UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Migración de plantillas generadoras de informes usando tecnología de microservicios y generación automática de código

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Chen , Alfonso Xiwen

Tutor/a: Letelier Torres, Patricio Orlando

Cotutor/a externo: SUAREZ GRUESO, FRANCISCO MANUEL

CURSO ACADÉMICO: 2022/2023

Dedicatoria

A mis queridos padres, cuyo amor y guía han sido la brújula de mi vida. Este Trabajo Final de Grado es más que un hito académico; es un tributo a su perseverancia, su sacrificio y su fe inquebrantable en mis capacidades. Lo dedico a ellos, que, con cada lección aprendida y cada logro obtenido, me enseñaron el verdadero significado de la dedicación y el amor incondicional.

Agradecimientos

Ante todo, mi profundo agradecimiento a mi tutor, Patricio Letelier. Su conocimiento, sabiduría y paciencia han sido los cimientos sobre los que se construyó este proyecto. Su capacidad para inspirar, motivar y desafiar ha superado los confines de un aula, convirtiéndose en una invaluable lección de vida.

A mis amigos, a quienes debo momentos de alivio y risas en medio de noches de estudio y trabajo arduo. Su apoyo y cariño incondicionales han sido un faro de luz en los momentos más oscuros y me han dado la fuerza para seguir adelante.

A la empresa ADD Informática y todos mis compañeros, que me brindaron una oportunidad única para poner en práctica mis conocimientos y crecer tanto profesional como personalmente. Vuestra orientación y confianza han sido un verdadero regalo que siempre recordaré con cariño.

Finalmente, un agradecimiento más para mis padres. Gracias por ser mi primer y más importante maestro, por enseñarme a creer en mí mismo y por ser siempre mi apoyo más firme. Este logro es nuestro, porque se construyó con su amor y su apoyo.

Resumen

Este trabajo se ha realizado en el contexto de unas prácticas de empresa en el departamento de I+D+i de una empresa del sector sociosanitario. La empresa está desarrollando una nueva versión de su ERP y este trabajo se ha centrado en el desarrollo de un proceso de migración de plantillas generadoras de informes. Estas plantillas son documentos que contienen una serie de variables que permiten diseñar la estructura y contenido de un informe.

La implementación del proyecto se ha realizado mediante la adopción de una estrategia de Desarrollo Dirigido por Modelos (MDD por sus siglas en inglés), haciendo uso de la tecnología DSL Tools para crear los modelos necesarios para el proceso de migración. Este enfoque ha permitido generar automáticamente partes del código a partir de los modelos, lo que ha mejorado la eficiencia y la calidad del desarrollo.

Para trabajar con los documentos de las plantillas, se ha utilizado el SDK de Open XML, que permite manipular documentos en formato WordprocessingML. Este SDK ha sido fundamental para leer y procesar las plantillas.

ASP.NET Core ha sido la tecnología seleccionada para el desarrollo del microservicio, un marco de trabajo robusto y flexible para el desarrollo de aplicaciones web. Se ha utilizado DSL Tools como instrumento para la construcción de los modelos, que ha permitido crear modelos precisos y completos que representan los conceptos y las relaciones del dominio del problema. El lenguaje de programación utilizado ha sido C#, un lenguaje moderno y potente que ofrece una amplia gama de características para el desarrollo de software.

Palabras clave: Desarrollo dirigido por Modelos, DSL Tools, WordProcessingML, Migración de plantillas, Informes, OpenXml SDK

Resum

Aquest treball s'ha realitzat en el context d'unes pràctiques d'empresa en el departament de R+D+i d'una empresa del sector sociosanitari. L'empresa està desenvolupant una nova versió de la seua ERP i aquest treball s'ha centrat en el desenvolupament d'un procés de migració de plantilles generadores d'informes. Aquestes plantilles són documents que contenen una sèrie de variables que permeten dissenyar l'estructura i contingut d'un informe.

La implementació del projecte s'ha realitzat mitjançant l'adopció d'una estratègia de Desenvolupament Dirigit per Models (MDD per les seues sigles en anglés), fent ús de la tecnologia DSL Tools per a crear els models necessaris per al procés de migració. Aquest enfocament ha permés generar automàticament parts del codi a partir dels models, la qual cosa ha millorat l'eficiència i la qualitat del desenvolupament.

Per a treballar amb els documents de les plantilles, s'ha utilitzat el SDK de Open XML, que permet manipular documents en format WordprocessingML. Aquest SDK ha sigut fonamental per a llegir i processar les plantilles.

ASP.NET Core ha sigut la tecnologia seleccionada per al desenvolupament del microservei, un marc de treball robust i flexible per al desenvolupament d'aplicacions web. S'ha utilitzat DSL Tools com a instrument per a la construcció dels models, que ha permés crear models precisos i complets que representen els conceptes i les relacions del domini del problema. El llenguatge de programació utilitzat ha sigut C#, un llenguatge modern i potent que ofereix una àmplia gamma de característiques per al desenvolupament de programari.

Palabras clave: Desenvolupament dirigit per Models, DSL Tools, WordProcessingML, Migració de plantilles, Informes, OpenXml SDK

Abstract

This work has been carried out in the context of an internship in the R&D&I department of a company in the health and social care sector. The company is developing a new version of its ERP and this work has focused on the development of a migration process of report-generating templates. These templates are documents that contain a series of variables that allow the structure and content of a report to be designed.

The project has been implemented by adopting a Model Driven Development (MDD) strategy, making use of the DSL Tools to create the necessary models for the migration process. This approach has allowed parts of the code to be automatically generated from the models, which has improved the efficiency and quality of the development.

To work with the template documents, the Open XML SDK has been used, which allows manipulation of documents in WordprocessingML format. This SDK has been essential for reading and processing the templates.

ASP.NET Core has been the selected technology for the development of the microservice, a robust and flexible framework for the development of web applications. DSL Tools have been used as a tool for the construction of the models, which has allowed the creation of accurate and complete models that represent the concepts and relationships of the problem domain. The programming language used was C#, a modern and powerful language that offers a wide range of features for software development.

Keywords: Model Driven Development, DSL Tools, WordProcessingML, Template Migration, Reporting, OpenXml SDK

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

1.1 Motivación

La evolución constante en el ámbito de la informática requiere que las empresas se mantengan al día para seguir siendo competitivas. Para lograr una mayor eficiencia y eficacia, muchas empresas buscan implementar nuevos sistemas de planificación de recursos empresariales (ERP) que se adapten mejor a sus necesidades.

En el departamento de I+D+i de una empresa del sector sociosanitario, se está desarrollando una nueva versión de su ERP. Para ello, se ha trabajado en la implementación de un sistema de migración de plantillas generadoras de informes en una arquitectura de microservicios. Este trabajo se ha llevado a cabo en el contexto de unas prácticas en empresa, donde se ha buscado desarrollar un ERP que cumpla con las necesidades específicas del sector.

Las plantillas generadoras de informes son esenciales para el proceso de *reporting*, el cual implica la recolección y presentación de datos en un formato estructurado que facilite su análisis y posterior toma de decisiones. Estas plantillas proporcionan un marco estandarizado para la presentación de datos, aspecto de crucial importancia en un entorno empresarial, donde la precisión y coherencia son vitales.

El producto final del proceso de *reporting* es un informe o *report*, el cual presenta los datos recolectados de manera organizada y fácilmente interpretable. Los informes pueden variar desde simples resúmenes de datos hasta análisis detallados con gráficos y tablas.

En la empresa del sector sociosanitario en cuestión, la creación de plantillas y generación de informes son funciones de suma importancia en el ERP que está siendo implementado. Estas funciones son ofrecidas a sus clientes, quienes pueden ser centros o instituciones relacionadas con la atención sociosanitaria.

Además, las plantillas de informes pueden ser altamente personalizables, permitiendo a las organizaciones adaptarlas a sus necesidades específicas. Esto puede abarcar desde la inclusión de su propia marca y estilo, hasta la inclusión de campos y secciones específicas relevantes para su operación.

Es imprescindible que la migración de datos, incluyendo las plantillas generadoras de informes, desde el ERP actual a la nueva versión, se realice de manera detallada y meticulosa. De este modo, se asegura la continuidad del negocio y la disponibilidad de información crítica en el nuevo sistema.

La migración de las plantillas generadoras de informes es esencial para la organización y sus clientes debido a diversas razones. En primer lugar, se estima que más del 80% de los datos documentados por las entidades cliente son procesados a través del módulo de generación de informes. En segundo lugar, se reciben diariamente diversas consultas con respecto al diseño de plantillas y la generación de informes, indicando que estas son áreas de gran relevancia y uso frecuente para los clientes.

Además, es importante destacar que el diseñador de plantillas de la nueva versión del ERP aún no está completamente implementado, lo que significa que los clientes necesitarán tener sus plantillas migradas para poder continuar con su trabajo una vez que se complete la transición a la nueva aplicación.

Finalmente, la migración anticipada de las plantillas permitirá a la organización y a sus clientes tener más tiempo para familiarizarse con el nuevo sistema y adaptarse a él. Por todas estas razones, la migración de las plantillas generadoras de informes es un proyecto esencial y necesario para la organización y sus clientes.

1.2 Objetivos

El objetivo principal del presente trabajo es diseñar e implementar un proceso de migración de plantillas generadoras de informes dentro del contexto compuesto por microservicios. Además del objetivo principal, se incluyen diversos requisitos u objetivos secundarios que deben ser alcanzados para asegurar su funcionamiento completo y adecuado:

- Desarrollar un mecanismo de control para evitar migrar las plantillas que ya han sido migradas previamente: se busca implementar un sistema de control que permita identificar las plantillas que ya han sido migradas, evitando así migraciones innecesarias y garantizando la consistencia de los datos en la nueva versión del ERP.

- Evaluar y mejorar el rendimiento del proceso de migración, identificando posibles áreas de optimización y proponiendo soluciones: Se realizarán pruebas exhaustivas para evaluar el rendimiento del proceso de migración de las plantillas generadoras de informes, identificando posibles cuellos de botella o puntos de mejora, y proponiendo soluciones para aumentar la eficiencia y la velocidad de migración.

- Implementar una estrategia de migración incremental que comienza con las plantillas con estructuras de campos más sencillas y progresa hacia estructuras de campos más complejas. Con este enfoque, buscamos garantizar una transición suave y minimizar los posibles errores durante la migración, proporcionando una mayor comprensión de los posibles problemas y cómo abordarlos a medida que la complejidad de las plantillas aumenta.

Al lograr estos objetivos, se establecerá un proceso sólido de migración de plantillas generadoras de informes basado en microservicios, permitiendo una transición exitosa hacia la nueva versión del ERP y mejorando la eficiencia en la generación de informes.

1.3 Estructura del documento

Esta memoria se organiza en los capítulos siguientes:

- El capítulo 1 introduce el trabajo que se va a desarrollar, explicando la motivación detrás de este, los objetivos que se buscan alcanzar y cómo se organiza el documento.

- El capítulo 2 se aborda el diseño de las plantillas en los dos ERP.

- El capítulo 3 ofrece un panorama general de las tecnologías actuales utilizadas en la gestión y manipulación de datos, y también se mencionan otras herramientas que podrían haber sido aplicadas en un proyecto de características similares.

- El capítulo 4 se dedica a analizar las tecnologías que se han implementado en el desarrollo de este proyecto.

