_doc_name,4,5) PV_CR_PLAZA , substr(pv_doc_name,9,5) PV_CR_TIENDA from pos_inbound_docs where STATUS = 'I' and DOC_TYPE= 'RHE' and substr(pv_doc_name,4,5) = '10BCA' and substr(pv_doc_name,9,5) = '50OR0' Verificar el estatus en WMLOG: SELECT RUN_ID, interface, start_dt, end_dt, status FROM wm_log_run WHERE interface = 'PB11' AND status = 'S' AND TRUNC(end_dt) = TO_DATE([Fecha_de_Dificación], 'DD/MM/YYYY') order by run_id desc; Verificar que existan registros en WM_LOG_THREAD: SELECT * FROM wm_log_thread WHERE ATT1 = '10BCA' AND ATT2 = '50OR0' AND TRUNC(END_DT) = TO_DATE([Fecha_Ejecución], 'DD/MM/YYYY') AND PARENT_ID = [wm_log_run.run_id] AND STATUS = 'S'

	Verificar la información en POS_RHE y POS_RHE_DETL en POSREP: SELECT distinct pos_rhe.pid_id, substr(pv_doc_name,4,5), substr(pv_doc_name,9,5) FROM pos_rhe_detl, pos_rhe WHERE pos_rhe.pid_id = pos_rhe_detl.pid_id AND POS_RHE.PID_ID = [POS_INBOUND_DOCS.ID] AND substr(pv_doc_name,4,5) = '10BCA' AND substr(pv_doc_name,9,5) = '50OR0'; Validar que el estatus sea 'E' en la tabla POS_INBOUND_DOCS: SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA, Pos_inbound_docs.id, status FROM pos_inbound_docs WHERE status = 'E' AND substr(pv_doc_name,4,5)= '10BCA' AND DOC_TYPE = 'RHE' AND target_id = [wm_log_thread.thread_id] AND SUBSTR(PV_DOC_NAME,9,5) = '50OR0';
Pasos a seguir	1. Ejecutar el servicio PB11.Pub:run. 2. Verificar que el estatus sea igual a 'S' para la interface PB11 en la tabla WM_LOG_RUN de la BD del WMLOG. 3. Verificar que existan registro en la tabla WM_LOG_THREAD para la plaza 10BCA y la tienda 50OR0 con status S. 4. Verificar que la informacion de la plaza 10BCA y la tienda 50OR0 sea insertada en las tablas POS_RHE y POS_RHE_DETL en POSREP 5. Validar que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BCA y la tienda 50OR0 con tipos de documeto RHE Base de Datos del POSUSER.
Resultados esperados	La información del POS es insertada en las tablas de POS_RHE, y el estatus en la tabla POS_INBOUND_DOCS del POSUSER es actualizado a 'E'
Comentarios	La información se procesó correctamente y la informaión viajo a su destino de forma correcta.
Fecha	23/11/2018

Tabla 16 Prueba unitaria PB11

A continuación se muestran los pasos que se siguieron para realizar la prueba.

Prerrequisitos 1 . Se valida que exista información en las tablas, *véase la Figura 46 Validación de información para interfaz PB11.*

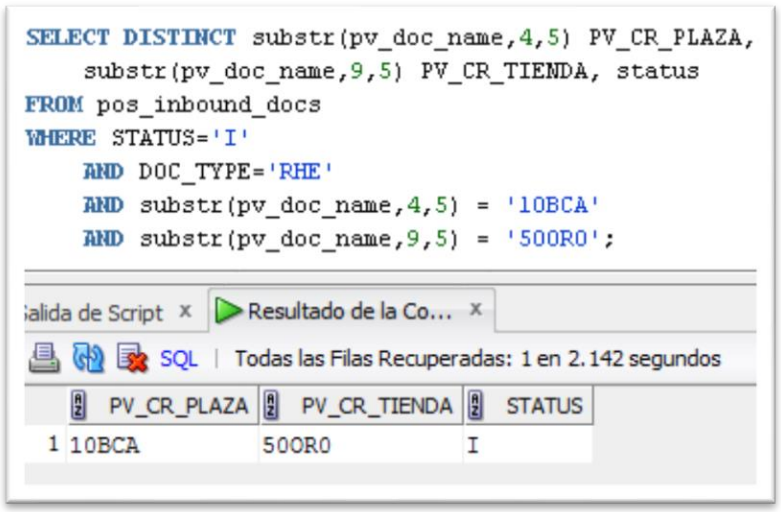


Figura 46 Validación de información para interfaz PB11

Paso 1. Se ejecutó la interfaz en el IDE, *véase la Figura 47 Ejecución de interfaz PB11.*

