Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención Tecnologías de la información)

Desarrollo y despliegue de un sistema CRM escalable basado en Microsoft Azure, Dynamic CRM 365 Online y Azure Functions

Autora:
Dña. Alba Francisco Gutiérrez

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención Tecnologías de la información)

Desarrollo y despliegue de un sistema CRM escalable basado en Microsoft Azure, Dynamic CRM 365 Online y Azure Functions

Autora:
Dña. Alba Francisco Gutiérrez

Tutor:
D. Benjamín Sahélices Fernández

Tutor:
D. Álvaro Estébanez Barrena

Agradecimientos

Me gustaría agradecer principalmente a mi tutor Benjamín Sahelices por sus consejos y apoyo en el desarrollo de este proyecto.

También me gustaría agradecer a mi tutor dentro de la empresa Everis, Álvaro Estébanez, promotor de la idea detrás de este TFG, por haberme dado la oportunidad de llevarlo a cabo.

También quiero agradecer a Alberto Casero, co-tutor de este trabajo dentro de la empresa aunque no aparezca en la documentación.

Me gustaría agradecer a mi familia por todo su apoyo, no sólo durante el desarrollo de este proyecto, si no siempre. Gracias a ellos, he podido realizar la carrera que deseaba y emprender cosas que sin ellos no me hubiera atrevido.

Y una mención especial a mi pareja, por todo su apoyo y cariño. Siempre dispuesto a escucharme y ayudarme en cualquier circunstancia. Sin su apoyo, este proyecto no hubiera podido llevarse acabo.

Resumen

El proyecto se basa en la tecnología en la nube, conocida también como servicios en la nube, informática en la nube, nube de cómputo, nube de conceptos o simplemente "la nube", la cual es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.

Utilizando de base dicha tecnología, se elaborará una solución que permita mantener la persistencia de los registros almacenados en un sistema en la nube, como lo es *Microsoft Dynamics 365*, en un sistema de bases de datos de Oracle. De esta manera, se pretende mantener un duplicado de la información, que se considere más sensible, en otro sistema.

El desarrollo de esta solución se ha enfocado a que la información se almacene de forma autónoma y desatendida. Para ello se ha creado un conjunto de *Azure Functions*, que son un servicio de proceso sin servidor que permiten ejecutar código a petición sin necesidad de aprovisionar ni administrar explícitamente la infraestructura, que se encargarán de extraer los datos de *Microsoft Dynamics 365* y almacenarlos en el sistema de base de datos de Oracle, utilizando el sistema de colas de *Service Bus* como intermediario.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

La tendencia actual, en lo que a procesamiento y almacenamiento de datos se refiere, es obtener y proveer estos servicios en la nube. Es una causa directa de las múltiples ventajas, que este nuevo modelo ofrece al mundo empresarial. Atendiendo a su menor complejidad, a nivel físico, con la que se puede escalar los sistemas, siendo así, más accesible la manera de distribuir los contenidos, es una tendencia que tiene una fuerte presencia y que se instaura en muchos ámbitos de aplicación. A pesar de ello, la necesidad de tener un respaldo que asegure la disponibilidad y redundancia de los datos ha llevado a muchas empresas a duplicar los registros que tienen almacenados en la nube en sistemas propios como son las bases de datos.

Como es evidente, duplicar estos datos lleva a la necesidad de generar sistemas que automaticen la réplica de los datos alojados en la nube en otros sistemas. Por ello, a lo largo del siguiente proyecto, se elabora una propuesta que permite esto mismo. A continuación, se detalla las herramientas y sistemas que se utilizan para alcanzar dicho objetivo.

1.2. Objetivos

El objetivo de este proyecto es desarrollar un sistema que automatice la persistencia de los datos almacenados dentro de un sistema *Microsoft Dynamics 365* , explicado en detalle en la sección 2.2, en una base de datos Oracle, también detallado en la sección 2.3. Para ello, se utiliza las *Azure Functions*, que se detalla en la subsección 2.1.2.

Para permitir la comunicación entre las *Azure Functions*, que extraen datos del sistema *Microsoft Dynamics 365* y las *Azure Functions*, que gestionarán la comunicación con el sistema de bases de datos de Oracle se utilizarán el servicio de mensajería de *Service Bus*, explicado en detalle en la subsección 2.1.1. Se utilizará como intermediario de estos componentes los *Topics*, mediante los cuales se intercambiarán los mensajes usando las *Suscription* para como elemento de recepción del mensaje para los consumidores.

Al ser un sistema distribuido, se debe implementar mecanismos en caso de fallo. Un objetivo de este trabajo, es realizar un protocolo que abarque tres tipos de fallo:

- Fallos al intentar manejar mensajes con un formato erróneo dentro de la base de datos de Oracle.
- Fallos al intentar comunicarse con la base de datos Oracle.

- Fallos inesperados dentro del procesamiento de las *Azure Functions* que gestionan los mensajes extraídos de *Service Bus*.

La política de reintentos asociada a estos errores se describirá en detalle en el presente trabajo, además de lo mecanismos implementados para mantener una coherencia entre los datos almacenados dentro de *Microsoft Dynamics 365* con los datos que se encuentran en la base de datos de Oracle.

1.3. Tareas del proyecto

Para la realización de este proyecto, el cual pertenece a la asignatura de Trabajo de Fin de Grado, se han realizado diferentes tareas. Se ha intentado seguir un procedimiento adecuado y aceptado dentro del marco de la ingeniería.

- En primer lugar, se analizó la aplicación de *Microsoft Dynamics 365*. Fue necesario investigar la manera en que genera las entidades y como almacena sus registros internamente para poder extraerlos correctamente y aprovechar las ventajas que ofrece.
- Por otro lado, también fue necesario investigar *Azure*. Es una plataforma informática en la nube la cual ofrece un conjunto de herramientas muy útiles, en especial las *Azure Functions*, que son las herramientas para elaborar este proyecto. Una vez que el marco queda definido es necesario familiarizarse con los lenguajes que se utilizan en el proyecto.
- La tarea de planificación permite controlar el progreso del proyecto, estudiar su viabilidad y calcular el coste final del mismo, además de permitir definir el alcance del mismo y su duración. En esta fase del proyecto, también es necesario definir que metodología de desarrollo se implementa.
- El análisis de los requisitos que se piden, incluyendo los funcionales y los no funcionales, y partir de ellos, obtener los casos de uso y las herramientas que, finalmente, se implementan en el proyecto.
- La realización del proyecto final en base de esta fase de análisis y la información obtenida.
- La implementación del proyecto describirá las etapas por las cuales ha pasado el proceso de desarrollo y las razones que llevaron a tomar ciertas decisiones de diseño, siempre basándose en los requisitos proporcionados por el cliente. El despliegue del sistema dentro *Azure*, por cuestión de licencias y alcance de los requisitos del proyecto, no se llevará a cabo y permanecerá simulado en un entorno local.
- La realización de la batería de pruebas y resolución de errores en relación a los fallos detectados mediante las mismas.
- Finalmente, se agrupa esta información en el presente documento y se prepara la documentación necesaria.

Capítulo 2

Entorno tecnológico

En las siguientes secciones, se explica más en detalle las particularidades de las tecnologías que se han utilizado para desarrollar el proyecto.

2.1. Azure

Azure es una plataforma de nube completa que puede hospedar las aplicaciones existentes, simplificar el desarrollo de nuevas aplicaciones o mejorar las aplicaciones locales. *Azure* integra los servicios en la nube que son necesarios para desarrollar, probar, implementar y administrar aplicaciones, mientras se aprovechan las ventajas en la nube.

Con el hospedaje de las aplicaciones en *Azure*, se puede empezar con tamaño pequeño y escalar fácilmente una aplicación a medida que aumente la demanda de los clientes. *Azure* ofrece también la confiabilidad que se necesita para las aplicaciones de alta disponibilidad, e incluye conmutación por error entre diferentes regiones. *Azure Portal* le permite administrar fácilmente todos los servicios de *Azure*. También puede administrar los servicios mediante programación, con las API y las plantillas específicas del servicio [1].

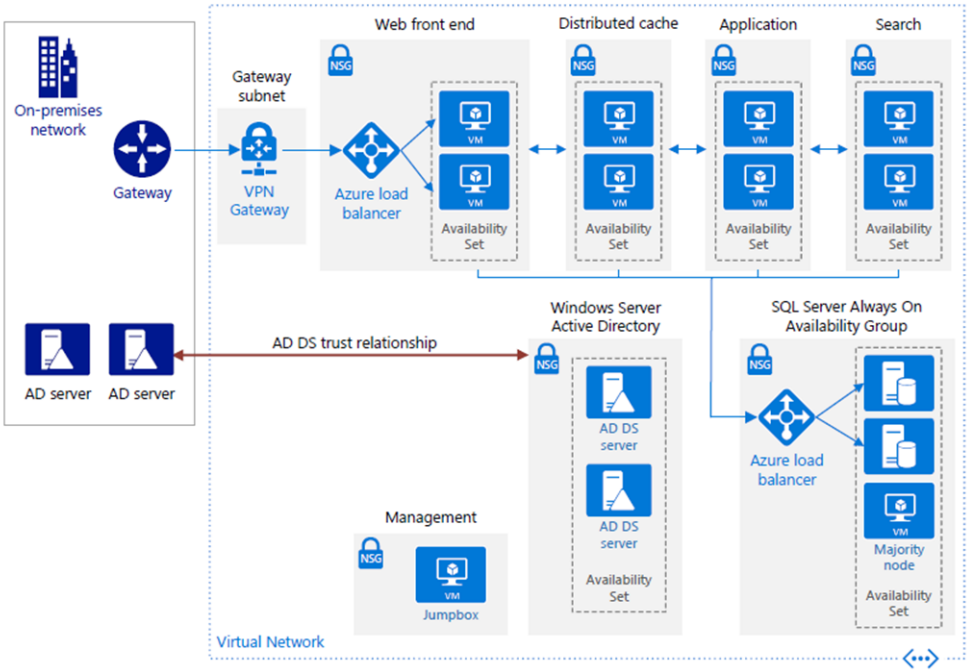

Figura 2.1: La plataforma de servicios de *Azure* admite aplicaciones que se ejecutan en la nube y en local.

Azure proporciona las ventajas que ofrece la nube pero, además, es una solución abierta y flexible.

Azure Cloud es compatible con una variedad de sistemas operativos, idiomas, herramientas, plataformas, utilidades, y marcos. Es compatible con *Linux y Windows, SQL Server, MySQL, Postgres..., C#, Python, Java, Node.js, Bash* y más lenguajes, *MongoDB y DocumentDB Bases de datos NoSQL, y Jenkins* para *VSTS* como herramientas de integración continua.

Toda la idea detrás de este ecosistema está en permitir a los usuarios poder elegir el lenguaje, la plataforma y el sistema operativo, la base de datos, y el tipo de almacenamiento que les resulte más conveniente, además de herramientas y utilidades. Esto está sujeto a la premisa de que los usuarios no tienen que estar ligados a una tecnología específica, sino concentrarse en crear soluciones de negocio. *Azure* es compatible con la elección del usuario de la tecnología que más se le adapte.

Por ejemplo, *Azure* proporciona disponibilidad de todos los populares (código abierto o comercial) entornos de bases de datos. *Azure* proporciona el servicio *Azure SQL, MySQL* y *Postgres PaaS*. Eso proporciona un ecosistema de *Hadoop* y ofrece *HDInsight*, un *PaaS* basado en *Apado Hadoop* al 100 % servicios. También proporciona Hadoop en la implementación de *VM* de Linux para clientes que prefieren enfoque *IaaS*.

Azure también proporciona un servicio de caché Redis y soporta otros populares entornos de bases de datos, como MongoDB, Couchbase, Oracle y muchos otros como la implementación de IaaS.[2] Windows *Azure* hace dos cosas principales: ejecuta aplicaciones y almacena sus datos. En las siguientes secciones sólo se mencionan las colas y las *Azure Functions*, que son una parte relevante de este trabajo.

2.1.1. Service Bus

Microsoft Azure Service Bus es un agente de mensajes de integración empresarial completamente administrado. *Service Bus* se usa normalmente para desacoplar las aplicaciones y los servicios entre sí, además de ser una plataforma segura y confiable para datos asincrónicos, así como la transferencia de estado. Los datos se transfieren entre distintas aplicaciones y servicios mediante mensajes. Un mensaje está en formato binario, que puede contener solo texto, JSON o XML.

Algunos escenarios de mensajería comunes son:

Mensajería: transferencia de datos de empresa, como ventas o pedidos de compra, diarios o movimientos del inventario.

Desacoplamiento de aplicaciones: mejora de la confiabilidad y escalabilidad de las aplicaciones y los servicios (el cliente y el servicio no necesitan estar conectados al mismo tiempo).

Temas y suscripciones: habilitación de relaciones 1:n entre publicadores y suscriptores.

Sesiones de mensajes: implementación de flujos de trabajo que requieren ordenación en los mensajes o aplazamiento de los mensajes.

En relación a *Service Bus*, es necesario tener en cuenta tres características principales:

Espacios de nombres: Un espacio de nombres es un contenedor con un ámbito para todos los componentes de la mensajería. Varias colas y temas pueden residir en un único espacio de

nombres, y los espacios de nombres suelen servir de contenedores de aplicación.

Colas: Los mensajes se envían y se reciben desde colas. Las colas permiten almacenar mensajes hasta que la aplicación receptora está disponible para recibirlos y procesarlos 2.2. Los

Figura 2.2: Modelo de cola.

mensajes de las colas se ordenan y se les asigna una marca de tiempo a su llegada. Una vez aceptado, el mensaje se conserva de forma segura en un almacenamiento redundante. Los mensajes se entregan en modo de extracción, que entrega los mensajes cuando se solicitan.

***Topic*:** Los mensajes se envían y se reciben desde colas. Las colas permiten almacenar mensajes hasta que la aplicación receptora está disponible para recibirlos y procesarlos 2.3. Los *Topic*

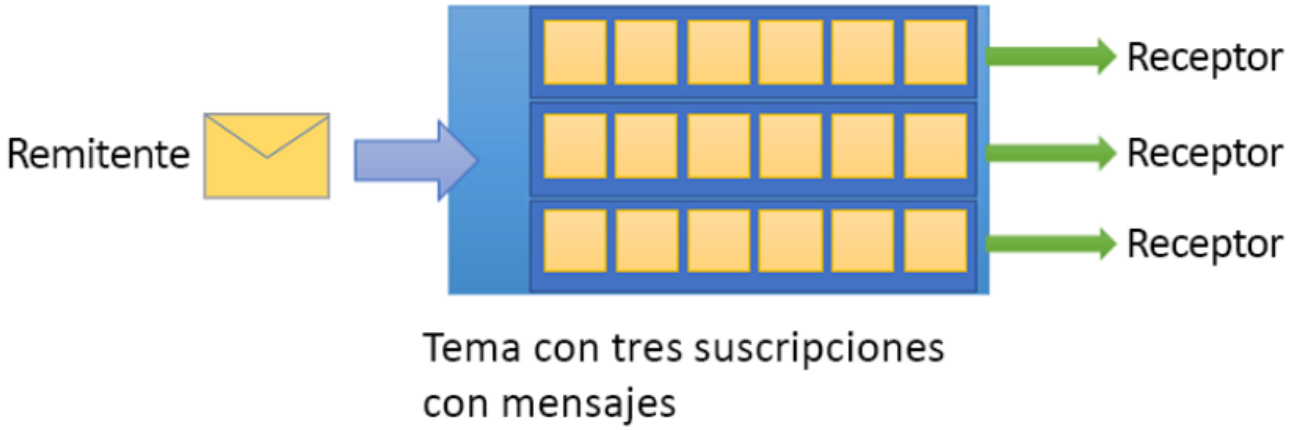

Figura 2.3: Modelo de cola con tema y suscripciones.

pueden tener varias suscripciones independientes. Un suscriptor a un tema puede recibir una copia de cada mensaje enviado a ese tema. Las suscripciones son entidades con nombre, que se crean de forma duradera pero pueden, opcionalmente, expirar o eliminarse automáticamente.

Service Bus también tiene características avanzadas que permiten solucionar problemas de mensajería más complejos. A continuación, se describen estas características principales:

Sesiones de mensajes: Para realizar una garantía primero en entrar/primero en salir (*FIFO*) en *Service Bus*, se usa sesiones. Las sesiones de mensajes permiten la administración ordenada y conjunta de secuencias sin enlace de mensajes relacionados.

Reenvío automático: La característica de reenvío automático permite encadenar una cola o suscripción a otra cola o tema que forme parte del mismo espacio de nombres. Cuando el

reenvío automático está habilitado, *Service Bus* elimina automáticamente los mensajes que se colocan en la primera cola o suscripción (origen) y los coloca en la segunda cola o en el segundo tema (destino).

Colas de mensajes fallidos (DeadLetter): *Service Bus* admite una cola de mensajes fallidos (DLQ) para mantener los mensajes que no se pudieron entregar a ningún destinatario o los mensajes que no se pudieron procesar. A continuación, permite eliminar mensajes de la cola DLQ y examinarlos.

Entrega programada: Se puede enviar mensajes a una cola o un tema para su procesamiento retrasado; por ejemplo, para programar un trabajo de forma que esté disponible para que lo procese el sistema a una hora determinada.

Aplazamiento de mensajes: Cuando un cliente de una cola o una suscripción recibe un mensaje que se desea procesar, pero cuyo procesamiento no es posible en ese momento debido a circunstancias especiales dentro de la aplicación, la entidad tiene la opción de aplazar la recuperación del mensaje para un momento posterior. El mensaje permanece en la cola o suscripción, pero se mantiene separado.

Lotes: El procesamiento por lotes en el lado del cliente permite que un cliente de una cola o un tema retrase el envío de un mensaje durante un período determinado. Si el cliente envía más mensajes durante este período, los transmite en un único lote.

Transacciones: Una transacción agrupa dos o más operaciones en un ámbito de ejecución. *Service Bus* admite operaciones de agrupación en una sola entidad de mensajería (cola, tema, suscripción) dentro del ámbito de una transacción.

Filtrado y acciones: Los suscriptores pueden definir los mensajes que quieren recibir de un tema. Estos mensajes se especifican en forma de una o varias reglas de suscripción con nombre. Con cada condición de regla de coincidencia, la suscripción crea una copia del mensaje, que se puede anotar de manera diferente para cada regla coincidente.

Eliminación automática en estado inactivo: La eliminación automática en estado inactivo permite especificar un intervalo de inactividad después del cual se eliminará automáticamente la cola. La duración mínima es de 5 minutos.

Detección de duplicados: Si se produce un error que hace que el cliente tenga dudas sobre el resultado de una operación de envío, la detección de duplicados saca de dudas en estas situaciones al permitir que el remitente reenvíe el mismo mensaje y la cola o el tema descartan cualquier copia duplicada.

SAS, RBAC e Identidades administradas para recursos de *Azure*: *Service Bus* admite protocolos de seguridad como las firmas de acceso compartido *(SAS)*, el Control de acceso basado en rol *(RBAC)* y Entidades administradas para recursos de *Azure*.

Recuperación ante desastres geográficos: Cuando las regiones o los centros de datos de *Azure* experimentan un tiempo de inactividad, la recuperación ante desastres geográficos permite que el procesamiento de datos siga funcionando en una región o un centro de datos diferentes.

Seguridad: *Service Bus* admite los protocolos estándar *AMQP 1.0* y *HTTP/REST*.

Por otro lado, *Service Bus* es compatible con las bibliotecas de cliente para *.NET*, Java y JMS. Para el desarrollo de este proyecto se utilizarán la biblioteca de *.NET*.[3]

Service Bus no es el único sistema de colas que puede integrarse con *Azure*, también se podría haber usado el sistema *RabbitMQ*. La comparativa entre ambos sistemas y las razones por las cuales se ha decidido implementar *Service Bus* se encuentran en el Anexo C.

2.1.2. Azure Functions

Azure Functions es una solución para ejecutar fácilmente pequeños fragmentos de código, o "funciones", en la nube. Simplemente, permite escribir el código que se necesita para el problema en cuestión, sin preocuparse de toda la aplicación o la infraestructura para ejecutarlo. *Azure Functions* permite utilizar el lenguaje de desarrollo que se prefiera, como *C#, F#, Node.js, Java o PHP.*

La idea detrás de la computación sin servidor, también conocida como *serverless*, es eliminar esas consideraciones de infraestructura para el usuario. Sin servidor, un usuario puede simplemente crear y cargar código, y luego definir los desencadenantes o eventos que ejecutarán el código. Los desencadenantes pueden provenir de una amplia gama de fuentes, incluida la aplicación de otro usuario u otros servicios en la nube, como bases de datos, centros de eventos y notificaciones. En el contexto de este trabajo, los desencadenantes serán los mensajes que se encuentran *Service Bus*.

Una vez que se produce un desencadenante o evento, es responsabilidad del proveedor de la nube cargar el código en un entorno de ejecución adecuado, ejecutar el código y luego liberar los recursos informáticos. Todavía hay servidores involucrados, pero el usuario ya no necesita aprovisionar o administrar instancias de cómputo. Además, en lugar de pagar por esas instancias de cómputo y otros recursos asociados cada mes, los usuarios pagan por la computación sin servidor en función de la cantidad de tiempo que una función se ejecuta en un ciclo de facturación determinado.

En este servicio, sólo se paga el tiempo durante el que se ejecuta el código y, si fuera necesario, *Azure* proporcionaría la infraestructura necesaria para escalarlo fácilmente [4] [5] [6] [7].

Estas son algunas características clave de *Azure Functions*:

Opción de lenguaje: Permite escribir funciones usando el lenguaje *C#, F#* o *Javascript* a elección del desarrollador.

Modelo de precios de pago por uso: El pago derivado va en relación al tiempo, sólo el que se haya empleado ejecutando el código.

Permite exportar otras dependencias: Las *Azure Functions* admiten *NuGet* y *NPM*, que permiten al desarrollador utilizar las bibliotecas que necesite.

Seguridad integrada: Permite proteger las *Functions* desencadenadas por HTTP con los proveedores de *OAuth* como *Azure Active Directory, Facebook, Google, Twitter y cuenta Microsoft.*

Integración simplificada: Permite aprovechar los servicios de *Azure* y ofertas de software como *SaaS.*

Desarrollo flexible: Permite codificar las Functions directamente en el portal o configurarlas mediante la integración continua y permite implementar el código mediante *GitHub, Azure DevOps Services* y otras herramientas de desarrollo compatibles.

Código abierto : *Azure Functions* es de código abierto y está disponible en *GitHub*.

Azure Functions ofrecen plantillas para comenzar con situaciones clave, incluidas las siguientes:

CosmosDBTrigger: Procesa documentos de *Azure Cosmos DB* cuando se agregan o se actualizan en las colecciones en una base de datos *NoSQL*.

QueueTrigger : Responde a mensajes conforme llegan a una cola de *Azure Storage*.

ServiceBusQueueTrigger: Permite conectar el código a otros servicios de *Azure* o servicios locales, mediante la escucha de las colas de mensajes.

ServiceBusTopicTrigger: Permite conectar el código a otros servicios de *Azure* o a servicios locales mediante la suscripción a temas.

2.2. Microsoft Dynamics 365

Un software *CRM (Customer Relationship Management)* es una solución que permite centrar la estrategia de una empresa entorno al cliente. Es una herramienta que permite identificar clientes potenciales, y ofrecer servicios personalizados a los actuales clientes, con el objetivo de fidelizar y ser más efectivos a la hora de interactuar con estos, como se ve en la imagen 2.4.

Figura 2.4: Funcionalidades de Microsoft Dynamics 365.

La decisión de implantar una solución de *Customer Relationship Management (CRM)* es un paso estratégico hacia la focalización de toda la actividad de la empresa en las necesidades del cliente. Las compañías están cambiando el modelo de negocio porque su futuro depende de:

- Su capacidad de analizar las preferencias de los clientes
- La planificación de la producción en función de la demanda del mercado
- Sus posibilidades de comercializar sus productos o servicios a los clientes adecuados y de la forma más eficiente

Microsoft Dynamicss 365 for Sales es el nuevo nombre del anterior *Microsoft Dynamics CRM*. Se suelen utilizar las dos nomenclaturas para referirse al mismo software. *Dynamicss 365 for Sales* proporciona al personal de ventas las herramientas necesarias para conseguir relaciones duraderas con los clientes, actuar en consecuencia según los detalles que tengan y cerrar ventas aún más rápido. *Dynamicss 365 for Sales* se utiliza para realizar el seguimiento de las cuentas y contactos de los usuarios, consolidar una posible venta con un cliente potencial y convertirla en una venta real, crear ventas adicionales, crear listas de marketing y campañas e incluso seguir casos del servicio asociados a cuentas y oportunidades específicas.[8]

2.3. Oracle Database

Oracle es básicamente una herramienta cliente/servidor para la gestión de base de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales, por norma general.

En el desarrollo de paginas Web pasa lo mismo, como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, *Access, MySQL, SQL Server* etc. [9]

Es el mayor y más usado Sistema Manejador de Base de Datos Relacional (RDBMS) en el mundo. La Corporación Oracle ofrece este RDBMS como un producto incorporado a la línea de producción. Además incluye cuatro generaciones de desarrollo de aplicación, herramientas de reportes y utilitarios.

Oracle corre en computadoras personales (PC), microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo. Soporta unos 17 idiomas, corre automáticamente en más de 80 arquitecturas de hardware y software distintos sin tener la necesidad de cambiar una sola línea de código. Esto es porque más del 80 % de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos.

El poderoso modelo relacional ha evolucionado desde herramientas y los modelos de datos de redes. La mayor aceptación y uso de un modelo de datos es el modelo relacional que fue conocido en 1969 con la revisión hecha por IBM, Dr. E. F. Codd.

Este modelo relacional posee tres grandes aspectos:

Estructuras: Definición de objetos que contengan datos y que son accesibles a los usuarios.

Operaciones: Definir acciones que manipulen datos u objetos.

Reglas: Leyes para gobernar la información, cómo y qué manipular.

Una base de datos relacional es definida como un modelo de información que se puede visualizar, estrictamente, por los usuarios mediante tablas. Una tabla está compuesta por una matriz bidimensional de filas y columnas. La información almacenada, dentro de dichas tablas, puede ser modificada mediante el uso de consultas realizadas por el usuario en el formato de filas/columnas[10].

2.4. Lenguajes de programación

Para el desarrollo de este proyecto, se ha utilizado el lenguaje de programación *C#*. Este lenguaje, es propiedad de *Microsoft* como parte de su plataforma *.NET* que después fue aprobado como un estándar por la *ECMA (ECMA-334) e ISO (ISO/IEC 23270). C#* es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de *C/C++* y utiliza el modelo de objetos de la plataforma *.NET*, similar al de *Java*, aunque incluye mejoras derivadas de otros lenguajes.

A pesar de que las *Azure Functions* soportan varios lenguajes, incluidos *JAVA* y *PHYTON*, se decidió utilizar este lenguaje, aun siendo necesario aprenderlo para el desarrollo de la aplicación, debido a que es el lenguaje que pertenece a *Microsoft* y, por tanto, es el que mejor integración tiene con todos sus sistemas y herramientas.

2.5. Herramientas

Para el desarrollo del proyecto se han utilizado un conjunto de herramientas. Para el desarrollo y depuración del proyecto se ha utilizado un ordenador portátil Toshiba Satellite P50 con procesador i7-4720HQ y 8 Gb de Ram.

Se ha utilizado el sistema operativo W10 puesto que se está desarrollando un proyecto que utiliza la tecnología *Microsoft* y, como entorno de desarrollo, *Visual Studio 2017* enlazado con repositorio *TFS* que permite un control de versiones y conexión con *Azure* integrada.

Para permitir la conexión con Oracle, se ha instalado una máquina virtual sobre *Virtual Box* sobre la cual está instalado un servidor de bases de datos. La razón por la cual se ha decidido utilizar un máquina virtual, es debido a la facilidad de administración y mantenimiento de la misma. Resulta sencillo recuperar un servidor alojado en una maquina virtual si se produce algún fallo durante el desarrollo y, por cuestiones de tiempo, limitaciones de presupuesto y requisitos del proyecto, no se podía crear y administrar un servidor completo alojado en la nube o gestionar uno en local que permitiera el acceso desde una aplicación alojada en la nube.

Capítulo 3

Análisis

En este capítulo se detallarán las cuestiones metodológicas, es decir, las metodologías y herramientas que se han utilizado para plantear el trabajo.

También quedarán definidos los requisitos de todos los elementos que conforman el sistema, que se acuerden con el cliente de este proyecto, en el contexto de este trabajo son los tutores.

3.1. Proceso de desarrollo

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada [2].

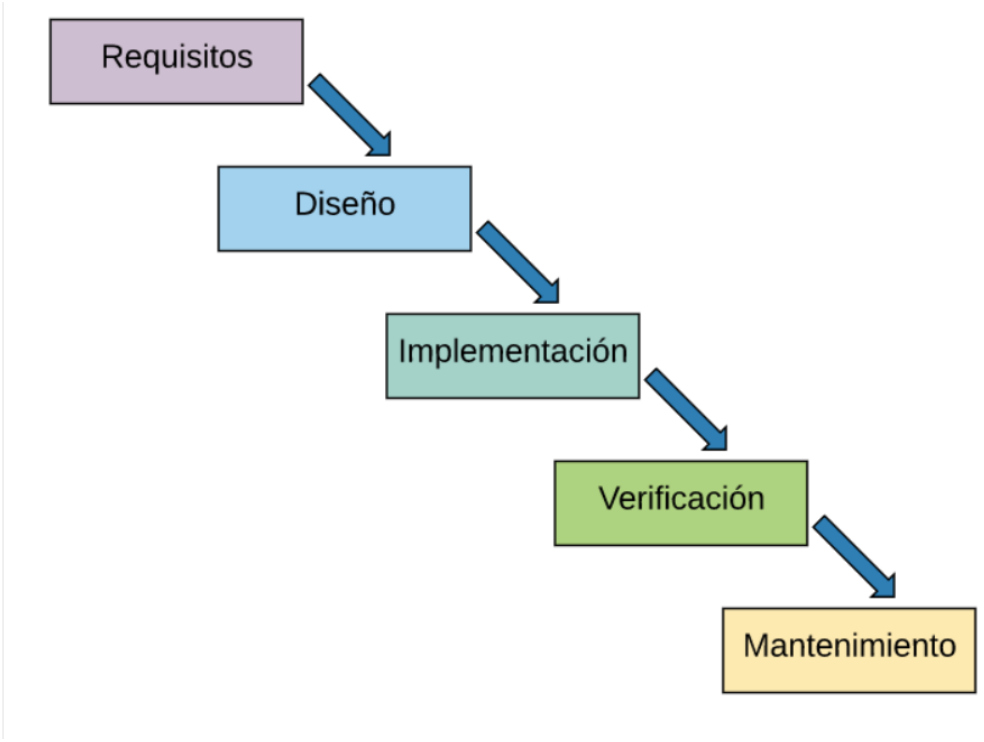

Figura 3.1: Las etapas del modelo en cascada.

Las ventajas asociadas a este modelo, que han llevado a la elección del mismo, son las siguientes:

- El tiempo que se invierte en diseñar el producto en las primeras fases del proceso puede evitar problemas que son más costosos cuando el proyecto ya está en fase de desarrollo. Al ser un sistema nuevo, se procura desde el primer momento tener muy definido el marco de desarrollo sobre el cual se trabajará.
- La documentación es muy exhaustiva y si se une al equipo un nuevo desarrollador, puede comprender el proyecto leyendo la documentación. Lo cual es un beneficio añadido al desarrollo del sistema y por ende, de la presente memoria.
- Al ser un proyecto muy estructurado, con fases bien definidas, es fácil entender el proyecto. Que, en el contexto de este trabajo se dan estas condiciones gracias al hecho de ser un modelo que ya se ha estudiado y realizado, previamente, en la escuela.
- Ideal para proyectos estables, donde los requisitos son claros y no sufren importantes cambios a lo largo del proceso de desarrollo. Puesto que los requisitos son proporcionados por un cliente informado que tiene los criterios sobre los cuales se desarrolla el siguiente trabajo muy definidos.

Como todo modelo de desarrollo, tiene una serie de desventajas que se describen a continuación y la manera en que se han intentado solventar:

- En muchas ocasiones, los clientes no tienen completamente definidos los requisitos, por tanto necesitarán una revisión en fases siguientes, donde existe una versión del software en funcionamiento. Entonces, serán capaces de proporcionar unos requisitos más adecuados lo cual da lugar a otros nuevos, lo que supone volver a refinar fases ya superadas y provoca un incremento del coste.

  Como se ha mencionado previamente, en el contexto de este proyecto, los clientes tienen muy claros los requisitos que debe cumplir el presente trabajo.

- No se va mostrando al cliente el producto a medida que se va desarrollando, si no que se ve el resultado una vez ha terminado todo el proceso. Esto puede provocar inseguridad al cliente que puede desear ver avances tangibles en el producto.

  En el caso de este proyecto, no existe este inconveniente por parte del cliente al ser un proyecto guiado.

- Los diseñadores pueden no tener en cuenta todas las dificultades que se encontrarán cuando estén diseñando el software, lo que conlleva rediseñar y refinar el proyecto para solventar este tipo de problemas.

  Puede suponer un riesgo calculado, que debe de ser asumido y se contemplará en la fase de análisis de riesgos.

- Para proyectos a largo plazo, este modelo puede ser menos flexible, en comparación con otros modelos o metodologías, al cambiar las necesidades del usuario a lo largo del tiempo. Si por ejemplo, tenemos un proyecto que va a durar 5 años, es muy probable que los requisitos necesiten adaptarse a los gustos y novedades del mercado.

  El presente proyecto esta acotado en base a una duración de 300 horas, lo cual evita el problema mencionado.

3.2. Roles y Responsabilidades

Este proyecto ha sido realizado por la alumna Alba Francisco Gutiérrez, por tanto, debe encargarse de las tareas asociadas a los siguientes roles.

Rol	**Responsabilidades**	**Persona encargada**
Jefe de proyecto	Encargado de gestionar proyectos de creación de programas. Entre las responsabilidades de este puesto, está la gestión de equipos . Es el máximo responsable de que éste se ejecute en los plazos establecidos con el cliente final, según los estándares de calidad definidos, y dentro de un espectro de costes determinado.	Alba Francisco Gutiérrez
Analista	Analiza e identifica los requisitos y, en función a ellos, desarrolla la solución más adecuada. Por otro lado, también debe de analizar el entorno donde se implementara la solución y elegir las herramientas más adecuadas a menos que dichas herramientas ya sean un requisito en sí mismo.	Alba Francisco Gutiérrez
Diseñador	Diseña los componentes que conforman el sistema a desarrollar.	Alba Francisco Gutiérrez
Desarrollador	Implementa los componentes en relación a los diagramas de diseño.	Alba Francisco Gutiérrez
Quality assurance	Testea la solución en busca de comportamientos no esperados.	Alba Francisco Gutiérrez

Tabla 3.1: Roles y responsabilidades

3.3. Plan de proyecto

El tiempo estimado para la realización del proyecto son de 300 horas. Se decidió establecer un horario por el cual se hacían una media de tres horas diarias de lunes a domingo. Como la alumna se autogestiona el tiempo, puede realizar horas extras si las tareas así lo requieren.

Debido a la necesidad de formarme en una tecnología nueva y un paradigma con el cual no había trabajado con anterioridad, se realiza un mes previo de formación que no se incluirá en la planificación del proyecto.

En la siguiente tabla se exponen las fases del proyecto y las iteraciones que se fijaron para cada una de las mismas:

Fase	**Nº de iteraciones**	**Fecha de inicio**	**Fecha de fin**	**Duración**
Inicio	1	01/02/2019	15/02/2019	3 semanas
Elaboración	1	18/02/2019	12/03/2019	4 semanas
Construcción	2	12/03/2019	02/05/2019	7 semanas
Transición	1	03/05/2019	10/05/2019	2 semanas

Tabla 3.2: Fases del proyecto

Y a continuación, se explicará más en detalle en que consiste cada una de las iteraciones mencionadas:

Inicio

- Se estudia el problema y se plantean soluciones para el mismo.
- En esta fase se definen los roles, los recursos que serán indispensables para la resolución del problema además de los riesgos asociados.
- Se estima los costes derivados del desarrollo.
- Se comienza a planificar el proyecto.

Elaboración

- Se analizan los participantes del proyecto y los roles que tendrán dentro del desarrollo del mismo.
- Se definen los requisitos y el alcance que tendrá el proyecto además de la arquitectura y las herramientas que se utilizarán.
- Se identifican los casos de uso y se genera el modelos de domino.
- Se inicia la planificación de la primera etapa de la fase de construcción.

Construcción

- Se generan los diagramas de clases y de secuencia los cuales se seguirán en la fase de implementación.
- Se revisan para asegurar que se ajustan a los requisitos establecidos.
- Se crean los manuales de instalación.
- Se planifica la fase de transición.

Transición

- Se corrigen los errores encontrados tras la aplicación de la batería de pruebas.
- Se revisa la documentación.
- Se graba un DVD con todos los archivos que se deben entregar.

3.3.1. Recursos del proyecto

En la siguiente tabla, se agrupan los recursos necesarios para el desarrollo del proyecto (se debe de tener en cuenta que la mayoría del software utilizado para este proyecto esta alojado en la nube):

Fase	**Recursos**		
	Humanos	Hardware	Software
Inicio	Alba Francisco Gutiérrez	Ordenador de la alumna	Overleaf LaTeX editor Microsoft Project 365 Microsoft Planner 365
Elaboración	Alba Francisco Gutiérrez	Ordenador de la alumna	Overleaf LaTeX editor StarUML Edraw Max
Construcción	Alba Francisco Gutiérrez	Ordenador de la alumna	Overleaf LaTeX editor Virtual Box Oracle Database Virtual Box Appliance / Virtual Machine *Azure* *Microsoft Dynamics 365* Visual Studio 2017 C#
Transcción	Alba Francisco Gutiérrez	Ordenador de la alumna	Overleaf LaTeX editor Virtual Box Oracle Database Virtual Box Appliance / Virtual Machine *Azure* *Microsoft Dynamics 365* Visual Studio 2017 C#

Tabla 3.3: Recursos del proyecto

Como la mayoría del software utilizado para este proyecto son aplicaciones alojadas en la nube, las especificaciones de las misma no se pueden incluir aunque, en la siguiente tabla, se incluirán las especificaciones del ordenador utilizado para desarrollar y depurar el código. Además, en él está alojada la máquina virtual donde se encuentra Oracle Server.