- El capítulo 5 detalla el proceso de desarrollo de la solución propuesta. Comienza con la metodología empleada durante el proceso, seguido de una visión general del proyecto, una especificación de requisitos, el diseño y estructura de este. Además, incluye una sección sobre programación, las pruebas realizadas y finaliza con la cronología del proceso.

- El capítulo 6, el cual es el último, ofrece las conclusiones extraídas de este trabajo y también describe las actividades pendientes que aún deben completarse para culminar el proyecto hasta su potencial implementación.

CAPÍTULO 2

Plantillas generadoras de informes

En este apartado, se abordará una discusión detallada acerca de las plantillas generadoras de informes en las dos versiones del ERP, tanto la actual como la nueva. Desglosaremos cómo funcionan en cada versión, destacando las diferencias fundamentales y la evolución que han experimentado, con el objetivo de entender su rol vital en la migración y adaptación al nuevo sistema ERP. Este análisis nos ayudará a comprender el panorama completo y a apreciar la complejidad y los desafíos asociados a esta transición tecnológica.

2.1 Plantilla en el ERP actual

En el ERP actual, las plantillas de informes son diseñadas por los usuarios mediante un diseñador de plantillas basado en Word. Este enfoque requiere que los usuarios tengan Word instalado en sus sistemas, lo que permite una gran flexibilidad y familiaridad, ya que Word es una herramienta ampliamente utilizada y conocida.

Las plantillas permiten la inserción de uno o varios campos de las tablas de la base de datos. Estos campos pueden representar un solo registro o una colección de registros. Cada tabla puede contener campos normales o incluso otras tablas con sus propios campos, reflejando la naturaleza relacional de las bases de datos SQL.

Podemos ver un ejemplo de tabla anidada en la figura 1 donde “RESIDENTE” corresponde a la tabla raíz y “DIRECCIÓN” es una de sus tablas anidadas. Por otra parte, para insertar el campo es necesario marcar el cuadrado relacionado con cada valor:

Figura 1. Ejemplo Tabla anidada

Sin embargo, existen algunas peculiaridades y desafíos asociados con este sistema. En particular, los datos de las tablas anidadas no siempre se recogen correctamente. Esto puede ser un problema cuando se intenta generar informes que incluyen información de varias tablas relacionadas.

Además, la aplicación de filtrado en los datos es propensa a fallos. Aunque los usuarios tienen la opción de aplicar varios criterios de filtrado, como ordenar los registros en orden ascendente o descendente, o aplicar criterios de selección, estos filtros no siempre funcionan como se espera.

Este nivel de personalización permite a los usuarios diseñar informes que se ajusten exactamente a sus necesidades, lo que es especialmente útil en el sector sociosanitario, donde los requisitos de informes pueden variar ampliamente. Sin embargo, estos desafíos y peculiaridades también plantean problemas en el contexto de la migración a una nueva versión del ERP, ya que cada plantilla debe ser cuidadosamente transformada para garantizar que sigue funcionando correctamente en el nuevo sistema.

En la figura 2 se muestra un ejemplo de una plantilla del ERP actual. Los campos subrayados en gris representan aquellos que han sido insertados. Podemos observar el nombre del campo, seguido por la cardinalidad. En este contexto, '1' representa un campo específico, mientras que 'N' indica una sección repetible. Si se trata de una tabla anidada, el siguiente valor será el nombre de la tabla anidada. Al extremo derecho, encontramos la tabla raíz a la que pertenece el campo. Este formato brinda una visión clara de la estructura y el funcionamiento de las plantillas dentro del sistema ERP actual.

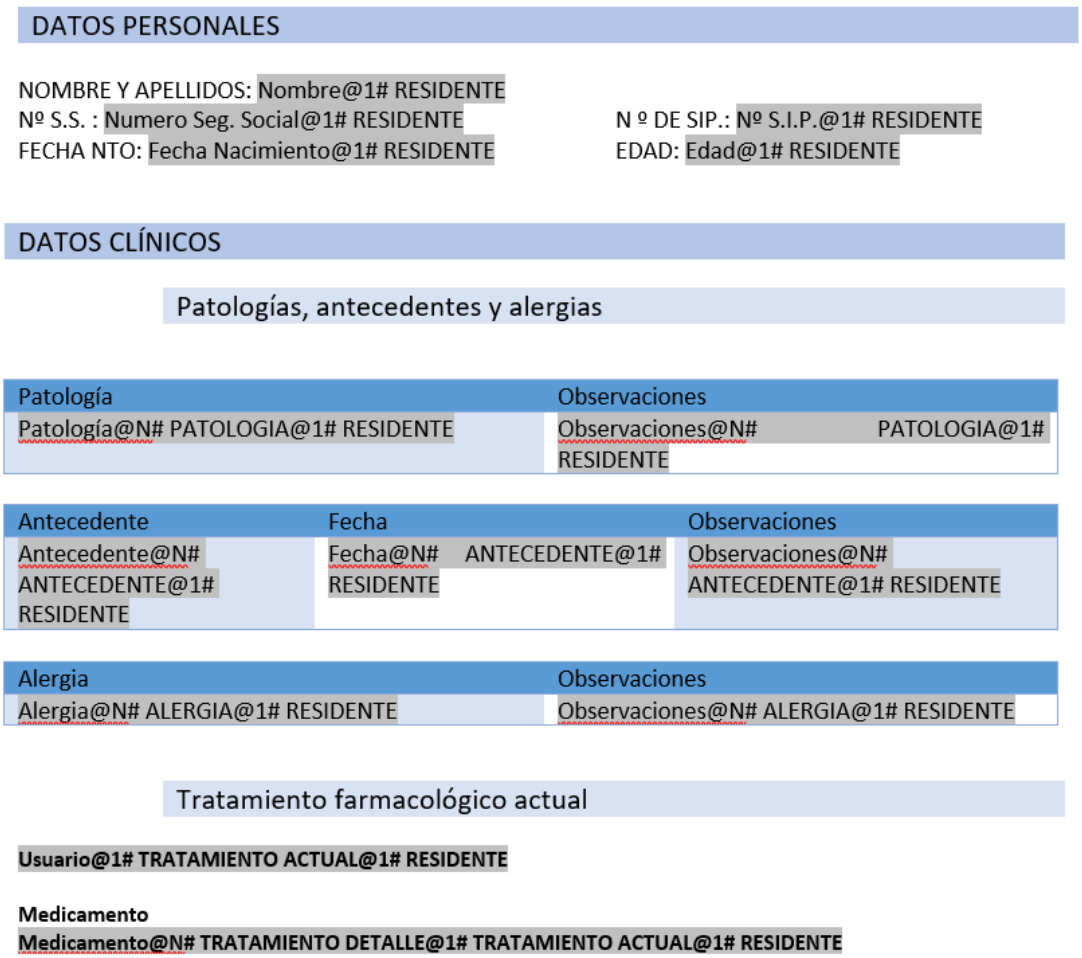

Figura 2.Ejemplo de plantilla en ERP actual

2.2 Plantilla en la nueva versión del ERP

En la nueva versión del ERP, las plantillas de informes, ahora conocidas como "Templates", son diseñadas por los usuarios de una manera diferente a la del sistema anterior. En lugar de utilizar Word, los usuarios ahora diseñan las plantillas mediante un formulario de Windows Presentation Foundation (WPF[1]), una tecnología de Microsoft para la creación de interfaces de usuario en aplicaciones de Windows.

Para trabajar con documentos, se utiliza DevExpress WPF Rich Text Editor[2], una herramienta que permite la edición de documentos de texto enriquecido en aplicaciones WPF. Esta herramienta es compatible con WordprocessingML, un formato de archivo basado en XML para documentos, que es parte de la especificación OpenXML. OpenXML es un conjunto de estándares de formato de archivo abierto para documentos de oficina, como documentos de Word, hojas de cálculo de Excel y presentaciones de PowerPoint.

En cuanto a los datos que se pueden incluir en las plantillas, los campos de las tablas y los criterios de filtrado ahora se representan como diferentes tipos de Data Transfer Objects (DTO). Los DTO son objetos que se utilizan para encapsular datos y transferirlos entre procesos o sistemas. Cada campo en un DTO hace referencia a un campo en una entidad de dominio, que es una representación de un objeto o concepto del mundo real en el sistema, de modo que cambia la inserción de datos.

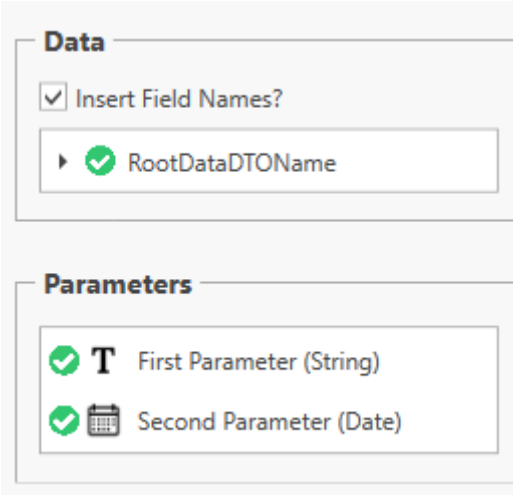

Figura 3. Ejemplo de campos en una plantilla en la nueva versión del ERP

La figura 3 ilustra los campos de la base de datos que se pueden insertar en una plantilla en la nueva versión del ERP. Cada bloque representa un DTO con sus respectivos campos. Para insertar un campo, el usuario simplemente realiza un doble

[1] WPF: https://visualstudio.microsoft.com/es/vs/features/wpf/
[2] DevExpress RichTextEditor: https://docs.devexpress.com/WPF/8651/controls-and-libraries/rich-text-editor

clic en el campo deseado, tras lo cual aparece un tick verde junto al campo para confirmar su correcta inserción.

Una de las principales ventajas de este nuevo sistema es que se aplica la seguridad en una plantilla de la misma manera que en un formulario en la nueva versión del ERP, algo que no sucede con las plantillas en el ERP actual. Esto proporciona una mayor consistencia y seguridad en el manejo de los datos.

Otra ventaja significativa es que los informes se crean utilizando la misma arquitectura y obteniendo los datos de la misma forma que la interfaz de usuario, a través de una API. Esto facilita el testeo unitario y de integración, algo que no era posible con las plantillas del Diseñador de Plantillas en el ERP actual.

Sin embargo, este nuevo sistema también plantea nuevos desafíos en el contexto de la migración de las plantillas desde el sistema ERP antiguo. Cada plantilla debe ser transformada de manera que los campos y criterios de filtrado en la plantilla original se mapeen correctamente a los DTO en la nueva plantilla.

En la figura 4 se muestra un ejemplo de una plantilla del ERP en desarrollo. Los valores entre llaves "{}" representan los campos de los DTO. Como se puede observar, estos campos pueden corresponder tanto a tablas completas como a campos concretos, lo que proporciona una gran flexibilidad y adaptabilidad en la creación de informes personalizados.

