Facultad de Ciencias

GESTIÓN DE PAGOS DE FACTURAS ASOCIADAS A TRANSPORTES MARÍTIMOS DE BUQUES PETROLEROS

(Management of payment of invoices associated with maritime transport of oil tankers)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Sergio García Fernández

Director: Patricia López Martínez

Codirector: Rubén Lorenzo Moratón

Julio-2022

Agradecimientos

Antes de comenzar me gustaría agradecer a todas las personas que me han ayudado a llegar hasta este momento. En primer lugar, me gustaría dar gracias a la universidad y a los profesores que me han dado clase durante estos cuatro años. Por enseñarme todo lo que pude aprender y por saber adaptarse a las situaciones adversas que hemos vivido para poder seguir con su labor de dar clase.

También quiero agradecer a NTT Data, empresa en la que he trabajado durante los últimos nueve meses. Por enseñarme y hacerme parte del equipo desde el primer momento. Por la cercanía y la confianza con la que se me ha tratado y sobre todo por la ayuda prestada.

Finalmente me gustaría agradecer a mi familia, amigos y compañeros con los que he compartido esta etapa de mi vida, ya que sin ellos no hubiera sido lo mismo. Siempre estaré infinitamente agradecido.

Tabla de contenidos

Índice de figuras

Índice de tablas

Resumen

La empresa Repsol dispone de un sistema denominado Triton que da soporte a las operaciones de compraventa de la empresa tanto de físicos, como derivados y contratos en mercados financieros. Cuenta también con una gran base de datos en la que se almacenan todos los datos generados en la operativa de la compañía. De manera paralela a Triton existe el sistema IMOS, el cual gestiona los desplazamientos de mercancías costeados por la empresa Repsol, lo que incluye gastos de personal, combustible y pagos portuarios, entre otros.

Estos dos sistemas se comunican entre sí, a través de servicios web, para el intercambio de información en ambos sentidos. Triton envía datos al sistema IMOS cuando se cierra una compraventa que supone un desplazamiento en barco de mercancía. Por su parte, IMOS comunica a Triton los costes del movimiento del barco para que desde Triton se gestionen los pagos y se almacenen en la base de datos.

El objetivo de este TFG es desarrollar un sistema de cancelación para el sistema Triton que permita la gestión de la cancelación de facturas notificada desde IMOS, manteniendo de esta manera la sincronía entre ambos sistemas. El desarrollo de esta funcionalidad requerirá realizar una fase de análisis previa del flujo de comunicación entre ambos sistemas, para diseñar el modo de incluir en él el proceso de cancelación.

La nueva funcionalidad se añadirá al servicio web existente en Triton, utilizando para ello el framework .NET y los lenguajes C# y SQL.

Palabras clave: Servicio web, flete, factura, IMOS, Triton, cancelación.

Abstract

The Repsol company has a system called Triton that supports the company's trading operations of both physical, derivatives and contracts in financial markets. It also has a large database in which all the data generated in the company's operations are stored. Parallel to Triton, there is the IMOS system, which manages the movement of goods paid by Repsol company, which includes personnel expenses, fuel and port payments, among others.

These two systems communicate with each other through web services, for the exchange of information in both directions. Triton sends data to the IMOS system when a sale is closed that involves a shipment of merchandise. While IMOS communicates to Triton the costs of the movement of the ship so that Triton can manage the payments and store them in the database.

The objective of this TFG is to develop a cancellation system for theTriton system that provides management for the cancellation of invoices notified from IMOS, thus maintaining the synchrony between both systems. The development of this functionality will require a preliminary analysis phase of the communication flow between both systems, to design the way to include the cancellation process in it.

This functionality will be added to the existing web service in Triton, using the .NET framework and the C# and SQL languages.

Keywords: Web service, freight, invoice, IMOS, Triton, cancellation.

1. Introducción

En este apartado se exponen el contexto donde se realiza el Trabajo de Fin de Grado y los objetivos que se definen para él.

1.1 Contexto

La empresa Repsol es una empresa española energética y petroquímica que realiza las tareas de exploración, producción, transporte y refinado de petróleo y gas. Utiliza para su gestión un sistema ETRM [1] (Energy Trading Risk Management System) construido a medida denominado Triton. El objetivo principal de este sistema es gestionar el seguimiento de todas las operaciones, resultados y riesgos que conlleva la compraventa de energía en función de la posición geográfica, el precio, la divisa y el crédito.

Triton da soporte a las operaciones de físico y productos derivados como contratos financieros en mercados regulados y OTC [2] (Over-The-Counter) para petróleo, CO2 y Energía. Es un sistema que se utiliza en diferentes localizaciones del mundo (Madrid, Houston, Singapur y Lima) y es utilizado por más de 500 usuarios diariamente. Es un sistema IaaS [3] (Infrastructure as a Service) desplegado en Azure, servicio de computación en la nube creado por Microsoft.

Por otro lado, Repsol utiliza un SaaS [4] (Software as a Service) comercial denominado Veslink IMOS para la gestión de los diferentes viajes que Repsol realiza con el fin de transportar mercancías, generalmente petróleo. Los operadores emiten las solicitudes de emisión, regularización y cancelación de facturas u órdenes de pago desde IMOS. Esto provoca la creación de las diferentes facturas u órdenes de pago en Triton y a su vez el envío de un aviso a otra plataforma hecha a medida por Repsol para la gestión de alertas, denominada GTA, para que el departamento de facturación procese las diferentes facturas/órdenes de pago y sean mandadas a la plataforma SAP con el fin de pagar los fondos.

Durante los diferentes viajes, Repsol debe asumir pagos anticipados por utilizar los puertos para sus operaciones. Esto implica la generación de una orden de pago en Triton con un importe provisional que debe pagarse al agente portuario. Una vez terminan las diferentes operaciones, el agente portuario envía una factura con el importe final y el operador desde IMOS deberá solicitar la regularización por diferencias del pago anticipado para solo pagar el importe restante. En ocasiones por cambio de acuerdos de última hora o por error del operador se deben cancelar estos pagos finales pero el sistema no manda este aviso al equipo de Global Service por lo que existe un riesgo de que los fondos no se devuelvan afectando a los resultados de REPSOL.

Triton e IMOS mantienen una comunicación basada en el intercambio de ficheros XML a través de servicios web. Esta comunicación permite el intercambio de toda la información necesaria para la gestión de facturas y fletamentos (o fletes, esto es, movimientos de mercancías por vía marítima). En Triton existen,

además, dos bases de datos que permiten el almacenado tanto de los datos de las facturas como de los mensajes enviados y recibidos y del resultado de las operaciones realizadas.

La empresa NTT Data lleva a cabo desde hace 15 años, lo que puede dar idea de la complejidad de este sistema, el soporte a la operativa diaria de Triton: la gestión de peticiones de usuario sobre el sistema, la corrección de las incidencias detectadas y el desarrollo de evolutivos.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar un sistema que permita gestionar la cancelación de los pagos asociados a un fletamento cuando el operador de fletamentos lo requiera, de manera que los sistemas IMOS y Triton se mantengan alineados y sincronizados. Esta nueva funcionalidad deberá ser añadida al servicio web en que estos dos sistemas basan su comunicación, teniendo a su vez que modificar o añadir procedimientos de la base de datos subyacente que permitan llevar a cabo la operativa requerida.