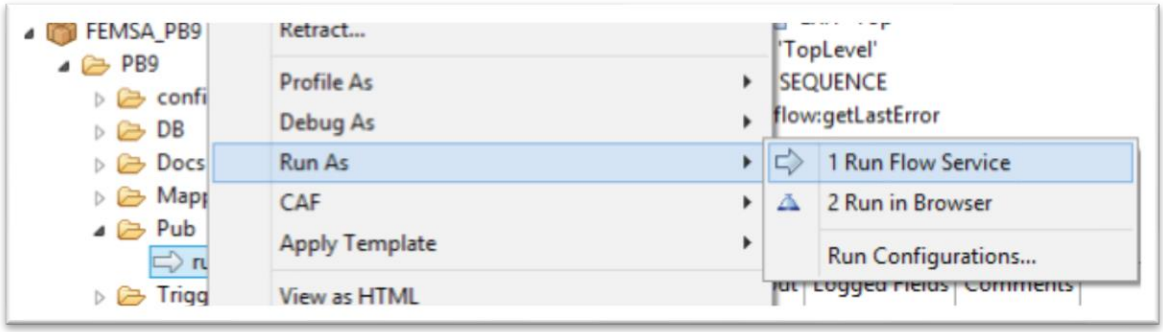


Figura 47 Ejecución de interfaz PB11

Paso 2. Se verificó que el estatus sea igual a 'S' en la tabla WM_LOG_RUN de la base de datos del WMLOG, *véase la Figura 48 Verificar estatus para interfaz PB11*.

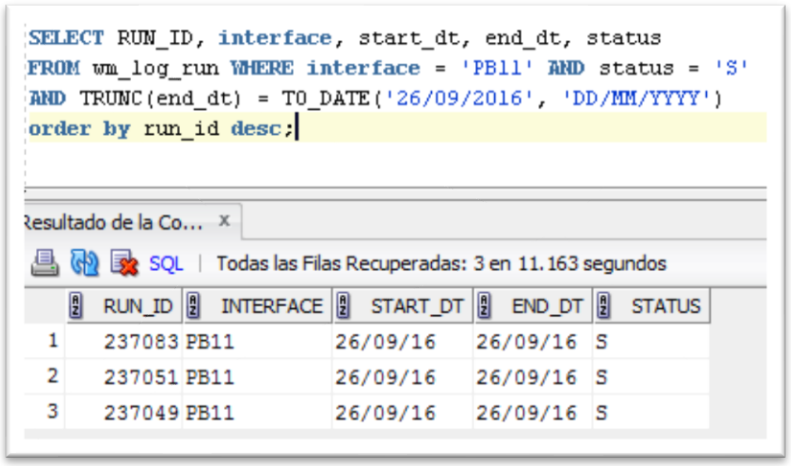


Figura 48 Verificar estatus para interfaz PB11

Paso 3. Se verificó que existan registro en la tabla WM_LOG_THREAD para la plaza 10BCA y la tienda 50OR0 con status S, *véase la Figura 49 Verificar información de interfaz PB11.*

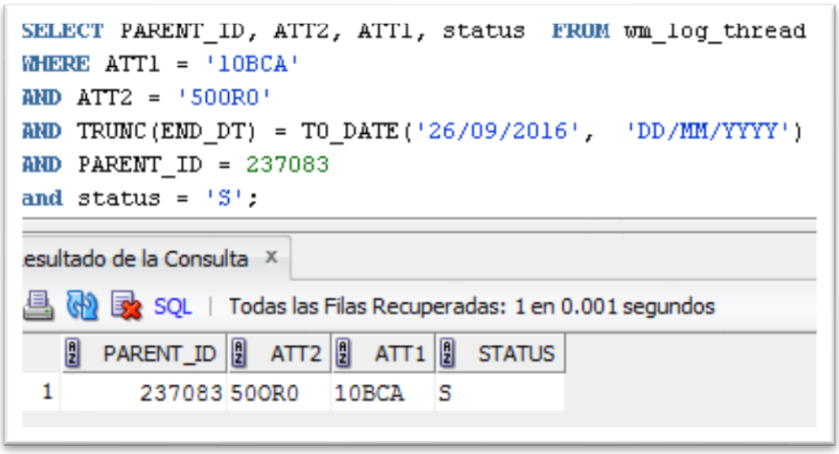


Figura 49 Verificar información de interfaz PB11

Paso 4. Se verificó que la informacion de la plaza 10BCA y la tienda 50OR0 sea insertada en las tablas POS_RHE y POS_RHE_DETL en POSREP, *véase la Figura 50 Validación de información en POS_RHE y POS_RHE_DETL*.