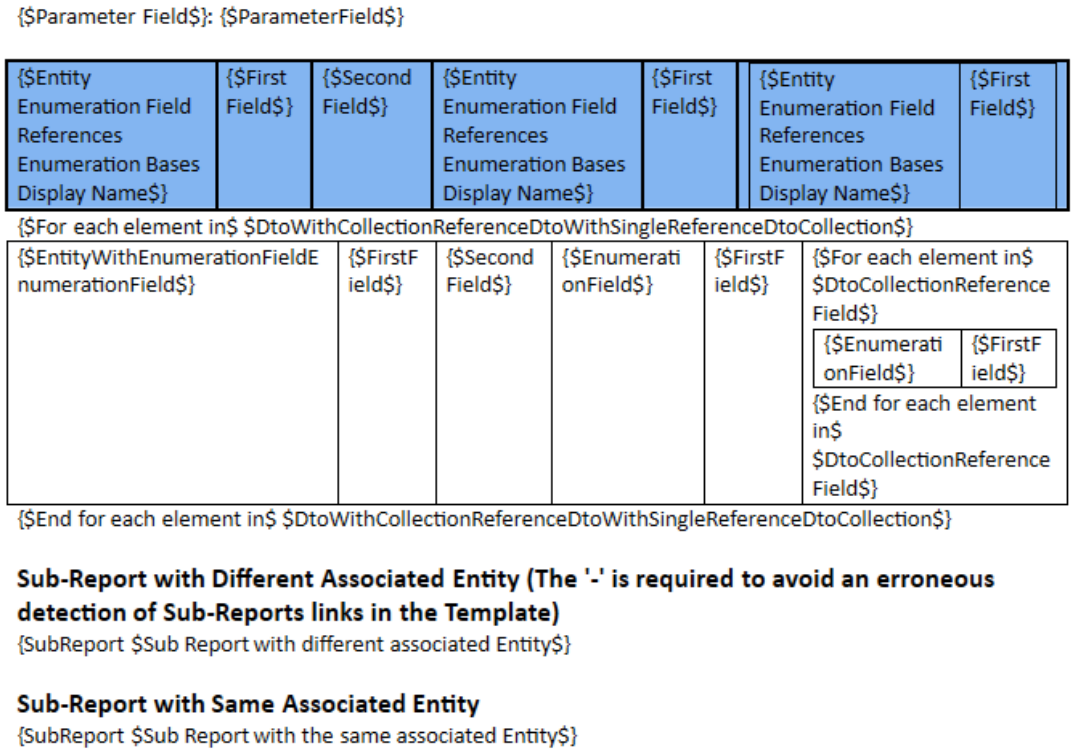

Figura 4. Ejemplo plantilla en nueva versión del ERP

CAPÍTULO 3

Estado del arte

En el ámbito de la informática y la gestión de datos, existen numerosas herramientas y tecnologías diseñadas para facilitar las tareas de migración y transformación de datos. Estas herramientas, que van desde soluciones de software comerciales hasta plataformas de código abierto, ofrecen una variedad de funcionalidades para extraer datos de una o más fuentes, transformar estos datos según sea necesario, y luego cargar los datos transformados en un nuevo sistema o base de datos.

Sin embargo, aunque estas herramientas pueden ser extremadamente útiles y potentes, no siempre se adaptan totalmente a las necesidades específicas de un proyecto. Cada herramienta tiene sus propias fortalezas y debilidades, y puede ser más o menos adecuada para diferentes tipos de proyectos, dependiendo de factores como la naturaleza y el volumen de los datos a migrar, las capacidades de procesamiento del sistema de destino, y los requisitos específicos del proyecto.

3.1 Herramientas existentes

3.1.1 Talend

Talend[3] es una plataforma de integración de datos que proporciona una variedad de herramientas y funcionalidades para ayudar a las organizaciones a gestionar, integrar, migrar y transformar sus datos. Talend es conocido por su flexibilidad, ya que es compatible con una amplia gama de fuentes de datos y plataformas de destino, y ofrece tanto soluciones en la nube como en las instalaciones.

Migración de datos con Talend [1]

La migración de datos con Talend generalmente implica los siguientes pasos:

1. **Conexión a las fuentes de datos**: Talend permite conectar a una amplia variedad de fuentes de datos, incluyendo bases de datos relacionales, archivos planos, servicios web, y más.

[3] Talend: https://www.talend.com/

2. **Extracción de datos**: Una vez establecida la conexión, puedes extraer los datos de la fuente utilizando los componentes de entrada de Talend.

3. **Carga de datos**: Después de extraer los datos, puedes cargarlos en la plataforma de destino utilizando los componentes de salida de Talend.

Transformación de datos con Talend [2]

La transformación de datos es un aspecto clave de la integración de datos, y Talend proporciona una variedad de componentes y funcionalidades para transformar los datos según sea necesario. Algunas de las transformaciones que puedes realizar con Talend incluyen:

- **Limpieza de datos**: Talend ofrece componentes para limpiar los datos, como la eliminación de duplicados, la corrección de errores de entrada, y la normalización de datos.

- **Enriquecimiento de datos**: Se puede utilizar Talend para enriquecer tus datos, por ejemplo, agregando datos de otras fuentes, calculando nuevos campos, o realizando análisis de datos.

- **Reestructuración de datos**: Talend también permite reestructurar los datos, por ejemplo, cambiando el formato de los datos, transformando los tipos de datos, o reorganizando la estructura de los datos.

Talend utiliza un enfoque basado en componentes, lo que significa que puedes arrastrar y soltar componentes en un lienzo para diseñar tus flujos de trabajo de integración de datos. Cada componente realiza una tarea específica, como la conexión a una base de datos, la extracción de datos, la transformación de datos, o la carga de datos. Esto hace que Talend sea muy flexible y fácil de personalizar según las necesidades de tu proyecto.

3.2.2 Oracle Data Integrator

Oracle Data Integrator (ODI)[4] es una plataforma de integración de datos completa que facilita la migración y transformación de datos. ODI proporciona una variedad de características y funcionalidades que permiten a los usuarios extraer datos de una variedad de fuentes, transformarlos según sea necesario, y luego cargarlos en un sistema de destino.

[4] ODI: https://www.oracle.com/es/middleware/technologies/data-integrator.html

ODI permite realizar la migración y transformación de datos [3] de la siguiente manera:

1. **Extracción de Datos**: ODI puede extraer datos de una amplia variedad de fuentes, incluyendo bases de datos relacionales, archivos planos, XML, y más. Esto se realiza a través de la funcionalidad de "conectores" de ODI, que son módulos de software que permiten a ODI interactuar con diferentes tipos de sistemas de almacenamiento de datos.

2. **Carga de Datos**: Una vez que los datos han sido transformados, ODI los carga en el sistema de destino. Esto puede ser cualquier tipo de sistema de almacenamiento de datos, incluyendo bases de datos relacionales, Data Warehouses, y más.

   ODI proporciona una interfaz gráfica de usuario que permite a los usuarios diseñar y configurar sus procesos de migración y transformación de datos. Esto incluye la creación de "mappings" que definen cómo se deben transformar los datos, y "workflows" que definen el orden en que se deben ejecutar las diferentes etapas del proceso.

3. **Transformación de Datos**: Una vez que los datos han sido extraídos, ODI puede transformarlos según sea necesario. Esto puede incluir la limpieza de datos (por ejemplo, la eliminación de datos duplicados o incorrectos), la transformación de formatos de datos (por ejemplo, la conversión de fechas o números a un formato estándar), y la combinación de datos de diferentes fuentes.

   ODI utiliza un enfoque de "ELT" (Extracción, Carga, Transformación) en lugar del enfoque tradicional de "ETL" (Extracción, Transformación, Carga). Esto significa que los datos se cargan en el sistema de destino antes de ser transformados, lo que puede resultar en un rendimiento mejorado en ciertos escenarios.

En resumen, Oracle Data Integrator es una herramienta poderosa y flexible para la migración y transformación de datos. Su enfoque ELT y su amplia compatibilidad con diferentes sistemas de almacenamiento de datos lo hacen adecuado para una variedad de escenarios de integración de datos.

3.2.3 Microsoft SQL Server Integration Services (SSIS)

Microsoft SQL Server Integration Services (SSIS)[5] es una plataforma para la integración de datos y la transformación de datos en el entorno de Microsoft. SSIS es una parte de Microsoft SQL Server, que es un sistema de gestión de bases de datos relacional.

SSIS proporciona una variedad de características [4] que facilitan la migración y transformación de datos, incluyendo:

- **Extracción, Transformación y Carga (ETL)**: SSIS es una herramienta ETL, lo que significa que puede extraer datos de una variedad de fuentes, transformar los datos en el camino y luego cargar los datos transformados en un destino. Las transformaciones pueden incluir operaciones como la limpieza de datos, la agregación de datos, la fusión de datos de diferentes fuentes, y más.

- **Flujos de trabajo**: SSIS permite a los usuarios diseñar flujos de trabajo de integración de datos utilizando una interfaz gráfica. Estos flujos de trabajo, conocidos como "paquetes", pueden incluir una variedad de tareas, como la ejecución de consultas SQL, la ejecución de scripts, la transferencia de archivos, y más.

- **Conectividad**: SSIS proporciona conectores para una variedad de fuentes de datos, incluyendo bases de datos SQL y NoSQL, archivos CSV y Excel, servicios web, y más.

- **Programación**: SSIS permite a los usuarios escribir scripts y código personalizado para realizar transformaciones de datos complejas o tareas de integración de datos.

Para realizar una migración de datos con SSIS [5], los pasos típicos serían los siguientes:

1. **Crear un nuevo paquete SSIS**: Un paquete es esencialmente un flujo de trabajo que define cómo se extraen, transforman y cargan los datos.

2. **Definir la fuente de datos**: Esto implica configurar la conexión a la base de datos o archivo de donde se extraerán los datos.

3. **Definir las transformaciones**: Esto implica configurar las transformaciones que se aplicarán a los datos. Esto puede incluir la limpieza de datos, la conversión de tipos de datos, la agregación de datos, y más.

[5] SSIS: https://learn.microsoft.com/es-es/sql/integration-services/sql-server-integration-services?view=sql-server-ver16

4. **Definir el destino de los datos**: Esto implica configurar la conexión a la base de datos o archivo donde se cargarán los datos transformados.

5. **Ejecutar el paquete**: Una vez que el paquete está configurado, se puede ejecutar para realizar la migración de datos.

Es importante tener en cuenta que SSIS es una herramienta poderosa y flexible, pero también puede ser compleja. Requiere un buen entendimiento de los conceptos de integración de datos y una cierta experiencia con el desarrollo de software y las bases de datos.

3.2 Comparación

En la figura 5, se presenta una comparación de las herramientas de integración de anteriormente mencionadas. Las herramientas se comparan en términos de precio, destino de almacenamiento final, personalización y tiempo de desarrollo. Es importante tener en cuenta que estos son valores aproximados y pueden variar dependiendo de factores específicos como el tamaño de la empresa, los requerimientos del proyecto, y otros.

	Precio	Destino de almacenamiento	Personalización	Tiempo de desarrollo
Talend	Existe una versión gratuita y otra de pago (basada en suscripciones)	Variedad de bases de datos y almacenes de datos	Alta (a través de su interfaz gráfica y scripting)	Medio (requiere configuración y desarrollo de flujos de trabajo)
Oracle Data Integrator (ODI)	Alto (basado en licencias y servicios)	Variedad de bases de datos y almacenes de datos, incluyendo Oracle Database	Alta (a través de su interfaz gráfica y scripting)	Medio (requiere configuración y desarrollo de flujos de trabajo)
Microsoft SQL Server Integration Services (SSIS)	Incluido con SQL Server, pero SQL Server tiene un costo (basado en licencias)	SQL Server y otros destinos compatibles	Alta (a través de su interfaz gráfica y scripting)	Medio (requiere configuración y desarrollo de flujos de trabajo)

Figura 5. Tabla comparativa de herramientas

3.3 Conclusiones

Actualmente, estamos en medio de un proceso de migración de datos desde un producto de software existente, que es utilizado por miles de usuarios, a un nuevo producto que se encuentra en fase de desarrollo. Este es un proyecto significativo y complejo, dada la magnitud de datos involucrados y la imperante necesidad de garantizar que la migración se realice de manera eficiente y sin pérdida de datos.