El desarrollo de esta funcionalidad requiere realizar una exhaustiva fase de análisis previa de los mecanismos de comunicación e integración entre los sistemas IMOS y Triton, centrado en la gestión de los fletes o facturas asociadas al movimiento de un barco para el traslado de mercancías entre diferentes puertos. Se deberá analizar el sistema Triton de recepción de mensajes emitidos por el sistema IMOS que permite la gestión de las facturas resultantes de los movimientos marítimos , así como el proceso de conexión con las bases de datos de Triton para el almacenamiento de la información relacionada con el flete.

Teniendo en cuenta los puntos anteriores, el Trabajo de Fin de Grado consta de los siguientes objetivos:

- Análisis del estado actual de la interfaz de interconexión entre los sistemas IMOS y Triton, así como de la conexión con las bases de datos de soporte.
- Desarrollo e integración del sistema de cancelación de facturas asociadas a un viaje que mantenga al sistema Triton alineado con el sistema IMOS, abordando todas las fases del proceso de desarrollo software, incluyendo la realización de pruebas y el despliegue.

1.3 Organización de la memoria

La memoria de este Trabajo de Fin de Grado está dividida en los siguientes apartados:

- Apartado 2. Metodología de la empresa y organización.
  En este apartado se comentará la metodología utilizada por la empresa para diseñar software, a su vez se comentará la organización interna de la misma.

- Apartado 3. Herramientas y entorno de trabajo.
  En el apartado de herramientas y entrono de trabajo se exponen las tecnologías utilizadas y la configuración del entorno de trabajo para llevar a cabo el desarrollo.
- Apartado 4. Análisis y especificación de requisitos.
  Se comentarán aquí los requisitos del sistema y se realizará un análisis completo del mismo incluyendo la captura de requisitos funcionales y requisitos no funcionales.
- Apartado 5. Diseño.
  Tras exponer los requisitos se presenta el diseño del sistema, donde se comenta el diseño, la interfaz y la arquitectura de la funcionalidad desarrollada.
- Apartado 6. Desarrollo.
  En el apartado se muestra cómo se lleva a cabo el desarrollo del sistema siguiendo el diseño previamente comentado, para cumplir los requisitos expuestos.
- Apartado 7. Pruebas y despliegue.
  Una vez se ha explicado el proceso de desarrollo se comentarán las pruebas realizadas y el proceso de despliegue del sistema en producción.
- Apartado 8. Conclusiones y trabajos futuros.
  Finalmente, se expone una conclusión sobre el desarrollo y el trabajo realizado en la empresa y se comentan los trabajos futuros sobre el sistema.

2. Metodología de la empresa y organización

En este apartado se expone la metodología de desarrollo aplicada en la empresa y su organización.

2.1 Metodología Scrum

Scrum [5] es un modelo de desarrollo de software ágil, en el que se aplican un conjunto de procesos y buenas prácticas para fomentar y mejorar el trabajo colaborativo y en equipo con el objetivo final de obtener los mejores resultados posibles en el producto software. Es una metodología que se basa en entregas parciales del producto software, que suponen grandes beneficios para la empresa receptora del producto. El modelo Scrum está altamente enfocado a entornos complejos con necesidad de adaptación, innovación, flexibilidad y donde se busque un resultado lo más inmediato posible.

Como podemos ver en la figura 1, en el modelo Scrum el tiempo de desarrollo se divide en periodos de tiempo denominados “Sprint”, de entre 1 y 4 semanas de duración. En estos periodos de tiempo se produce el desarrollo del producto software. El “Product Owner” (Dueño del producto) es el encargado de definir y priorizar el trabajo (Product Backlog) que se va a realizar durante el próximo sprint. Los objetivos de cada Sprint deben ser determinados (elegidos de entre los definidos en el Product Backlog) y comunicados al equipo scrum durante el Sprint Planning o reunión previa al sprint. Durante el sprint el Product Backlog no puede ser modificado.

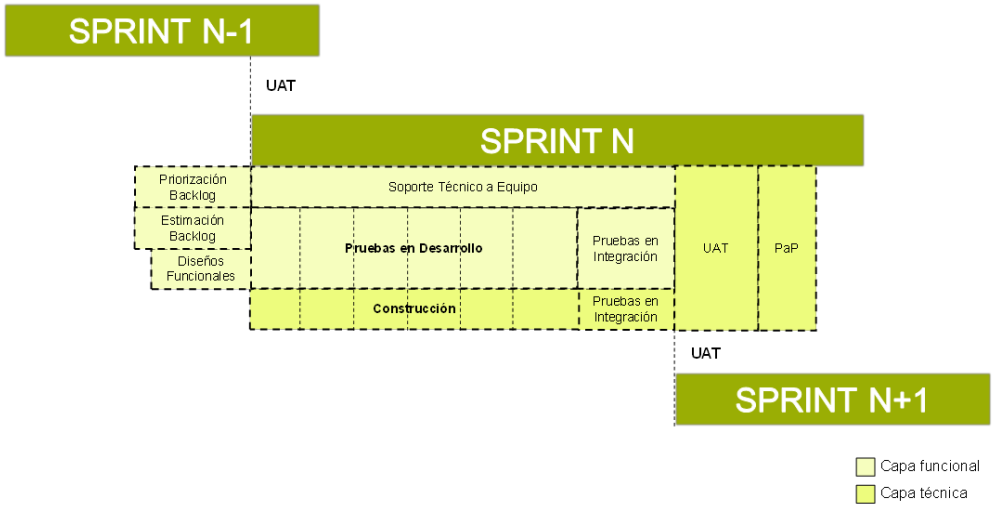

Figura 1. Modelo Scrum

2.2 Organización de la empresa

Para el mantenimiento y desarrollo de la plataforma Triton la empresa NTT Data cuenta con un equipo de 37 personas distribuidas entre las oficinas de Madrid y Santander. La empresa se divide en 5 verticales siguiendo el modelo de Repsol, por lo que cada vertical es paralela a la correspondiente vertical en el equipo de

Repsol. A su vez existen los departamentos de Cross, Power BI y Automatizaciones.

El equipo, siguiendo con la metodología DevOps [6] cuenta con desarrolladores y profesionales IT y un líder de equipo que se encarga de la supervisión y correcto funcionamiento de las verticales tanto de forma individual como colectiva.

El modelo utilizado por la empresa para el desarrollo es el modelo Scrum, como se ha mencionado anteriormente. En el caso del proyecto Triton, se utilizan Sprints de una semana que comienzan los miércoles tras realizar la subida a producción de los desarrollos finalizados y probados en los diferentes entornos. De forma diaria se realizan reuniones con el Product Owner para comentar la situación de desarrollo, y de forma semanal se realiza una reunión de las verticales con el líder de proyecto y otra reunión con todo el equipo Triton.

En estas reuniones se analiza la situación, se da un orden de prioridad a los diferentes desarrollos, se crea una estimación de tiempos y se realiza un plan de actuación que marcará el sprint. Para realizar la subida a integración los desarrollos deben estar terminados, probados en los diferentes niveles y de forma cruzada el día antes del despliegue de la release semanal, así como debidamente documentadas las pruebas realizadas en la plataforma Azure DevOps. Las pruebas cruzadas consisten en pruebas realizadas inicialmente por el desarrollador del producto software y posteriormente por el personal propietario del producto, en este caso Repsol.