Ordenador utilizado para el desarrollo	
Modelo	Toshiba Satellite P50-B-11L
Procesador	Intel Core i7-4720HQ
Tarjeta gráfica	AMD Radeon R9 M265X(2GB de memoria dedicada DDR5)
Memoria RAM	8 Gb
Almacenamiento	SSHD 1TB
Sistema operativo	Windows 10 Home

Tabla 3.4: Características del ordenador utilizado.

3.4. Objetivos del proyecto

Antes de enumerar los objetivos que deben cumplir con este proyecto, se agrupa en las siguientes tablas los niveles de los mismos para su mejor compresión:

Niveles de importancia	
Vital	Es indispensable cumplir este objetivo para que el sistema sea funcional y se ajuste a las necesidades del cliente.
Alta	Se necesita este objetivo para ajustarse a la necesidades del cliente.
Baja	Es un objetivo que no es indispensable cumplir para ajustarse a las necesidades del cliente.

Tabla 3.5: Niveles de importancia

Niveles de importancia	
Alta	Se debe cumplir lo antes posible.
Baja	Se puede dejar con vistas a futuro.

Tabla 3.6: Niveles de urgencia

A continuación, los objetivos que se pretender alcanzar con el desarrollo del proyecto:

Objetivo 001	
Nombre	Persistencia de registros creados
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema tendrá que ser capaz de mantener la persistencia de los datos creados en *Microsoft Dynamics 365* en una entidad, por un usuario en una base de datos de Oracle, de manera automatizada y distribuida, mediante el uso de *Azure Functions*.
Importancia	Vital
Urgencia	Alta

Tabla 3.7: Objetivo 001

Objetivo 002	
Nombre	Persistencia de registros actualizados
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema tendrá que ser capaz de mantener la persistencia de los datos que han sido actualizados en una entidad en *Microsoft Dynamics 365*, por un usuario en una base de datos de Oracle, de manera automatizada y distribuida, mediante el uso de *Azure Functions*.
Importancia	Vital
Urgencia	Alta

Tabla 3.8: Objetivo 002

Objetivo 003	
Nombre	Persistencia de registros eliminados
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema tendrá que ser capaz de mantener la persistencia de los datos eliminados, en una entidad en *Microsoft Dynamics 365*, por un usuario en una base de datos de Oracle, de manera automatizada y distribuida, mediante el uso de *Azure Functions*. Esta *Function* también será la encargada de eliminar los registros consumidos de la entidad Papelera, después de ser encolados dentro de *Service Bus*.
Importancia	Vital
Urgencia	Alta

Tabla 3.9: Objetivo 003

Objetivo 004	
Nombre	Cola de reintentos manual
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema contará con una cola de reintentos manual. En esta cola quedarán almacenados los mensajes que no se han podidos tratar en Oracle, en relación a errores dentro del mensaje, como pueden ser incongruencias en los datos que se intentan manejar dentro de la base de datos. Estos mensajes deberán ser tratados de manera manual por el equipo de gestión. Para facilitar dicho tratamiento, estos mensajes se insertaran en forma de registro a *Microsoft Dynamics 365*.
Importancia	Vital
Urgencia	Alta

Tabla 3.10: Objetivo 004

Objetivo 005	
Nombre	Envío de email al equipo de desarrollo
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema deberá mandar un e-mail al equipo de desarrollo, en el momento que se detecte un mensaje dentro de las colas de *DeadLetter* asociadas a los *Topic*
Importancia	Media
Urgencia	Media

Tabla 3.11: Objetivo 005

Objetivo 006	
Nombre	Envío de e-mail al equipo de gestión
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema deberá mandar un e-mail al equipo de gestión, en el momento que se detecte que se han superado el valor de límite de mensajes. En este caso este valor es de 10 registro en *Microsoft Dynamics 365*, creados por la cola manual.
Importancia	Baja
Urgencia	Media

Tabla 3.12: Objetivo 006

Objetivo 007	
Nombre	Regularización de registros.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Se debe de tener un programa de consola que, en el caso de que las *Azure Functions* que recolectan los registros del CRM, fallen por un intervalo de tiempo determinado, regularicen los registros enviando dichos registros no consumidos, a la cola para su posterior tratamiento en Oracle.
Importancia	Baja
Urgencia	Baja

Tabla 3.13: Objetivo 007

Objetivo 008	
Nombre	Implementación de cola de reintentos automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema contará con una cola de reintentos automática. En esta cola quedarán almacenados los mensajes que no se hayan podidos tratar en Oracle, por un fallo de conexión con el mismo. En el momento que la *Function*, asociada a esta cola, detecte que Oracle está de nuevo en funcionamiento, reencolará los mensajes a su suscripción y *Topic* asociado.
Importancia	Vital
Urgencia	Alta

Tabla 3.14: Objetivo 008

Objetivo 009	
Nombre	Plugin para mantener un histórico de registros eliminados.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Dentro del sistema de *Microsoft Dynamics 365*, habrá instalado un Plugin que, cuando se realice la eliminación de un registro, se encargue de almacenar los datos que serán necesarios para su posterior eliminación en Oracle, dentro de una entidad intermedia que hará las veces de "papelera".
Importancia	Alta
Urgencia	Alta

Tabla 3.15: Objetivo 009

3.5. Presupuesto económico

Para la estimación del presupuesto se tendrán en cuenta las herramientas utilizadas (hardware y software, con los factores de impacto que correspondan a la duración del proyecto), junto con los recursos humanos necesarios, según la planificación de tareas, y el tipo de rol (analista, programador, etc) correspondiente a cada tarea. Para ello conviene tener una tabla actualizada, del coste por hora de cada tarea del proyecto, además del estudio previo de la planificación de tareas.

La estimación de los gastos generados se dividirán en tres partes: Hardware, Software y RRHH para facilitar la visualización de los mismos.

Hardware

Al ser un proyecto enfocado en la nube, el hardware necesario como tal es limitado. En el siguiente presupuesto se incluirá el ordenador portátil que se utilizado para el desarrollo del trabajo 3.16.

Coste de recursos de hardware			
Hardware	**Coste unitario(€)**	**Cantidad (unidades)**	**Coste(€)**
Toshiba Satellite P50-B-11L	797,27€	1	797,27€
Total			**797,27€**

Tabla 3.16: Coste de recursos de hardware

Software

Para el desarrollo de este presupuesto se ha utilizado la herramienta *Azure* Pricing [11]

Tipo de Servicio	**Región**	**Descripción**	**Coste estimado**
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 44.000 ejecuciones/mes	0.00 €
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 44.000 ejecuciones/mes	0.00€
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 44.000 ejecuciones/mes	0.00€
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 1.000 ejecuciones/mes	0.00€
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 3.000.000 ejecuciones/mes	174.00€
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 3.000.000 ejecuciones/mes	174.00€
Azure Function	West US	128 MB de memoria, 30 segundos de tiempo de ejecución y 3.000.000 ejecuciones/mes	174.00€
Service Bus	East US	Nivel Basic: <1 millón de operaciones de mensajería/mes	0.00€
Support		**Support**	**0.00€**
		Licensing Program	**MOSP**
		Total Mensual	**401.60€**
		Total para 4 meses	**1.606,4€**

Tabla 3.17: Presupuesto de servicios de *Azure*

Se debe de tener en cuenta que Oracle proporciona una licencia gratuita para desarrolladores, siempre y cuando, el sistema generado no se use con fines comerciales [3]. Como el presente trabajo

Aplicación	Aplicaciones disponibles en el plan	Precio/Usuario
Dynamicss 365 Plan	▪ Finance and Operations ▪ Retail ▪ Talent ▪ Customer Service ▪ Project Service Automation ▪ Field Service ▪ Marketing ▪ Solución Microsoft Relationship Sales ▪ PowerApps para Dynamicss 365 Flow	177.10 €
Total (4 meses / un usuario)		**2.833,62 €**

no abarca la implementación de un sistema en producción al uso, no se incluirá gastos por usar el sistema de bases de datos de Oracle.

RRHH

Nombre del recurso	Coste unitario / hora(€)	Cantidad (Horas)	Coste(€)
Jefe de proyecto	55,9 €	85	4751,5 €
Diseñador	53,9 €	38	2048,2 €
Desarrollador	31,2 €	109	3400,8 €
Tester	31,2 €	18	561,6 €
Analista	38,9 €	70	2723 €
Total			**13.485,1 €**

Tabla 3.18: Coste de recursos humanos

Horas extra

En base a los precios aportados anteriormente, y en base a estas condiciones:

- Las aplicaciones utilizadas para el desarrollo de este proyecto tienen suscripción mensual. Por ello, el presupuesto se ha realizado en base a la duración del TFG, que son 300 horas. Redondeando esta cifra en base a la planificación, son cuatro meses de desarrollo.
- Se produjo un retraso de quince horas en la fase de desarrollo. Este retraso no impidió completar el trabajo en las fechas previstas, pero aumento el presupuesto en consecuencia 3.19.

Coste extra de la tarea de desarrollo.			
Rol	**Coste unitario/Hora**	**Aumento precio/ horas extra**	**Coste total €**
Desarrollador	31,2 €	25 %	585 €

Tabla 3.19: Coste de las horas extra realizadas.

3.5.1. Costes totales

Finalmente, la estimación del presupuesto queda de la siguiente forma. Este presupuesto no representa, de ninguna forma, todos los gastos que podrían generarse en un proyecto real pero sirve de aproximación:

Coste total del proyecto	
Humanos	14070,1 €
Hardware	797,27 €
Software	4440,02€
Total	**19307,39€**

Tabla 3.20: Coste total del proyecto

3.6. Participantes del proyecto

En la siguientes tablas queda reunida la información acerca de los participantes que han formado parte del proyecto.

Participante 001	
Nombre	Alba Francisco Gutiérrez
Organización	Estudiante de Grado en Ingeniería Informática de la UVa
Rol	Jefa de proyecto, analista, diseñadora, desarrolladora y Tester
Es desarrollador	Sí
Es cliente	No
Es usuario	No

Tabla 3.21: Participante 001

Participante 002	
Nombre	Benjamín Sahelices Fernández
Organización	Departamento de informática de la UVa Directo de la Escuela de Ingeniería Informática de la UVa
Rol	Tutor de TFG
Es desarrollador	No
Es cliente	Sí
Es usuario	Sí

Tabla 3.22: Participante 002

Participante 003	
Nombre	Alberto Casero de la Calle
Organización	Everis and NTT Data Company
Rol	Tutor de TFG
Es desarrollador	No
Es cliente	Sí
Es usuario	Sí

Tabla 3.23: Participante 003

Participante 004	
Nombre	Álvaro Estébanez Barrena
Organización	Everis and NTT Data Company
Rol	Tutor de TFG
Es desarrollador	No
Es cliente	Sí
Es usuario	Sí

Tabla 3.24: Participante 004

3.7. Análisis de riesgos

En la siguiente sección se detallará el análisis de riegos realizado. El análisis del riesgo es un método sistemático de recopilación, evaluación, registro y difusión de información necesaria para formular recomendaciones orientadas a la adopción de una posición o medidas en respuesta a un peligro determinado. Además, de permitir monitorizar el transcurso de un proyecto para evaluar el estado de los riesgos y actuar en consecuencia.

BAJA TEMPORAL		
Número: 001	Fecha: 25/02/2019	Categoría: Predecible
Probabilidad: Alta	Tiempo: Cualquiera	Consecuencia: Modificación en la planificación
Proyecto: TFG		Iteración: Cualquier iteración
Creador:Alba Francisco		
Descripción		
El único miembro puede sufrir una baja temporal como consecuencia de enfermedades de corta o media duración.		
Plan de contingencia		
Estrategia:	Tener tiempos lo suficientemente flexibles para evitar que tener un tiempo parado no afecte a las fechas de entrega.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: El empleado afectado debe comunicar a sus tutores de proyecto su estado sintomático	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.25: Baja temporal

EL CLIENTE NO ES ESPECÍFICO CON LOS REQUISITOS DEL PROYECTO		
Número: 005	Fecha: 25/02/2019	Categoría: Alta
Probabilidad: Alta	Tiempo: Cualquier momento	Consecuencia: Retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
El cliente no tiene definidos los requisitos que quiere que cumplimente la aplicación a desarrollar.		
Plan de contingencia		
Estrategia:	Reiterar las reuniones con el clientes hasta tener un requisitos mínimos a cumplimentar definidos.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: El cliente puede dar síntomas de no tener definido unos propósitos concretos a los que quiere llegar.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.26: El cliente no es específico con los requisitos del proyecto

FALLO DE HARDWARE		
Número: 005	Fecha: 25/02/2019	Categoría: Alta
Probabilidad: Baja	Tiempo: Cualquier momento	Consecuencia: Retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Un fallo de hardware en el ordenador de la alumna.		
Plan de contingencia		
Estrategia:	En caso extremo, se podría recurrir al ordenador que proporciona la empresa Everis, para la finalización del proyecto.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: El ordenador puede evidenciar fallas con anterioridad.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.27: Fallo de hardware

FALLO DE SOFTWARE		
Número: 005	Fecha: 25/02/2019	Categoría: Alta
Probabilidad: Baja	Tiempo: Cualquier momento	Consecuencia: Retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Un fallo de software en el ordenador de la alumna.		
Plan de contingencia		
Estrategia:	Utilizar herramientas que la alumna haya probado con anterioridad y conocer herramientas similares que puedan sustituir a las que se ven afectadas	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: El ordenador puede evidenciar fallas con anterioridad.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.28: Fallo de software

PÉRDIDA DE INFORMACIÓN		
Número: 005	Fecha: 25/02/2019	Categoría: Alta
Probabilidad:Media	Tiempo: Cualquier momento	Consecuencia: Retrasos y repetición de tareas ya realizadas
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Pérdida de datos bien sea por fallo humano, de hardware o software. Puede perderse avances en la programación, documentación o estado de la máquina virtual donde se aloja Oracle Database.		
Plan de contingencia		
Estrategia:	Realizar copias de seguridad periódicas de la documentación en local , ya que se trabaja en la nube, y de la máquina virtual en un sistema externo de almacenamiento. Establecer un sistema de control de versiones, como puede TFS, para mantener los avances del código.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: El ordenador puede evidenciar fallas con anterioridad.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.29: Pérdida de información

INCAPACIDAD PARA AJUSTARSE A LA PLANIFICACIÓN		
Número: 005	Fecha: 25/02/2019	Categoría: Media
Probabilidad: Media	Tiempo: Cualquier momento	Consecuencia: Retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Puede darse el caso de que el equipo sea incapaz de ajustarse a la planificación bien sea por una mala planificación u otras causas.		
Plan de contingencia		
Estrategia:	Ajustar las tareas críticas y rehacer la planificación.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: Se puede dar que debido al desconocimiento la planificación no sea del todo correcta.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.30: Incapacidad para ajustarse a la planificación

FALTA DE COMUNICACIÓN CON LOS TUTORES		
Número: 005	Fecha: 25/02/2019	Categoría: Media
Probabilidad: Media	Tiempo: Cualquier momento	Consecuencia: Desarrollo de un proyecto incompleto o una documentación incorrecta
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Falta de comunicación con los tutores para resolver dudas y ponerse al día con los progresos del proyecto.		
Plan de contingencia		
Estrategia:	Ajustar las tareas críticas y rehacer la planificación.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios:Se establece un sistema de comunicación basado en reuniones y mensajes de correo.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.31: Falta de comunicación con los tutores

DISEÑO EQUIVOCO		
Número: 006	Fecha: 25/02/2019	Categoría: impredecible
Probabilidad: media	Tiempo: Fase inicial	Consecuencia: Rehacer trabajo ya realizado
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Pueden darse fallos ya bien sea por despiste, desconocimiento o bien por concepto equivocado.		
Plan de contingencia		
Estrategia:	Identificar los errores y corregirlos lo mas pronto posible.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: Durante la primera fase se puede advertir errores por desconocimiento de la tecnología u cualquier otra razón con el carácter de los implicados.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.32: Diseño equivoco

DESASTRE NATURAL		
Número: 007	Fecha: 25/02/2019	Categoría: impredecible
Probabilidad: muy baja	Tiempo: Cualquier fase	Consecuencia: Grandes retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Pueden producirse desastres que afecten al servidor donde esta almacenado el proyecto.		
Plan de contingencia		
Estrategia:	Recuperar las copias de seguridad y restaurar	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: Se suele informar a la población del mismo.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.33: Desastre natural

CORTE DE LUZ		
Número: 008	Fecha: 25/02/2019	Categoría: impredecible
Probabilidad: muy baja	Tiempo: Fase inicial	Consecuencia: Grandes retrasos
Proyecto: TFG		Iteración: Cualquier iteración
Creador: Alba Francisco		
Descripción		
Pueden producirse un fallo que afecten a las máquinas en las cuales se está desarrollando el proyecto		
Plan de contingencia		
Estrategia:	Realizar copias de seguridad en periodos de tiempos razonables para evitar perdidas de información.	
Indicador de riesgo		
Indicador: Síntomas previos	Comentarios: No se puede predecir.	
Resolución del riesgo		
Responsable:		Jefe de proyecto:
Fecha:		Fecha:

Tabla 3.34: Corte de luz

AUSENCIA DE LICENCIAS		
Número:009	Fecha: 25/02/2019	Categoría: Alta
Probabilidad:Alta	Tiempo: Cualquier fase	Consecuencia: Problemas para llevar a cabo el proyecto
Proyecto: TFG		Iteración: Cualquier iteración.
Creador: Alba Francisco Gutiérrez		
Descripción		
Al ser un trabajo de fin de grado con un presupuesto limitado, la necesidad de utilizar opciones de carácter privativo, cuyo gasto no es posible afrontar, puede dar lugar a la necesidad de simular determinadas partes del proyecto		
Plan de contingencia		
Estrategia:	Tener opciones a partir de las cuales poder desarrollar el proyecto a pesar de la carencia de licencias ya se mediante el uso de Trial gratuitas o de simuladores.	
Indicadores de riesgo		
Indicador: Síntomas previos	Comentarios: Al realizar la fase de análisis se realiza un presupuesto de los gastos que generará el desarrollo del proyecto.	
Resolución del riesgo		
Responsable:		Jefe de proyecto
Fecha:		**Fecha:**

Tabla 3.35: Ausencia de licencias

DESARROLLO DE UNA APLICACIÓN QUE NO CUMPLA CON LOS REQUISITOS DE RENDIMIENTO		
Número:009	Fecha: 25/02/2019	Categoría: Alta
Probabilidad:Media	Tiempo: Cualquier fase	Consecuencia: Obtener una aplicación que no cumple con los requisitos de rendimiento
Proyecto: TFG		Iteración: Fase de transacción.
Creador: Alba Francisco Gutiérrez		
Descripción		
Se puede desarrollar una aplicación la cual no cumpla los requisitos de rendimiento y no funcionar correctamente en la fase de transición.		
Plan de contingencia		
Estrategia:	Tener comunicación constante con los tutores acerca de cuales son los mínimos requisitos de rendimiento	
Indicadores de riesgo		
Indicador: Síntomas previos	Comentarios: Al realizar las pruebas puede que la aplicación no responda tal y como se espera de ella	
Resolución del riesgo		
Responsable:		Jefe de proyecto
Fecha:		**Fecha:**

Tabla 3.36: Aplicación que no cumple con los requisitos de rendimiento

INESTABILIDAD DE LA RED		
Número:009	Fecha: 25/02/2019	Categoría: Alta
Probabilidad:Media	Tiempo: Fase de desarrollo	Consecuencia: A causa de una red inestable se puede provocar falsos negativos de funcionamiento.
Proyecto: TFG		Iteración: Fase de desarrollo.
Creador: Alba Francisco Gutiérrez		
Descripción		
Debido a la inestabilidad de la red, se puede considerar como fallido un flujo de procesamiento aunque solo este tardando en arrojar resultados válidos.		
Plan de contingencia		
Estrategia:	Modificar la red.	
Indicadores de riesgo		
Indicador: Síntomas previos	Comentarios: La red puede haber dando síntomas de inestabilidad anteriormente.	
Resolución del riesgo		
Responsable:		Jefe de proyecto
Fecha:		**Fecha:**

Tabla 3.37: Inestabilidad de la red.

3.8. Planificación de fases

Puesto que el desarrollo de este proyecto es realizado por una sola persona, todos los roles y tareas asociadas serán llevados a cabo por la alumna Alba Francisco Gutiérrez, de manera secuencial. En un proyecto en el cual estuvieran implicadas varias personas, estas tareas pueden ser susceptibles de paralelizarse. Los roles como recursos de planificación 3.38.

Nombre del recurso	Tipo	Grupo	Capacidad Máxima	Trabajo restante
Jefe de proyecto	Trabajo	Alba Francisco	100 %	85 horas
Diseñador	Trabajo	Alba Francisco	100 %	38 horas
Desarrollador	Trabajo	Alba Francisco	100 %	109 horas
Tester	Trabajo	Alba Francisco	100 %	18 horas
Analista	Trabajo	Alba Francisco	100 %	70 horas

Tabla 3.38: Recursos de la planificación

3.8.1. Fase de inicio

Fase de inicio	**13 días**	**vie 01/02/19**	**vie 15/02/19**		**Jefe del proyecto**
Iteración 1	**13 días**	**vie 01/02/19**	**vie 15/02/19**		**Jefe del proyecto**
Estudiar el problema a resolver	8 horas	lun 28/01/19	mar 29/01/19		Jefe del proyecto
Elegir un proceso de desarrollo	8 horas	mar 29/01/19	mié 30/01/19	3	Jefe del proyecto
Definir roles y responsabilidades asociadas	5 horas	mié 30/01/19	jue 31/01/19	4	Jefe del proyecto
Definir las fases de las que consta el proyecto	6 horas	jue 31/01/19	vie 01/02/19	5	Jefe del proyecto
Planificar la fase de inicio	4 horas	lun 04/02/19	lun 04/02/19	6	Jefe del proyecto
Creación de las tareas	15 horas	mar 05/02/19	vie 08/02/19	7	Jefe del proyecto
Establecer los recursos de los que se disponen	8 horas	vie 08/02/19	lun 11/02/19	8	Jefe del proyecto
Realizar análisis de riesgos	12 horas	lun 11/02/19	mié 13/02/19	9	Jefe del proyecto
Realizar estimación de los costes	4 horas	jue 14/02/19	jue 14/02/19	10	Jefe del proyecto
Realizar la planificación de la fase de elaboración	4 horas	vie 15/02/19	vie 15/02/19	11	Jefe del proyecto

Figura 3.2: Tareas fase de inicio.

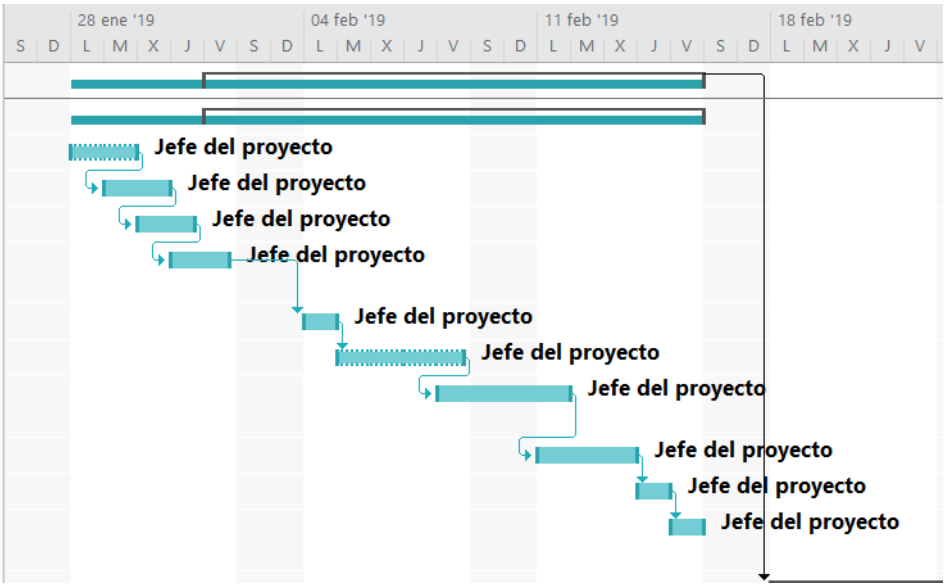

Figura 3.3: Diagrama de Gantt de la fase de inicio.

3.8.2. Fase de elaboración

Fase de Elaboración	68 horas	lun 18/02/19	mar 12/03/19	1	
Iteración 1	68 horas	lun 18/02/19	mar 12/03/19		
Evaluar el equipo que conforma el proyecto	1 hora	lun 18/02/19	lun 18/02/19		
Establecer objetivos del proyecto	8 horas	lun 18/02/19	mar 19/02/19	15	Jefe del proyecto
Recabar requisitos	16 horas	mar 19/02/19	vie 22/02/19	16	Analista
Describir los actores	2 horas	vie 22/02/19	vie 22/02/19	17	Analista
Realizar casos de uso	24 horas	vie 22/02/19	vie 01/03/19	18	Analista
Realizar modelo de dominio	12 horas	vie 01/03/19	mar 05/03/19	19	Diseñador
Establecer arquitectura	16 horas	mar 05/03/19	vie 08/03/19	20	Analista
Evaluar las herramientas a utilizar	4 horas	lun 11/03/19	lun 11/03/19	21	Analista
Planificar iteración 1 de la fase de construcción	4 horas	mar 12/03/19	mar 12/03/19	22	Jefe del proyecto

Figura 3.4: Tareas fase de inicio.

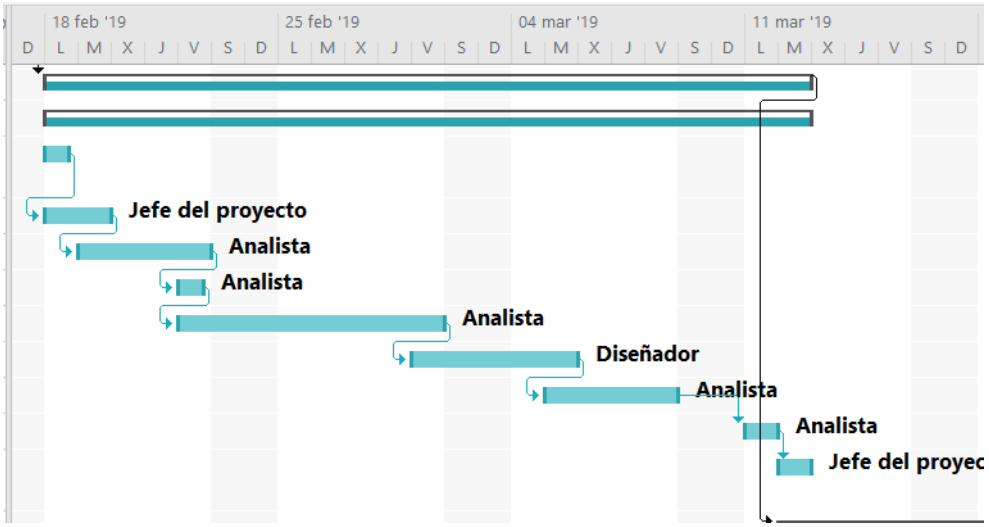

Figura 3.5: Diagrama de Gantt de la fase de elaboración.

3.8.3. Fase de construcción

Fase de construcción	**80 horas**	**mar 12/03/19**	**lun 08/04/19**	**13**	
Iteración 1	**80 horas**	**mar 12/03/19**	**lun 08/04/19**		
Diagramas de clases	10 horas	mié 13/03/19	vie 15/03/19		Diseñador
Diagramas de secuencia	10 horas	vie 15/03/19	mar 19/03/19	26	Diseñador
Implementación de las functions y plugin	50 horas	mié 20/03/19	vie 05/04/19	27	Desarrollador
Planificar iteración 2 de la fase de construcción	8 horas	vie 05/04/19	lun 08/04/19	28	Jefe del proyecto
Iteración 2	**19 días**	**lun 08/04/19**	**jue 02/05/19**	**29**	
Revisar diagramas de clases	5 horas	lun 08/04/19	mar 09/04/19		Diseñador
Revisar diagramas de secuencia	5 horas	mar 09/04/19	mié 10/04/19	31	Diseñador
Implementación de las functions y plugin	50 horas	jue 11/04/19	lun 29/04/19	32	Desarrollador
Manual de instalación y programación	6 horas	lun 29/04/19	mar 30/04/19	33	Desarrollador
Manual de usuario	4 horas	mié 01/05/19	mié 01/05/19	34	Analista
Planificar fase de transicción	4 horas	jue 02/05/19	jue 02/05/19	35	Jefe del proyecto

Figura 3.6: Tareas fase de construcción.

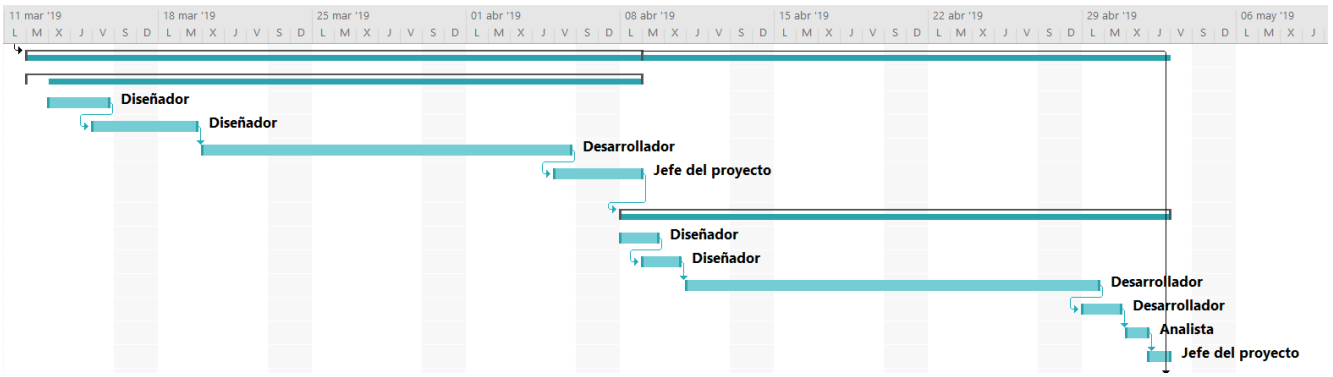

Figura 3.7: Diagrama de Gantt de la fase de construcción.

3.8.4. Fase de transición

Fase de Transición	**24 horas**	**vie 03/05/19**	**vie 10/05/19**	**24**	
Iteración 1	**24 horas**	**vie 03/05/19**	**vie 10/05/19**		
Depuración de errores	4 horas	vie 03/05/19	vie 03/05/19		
Pruebas de caja negra	5 horas	lun 06/05/19	mar 07/05/19	39	Quality assurance
Pruebas de caja blanca	5 horas	mar 07/05/19	mié 08/05/19	40	Quality assurance
Manual de instalación	2 horas	mié 08/05/19	mié 08/05/19	41	Desarrollador
Revisión del documento	8 horas	jue 09/05/19	vie 10/05/19	42	Jefe del proyecto
Revisión de los documentos a anexar en el DVD	2 horas	vie 10/05/19	vie 10/05/19	43	Jefe del proyecto

Figura 3.8: Tareas fase de transición.

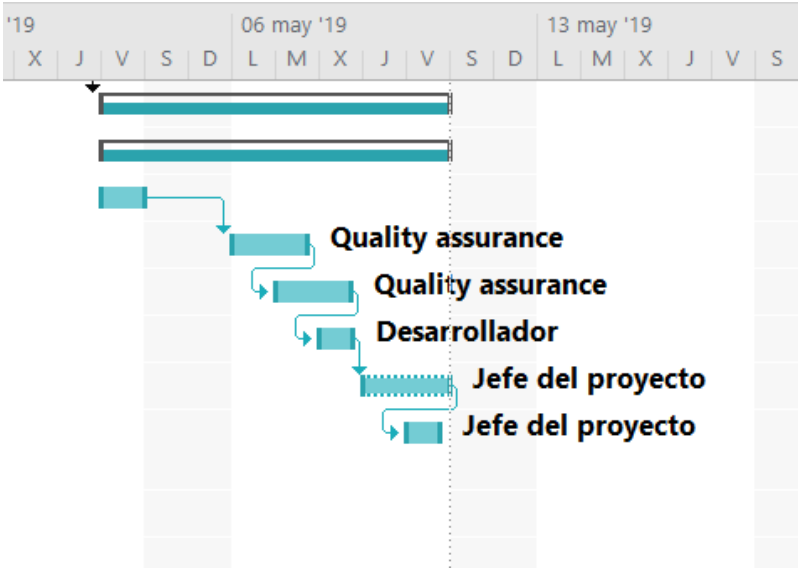

Figura 3.9: Diagrama de Gantt de la fase de transición.

3.9. Requisitos del sistema

Los requisitos quedan separados en dos secciones, funcionales y no funcionales.

3.9.1. Funcionales

El la siguiente sección se agruparan los requisitos funcionales que se ha seguido para la implementación de este trabajo. Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares en base a la necesidades del cliente.

Requisito funcional 001	
Nombre	El sistema evaluará los cambios producidos en los registros de una única entidad.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema evaluará los cambios que se producen en los registros de una única entidad del sistema *Microsoft Dynamics 365*.

Tabla 3.39: Requisito funcional 001

Requisito funcional 002	
Nombre	Comprobación del sistema de la adición de nuevos registros.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema comprobará si hay nuevos registros creados, en un intervalo de tiempo de un minuto, en *Microsoft Dynamics 365*.

Tabla 3.40: Requisito funcional 002

Requisito funcional 003	
Nombre	Sistema comprueba la actualización de registros.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema comprobará si hay nuevos registros actualizados, en un intervalo de tiempo de un minuto, en *Microsoft Dynamics 365*.

Tabla 3.41: Requisito funcional 003

Requisito funcional 004	
Nombre	Comprobación de la eliminación de registros en CRM.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema comprobará si hay nuevos registros eliminados, en un intervalo de un minuto, en *Microsoft Dynamics 365*.

Tabla 3.42: Requisito funcional 004

Requisito funcional 005	
Nombre	Generación de mensajes en relación de los datos extraídos.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema generará mensajes, con un formato definido, usando la información de los registros y los encolará en un *Topic*, asociado a una *Subscription*, en función del tipo de mensaje que se genere.

Tabla 3.43: Requisito funcional 005

Requisito funcional 006	
Nombre	Formato de mensajes
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema generará mensajes con el siguiente formato, que se encolaran en Service Bus: *tipo*: *Create, Update* o *Delete*. *name_ entity:* Nombre de la entidad a partir de la cual se ha extraído el registro. *id*: Identificador único del registro. *key*: Nombre de los campos a partir de los cuales se compone el registro. *value*: Valores almacenado en los campos. *error*: Lugar donde se almacenará, si se produce, el tipo de error. *retryCount*: Número de reintentos.

Tabla 3.44: Requisito funcional 006

Requisito funcional 007	
Nombre	Diferenciación entre registros creados y actualizados
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	La diferenciación entre los registros creados y los registros actualizados se realiza mediante la evaluación de equidad entre los campos *modifiedon* y *createdon*.

Tabla 3.45: Requisito funcional 007

Requisito funcional 008	
Nombre	Cola manual
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Aquellos mensajes que Oracle no pueda gestionar debido a un error, exceptuando aquellos relacionados con la conexión de la base de datos, serán reencolados en una cola manual que será gestionada por el equipo de gestión.

Tabla 3.46: Requisito funcional 008

Requisito funcional 009	
Nombre	Cola automática
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Los mensajes que no puedan ser gestionados por Oracle debido a problemas de conexión serán reencolados en una cola automática.

Tabla 3.47: Requisito funcional 009

Requisito funcional 010	
Nombre	Incremento de *retryCount* en mensajes que pasán a cola automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Los mensajes que pasen a cola automática aumentarán en uno su valor de *retryCount*.

Tabla 3.48: Requisito funcional 010

Requisito funcional 011	
Nombre	Máximo número de reintentos
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Los mensajes que no puedan ser gestionados por Oracle, por cualquier motivo, y cuyo *retryCount* supere el valor de tres, serán reencolados en la cola manual.

Tabla 3.49: Requisito funcional 011

Requisito funcional 012	
Nombre	Creación de registros cuando no existe el registro a actualizar.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si, cuando intentamos actualizar un registro en Oracle, la base de datos informa de que no hay ningún registro con esos datos, se creará el registro.

Tabla 3.50: Requisito funcional 012

Requisito funcional 013	
Nombre	Actualización de los campos que han sido modificados.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Cuando se actualice un registro, se evaluará que campos han sido cambiado en relación a la información que viene en el mensaje y la almacenada en Oracle. En relación a estos, se actualizarán los campos en la base de datos.

Tabla 3.51: Requisito funcional 013

Requisito funcional 014	
Nombre	Antes de crear un registro nuevo se evalúa el retryCount.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si el *retryCount* es superior a cero, antes de crear un registro nuevo se evaluará s este ya ha sido creado previamente. Si ha sido así, el mensaje se desecha.

Tabla 3.52: Requisito funcional 014

Requisito funcional 015	
Nombre	Actualización de registro en relación al campo *MODIFIEDON*
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si el registro almacenado en Oracle tiene un valor en *MODIFIEDON* posterior al valor de ese mismo campo almacenado en el mensaje, entonces el mensaje se desechará. Siempre debe prevalecer el mensaje más reciente.

Tabla 3.53: Requisito funcional 015

Requisito funcional 016	
Nombre	Almacenamiento de los valores más importantes de un registro antes de su eliminación.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	La información más relevante de un registro: nombre de la entidad valor del identificador único nombre del campo donde se almacena dicho valor se almacenará en la entidad *af_ entitydelete* cuando se elimine del *Microsoft Dynamics 365*.

Tabla 3.54: Requisito funcional 016

Requisito funcional 017	
Nombre	Eliminación de registro de af_entitydelete
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Una vez que el mensaje haya sido almacenado en la cola, el registro asociado a dicho mensaje se eliminará de la entidad *af_entitydelete*.

Tabla 3.55: Requisito funcional 017

Requisito funcional 018	
Nombre	Comprobación de disponibilidad de Oracle
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Cada cinco minutos, se evaluará si Oracle está disponible para el tratamiento de mensajes.

Tabla 3.56: Requisito funcional 018

Requisito funcional 019	
Nombre	Condiciones para la no extracción de mensajes de la cola automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si Oracle no está disponible, no se extraerán mensajes de la cola automática.

Tabla 3.57: Requisito funcional 019

Requisito funcional 020	
Nombre	Condiciones para la extracción de mensajes de cola automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si Oracle está disponible, se extraerán mensajes de la cola automática.