El nuevo producto se está desarrollando con una arquitectura basada en microservicios. Esta arquitectura divide la funcionalidad del software en servicios más pequeños y autónomos que pueden desarrollarse, implementarse y escalarse de manera independiente. Esta arquitectura ofrece varias ventajas, incluyendo una mayor flexibilidad, escalabilidad y facilidad de mantenimiento.

Dada la arquitectura basada en microservicios del nuevo producto, se ha decidido utilizar un enfoque similar para la migración y transformación de datos. Esto implica la creación de microservicios específicos para la migración y transformación de datos. Estos microservicios se encargarán de extraer los datos del producto de software existente, transformarlos según sea necesario para el nuevo producto, y luego cargarlos en el nuevo sistema.

Este enfoque permite que la migración y transformación de datos se realice de manera más eficiente y escalable. Cada microservicio puede desarrollarse y escalarse de manera independiente, lo que permite un mayor control sobre el proceso de migración y transformación. Además, este enfoque también facilita la detección y corrección de problemas, ya que cada microservicio puede ser monitoreado y depurado de manera independiente.

Existen varias razones por las que se prefiere desarrollar nuestros propios microservicios para la migración y transformación de datos en lugar de utilizar herramientas existentes:

- **Personalización**: Al desarrollar nuestros propios microservicios, tenemos control total sobre su funcionalidad y podemos personalizarlos para satisfacer exactamente las necesidades del proyecto. Las herramientas existentes pueden no ofrecer el nivel de personalización que necesitamos o pueden incluir funcionalidades adicionales que no necesarios y que podrían complicar el proceso.

- **Integración**: Como ya se está desarrollando una arquitectura de microservicios en la nueva versión del ERP, desarrollar nuestros propios microservicios para la migración y transformación de datos puede facilitar su integración con el resto de tu sistema. Las herramientas existentes pueden no integrarse tan fácilmente con la arquitectura actual.

- **Costo**: Aunque las herramientas existentes pueden ofrecer funcionalidades avanzadas, también pueden ser costosas. Desarrollar nuestros propios microservicios puede ser más rentable, especialmente si ya se tiene el conocimiento y la experiencia necesarios en tu equipo.

- **Aprendizaje y crecimiento**: Desarrollar nuestros propios microservicios puede ser una excelente oportunidad para aprender y crecer como desarrollador o equipo de desarrollo. Ya que permite adquirir experiencia en áreas como la arquitectura de microservicios, la migración de datos y la transformación de datos.

- **Control sobre los datos**: Al utilizar nuestros propios microservicios, mantenemos un control total sobre los datos. No hay que preocuparse por la seguridad de los datos o por la conformidad con las regulaciones de privacidad de datos al transferir los datos a una herramienta externa.

CAPÍTULO 4

Tecnología utilizada

En primer lugar, ASP.Net Core[6] y C# son las tecnologías de back-end principales que se ha empleado para desarrollar la funcionalidad del presente proyecto. Estas dos tecnologías junto con Visual Studio como entorno de desarrollo integrado (IDE), han sido la columna vertebral del trabajo, permitiendo construir microservicios robustos y eficientes.

Para la administración del código y la colaboración del trabajo, se ha utilizado Azure DevOps Server[7], plataforma de la cual mantiene el código en un repositorio seguro, siguiendo un proceso de integración y despliegue continuos (CI/CD), y colaborando de manera eficiente con otros.

Por último, cabe destacar que se ha empleado las Herramientas de Lenguaje Específico de Dominio (DSL Tools) que han permitido la creación de lenguajes específicos de dominio, facilitando un enfoque de Desarrollo Dirigido por Modelos (MDD) y la generación automática de código. Además, se ha empleado unas bibliotecas de cara a la migración. La biblioteca Moq[8] ha sido utilizada para las pruebas unitarias, permitiendo la creación de objetos simulados para probar interacciones entre componentes. Newtonsoft[9] ha facilitado la serialización y deserialización de objetos JSON, esencial para la comunicación entre diferentes partes del sistema y la generación de modelos. Finalmente, las bibliotecas OpenXML SDK[10] y OpenXmlPowerTools[11] han permitido trabajar con documentos en formato OpenXML, el formato utilizado por las plantillas en la nueva versión del ERP, facilitando su lectura, manipulación y transformación.

4.1 DSL Tools

Las herramientas de Lenguaje Específico de Dominio (DSL Tools) son un conjunto de utilidades que permiten a los desarrolladores definir y trabajar con lenguajes específicos de dominio. Un lenguaje específico de dominio es un lenguaje de programación o especificación que está diseñado para resolver problemas en un dominio particular [6].

[6] ASP.Net Core: https://learn.microsoft.com/es-es/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0
[7] Azure DevOps Server: https://azure.microsoft.com/en-us/products/devops/
[8] Moq: https://github.com/moq/moq
[9] NewtonSoft: https://www.newtonsoft.com/json
[10] OpenXML SDK: https://github.com/dotnet/Open-XML-SDK
[11] OpenXMLPowerTools: https://github.com/OfficeDev/Open-Xml-PowerTools

En el contexto de este trabajo, las DSL Tools se utilizan para facilitar un enfoque de Desarrollo Dirigido por Modelos (MDD). En MDD, los modelos de alto nivel se utilizan para definir la estructura y el comportamiento de un sistema de software. Estos modelos luego se traducen automáticamente en código a través de plantillas de generación de código.

Las DSL Tools pueden ser muy útiles en este proceso, ya que permiten a los desarrolladores definir lenguajes que se ajusten a los modelos de dominio específicos que están utilizando. Esto puede hacer que el proceso de definición de modelos y la generación de código sea más eficiente y menos propenso a errores.

Algunas de las ventajas de utilizar las DSL Tools en un enfoque MDD incluyen [7]:

- **Eficiencia**: Al automatizar la generación de código, los desarrolladores pueden ahorrar tiempo y evitar errores comunes. Esto puede hacer que el proceso de desarrollo sea más rápido y eficiente.

- **Consistencia**: Al utilizar plantillas de generación de código, se puede garantizar que el código generado sea consistente. Esto puede hacer que el código sea más fácil de entender y mantener.

- **Flexibilidad**: Las DSL Tools permiten a los desarrolladores definir lenguajes que se ajusten a sus necesidades específicas. Esto puede hacer que el proceso de modelado sea más flexible y adaptable.

- **Automatización de pruebas**: Con las DSL Tools, también es posible generar automáticamente pruebas unitarias a partir de los modelos. Esto puede mejorar la calidad del código y facilitar el proceso de prueba.

4.2 Moq

En el desarrollo de este proyecto, se ha hecho un uso extensivo de la biblioteca Moq para la realización de pruebas unitarias. Moq es una biblioteca para .NET que permite a los desarrolladores crear y configurar objetos simulados, o "mocks", que pueden ser utilizados para imitar el comportamiento de objetos reales en un entorno de prueba [8].

El uso de Moq es especialmente relevante en el contexto de las pruebas unitarias, ya que permite probar el comportamiento de los métodos de una clase en aislamiento, sin la necesidad de instanciar sus dependencias reales. Esto es especialmente útil cuando se trata de probar clases que dependen de servicios externos o recursos que pueden ser difíciles de instanciar en un entorno de prueba, como bases de datos o servicios web.

En este proyecto, se ha utilizado Moq para probar la funcionalidad implementada en varios microservicios, incluyendo el Migrador y el Transformer. Por ejemplo, se han creado objetos simulados de la base de datos para imitar la lectura de nombres de

plantillas y rutas. Esto ha permitido probar la funcionalidad del Migrador en aislamiento, sin la necesidad de conectar a una base de datos real.

Además, se han utilizado objetos simulados para imitar la transformación de plantillas en el microservicio Transformer. Esto ha permitido probar cómo se comporta el Transformer cuando se le proporcionan diferentes tipos de plantillas, y cómo maneja situaciones de error, como plantillas mal formadas o inexistentes.

El uso de Moq ha permitido a los desarrolladores de este proyecto escribir pruebas robustas y confiables que pueden ser ejecutadas rápidamente y en cualquier entorno, lo que ha facilitado la detección y corrección de errores durante el desarrollo.

4.3 Newtonsoft

La biblioteca Newtonsoft, también conocida como Json.NET, es una popular biblioteca de C# que proporciona funcionalidades para trabajar con JSON, el formato ligero de intercambio de datos. Esta biblioteca ofrece funcionalidades para serializar y deserializar objetos a y desde JSON, así como capacidades para manipular y consultar datos JSON [9].

Json.NET ofrece varias ventajas significativas que la han hecho extremadamente popular en el desarrollo de aplicaciones .NET. Una de las principales ventajas es su facilidad de uso. Json.NET proporciona una API intuitiva y fácil de usar que simplifica enormemente la tarea de trabajar con datos JSON en C#.

Otra ventaja de Json.NET es su rendimiento. Json.NET está optimizado para ser extremadamente rápido, lo que puede ser crítico en aplicaciones que necesitan procesar grandes cantidades de datos JSON.

Además, Json.NET es muy flexible, con soporte para convertir entre JSON y XML, LINQ a JSON para manipular y consultar datos JSON, y serialización personalizada para manejar casos más complejos.

La generación de modelos es un proceso crucial en este proyecto, ya que se utiliza para transformar las plantillas generadoras de informes al formato de la nueva versión del ERP. Dado que el microservicio de transformación está desarrollado en .NET 6.0 y la generación de modelos se realiza en .NET 4.7.2, se ha hecho necesario este mecanismo para pasar datos entre estos dos entornos. De esta manera, se puede garantizar una transferencia de datos fluida y eficiente entre diferentes partes del sistema, a pesar de las diferencias en las versiones de .NET utilizadas.

4.4 OpenXmlSDK y OpenXmlPowerTools

Dentro del ámbito de este proyecto, se ha hecho un uso intensivo de las bibliotecas OpenXML SDK y OpenXmlPowerTools para manejar las plantillas de informes. Estas bibliotecas son esenciales para trabajar con documentos en formato OpenXML, que es el formato que las plantillas en la nueva versión del ERP utilizan.

OpenXML es un formato de archivo que se utiliza en Microsoft Office y que consta de varios componentes. Estos componentes se representan utilizando WordprocessingML, un lenguaje de marcado que forma parte del estándar OpenXML y que se utiliza para representar documentos de Word. Los componentes de un documento WordprocessingML incluyen el Documento Principal, Comentarios, Pies de Página, Cabeceras, Definiciones de Estilo, entre otros. Cada uno de estos componentes se almacena en un archivo separado dentro del archivo OpenXML, lo que permite una gran flexibilidad y control sobre el contenido del documento [10].

OpenXML SDK es una biblioteca desarrollada por Microsoft que proporciona una interfaz programática para manipular documentos OpenXML. Esta biblioteca ofrece una amplia gama de clases y métodos que facilitan la creación, manipulación y lectura de documentos OpenXML. En este proyecto, OpenXML SDK se utiliza para leer las plantillas de Word existentes y convertirlas en un formato que la nueva versión del ERP puede entender [11].

OpenXmlPowerTools es una biblioteca que amplía las funcionalidades de OpenXML SDK, proporcionando herramientas adicionales para trabajar con documentos OpenXML. Esta biblioteca incluye funcionalidades para la manipulación de documentos, la generación de informes, la comparación de documentos y mucho más. En el contexto de este proyecto, OpenXmlPowerTools se utiliza para realizar operaciones más complejas en las plantillas, como la inserción de datos dinámicos o la modificación de estilos y formatos [12].