Una vez se han superado las pruebas se está en disposición de subir los cambios realizados al servidor. Para ello se debe informar al equipo de despliegue incluyendo el desarrollo en el sistema de comunicación de despliegue donde se deben indicar todos los componentes a desplegar. De forma paralela, si durante el desarrollo se ha requerido modificar o crear procedimientos en las bases de datos, estos deben ser informados al igual que los desarrollos .NET. Para ello se cuenta con otro sistema de comunicación de despliegue de las bases de datos.

3. Herramientas y entorno de trabajo

En este apartado se resumirán las tecnologías y herramientas utilizadas así como el proceso de configuración del entorno de trabajo.

3.1 Tecnologías y herramientas

- C#
  Lenguaje de programación orientado a objetos multiparadigma desarrollado por Microsoft. Es el lenguaje utilizado para el desarrollo en el framework .NET, también desarrollado por Microsoft. Tiene sus raíces en los lenguajes C y C++ y es muy similar al lenguaje Java.

- XML
  Extensible Markup Language, lenguaje de marcado de propósito general. No utiliza etiquetas predefinidas, si no que estas son definidas por el usuario dando mucha libertad para crear esquemas propios.

- .NET
  Plataforma para el desarrollo de aplicaciones implementada por Microsoft, que permite el desarrollo multiplataforma y la creación de todo tipo de aplicaciones. Es un sistema que integra todas las herramientas de desarrollo de Microsoft: librerías, lenguajes, tecnologías, etc. en un mismo framework. Actualmente .NET es el sistema que agrupa las plataformas .NET Core, .NET Framework y Xamarin en un único Framework.

- SQL
  SQL es un lenguaje de consulta estructurado, diseñado para gestionar y facilitar el acceso a bases de datos relacionales. Basado en álgebra lineal, es un lenguaje de definición, manipulación y control de datos.

- Microsoft SQL Server Management Studio
  Microsoft SQL Server Management Studio es un entorno que permite la configuración, la administración y el desarrollo de infraestructuras SQL.

- Visual Studio
  Es un entorno de desarrollo integrado que permite la programación en múltiples lenguajes. Permite la creación de aplicaciones y servicios web con la plataforma .NET.

- Postman
  Es una aplicación que nos permite realizar pruebas de interfaces o servicios web. Se trata de un cliente HTTP que nos permite comprobar el funcionamiento de la resolución de peticiones HTTP de nuestro sistema.

- Azure DevOps
  Solución en la nube accesible vía web que facilita y permite la inclusión de la metodología DevOps en la empresa mediante diferentes funcionalidades disponibles para la gestión de proyectos.

3.2 Configuración del entorno de desarrollo

Tanto para el desarrollo como para la realización de pruebas en el sistema Triton necesitamos contar con una máquina virtual bien acondicionada con todas las herramientas utilizadas y con conexión a los diferentes entornos de desarrollo.

Para ello, la empresa Repsol proporciona una máquina a cada uno de los desarrolladores del equipo, la cual debe ser configurada por el usuario. Necesitaremos instalar primeramente las herramientas necesarias: Visual Studio, SQL Server Management Studio, Node.js y Angular CLI.

Triton es un sistema compuesto por cuatro entornos de desarrollo: local, desarrollo, preproducción y producción. Los entornos preproducción y producción cuentan a su vez con cuatro nodos cada uno, transparentes para el usuario.

Desde SQL Management Studio debemos tener acceso a las tres bases de datos que existen para Triton y a su vez deberemos crear una propia en local utilizando los datos del backup de la base de datos de Triton Producción. Esta base de datos local deberá ser actualizada cada semana con los nuevos datos del backup de Triton producción.

La gestión y administración del código fuente de Triton se lleva a cabo con Microsoft Team Foundation Server, lo cual permite llevar un control de versiones de las aplicaciones. Por lo que deberemos conectarnos al TFS de Repsol y crear un workspace donde descargaremos y compilaremos las ramas del proyecto requeridas. Deberemos cambiar los ficheros de configuración de la máquina “machine.config” y “web.config”. Tras todo esto tenemos que configurar el localhost para poder acceder a la aplicación con nuestro usuario y contraseña y más adelante poder realizar las pruebas del servicio web utilizando la aplicación y base de datos local.

4. Análisis y especificación de requisitos

En el apartado de análisis se explicará en detalle el flujo de comunicación entre Triton y se definirán en los requisitos de la nueva funcionalidad del servicio web.

4.1 Análisis

Actualmente Triton dispone de un sistema de comunicación con el sistema IMOS basado en servicios web para la gestión de fletamentos. Se quiere añadir una nueva funcionalidad al servicio web del sistema Triton que permita la gestión y trámite de la cancelación de facturas. Dicho servicio recibirá desde IMOS los mensajes que indican la operación a llevar a cabo y los datos sobre los que actuar.

Para diseñar el modo en que dar soporte a esta nueva funcionalidad, fue necesario llevar a cabo un análisis en profundidad del flujo de comunicación entre IMOS y Triton. A modo de resultado de este análisis y para entender mejor el funcionamiento del sistema, en el diagrama mostrado en la Figura 2 se representa el flujo que sigue la gestión de un viaje para transporte de mercancías, empezando por su petición de creación desde el sistema Triton, y mostrando como se realiza su gestión en IMOS y su posterior procesado en Triton:

- La operación comienza en Triton cuando se cierra una compraventa de petróleo que requiere un movimiento de la mercancía. En ese momento se realiza una petición de Triton a IMOS para crear un nuevo fletamento.
- El operador de fletamentos se encargará de gestionar el movimiento y reportar a Triton un informe con la estimación de costes del mismo.
- El itinerario del viaje puede ser modificado en cualquier momento del mismo.
- Una vez el viaje ha comenzado se puede crear una factura inicial denominada PDA que contiene unos costes aproximados de lo que será el viaje. Esta factura será enviada a Triton donde será procesada, almacenada en la base de datos y reportada a Global Service para su aprobación y pago.
- Una vez el viaje ha finalizado se creará la factura final del viaje denominada FDA, donde se recogen los gastos totales del movimiento. Esta factura al igual que la PDA es enviada a Triton, donde se procesa. En caso de que se reciba una FDA asociada a una PDA previa se deberá generar una regularización. La regularización no es más que la factura pendiente de pago por el viaje y se calcula como la diferencia entre la FDA y la PDA. Todas las facturas en Triton tienen dos estados posibles: procesado y no procesado, dependiendo de si han sido aprobadas o no por el equipo de Global Service.
- Además de recibir facturas, Triton puede recibir peticiones de regularización de una factura, las cuales se pueden solicitar una vez se ha creado la FDA para actualizar los costes de la misma.