Tabla 3.58: Requisito funcional 020

Requisito funcional 021	
Nombre	Reasignación de los mensajes extraídos de cola automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Se evaluará el campo *tipo* del mensaje extraído de la cola automática. En relación al valor almacenado en dicho campo se enviará los mensajes desde la cola automática a los *Topic* adecuados.

Tabla 3.59: Requisito funcional 021

Requisito funcional 022	
Nombre	Mensajes extraídos de cola automática incorrectos.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si el mensaje extraído de la cola automática es incorrecto, se enviará a la cola manual.

Tabla 3.60: Requisito funcional 022

Requisito funcional 023	
Nombre	Sin mensajes dentro de la cola automática.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si no hay mensajes en la cola automática, no se realizará ninguna acción.

Tabla 3.61: Requisito funcional 023

Requisito funcional 024	
Nombre	Evaluación de las colas de *DeadLetter* y registros creados por la cola manual.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Cada quince minutos se evaluarán las colas de *DeadLetter* asociadas a *Create, Delete, Update* y registros creados por la cola manual.

Tabla 3.62: Requisito funcional 024

Requisito funcional 025	
Nombre	E-mail al equipo de desarrollo por mensajes en las cola de *DeadLetter* de *Create*.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si hay un mensaje dentro de la cola de *DeadLetter* del *Topic* y la *Subscription* *"Create"* se enviará un e-mail al equipo de desarrollo.

Tabla 3.63: Requisito funcional 025

Requisito funcional 026	
Nombre	E-mail al equipo de desarrollo por mensajes en las cola de *DeadLetter* de *Update*,
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si hay un mensaje dentro de la cola de *DeadLetter* del *Topic* y la *Subscription* " *Update* " se enviará un e-mail al equipo de desarrollo

Tabla 3.64: Requisito funcional 026

Requisito funcional 027	
Nombre	E-mail al equipo de desarrollo por mensajes en las cola de *DeadLetter* de *Delete*.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Si hay un mensaje dentro de la cola de *DeadLetter* del *Topic* y la *Subscription* " *Delete*" se enviará un e-mail al equipo de desarrollo.

Tabla 3.65: Requisito funcional 027

Requisito funcional 028	
Nombre	Formato de la fecha y hora que utilizará el programa de consola.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El programa de consola deberá utilizar un intervalo de fecha de fecha y hora con el siguiente formato: *dd/mm/yyyy hh:mm:ssss*.

Tabla 3.66: Requisito funcional 028

Requisito funcional 029	
Nombre	Regularización de datos por el programa de consola.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El programa de consola regularizará los registros creados y actualizados dentro del intervalo proporcionado por el usuario.

Tabla 3.67: Requisito funcional 029

Requisito funcional 030	
Nombre	No regularización de registros eliminados.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El programa de consola no extraerá los datos de los registros eliminados.

Tabla 3.68: Requisito funcional 030

Requisito funcional 031	
Nombre	Operaciones del programa de consola.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El programa de consola realizará las mismas operaciones de extracción, codificación del mensaje y envío que el sistema ordinario.

Tabla 3.69: Requisito funcional 031

Requisito funcional 032	
Nombre	*Azure Functions* tipo *TimerTrigger* .
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Las *Azure Functions* que conectan *Microsoft Dynamics 365* con *Service Bus* deben de ser del tipo *TimerTrigger*.

Tabla 3.70: Requisito funcional 032

Requisito funcional 033	
Nombre	Tiempo de *TimerTrigger*.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Las *Azure Functions* tipo *TimerTrigger*, que extraen datos de *Microsoft Dynamics 365*, deben conectarse cada minuto.

Tabla 3.71: Requisito funcional 033

Requisito funcional 034	
Nombre	*Azure Functions* tipo *ServiceBusTrigger*.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Las *Azure Functions* encargadas de extraer datos de *Service Bus* deben de ser del tipo *ServiceBusTrigger*.

Tabla 3.72: Requisito funcional 035

Requisito funcional 036	
Nombre	Creación de registros con la información de los mensajes fallidos almacenados en cola manual.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	La información almacenada en los mensajes fallidos de la cola manual se insertarán, en forma de registro, en *Microsoft Dynamics 365* para facilitar el tratamiento del equipo de gestión.

Tabla 3.73: Requisito funcional 036

Requisito funcional 036	
Nombre	Todo el flujo descrito, se utilizará para mantener la persistencia de datos de una entidad concreta.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	Todo el flujo descrito, se utilizará para mantener la persistencia de datos de una entidad concreta. Cada vez que sea necesario almacenar los datos de otra entidad de *Microsoft Dynamics 365*, se generará un conjunto de *Azure Functions* siguiendo los requerimientos descritos.

Tabla 3.74: Requisito funcional 036

3.9.2. No funcionales

A continuación se especificarán los requisitos no funcionales que definirán las características de funcionamiento. En esta sección quedarán definidas las restricciones o condiciones que impone el cliente al programa que necesita.

Requisito no funcional 001	
Nombre	Tiempo de aprendizaje.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El tiempo de aprendizaje de la utilización del sistema, para los distintos equipos que lo utilicen, sera menor a cuatro horas.

Tabla 3.75: Requisito no funcional 001

Requisito no funcional 002	
Nombre	Tasa de errores.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	La tasa de errores cometidos deberá ser menor del 5 % de las transacciones totales ejecutadas en el sistema.

Tabla 3.76: Requisito no funcional 002

Requisito no funcional 003	
Nombre	Manuales de usuario.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema debe contar con manuales de usuario estructurados adecuadamente.

Tabla 3.77: Requisito no funcional 003

Requisito no funcional 004	
Nombre	Manuales de instalación.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema debe contar con manuales de instalación de usuario estructurados adecuadamente.

Tabla 3.78: Requisito no funcional 004

Requisito no funcional 005	
Nombre	Mensajes de error explicativos.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final.

Tabla 3.79: Requisito no funcional 005

Requisito no funcional 006	
Nombre	Disponibilidad.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema debe tener una disponibilidad del 99,99 % de las veces en que un usuario intente acceder a él.

Tabla 3.80: Requisito no funcional 006

Requisito no funcional 007	
Nombre	Probabilidad de fallo.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	La probabilidad de falla del Sistema no podrá ser mayor a 0,05.

Tabla 3.81: Requisito no funcional 007

Requisito no funcional 008	
Nombre	Lenguaje de codificación.
Versión	1.0 -05/04/2019
Autora	Alba Francisco Gutiérrez
Fuentes	Álvaro Estébanez Barrena y Alberto Casero De La Calle
Descripción	El sistema deberá ser codificado en lenguaje C#.

Tabla 3.82: Requisito no funcional 008

3.10. Actores

Actor 001	
Nombre	Usuario
Versión	1.0-04/05/2019
Autor	Alba Francisco Gutiérrez
Descripción	Será un cliente que gestionará los datos de *Microsoft Dynamics 365*. Creará, modificará y actualizará los registros.

Tabla 3.83: Actor 001

Actor 002	
Nombre	Tiempo
Versión	1.0-04/05/2019
Autor	Alba Francisco Gutiérrez
Descripción	Las *Azure Functions* que conectan con *Microsoft Dynamics 365* lo hacen de manera periódica. La manera más adecuada de representar este comportamiento es con un actor.

Tabla 3.84: Actor 002

Actor 003	
Nombre	Equipo de gestión
Versión	1.0-04/05/2019
Autor	Alba Francisco Gutiérrez
Descripción	Es un usuario final con funciones de administración. Será el que recibas lo e-mail cuando se sobrepase un número concreto de mensajes en las colas manuales.

Tabla 3.85: Actor 003

3.11. Casos de uso

Nombre de ID del CU	**CU-01 Extracción de registros creados en *Microsoft Dynamics 365***
Actor	Sistema
Descripción	La información de un registro, que previamente ha sido creados en *Microsoft Dynamics 365*, se encolará en *Service Bus*.
Pre-condiciones	El registro debe de haber sido creado, durante el intervalo del último minuto, en *Microsoft Dynamics 365*.
Post-condiciones	La información del registro estará almacenada, en forma de mensaje, en *Service Bus*.
Flujo normal	1. El sistema se conecta a *Microsoft Dynamics 365*
	2. El sistema extrae todos los registros que cumplen la condición de haber sido modificados dentro del intervalo del último minuto al momento de la conexión.
	3. El sistema evaluará si el registro extraído es un registro recién creado.
	4. El sistema genera un mensaje con los datos extraídos.
	5. El sistema enviará al *Service Bus* el mensaje.
Flujo Alternativo	3.1 Si el registro extraído ha sido actualizado se realizará el caso de uso Extracción de registros actualizados en *Microsoft Dynamics 365.*.
Excepciones	**1. El sistema no puede conectar con *Microsoft Dynamics 365*.**
	4. El sistema no puede conectar con *Service Bus*.

Tabla 3.86: Requisito CU-01 Extracción de registros creados en *Microsoft Dynamics 365.*

Nombre de ID del CU	**CU-02 Extracción de registros actualizados en *Microsoft Dynamics 365***
Actor	Sistema
Descripción	La información de un registro, que previamente ha sido actualizado en *Microsoft Dynamics 365*, se encolará en *Service Bus*.
Pre-condiciones	El registro debe de haber sido actualizado, durante el intervalo del último minuto, en *Microsoft Dynamics 365*.
Post-condiciones	La información del registro estará almacenada, en forma de mensaje, en *Service Bus*.
Flujo normal	1. El sistema se conecta a *Microsoft Dynamics 365*
	2. El sistema extrae todos los registros que cumplen la condición de haber sido modificados dentro del intervalo del último minuto al momento de la conexión.
	3. El sistema evaluará si el registro extraído es un registro que sido actualizado.
	4. El sistema generará un mensaje con los datos extraídos.
	5. El sistema enviará al *Service Bus* el mensaje.
Flujo Alternativo	3.1 Si el registro extraído es de nueva creación se realizará el caso de uso Extracción de registros creados en el CRM.
Excepciones	**1. El sistema no puede conectar con *Microsoft Dynamics 365*.**
	4. El sistema no puede conectar con *Service Bus*.

Tabla 3.87: Requisito CU-02 Extracción de registros actualizados en *Microsoft Dynamics 365.*

Nombre de ID	**CU-03 Extracción de registros eliminados en *Microsoft Dynamics 365***
Actor	Sistema
Descripción	La información de un registro, que previamente ha sido eliminado en *Microsoft Dynamics 365*, se encolará en *Service Bus*.
Pre-condiciones	El mensaje debe de estar almacenado dentro de *Service Bus*.
Post-condiciones	La información del registro estará almacenada, en forma de mensaje, en *Service Bus*.
Flujo normal	1. El sistema se conecta a *Microsoft Dynamics 365*
	2. El sistema extrae todos los registros que han sido eliminados de *Microsoft Dynamics 365*
	3. El sistema genera un mensaje con los datos extraídos.
	4. El sistema enviará al *Service Bus* el mensaje.
	5.El sistema eliminará, definitivamente, el registro dentro de *Microsoft Dynamics 365*.
Excepciones	**1. El sistema no puede conectar con *Microsoft Dynamics 365*.**
	4. El sistema no puede conectar con *Service Bus*.

Tabla 3.88: Requisito CU-03 Extracción de registros eliminados en *Microsoft Dynamics 365.*

Nombre de ID	**CU-04 Creación de registros en la base de datos Oracle**
Actor	Sistema
Descripción	La información de un registro, que estaba encolado en *Service Bus*, se almacenará dentro del sistema de bases de datos de Oracle.
Pre-condiciones	El mensaje debe de estar almacenado dentro de *Service Bus*.
Post-condiciones	La información del registro estará almacenada, en forma de mensaje, en *Service Bus*.
Flujo normal	1. El sistema detectará que hay un mensaje dentro de *Service Bus*.
	2. El sistema extrae el mensaje de la cola de *Service Bus*.
	3. El sistema comprueba que si se ha intentado almacenar más de una vez el mensaje recibido en Oracle.
	4. El sistema creará un comando de tipo SQL con la información recibida dentro del mensaje extraído de *Service Bus*.
	5. El sistema enviará el comando al sistema de bases de datos Oracle.
	6. El sistema de bases de datos Oracle creará un nuevo registro con la información recibida
	7. El caso de uso se ejecuta hasta que no exista más mensajes en *Service Bus*.
Flujo alternativo	**3.1 Si el mensaje recibido se ha intentado más de una vez.**
	3.1.1 El sistema creará un comando de consulta.
	3.1.2 El sistema envía el comando al sistema de bases de datos Oracle.
Excepciones	**5. y de 3.1.2 El sistema de bases de datos de Oracle lanza un error relacionado con problemas de conexión.**
	5.1 El sistema aumenta en el contador de reintentos del mensaje.
	5.2 El sistema envía el mensaje a la cola de reintentos automática.
	5. y de 3.1.2 El sistema de bases de datos de Oracle lanza un error relacionado con problemas asociados al mensaje.
	5.1 El sistema envía el mensaje a la cola de reintentos manual.

Tabla 3.89: Requisito CU-04 Creación de registros en Oracle

Nombre de ID	**CU-05 Actualización de registro en la base de datos de Oracle.**
Actor	Sistema
Descripción	Los mensajes extraídos del *Service Bus* serán almacenados en el sistema de bases de datos de Oracle.
Pre-condiciones	El mensaje debe de estar almacenado en la cola de *Service Bus*.
Post-condiciones	Las información extraída del mensaje, actualizará la información que está almacenada en el sistema de bases de datos de Oracle.
Flujo normal	1. El sistema detecta que hay un nuevo mensaje dentro de *Service Bus*.
	2. El sistema extrae el mensaje del *Service Bus*.
	3. El sistema crea un comando *SQL* de consulta en relación al mensaje extraído.
	4. El sistema envía el comando de consulta a Oracle.
	5. El sistema de bases de datos de Oracle devuelve la información que tiene relativa a la información del mensaje.
	6. El sistema evalúa que campos han sido modificados en el mensaje extraído en comparación a la información almacenada en Oracle
	7. El sistema genera un comando *SQL* con los campos modificados.
	8. El sistema envía el comando al sistema de bases de datos de Oracle.
Flujo Alternativo	4.1 Si no se encuentra información en Oracle relativa a la información recibida por *Service Bus*.
	4.1.1 El sistema crea un comando *SQL* de creación de registro con la información del mensaje.
	4.1.2 El sistema envía el comando al sistema de bases de datos de Oracle.
	5.1 Si la información almacenada en Oracle es posterior a la información recibida por *Service Bus* el caso de uso finaliza.
Excepciones	**4 y 8 El sistema no puede conectar con Oracle debido a fallos relativos a la conexión.**
	1. El mensaje aumentará en uno el contador de reintentos del mensaje
	2. El sistema envía el mensaje a la cola de reintentos automáticos.
	4. El sistema no puede conectar con Oracle debido a fallos relacionados con el mensaje o el contador de reintentos supera el límite.
	1. El sistema envía el mensaje a la cola de reintentos manual.

Tabla 3.90: Requisito CU-05 Actualización de registros en Oracle

Nombre de ID del CU	**CU-06 Eliminación de registro de la base de datos de Oracle.**
Actor	Sistema
Descripción	Los mensajes extraídos del *Service Bus* serán eliminados en el sistema de bases de datos de Oracle.
Pre-condiciones	El mensaje debe de estar almacenado en la cola de *Service Bus*.
Post-condiciones	Las información extraída del mensaje, actualizará la información que esta almacenada en el sistema de bases de datos de Oracle.
Flujo normal	1. El sistema detecta que hay un nuevo mensaje dentro de *Service Bus*.
	2. El sistema extrae el mensaje del *Service Bus*.
	3. El sistema crea un comando *SQL* de eliminación en relación al mensaje extraído.
	4. El sistema envía el comando de consulta a Oracle.
Excepciones	**4 y 8 El sistema no puede conectar con Oracle debido a fallos relativos a la conexión.**
	1. El mensaje aumentará en un el contador de reintentos del mensaje
	2. El sistema envía el mensaje a la cola de reintentos automáticos.
	4. El sistema no puede conectar con Oracle debido a fallos relacionados con el mensaje o el contador de reintentos supera el límite.
	1. El sistema envía el mensaje a la cola de reintentos manual.

Tabla 3.91: Requisito CU-06 Eliminación de registros en Oracle

Nombre de ID del CU	**CU-07 Reenvío de mensajes que están almacenados en la cola de reintentos autómatica.**
Actor	Sistema
Descripción	Los mensajes almacenados dentro de la cola de reintentos automática deberán ser redigidos a las colas pertinentes en el momento en que el sistema de bases de datos de Oracle este disponible.
Pre-condiciones	El mensaje debe de estar almacenado en la cola de *Service Bus* de reintentos automáticos.
Post-condiciones	Los mensajes extraídos de la cola de reintentos automáticos deberán estar en las colas pertinentes.
Flujo normal	1. El sistema envía un comando de pruebas al sistema de bases de datos de Oracle para comprobar su disponibilidad.
	2. El sistema extrae el mensaje de la cola.
	3. El sistema evalúa a cual cola debe de ser redigido el mensaje.
	4. El sistema envía el mensaje a la cola pertinente.
Excepciones	**1. El sistema no puede conectar con Oracle.**
	El caso de uso finaliza.
	3. El mensaje ha llegado erróreno y no se puede enviar a ninguna cola.
	El sistema reenvía el mensaje erróneo a la cola de reintentos manual.
	4. El sistema no puede conectar con *Service Bus*.

Tabla 3.92: Requisito CU-07 Reenvío de mensajes que están almacenados en la cola de reintentos automática.

Nombre de ID del CU	**CU-08 Regularización de registros en *Microsoft Dynamics 365***
Actor	Sistema
Descripción	Se realizará una regularización de registros creado o actualizados en un intervalo de tiempo concreto enviando la información relativa a ellos a *Service Bus*.
Pre-condiciones	Debe de haber mensajes, dentro de un intervalo de tiempo, creados o actualizados que no fueron enviados a *Service Bus*.
Post-condiciones	Los mensajes extraídos estarán en *Service Bus*.
Flujo normal	1. El sistema pide que se introduzca la fecha de inicio.
	2. El usuario proporciona fecha de inicio.
	3. El sistema pide que se introduzca la fecha de fin.
	4. El usuario introduce la fecha de fin.
	5. El sistema se conecta a *Microsoft Dynamics 365*.
	6. El sistema extrae todos los registros, creados o actualizados, dentro del periodo de tiempo proporcionado por el usuario.
	7. El sistema evalúa que registros han sido de nueva creación y cual han sido actualizados.
	8. El sistema genera un mensaje con la información extraída de *Microsoft Dynamics 365*.
	9. El sistema envía el mensaje a *Service Bus*.
Excepciones	**5. El sistema no puede conectar con *Microsoft Dynamics 365*.**
	9. El sistema no puede conectar con *Service Bus*.

Tabla 3.93: Requisito CU-08 Regularización de registros en *Microsoft Dynamics 365*

Nombre de ID del CU	**CU-09 Almacenamiento de los campos relevantes ante la eliminación de un registro en *Microsoft Dynamics 365***
Actor	Sistema
Descripción	Se almacenarán en otra entidad los campos, y la información asociada a ellos, de los registros eliminados en *Microsoft Dynamics 365* para su posterior eliminación en el sistema de bases de datos de Oracle.
Pre-condiciones	Debe de haber un registro creado en *Microsoft Dynamics 365*.
Post-condiciones	Los datos salvaguardados estarán almacenados dentro de otra entidad en *Microsoft Dynamics 365*.
Flujo normal	1. El usuario elimina un registro dentro del *Microsoft Dynamics 365*.
	2. El sistema extrae la información necesaria de ese registro.
	3. El sistema crea un registro en otra entidad.
	4. El sistema almacena la información salvaguardada dentro del nuevo registro.
Excepciones	**2. El sistema no puede conectar con *Microsoft Dynamics 365*.**
	2. El sistema no puede extraer la información que debe de ser salvaguardada.
	3. El sistema no puede crear un nuevo registro.
	4. El sistema no puede almacenar la información en el nuevo registro.

Tabla 3.94: Requisito CU-09 Almacenamiento de los campos relevantes ante la eliminación de un registro en *Microsoft Dynamics 365*

Nombre de ID del CU	**CU-10 Envío de e-mail al equipo de desarrollo por superación de límites en la cola *DeadLetter* de *Create***
Actor	Sistema
Descripción	Se enviará un e-mail al equipo de desarrollo cuando se supere el límite de mensajes dentro de la cola de DeadLetter de Create.
Pre-condiciones	Debe de haber mensajes en la cola de DeadLetter de Create.
Post-condiciones	El equipo de desarrollo recibirá un e-mail de aviso.
Flujo normal	1. El sistema evalúa la cantidad de mensajes almacenados en la cola
	2. El sistema conecta con *Microsoft Dynamics 365*.
	3. El sistema crea un registro email en *Microsoft Dynamics 365*.
	4. *Microsoft Dynamics 365* envía el e-mail al equipo de desarrollo.
Flujo Alternativo	1. El número de mensajes en la cola no supera el límite.
	1.1 El caso de uso finaliza.
Excepciones	**1.El sistema no puede conectar con *Service Bus*.**
	3.El sistema no puede conectar con *Microsoft Dynamics 365*.
	3. El sistema no puede crear un nuevo registro email.
	4. *Microsoft Dynamics 365* no puede enviar el email.

Tabla 3.95: Requisito CU-10 Envío de e-mail al equipo de desarrollo por superación de limites en la cola *DeadLetter* de *Create*

Nombre de ID del CU	**CU-11 Envío de e-mail al equipo de desarrollo por superación de limites en la cola *DeadLetter* de *Update***
Actor	Sistema
Descripción	Se enviará un e-mail al equipo de desarrollo cuando se supere el límite de mensajes dentro de la cola de *DeadLetter* de *Update*.
Pre-condiciones	Debe de haber mensajes en la cola de DeadLetter de *Update*.
Post-condiciones	El equipo de desarrollo recibirá un e-mail de aviso.
Flujo normal	1. El sistema evalúa la cantidad de mensajes almacenados en la cola
	2. El sistema conecta con *Microsoft Dynamics 365*.
	3. El sistema crea un registro email en *Microsoft Dynamics 365*.
	4. *Microsoft Dynamics 365* envía el e-mail al equipo de desarrollo.
Flujo Alternativo	1. El número de mensajes en la cola no supera el límite.
	1.1 El caso de uso finaliza.
Excepciones	**1.El sistema no puede conectar con *Service Bus*.**
	3.El sistema no puede conectar con *Microsoft Dynamics 365*.
	3. El sistema no puede crear un nuevo registro e-mail.
	4. *Microsoft Dynamics 365* no puede enviar el e-mail.

Tabla 3.96: Requisito CU-11 Envío de e-mail al equipo de desarrollo por superación de límites en la cola *DeadLetter* de *Update*

Nombre de ID del CU	**CU-12 Envío de e-mail al equipo de desarrollo por superación de límites en la cola *DeadLetter* de *Delete***
Actor	Sistema
Descripción	Se enviará un e-mail al equipo de desarrollo cuando se supere el límite de mensajes dentro de la cola de *DeadLetter* de *Delete*.
Pre-condiciones	Debe de haber mensajes en la cola de DeadLetter de Delete.
Post-condiciones	El equipo de desarrollo recibirá un e-mail de aviso.
Flujo normal	1. El sistema evalúa la cantidad de mensajes almacenados en la cola
	2. El sistema conecta con *Microsoft Dynamics 365*.
	3. El sistema crea un registro email en *Microsoft Dynamics 365*.
	4. *Microsoft Dynamics 365* envía el e-mail al equipo de desarrollo.
Flujo Alternativo	1. El número de mensajes en la cola no supera el límite.
	1.1 El caso de uso finaliza.
Excepciones	**1.El sistema no puede conectar con *Service Bus*.**
	3.El sistema no puede conectar con *Microsoft Dynamics 365*.
	3. El sistema no puede crear un nuevo registro e-mail.
	4. *Microsoft Dynamics 365* no puede enviar el e-mail.

Tabla 3.97: Requisito CU-12 Envío de e-mail al equipo de desarrollo por superación de límites en la cola *DeadLetter* de *Delete*

Nombre de ID del CU	**CU-13 Envío de e-mail al equipo de gestión por superación de límite de registros creados por la cola de reintentos manual.**
Actor	Sistema
Descripción	Se enviará un e-mail al equipo de gestión cuando se supere el límite de registros asociados a mensajes fallidos, creados por la cola de reintentos manuales.
Pre-condiciones	Debe de haber registros de este tipo en *Microsoft Dynamics 365*.
Post-condiciones	El equipo de desarrollo recibirá un e-mail de aviso.
Flujo normal	1. El sistema evalúa la cantidad de registros creados por la cola manual.
	2. El sistema conecta con *Microsoft Dynamics 365*.
	3. El sistema crea un registro email en *Microsoft Dynamics 365*.
	4. *Microsoft Dynamics 365* envía el e-mail al equipo de desarrollo.
Flujo Alternativo	1. El número de registros creados por la cola no supera el límite.
	1.1 El caso de uso finaliza.
Excepciones	**1.El sistema no puede conectar con *Service Bus*.**
	3.El sistema no puede conectar con *Microsoft Dynamics 365*.
	3. El sistema no puede crear un nuevo registro email.
	4. *Microsoft Dynamics 365* no puede enviar el e-mail

Tabla 3.98: Requisito CU-13 Envío de e-mail al equipo de gestión por superación de límite de registros creados por la cola de reintentos manual

Nombre de ID del CU	**CU-14 Inserción de mensajes fallidos en CRM Dynamcis**
Actor	Sistema
Descripción	Los mensajes fallidos, se insertarán en *Microsoft Dynamics 365* para facilitar su tratamiento por el equipo de gestión.
Pre-condiciones	Debe de haber mensajes en la cola de mensajes manual
Post-condiciones	El mensaje fallido estará almacenado dentro de *Microsoft Dynamics 365*.
Flujo normal	1. El sistema detecta que hay un nuevo mensaje almacenado en la cola manual.
	2. El sistema extrae el mensaje de la cola manual
	3. El sistema conecta con *Microsoft Dynamics 365*.
	4. El sistema crea una nueva entidad en *Microsoft Dynamics 365* con la información del mensaje.
Excepciones	**1. El sistema no puede conectar con *Service Bus***
	3. El sistema no puede realizar la conexión con *Microsoft Dynamics 365*
	4. El sistema no puede crear una nueva entidad con la información asociada.

Tabla 3.99: Requisito CU-13 Inserción de mensajes fallidos en CRM Dynamcis 365

3.12. Diagrama de clases

En la siguiente imagen 3.10 se detalla el diagrama de clases de análisis asociado al sistema:

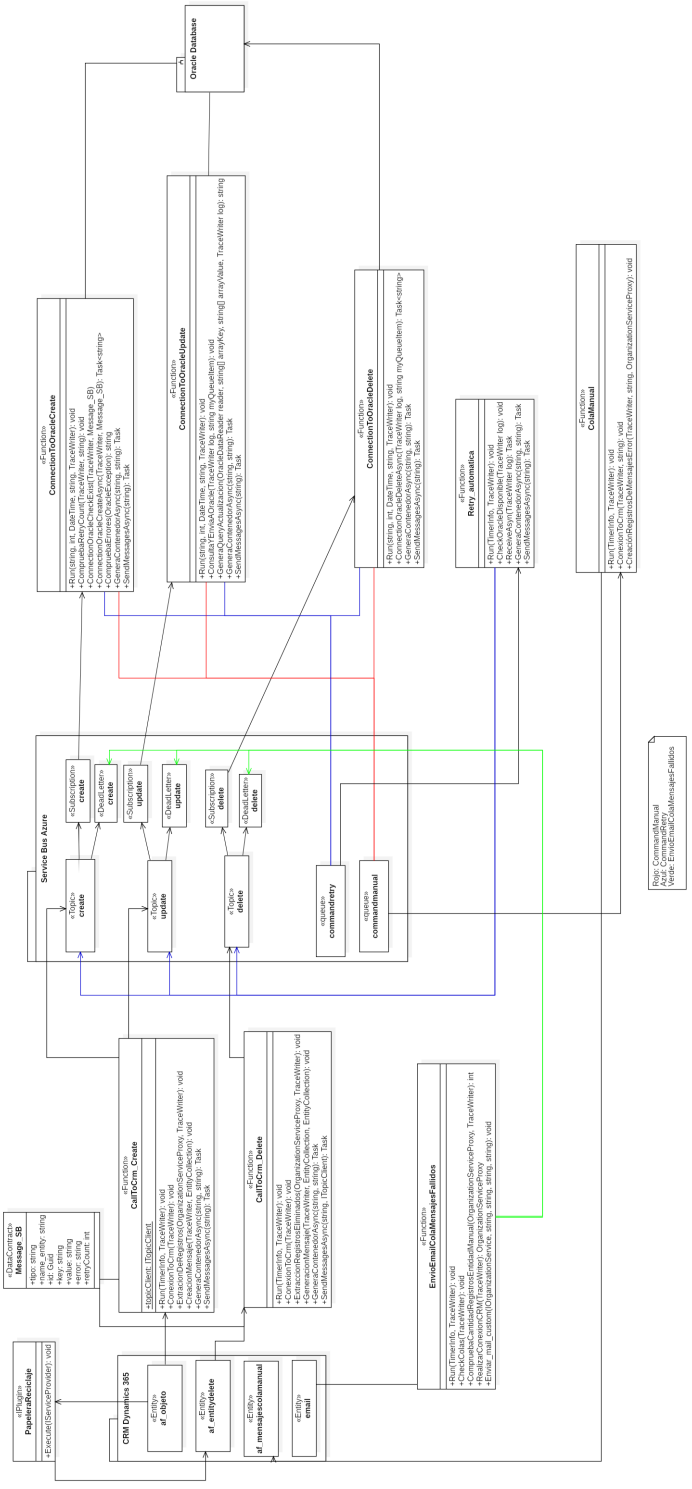

Figura 3.10: Diagrama de clases de análisis.

3.13. Diagramas de secuencia

En la siguiente sección se incluirán los diagramas de secuencia de análisis, los cuales son un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML, asociados al sistema.

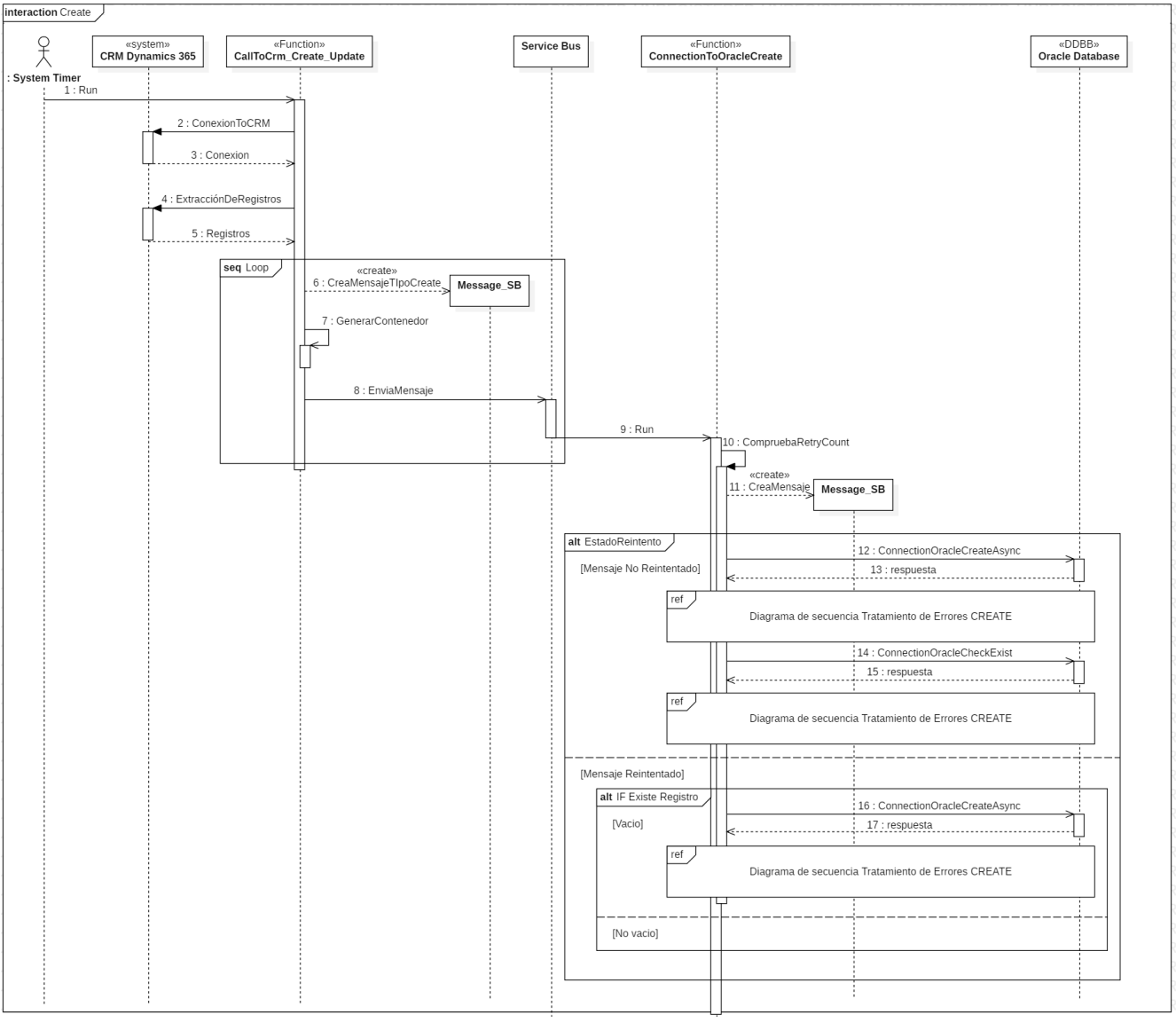

Figura 3.11: Diagrama de secuencia de *Create*.

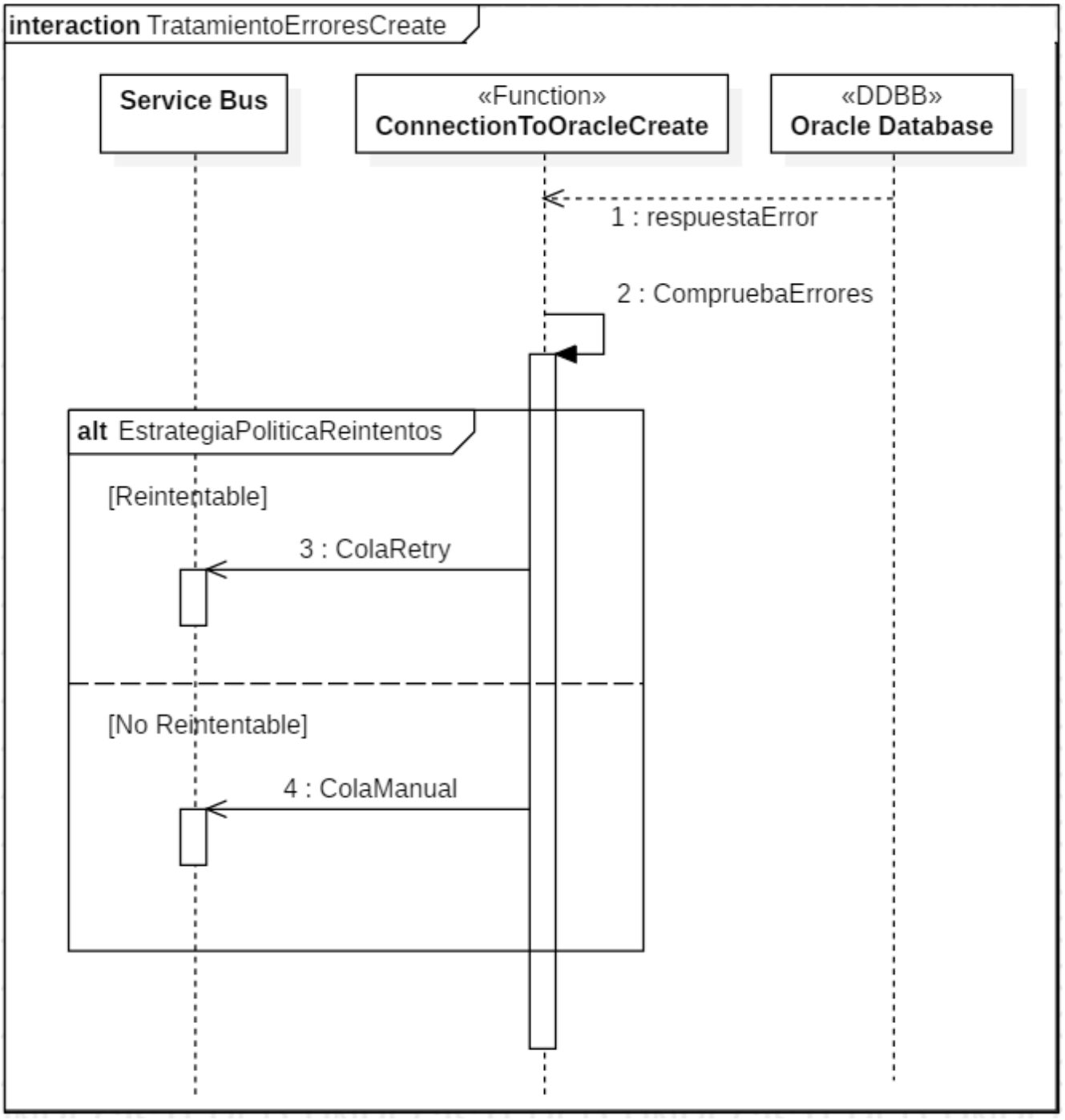

Figura 3.12: Diagrama de secuencia de la política de reintentos de Create.