El uso de estas bibliotecas ofrece varias ventajas en este proyecto:

- **Compatibilidad nativa con las plantillas**: Al trabajar con las plantillas en su formato nativo, se facilita la migración de las plantillas a la nueva versión del ERP.

- **Flexibilidad**: Las bibliotecas proporcionan una gran flexibilidad para manipular los documentos, permitiendo realizar cambios en las plantillas de forma programática. Esto es especialmente útil para adaptar las plantillas a las necesidades específicas de la nueva versión del ERP.

- **Compatibilidad y soporte**: Al ser bibliotecas desarrolladas por Microsoft, ofrecen un alto nivel de compatibilidad y soporte, lo que garantiza su fiabilidad y durabilidad a largo plazo.

- **Manipulación de componentes específicos del documento**: Con estas bibliotecas, puedes manipular componentes específicos del documento OpenXML, como Comentarios, Pies de Página, Cabeceras, Definiciones de

Estilo, etc. Esto permite un control granular sobre el contenido y la presentación de las plantillas.

4.5 Microservicios

Los microservicios [13] representan un estilo de arquitectura de software que estructura una aplicación como una colección de servicios pequeños, autónomos y modulares. Cada uno de estos servicios se ejecuta en su propio proceso y se comunica con los demás a través de mecanismos bien definidos, generalmente una API HTTP/REST o un servicio de mensajería asíncrona [14], como podemos ver en la figura 6.

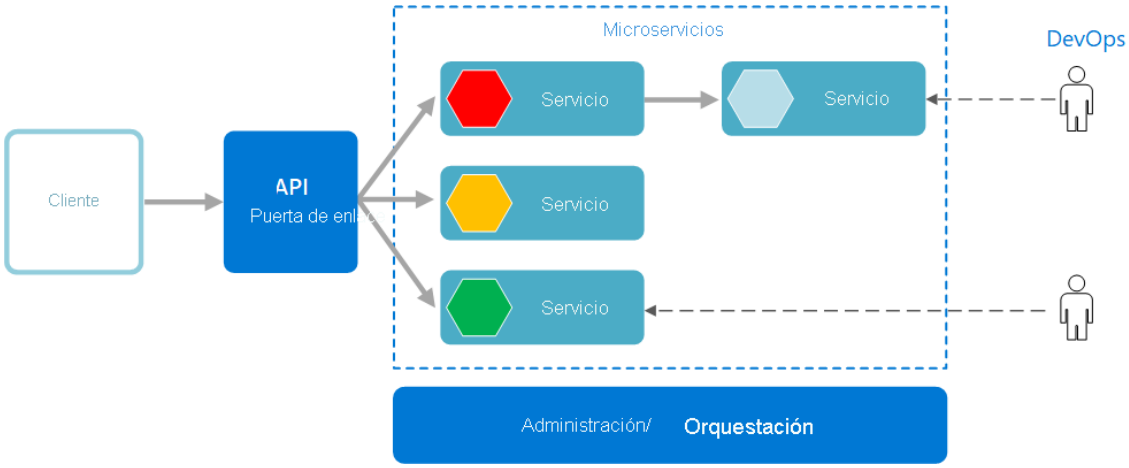

Figura 6.Estilo de arquitectura de microservicios [15]

En el contexto de este trabajo, la nueva versión del ERP que se está desarrollando se basa en una arquitectura de microservicios. Para operar de forma integrada, los diferentes componentes del ERP interactúan entre sí mediante el uso de la API de cada microservicio. Esta arquitectura permite un alto grado de modularidad y flexibilidad, facilitando la adaptación y evolución del sistema a medida que cambian los requisitos y las condiciones.

Las ventajas de utilizar microservicios como parte de este trabajo y en general incluyen [16]:

- **Desacoplamiento**: Los microservicios permiten desacoplar diferentes aspectos de la aplicación, lo que facilita la modificación y actualización de cada servicio de forma independiente sin afectar al resto de la aplicación.

- **Escalabilidad**: Cada microservicio puede escalarse de forma independiente según las necesidades de la aplicación. Esto es especialmente útil en aplicaciones empresariales como un sistema ERP, donde diferentes componentes pueden tener diferentes requisitos de carga de trabajo.

- **Resiliencia**: Si un microservicio falla, el resto de la aplicación puede seguir funcionando normalmente, mejorando la disponibilidad y la resistencia de la aplicación.

- **Flexibilidad tecnológica**: Cada microservicio puede utilizar su propia pila tecnológica, lo que permite seleccionar la tecnología más adecuada para cada servicio.

- **Despliegue independiente**: Los microservicios pueden desplegarse, actualizarse y escalarse de forma independiente, lo que facilita la gestión y el mantenimiento de la aplicación.

- **Optimización de recursos**: Los microservicios permiten optimizar el uso de recursos al permitir que cada servicio escale según sus propias necesidades.

En resumen, los microservicios ofrecen un enfoque poderoso y flexible para el desarrollo de aplicaciones de software. En el contexto de este TFG, los microservicios han demostrado ser una herramienta valiosa para la migración de plantillas generadoras de informes en el contexto de una transición de un sistema ERP a otro, facilitando la comunicación y la interoperabilidad entre las distintas partes del sistema.

CAPÍTULO 5

Desarrollo de la solución

5.1 Metodología

En este proyecto, hemos adoptado una metodología ágil para garantizar la eficiencia y la adaptabilidad en el desarrollo. La metodología ágil es un enfoque iterativo e incremental que enfatiza la flexibilidad, la colaboración y la satisfacción del cliente.

Para empezar, el trabajo se organiza en sprints, que son ciclos de trabajo de una o dos semanas de duración. Cada sprint comienza con una reunión de planificación en la que se establecen los objetivos y se seleccionan las tareas del backlog del producto que se van a abordar. Este backlog del producto se mantiene en OneNote[12], lo que facilita la colaboración y el seguimiento del progreso.

Durante cada sprint, se realiza un seguimiento constante del progreso y se mantienen reuniones regulares con los tutores y el director del departamento. Estas reuniones permiten discutir los avances, identificar posibles obstáculos y ajustar el plan de trabajo si es necesario.

Además, se utiliza un tablero Kanban para visualizar el flujo de trabajo. El tablero Kanban divide el trabajo en columnas que representan diferentes etapas del proceso, desde las tareas pendientes hasta las completadas. Esto ayuda a identificar cuellos de botella y a mantener un flujo de trabajo constante.

En el centro de este enfoque ágil se encuentra el director de nuestro departamento, quien ha asumido el rol de *Product Owner*. Su liderazgo ha permitido priorizar tareas, tomar decisiones informadas y mantener el proyecto alineado con las metas de la empresa. Ha guiado eficazmente las reuniones de planificación y revisión de cada sprint, garantizando la entrega de un producto que satisface las expectativas de los clientes.

En cuanto a la entrega, se sigue un enfoque incremental. Esto significa que se entregan partes funcionales del proyecto al final de cada sprint. Este enfoque permite obtener feedback temprano y realizar ajustes antes de que se invierta demasiado tiempo y esfuerzo en una dirección equivocada.

Finalmente, se realizan pruebas automatizadas para garantizar la calidad del software. Estas pruebas se ejecutan regularmente y permiten detectar y corregir errores rápidamente. Además, ayudan a asegurar que las nuevas características o cambios no rompan la funcionalidad existente.

[12] OneNote: https://www.microsoft.com/es-es/microsoft-365/onenote/digital-note-taking-app

En resumen, la adopción de una metodología ágil en este proyecto, liderada por el director del departamento en su papel de *Product Owner*, ha permitido un desarrollo más eficiente y adaptable, centrado en la mejora continua y la satisfacción del cliente.

5.2 Planteamiento general

Actualmente, la empresa lleva desarrollando un microservicio de migración, llamado "Migrator", que se encarga de migrar los datos de la base de datos del ERP actual a la nueva. Sin embargo, las plantillas de informes no están incluidas en este proceso, ya que en la base de datos solo se guarda la ruta donde los clientes han guardado las plantillas.

Para abordar este problema, se propone ampliar la funcionalidad del microservicio "Migrator" para que pueda obtener las plantillas de informes a partir de las rutas almacenadas en la base de datos. Una vez obtenidas las plantillas, se enviarán a un nuevo microservicio de transformación que se creará como parte de este proyecto.

El microservicio de transformación que recibe el nombre de “Transformer” se encargará de adaptar las plantillas para que sean compatibles con la nueva versión del ERP. Este proceso de adaptación incluirá varias etapas. Primero, se realizará un mapeo de los datos en las plantillas originales a los correspondientes campos y entidades en la nueva base de datos. Luego, se generarán los modelos de aplicación de plantillas, que servirán para la generación de código en el nuevo sistema ERP.

Una vez que los modelos de aplicación de plantillas estén listos, se llevará a cabo la transformación de las plantillas. Este proceso convertirá las plantillas en un formato que sea compatible con el nuevo sistema ERP y que refleje la estructura y los requisitos de los nuevos modelos de aplicación de plantillas.

Finalmente, las plantillas transformadas se enviarán al microservicio destino, que se encargará de guardar las plantillas en la base de datos del nuevo sistema ERP y también se notificará al microservicio “Migrator” del resultado de migración.

Este proyecto proporcionará una solución integral para la migración de plantillas de informes, garantizando que los clientes puedan seguir generando informes consistentes y precisos después de la transición al nuevo sistema ERP.

5.3 Especificación de requisitos

5.3.1 Casos de uso

A continuación, se detallarán las distintas funciones que se proyecta que la migración de plantillas pueda realizar, las cuales se ilustrarán a través de la figura 7 de un Diagrama de Casos de Uso. Cada uno de estos Casos de Uso estará marcado con un identificador único para facilitar su referencia, y vendrá acompañado de un nombre y una descripción adecuada.