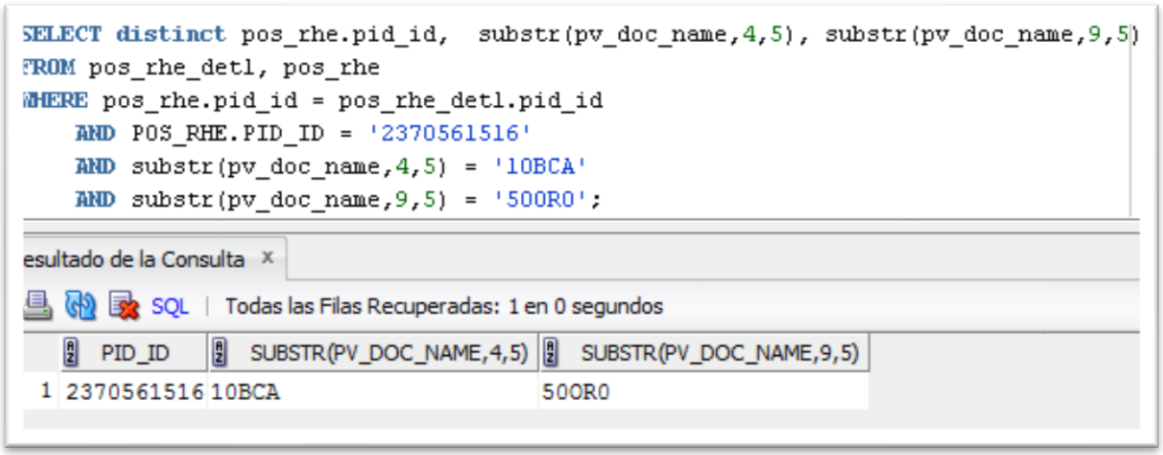


Figura 50 Validación de información en *POS_RHE y POS_RHE_DETL*

Paso 5. Se validó que el estatus en la tabla POS_INBOUND_DOCS sea igual a 'E' en la plaza 10BCA y la tienda 50OR0 con tipos de documeto RHE Base de Datos del POSUSER, *véase la Figura 51 Validación de estatus en tabla POS_INBOUND_DOCS*.

```
SELECT SUBSTR(PV_DOC_NAME,4,5) PLAZA,
       pos_inbound_docs.id, status
FROM pos_inbound_docs
WHERE status = 'E'
       AND substr(pv_doc_name,4,5)= '10BCA'
       AND DOC_TYPE = 'RHE'
       AND target_id = 237091
       AND SUBSTR(PV_DOC_NAME,9,5) = '500R0';
```

esultado de la Consulta x

SQL | Todas las Filas Recuperadas: 2 en 0.015 s

	PLAZA	ID	STATUS
1	10BCA	2369523197	E
2	10BCA	2370561516	E

Figura 51 Validación de estatus en tabla POS_INBOUND_DOCS

4. Conclusiones y recomendaciones

Como resultado de la realización de mi estadía profesional se obtuvieron cinco interfaces que trabajan en conjunto para la centralización y respaldo de folios fiscales que a futuro será implementado dentro de una cadena de tiendas, la realización de este proyecto me ayudó a conocer diferentes herramientas que son de gran uso y tienen un impacto de implantación en las nuevas empresas que están surgiendo, esto a su vez me ha enriquecido con conocimiento ya que nunca había manejado un lenguaje como éste y pienso que desarrollar en este tipo de proyecto fue sencillo gracias a mis capacidades desarrolladas en la universidad, también cabe mencionar que no había trabajado en un ambiente laboral que exigiera la cooperación entre grupos de trabajo y la sincronización de los mismos.

Ya para finalizar, recomiendo ampliamente el uso de esta tecnología gracias al buen proceso que sigue y a la buena administración e integración de los datos.

5. Glosario

Prototipo: Es un primer ejemplar que se fabrica de una figura, un invento u otra cosa, y que sirve de modelo para fabricar otras iguales, o molde original con el que se fabrica.

IDE: Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

SQL: Es un lenguaje específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

Suite: Un paquete de programas, programa, software, un paquete de aplicaciones, es el conjunto lógico de archivos y aplicaciones.

Java: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Job: Es una serie de pasos programada para ser ejecutada en una hora determinada la cual cumple una función.

Thread: Con significado en ingles de “Hilo” considerado como un subproceso que es una secuencia de tareas encadenadas muy pequeña que pueden ser ejecutadas.

Batch Insert: Usado en para realizar inserciones a una base de datos de una manera veloz y con muchos datos a la vez.

Control-M: Programa informatico que ayuda a la gestión, organización y automatización de procesos.

6. Referencias bibliográficas

[1] Vicen, Fernández. “Desarrollo de sistemas de información, una metodología basada en el modelo”. Editorial edicionsUPC, 2006. Disponible en: https://books.google.com.mx/books?id=Sqm7jNZS_L0C&lpg=PA1&dq=isbn%3A8483018624&hl=es&pg=PA7#v=onepage&q&f=false

[2] URL: http://metodologiarad.weebly.com/ Página principal del blog Metodología RAD, en ella se puede consultar información acerca las fases de la metodología RAD. Fecha de consulta: 12/octubre/2018

[3] URL: http://www.elaprendizdelprogramador.xyz/que-es-webmethods/ Página de El aprendiz del programador, en ella se puede consultar información acerca de webMethods y sus componentes. Fecha de consulta: 12/octubre/2018

[4] URL: http://www.dataprix.com/oracle-sql-developer Página de dataprix, en ella se puede consultar información acerca de Oracle SQL developer. Fecha de consulta: 12/octubre/2018

[5] URL: http://dia-installer.de/doc/en/intro-chapter.html#intro Página de introducción a DIA, en ella se puede consultar información acerca del programa DIA. Fecha de consulta 12/Octubre/2018

Universidad Politécnica de Puebla
Ingeniería en Informática

José Antonio Xochimitl Tlamani
Modesto Romero Martínez
Rebeca Rodríguez Huesca