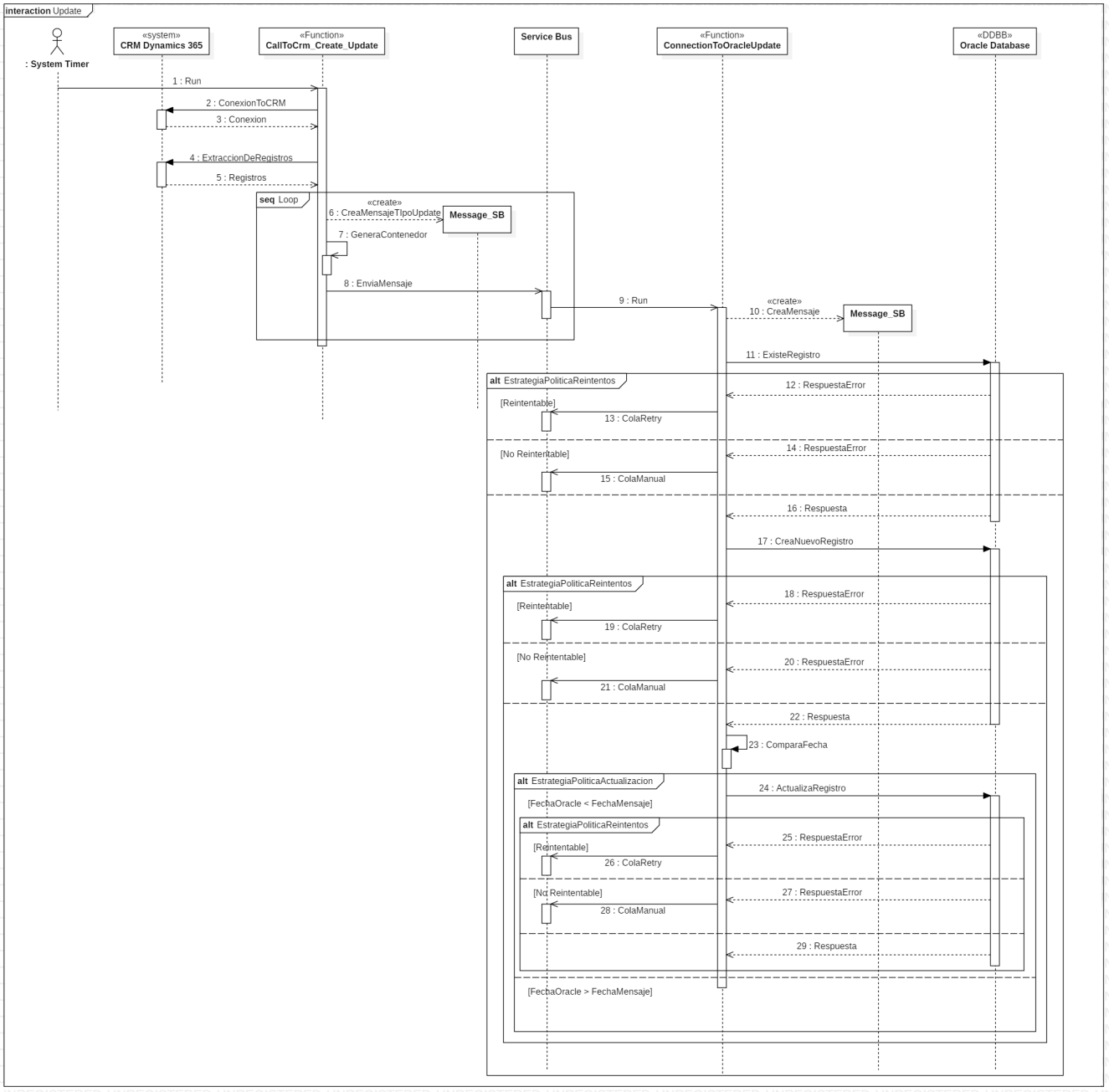

Figura 3.13: Diagrama de secuencia de *Update*.

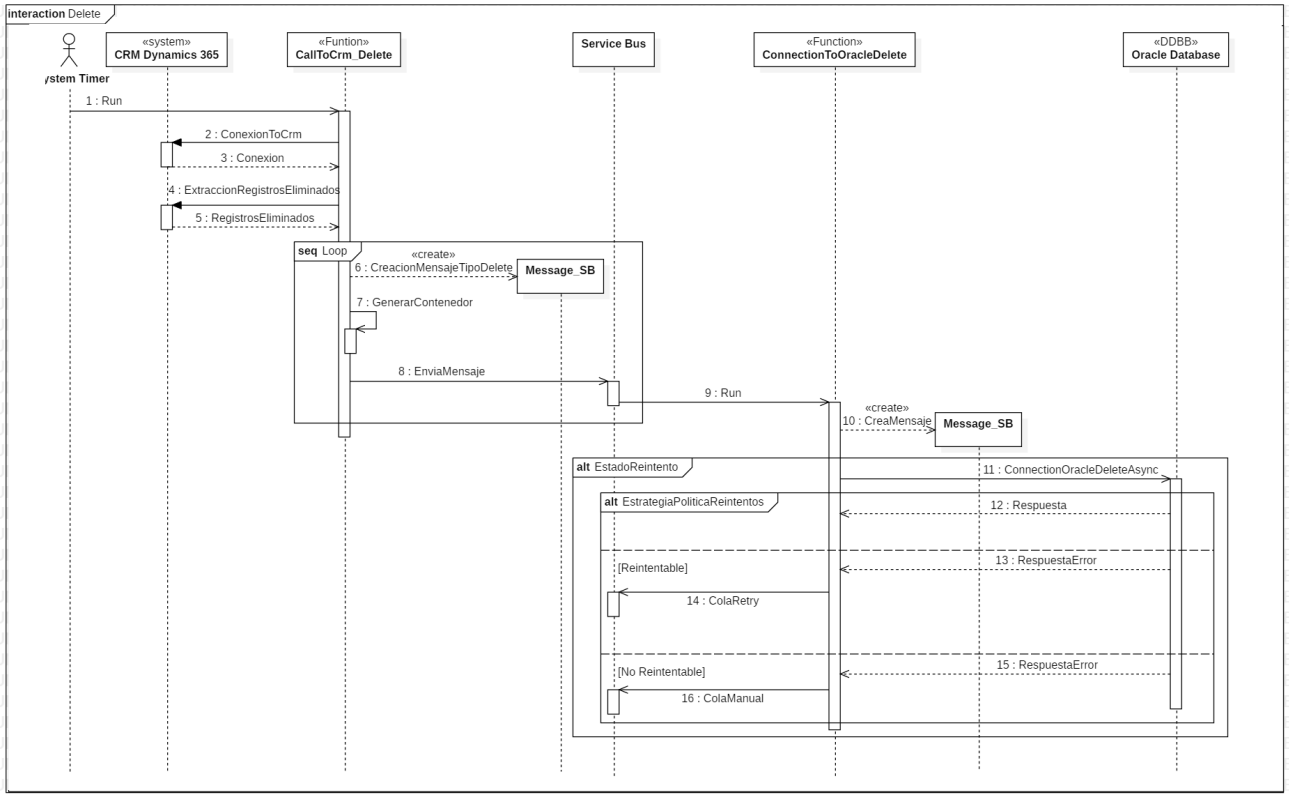

Figura 3.14: Diagrama de secuencia de Delete.

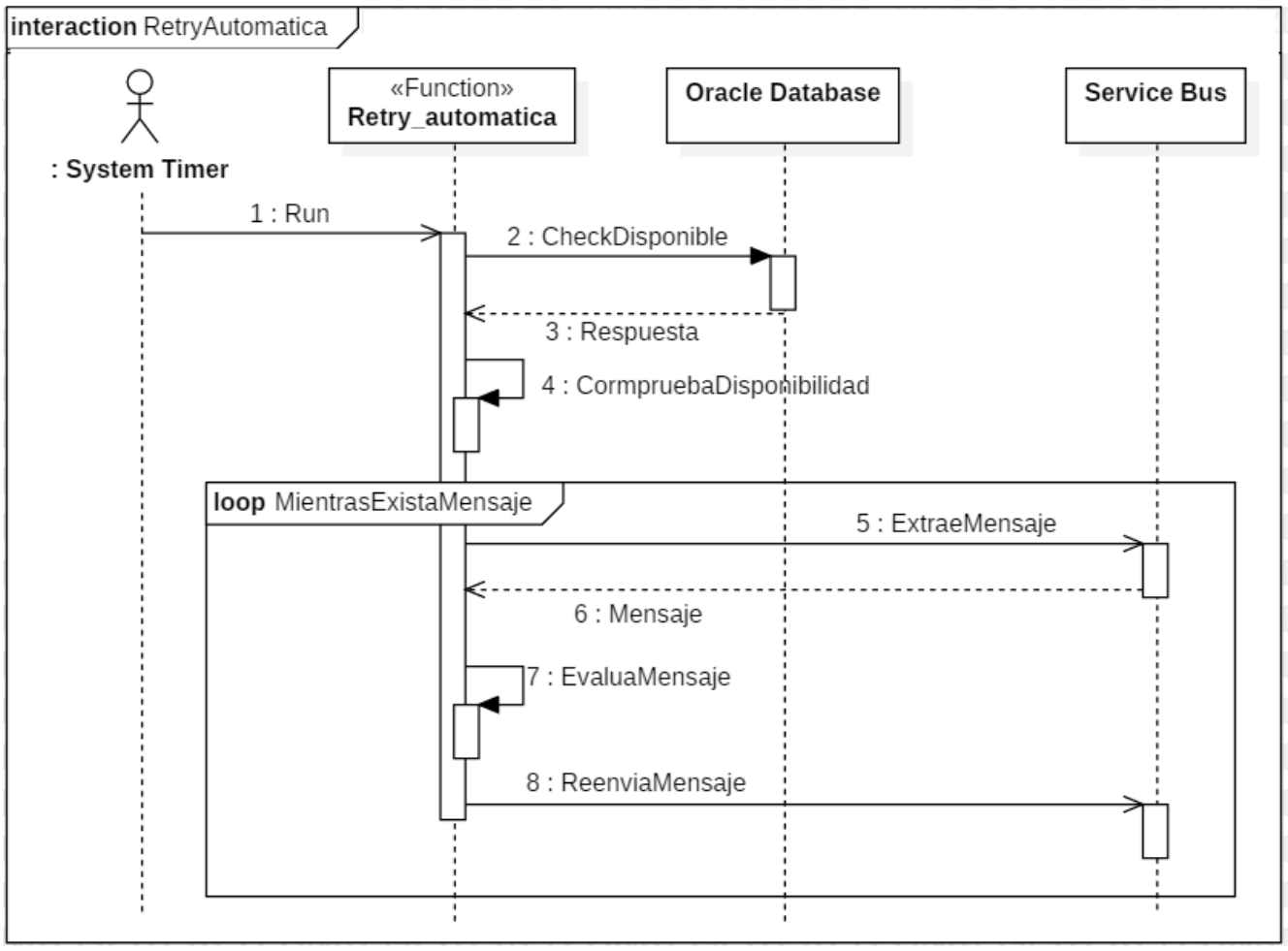

Figura 3.15: Diagrama de secuencia de la *Azure Function* encargada de reintentar automáticamente los mensajes.

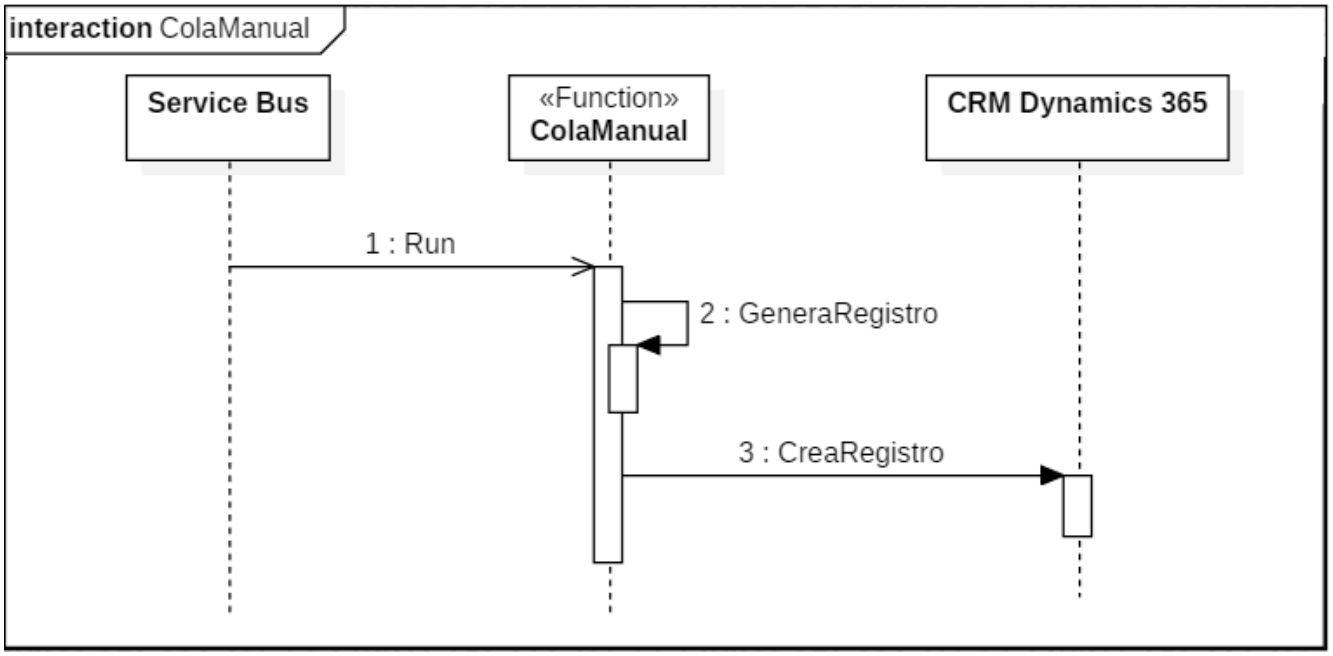

Figura 3.16: Diagrama de secuencia de la *Azure Function* encargada de reintentar automáticamente los mensajes.

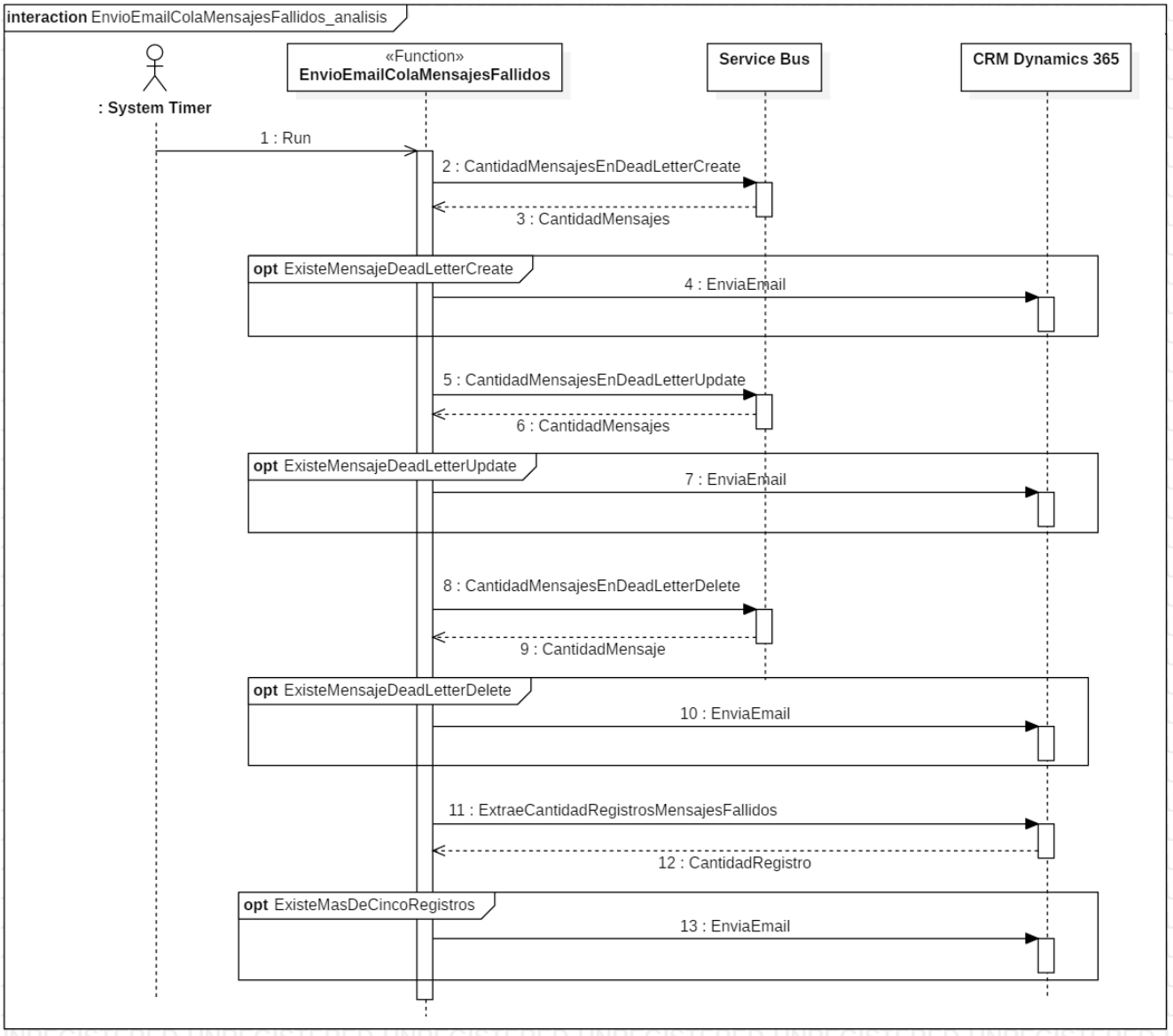

Figura 3.17: Diagrama de secuencia de la *Azure Function* encargada enviar e-mail .

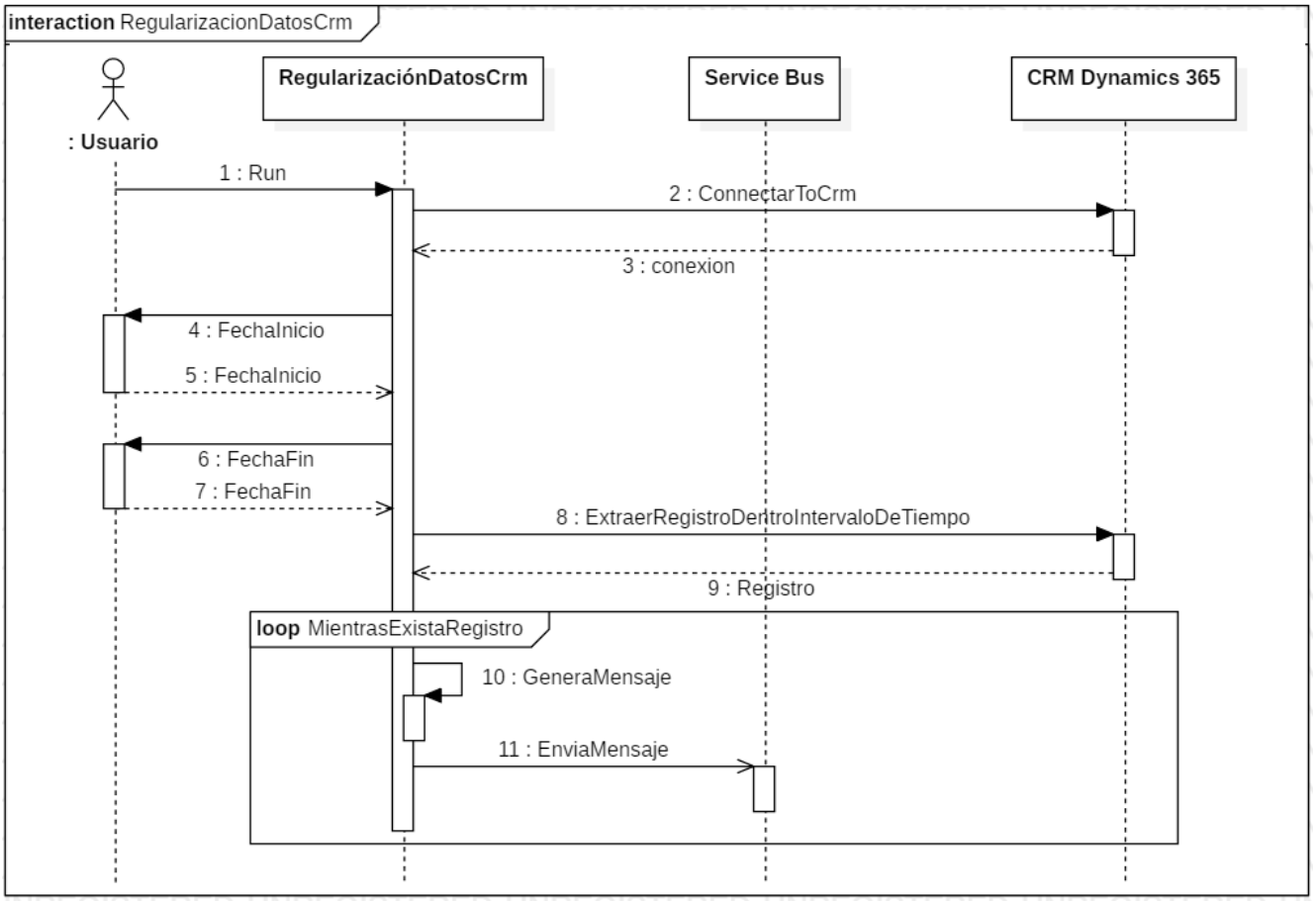

Figura 3.18: Diagrama de secuencia del programa de consola que regulariza registros .

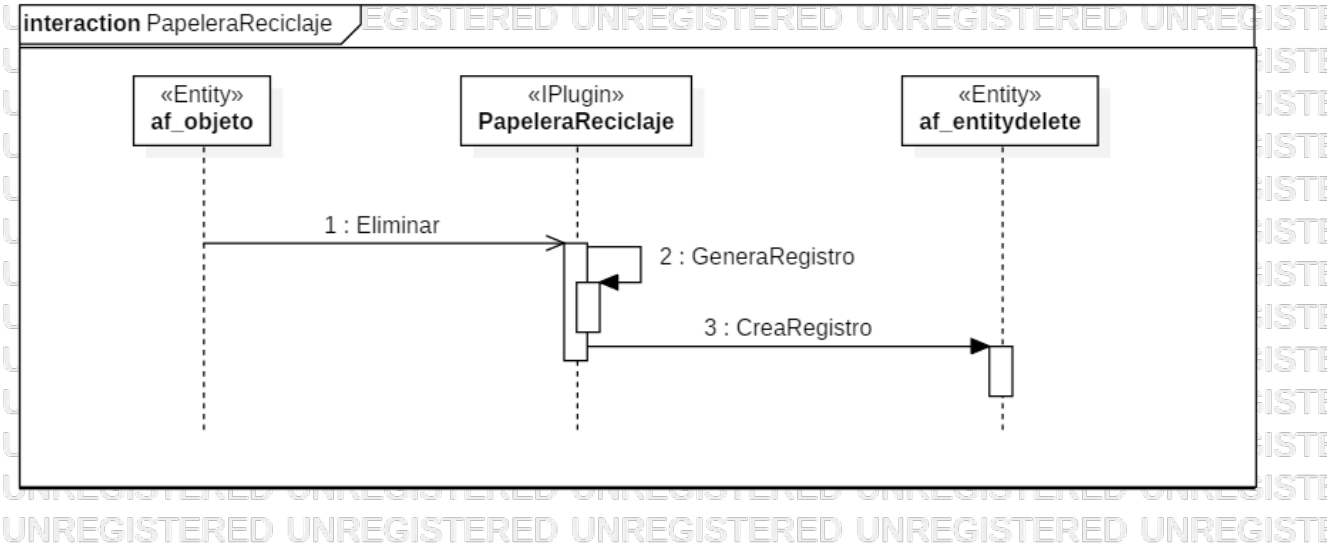

Figura 3.19: Diagrama de secuencia del *Plugin*.

Capítulo 4

Diseño

En el presente capítulo, se detallará el diseño del sistema. Esta etapa permite determinar como funcionará de forma general, sin entrar en detalles, el sistema. Se detallará el diseño de los componentes del sistema que dan respuesta a las funcionalidades, descritas en la etapa de análisis, también conocidas como las entidades de negocio. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación y para describir la interacciones entre la entidades y el secuenciado, se realizan diagramas que lo muestren claramente.

4.1. Patrones

Para el desarrollo de este proyecto se ha utilizado los siguientes patrones, que explicaremos más en detalle a continuación, además de detallar las razones detrás de la elección de su uso.

4.1.1. Patrón *Command and Query Responsibility Segregation* (CQRS)

Command and Query Responsibility Segregation (CQRS) es un patrón que segrega las operaciones que leen datos (consultas) de las operaciones que actualizan los datos (comandos) mediante interfaces independientes. Esto significa que los modelos de datos utilizados para realizar las consultas y las actualizaciones son diferentes. Posteriormente, se pueden aislar los modelos, como se indica en la imagen 4.1, aunque eso no es un requisito imprescindible.

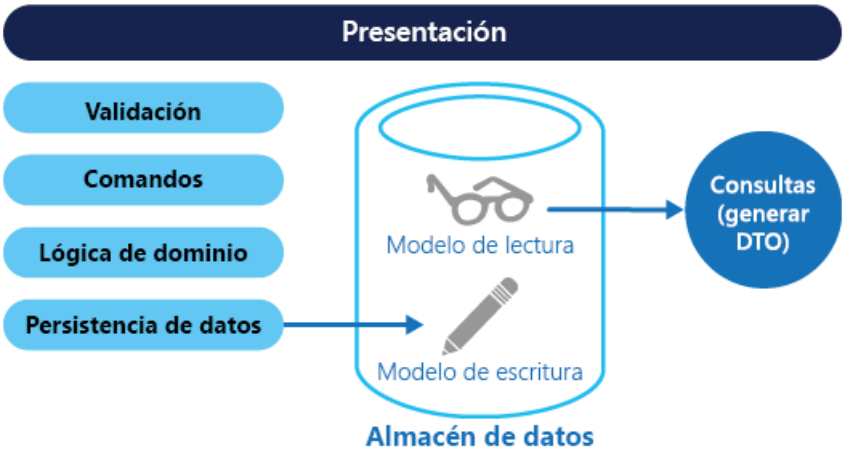

Figura 4.1: Aislamiento de modelos.

En comparación con el modelo de datos único que se usa en los sistemas basados en *CRUD*, el uso de modelos de consulta y actualización independientes para los datos de los sistemas basados en *CQRS* simplifica el diseño y la implementación. Sin embargo, una desventaja es que, a diferencia de los diseños *CRUD*, el código *CQRS* no se genera automáticamente mediante los mecanismos de scaffolding.

El modelo de consulta para leer datos y el de actualización para escribirlos pueden acceder el mismo almacén físico, quizás mediante vistas *SQL* o mediante la generación de proyecciones sobre la marcha. Sin embargo, es habitual separar los datos en almacenes físicos diferentes para maximizar el rendimiento, la escalabilidad y la seguridad, tal como se muestra en la figura 4.2.

Figura 4.2: Separación de datos.

El almacén de lectura puede ser una réplica de solo lectura del almacén de escritura o los almacenes de lectura y escritura pueden tener una estructura diferente por completo. El uso de réplicas de solo lectura del almacén de lectura puede aumentar mucho el rendimiento de las consultas y la capacidad de respuesta de la interfaz de usuario de la aplicación, especialmente en escenarios distribuidos en los que las réplicas de solo lectura se sitúan cerca de las instancias de la aplicación. Algunos sistemas de bases de datos *(SQL Server)* proporcionan características adicionales como la conmutación por error de réplicas para maximizar la disponibilidad.

La separación de los almacenes de lectura y escritura permite también que cada uno de ellos pueda escalarse adecuadamente para adaptarse a la carga 4.3. Por ejemplo, los almacenes de lectura normalmente se encuentran con una carga mayor que los de escritura [12]. En el presente trabajo, solo se implementará la parte de *command* del patrón por cuestiones de requisitos de cliente.

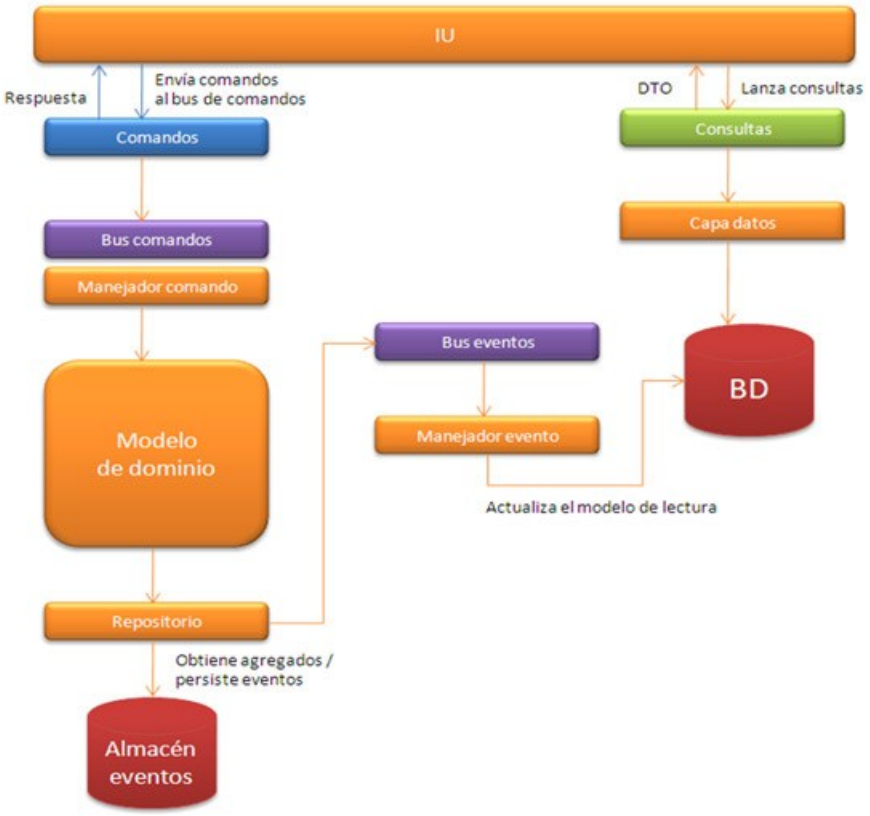

Figura 4.3: Modelo Command and Query Responsibility Segregation (CQRS) desarrollado.

Razones por las cuales se ha implementado este patrón

En este apartado se desarrollará las razones por las cuales se ha decidido implementar este patrón en relación a las recomendaciones de Microsoft.

CRM Dynamics cumple las siguientes permisas:

- Dominios colaborativos en los que se realizan varias operaciones en paralelo en los mismos datos. *CQRS* permite definir comandos con la suficiente granularidad para minimizar los conflictos de combinación en el nivel de dominio (cualquier conflicto que surja se podrá combinar mediante el comando), incluso cuando se actualiza lo que parece ser el mismo tipo de datos.

- Interfaces de usuario basadas en tareas en las que se guía al usuario a través de un complejo proceso como una serie de pasos o con modelos de dominio complejos. El modelo de escritura tiene una pila de procesamiento de comandos completa con lógica de negocios, validación de entradas y validación empresarial para garantizar que todo es siempre coherente para cada uno de los agregados del modelo de escritura.

  El modelo de lectura no tiene ninguna lógica de negocios ni pila de validación y solo devuelve un *DTO* para su uso en un modelo de vista. El modelo de lectura tiene coherencia final con el modelo de escritura.

- Los escenarios en los que el rendimiento de las lecturas de datos se debe ajustar de manera independiente a la del rendimiento de las escrituras de datos, especialmente cuando la relación de lectura/escritura es muy alta y se requiere un escalado horizontal.

Y, a razón del uso de *Azure Funtions* :

- Escenarios en los que un equipo de desarrolladores pueda centrarse en el modelo de dominio complejo que forma parte del modelo de escritura, y otro equipo pueda centrarse en el modelo de lectura y en las interfaces de usuario. En el caso del alcance de este TFG, solo se llevará acabo la implementación de los modelos de escritura.
- Escenarios en los que se espera que el sistema evolucione con el tiempo y que podrían contener varias versiones del modelo, o en los que las reglas de negocio cambian con regularidad.

4.1.2. Patrón *Imperative binder*

El patrón *Imperative binder* se puede utilizar en *C#* y otros lenguajes *.NET*, en contraposición a los enlaces declarativos de *function.json* y los atributos. Los enlaces imperativos resultan útiles cuando los parámetros de enlace tienen que calcularse en tiempo de ejecución, en lugar de en el tiempo de diseño [13]. Al definir la entrada y la salida declarativa, no tiene la opción de cambiar algunas de las propiedades de enlace como el nombre o realizar varias salidas desde una entrada.

Este patrón se utilizara, en las *Azure Functions* que extraen mensajes de *Service Bus*. Estas *Azure Functions*, son del tipo "*ServiceBusTrigger*" y comienzan a funcionar el momento en que detectan un cambio en el estado de la cola o tema a la cual están asociadas, en este caso, el cambio es la inserción de un mensajes en *Service Bus*.

4.1.3. Patrón *Transaction Script* y Patrón *Data Access Object (DAO)*

El patrón *Transaction Script* organiza toda la lógica como un solo procedimiento y se comunica con la base de datos generando una sola sesión transaccional de comunicación con ella. Cada procedimiento es independiente de las cases de presentación y de la base de datos.

Se utiliza este patrón debido a que el patrón *DAO* está implícito en las bibliotecas de acceso a la base de datos de Oracle y ambos patrones se aconseja que se utilicen de de manera conjunta. Ambos patrones permiten un acceso sólido y robusto a la base de datos y fácilmente mutable a los cambios en las reglas de negocio. En el contexto de este proyecto, las *Azure Functions* que se comunican con la base de datos de Oracle implementan el patrón. El objeto de transferencia de datos (*DTO*) está representada por la clase *DataContract Menssage_SB*.

4.1.4. Patrón *Retry*

En la nube, las fallas transitorias no son infrecuentes y una aplicación debe diseñarse para manejarlas con elegancia y transparencia. Esto minimiza los efectos que las fallas pueden tener en las tareas comerciales que la aplicación está realizando.

Para manejar los casos en los cuales se ha detectado un fallo, se han manejado la siguiente estrategia, ajustada a las condiciones de la aplicación y a los requisitos de cliente.

Cancelar: Cuando la causa del fallo es poco probable que sea transitoria o que tenga éxito si se reintenta, la operación se cancela y se debe de informar del error. En este sistema, el mensaje fallido pasa a una cola manual donde el personal encargado de gestionar el sistema deberá evaluar las razones de tal fallo en base al error que ha producido.

También se cancelará si el sistema ha intentado realizar la operación mas de tres veces y sigue sin poderlo conseguir.

Reintentar después de una demora: Si el fallo esta relacionado con problemas de conexión, no tiene sentido reintentar al momento. Por ello, pasa a una cola de reintentos autómatica que reenviará los mensajes una vez que se comprueba que el sistema de bases de datos vuelve a estar disponible.

4.1.5. Patrón *Proxy*

El patrón *Proxy* proporciona un objeto intermediario entre el cliente y el objeto a utilizar, que permite configurar ciertas características (como el acceso) sin necesidad de modificar la clase original [14]. No se ha implementado el patrón *Broker*, junto con el patrón *Proxy*, al haber una única base de datos. De esta manera, se ha evitado añadir una capa de complejidad innecesaria, que podría ralentizar el sistema completo.

Este patrón se utilizará realizar las siguientes conexiones:

- *Microsoft Dynamics 365* con las *Azure Function* y viceversa.
- Las *Azure Functions* a *Service Bus* y viceversa.
- Las *Azure Functions* al servidor de bases de datos de Oracle.

4.2. Diagrama de clases

En la siguiente imagen 4.4 se detalla el diagrama de clases de diseño asociado al sistema:

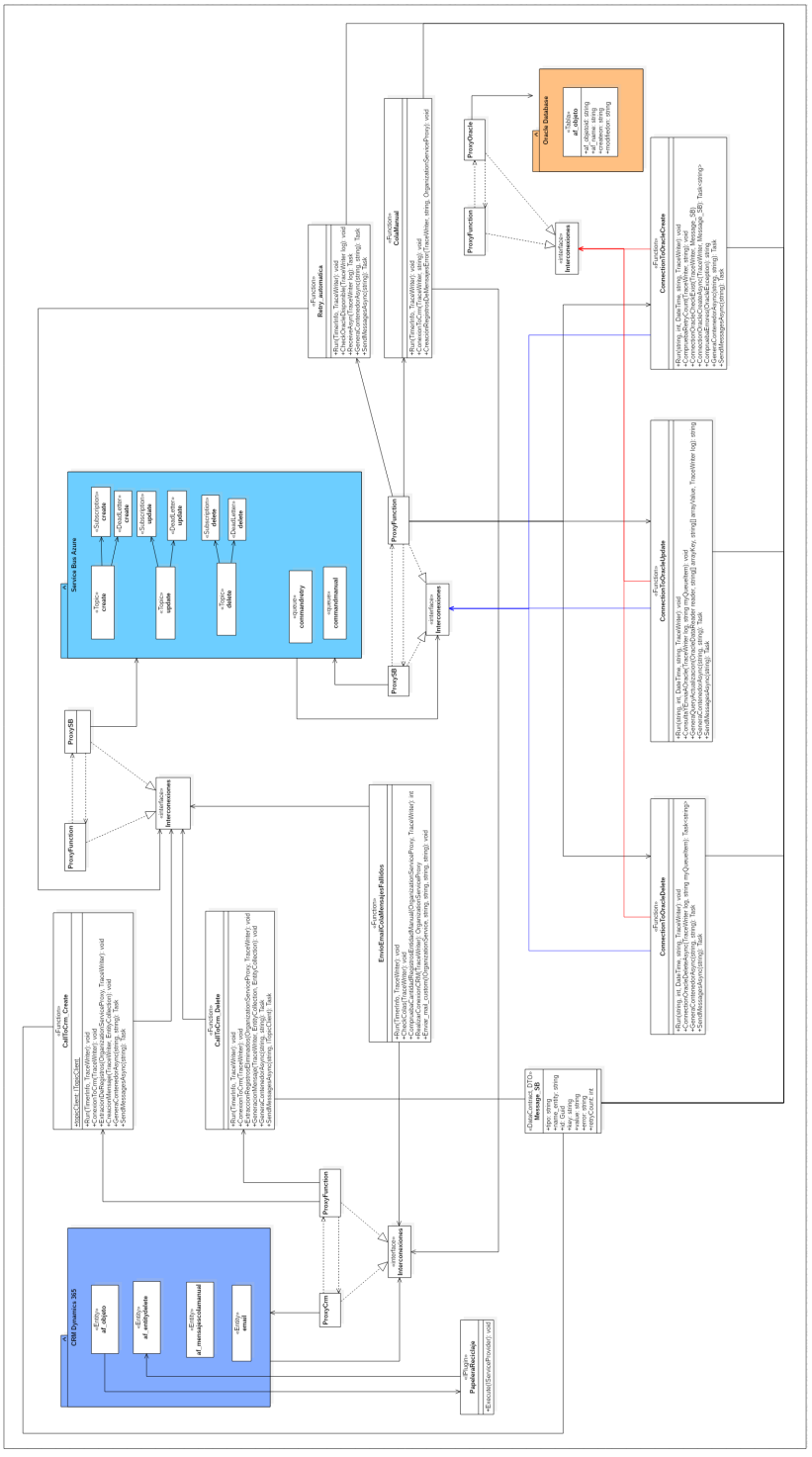

Figura 4.4: Diagrama de clases de diseño

A continuación, se van a detallar las clases que conforman los elementos más importantes del sistema.