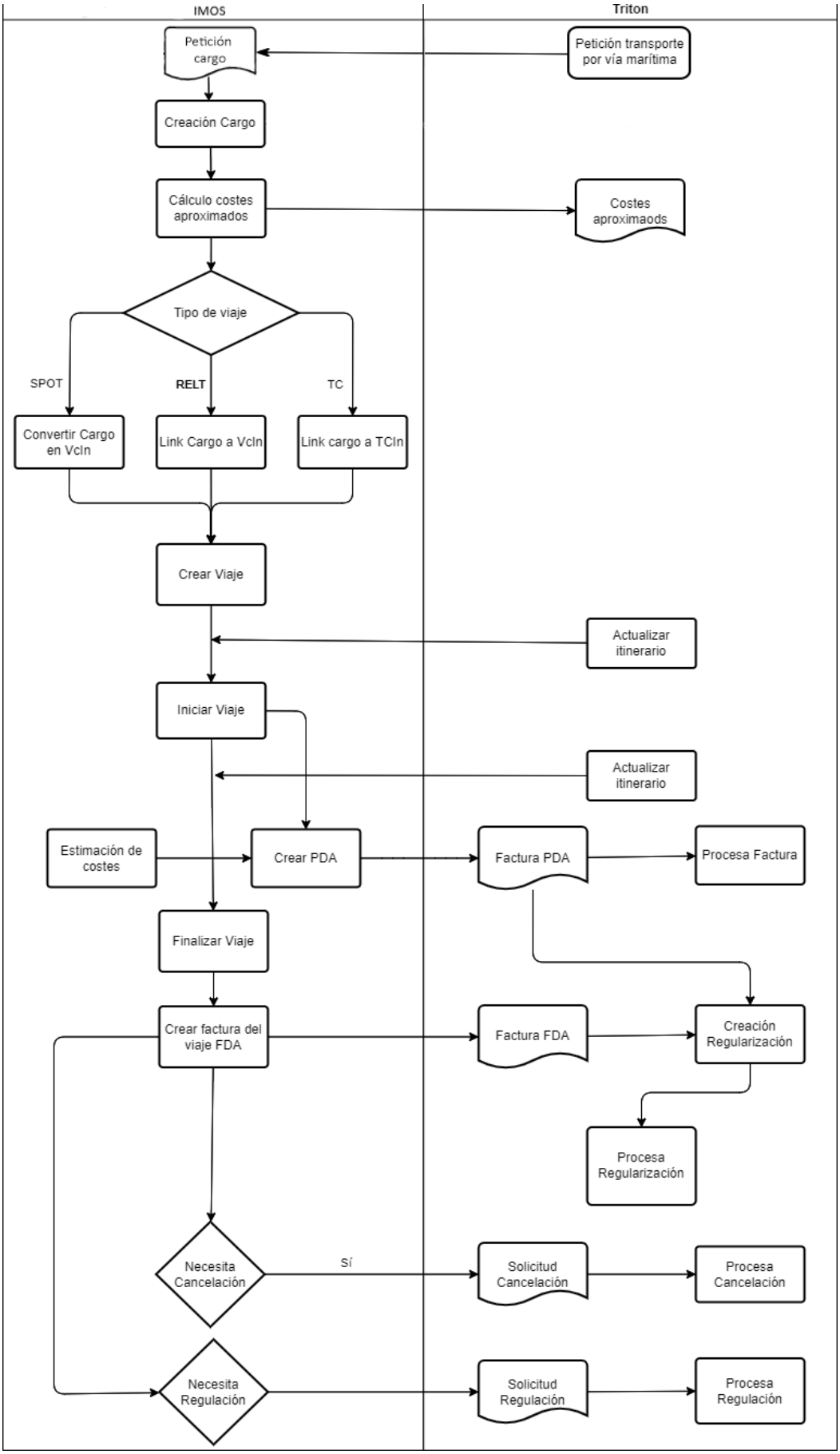

Figura 2. Comunicación IMOS-Triton para la gestión de fletamentos

A este flujo se quiere añadir la posibilidad de que Triton reciba también peticiones de cancelación de facturas, para cuyo procesado se va a requerir la modificación de la funcionalidad ya existente en cuanto a la gestión de las facturas (PDA y FDA) y las regularizaciones. En caso de recibir una petición de cancelación se deberá comprobar el estado de la factura ya que en caso de estar procesada no se podrá cancelar directamente si no que se deberá avisar al equipo de Global Service para que gestione la cancelación, pues la factura ya ha sido aceptada y se ha asumido su coste.

4.2 Requisitos funcionales

Tras conocer las necesidades del sistema pasamos a realizar la especificación detallada de requisitos funcionales, los cuales se exponen en la Tabla 1.

La captura de requisitos fue realizada entre el equipo de desarrollo y el Product Owner durante las reuniones que se realizan semanalmente.

Tabla 1. Requisitos funcionales

Identificador	Descripción
ReqF01	El sistema deberá ser capaz de procesar facturas PDA.
ReqF01.1	El sistema deberá ser capaz de recibir mensajes de creación de facturas PDA.
ReqF01.2	El sistema procesará el mensaje, almacenará la factura en la base de datos TritonDB y notificará el procesado en la base de datos Arquitectura.
ReqF01.3	El sistema reportará la nueva factura al equipo de Global Service a través del servicio de notificaciones.
ReqF01.4	El sistema notificará a IMOS el procesado de la PDA como respuesta de la petición HTTP.
ReqF02	El sistema debe ser capaz de procesar facturas FDA.
ReqF02.1	El sistema deberá ser capaz de recibir mensajes de creación de facturas FDA.
ReqF02.2	El sistema procesará el mensaje, almacenará la factura en la base de datos TritonDB y notificará el procesado en la base de datos Arquitectura.
ReqF02.3	El sistema creará una regularización donde se determine el coste a pagar.
ReqF02.4	El sistema reportará la nueva factura al equipo de Global Service a través del servicio de notificaciones.
ReqF02.5	El sistema notificará a IMOS el procesado de la FDA como respuesta de la petición HTTP.

ReqF3	El sistema deberá ser capaz de procesar solicitudes de regularización.
ReqF03.1	El sistema deberá ser capaz de recibir mensajes de regularización de una factura.
ReqF03.2	El sistema procesará los datos de la regularización.
ReqF03.3	El sistema reportará la nueva factura al equipo de Global Service través del servicio de notificaciones.
ReqF03.4	El sistema notificará a IMOS el resultado de la solicitud de regularización como respuesta de la petición HTTP.
ReqF04	El sistema deberá ser capaz de procesar peticiones de cancelación de facturas.
ReqF04.1	El sistema deberá ser capaz de recibir mensajes de cancelación de facturas.
ReqF04.2	El sistema procesará el mensaje y cancelará la factura directamente en caso de que no esté procesada en Triton.
ReqF04.3	El sistema deberá solicitar la cancelación de la factura al equipo de Global Service en caso de que esté procesada en Triton.
ReqF04.4	El sistema notificará a IMOS el resultado de la solicitud de cancelación como respuesta de la petición HTTP.

4.3 Requisitos no funcionales

Al tratarse de un servicio web sin interfaz gráfica los requisitos no funcionales son aquellos que abordan las necesidades de rendimiento, seguridad, disponibilidad, etc. En la tabla 2 se exponen los requisitos no funcionales definidos.

Tabla2. Requisitos no funcionales

Identificador	**Tipo**	**Descripción**
ReqNF01	Rendimiento	El tiempo de respuesta debe ser razonable teniendo en cuenta el posible volumen de la petición
ReqNF02	Seguridad	Se deben controlar los errores, informar y registrarlos para su análisis.
ReqNF03	Fiabilidad	El acceso a los datos debe ser seguro y atómico.

5. Diseño

Tras el análisis de los requisitos del servicio, pasamos a realizar el diseño del sistema para su posterior implementación, siguiendo la metodología utilizada en el resto de las funcionalidades del sistema.