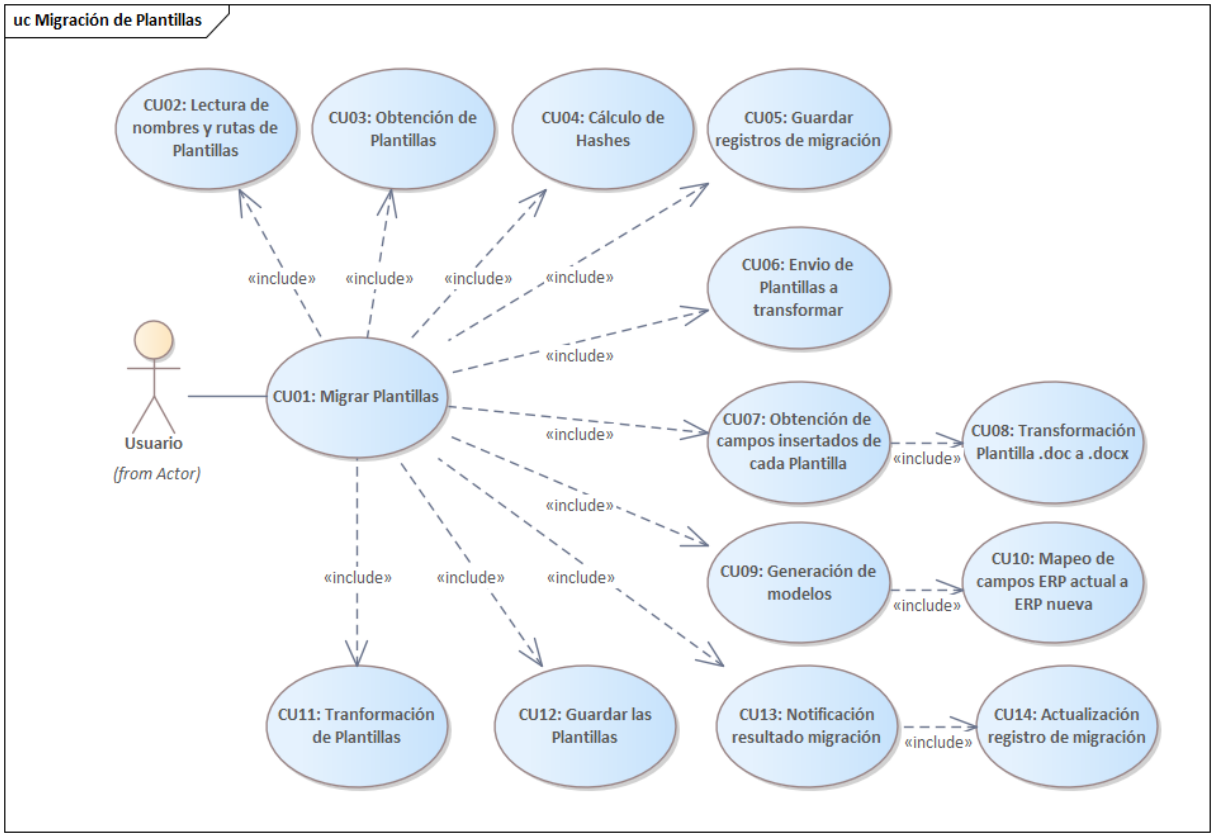

Figura 7. Diagrama de casos de uso

Identificador	CU01
Nombre	Migrar Plantillas
Actor	Usuario de la aplicación
Descripción	Un usuario del producto actual o el sistema inicia la migración de los plantillas para tener en la nueva versión del ERP las plantillas diseñadas en el ERP actual.

Identificador	CU02
Nombre	Lectura de nombres y rutas de Plantillas
Actor	Usuario de la aplicación
Descripción	Acceso a la base de datos para obtener los nombres y rutas donde se encuentran guardadas las plantillas. Ya que en el ERP actual no se guarda las plantillas sino la ruta donde se ubica ya sea en local o en el servidor. Esta información es necesaria para localizar las plantillas que se van a migrar.

Identificador	CU03
Nombre	Obtención de Plantillas
Actor	Usuario de la aplicación
Descripción	Obtención de las plantillas según la ruta donde se ubica, ya sea en local o en el servidor, depende de la ubicación que lo ha guardado los usuarios al diseñar una plantilla nueva. Estas pueden estar en formato .docx o .docx.

Identificador	CU04
Nombre	Cálculo de Hashes
Actor	Usuario de la aplicación
Descripción	Se calcula el hash de cada plantilla. Este hash se utiliza para identificar de manera única cada plantilla y evitar la migración repetida de las mismas plantillas. Que posteriormente se guardará en el histórico de migración.

Identificador	CU05
Nombre	Guardar registros de migración
Actor	Usuario de la aplicación
Descripción	Guarda el nombre de la plantilla y su hash correspondiente en el "TemplateMigrationRecord", entidad de dominio que corresponde al histórico de migraciones de plantillas. Esto proporciona un registro histórico de las migraciones que se han realizado para evitar duplicaciones de plantillas ya migradas.

Identificador	CU06
Nombre	Envío de Plantillas a transformar
Actor	Usuario de la aplicación
Descripción	Envío de las plantillas al microservicio de transformación responsable de transformar las plantillas para que sean compatibles con la nueva versión del ERP.

Identificador	CU07
Nombre	Obtención de campos insertados de cada Plantilla
Actor	Usuario de la aplicación
Descripción	Se realiza una lectura del contenido de cada plantilla para obtener las campos que corresponden a ser insertados de la base de datos del ERP actual, con el objetivo de identificar cuáles son los campos que necesitan ser transformados, y encontrar su respectivo valor en la nueva versión del ERP, es decir a que campo de que entidad pertenecería.

Identificador	CU08
Nombre	Transformación Plantilla .doc a .docx
Actor	Usuario de la aplicación
Descripción	En caso de que la plantilla esté en formato .doc es necesario convertirlo a .docx ya que en la nueva versión del ERP se trabaja con OpenXML para la obtención de los campos insertados, del cual es incompatible con .doc.

Identificador	CU09
Nombre	Generación de modelos
Actor	Usuario de la aplicación
Descripción	Generación de los modelos de las plantillas con las DSL Tools. Que serán necesarios tanto para la generación de código como la transformación de plantillas ya que los campos insertados son los DTO generados.

Identificador	CU10
Nombre	Mapeo de campos ERP actual a nueva versión del ERP
Actor	Usuario de la aplicación
Descripción	Es necesario mapear los tablas y sus respectivos campos del ERP actual a como estaría en la nueva versión del ERP ya que ahora se convierten en entidades con sus respectivos campos y además pasan a estar todos en inglés.

Identificador	CU11
Nombre	Transformación de Plantillas
Actor	Usuario de la aplicación
Descripción	La adaptación de las plantillas del ERP actual, asegurando que las plantillas sean compatibles con el nuevo sistema. Sustituyendo los campos insertados que corresponden a campos de tablas de la base de datos del ERP actual a los campos de los DTO generados.

Identificador	CU12
Nombre	Guardar las Plantillas
Actor	Usuario de la aplicación
Descripción	Guardar la plantilla en el microservicio de "ReportDesigner", donde tiene que estar las plantillas en la nueva versión del ERP y actualizar la plantilla del modelo generado.

Identificador	CU13
Nombre	Notificación resultado migración
Actor	Usuario de la aplicación
Descripción	Una vez terminado migración se le notifica al microservicio de migración el resultado tras transformar y guardar las plantillas.

Identificador	CU14
Nombre	Actualización registro de migración
Actor	Usuario de la aplicación
Descripción	Se actualiza el histórico de migración con los resultados obtenidos ya que de esta manera se puede evitar duplicaciones a la siguiente ejecución del proceso de migración.

5.3.2 Requisitos no funcionales

Para asegurar la calidad de un sistema de software, es esencial considerar tanto los requisitos funcionales como los no funcionales. Los requisitos no funcionales, que describen cómo debe funcionar el sistema, son fundamentales para garantizar la eficiencia, seguridad, usabilidad y mantenibilidad del sistema, entre otras cualidades.

La norma ISO/IEC 25010 [17] proporciona un marco para evaluar y mejorar la calidad del software. Define un modelo de calidad que incluye la calidad del sistema, que se refiere a las características intrínsecas del software, y la calidad del software en uso, que se refiere a las características que se hacen evidentes cuando el software se utiliza en un contexto específico.

En el contexto de este proyecto, que implica la migración de plantillas de informes de un sistema ERP a otro, es crucial considerar estos requisitos no funcionales. Esto garantizará que el sistema de migración sea eficiente, preciso, seguro y fácil de mantener, asegurando así la calidad del sistema.

Identificador	RNF01
Nombre	Rendimiento Eficiente
Característica	Eficiencia en el desempeño
Descripción	Los microservicios implementados deben tener una respuesta rápida durante el proceso de migración. Esto se puede probar midiendo el tiempo que tardan en procesar un conjunto de plantillas de prueba.

Identificador	RNF02
Nombre	Compatibilidad en el formato de las plantillas
Característica	Compatibilidad
Descripción	El sistema debe ser capaz de leer y procesar correctamente plantillas de Word en múltiples formatos (.doc y .docx). Puede probarse usando plantillas de prueba en diferentes formatos y comprobando que se leen y procesan correctamente.

Identificador	RNF03
Nombre	Integridad de la plantilla migrada
Característica	Seguridad
Descripción	Las plantillas migradas deben mantener su formato y contenido originales en el proceso de migración. Esto se puede probar comparando las plantillas antes y después de la migración.

Identificador	RNF04
Nombre	Análisis de seguridad de plantillas
Característica	Seguridad
Descripción	Antes de procesar las plantillas, el sistema debe pasarlas a través de un análisis antivirus para asegurar que están libres de malware. Si se detecta algún malware, el sistema debe ser capaz de manejar esta situación de manera segura, por ejemplo, negándose a procesar la plantilla infectada y registrando un mensaje de error apropiado. La eficacia de esta característica se puede probar utilizando plantillas de prueba que contienen malware conocido y observando si el sistema puede detectarlo y manejarlo correctamente.

Identificador	RNF05
Nombre	Mantenibilidad del código
Característica	Mantenibilidad
Descripción	El código del sistema debe ser legible, modular y fácil de modificar. Esto se puede probar realizando cambios en el código y observando cuánto tiempo y esfuerzo se requiere para implementarlos. A su vez que los tutores del equipo revisen el código implementado.

Identificador	RNF06
Nombre	Usabilidad
Característica	Usabilidad
Descripción	Aunque este es un sistema de back-end, todavía puede haber aspectos de usabilidad a considerar, como la claridad de los mensajes de error o la facilidad de configuración. Estos pueden ser probados pidiendo a otros desarrolladores que interactúen con el sistema y recojan sus comentarios. A su vez que los tutores del equipo revisen el código implementado.

5.4 Diseño y estructura

En este apartado, exploraremos en detalle la estructura de los microservicios utilizados en un proyecto específico y el diseño del proceso de migración implementado.

5.4.1 Estructura de la solución

La estructura interna que de los microservicios se asemeja a la de otros microservicios que existen en el ERP en desarrollo, y consta de ocho capas distintas. A continuación, se detalla cada una de ellas:

- **Contratos**: Esta capa aloja las interfaces que definen las operaciones del backend que pueden ser llamadas desde fuera, como por otros microservicios. Importante destacar que aquí se encuentran los DTO, que permiten separar los servicios de la API de su representación interna en el sistema. Los DTO facilitan el manejo y devolución de objetos en un formato distinto al que se guardan internamente.

- **Aplicación**: Esta capa contiene el código generado automáticamente por las herramientas de DSL de aplicación. Dicho código facilita las operaciones CRUD y verifica las autorizaciones del usuario sobre las entidades y campos del microservicio.

- **Lógica**: Esta capa contiene el código generado automáticamente por las herramientas de DSL de aplicación. Dicho código facilita las operaciones CRUD y verifica las autorizaciones del usuario sobre las entidades y campos del microservicio.

- **Dominio**: Aquí se localizan las entidades de dominio modeladas en la herramienta DSL de dominio. Además, incluye código asociado a campos calculados y validaciones, aunque estos no se emplean en este microservicio.

- **Persistencia**: Esta capa es responsable de la interacción con la base de datos para la persistencia de datos. En el microservicio de migración, esta capa se utiliza únicamente para la gestión de conflictos e incidencias y para registrar qué datos han sido migrados o cuáles han producido errores.

- **Servicios**: Actúa como punto de entrada a las acciones o métodos que el microservicio expone a través de controladores. Cada método se define a través de acciones HTTP.

- **Proxy**: Funciona como punto de acceso a las acciones o métodos que el microservicio pone a disposición a través de controladores. Cada método se define a través de acciones HTTP.

- **Referencias externas**: Incluye los métodos necesarios para llamar al backend, generando una llamada HTTP para ello. Esta capa es referenciada por el frontend u otros microservicios a través de un paquete NuGet con el objetivo de comunicarse con este microservicio.