4.2.1. Message_SB

La clase de la figura 4.5, estará asociada a todas las *Azure Functions*. Es la clase que marca la estructura que tienen los mensajes que se intercambian a través de *Service Bus* 4.5.

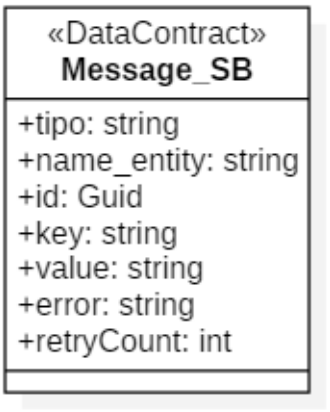

Figura 4.5: Mensaje

4.2.2. CallToCrmCreateUpdate

La *Azure Function* representada en la figura 4.6, es la encargada de extraer los registros que, o bien han sido creados o bien actualizados, dentro del intervalo de tiempo desde el momento en que entra a funcionar la *Azure Function* y un minuto anterior. Una vez que generá un mensaje con los datos del registro extraído, los envía a *Service Bus* .

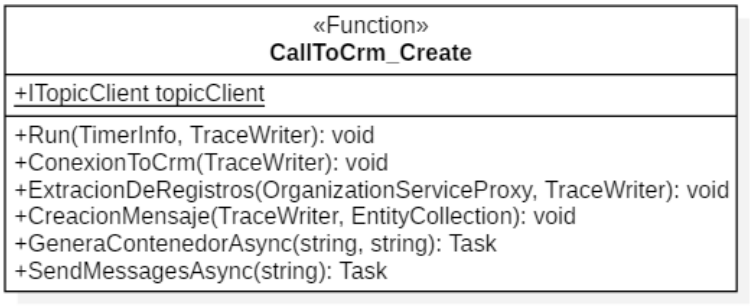

Figura 4.6: Function CallToCrm_Create_Update

ConexionToCrm: Método encargado de realizar la conexión con *Microsoft Dynamics 365*.

ExtracionDeRegistros: Método encargado de realizar la parametrización de la Query y de ejecutarla en *Microsoft Dynamics 365*.

CreacionMensaje: Método encargado de generar mensajes con las información almacenada en los registros extraídos de *Microsoft Dynamics 365*. También se encargará de asignar el *Topic* y la *Subscription* a la cual se agregara el mensaje.

GeneraContenedorAsync: Método encargado de generar el recipiente asíncronamente que enviara el mensaje al *Topic*, con la *Subscription*, de "create" o a la "upate" dependiendo del tipo de mensaje.

SendMessagesAsync: Método encargado de enviar, asíncronamente, el mensaje a la cola de suscripción.

4.2.3. CallToCrmDelete

La *Azure Function* representada en la figura 4.7, es encargada de extraer los registros que han sido eliminados del *Microsoft Dynamics 365*, utilizando para ello la entidad auxiliar donde quedan almacenados los elementos necesarios para proceder a realizar el borrado en la base de datos de Oracle, para enviarlos a *Service Bus*. También elimina los registros de la entidad auxiliar una vez realizado el envío .

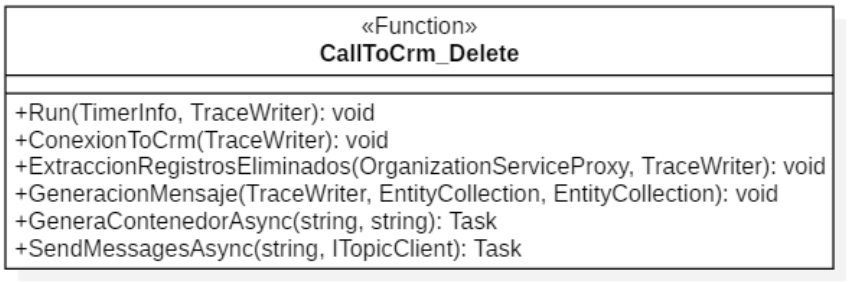

Figura 4.7: Function CallToCrm_Delete

ConexionToCrm: Método encargado de realizar la conexión con *Microsoft Dynamics 365*.

ExtraccionRegistrosEliminados: Método encargado de realizar la parametrización de la Query y de ejecutarla en *Microsoft Dynamics 365*.

GeneracionMensaje: Método encargado de generar mensajes con las información almacenada en los registros extraídos de *Microsoft Dynamics 365*. También se encargara de eliminar el registro de la entidad auxiliar una vez completado el envío a *Service Bus*.

GeneraContenedorAsync: Método encargado de generar el recipiente asíncronamente que enviara el mensaje al *Topic*, con la *Subscription*, de "*delete*".

SendMessagesAsync: Método encargado de enviar, asíncronamente, el mensaje a la cola de suscripción.

4.2.4. ConnectionToOracleCreate

La *Azure Function* representada en la figura 4.8, es la encargada el almacenar en la base de datos de Oracle la información recibida en los mensajes del *Topic* con la *Subscription* "*create*" .

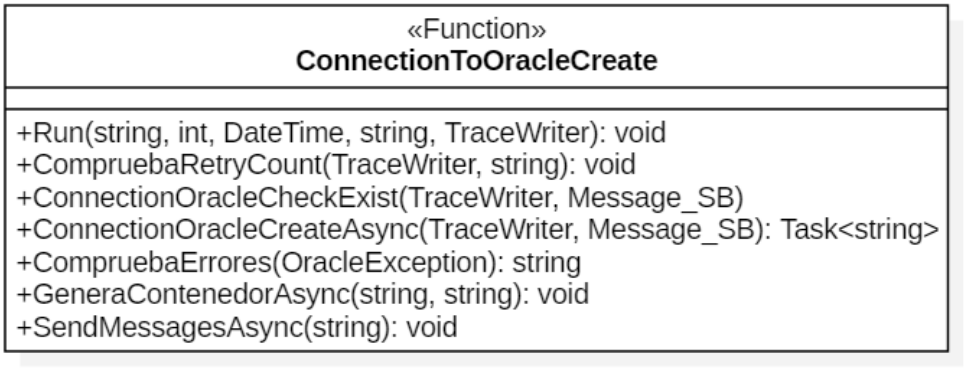

Figura 4.8: Function ConnectoToOracleCreate

CompruebaRetryCount: Método que comprueba si el mensaje ha sido reitentando con anterioridad.

ConnectionOracleCheckExistAsync: Método que evalúa, en el caso de un mensaje que ha sido reintentado, si el registro asociado al mensaje ya ha sido creado en la base de datos de Oracle.

ConnectionOracleCreateAsync: Método encargado de crear un nuevo registro en la base de datos de Oracle utilizando la información recibida dentro del mensaje.

CompruebaErrores: Método encargado de evaluar los errores arrojados por Oracle.

GeneraContenedorAsync: Método encargado de generar el recipiente que enviara los mensajes fallidos o bien a la cola de reintentos manual o bien a la cola de reintentos automática.

SendMessagesAsync: Método encargado de enviar el mensaje a la cola correspondiente.

4.2.5. ConnectionToOracleDelete

La *Azure Function* representada en la figura 4.9, es la encargada el eliminar en la base de datos de Oracle la información recibida en los mensajes de la *Topic* con la *Subscription* de "*delete*"

ConnectionOracleDeleteAsync: Método encargado de, asíncronamente, eliminar los registros asociados a lo mensajes recibidos.

GeneraContenedorAsync: Método encargado de generar el recipiente que enviará los mensajes fallidos o bien a la cola de reintentos manual o bien a la cola de reintentos automática.

SendMessagesAsync: Método encargado de enviar el mensaje a la cola correspondiente.

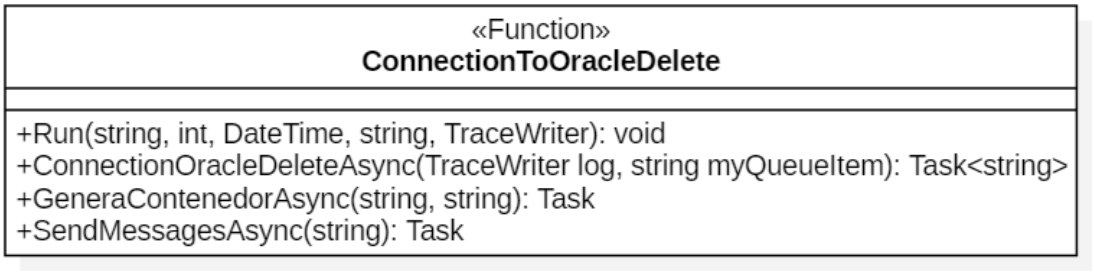

Figura 4.9: Function ConnectoToOracleDelete

4.2.6. ConnectToOracleUpdate

La *Azure Function* representada en la figura 4.10, es la encargada el actualizar en la base de datos de Oracle la información recibida en los mensajes de la *Topic* con la *Subscription* de "*update*".

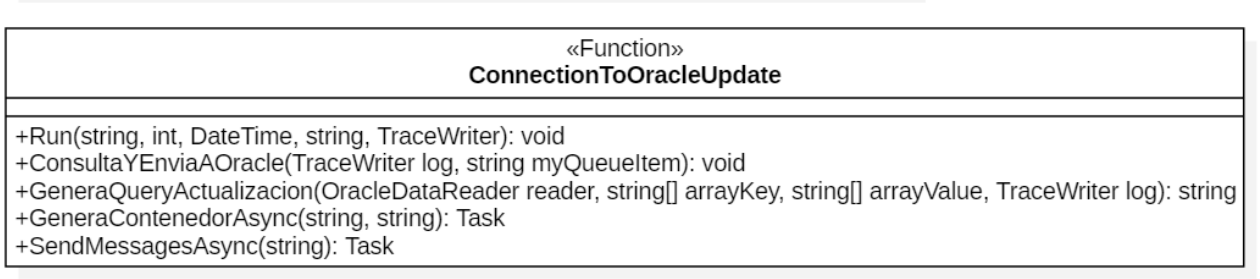

Figura 4.10: Function ConnectoToOracleUpdate

ConsultaYEnviaAOracle: Método encargado de consultar a Oracle la información relativa al mensaje recibido. Usando la información devuelta, se genera una query de actualización y se ejecuta si procede.

GeneraQueryActualizacion: Método que genera la query de actualización en base a la información del mensaje y los datos almacenados en Oracle.

GeneraContenedorAsync: Método encargado de generar el recipiente que enviara los mensajes fallidos o bien a la cola de reintentos manual o bien a la cola de reintentos automática.

SendMessagesAsync: Método encargado de enviar el mensaje a la cola correspondiente.

4.2.7. RetryAutomatica

La *Azure Function* representada en la figura 4.11, es la encargada de manejar la cola de reintentos automática. Una vez detectada la disponibilidad de Oracle reenviará los mensajes a los *Topic* ,con sus *Subscription*, adecuados .

CheckOracleDisponible Método encargado de comprobar la disponibilidad de la base de datos de Oracle.

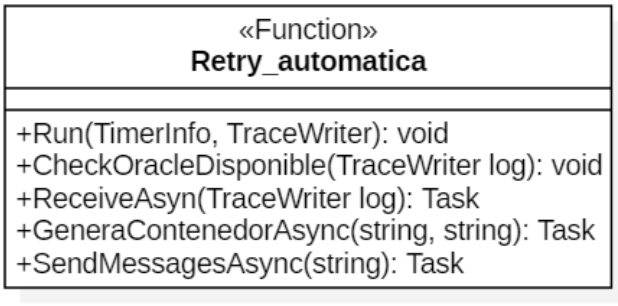

Figura 4.11: Function Retry_automatica

ReceiveAsyn: Método encargado de recibir los mensajes encolados y de reenviarlos al *Topic*, con su respectiva *Subscription*, adecuado.

GeneraContenedorAsync: Método de generar el contenedor que enviará los mensajes fallidos o bien a la cola de reintentos manual o bien a la cola de reintentos automática.

SendMessagesAsync: Método encargado de enviar el mensaje a la cola correspondiente.

4.2.8. ColaManual

La *Azure Function* representada en la figura 4.12, es la encarga de manejar la cola manual de reintentos. Los mensajes almacenados se insertaran el *Microsoft Dynamics 365*, para su posterior evaluación por el equipo de gestión.

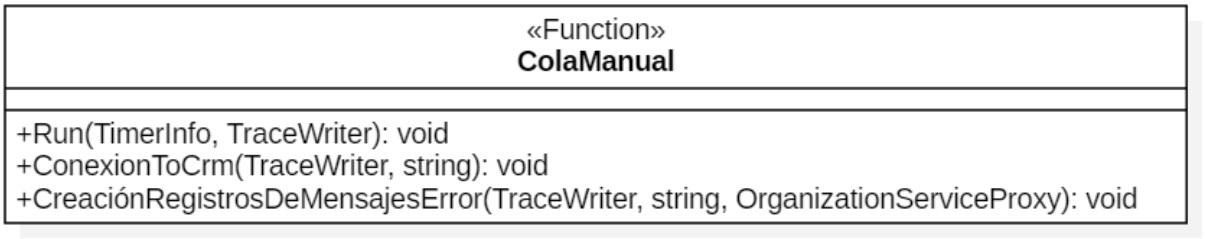

Figura 4.12: Function ColaManual

ConexionToCrm: Método encargado de realizar la conexión con *Microsoft Dynamics 365*.

CreaciónRegistrosDeMensajesError: Método encargado de crear un registro en *Microsoft Dynamics 365*, en base a los mensajes fallidos extraídos de la cola.

4.2.9. RegularizaciónDatosCrm

El programa de consola representado en la figura 4.13, es el encargado de regularizar los datos en Dynamics CRM 365 dentro de un intervalo de tiempo.

ConnectaToCRM: Método encargado de realizar la conexión con *Microsoft Dynamics 365*.

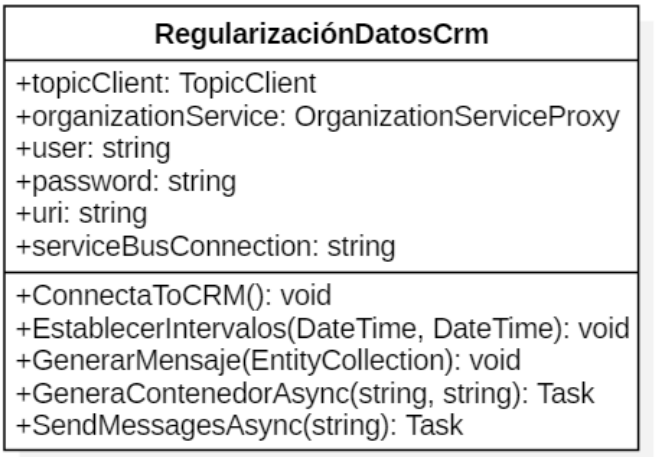

Figura 4.13: Function RegularizaciónDatosCrm

EstablecerIntervalos: Método encargado de recoger el intervalo de tiempo en el cual se realiza la regularización de los datos.

ExtraerRegistrosEnIntervalo: Método encargado de parametrizar la query que realiza la extracción de *Microsoft Dynamics 365*.

GenerarMensaje: Método encargado de generar mensajes en base a los datos extraídos y determinar a que suscripción y topic deben de ser asignados.

GeneraContenedorAsync: Método encargado de generar el recipiente asíncronamente que enviara el mensaje a la cola de suscripción adecuada.

SendMessagesAsync: Método encargado de enviar el mensaje a la cola correspondiente.

4.2.10. EnvioEmailColaMensajesFallidos

La *Azure Function* representada en la figura 4.14, es la encargada de comprobar la cantidad de registros que ha generado la cola manual y el número de mensajes que están almacenados en las colas de mensajes fallidos de los *Topic*. Si superan cierta cantidad de mensajes, enviara un e-mail de aviso.

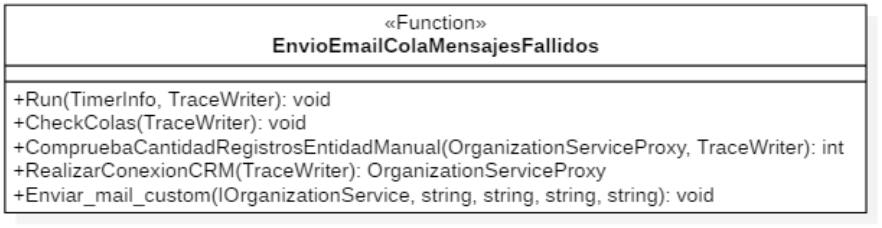

Figura 4.14: Function EnvioEmailColaMensajesFallidos

CheckColas: Método encargado de comprobar la cantidad de mensajes de las colas y si el número de mensajes almacenado supera al permitidido.

CompruebaCantidadRegistrosEntidadManual: Método encargado de contabilizar el número de registros que ha creado la cola manual en base a los mensajes fallidos.

RealizarConexionCRM: Método encargado de crear la conexión*Microsoft Dynamics 365*.

Enviar_mail_custom: Método encargado de crear en *Microsoft Dynamics 365* la entidad email. *Microsoft Dynamics 365* será el encargado de enviar el e-mail.

4.2.11. PapeleraReciclaje

El Plugin representado en la figura 4.15, es el encargado de salvaguardar los datos, que han sido eliminados en *Microsoft Dynamics 365*, que serán necesarios para eliminar, posteriormente, en la base de datos de Oracle. Esta información quedará almacenada en otra entidad.

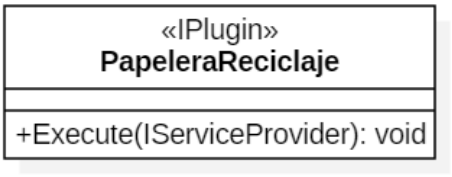

Figura 4.15: Plugin que salvaguarda los datos de los registros eliminados.

Execute: Método encargado de extraer los datos requeridos de la imagen del registro eliminado y, utilizando dicha información, crear un nuevo registro en otra entidad con dicha información.

4.3. Diagramas de Secuencia

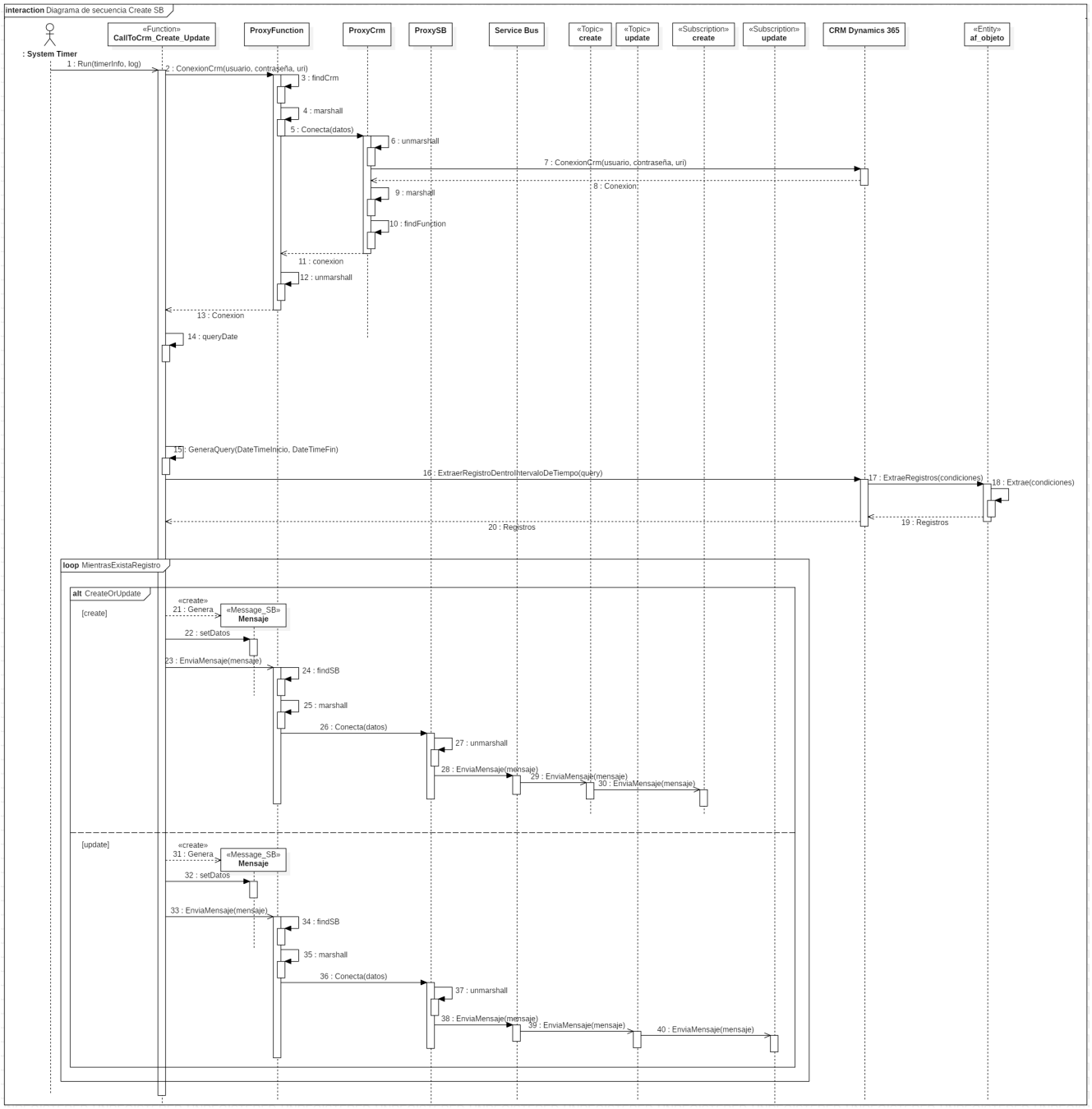

Figura 4.16: Diagrama de diseño que específica el flujo de los mensajes de *Delete* desde *Microsoft Dynamics 365* hasta *Service Bus*.

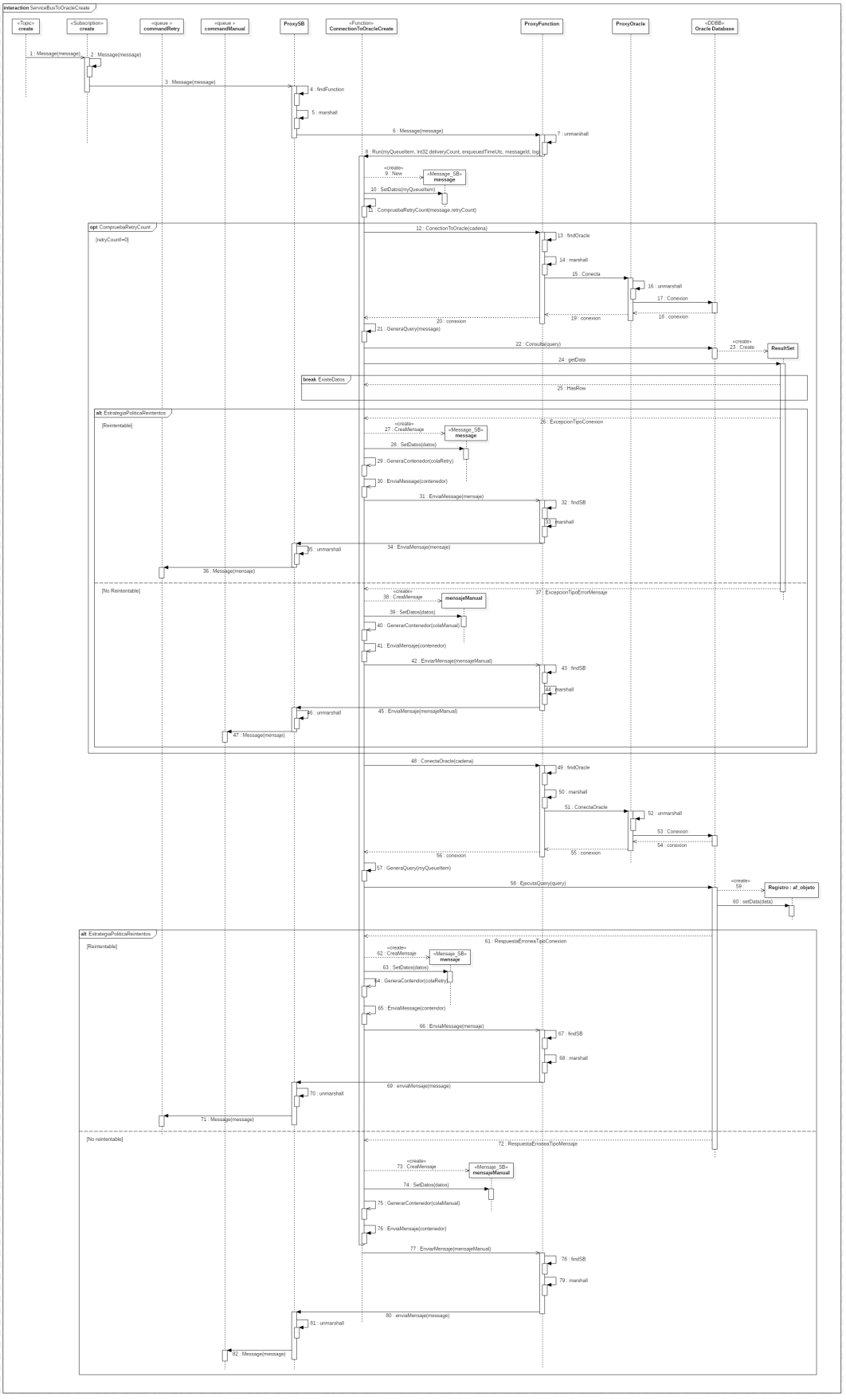

Figura 4.17: Diagrama de diseño que específica el flujo de los mensajes de *Create* desde *Service Bus* hasta la base de datos de Oracle.

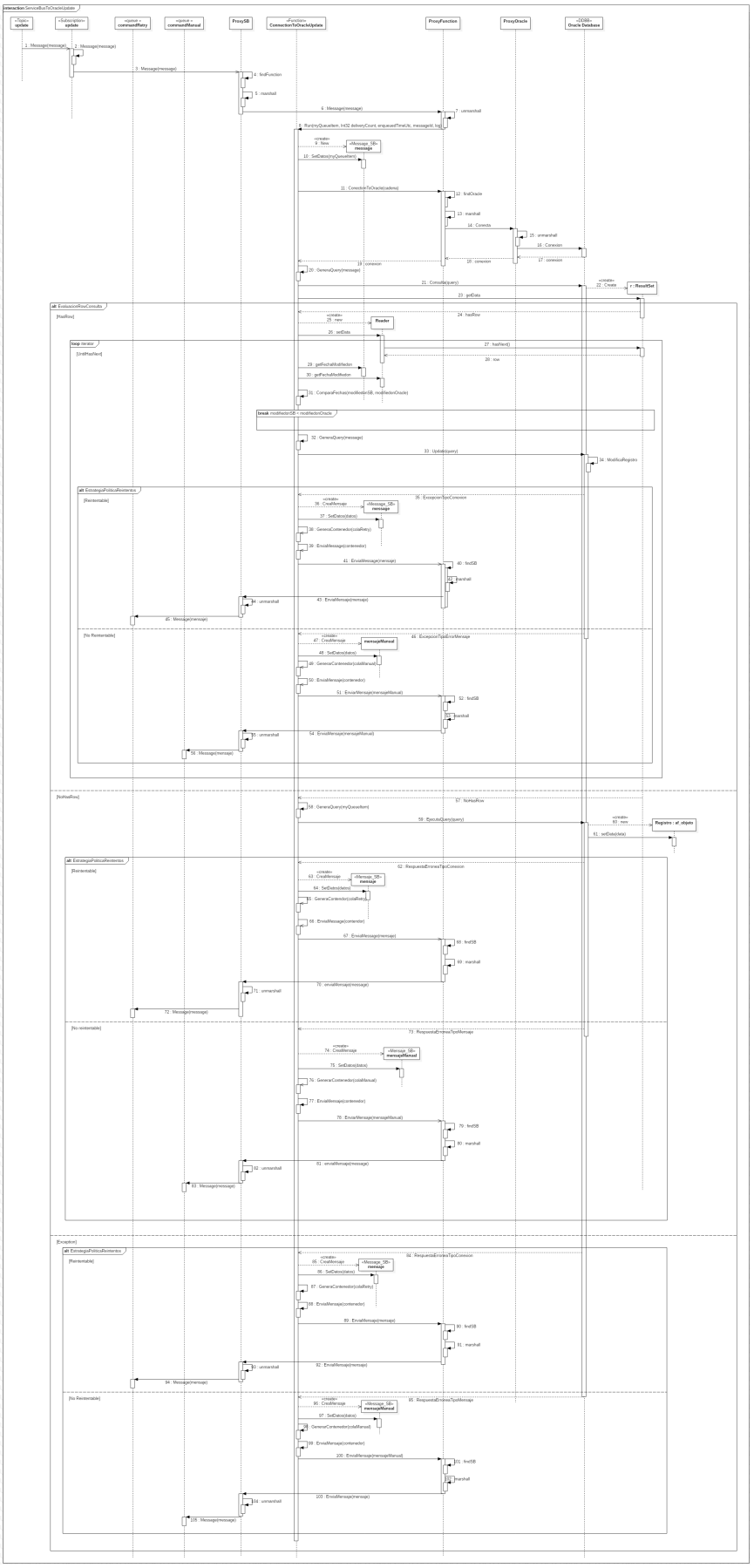

Figura 4.18: Diagrama de diseño que específica el flujo de los mensajes de *Update* desde *Microsoft Dynamics 365* hasta *Service Bus*.

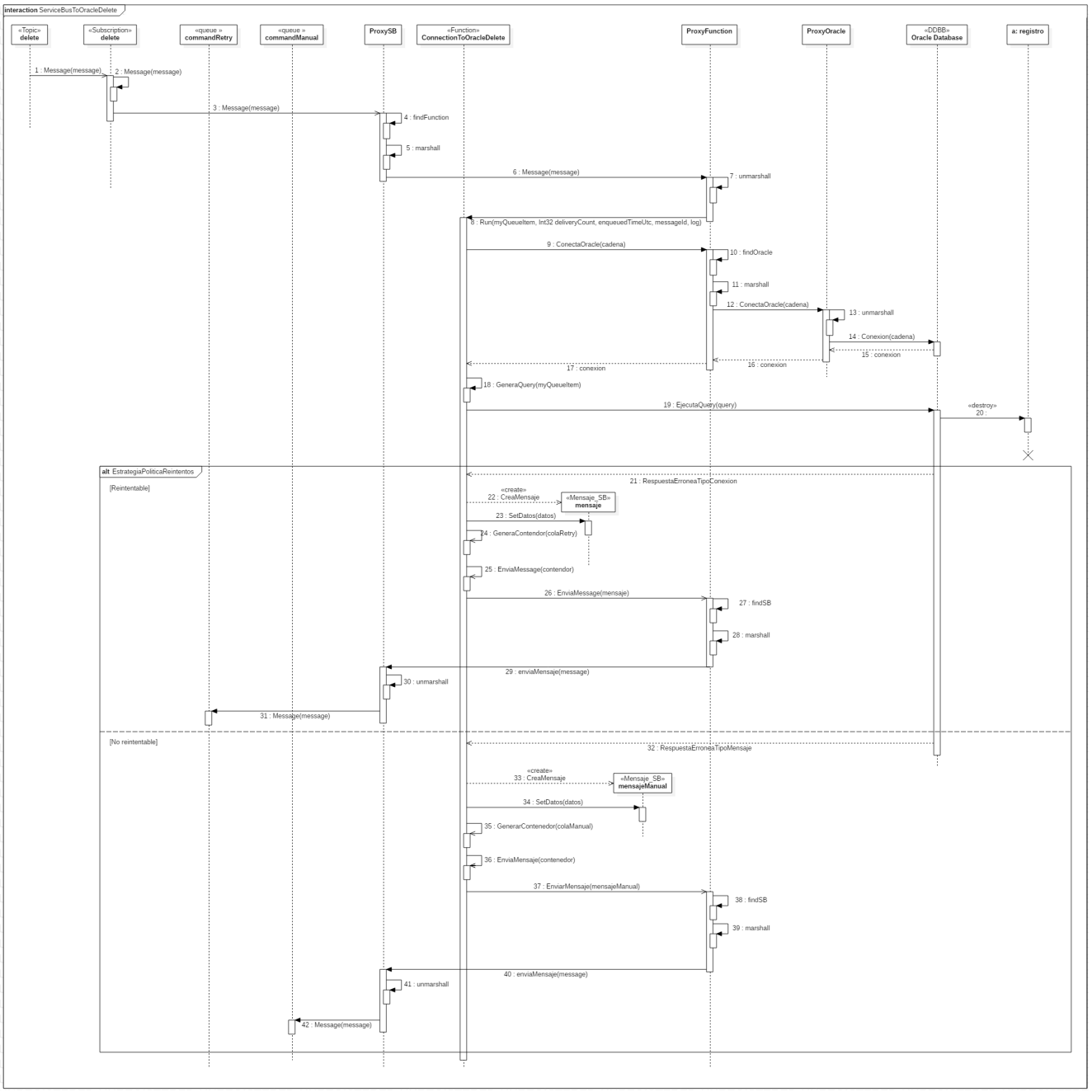

Figura 4.19: Diagrama de diseño que específica el flujo de los mensajes de *Delete* desde *Service Bus* hasta la base de datos de Oracle.