5.1 Diseño global del sistema

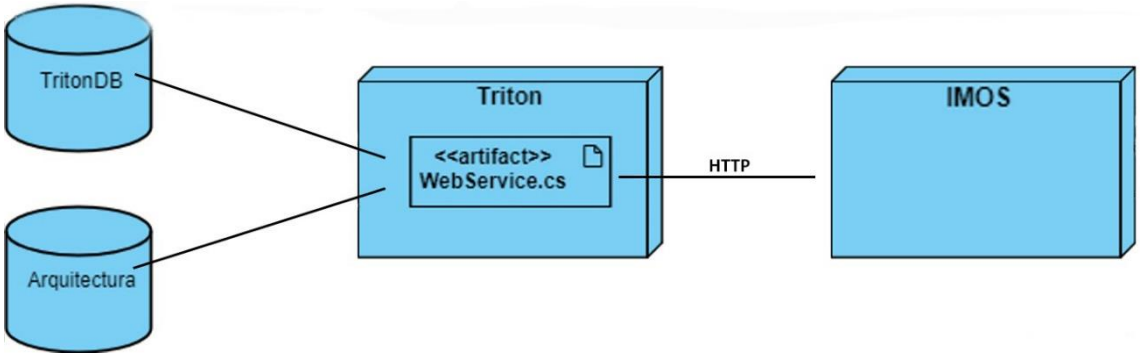

Figura 3. Diseño global del sistema

La figura 3 muestra la arquitectura global del sistema, con los sistemas Triton e IMOS, y como se ha comentado anteriormente, con dos bases de datos: TritonDB, que es la base de datos principal de Triton donde se cuenta con toda la información de la operativa diaria de Repsol, y Arquitectura, formada por un conjunto de tablas para la propia gestión del sistema.

TritonDB cuenta con 1166 tablas, es una gran base de datos de la cual, para la funcionalidad a implementar no se necesita conocer todas y cada una de las tablas, sino solo unas diez. En concreto, las tablas en las que se encuentra la información de las facturas, la cual deberá ser completada con nuevas facturas de fletamentos y accedida para conocer los estados y costes de las facturas anteriores así como las compañías involucradas, el tipo de producto, los puertos utilizados, los costes asociados a estos viajes y otros conceptos necesarios para el procesado de la factura.

Por otro lado la base de datos Arquitectura, cuenta con 76 tablas de las cuales solo una nos interesa para este proceso. Esta tabla será el lugar en el que se reporten los mensajes intercambiados entre IMOS y Triton, así como las acciones llevadas a cabo sobre el sistema Triton. Es decir, cuando se recibe un mensaje desde IMOS, Triton procesará el mensaje realizando la acción que se indique en el mismo, ya sea la creación de un viaje, la creación de una PDA o la cancelación de una orden de pago. En esta tabla aparecerá reportado el mensaje XML recibido, el tipo de mensaje, el identificador del elemento, la dirección del mensaje, el estado, la fecha y la respuesta que se produce a dicho mensaje en el sistema receptor. Mediante estos campos podemos conocer qué ha ocurrido y cómo ha procesado el sistema dicho mensaje. En el caso que nos aborda los mensajes serán siempre de IMOS a Triton, y las respuestas de la petición HTTP indicarán el resultado de la acción que acción se ha llevado a cabo.

5.2 Diseño de la interfaz de servicio

El servicio web de Triton dispone de diferentes URLs para recibir las peticiones referentes a los diferentes tipos de información que gestiona: facturas, viajes y fletamentos. El modelo que se sigue es el de una URL única para cada ámbito utilizando el cuerpo de la petición para indicar la acción concreta a realizar.

En el caso que nos ocupa, la URL utilizada para enviar los mensajes es /invoice, a través de la cual se realizan todas las gestiones sobre facturas. El sistema IMOS enviará una petición POST a esta URL con un mensaje en formato XML como cuerpo de la petición. En él se ha de definir el tipo de factura, los datos de la misma y que acción se quiere realizar.

XML es un lenguaje de marcado con etiquetas no predefinidas, lo que nos permite crear etiquetas propias, mediante las cuales se especifican los datos de la factura y las acciones que se deben realizar sobre las mismas. A continuación se describe el formato de datos definido para el intercambio de información. El fichero XML incluye un campo (action) en el que se especifica el tipo de acción que se ha llevado a cabo en IMOS y que debe ser replicado en Triton, así como los datos necesarios para la operación, identificados mediante etiquetas propias. En la Figura 4 podemos ver un fragmento de un ejemplo de fichero XML y sus campos (no se muestra completo debido a su extensión de alrededor de dos mil líneas).

```
<invoice...imosmsg:action="delete">
    ...
    <vendorNo>100006589</vendorNo>
    <vendorName>WOSHIP MARITIME SL</vendorName>
    <vendorShortName>WOSHIP</vendorShortName>
    <vendorExternalRef />
    <vendorReferenceCode />
    <vendorType>A</vendorType>
    <vendorCountryCode>ES</vendorCountryCode>
    <vendorCrossRef>0001336009</vendorCrossRef>
    <vendorUserProperties />
    <vendorCareOf>0</vendorCareOf>
    <vendorCareOfRef />
    <vendorCareOfCountryCode />
    <invoiceNo>RYTT-AP0000024258</invoiceNo>
    <revInvoiceNo>WS2200300</revInvoiceNo>
    ...
```

Figura 4. Fragmento de fichero XML con especificación de la acción IMOS y los datos de la factura

5.3 Diseño de la arquitectura del servicio

El diseño de la arquitectura es el proceso en el que se modela la estructura interna de un sistema, en este caso, del servicio web. Nos encontramos ante el caso de querer desarrollar un servicio web donde mediante el protocolo HTTP se reciben mensajes para su gestión. No contamos con capa front-end ya que los mensajes son emitidos desde otro sistema y los servicios web no cuentan

con interfaz gráfica. La capa back-end es parte de un sistema IaaS y está conectada a las bases de datos del sistema Triton para el acceso tanto a lectura como a escritura de los datos relativos a las facturas.

Como se muestra en la Figura 5, la arquitectura del servicio es una arquitectura en capas:

- **Controlador:** capa inicial que recibe los mensajes XML e invoca al método encargado de la gestión del mensaje pasándole el fichero XML recibido.
- **Lógica de negocio:** construida en .NET utilizando el lenguaje C#. Esta capa es donde se lleva a cabo la operativa y permite la gestión y tratamiento de los datos antes de su almacenado en la base de datos, así como la conexión con la misma para la lectura. A su vez esta capa de negocio se divide en otras dos para llevar a cabo la operativa.
  El primer nivel es el encargado de la propia gestión y procesado del mensaje XML. Mediante el método `ProcessMessageInvoice` se analiza el mensaje y el formato del XML para evitar posibles errores de sintaxis, se obtienen los datos de la factura y las acciones a realizar y se procesa dependiendo del tipo de acción o tipo de proceso que se quiera realizar.
  La segunda capa, el adaptador, es una capa intermedia que hace de enlace entre la capa de procesado del mensaje y la capa de acceso a datos. Implementa, entre otros, el método `CancellationRequest,` que realiza la gestión de errores antes de enviar la petición a la capa de acceso a datos, comprobando que los datos recibidos son válidos. También comprueba el estado de la conexión con la base de datos antes de enviar peticiones a la capa de datos, asegurándonos así que los datos llegaran a la última capa sin errores. En la Figura 5 se muestra más claramente la relación entre las capas.