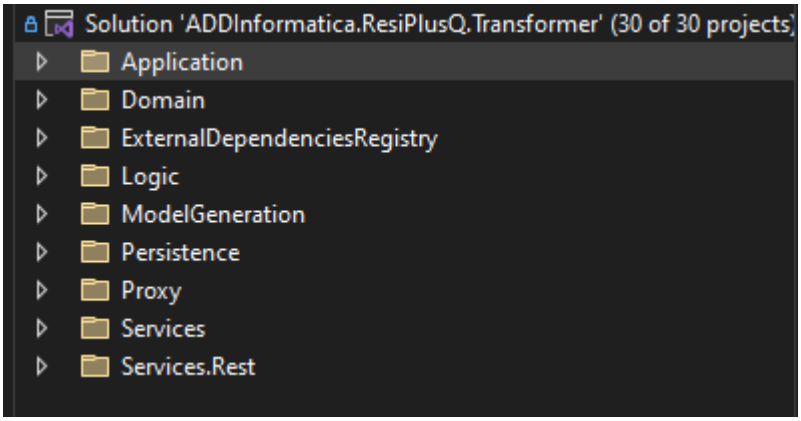

Figura 8. Estructura solución microservicio

En la figura 8 se puede apreciar la estructura de la solución del microservicio "Transformer" con las capas mencionadas. Sin embargo, este microservicio contiene una capa adicional, denominada "ModelGeneration".

La inclusión de esta capa adicional se debe a que la generación de modelos hace uso de las DSL Tools, un proceso diferente al resto del funcionamiento de la solución. Es importante señalar que esta parte del sistema requiere estar implementada con .NET 4.7.2.

En contraste, el desarrollo general del microservicio de Transformer se ha llevado a cabo con .NET 6.0. Aquí radica una limitación técnica, ya que .NET 4.7.2 y .NET 6.0 no son compatibles entre sí.

Por lo tanto, a pesar de que ambas partes del microservicio forman parte de la misma solución, debemos tener en cuenta que operan bajo diferentes versiones de .NET debido a los requerimientos específicos de las DSL Tools utilizadas en la generación de modelos. Esto representa un reto y una consideración importante en la arquitectura y el mantenimiento de nuestra solución.

5.4.2 Estructura de modelos DSL

Una de las características destacadas de este proyecto es la utilización de las herramientas DSL desarrolladas por la empresa. Estas herramientas posibilitan la generación automática de ciertas partes de la solución mediante los modelos que se han creado utilizando dichas herramientas. Este enfoque se alinea con la metodología de desarrollo dirigido por modelos (MDD), que pone el énfasis en la creación de modelos precisos y completos para dirigir el desarrollo del software.

Para generar el código, se utilizan plantillas que definen la estructura y la lógica básica del código. Estas plantillas se llenan con los datos de los modelos creados con las DSL Tools, generando así el esqueleto del código específico a desarrollar en el microservicio.

Aunque el código generado proporciona una base sólida, es importante destacar que aún se requiere la intervención de un programador para implementar las partes específicas de las funcionalidades de la aplicación. El código generado sirve como un punto de partida, pero la lógica de negocio específica y los detalles de implementación deben ser programados manualmente.

El propósito principal del uso de las DSL Tools y del enfoque MDD es facilitar el desarrollo de software. Al proporcionar una estructura y una lógica base, estas herramientas permiten a los programadores centrarse en las partes más complejas y específicas del código, mejorando así la eficiencia y la calidad del desarrollo.

Se han creado modelos utilizando dos tipos de DSL Tools: las DSL Tools de dominio y las DSL Tools de aplicación.

DSL Tools de dominio

Las DSL Tools de dominio se utilizan para modelar el dominio del problema. En otras palabras, permiten representar las entidades, relaciones y reglas de negocio que son fundamentales para el sistema. Los modelos de dominio proporcionan una representación abstracta y de alto nivel del sistema, que puede ser utilizada para guiar el desarrollo y entender mejor el problema que se está resolviendo.

Las entidades, propiedades y relaciones definidas en el modelo pueden ser transformadas automáticamente en una estructura de base de datos SQL. Esto significa que los datos utilizados en el microservicio pueden ser persistidos siguiendo exactamente la misma estructura que se ha modelado.

Para el proceso de migración de plantillas generadoras de informes se le ha añadido en el modelo de dominio del microservicio de migración un nuevo tipo de histórico relacionado con las plantillas que hereda del histórico de migración base. Este histórico es para evitar migraciones duplicadas e innecesarias, y también crear incidencias si la migración del dato en particular encuentra algún problema.

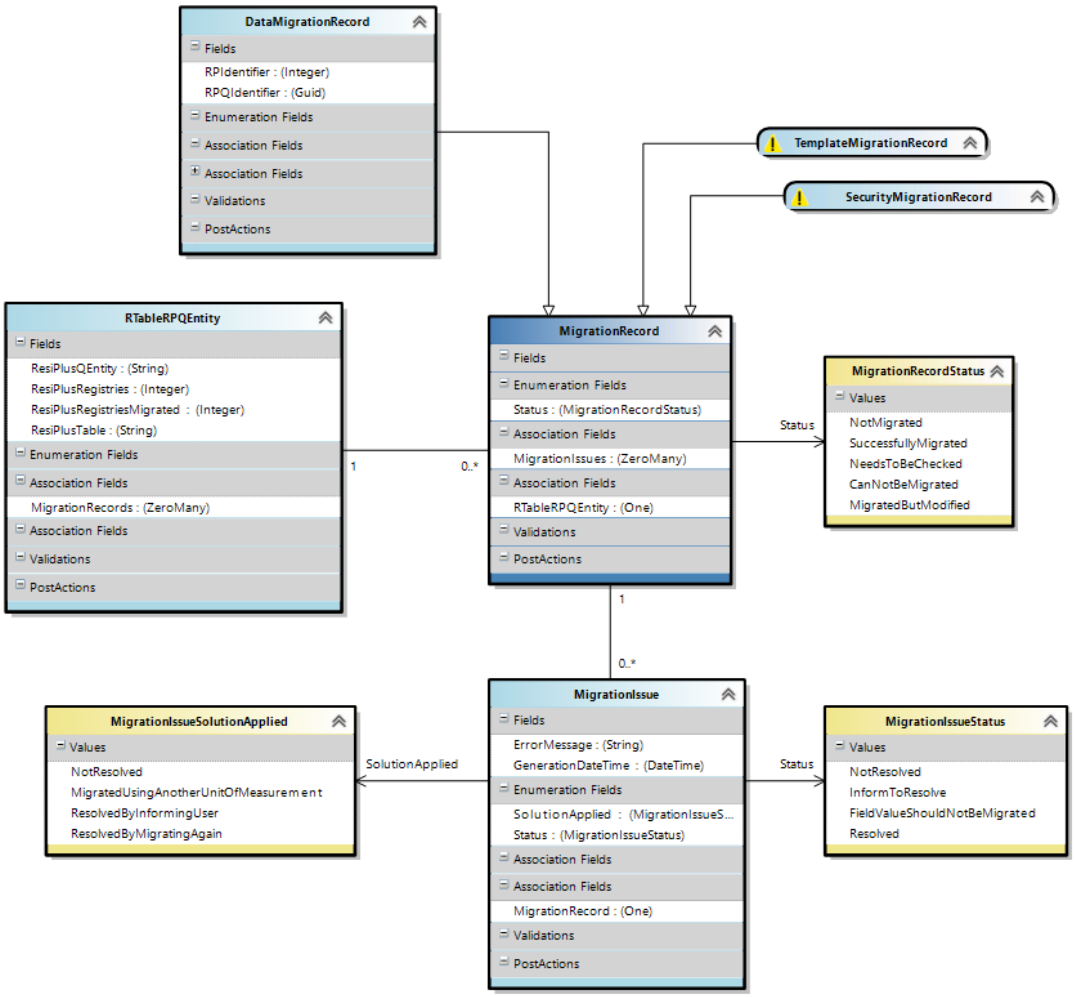

Figura 9. Modelo de dominio migración

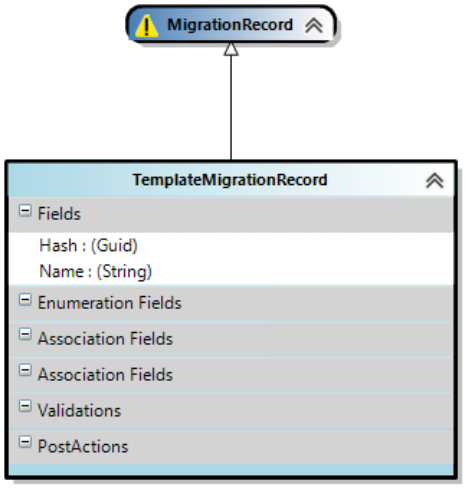

Figura 10. Entidad historico de migración de plantillas

Las figuras 9 y 10 representan las entidades fundamentales que constituyen el registro de migración. Estas entidades, resaltadas en azul, enlazan los datos del producto actual con los nuevos datos generados en el nuevo producto a través de los identificadores de cada registro. Los elementos enumerados, destacados en amarillo, organizan la información relativa al estado de la migración de datos y las incidencias generadas.

En nuestro caso, el diseño del historial de migración de plantillas se fundamenta en tres entidades principales:

- **RTableRPQEntity**: Esta entidad conecta las tablas de la base de datos del producto actual con las entidades modeladas en el microservicio de destino. Alberga el número de registros presentes en la tabla de origen y el número de registros migrados exitosamente, ofreciendo una panorámica de la cantidad de datos migrados y los que restan por migrar.

- **TemplateMigrationRecord**: Esta entidad contiene el nombre y Hash calculado de cada plantilla que necesita ser migrada y es heredera de la entidad base **MigrationRecord**. Señala el estado de migración de cada dato empleando el elemento enumerado **MigrationRecordStatus**. Cada instancia de **MigrationRecord** se encuentra asociada a una instancia de **RTableRPQEntity**, y **RTableRPQEntity** mantiene una relación de cero a muchos con **MigrationRecord**.

- **MigrationIssue**: Esta entidad guarda información acerca de cada incidencia y su estado actual. Conserva un mensaje de error, la fecha de creación de la incidencia, el estado de resolución y el tipo de solución implementada, utilizando los elementos enumerados **MigrationIssueStatus** y **MigrationIssueSolutionApplied**. Una instancia de **MigrationRecord** puede presentar una o varias incidencias y una instancia de **MigrationIssue** debe estar vinculada a una instancia de **MigrationRecord**.

DSL Tools de aplicación

Las DSL Tools de aplicación se utilizan para modelar la lógica de la aplicación y las interacciones entre los diferentes componentes del sistema. Estos modelos pueden incluir detalles sobre la interfaz de usuario, la lógica de negocio, la persistencia de datos, y otros aspectos de la aplicación.

Estos modelos contienen elementos llamados "Acciones Ad hoc", representados por cajas moradas. Estas acciones ad hoc son funciones que pueden ser ejecutadas por el microservicio.

Hay dos categorías de acciones ad hoc: las públicas y las privadas. Las acciones públicas son aquellas que pueden ser invocadas a través de solicitudes HTTP o desde otro microservicio. En cambio, las acciones privadas son las que solo pueden ser empleadas desde el mismo microservicio en el que se encuentran.

Estas acciones ad hoc suelen interactuar con los DTO. Los DTO actúan como parámetros de las acciones, transportando los datos necesarios para que las acciones puedan realizar su trabajo.