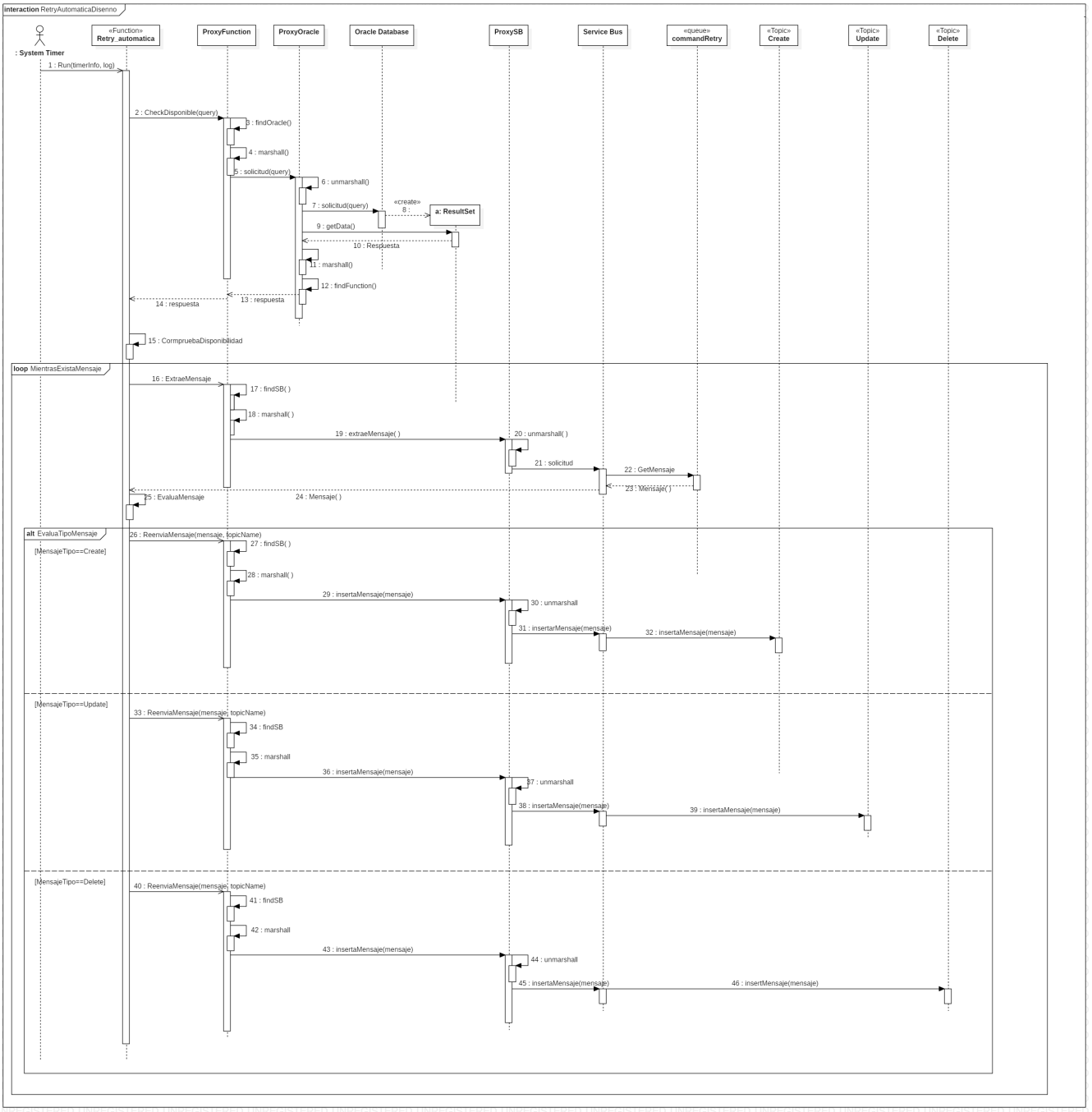

Figura 4.20: Diagrama de diseño de *RetryAutomatica*

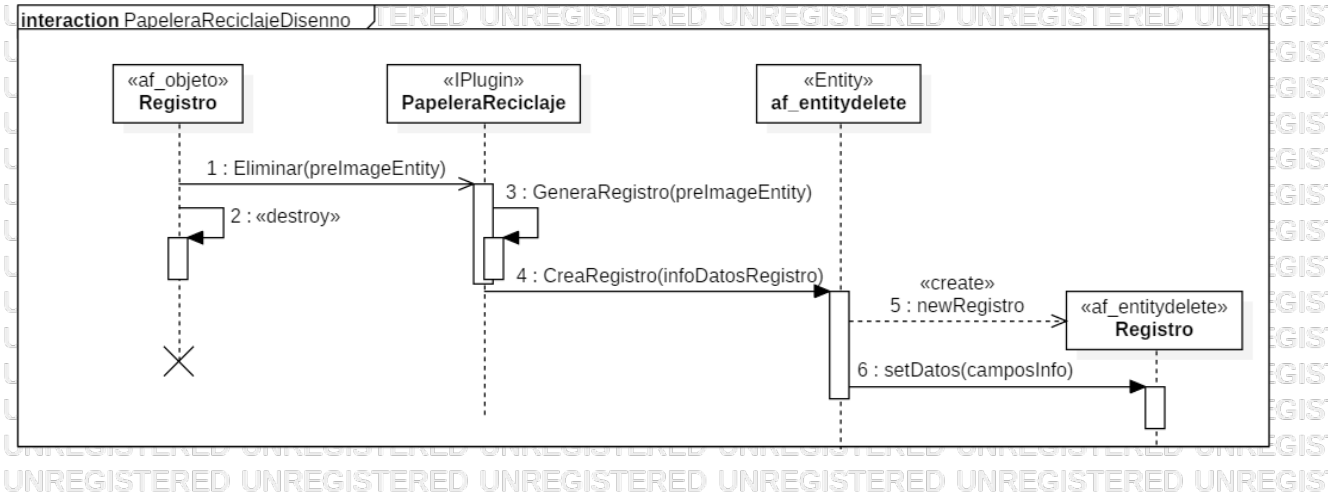

Figura 4.21: Diagrama de diseño de *PapeleraReciclaje*

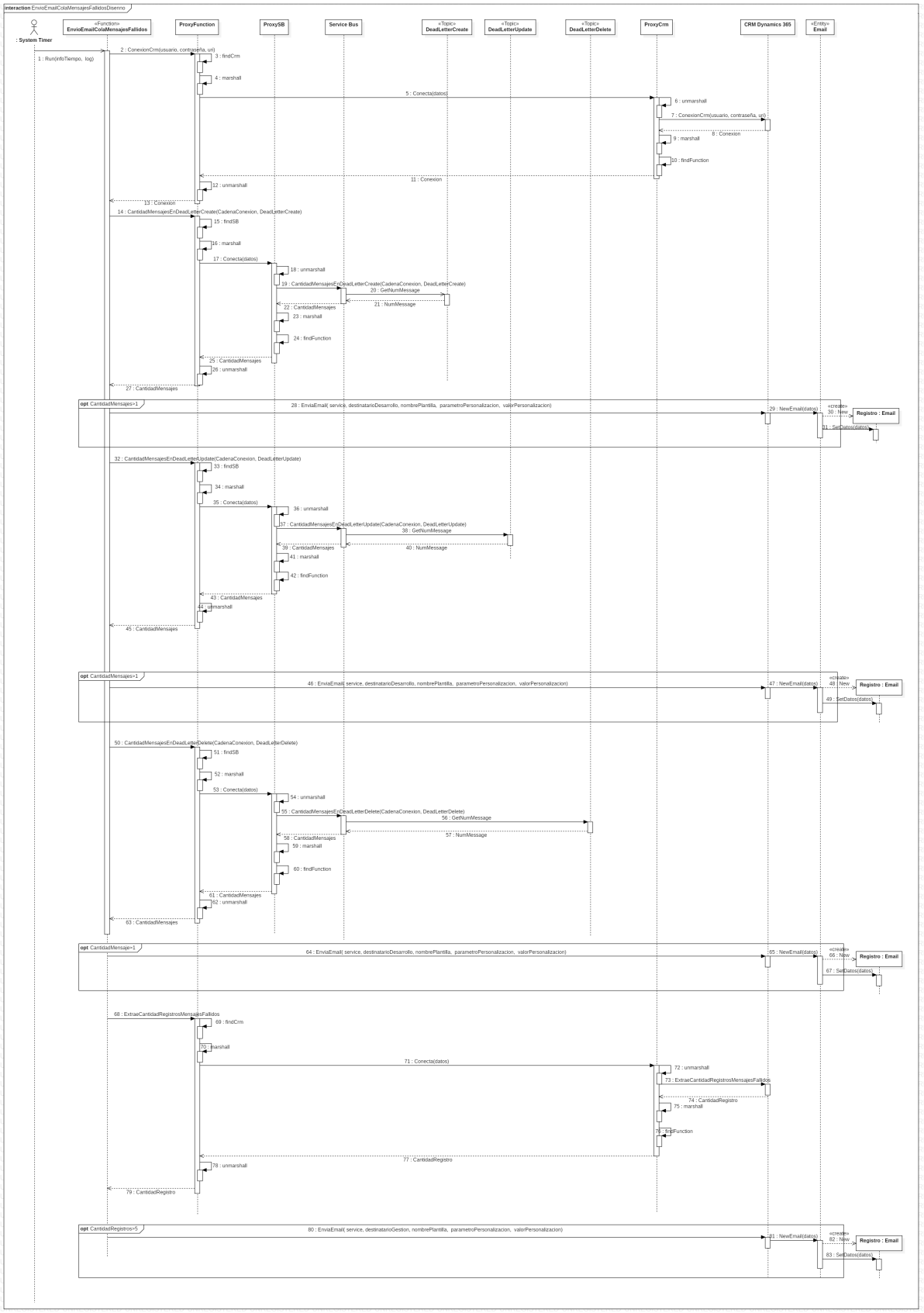

Figura 4.22: Diagrama de diseño de EnvioEmailColaMensajesFallidos

Capítulo 5

Implementación

En la siguiente sección se especifica las etapas por la cuales ha pasado la fase de desarrollo del sistema, justificando de esta manera, las decisiones de diseño que se integran poco a poco en el sistema final ajustándose, en cada etapa, a los requisitos proporcionados por el cliente en la fase de análisis 3 de una manera incremental.

5.0.1. Primera etapa

En esta primera fase se plantea el uso de *Logic App*, que es un servicio en la nube que permite automatizar flujos de trabajo [15]. Usando este servicio, se establece un flujo, a través del cual, cada vez que se crea un nuevo registro en *Microsoft Dynamics 365* la *Logic App* llamará a la *Azure Function* encargada de enviar el registro a *Service Bus*.

Esta idea se descartó, finalmente, para no añadir una capa de complejidad intermedia al sistema no estaba incluida en los requisitos dados por el cliente.

5.0.2. Segunda etapa

En esta etapa se desarrolla una *Azure Function* que realiza los siguiente pasos:

- Se realiza la conexión con el sistema de *Microsoft Dynamics 365*.
- Se parametriza una *query* sencilla en base a los registro que debe de ser extraídos.
- Se ejecuta la *query* y se comprueba que los registros se extraen de manera correcta.
- Utilizando la información extraída, se genera un mensaje sin formato y en texto plano.
- Por último, se envía el mensaje generado a una cola de *Service Bus*.

5.0.3. Tercera etapa

En la tercera etapa, se utiliza el campo "consumido" del registro para extraer los mensajes. Este campo, está a "*false*" cuando la *Azure Function* encargada de extraer los registros no ha enviado la información almacenada en ellos a *Service Bus*. Cuando la *Azure Function* envía el mensaje a *Service Bus*, se actualiza el campo a "*true*", de esta manera, se realiza la distinción entre los mensajes consumidos y los que aun deban ser tratados.

Esta versión, no resultaba óptima por la cantidad de actualizaciones, con las consecuentes conexiones que deben realizarse con *Microsoft Dynamics 365*, que se realizan para mantener la coherencia de los registros. Además de no ser coherente con los requisitos dados por el cliente.

5.0.4. Cuarta etapa

En la cuarta etapa, se utiliza el campo *MODIFIEDON* para hacer la distinción entre los registros consumidos por la *Azure Function* de aquellos que no lo han sido. Se extraen los mensajes que han sido modificados dentro del intervalo de tiempo conformado entre las conexiones de la *Azure Function*, que es de un minuto.

De esta manera, se limita el número de conexiones hacia *Microsoft Dynamics 365*, pues todo el peso de la diferenciación entre los registros se realiza mediante la consulta y no es necesario realizar actualizaciones regulares para mantener la coherencia de datos.

5.0.5. Quinta etapa

En esta etapa, se utiliza el campo *MODIFIEDON*, que ya se utilizaba en la anterior etapa, y el campo *CREATEON* para realizar la diferenciación entre los registros que han sido creados, en el último minuto, de los registros que han sido actualizados dentro del mismo intervalo.

Por otro lado, se establece una estructura de mensaje para el envío de información a *Service Bus*. Se define los campos de la estructura de la siguiente manera:

- Tipo: *Create, Delete* o *Update*
- Name_entity: Nombre de la entidad a partir de la cual se extraen los registros.
- Id: Campo en el cual se guardará la identificación única del registro.
- Key: Se almacena los nombres de los campos del registro.
- Value: Se almacena el valor de los campos del registro.

5.0.6. Sexta etapa

En esta etapa, se extraen los registros que han sido eliminados de *Microsoft Dynamics 365*. Por la forma en que trabaja *Microsoft Dynamics 365*, cuando un registro se elimina no queda rastro alguno del mismo, a excepción de las entidades que tienen activa la característica de auditoría [**auditoria**].

Esta característica, registra los cambios que se realicen en registros de los clientes y el acceso de un usuario para se que pueda revisar la actividad posteriormente. La característica de auditoría está diseñada para cumplir las directivas de auditoría, cumplimiento, seguridad, y gobierno de muchas compañías reguladas. El problema derivado de esta característica, es la cantidad de información que genera, la cual es incremental a medida que el sistema crece, esto lo convierte en una característica poco recomendable para el planteamiento de este TFG. Por estas razones, se descarta.

Descartada dicha opción, se resuelve una solución intermedia. Utilizando el complemento de *Plugin* de *Microsoft Dynamics 365*, que permite ejecutar una lógica concreta cuando se produce un cambio en un registro (además de varias funcionalidades más que no aplican en este contexto). La solución utilizando dicho complemento, es realizar un flujo a partir del cual, cada vez que se elimine un registro de la entidad observada por la *Azure Function*, se guardarán los siguientes campos:

- af_entityname: Donde se almacena el nombre de la entidad.
- af_id_name_key: Nombre del campo en cual se almacena el identificador único del registro.
- af_guid: Valor del identificador único del registro.

que serán necesarios para mantener la coherencia en la base de datos de Oracle, en otra entidad. En dicha entidad, es de donde la *Azure Function* se conecta para extraer la información. Una vez que la información se haya extraído y se haya enviado el mensaje a *Service Bus*, siguiendo el formato del mensaje mencionado en la quinta etapa 5.0.5, el registro extraído se elimina de esta entidad. De esta manera, toda la información asociada al registro original se borrará definitivamente de *Microsoft Dynamics 365*.

5.0.7. Séptima etapa

En esta etapa, se establece el uso de *Topic* y *Subscription* en *Service Bus*. Hasta este momento, los mensajes de creación, actualización y borrado se enviaban todos a la misma cola "*command*" sin realizar ninguna distinción por el tipo de mensaje por la facilidad de implementación que suponía en las primeras etapas. En esta etapa, se modifica esto para enviar los mensajes al *Topic* y *Subscription* relacionados con el tipo de mensaje.

Se decide implementar el uso de una cola por el uso de Temas o Topics por la facilidad con la que se puede escalar un sistema en relación al procesamiento de mensajes en una serie amplia de usuarios y aplicaciones.

Por otra parte, se instala y configura una base de datos de Oracle en una máquina virtual. Esta maquina hará las veces de servidor, puesto que la realización de un servidor completo, alcanzado por la *Cloud*, esta fuera del alcance de este trabajo.

5.0.8. Octava etapa

En esta etapa, se desarrolla una *Azure Function* que permita crear registros en la base de datos de Oracle en base a la información extraída de *Service Bus*. Para ello se utiliza un tipo de *Azure Function* llamada *ServiceBusTrigger*, que permite asociar dicha *Azure Function* a un *Topic* y una *Subscription* concretos. De esta manera, cada vez que la *Azure Function* detecte un nuevo mensaje en el *Topic*, con su *Subscription* asociada, empezará a funcionar.

Se estable la conexión con Oracle y, utilizando la información decodificada del mensaje, se crea una *query* se creación en Oracle. Todo esto realizado de forma síncrona.

5.0.9. Novena etapa

En esta etapa, se desarrolla una *Azure Function* que, siguiendo las misma condiciones expuestas en la octava etapa 5.0.8, se genera una *query* y se ejecuta, en base a los datos extraídos del *Topic* y de la *Subscription*, que permite borrar el registro asociado en la base de datos de Oracle.

5.0.10. Décima etapa

En la décima etapa, se desarrolla la *Azure Function* que actualiza los registros almacenados en la base de datos de Oracle en función de la información recibida en el mensaje. Para ello, se realiza una *query* de consulta, a partir de la cual, se recupera la información almacenada en la base de datos de Oracle sobre el registro asociado al mensaje recibido.

Utilizando la información devuelta, se compara con la información recibida desde el *Topic* para generar una nueva *query* en base a la información que ha sido modificada. Si el registro aún no ha sido creado, puede haberse actualizado el registro en *Microsoft Dynamics 365* antes de ser consumido por la *Azure Function*, el registro asociado a la información almacenada en el mensaje se crea en la base de datos de Oracle.

También puede darse el caso en el cual no hay registro almacenado en la base de datos de Oracle asociado al mensaje recibido. Este caso puede producir debido a que el registro almacenado en *Microsoft Dynamics 365* fue actualizado dentro el intervalo de tiempos entre las *Azure Functions* que consumen estos datos entren en funcionamiento. En este caso, el registro se crea en la base de datos de Oracle.

5.0.11. Undécima etapa

En esta fase, las inserciones de comandos a Oracle, utilizadas en la *Azure Function* de crear registros y en la *Azure Function* de actualizar registros se convierten en asíncronas para aumentar la eficiencia del procesamiento. Además, se filtran las excepciones arrojadas por la base de datos de Oracle para mejorar el entendimiento del funcionamiento del sistema y de los errores arrojados por la base de datos de Oracle.

5.0.12. Duodécima etapa

En esta etapa, se crea una *Azure Function* que accede a la cola de *DeadLetter* o cola de mensajes fallidos del *Topic*. La finalidad de la cola de mensajes fallidos es mantener los mensajes que no se pueden entregar a ningún destinatario o los mensajes que no se pudieron procesar. Después, los mensajes se quitan de la cola de mensajes fallidos y se inspeccionan. Una aplicación podría, con ayuda de un operador, corregir los problemas y volver a enviar el mensaje, registrar el hecho de que se produjo un error y llevar a cabo medidas correctivas.

Desde el punto de vista de la *API* y el protocolo, la cola de mensajes fallidos es muy similar a cualquier otra cola, salvo que los mensajes solo se pueden enviar a través de la operación de mensajes fallidos de la entidad principal. Además, no se observa el período de vida, y no puede

tratar como fallido un mensaje desde una cola de mensajes fallidos. La cola de mensajes fallidos es totalmente compatible con las operaciones transaccionales y de entrega de bloqueo de información [**deadletter**].

Los mensajes permanecen en la cola de mensajes fallidos hasta que se recuperan explícitamente, de ahí la necesidad de realizar una *Azure Function* encargada de extraerlos en esta etapa para poder realizar pruebas en base a ella más realistas aunque no sea parte de los requisitos de cliente.

5.0.13. Decimotercera etapa

En esta etapa, se modifica la estructura del mensaje que se envía a través de *Service Bus* añadiendo los siguientes campos:

- error: Donde quedará almacenado el error, si se produce, al intentar ejecutar el comando asociado a dicho mensaje en la base de datos de Oracle.
- retryCount: Donde se almacenará el número de veces que se ha intentado procesar el mensaje, si procede.

Estos nuevos campos se añaden debido a la necesidad de realizar un control sobre la cola de reintentos que se va a implementar próximamente. En la fase de análisis, durante las conversaciones con cliente, se planteó en utilizar la cola de *DeadLetter* para este cometido, pero se descartó por lo motivos que se explican a continuación:

- La necesidad de utilizar un modelo de mensaje que Microsoft considera en desuso, y no recomienda utilizar, para poder enviar mensajes explícitamente a esta cola.
- La robustez del sistema se basa en la no pérdida de mensajes. Si el sistema falla cuando se extraen mensajes de la *DeadLetter* el mensaje puede perderse.
- La necesidad de realizar una diferenciación entre los tres tipos de fallos detectados durante las pruebas:
  - Los fallos producidos por la imposibilidad de de conectar con la base de datos de Oracle.
  - Los fallos asociados al mensaje por lo cual son rechazados por la base de datos de Oracle.
  - Los fallos inesperados relacionados directamente con las *Azure Function* y con *Azure*.

  Esta diferenciación da como resultado, un sistema más sólido y aislado. Al tener una fuerte diferenciación entre los errores, el tratamiento de los mismos puede ser llevado a cabo por distintos equipos, los cuales estarán mas especializados.

Por la razones anteriormente mencionadas, la nueva estructura de colas queda, ajustándose a los requisitos de cliente, así (a mayores de las colas asociadas a los *Topic* y *Subscription* mencionados en las anteriores iteraciones):

- Cola de reintentos automática: Donde quedan almacenados los mensajes que no han podido se manejados por la base de datos de Oracle debido a problemas de conexión con la misma. Esta cola sera manejada por una *Azure Function*, que se desarrollará en las siguientes iteraciones.

- Cola de reintentos manual: Será la encargada de almacenar los mensajes que tienen algún tipo de error por asociado que hace que no puedan ser manejados la base de datos de Oracle.
- *DeadLetter*(*Create, Update* y *Delete*): Hay una por cada *Topic* y será la última salvaguarda de los mensajes. En caso de haber fallos inesperados de manera reiterada por parte de las *Azure Functions* o de *Azure* los mensajes serán almacenado, automáticamente por *Service Bus*, en esta cola.

5.0.14. Decimocuarta etapa

En esta etapa, se modifican las *Azure Functions* en base da los campos creados en la anterior etapa 5.0.12.

Se comienza por las *Azure Functions* encargadas de extraer los mensajes desde *Microsoft Dynamics 365*. Se crea el campo error vacío y se inicializa el contador de *retryCount* a cero.

Por otro lado, las *Azure Functions* que manejan las conexiones con la base de datos de Oracle también sufren cambios. Cuando se produce una excepción, se almacenará la información asociada a dicha excepción, en base al filtro realizado en la undécima etapa 5.0.11, en el campo de error. También se aumenta el contador de *retryCount* en uno, se evalúa en base al error que se ha producido a que cola debe ser enviado el mensaje y se reenvía a dicha cola.

5.0.15. Decimoquinta etapa

En esta etapa, se genera la *Azure Function* encargada de gestionar la cola de reenvío automático. Los pasos que sigue son los siguientes:

- La *Azure Function* realiza una consulta genérica a la base de datos de Oracle. Si dicha consulta no es respondida por Oracle, no se hará nada mas puesto que no tendría sentido reencolar mensajes que no podrán ser procesados.
- En el caso de que sí responda correctamente a la petición, se extraen los mensajes almacenados en la cola "*retryAutomatica*".
- Se evalúan los mensajes para determinar a que *Topic* y a que *Subscription* pertenecen.
- Se envían todos los mensajes que estaban almacenados en la cola al momento de iniciar la evaluación. Aquellos que lleguen después no se evalúan puesto que se considera que, si están siendo almacenados en la cola de "*retryAutomatica*" la base de datos de Oracle vuelve a no estar disponible.

Esta consulta de disponibilidad se realizará cada quince minutos, de acuerdo a los requisitos proporcionados por el cliente.

5.0.16. Decimosexta etapa

En esta fase, se modificarán las *Azure Functions* encargadas de conectar a la base de datos de Oracle para ajustar las condicione en base a la funcionalidad de reintentar los mensajes.

A la *Azure Function* encargada de gestionar los mensajes asociados a registros de nueva creación, se le añade la comprobación de mensaje reintentado, si este es el caso, es necesario consultar a la base de datos de Oracle si dicho registro ya ha sido creado. Esto puede pasar en el caso de que el registro original haya sido actualizado en el lapso de tiempo que se produce cuando el que mensaje de nueva creación se encola en la cola de reintentos automática. Si el mensaje de actualización ha llegado antes que el mensaje de creación a la base de datos de Oracle, el registro ya esta creado con los datos más recientes, por tanto, el mensaje de creación debe desecharse.

Por otro lado, la *Azure Function* encargada de manejar los mensajes de actualización también se modifica. Se añade una condición, a partir de la cual, el mensaje solo se insertará si su campo de *MODIFIEDON* tiene una fecha posterior a la fecha almacenada en el mismo campo de la base de datos de Oracle.

La *Azure Function* encargada de manejar los mensajes de eliminación no necesitara ninguna modificación puesto que no se vera afectada por el retardo de un mensaje de ese tipo.

5.0.17. Decimoséptima iteración

En esta etapa, se crea una nueva *Azure Function*. Esta *Azure Function* será la encargada de enviar e-mail en las condiciones que se detallarán a continuación. Esta *Azure Function* se ejecutará, automáticamente, cada quince minutos.

Se enviará un e-mail al equipo de gestión en el caso de que haya más de cinco mensajes almacenados en la cola manual. Como en esta cola se almacenan mensajes, cuyo contenido es erróneo, sera necesario que una persona los evalúe manualmente para poder concretar donde está el fallo. Esta *Azure Function* enviará el aviso al personal que se encargará de ello facilitando las tareas de gestión.

Por otro lado, se envía un e-mail al equipo de desarrollo en el caso de que haya un mensaje en cualquiera de las colas de *DeadLetter* asociadas a los *Topic*. En un flujo normal, no debería haber mensajes en estas colas puesto que, supondría, que han sido reintentados diez veces automáticamente por *Service Bus* teniendo fallos inesperados en cada una de esas ocasiones. Si estas condiciones se están produciendo, el equipo de desarrollo debe evaluar que circunstancias están provocando el comportamiento anormal del sistema.

5.0.18. Decimoctava etapa

En esta etapa, se realiza un programa de consola para regularizar los datos de *Microsoft Dynamics 365*. La necesidad de este programa, se produce en dos ocasiones.

La primera vez se da justo antes de poner en funcionamiento el sistema. Las *Azure Functions* encargadas de extraer los registros solo extraerán los susodichos registros que hayan sido modificados en el último minuto. Por tanto, el resto de registros no se enviarían a la base de datos de Oracle.

La segunda ocasión es similar a la primera. En el caso de que el sistema esté un intervalo de tiempo sin funcionar, por la razón que sea, hará que una cierta cantidad de registros no sean

enviados a la base de datos de Oracle.

Para solucionar esto, se ejecuta el programa de consola pasándole como argumentos el intervalo temporal en el cual es necesario enviar los registros que han sido creados o actualizados. No será necesario regularizar los registros eliminados puesto que la *Azure Function* que maneja estos mensajes los extrae de una entidad con unas condiciones especiales mencionadas en la sexta etapa 5.0.6.

5.0.19. Decimonovena etapa

En esta etapa, se crea una nueva *Azure Function* encargada de manejar los mensajes almacenados en la cola manual. Como los mensajes almacenados en esta cola deben de ser tratados manualmente, la información de dichos mensajes se almacenará en *Microsoft Dynamics 365* en una entidad llamada "Mensaje cola manual", que comparte la estructura del mensaje, definida en la quinta etapa 5.0.5 y en la decimotercera etapa 5.0.13.

De esta manera, se pretende facilitar el análisis del mensaje al equipo encargado de esta tarea.

5.0.20. Vigésima etapa

En esta etapa, se plantea mover la base de datos de Oracle de una máquina en local a la versión *Cloud* de Oracle. Después de varias pruebas, que incluyó contacto con el servicio técnico de la empresa Oracle, esta opción se descarta. Después de plantear esta opción a los clientes, en el contexto del TFG son los tutores, se decide que es mejor mantener la versión de base de datos de Oracle en local tal y como estaba definido en los requisitos 3.9 proporcionados durante la fase análisis .

Todos los cambios realizados en esta etapa se descartan.

5.0.21. Vigésimo primera etapa

En esta etapa, se modifica la *Azure Function* que envía los e-mail a los distintos equipos. En vez de comprobar la cola manual, ahora se comprueba la cantidad de mensajes almacenados en la entidad "Mensaje cola manual", que es donde se almacenan los mensajes fallidos. Cuando haya más de cinco registros en dicha entidad, se envía el e-mail.

Se considera que el personal encargado de manejar los mensajes fallidos, los borrará de dicha entidad una vez que se han corregido.

Al finalizar esta etapa se puede dar por concluida la fase de desarrollo al haber cumplimentado los requisitos proporcionados por el cliente.

Capítulo 6

Verificación

En este capitulo, se realizan las pruebas pertinentes al sistema. La pruebas se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren cumpliendo, de esta manera, los requisitos proporcionados por el cliente 3.9 y confirmando la calidad del software desarrollado.

6.1. Pruebas de caja negra

En esta sección se incluirán las pruebas de caja negra que se han realizado para comprobar el correcto funcionamiento del sistema desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

6.1.1. Pruebas de caja negra CallToCrmCreateUpdate

Prueba de caja negra 001	
Descripción	Se comprueba que los registros creados de añaden al *Topic* "*Create*" con la *Subscription* "*Create*"
Entrada	Registro de nueva creación.
Resultado esperado	El registro se añade correctamente.
Resultado de la prueba	Correcto

Tabla 6.1: Prueba de caja negra 001

Prueba de caja negra 002	
Descripción	Se comprueba que los registros actualizados de añaden al *Topic* "*Update*" con la *Subscription* "*Update*"
Entrada	Registro actualizado.
Resultado esperado	El registro se añade correctamente.
Resultado de la prueba	Correcto

Tabla 6.2: Prueba de caja negra 002

6.1.2. Pruebas de caja negra CallToCrmDelete

Prueba de caja negra 003	
Descripción	Se comprueba que los registros eliminados de añaden al *Topic "Delete"* con la *Subscription "Delete"*.
Entrada	Registro eliminado.
Resultado esperado	El registro se añade correctamente.
Resultado de la prueba	Correcto

Tabla 6.3: Prueba de caja negra 003

Prueba de caja negra 004	
Descripción	Se comprueba que el registro asociado al mensaje que se acaba de enviar a *Service Bus*, ha sido eliminado correctamente de *Microsotf Dynamics 365*.
Entrada	Registro con la id: 6c44efb1-f6a1-4271-894a-9240de45ad4b.
Resultado esperado	El registro con la id: 6c44efb1-f6a1-4271-894a-9240de45ad4b ya no está en la entidad.
Resultado de la prueba	Correcto

Tabla 6.4: Prueba de caja negra 004

6.1.3. Pruebas de caja negra ConnectionToOracleCreate

Prueba de caja negra 005	
Descripción	Se comprueba que los registros creados almacenados al *Topic* "*Create*" con la *Subscription* "*Create*" se pueden extraer de la cola.
Entrada	Conexión a la cola de Service Bus.
Resultado esperado	El registro se extrae correctamente.
Resultado de la prueba	Correcto

Tabla 6.5: Prueba de caja negra 005

Prueba de caja negra 006	
Descripción	El comando de creación de un registro funciona correctamente en la base de datos de Oracle.
Entrada	INSERT INTO af_objeto (af_objetoid,af_name,createdon,modifiedon) values ('83b17e23-0a8d-e911-a971-000d3ab5a0d7','hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:03:53')
Resultado esperado	La información proporcionada por el comando queda reflejada en Oracle.
Resultado de la prueba	Correcto.

Tabla 6.6: Prueba de caja negra 006

Prueba de caja negra 007	
Descripción	Si el comando de creación falla por error relacionado con la conexión hacia la base de datos.
Entrada	INSERT INTO af_objeto (af_objetoid,af_name,createdon,modifiedon) values ('83b17e23-0a8d-e911-a971-000d3ab5a0d7','hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:03:53')
Resultado esperado	El mensaje sera enviado a la cola de reintentos automática.
Resultado de la prueba	Correcto.

Tabla 6.7: Prueba de caja negra 007

Prueba de caja negra 008	
Descripción	Si el comando de creación falla por error relacionado relacionado con el mensaje.
Entrada	INSERT INTO af_objeto (af_objetoid,af_name,createdon,modifiedon) values ('83b17e23-0a8d-e911-a971','hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:03:53')
Resultado esperado	El mensaje sera enviado a la cola de reintentos manual.
Resultado de la prueba	Correcto.

Tabla 6.8: Prueba de caja negra 008

Prueba de caja negra 009	
Descripción	Si el comando falla y el mensaje se ha intentado entregar tres veces pasa a la cola manual.
Entrada	{"tipo":"Create","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-000d3ab5a0d7","key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7',Úpdate hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:03:53"','error":"","retryCount":3}
Resultado esperado	El mensaje sera enviado a la cola de reintentos manual.
Resultado de la prueba	Correcto.

Tabla 6.9: Prueba de caja negra 009

6.1.4. Pruebas de caja negra ConnectToOracleUpdate

Prueba de caja negra 010	
Descripción	Se comprueba que los registros actualizados almacenados en la al *Topic "Update"* con la *Subscription "Update"*se pueden extraer de la cola.
Entrada	Conexión a la cola de Service Bus.
Resultado esperado	El registro se extrae correctamente.
Resultado de la prueba	Correcto

Tabla 6.10: Prueba de caja negra 010

Prueba de caja negra 011	
Descripción	El comando de consulta a la base de datos devuelve el registro relacionado con el mensaje de actualización
Entrada	select * from af_objeto where af_objetoid='83b17e23-0a8d-e911-a971-000d3ab5a0d7';
Resultado esperado	af_objetoid:83b17e23-0a8d-e911-a971-000d3ab5a0d7 af_name:Update createdon: 12/06/2019 12:03:53 modifiedon:12/06/2019 15:03:53
Resultado de la prueba	Correcto.

Tabla 6.11: Prueba de caja negra 011

Prueba de caja negra 012		
Descripción	En base a la información devuelta por Oracle y a la información almacenada en el registro se creara un comando de actualización coherente a los datos que han sido actualizados.	
Entrada	Mensaje	{"tipo":"Update","name_entity":"af_objeto", "id":"83b17e23-0a8d-e911-a971-111d3ab5a0d7", "key":"af_objetoid, af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7', Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:23:53"', "error":"","retryCount":1}
	Registro Oracle	af_objetoid:83b17e23-0a8d-e911-a971-000d3ab5a0d7 af_name: update createdon: 12/06/2019 12:03:53 modifiedon: 12/06/2019 12:03:53
Resultado esperado	En Oracle: af_objetoid:83b17e23-0a8d-e911-a971-000d3ab5a0d7 af_name: Update dsfdfff' createdon: 12/06/2019 12:03:53 modifiedon: 12/06/2019 12:03:53	
Resultado de la prueba	Correcto.	

Tabla 6.12: Prueba de caja negra 012

Prueba de caja negra 013		
Descripción	Si el registro almacenado en Oracle tiene una fecha almacenada en *MODIFIEDON* posterior a la que esta guardada en el mensaje dicho mensaje se desechará. La información más reciente es la que debe prevalecer.	
Entrada	Mensaje	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-111d3ab5a0d7", "key":"af_objetoid,af_name,createdon,modifiedon","value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7',Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:03:53"','error":"","retryCount":1}
	Registro Oracle	af_objetoid:83b17e23-0a8d-e911-a971-000d3ab5a0d7 af_name: update createdon: 12/06/2019 12:03:53 modifiedon: 12/06/2019 12:20:53
Resultado esperado	El mensaje se desechará.	
Resultado de la prueba	Correcto.	

Tabla 6.13: Prueba de caja negra 013

Prueba de caja negra 014	
Descripción	Los mensajes de creación que den error al intentar crearlos en la base de datos de Oracle, debido a un mensaje incorrecto se enviara a la cola de reintentos manual.
Entrada	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971", "key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7',Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:23:53"','error":"","retryCount":0}
Resultado esperado	Identificador no válido. El mensaje se reenvía la cola de reintentos manual.
Resultado de la prueba	Fallo. No se añadían correctamente al haber un error ortográfico en la cadena de conexión.

Tabla 6.14: Prueba de caja negra 014

Prueba de caja negra 015	
Descripción	Los mensajes de creación que den error al intentar crearlos en la base de datos de Oracle debido a un problema relacionado con la conexión se envían a la cola de reintentos automática.
Entrada	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-111d3ab5a0d7", "key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7', Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:23:53"','error":"","retryCount":0}
Resultado esperado	No se puede conectar con Oracle, *Timeout*. El mensaje se reenvía la cola de reintentos automática.
Resultado de la prueba	Fallo. No se añadían correctamente al haber un error ortográfico en la cadena de conexión.

Tabla 6.15: Prueba de caja negra 015

Prueba de caja negra 016	
Descripción	Los mensajes de creación que dan error al intentar crearlos en la base de datos de Oracle y ya se ha intentado entregarlos tres veces pasan a cola manual.
Entrada	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-111d3ab5a0d7","key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7',Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:23:53"','error":"","retryCount":3}
Resultado esperado	El mensaje se reenvía la cola de reintentos manual..
Resultado de la prueba	Correcto.

Tabla 6.16: Prueba de caja negra 016

6.1.5. Pruebas de caja negra ConnectionToOracleDelete

Prueba de caja negra 017	
Descripción	Se comprueba que los registros eliminados almacenados en el Tema "Delete" con la suscripción "Delete" pueden extraer de la cola.
Entrada	Conexión a la cola de Service Bus.
Resultado esperado	El registro se extrae correctamente.
Resultado de la prueba	Correcto

Tabla 6.17: Prueba de caja negra 017

Prueba de caja negra 018	
Descripción	Al intentar ejecutar el comando de eliminación Oracle informa de que se ha producido un error relacionado con el formato del mensaje. Dicho mensaje será reenviado a la cola manual.
Entrada	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-111d3ab5a0d7","key":"af_-objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7',Úpdate dsfdfff','12/06/2019 12:03:53', '12/06/2019 12:23:53"',"error":"","retryCount":0}
Resultado esperado	No se puede conectar con Oracle, Timeout. El mensaje se ha enviado a la cola automática..
Resultado de la prueba	Incorrecto. Las condiciones no estaban correctamente definidas.

Tabla 6.18: Prueba de caja negra 018

Prueba de caja negra 019	
Descripción	Al intentar ejecutar el comando de eliminación Oracle informa de que se ha producido un error relacionado con el conexión a Oracle. Dicho mensaje será reenviado a la cola de reintentos automática.
Entrada	{"tipo":"Delete","name_entity":"af_objeto", "id":"00000000-0000-0000-0000-000000000000","key":"af_-objetoid","value":"'7bee8edb-318d-e911-a971"','error":"","retryCount":0}
Resultado esperado	Se ha producido un error al intentar insertar un registro duplicado. El mensaje se ha enviado a la cola manual.
Resultado de la prueba	Incorrecto. Las condiciones no estaban correctamente definidas. .

Tabla 6.19: Prueba de caja negra 018

Prueba de caja negra 020		
Descripción	Al intentar ejecutar el comando de eliminación Oracle informa de que se ha producido error y el mensaje ya ha se ha intentado entregar tres veces. Pasa a cola manual.	
Entrada	Mensaje	{"tipo":"Delete","name_entity":"af_objeto", "id":"00000000-0000-0000-0000-000000000000", "key":"af_objetoid","value":"'7bee8edb-318d-e911-a971"', "error":"","retryCount"3}
	Base de datos Oracle	Sin iniciar.
Resultado esperado	. El mensaje se ha enviado a la cola manual.	
Resultado de la prueba	Correcto.	

Tabla 6.20: Prueba de caja negra 020

6.1.6. Pruebas de caja negra Papelera de reciclaje

Prueba de caja negra 021	
Descripción	Se elimina un registro de af_objeto y comprobamos que se ha añadido los datos: af_entityname, af_id_name_key y af_guid en la entidad af_entitydelete
Entrada	Eliminar registro, de la entidad af_objeto, con los siguientes datos: af_entityname: af_objeto af_id_name_key: af_objetoid af_guid:{25794757-D266-E911-A96C-000D3AB5A6AE}
Resultado esperado	En la entidad af_entitydelete, quedan almacenados en los campos: af_entityname: af_objeto af_id_name_key: af_objetoid af_guid: {25794757-D266-E911-A96C-000D3AB5A6AE}
Resultado de la prueba	Correcto

Tabla 6.21: Prueba de caja negra 021

6.1.7. Pruebas de caja negra RetryAutomatica

Prueba de caja negra 022	
Descripción	Se envía los mensajes de tipo create al tema "Create" con la suscripción"Create".
Entrada	{"tipo":"Create","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-000d3ab5a0d7", "key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7','hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:03:53"', "error":"","retryCount":1}
Resultado esperado	Mensajes enviado al tema 'create'
Resultado de la prueba	Correcto.

Tabla 6.22: Prueba de caja negra 022

Prueba de caja negra 023	
Descripción	Se envía los mensajes de tipo update al tema "Update" con la suscripción "Update".
Entrada	{"tipo":"Update","name_entity":"af_objeto","id":"83b17e23-0a8d-e911-a971-000d3ab5a0d7", "key":"af_objetoid,af_name,createdon,modifiedon", "value":"'83b17e23-0a8d-e911-a971-000d3ab5a0d7','hfratpnwbf','12/06/2019 12:03:53','12/06/2019 12:25:53"' ,"error":"","retryCount":1}
Resultado esperado	Mensajes enviado al tema úpdate'
Resultado de la prueba	Correcto.

Tabla 6.23: Prueba de caja negra 023

Prueba de caja negra 024	
Descripción	Se envía los mensajes de tipo delete al tema "DElete" con la suscripción "Delete".
Entrada	Eliminar registro, de la entidad af_objeto, con los siguientes datos: af_entityname: af_objeto af_id_name_key: af_objetoid af_guid:{25794757-D266-E911-A96C-000D3AB5A6AE}
Resultado esperado	Mensajes enviado al tema 'delete'
Resultado de la prueba	Correcto.

Tabla 6.24: Prueba de caja negra 024

6.1.8. Pruebas de caja negra EnvioEmailColaMensajesFallidos

Prueba de caja negra 025	
Descripción	Cuando el número de registros almacenados por la cola manual es superior a 5 se enviara un email al equipo de gestión.
Entrada	Mensajes en la cola manual >5
Resultado esperado	La entidad email estará creada en CRM Dynamics 365
Resultado de la prueba	Correcto.

Tabla 6.25: Prueba de caja negra 025

Prueba de caja negra 026	
Descripción	Cuando el número de mensajes en la DeadLetter de ”create” es superior a 1 se enviara un email al equipo de gestión.
Entrada	DeadLetter de ”create” >1
Resultado esperado	La entidad email estará creada en CRM Dynamics 365
Resultado de la prueba	Correcto.

Tabla 6.26: Prueba de caja negra 026

Prueba de caja negra 027	
Descripción	Cuando el número de mensajes en la DeadLetter de ”update” es superior a 1 se enviara un email al equipo de gestión.
Entrada	DeadLetter de ”update” >1
Resultado esperado	La entidad email estará creada en CRM Dynamics 365
Resultado de la prueba	Correcto.