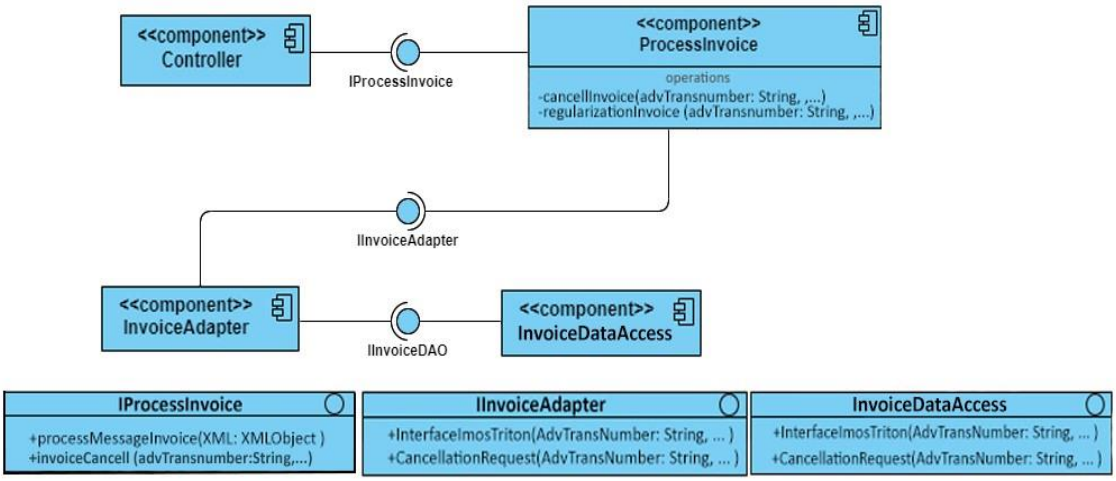

Figura 5. Diagrama de componentes [1]

[1] No se muestran la totalidad de parámetros de algunas operaciones por hacer más clara la figura, pues se trata de una amplia lista de parámetros.

- **Capa de acceso a datos:** Compuesta por las bases de datos SQL TritonDB y Arquitectura, y el componente a través del que se realiza el acceso a ellas, tanto para leer posibles datos requeridos en la operativa como para el almacenado de los datos procedentes de IMOS. Esta capa no accede de manera directa a los datos si no que hace uso de procedimientos almacenados en las bases de datos.

6. Desarrollo

Tras haber analizado los requisitos del sistema, resumido el funcionamiento de este y diseñado el modo de implementar la nueva funcionalidad, pasamos a explicar el proceso de desarrollo e implementación del software que se encargará de la gestión de cancelaciones de facturas.

Este desarrollo se dividirá en dos partes, un desarrollo en .NET para la capa de negocio y el desarrollo de un procedimiento para el acceso a la base de datos.

6.1 Desarrollo SQL

Como se ha comentado anteriormente, contamos con dos bases de datos completamente independientes. La gestión y el acceso a los datos no se realiza directamente desde los componentes del servicio, si no que éste hace uso de procedimientos almacenados de la base de datos a los que les suministra los datos requeridos para su funcionamiento.

Para la implementación de la funcionalidad requerida, se ha desarrollado un nuevo procedimiento almacenado en la base de datos TritonDB que permite leer datos sobre las facturas y modificar las mismas dependiendo de su estado. El funcionamiento de este procedimiento es el siguiente:

- En caso de que el estado de la factura sea borrador, se pueden modificar directamente los costes de la factura, por lo que en este caso simplemente se actualiza la tabla de la base de datos TritonDB con los nuevos datos de la factura reportados por IMOS.
- En caso de que la factura ya esté procesada en Triton, este procedimiento devolverá los datos de la factura con los costes asociados al número de viaje que se recibe desde IMOS. Los datos obtenidos corresponden a la factura que ya ha sido procesada en Triton y que se quiere regularizar o cancelar desde IMOS.

6.2 Desarrollo .NET

El siguiente desarrollo se va a dividir en tres subdesarrollos siguiendo el modelo en tres capas, mostrado en la Figura 5, para facilitar su comprensión.

6.2.1 Capa de procesado

Disponemos de un método `ProcessMessageInvoice` que es llamado desde el controlador de mensajes cuando se realiza un POST a la url /invoice utilizando como cuerpo un elemento XML el cual es pasado al método anteriormente citado. Dentro de este, realizaremos las gestiones necesarias en función de los requerimientos de IMOS.

En primer lugar se realiza la gestión de posibles errores del fichero XML, para lo cual se dispone de un método auxiliar que se encarga de validar el mensaje XML contra su correspondiente XML Schema. En caso de error se inserta el mensaje en la base de datos Arquitectura junto con un mensaje indicando el motivo del error. Si la validación se produce de forma correcta se añade también el mensaje

a la tabla Arquitectura indicando en este caso que el mensaje está procesándose y se sigue con el resto de las gestiones.

A continuación, se obtienen los datos del mensaje, en caso de encontrarnos con una FDA accederemos al procedimiento almacenado de TritonDB que se ha comentado anteriormente (a través de un método del adaptador), que nos devolverá los datos de la factura PDA asociada a esta factura FDA. En caso de ser una factura PDA no se realizará ninguna petición a la base de datos ya que esta factura PDA es la primera factura del viaje.

A continuación, para ambos tipos de facturas, se obtienen del XML los diferentes costes asociados a la factura y los datos relacionados con estos costes, como las compañías involucradas, los tipos de coste o la unidad utilizada. Se comprueba que los costes han sido creados o modificados en IMOS, esto es, no están sincronizados en ambas plataformas, para lo cual se cuenta con una etiqueta del XML en la que se indican los costes añadidos en cada plataforma. Una vez comprobado que existen costes no reflejados en Triton se realiza el procesado de los mismos para su almacenado en la base de datos TritonDB, en este caso se llama a un método del adaptador encargado de añadir o actualizar los costes de la factura.

En caso de que el mensaje de IMOS sea un mensaje Reversed, tipo de mensaje que indica que se quiere realizar una cancelación o regularización, se gestiona según el tipo de petición:

- En el caso de las regularizaciones se debe escribir en la base de datos Arquitectura, informando de la petición de regularización para la factura indicada.
- En caso de cancelación se llama al método encargado de gestionar la cancelación, `cancelInvoice`, pasándole los datos de la factura. Este método recopilará todos los costes asociados a la factura que se quiere cancelar, y en función del estado de la factura en Triton, realizará lo siguiente:
    - Las facturas en estado borrador se eliminarán sin necesidad de realizar una solicitud de cancelación, para lo que llamamos a un método de la clase adaptador al que le pasamos los datos de la factura.
    - Si la factura no se encuentra en estado borrador se debe realizar una solicitud de cancelación mediante el método `cancellationRequest`, para lo cual se deben actualizar varios campos de la tabla general de facturas para la factura indicada. Con estas indicaciones el equipo de Global Service deberá ser quien elimine estas facturas posteriormente. Tras esta gestión se reporta en la base de datos Arquitectura que se ha realizado la solicitud de cancelación.

En cualquiera de las situaciones, el resultado de la operación se reporta a IMOS como texto plano en el cuerpo de la respuesta de la petición HTTP. En esta

respuesta se indicará si el procesado del mensaje ha sido válido o no y en caso de error, por qué se ha producido el mismo.

6.2.2 Capa adaptador

La capa adaptador es una capa intermedia entre la capa de procesado y la capa de acceso a datos, se encarga de la gestión de errores antes del acceso a la base de datos. En esta capa disponemos de varios métodos utilizados en nuestro desarrollo los cuales siguen todos un mismo esquema:

- En primer lugar, se realiza la comprobación del estado de la base de datos, lo que consiste en comprobar la conexión para el acceso tanto a lectura como a escritura antes de realizar invocaciones en la capa de acceso a datos.
- En segundo lugar, se realiza la validación de los datos proporcionados, comprobando que todos son válidos respecto a la lógica de negocio.
- A continuación, cada método realizará su lógica correspondiente, invocando a los métodos de la capa de acceso a datos.
- Además, en esta capa se realizan una serie de medidas para la obtención de información sobre aspectos relacionados con rendimiento y trazabilidad. Se realiza un cálculo de los tiempos empleados en el acceso a la base de datos para reportarlos, registrando por cada invocación información acerca de la operación invocada, los datos utilizados, los tiempos de ejecución y el resultado.
  Se realizan dos medidas de tiempo, una total del método del adaptador y una solo de la capa de acceso a datos. Tras tomar el tiempo de inicio total, se comprueban los permisos y si el servicio está habilitado. En caso de que todo esté correcto se toma el tiempo de inicio de la capa de acceso a datos y se llama al método correspondiente de dicha capa gestionando las posibles excepciones que salten en la ejecución. Cuando el acceso a la base de datos ha terminado, ya sea de forma correcta o errónea se vuelven a tomar los datos del tiempo de finalización. Y finalmente se registra la actividad con los tiempos y el resultado de la operación: correcto si no ha producido excepciones y erróneo en el caso contrario.

6.2.3 Capa de acceso a datos

En esta capa se realiza la conexión a la base de datos tanto para lectura como para escritura. Al igual que en la capa anterior, contamos con diferentes métodos que siguen la misma estructura:

- En primer lugar, se crea una lista de parámetros SQL para pasarle al procedimiento de la base de datos, para ello se realiza una conversión de los datos pasados como parámetros a formato SQL y una vez convertidos se almacenan en una lista para su ejecución en el procedimiento almacenado.
- Estos datos son pasados a una función genérica que recibe como entradas el procedimiento a ejecutar, los parámetros del procedimiento y la base de datos en la que se encuentra. Esta función se encarga de crear

la conexión con la base de datos o de reutilizar una conexión existente para lanzar el procedimiento con los parámetros SQL indicados. En caso de error en la ejecución se reportará el error obtenido.

7. Pruebas y despliegue

Se comentará en este apartado las pruebas realizadas y el procedimiento de despliegue del sistema en producción.

7.1 Pruebas

Una vez comentado el desarrollo del servicio pasamos a documentar el proceso de pruebas que se ha llevado a cabo para comprobar su funcionamiento. Conviene destacar que las pruebas realizadas pertenecen al nivel de pruebas de integración, o pruebas funcionales.

Para la realización de pruebas y su documentación se hace uso de Azure DevOps, donde se creará un Product Backlog Item (PBI) por cada desarrollo a probar. A este PBI se le asocia una batería de pruebas, definidas como Shared Steps, las cuales deben estar acordadas por el equipo de desarrollo y la vertical. Los Shared Steps corresponden con cada uno de los pasos llevados a cabo durante las pruebas de funcionamiento, como pueden ser comprobar la recepción del mensaje, la validación del fichero XML, etc. La idea principal es disponer de un conjunto de Shared Steps que se adapten a diferentes casuísticas y que sean utilizadas como plantillas para cada uno de los flujos diferenciados de la aplicación. Estos Shared Steps buscan ser reutilizables en los diferentes desarrollos a probar utilizando un conjunto de ellos para adaptarse a los requisitos de la nueva funcionalidad.

Una vez tenemos definidos los Shared Steps necesarios para realizar pruebas en nuestro desarrollo debemos definir un Test Case. El Test Case agrupa uno o más Shared Steps para realizar las pruebas completas de la casuística desarrollada. En este caso, cada uno de estos Test Case se relaciona directamente con una funcionalidad del servicio web. En concreto, se definieron los siguientes Test Case:

- Probar cancelación de FDA ya procesada en Triton.
- Probar cancelación de PDA ya procesada en Triton.
- Probar cancelación de FDA no procesada en Triton.
- Probar cancelación de PDA no procesada en Triton.
- Probar cancelación de PDA procesada en Triton tras realizar la cancelación de FDA.
- Probar cancelación de PDA no procesada en Triton tras realizar la cancelación de FDA.
- Probar cancelación de Flete.
- Probar cancelación de costes secundarios.

Una vez se han definido las pruebas a realizar se comienza con su ejecución, realizada de forma manual a través de la herramienta Postman, y registrando a su finalización en Azure DevOps la siguiente información por cada prueba: el entorno en el que se realiza, las incidencias de la prueba, como podría ser un deadlock, o un retraso en el procesado y el resultado final. En caso de que las pruebas finalicen de forma positiva se podría realizar la subida al siguiente

entorno de desarrollo, donde se deberían volver a realizar las mismas pruebas desde el inicio. En caso de que el resultado sea no favorable debemos indicar cual ha sido el error en la plataforma Azure DevOps, las causas y los cambios a realizar. Una vez corregido el error se debe repetir toda la batería de pruebas desde el principio, siguiendo la misma metodología. El histórico de las pruebas realizadas se mantendrá de este modo, almacenado en la plataforma Azure DevOps.

En el caso que nos ocupa la batería de pruebas a ejecutar suponían realizar el flujo completo que siguen los diferentes mensajes emitidos por IMOS desde que se lanzan hasta que los datos son almacenados en la base de datos de Triton. Para esto se utilizó Postman, una herramienta que nos permite enviar peticiones HTTP al servicio web y comprobar el resultado retornado. Indicaremos en esta aplicación la URL /invoice, donde se enviarán los mensajes los cuales contendrán en el cuerpo el fichero XML con los datos de la factura, así como la operación a realizar, como se ha mostrado en figuras anteriores. Para realizar las pruebas utilizaremos el repositorio localhost el cual apunta a nuestra base de datos local, por lo que Postman deberá apuntar a la URL /invoice de nuestro localhost. Para generar los mensajes XML de forma correcta y utilizar mensajes reales haremos uso del sistema Veslink IMOS en modo mantenimiento. Esto nos permite obtener un mensaje XML como el que se genera en situaciones reales para realizar pruebas más fiables. Una vez se ha generado el fichero XML este se inserta en el cuerpo del mensaje en Postman.

El entorno Visual Studio nos permite hacer un debugging muy completo de los que está sucediendo en la aplicación cada vez que se envía una petición a través de Postman, haciendo uso de los puntos de interrupción y la ejecución guiada. De esta manera se puede comprobar el estado del sistema en cada momento y si se está realizando de forma correcta el procesado de las diferentes facturas.

7.2 Despliegue

Como se ha mencionado anteriormente, Triton es un sistema muy voluminoso con una base de datos muy extensa. A la hora de desarrollar o modificar procedimientos en la base de datos o del servicio web, únicamente utilizamos cierta parte de esta por lo que el procedimiento a seguir consiste en crear dos nuevos proyectos en Visual Studio, partiendo del proyecto compartido, los cuales serán denominados Partial y PartialWork. Estos dos nuevos proyectos serán creados seleccionando los procedimientos de TritonDB que puedan verse modificados durante nuestro desarrollo. El proyecto Partial contendrá el estado inicial de estos objetos, mientras que PartialWork será nuestro lugar de trabajo. Una vez terminado el desarrollo, contar con el proyecto Partial nos permitirá realizar una fusión (merge) sobre este de forma mucho más rápida que si tuviéramos que hacerlo sobre el total de TritonDB, acelerando así el proceso. Se debe tener en cuenta que a la hora de realizar el despliegue, se utilizan scripts donde se definen los cambios que se han insertado. Para generar este script de forma más rápida se hacen uso de estos dos proyectos parciales evitando tener que realizar la comparación con todo el proyecto. Con los scripts generados el

equipo de despliegue de base de datos será capaz de añadir las modificaciones desarrolladas.