Tabla 6.27: Prueba de caja negra 027

Prueba de caja negra 028	
Descripción	Cuando el número de mensajes en la DeadLetter de ”delete” es superior a 1 se enviara un email al equipo de gestión.
Entrada	DeadLetter de ”delete” >1
Resultado esperado	La entidad email estará creada en CRM Dynamics 365
Resultado de la prueba	Correcto.

Tabla 6.28: Prueba de caja negra 028

6.1.9. Pruebas de caja negra RegularizaciónDatosCrm

Prueba de caja negra 029	
Descripción	Se comprueba que la conexión con CRM Dynamics 365 se realiza correctamente.
Entrada	User, Password y URI
Resultado esperado	La conexión con CRM Dynamics 365 se ha realizado correctamente.
Resultado de la prueba	Correcto.

Tabla 6.29: Prueba de caja negra 029

Prueba de caja negra 030	
Descripción	Los mensajes de tipo create se enviaran al tema "Create" con la suscripción "Create".
Entrada	Mensaje de tipo create
Resultado esperado	El mensaje esta almacenado en el tema create
Resultado de la prueba	Correcto.

Tabla 6.30: Prueba de caja negra 030

Prueba de caja negra 031	
Descripción	Los mensajes de tipo update se enviaran al tema "Update" con la suscripción "Update".
Entrada	Mensaje de tipo update
Resultado esperado	El mensaje esta almacenado en el tema update
Resultado de la prueba	Correcto.

Tabla 6.31: Prueba de caja negra 031

6.2. Pruebas de caja blanca

En esta sección se incluirán las pruebas de caja blanca, las cuales se realizan en base al código y la estructura del producto del sistema y usa ese conocimiento para la realización de las prueba.

6.2.1. Pruebas de caja blanca CallToCrmCreateUpdate

Prueba de caja blanca 001	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros de *Microsoft Dynamics 365.*
Entrada	Registro creado en CRM con el GUID: {25794757-D266-E911-A96C-000D3AB5A6AE}
Resultado esperado	La *Azure Function* extrae el registro con el *GUID*: {25794757-D266-E911-A96C-000D3AB5A6AE}
Resultado de la prueba	Correcto

Tabla 6.32: Prueba de caja blanca 001

Prueba de caja blanca 002	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros de *Microsoft Dynamics 365* dentro de un intervalo de tiempo determinado.
Entrada	Registro creado en CRM con el GUID: {F949E58F-D766-E911-A964-000D3AB5A84E} 50 segundos antes de iniciar la *Azure Function* .
Resultado esperado	La Function extrae el registro con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E}
Resultado de la prueba	Correcto

Tabla 6.33: Prueba de caja blanca 002

Prueba de caja blanca 003	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros creados de *Microsoft Dynamics 365* dentro de un intervalo de tiempo determinado.
Entrada	Registro creado en CRM con el GUID: {F949E58F-D766-E911-A964-000D3AB5A84E} 70 segundos antes de iniciar la *Azure Function.*
Resultado esperado	La *Azure Function* no extrae el registro con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E}
Resultado de la prueba	Correcto

Tabla 6.34: Prueba de caja blanca 003

Prueba de caja blanca 004	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros actualizados de *Microsoft Dynamics 365* dentro de un intervalo de tiempo determinado.
Entrada	Registro actualizado en *Microsoft Dynamics 365* con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E} 50 segundos antes de iniciar la functions
Resultado esperado	La *Azure Function* extrae el registro con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E}
Resultado de la prueba	Correcto

Tabla 6.35: Prueba de caja blanca 004

Prueba de caja blanca 005	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros actualizados de *Microsoft Dynamics 365* dentro de un intervalo de tiempo determinado.
Entrada	Registro actualizado en *Microsoft Dynamics 365* con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E} 70 segundos antes de iniciar la functions
Resultado esperado	La *Azure Function* extrae el registro con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E}
Resultado de la prueba	Correcto

Tabla 6.36: Prueba de caja blanca 005

Prueba de caja blanca 006	
Descripción	Se comprueba que la *Azure Function* distingue los registros actualizado de los creado mediante los campos *createOn* y *modifyOn*
Entrada	Probamos con dos registros, uno actualizado y otro recién creado .
Resultado esperado	La *Azure Function* distingue correctamente entre ambos.
Resultado de la prueba	Correcto

Tabla 6.37: Prueba de caja blanca 006

6.2.2. Pruebas de caja blanca CallToCrmDelete

Prueba de caja blanca 007	
Descripción	Se comprueba que la *Azure Function* extrae correctamente lo registros eliminamos del *Microsoft Dynamics 365* dentro de un intervalo de tiempo determinado.
Entrada	Registro actualizado en *Microsoft Dynamics 365* con el GUID: {F949E58F-D766-E911-A964-000D3AB5A84E} 70 segundos antes de iniciar la *Azure Function*
Resultado esperado	La *Azure Function* extrae el registro con el *GUID*: {F949E58F-D766-E911-A964-000D3AB5A84E}
Resultado de la prueba	Correcto

Tabla 6.38: Prueba de caja blanca 007

6.2.3. Pruebas de caja blanca ConnectionToOracleCreate

Prueba de caja blanca 009	
Descripción	Se comprueba que el campo *retryCount* se evalúa correctamente.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b8c-e911-a96a-000d3ab5a3d0”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e911-a96a-000d3ab5a3d0’,’pvvglyjduw’,’11/06/2019 11:23:07’,’11/06/2019 11:23:07”’, ”error”:””,retryCount”:0}
Resultado esperado	El mensaje no ha sido reintentado.
Resultado de la prueba	Correcto.

Tabla 6.39: Prueba de caja blanca 009

Prueba de caja blanca 010	
Descripción	Cuando el mensaje de creación tenga almacenado un valor en su *retryCount* superior a 0, se consultará a la base de dats de Oracle si la información almacenada en dicho mensaje ya está guardada dentro del sistema de bases de datos de Oracle.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b9c-e911-a96a-000d3ab5a3d2”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e911-a96a-000d3ab5a3d0’,’pvvpllyjduw’,’11/06/2019 11:24:07’,’11/06/2019 11:24:07”’,”error”:””,retryCount”:1}
Resultado esperado	En la base de datos ya está creado el registro asociado.
Resultado de la prueba	Correcto.

Tabla 6.40: Prueba de caja blanca 010

Prueba de caja blanca 011	
Descripción	Si el mensaje de creación, que tiene un valor en *retryCount* superior a 0, tiene información de un registro que ya no esta creado en Oracle se creará dicho mensaje en la base datos de Oracle.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b9c-e911-a96a-000d3ab5a3d2”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e911-a96a-000d3ab5a3d1’,’pvvpplyjduw’,’11/06/2019 11:24:07’,’11/06/2019 11:24:07”’,”error”:””,retryCount”:1}
Resultado esperado	El mensaje se crea en la base de datos de Oracle.
Resultado de la prueba	Correcto.

Tabla 6.41: Prueba de caja blanca 011

Prueba de caja blanca 012	
Descripción	Se comprueba si el comando de consulta, generado con la información del mensaje recibido que tiene un *retryCount* suprior a 0, es correcto.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b9c-e911-a85a-000d3ab5a3d2”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e915-a96a-000d3ab5a3d1’,’ssppslyjduw’,’11/06/2019 11:24:07’,’11/06/2019 11:24:07”’,”error”:””,retryCount”:1}
Resultado esperado	select * from af_objeto where af_objetoid=’6367ff45-3b8c-e915-a96a-000d3ab5a3d1’;
Resultado de la prueba	Correcto.

Tabla 6.42: Prueba de caja blanca 012

Prueba de caja blanca 013	
Descripción	Si el mensaje se ha consultado correctamente, del mensaje de creación con *retryCount* superior a 0, el mensaje no se enviará a ninguna cola.
Entrada	message.error=Succefull
Resultado esperado	Ninguna acción
Resultado de la prueba	Correcto.

Tabla 6.43: Prueba de caja blanca 013

Prueba de caja blanca 014	
Descripción	Si al realizar la consulta, en relación a un mensaje creación cuyo *retryCount* es superior a 0, se produce un error relacionado con el formato del mensaje dicho mensaje se reenvía a la cola manual.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b9c-e911-a96a-000d3ab5a3d2”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e915-a96a-000d3ab5a3d1’,’sssslyjduw’,’11/06/2019 11:24:07’,’11/06/2019 11:24:07”’,”error”:””,retryCount”:1}
Resultado esperado	El mensaje queda almacenado en la cola manual.
Resultado de la prueba	Correcto.

Tabla 6.44: Prueba de caja blanca 014

Prueba de caja blanca 015	
Descripción	Si al realizar la consulta, en relación a un mensaje creación cuyo *retryCount* es superior a 0, se produce un error relacionado con la conexión hacia la base de datos dicho mensaje se reenvía a la cola automática.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”6367ff45-3b9c-e911-a85a-000d3ab5a3d2”, ”key":”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’6367ff45-3b8c-e915-a96a-000d3ab5a3d1’,’ssppslyjduw’,’11/06/2019 11:24:07’,’11/06/2019 11:24:07”’,”error”:””,retryCount”:1}
Resultado esperado	El mensaje queda almacenado en la cola automática.
Resultado de la prueba	Correcto.

Tabla 6.45: Prueba de caja blanca 015

Prueba de caja blanca 016	
Descripción	El comando de creación de registro se crea correctamente usando la información almacenada dentro del mensaje recibido.
Entrada	{”tipo”:”Create”,”name_entity”:”af_objeto”,”id”:”83b17e23-0a8d-e911-a971-000d3ab5a0d7”, ”key”:”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’83b17e23-0a8d-e911-a971-000d3ab5a0d7’,Úpdate hfratpnwbf’, ’12/06/2019 12:03:53’,’12/06/2019 12:03:53”’,”error”:””,”retryCount”:0}
Resultado esperado	INSERT INTO af_objeto (af_objetoid,af_name,createdon,modifiedon) values (’83b17e23-0a8d-e911-a971-000d3ab5a0d7’,Úpdate hfratpnwbf’,’ 12/06/2019 12:03:53’,’12/06/2019 12:03:53’)
Resultado de la prueba	Correcto.

Tabla 6.46: Prueba de caja blanca 016

Prueba de caja blanca 017	
Descripción	Los errores devueltos por Oracle son filtrados correctamente.
Entrada	Mensaje de consulta sin haber iniciado el servidor de Oracle.
Resultado esperado	No se puede conectar con Oracle, *Timeout*
Resultado de la prueba	Correcto.

Tabla 6.47: Prueba de caja blanca 017

6.2.4. Pruebas de caja blanca ConnectToOracleUpdate

Prueba de caja blanca 018	
Descripción	Los mensajes de actualización generan un comando correcto de consulta.
Entrada	{”tipo”:”Update”,”name_entity”:”af_objeto”,”id”:”83b17e23-0a8d-e911-a971-111d3ab5a0d7”, ”key”:”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’83b17e23-0a8d-e911-a971-000d3ab5a0d7’,Úpdate dsfdfff’, ’12/06/2019 12:03:53’,’12/06/2019 12:03:53”’,”error”:””,”retryCount”:0}
Resultado esperado	select * from af_objeto where af_objetoid=’83b17e23-0a8d-e911-a971-000d3ab5a0d7’;
Resultado de la prueba	Correcto.

Tabla 6.48: Prueba de caja blanca 018

Prueba de caja blanca 019		
Descripción	Si la consulta acerca del mensaje de creación informa de que no hay registro asociado, los datos se crearan como nuevos.	
Entrada	Mensaje	{”tipo”:”Update”,”name_entity”:”af_objeto”,”id”:”83b17e23-0a8d-e911-a971-111d3ab5a0d7”, ”key”:”af_objetoid,af_name,createdon,modifiedon”, ”value”:”’83b17e23-0a8d-e911-a971-000d3ab5a0d7’,Úpdate dsfdfff’,’12/06/2019 12:03:53’, ’12/06/2019 12:03:53”’,”error”:””,”retryCount”:1}
	Registro Oracle	Vacío
Resultado esperado	En Oracle: af_objetoid:83b17e23-0a8d-e911-a971-000d3ab5a0d7 af_name: update createdon: 12/06/2019 12:03:53 modifiedon: 12/06/2019 12:20:53	
Resultado de la prueba	Correcto.	

Tabla 6.49: Prueba de caja blanca 019

6.2.5. Pruebas de caja blanca ConnectionToOracleDelete

Prueba de caja blanca 020	
Descripción	El mensaje de eliminación se convierte en un comando de Oracle.
Entrada	{”tipo”:”Delete”,”name_entity”:”af_objeto”, ”id”:”00000000-0000-0000-0000-000000000000”, ”key":”af_objetoid”,”value”:”’7bee8edb-318d-e911-a971-000d3ab5a0d7”’, ”error”:””,”retryCount”:0}
Resultado esperado	DELETE FROM af_objetoid WHERE af_objetoid = ’7bee8edb-318d-e911-a971-000d3ab5a0d7’
Resultado de la prueba	Correcto.

Tabla 6.50: Prueba de caja blanca 020

6.2.6. Pruebas de caja blanca RetryAutomatica

Prueba de caja blanca 021	
Descripción	Se comprueba la disponibilidad de Oracle enviado un comando de consulta.
Entrada	select * from countries where country_id=ÁR’
Resultado esperado	Conexión con Oracle realizada
Resultado de la prueba	Correcto.

Tabla 6.51: Prueba de caja blanca 021

Prueba de caja blanca 022		
Descripción	Se comprueba la falta de disponibilidad de la base de datos de Oracle enviado un comando de consulta.	
Entrada	Mensaje	select * from countries where country_id=ÁR’
	Base de datos Oracle	Sin iniciar.
Resultado esperado	La *Azure Function* finaliza.	
Resultado de la prueba	Correcto.	

Tabla 6.52: Prueba de caja blanca 022

6.2.7. Pruebas de caja blanca EnvioEmailColaMensajesFallidos

Prueba de caja blanca 023	
Descripción	El número de mensajes almacenados en *Microsoft Dynamics 365* por la cola manual se extrae correctamente.
Entrada	*commandmanual*
Resultado esperado	El mismo valor que muestra el portal de *Microsoft Dynamics 365*.
Resultado de la prueba	Correcto.

Tabla 6.53: Prueba de caja blanca 023

Prueba de caja blanca 024	
Descripción	El número de mensajes almacenados en la *DeadLetter* de "*create*" se extrae correctamente.
Entrada	DeadLetter de "create"
Resultado esperado	El mismo valor que muestra el portal de *Azure*.
Resultado de la prueba	Correcto.

Tabla 6.54: Prueba de caja blanca 024

Prueba de caja blanca 025	
Descripción	El número de mensajes almacenados en la *DeadLetter* de "update" se extrae correctamente.
Entrada	*DeadLetter* de "*update*"
Resultado esperado	El mismo valor que muestra el portal de *Azure*.
Resultado de la prueba	Correcto.

Tabla 6.55: Prueba de caja blanca 025

Prueba de caja blanca 026	
Descripción	El número de mensajes almacenados en la *DeadLetter* de "*delete*" se extrae correctamente.
Entrada	*DeadLetter* de "*delete*"
Resultado esperado	El mismo valor que muestra el portal de *Azure*.
Resultado de la prueba	Correcto.

Tabla 6.56: Prueba de caja blanca 026

6.2.8. Pruebas de caja blanca RegularizaciónDatosCrm

Prueba de caja blanca 027		
Descripción	Se comprueba que los registros extraídos de *Microsoft Dynamics 365* fueron modificados dentro del intervalo temporal proporcionado por el usuario.	
Entrada	Fecha de inicio	05/05/2019 12:00:00
	Fecha de finalización	08/05/2019 12:00:00
Resultado esperado	El valor del campo *modifiedon*, de los registros extraídos, esta dentro del intervalo temporal proporcionado por el usuario.	
Resultado de la prueba	Correcto.	

Tabla 6.57: Prueba de caja blanca 027

Prueba de caja blanca 028	
Descripción	Se genera un mensaje acorde a los datos extraídos.
Entrada	Datos del registro.
Resultado esperado	Mensaje correcto.
Resultado de la prueba	Correcto.

Tabla 6.58: Prueba de caja blanca 028

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

El valor añadido que se puede extraer de la realización de este proyecto, es el aprendizaje que me ha supuesto. El cambio de paradigma, al ser en la nube, y el proceso de aprendizaje en una tecnología nueva, me ha permitido tener un cambio de mentalidad al afrontar los retos, que este trabajo ha puesto de manifiesto.

También, todo el aprendizaje acerca del funcionamiento del procesamiento en la nube, gracias a las *Azure Functions*, el modelo de colas y la potencia que se puede obtener de las mismas ha hecho que el desarrollo de este TFG haya sido muy satisfactorio.

Las *Azure Functions* son un modo de generar código útil muy enfocado a un problema concreto, que me ha permitido desarrollar soluciones muy específicas al contexto en que debía aplicarlas. El hecho de no tener que preocuparme por una infraestructura asociada, me ha dado la oportunidad de probar distintos enfoques sin tener que ajustar una infraestructura que limite dichas pruebas por el perjuicio de obligarme a desechar trabajo.

Por su parte, el uso de *Service Bus* y la investigación relativa a ello, me ha permitido conocer tecnologías y modelos de los cuales tenía un conocimiento muy superficial. Al igual que sucedió con el uso de *Azure Functions*, la ausencia de necesidad de administrar un servidor me ha permitido realizar pruebas y desarrollar soluciones muy diversas sin tener preocuparme de tener que cambiar toda la infraestructura para poder desarrollarlas.

Por otra parte, me ha hecho valorar, más si cabe, todo lo que he aprendido durante estos años en la carrera puesto que me ha permitido llevar a cabo todas esta enseñanzas en un trabajo, dándome una visión global de los procesos de ingeniería utilizados.

7.2. Trabajo futuro

Las líneas futuro de desarrollo de este sistema vendrán guiadas por las siguientes ideas:

- Aumentar la cantidad de entidades, de las cuales, se mantiene la persistencia de sus datos.
- Ampliar la cantidad de Functions que consumen los mensajes de *Service Bus* y los insertan en la base de datos. Para ello, se utilizará las ventajas que ofrecen las suscripciones de los Temas.

  A diferencia de las colas de *Service Bus*, en las que un solo destinatario procesa cada mensaje, los temas y las suscripciones proporcionan una forma de comunicación de uno a varios

mediante un patrón de publicación/suscripción. Es posible registrar varias suscripciones en un tema. Cuando un mensaje se envía a un tema, pasa a estar disponible para cada suscripción para la administración o el procesamiento de manera independiente. Una suscripción a un tema se asemeja a una cola virtual que recibe copias de los mensajes que se enviaron al tema. Opcionalmente, puede registrar reglas de filtros para un tema por suscripción, lo que le permite filtrar o restringir qué mensajes para un tema reciben las suscripciones a un tema.

De esta manera en el caso, por poner un ejemplo, que la cantidad de registros de creación aumentaran dentro del sistema, sería muy sencillo escalarlo.

- A largo plazo, sería recomendable hacer más robusto el algoritmo que consulta la disponibilidad de la base de datos, en la *Azure Function* que maneja los reintentos automáticos. Para ello, sería recomendable utilizar el patrón disyuntor [16].
- Implementar una IA encargada de arreglar o formatear los mensajes fallidos.

Bibliografía

[1] [Organización: Microsoft] , Patrón Command and Query Responsibility Segregation [Online],
Disponible en: https://docs.microsoft.com/es-es/azure/architecture/patterns/cqrs
[Publicado: Jun 23, 2017]

[2] [Organización: Uniwebsidad] , Modelo en cascada [Online],
Disponible en: https://uniwebsidad.com/libros/tdd/capitulo-1/modelo-en-cascada
[Publicado: 2010-2013]

[3] [Organización: Oracle] , Descargas de Software: Descarga gratuita, aprendizaje gratuito, evaluación ilimitada [Online],
Disponible en: https://www.oracle.com/technetwork/es/indexes/downloads/index.html
[Accedido: Feb 21, 2019

[4] [Organización: Microsoft] , Información general de auditoría [Online],
Disponible en: https://docs.microsoft.com/es-es/dynamics365/customer-engagement/developer/auditing-overview
[Publicado: Ene 29, 2019]

[5] [Organización: Microsoft] , Información general de colas de mensajes fallidos de Service Bus [Online],
Disponible en: https://docs.microsoft.com/es-es/azure/service-bus-messaging/service-bus-dead-letter-queues
[Publicado: May 21, 2019]

[6] [Organización: Microsoft] , Patrón de disyuntor [Online],
Disponible en: https://docs.microsoft.com/es-es/azure/architecture/patterns/circuit-breaker
[Publicado: Jun 23, 2017]

[7] [Organización: Microsoft] , Patrón Retry [Online],
Disponible en: https://docs.microsoft.com/es-es/azure/architecture/patterns/retry
[Publicado: Jun 23, 2017]

[8] [Organización: Microsoft] , Estilos de arquitectura [Online],
Disponible en: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/
[Publicado: Sep 30, 2018]

[9] [Organización: dofactory] , Patrón Iterator [Online],
Disponible en: https://www.dofactory.com/net/iterator-design-pattern
[Publicado: 2019]

[10] [Organización: Medium] , Patrón Observer [Online],
Disponible en: https://medium.com/datadriveninvestor/design-patterns-a-quick-guide-to-observer-pattern-d0622145d6c2
[Publicado: Jun 07, 2018]

[11] [Organización: martinfowler] , Transaction Script [Online],
Disponible en: https://martinfowler.com/eaaCatalog/transactionScript.html
[Publicado: Feb 07, 2019]

[12] [Organización: sourcemaking] , Proxy Design Pattern [Online],
Disponible en: https://sourcemaking.com/design_patterns/proxy
[Publicado: Feb 07, 2019]

[13] [Organización: Microsoft] , CQRS para la aplicación común [Online],
Disponible en: https://msdn.microsoft.com/magazine/mt147237
[Publicado: Jun, 2015]

[14] [Organización: Microsoft] , CQRS: Aprovechar CQRS para crear sistemas con alta capacidad de respuesta [Online],
Disponible en: https://msdn.microsoft.com/es-es/magazine/mt736455.aspx
[Publicado: Jun, 2015]

[15] [Organización: Microsoft] , Tecnología de vanguardia: CQRS y las aplicaciones basadas en mensajes [Online],
Disponible en: https://msdn.microsoft.com/es-es/magazine/mt238399.aspx
[Publicado: Jul , 2015]

[16] [Organización: Microsoft] , CQRS BUS and Windows Azure technologies [Online],
Disponible en: https://blogs.msdn.microsoft.com/cesardelatorre/2012/02/22/cqrs-bus-and-windows-azure-technologies/
[Publicado: February 22, 2012]

[17] [Organización: Medium] , Serverless CQRS in Azure: Part 1: Introduction, Commands Event sourcing [Online],
Disponible en: https://medium.com/@richard.j.gobbett/serverless-cqrs-in-azure-p1-e0f2c423f071/
[Publicado: Feb 9, 2018]

[18] [Organización: UM] , Bases de datos relacionales [Online],
Disponible en: https://www.um.es/geograf/sigmur/temariohtml/node63_mn.html
[Publicado: Nov 22, 2018]

[19] [Organización: Monografía] , Bases de datos relacionales [Online], Disponible en: https://www.um.es/geograf/sigmur/temariohtml/node63_mn.html
[Publicado: Feb 03, 2003]

[20] [Organización: Microsoft] , Qué es Azure Service Bus [Online], Disponible en: https://docs.microsoft.com/es-es/azure/service-bus-messaging/service-bus-messaging-overview
[Publicado: Nov 22, 2018]

[21] [Organización: XrmToolBox] , Plugin Registration [Online], Disponible en: https://www.xrmtoolbox.com/plugins/Xrm.Sdk.PluginRegistration/
[Publicado: 2019]

[22] [Organización: Microsoft] , Calculadora de precios [Online], Disponible en: https://azure.microsoft.com/es-es/pricing/calculator/
[Publicado: Ene 20, 2019]

[23] [Autor: Craig Larman] , Agile and Iterative Development: A Manager's Guide [Libro],
[Publicado: Addison-Wesley Professional, 2003]

[24] [Autor: Ritesh Modi] , Azure for Architects [Libro],
[Publicado:Packt, 2017]

Anexo A

Manual de Instalación

Creación de *Queue* , *Topic* y *Subscription* en *Service Bus*

En el siguiente apartado, se indicaran los pasos a seguir para crear y configurar *Service Bus* en *Azure*.

1. Se crea un nuevo servicio tipo *Service Bus*, para ello se pulsa sobre el botón *Crear nuevo recurso* y se busca el servicio A.1.

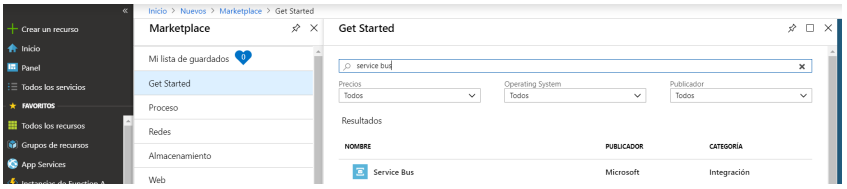

Figura A.1: Búsqueda un servicio nuevo en *Azure*.

2. Se crea el servicio A.2.

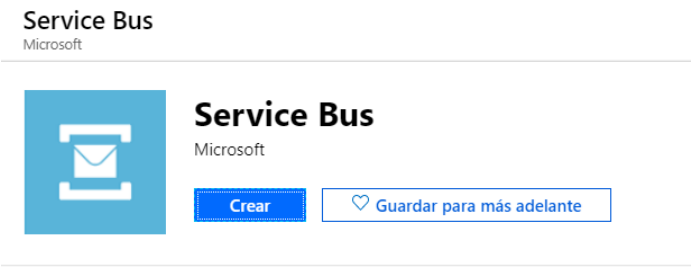

Figura A.2: Creación de un servicio nuevo en *Azure*.

3. Se crea el espacio de nombres y se elige la suscripción estándar, que es la que tiene mejor relación calidad/precio A.3.

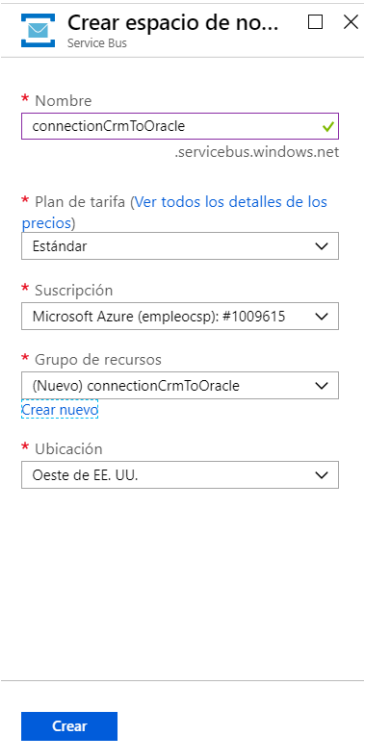

Figura A.3: Creación de espacio de nombres y elección de suscripción.

4. Una vez creado el espacio de nombres, se procede a crear una cola. Se pulsa sobre *cola.* A.4.

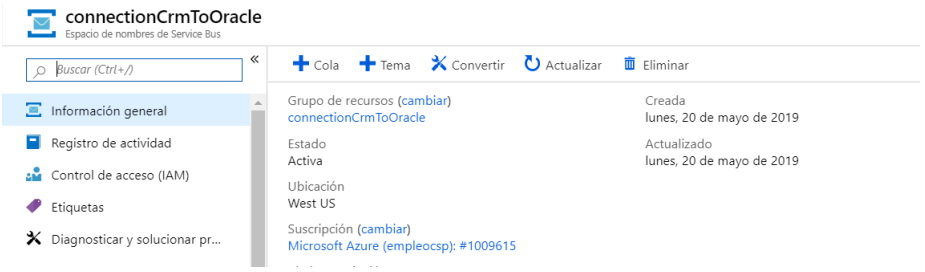

Figura A.4: Creación de una *cola*.

5. Se configura la cola con los parámetros A.5. Se debe crear dos iguales, una para la cola manual y otra para cola de reintentos automática.

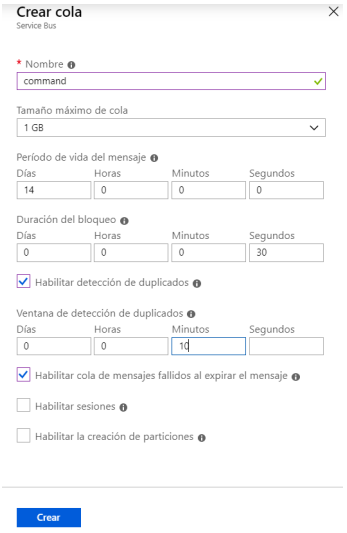

Figura A.5: Parámetros de una *cola*.

6. Se crea un *Topic* con los siguientes parámetros A.6. Se explica más en detalle que es un *Topic* en la sección 2.1.1.

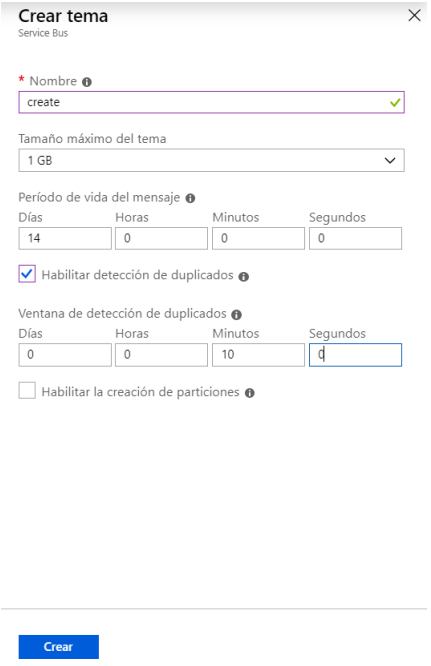

Figura A.6: Parámetros de un *Topic*.

7. Como se puede ver en la imagen A.7, se han creado tres *Topics*. Una para cada paso de cada mecanismo implementado de acceso a la base de datos, que trataran los mensajes de manera específica.

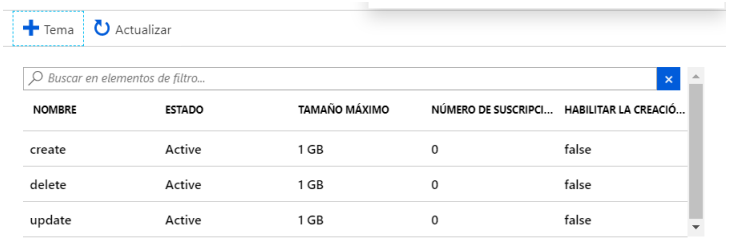

Figura A.7: *Topics*.

8. Se crea una *Subscription* para cada uno de los *Topics*. En A.8 se puede ver los parámetros de configuración utilizados.

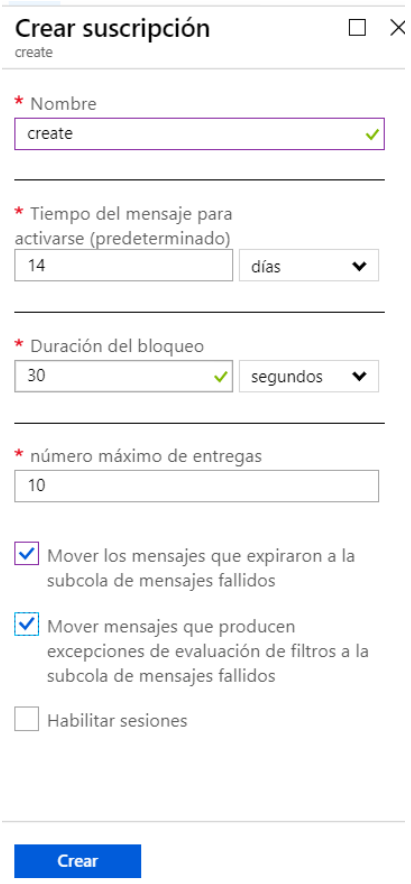

Figura A.8: Parámetros de un *Subscription*.

9. Así es como se ve finalmente A.9.

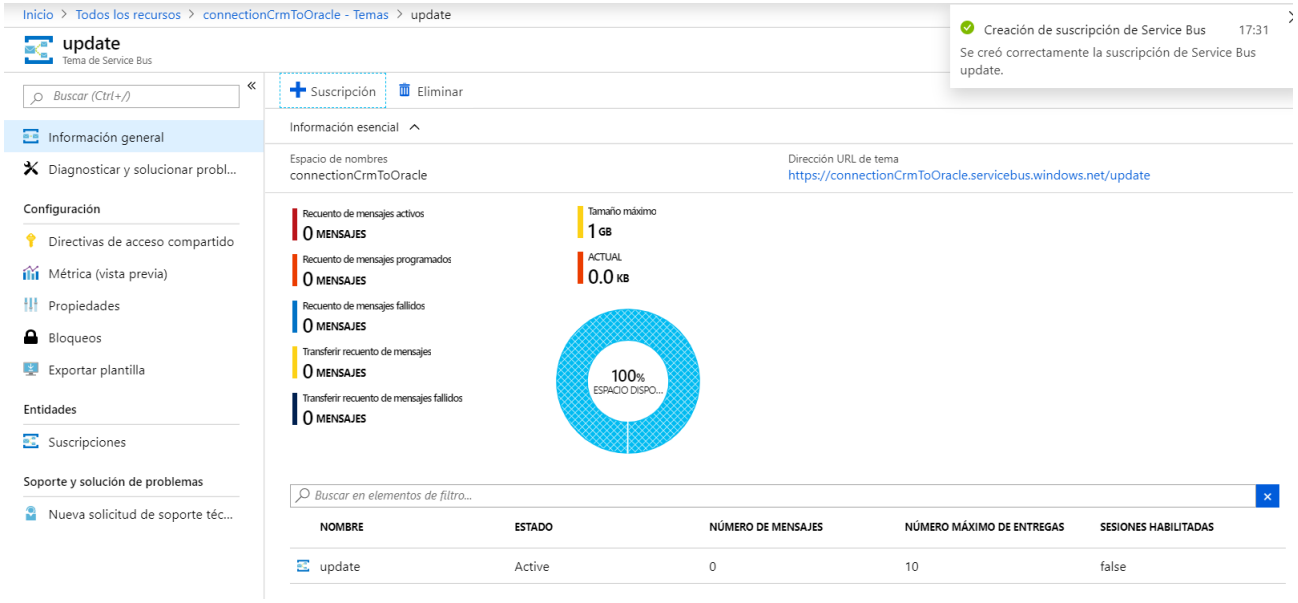

Figura A.9: Portal de *Azure*.

Despliegue de Plugin Papelera Reciclaje

En el siguiente apartado, indicaremos los pasos a seguir para la instalación de un *Plugin* encargado de salvar los datos necesarios de un registro para su posterior eliminación dentro de la base de datos de Oracle.

Requisitos previos

Como requisito previos serán necesarias los siguientes:

- Será necesario compilar el *Plugin* para obtener la *.dll* derivada. Además, dicha *.dll* debe de ir firmada.
- Será necesario tener instalado la herramienta *XrmToolBox* [17] con el *Plugin PluginRegistration* [18]
- En la herramienta *XrmToolBox* estará, previamente, creada la conexión a CRM dynamic 365.

1. Se busca el *Plugin*, llamado *Plugin Resgistration* en la herramienta y se abre A.10.

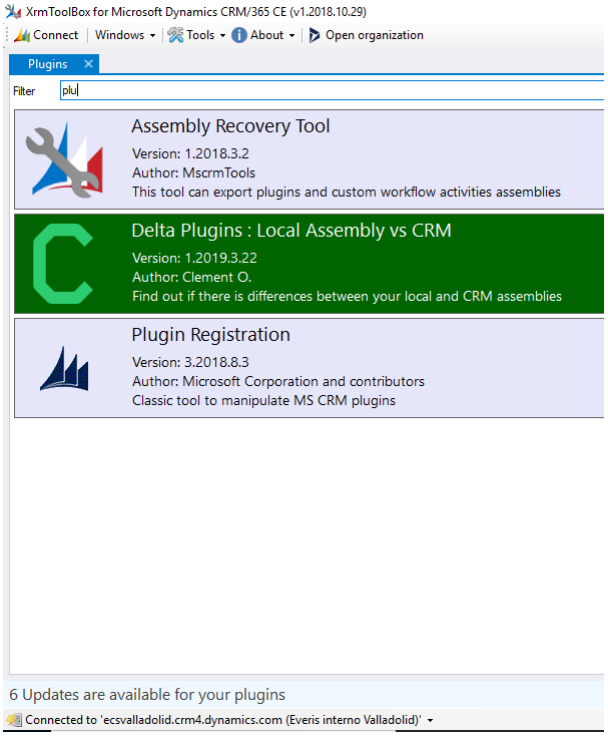

Figura A.10: Búsqueda de Plugin Resgistration.

2. Se abre *Register* para registrar una nueva *Assmeby* A.11.

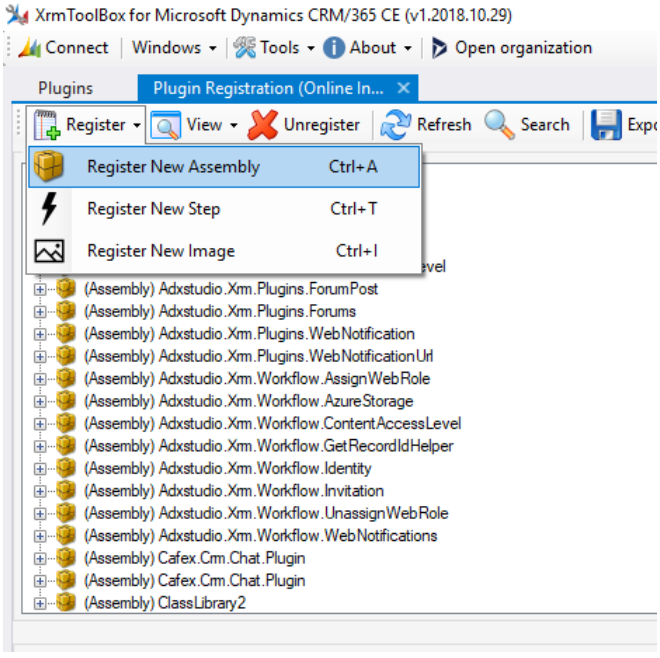

Figura A.11: Registro de una nueva *Assembly*.