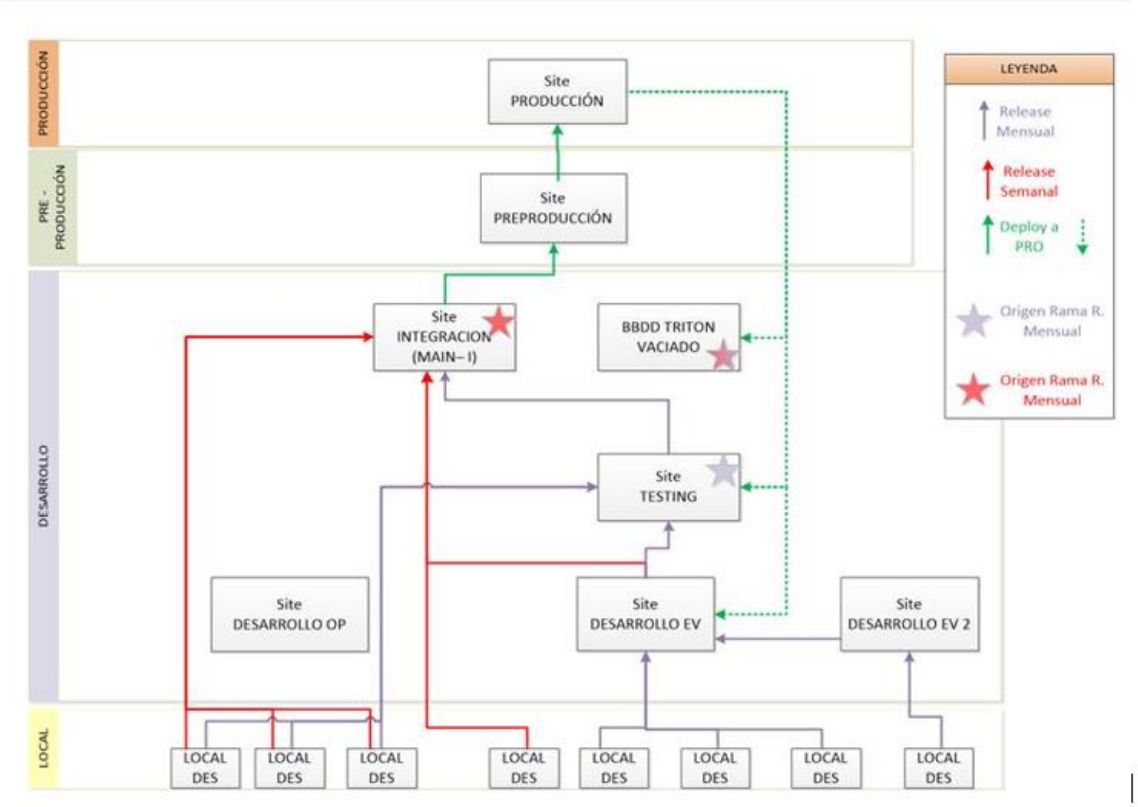

Figura 6. Disposición de entornos de desarrollo Triton

Tras terminar el periodo de pruebas en cada uno de los entornos deberemos realizar la subida de nuestro desarrollo al siguiente entorno. Como se muestra en la figura 6, Triton dispone de cuatro entornos distintos: local, desarrollo, preproducción y producción, por lo que las pruebas se realizan en todos los entornos anteriores a producción. Las subidas a los diferentes entornos las realiza el equipo de evolución y release. La subida a cada uno de los entornos tiene un día marcado a la semana, por lo tanto, para que esta se realice deberemos tener todas las pruebas terminadas y debidamente documentadas en el Excel de desarrollos, tanto del servicio web como de SQL. En este Excel se informa de los cambios realizados en el desarrollo, así como el estado de las pruebas realizadas, para que el equipo de despliegue pueda realizar la subida de los desarrollos finalizados.

8. Conclusiones y trabajos futuros

En este apartado se exponen las conclusiones obtenidas tras la realización de este Trabajo de Fin de Grado así como los posibles trabajos futuros sobre el sistema Triton.

8.1 Conclusiones

El objetivo de este Trabajo de Fin de Grado era realizar un sistema capaz de gestionar la cancelación de las facturas provenientes del sistema IMOS en Tritón siguiendo la metodología utilizada en la empresa. Este objetivo ha sido cumplido y actualmente el desarrollo se encuentra en producción sin errores registrados.

Para conseguir este objetivo se han necesitado muchas horas y trabajo de análisis. Triton es un sistema ETRM que lleva activo 15 años funcionando, esto complica mucho su análisis y su entendimiento. A su vez para poder lograr los objetivos marcados se deben conocer los requisitos y para ello ha sido necesaria mucha investigación y formación sobre el funcionamiento de un sistema ETRM y sobre la compraventa de petróleo.

Por otro lado, otro de los objetivos a nivel más personal de este proyecto era adaptarme a la metodología de trabajo de una gran empresa en un proyecto de gran importancia, la cual se ha conseguido también de manera satisfactoria. Para mí la posibilidad de haber trabajado estos tres meses en el proyecto Triton ha supuesto un grandísimo aprendizaje, afianzando conceptos aprendidos en la carrera y obteniendo mucho conocimiento nuevo sobre tecnologías nuevas o lenguajes desconocidos anteriormente. Sin embargo, creo que lo más importante es la experiencia que me llevo de haber trabajado en un gran equipo con un nivel alto de exigencia, conocer como es trabajar en un equipo de desarrollo software, la comunicación con los clientes, la adaptación a nuevos entornos o tecnologías y la forma de abordar los errores o contratiempos que se presentan en estos proyectos.

8.2 Trabajos futuros

Tras la finalización del desarrollo del sistema de cancelación y su despliegue en producción, se debe realizar un mantenimiento del sistema completo durante el tiempo que Triton sea operativo. En definitiva, no se trata de un proyecto que finalice con el despliegue si no que se sigue analizando y mejorando todas las semanas a través del equipo de mantenimiento que se encarga de solucionar los errores detectados durante la ejecución.

Triton es un sistema que debe adaptarse a todos los cambios que surgen en la legislación y trading de petróleo, por lo que está en constante desarrollo y adaptación. De manera reseñable se puede destacar la intención de migrar todas las interfaces web al Framework Angular.

Referencias

[1] Inatech - What is Energy Trading Risk Management (ETRM) software? https://www.inatech.com/blog/what-is-energy-trading-risk-management-etrm-software Recuperado el 11-07-2022

[2] BBVA - Qué son los mercados Over The Counter (OTC) https://www.bbva.com/es/que-son-los-mercados-over-the-counter-otc Recuperado el 11-07-2022

[3] RedHat - ¿Qué es la IaaS? https://www.redhat.com/es/topics/cloud-computing/what-is-iaas Recuperado el 11-07-2022

[4] Oracle - ¿Qué es SaaS (Software como servicio)? https://www.oracle.com/es/applications/what-is-saas Recuperado el 11-07-2022

[5] Scrum.org - ¿Qué es Scrum? Recuperado el 07-07-22 de https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf Recuperado el 11-07-2022

[6] Microsoft - ¿Qué es DevOps? Recuperado el 07-07-22 de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/ Recuperado el 11-07-2022