3. Se busca la *.dll* y se carga A.12.

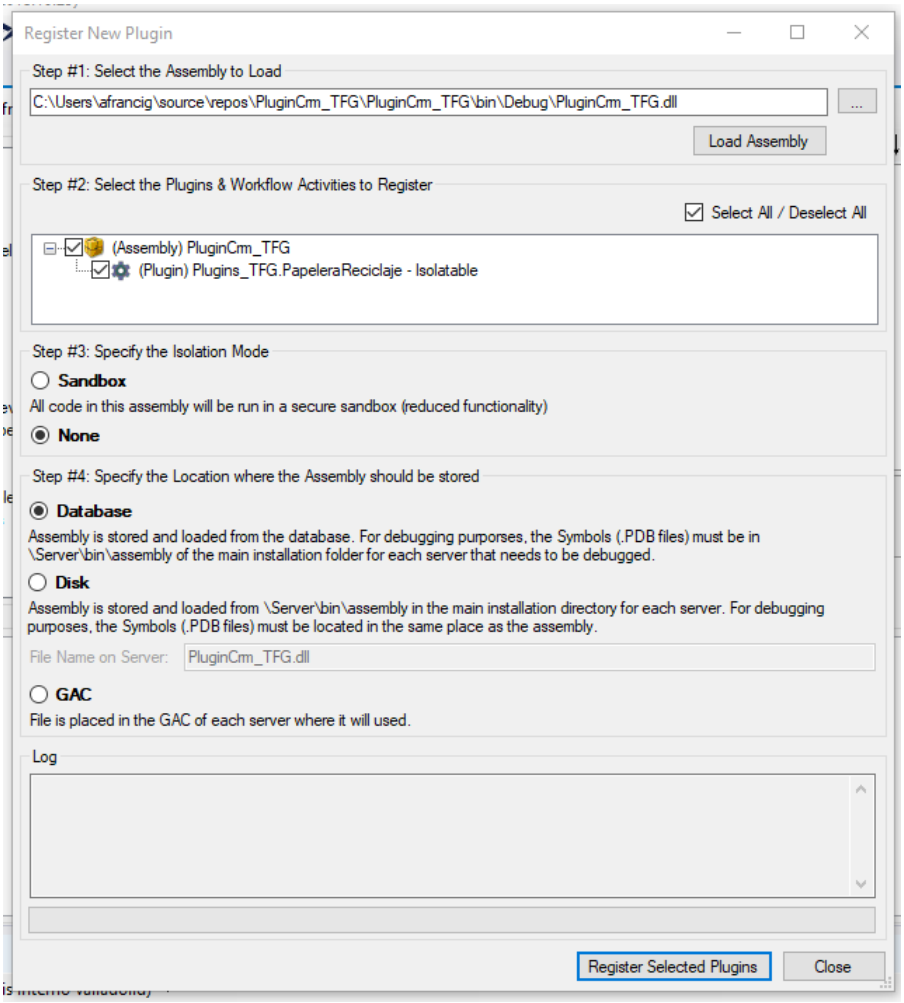

Figura A.12: Carga de la *.dll*.

4. Una vez registrada, es necesario registrar un nuevo paso A.13.

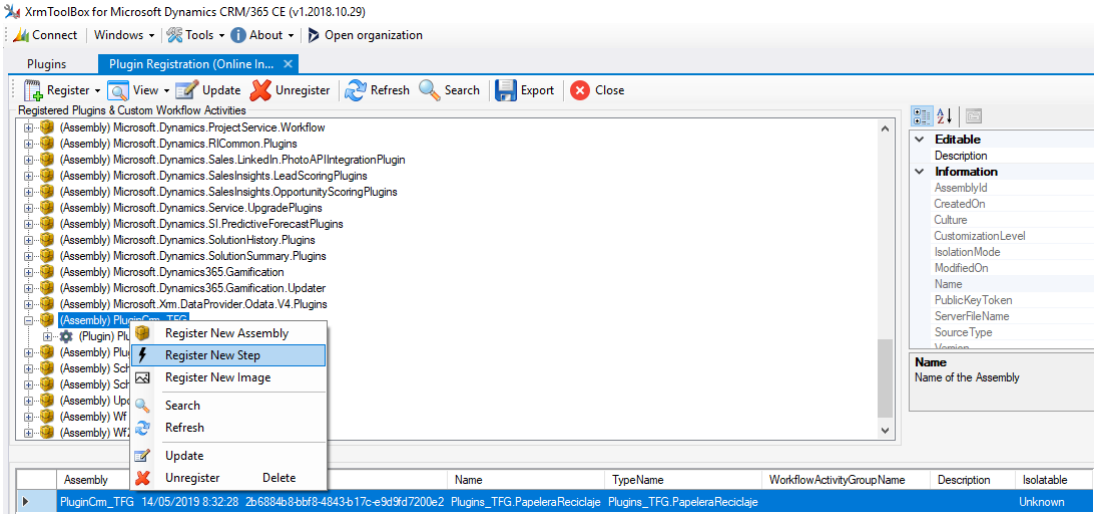

Figura A.13: Registro de un nuevo paso.

5. Se indica que, cada vez que se elimine una entidad dentro de *Microsoft Dynamics 365*, el *Plugin* debe de emperezar a funcionar A.14.

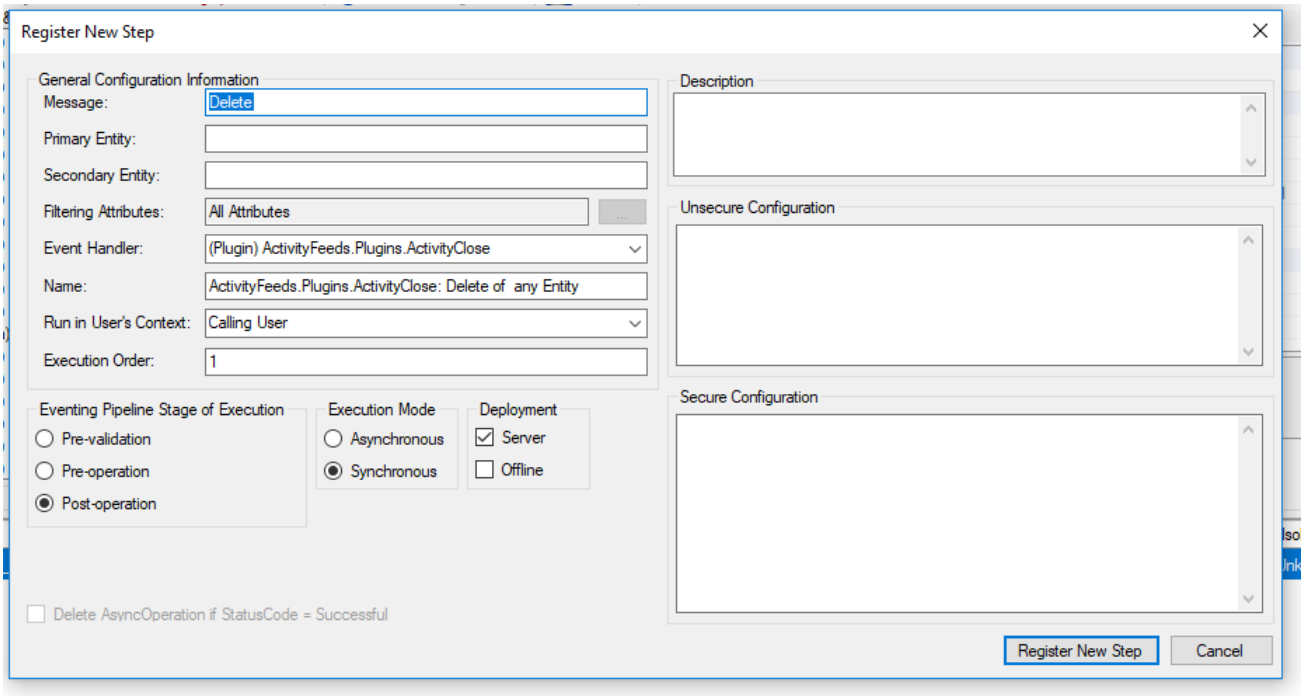

Figura A.14: Configuración del nuevo paso.

Anexo B

Manuales de Usuario

Programa de consola de regularización de registros en Microsoft Dynamics 365

El siguiente manual, será utilizado por el equipo de gestión para regularizar registros que no hayan podido ser consumidos por las *Azure Functions* debido a una interrupción del servicio.

1. Se inicia el programa de consola, usando para ello el *IDE Visual Studio 2017* B.1.

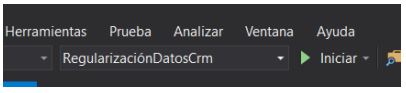

Figura B.1: Inicio de la aplicación de consola.

2. Se introducen las fechas que conforman el intervalo temporal en el cual se extraerán los datos de *Microsoft Dynamics 365*. La consola mostrará que registros han sido codificados como mensaje y ha que *Topic* se ha enviado B.2.

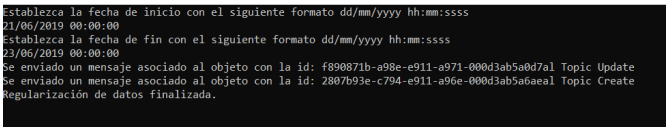

Figura B.2: Resultados arrojados por la consola.

Creación, actualización y borrado de registros dentro de *Microsoft Dynamics 365*

En el siguiente manual, se explicara como crear, actualizar y borrar dentro del sistema de *Microsoft Dynamics 365*, desde el punto de vista de un usuario.

Figura B.3: Vista de una entidad concreta.

1. Para crear un registro, dentro de una entidad concreta, solo es necesario pulsar sobre el botón *Nuevo* B.4.

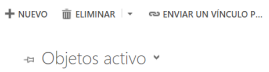

Figura B.4: Creación de registros en *Microsoft Dynamics 365*, paso 1

2. Se rellenan los campos del formulario y se pulsa sobre el botón *Guardar* para que esta información quede reflejada en el sistema B.5.

Figura B.5: Creación de registros en *Microsoft Dynamics 365*, paso 2

3. Finalmente, así es como queda almacenada la información en el nuevo registro B.6.

Figura B.6: Creación de registros en *Microsoft Dynamics 365*, paso 3

4. Para actualizar el registro creado anteriormente, solo es necesario cambiar la información que esta almacenada en los campos B.7.

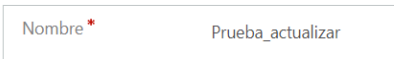

Figura B.7: Actualización de registros en *Microsoft Dynamics 365*, paso 1

5. Se pulsa sobre el icono de *Guardar* para que esta información sobreescriba a la que esta almacenada.B.8

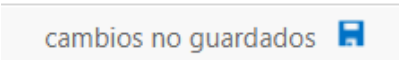

Figura B.8: Actualización de registros en *Microsoft Dynamics 365*, paso 2

6. Por último, si es necesario eliminar el registro se pulsa sobre el botón *Eliminar* B.9.

Figura B.9: Eliminación de registros en *Microsoft Dynamics 365*, paso 1

7. Aparece una ventana donde se pide confirmar la eliminación. Se pulsa sobre el botón Eliminar B.10.

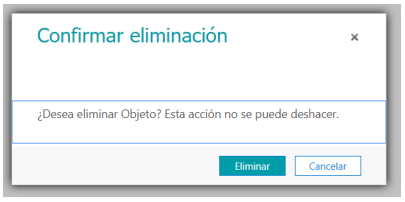

Figura B.10: Eliminación de registros en *Microsoft Dynamics 365*, paso 2

8. La información más relevante de la entidad eliminada aparece dentro de la entidad *Papelera* B.11.

Figura B.11: Eliminación de registros en *Microsoft Dynamics 365*, paso 3

Anexo C

Comparativa: Azure Service Bus vs RabbitMQ

Definición de sistemas

Azure Service

Microsoft Service Bus es un agente de mensajes de integración empresarial completamente administrado. *Service Bus* se usa normalmente para desacoplar las aplicaciones y los servicios entre sí y es una plataforma segura y confiable para datos asincrónicos y transferencia de estado. [19]

Las principales preocupaciones arquitectónicas, abordadas por los mensajes, son las siguientes:

Durabilidad: Los mensajes se almacenan en un lugar duradero y la aplicación puede leerlos después de que aparezcan.

Confiabilidad: Los mensajes ayudan a implementar la confiabilidad puesto que estos mensajes se almacenan en el disco

Disponibilidad de mensajes: los mensajes están disponibles para aplicaciones de consumo después de la restauración de la conectividad y el tiempo de inactividad anterior.

Azure proporciona colas y temas del bus de servicio para implementar patrones de mensajería dentro de las aplicaciones. El *Service Bus* de *Azure* brinda soporte para mensajes de 256 KB tamaño.[2]

RabbitMQ

RabbitMQ, un sistema de mensajería empresarial completo y altamente confiable basado en el estándar *AMQP*. Está licenciado bajo la licencia de código abierto *Mozilla Public License* y cuenta con una distribución independiente de plataformas. Debido a esto es de los más extendidos, incluso existe un módulo de *puppetlabs* que permite instalarlo, configurarlo y administrarlo. Sus características principales son:

- Garantía de entrega.
- Enrutamiento flexible.
- Clusterización.

- Federación.
- Alta disponibilidad.
- Tolerancia a fallos.

Comparativa

Conceptos

Namespaces

Microsoft Service Bus

Un *namespace* es un contenedor con un ámbito para todos los componentes de la mensajería. Varias colas y temas pueden residir en un único espacio de nombres, y los espacios de nombres suelen servir de contenedores de aplicación.

RabbitMQ

Rabbit no tiene un concepto de *namespace*, pero hay mucha similitud con el *host* virtual de *Rabbit*.

El *host* virtual es compatible con los controles de seguridad similares alrededor del *host* virtual, como lo haría Microsoft Service Bus con un espacio de nombres.

RabbitMQ emplea una realización más tangible de los *hosts* virtuales al convertirlo efectivamente en "clusters virtuales" encima del *broker*. Esto significa que no solo los *hosts* virtuales no comparten intercambios y colas, sino que los Usuarios, las *ACL*, las Políticas, etc. son específicos de cada *host* virtual.

Exchanges

Microsoft Service Bus

El uso del objeto "*Topic*" en *Service Bus* es muy similar al objeto "*Exchange*" en *Rabbit*.

Los *Topics* pueden tener múltiples registros. Esto es así debido al sistema que emplea basado en suscripciones, en el cual un *Topic* puede tener registradas múltiples suscripciones, estando cada una asociada a un receptor. Dicho receptor solo recibirá los mensajes que hayan sido previamente filtrados por la propia suscripción, lo que permite controlar la información que llega a cada receptor atendiendo, por ejemplo, al perfil del mismo.

RabbitMQ

El "*Exchange*" es el objeto de mensajería al que se envían los mensajes. Los hay de varios tipos:

***Direct Exchanges*:** Este tipo es útil cuando se desea distinguir los mensajes publicados en el mismo *exchange* utilizando un identificador simple. Entrega mensajes a colas basándose en las *routing keys* de los mensajes que funciona como la "dirección" que el *exchange* usa para

decidir dónde enrutar el mensaje. Un mensaje va a la/s cola/s cuya *binding key* coincida exactamente con la *routing key* del mensaje. Si la *routing key* no coincide con ninguna *binding key*, será descartado.

***Exchange* por defecto:** Es un *direct exchange* pre-declarado sin nombre, generalmente referido por la cadena vacía " ". Cuando se usa este *exchange*, el mensaje será entregado a la cola cuyo nombre sea igual a la *routing key* del mensaje. Cada cola se enlaza automáticamente al *exchange* por defecto con una routing key que es igual que el nombre de la cola.

***Topic Exchange*:** Este tipo de *exchanges* enrutan mensajes a colas basadas en coincidencias de comodines entre la *routing key* y algo llamado *routing pattern*, especificado por el *binding* de la cola. Los mensajes se enrutan a una o varias colas en función de si hay coincidencia entre una *routing key* de mensaje y este patrón.

***Fanout Exchange*:** Este *exchange* copia y enruta un mensaje recibido a todas las colas que están vinculadas a él, independientemente de las *routing keys* o la coincidencia de patrones como pasaba con los *exchanges* anteriores. Las *keys* proporcionadas simplemente serán ignoradas.

***Headers Exchange*:** Estos *exchanges* son muy similares a los *Topic exchanges*, pero se enruta en función de los valores de cabeceras en lugar de las *routing key*. Un mensaje se considera coincidente si el valor de la cabecera es igual al valor especificado en el binding.

***Dead Letter Exchange*:** Si no se puede encontrar una cola coincidente para un mensaje, el mensaje se eliminará silenciosamente. *RabbitMQ* proporciona una extensión *AMQP* conocida como "*Dead Letter Exchange*" que proporciona funcionalidad para capturar mensajes que no se pueden entregar

Colas

Microsoft Service Bus

Los mensajes se envían y se reciben desde colas. Las colas permiten almacenar mensajes hasta que la aplicación receptora está disponible para recibirlos y procesarlos C.1. Los mensajes de las

Figura C.1: Modelo de colas de *Service Bus*

colas se ordenan y se les asigna una marca de tiempo a su llegada. Una vez aceptado, el mensaje se conserva de forma segura en almacenamiento redundante. Los mensajes se entregan en modo de extracción, que entrega los mensajes a una solicitud.

RabbitMQ

Funciona exactamente igual.

Temas

Microsoft Service Bus

También puede usar temas para enviar y recibir mensajes. Mientras que una cola se utiliza a menudo para la comunicación punto a punto, los temas son útiles en escenarios de publicación y suscripción C.2. Los temas pueden tener varias suscripciones independientes. Un suscriptor a un

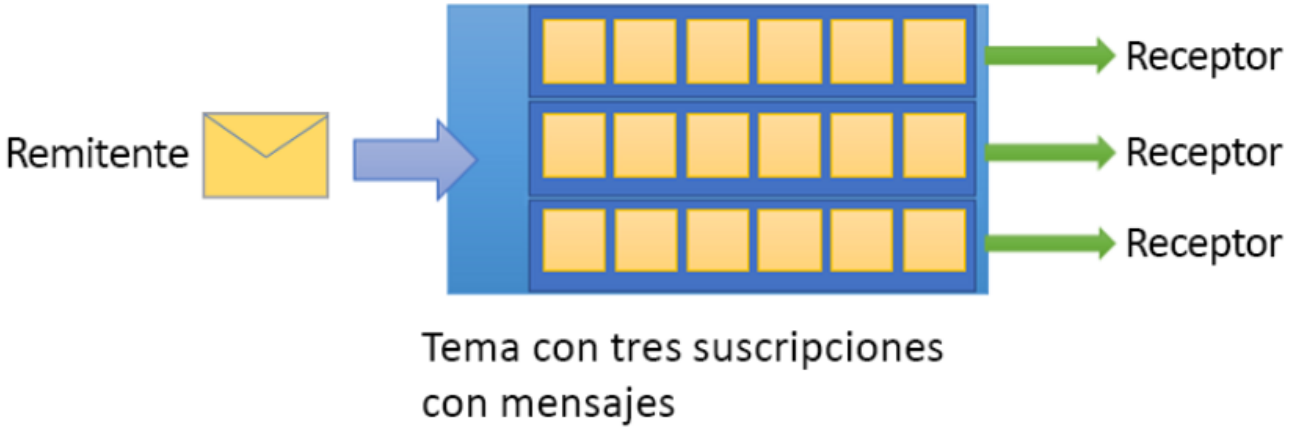

Figura C.2: Modelo de temas de *Service Bus*

tema puede recibir una copia de cada mensaje enviado a ese tema. Las suscripciones son entidades con nombre, que se crean de forma duradera pero pueden, opcionalmente, expirar o eliminarse automáticamente.

En algunos escenarios, se podría no desear suscripciones individuales para recibir todos los mensajes enviados a un tema. Si es así, se puede usar reglas y filtros para definir condiciones que desencadenan acciones opcionales, filtrar mensajes especificados y establecer o modificar las propiedades del mensaje.

RabbitMQ

Los intercambios de temas enrutan los mensajes a una o varias colas en función de la coincidencia entre una clave de enrutamiento de mensajes y el patrón que se usó para vincular una cola a un intercambio. El tipo de intercambio de temas se usa a menudo para implementar diversas variaciones de patrones de publicación / suscripción. Los intercambios de temas se utilizan comúnmente para el enrutamiento de mensajes de multidifusión.

Los intercambios de temas tienen un conjunto muy amplio de casos de uso. Cuando un problema involucra a varios consumidores / aplicaciones que eligen de forma selectiva qué tipo de mensajes desean recibir, se debe considerar el uso de intercambios de temas.

Suscripciones

Microsoft Service Bus

Los mensajes se envían a un tema y se entregan a una o varias suscripciones asociadas, según las reglas de filtro que se pueden establecer por suscripción. Las suscripciones pueden usar filtros adicionales para restringir los mensajes que desean recibir. Los mensajes se envían a un tema de la misma manera que se envían a una cola, pero no se reciben del tema directamente. En su lugar, se reciben de las suscripciones. Una suscripción al tema se parece a una cola virtual en que recibe copias de los mensajes que se envían al tema. Los mensajes se reciben de una suscripción exactamente de la misma forma en que se reciben de una cola.

RabbitMQ

Las suscripciones en *Rabbit* se basan en la clave de enrutamiento. Cuando coloca un mensaje en un intercambio, especifica una clave de enrutamiento que luego se usa de diferentes maneras según el tipo de intercambio que declare. Los tipos de intercambios se han explicado en la sección de "Exchange".

Routing Key

Microsoft Service Bus

Service Bus tiene un tipo de filtro llamado Filtro de correlación que ofrece características similares a la *routing key* en *RabbitMQ*. En los filtros de correlación hay una propiedad *CorrelationFilter* contiene un conjunto de condiciones para las que se busca la coincidencia con una o varias de las propiedades del usuario y del sistema de un mensaje entrante. Un uso común es buscar la coincidencia con la propiedad *CorrelationId*, pero la aplicación puede elegir también buscarla con *ContentType*, *Label*, *MessageId*, *ReplyTo*, *ReplyToSessionId*, *SessionId*, *To* y cualquier propiedad definida por el usuario.

Existe una coincidencia cuando el valor de un mensaje entrante de una propiedad es igual al valor especificado en el filtro de correlación. En las expresiones de cadena, la comparación distingue mayúsculas de minúsculas. Al especificar varias propiedades de coincidencia, el filtro las combina como una condición *AND* lógica, lo que significa que para que el filtro realice la coincidencia, todas las condiciones deben coincidir. Para realizar esta evaluación, el filtro evalúa las propiedades del mensaje.

RabbitMQ

Las *routing key* es un atributo de mensaje. El intercambio puede ver esta clave al decidir cómo enrutar el mensaje a colas (según el tipo de intercambio). El modo en se utiliza, se detalla más en profundidad en el apartado "exchange".

Colas duraderas

Microsoft Service Bus

Todas las colas son duraderas con *SQL Server* pues son persistentes.

RabbitMQ

Al declarar una cola, se puede hacer fácilmente duradero si se establece que esta propiedad es verdadera.

Colas no duraderas

Microsoft Service Bus

Service Bus no ofrece una cola solo en memoria en esta etapa, todos los mensajes son duraderos. Se puede lograr una funcionalidad similar a través de la propiedad *TTL* del mensaje, pero esto no pretende ser un equivalente a una cola no duradera.

RabbitMQ

Al declarar una cola, se puede hacer fácilmente no duradero al establecer la propiedad "es duradero" en falso. Esto significa que la cola está solo en la memoria y no sobrevivirá a una falla del servidor.

Colas de mensajes fallidos

Microsoft Service Bus

Las colas de *Azure Service Bus* y las suscripciones a temas proporcionan una sub-cola secundaria, llamada cola de mensajes fallidos (*DLQ*). La cola de mensajes fallidos no se necesita crear explícitamente y no se puede eliminar o administrar independientemente de la entidad principal.

La finalidad de la cola de mensajes fallidos es mantener los mensajes que no se pueden entregar a ningún destinatario o los mensajes que no se pudieron procesar. Después, los mensajes se quitan de la cola de mensajes fallidos y se inspeccionan. Una aplicación podría, con ayuda de un operador, corregir los problemas y volver a enviar el mensaje, registrar el hecho de que se produjo un error y llevar a cabo medidas correctivas.

RabbitMQ

Los mensajes de una cola pueden ser de "*dead-lettered*"; es decir, se vuelve a publicar en otro *exchange* cuando ocurre alguno de los siguientes eventos:

- El mensaje es rechazado (*basic.reject* o *basic.nack*) con *requeue* = *false*:
- El *TTL* para el mensaje caduca
- Se supera el límite de longitud de cola.

Los intercambios de dead letter (*DLX*) son *exchange* normales. Pueden ser cualquiera de los tipos habituales y se declaran como de costumbre. Para cualquier cola dada, un *DLX* puede ser definido por los clientes usando los argumentos de la cola, o en el servidor usando políticas. En el caso de que tanto la política como los argumentos especifiquen un *DLX*, el especificado en los argumentos inválida el especificado en la política. No tiene una cola definida para ellos.

Configuración dinámica de la cola y el tema

Microsoft Service Bus

Las colas y los temas se pueden administrar dinámicamente a través de la *API* de administración. También hay algunas características de seguridad dentro de un *namespace*, por lo que necesitarías tener permisos para administrar el espacio de nombres para hacer esto.

RabbitMQ

En *Rabbit*, las colas y los intercambios se pueden configurar dinámicamente a través de la *API* en tiempo de ejecución o a través de los complementos de administración. Hay configuraciones de seguridad que determinan si una conexión podría crear una cola o intercambio.

Detección de duplicados

Microsoft Service Bus

La habilitación de la detección de duplicados ayuda a mantener el seguimiento del valor *MessageId* controlado por la aplicación de todos los mensajes enviados a una cola o un tema durante una ventana de tiempo específica. Si se envía algún mensaje nuevo con *MessageId* que se haya registrado durante el periodo de tiempo, se notifica como aceptado (la operación de envío se realiza correctamente), pero el mensaje recién enviado se ignora y descarta al instante. No se tiene en cuenta ninguna otra parte del mensaje que no sea *MessageId*. El control de aplicación del identificador es esencial, ya que es lo único que permite que la aplicación enlace el valor *MessageId* a un contexto de proceso empresarial desde el que se pueda reconstruir de manera predecible en caso de error.

RabbitMQ

No tiene un sistema como tal para encargarse de eliminar mensajes duplicados.

Reintentos

Microsoft Service Bus

Service Bus implementa reintentos mediante implementaciones de la clase base *RetryPolicy* . Todos los clientes de *Service Bus* exponen una propiedad *RetryPolicy* que puede establecerse en una de las implementaciones de la clase base *RetryPolicy*. Las implementaciones integradas son:

- La clase *RetryExponential*: Esto expone las propiedades que controlan el intervalo de interrupción, el número de reintentos y la propiedad *TerminationTimeBuffer* que se utiliza para limitar el tiempo total para que se complete la operación.
- La clase *NoRetry*: Se utiliza cuando los reintentos en el nivel de la *API* de *Service Bus* no son necesarios, como cuando otro proceso administra los reintentos como parte de una operación en lotes o de múltiples pasos.

RabbitMQ

El sistema que implementa *RabbitMQ* hace que cada vez que ocurre un fallo durante el consumo de mensaje este vuelve, por defecto, a ponerse al final de la cola. Si vuelve a fallar hará exactamente lo mismo, no tiene un sistema de reintentos como tal aunque puede ser desarrollador por el desarrollador.

Retraso en la entrega de mensajes

Microsoft Service Bus

Puede enviar mensajes a una cola o un tema para su procesamiento retrasado; por ejemplo, para programar un trabajo de forma que esté disponible, para que lo procese el sistema a una hora determinada.

RabbitMQ

No parece haber ninguna capacidad de retardo de mensajes y esto debería ser desarrollado por un desarrollador.

Expiración de mensaje

Microsoft Service Bus

La orden o consulta que un mensaje transmite a un receptor, casi siempre está sujeta a algún tipo de fecha límite de caducidad de nivel de aplicación. Después de dicha fecha límite, el contenido ya no se entrega o la operación solicitada ya no se ejecuta. Esto es compatible con la función de mensaje programado que está disponible tanto en la nube como en las instalaciones. La caducidad de cualquier mensaje individual se puede controlar configurando la propiedad del sistema *TimeToLive*, que especifica una duración relativa. La caducidad se convierte en un instante absoluto cuando el mensaje se pone en cola en la entidad.

RabbitMQ

Hay una propiedad de extensión en una cola cuando se declara, lo que le permite especificar el tiempo de vida de los mensajes en esa cola. Se debería establecer la propiedad "*x-message-tt*". Un mensaje que ha estado en la cola durante más tiempo que el *TTL* configurado se dice que está "muerto" . Se debe de tener en cuenta que un mensaje enrutado a varias colas puede "morir" en diferentes momentos, o en absoluto, en cada cola en la que reside. La muerte de un mensaje en una cola no tiene ningún impacto en la vida del mismo mensaje en otras colas. El servidor garantiza que los mensajes muertos no se entregarán utilizando *basic.deliver* (a un consumidor) o incluidos en una respuesta *basic.get-ok* (para operaciones de recuperación de una sola vez). Además, el servidor intentará eliminar los mensajes a su vencimiento basado en *TTL* o poco después.

Interoperabilidad

Soporte AMQP

AMQP 1.0 es un protocolo de mensajes a nivel de red, confiable y eficaz que se puede utilizar para crear aplicaciones de mensajes robustas y compatibles entre plataformas. El protocolo tiene un objetivo simple: definir la mecánica de la transferencia segura, confiable y eficaz de mensajes entre dos partes. Los mismos mensajes se codifican usando una representación de datos portátiles que permite que remitentes y receptores heterogéneos intercambien mensajes empresariales estructurados con la máxima fidelidad. A continuación se incluye un resumen de las características más importantes:

- Eficaz: *AMQP 1.0* es un protocolo orientado a la conexión que utiliza una codificación binaria para las instrucciones de protocolo y los mensajes empresariales transferidos por él. Incorpora avanzados esquemas de control de flujo para potenciar al máximo la utilización de la red y los componentes conectados. Es decir, el protocolo se ha diseñado para lograr un equilibrio entre eficacia, flexibilidad e interoperabilidad.
- Confiable: El protocolo *AMQP 1.0* permite que se intercambien los mensajes con un rango de garantías de confiabilidad, desde el "enviar y olvidar" a la entrega confiable y confirmada.
- Flexible: *AMQP 1.0* es un protocolo flexible que se puede usar para admitir distintas topologías. El mismo protocolo se puede utilizar para las comunicaciones cliente-a-cliente, cliente-a-agente y agente-a-agente.
- Independiente del modelo del agente: la especificación *AMQP 1.0* no impone ningún requisito en el modelo de mensajes utilizado por un agente. Esto significa que es posible que se pueda agregar fácilmente compatibilidad con *AMPQ 1.0* a agentes de mensajes existentes.

Microsoft Service Bus

La compatibilidad con AMQP 1.0 en *Azure Service Bus* implica que ahora puede sacar partido de sus características de encolamiento de *Service Bus* y de la publicación/suscripción de mensajería asíncrona desde una amplia variedad de plataformas mediante un eficaz protocolo binario. Además, puede desarrollar aplicaciones formadas por componentes creados con una mezcla de lenguajes, marcos y sistemas operativos.

RabbitMQ

RabbitMQ fue desarrollado originalmente para soportar *AMQP*. Como tal, este protocolo es el protocolo çentral.ªdmitido por el intermediario. Todas estas variantes son bastante similares entre sí, con versiones posteriores que ordenan partes poco claras o poco útiles de versiones anteriores.

Protocolos soportados

Microsoft Service Bus

Service Bus admite los protocolos estándar *AMQP 1.0 y HTTP/REST*.

RabbitMQ

Admite los siguientes protocolos:

- *AMQP 0-9-1, 0-9 y 0-8*, y extensiones
- *STOMP*
- *MQTT*
- *AMQP 1.0*
- *HTTP*

Seguridad

Conexión al bus de servicio

Microsoft Service Bus

Las aplicaciones tienen acceso a los recursos de *Azure Service Bus* mediante la autenticación con token de firma de acceso compartido (SAS). Con SAS, las aplicaciones presentan a *Service Bus* un token que se ha firmado con una clave simétrica que tanto el emisor del token como *Service Bus* conocen (es decir "compartida") y esa clave está directamente asociada a una regla que concede derechos de acceso específicos, como el permiso para recibir y escuchar o enviar mensajes. Las reglas de *SAS* se configuran en el espacio de nombres o directamente en entidades como una cola o tema, lo que permite un control de acceso específico.

RabbitMQ

Utiliza usuario y contraseña.

Control de Acceso y Autorización

Azure Service Bus

La autenticación de *SAS* le permite conceder acceso a un usuario a los recursos de *Service Bus* con derechos específicos. La autenticación SAS en *Service Bus* implica la configuración de una clave criptográfica con derechos asociados en un recurso de *Service Bus*. Los clientes pueden obtener acceso a ese recurso presentando un token *SAS*, que consta del *URI* del recurso al que se tiene acceso y una fecha de expiración firmada con la clave configurada. Puede configurar claves para *SAS* en un espacio de nombres de *Service Bus*. La clave se aplica a todas las entidades de mensajes de ese espacio de nombres. También puede configurar claves en temas y colas de *Service Bus*. *SAS* también es compatible con *Azure Relay*.

RabbitMQ

Utiliza comandos que permite definir las políticas de control de acceso. Son más complejas de utilizar que las de *Microsoft*.

Patrones de intercambio de mensajes

Patrón de consumidores de la competencia

Permite que varios consumidores simultáneos procesen los mensajes recibidos en el mismo canal de mensajería. Este patrón permite que un sistema procese varios mensajes simultáneamente a fin de optimizar el rendimiento, mejorar la escalabilidad y disponibilidad, y equilibrar la carga de trabajo C.3.

Figura C.3: Patrón de consumidores de la competencia

Microsoft Service Bus

Se puede implementar fácilmente.

RabbitMQ

También permite su implementación, es fácil encontrar tutoriales en *Python*.

Patrón de cola de prioridad

Clasifica por orden de prioridad las solicitudes enviadas a los servicios para que aquellas con una prioridad más alta se reciban y procesen más rápidamente que las que tienen una prioridad más baja. Este patrón es útil en aplicaciones que ofrecen garantías de nivel de servicio diferentes a los clientes individuales C.4.

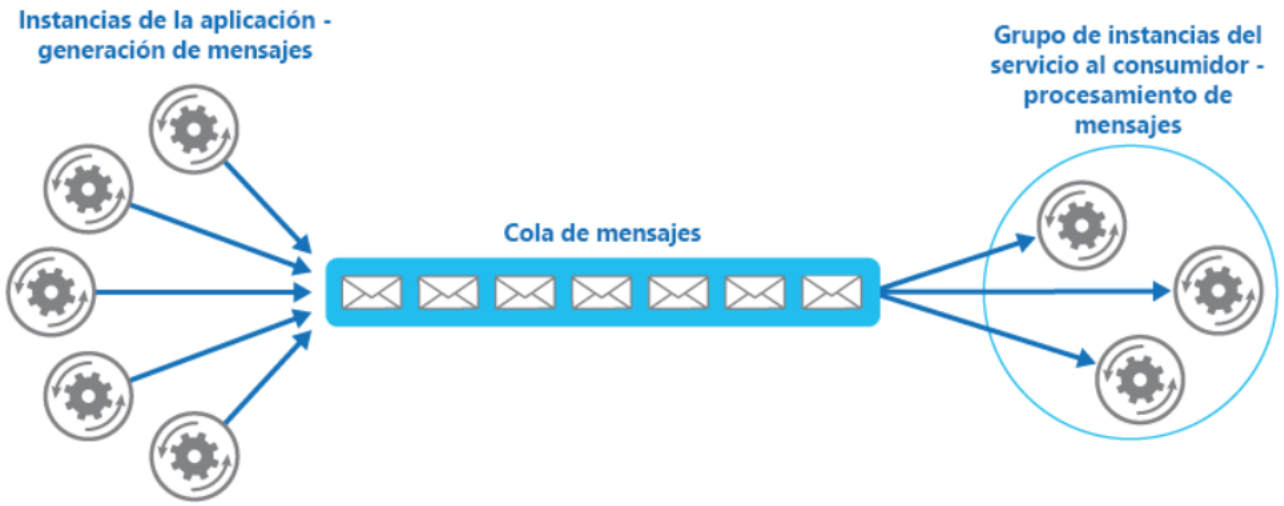

Figura C.4: Patrón de consumidores de la competencia

Microsoft Service Bus

Microsoftt Azure no proporciona un mecanismo de puesta en cola que admita de forma nativa la asignación de prioridad a los mensajes a través de la ordenación. Sin embargo, proporciona temas y suscripciones de *Azure Service Bus* que admiten un mecanismo de puesta en cola que ofrece filtrado de mensajes, además de un amplio conjunto de funcionalidades flexibles que lo hacen ideal para usarlo en las implementaciones de colas con la máxima prioridad.

RabbitMQ

RabbitMQ tiene implementación de colas de prioridad en el núcleo a partir de la versión 3.5.0. Cualquier cola se puede convertir en una prioridad usando argumentos opcionales proporcionados por el cliente (pero, a diferencia de otras características que usan argumentos opcionales, no políticas). La implementación admite un número limitado de prioridades: 255. Se recomiendan valores entre 1 y 10.

Coste y fuente abierta

Azure Service Bus

Es de código cerrado y el coste esta incluido en el servicio de *Azure*.

RabbitMQ

Gratuito

Conclusiones

En conclusión tras esta comparativa, ambos sistemas son bastante parecidos. Ambos manejan implementaciones similares y no hay una clara predominancia de uno frente al otro.

Finalmente, se implementara *Service Bus* por la facilidad de integración con *Azure* y *Microsoft Dynamics 365*.

Contenido del CD

El contenido que está incluido dentro del CD es el siguiente:

- La presente memoria en formato PDF.
- Manual de instalación.
- Manual de usuario.
- Solución de proyecto CallToCrmToOracle donde están incluidos:
  - CallToCrmCreateUpdate
  - CallToCrmDelete
  - ColaManual
  - ConnectionToOracleCreate
  - ConnectionToOracleDelete
  - ConnectionToOracleUpdate
  - EnvioEmailColaMensajesFallidos
  - RegularizaciónDatosCrm
  - RetryAutomatica
- Solución TFG_Plugin donde está incluida la codificación del *Plugin*.
- Carpeta con el diagrama de clases de análisis y los diagramas de secuencia de análisis.
- Carpeta con el diagrama de clases de diseño y los diagramas de secuencia de diseño.
- Archivo creación de base de datos de Oracle llamado *export.sql*.