Para la creación de la lógica, se ha hecho uso de un tipo específico de DTO: los ParameterDTO. Estos DTO son particularmente flexibles, ya que sus campos y nombres pueden ser creados por el programador. Esto permite que los ParameterDTO se adapten a las necesidades específicas de cada acción.

La Figura 11 ilustra la acción ad hoc pública que coincide con el comienzo de la migración de plantillas, durante la cual se recibe la información pertinente a la localización del servidor y los nombres de las bases de datos de las que se extraerán los datos requeridos para su migración.

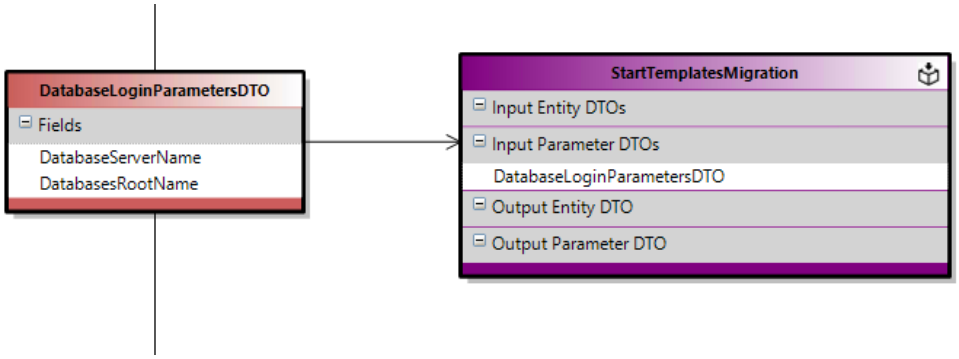

Figura 11. Ad hoc action privada inicio migración de plantillas

Esta acción procede a ejecutar las siguientes acciones ad hoc privadas, que podemos observar en la figura 12, cabe destacar que las plantillas están ubicadas en dos tablas diferentes en la base de datos, las cuales reciben el nombre de “DocumentaciónPlantillas” y “TiposInformes”. Después una vez obtenidos las plantillas, se calcula el Hash de cada una para almacenarlo en el histórico de migración para evitar la duplicación de migración y se le envía las plantillas a transformar al microservicio de transformación llamando a la acción ad hoc pública de esta, corresponde a la figura 13.

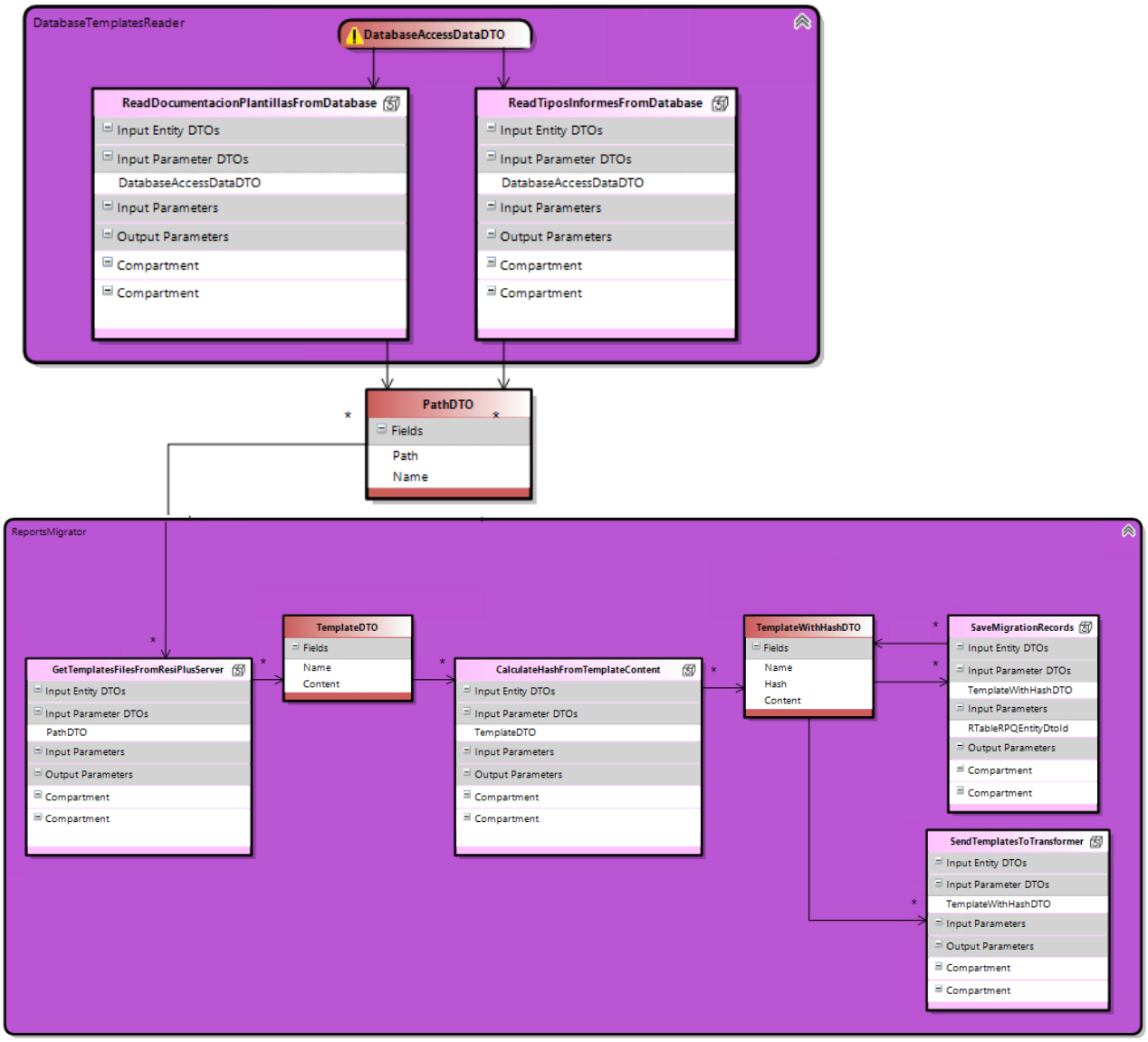

Figura 12. Modelo con las acciones en el microservicio de migración

Figura 13. Ad hoc action privada para la transformación de plantillas

Esta acción procede a ejecutar las acciones ad hoc privadas de la figura 14, y se puede observar que no contiene la acción de generar modelos ya que como se ha comentado anteriormente, existe un problema de compatibilidad de las versiones de .NET entonces se llama a la generación de modelo como un proceso externo.

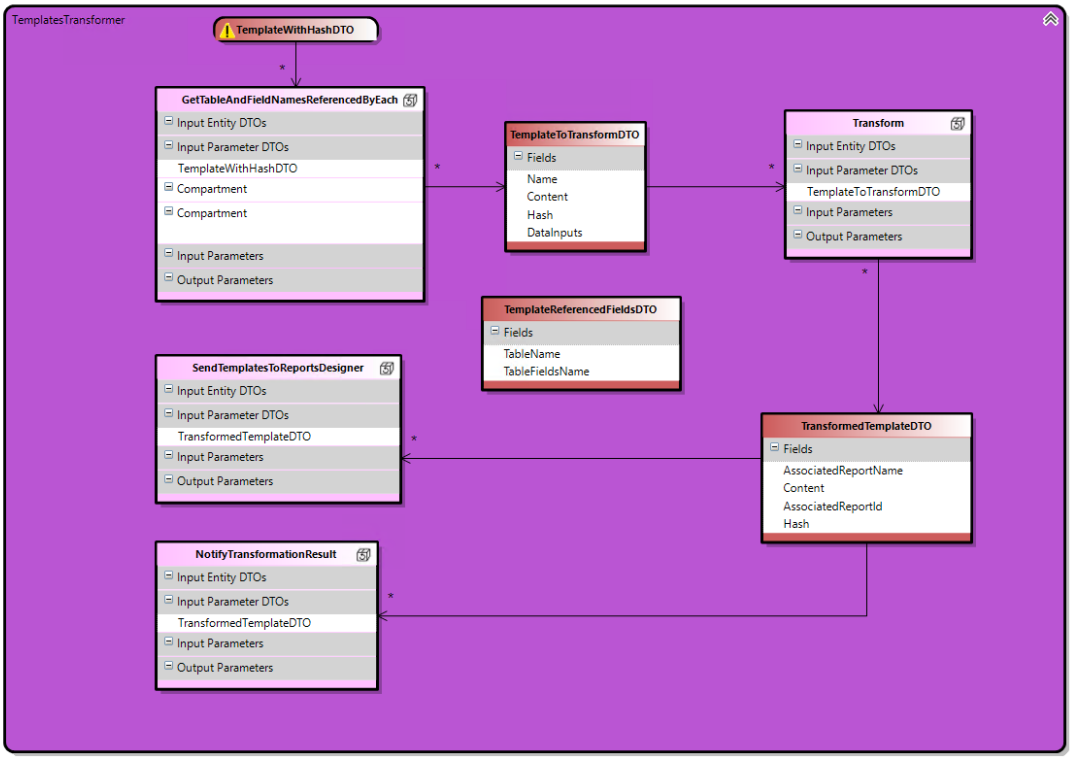

Figura 14. Modelo de las acciones en el microservicio de transformación

La acción de “NotifyTransformationResult” se encarga de notificar al microservicio del migrador los resultados de la transformación actualizando el histórico de migración de las plantillas llamando a la acción ad hoc publica de la figura 15 que se encarga de actualizarlo.

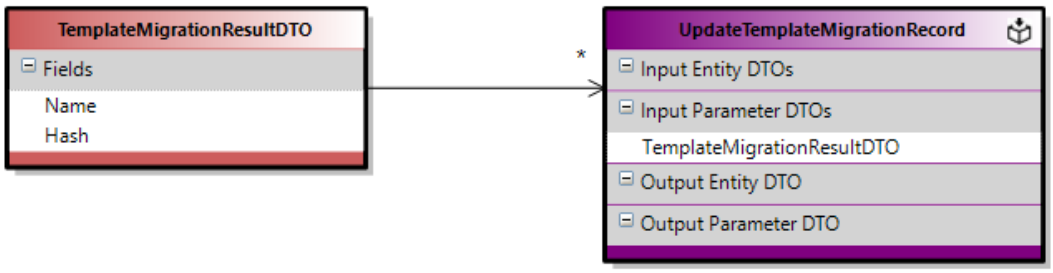

Figura 15.Ad hoc action privada para la actualización de histórico

Por último, para entender mejor el proceso de la migración de plantillas generadoras de informes se ha realizado un diagrama de flujo que corresponde a la figura 16 donde el proceso de migración de plantillas se inicia con la lectura de la base de datos de la ruta de las plantillas, ya que el contenido no se almacena en la versión actual del ERP.

Seguidamente, las plantillas se obtienen a partir de estas rutas y se calcula un hash para cada una, almacenando el nombre de la plantilla y el hash en el historial de migraciones para evitar duplicados y detectar modificaciones. Las plantillas se envían al microservicio de transformación, donde se convierten de .doc a .docx si se da el caso debido a la compatibilidad con OpenXML y se extraen los campos insertados en cada una. Estos campos son mapeados desde la ERP actual a la nueva, generando los modelos correspondientes para transformar las plantillas a la nueva versión del ERP.

El proceso culmina con la notificación del resultado de la migración, lo que permite actualizar el historial de migraciones, garantizando así un control eficiente del proceso de migración.

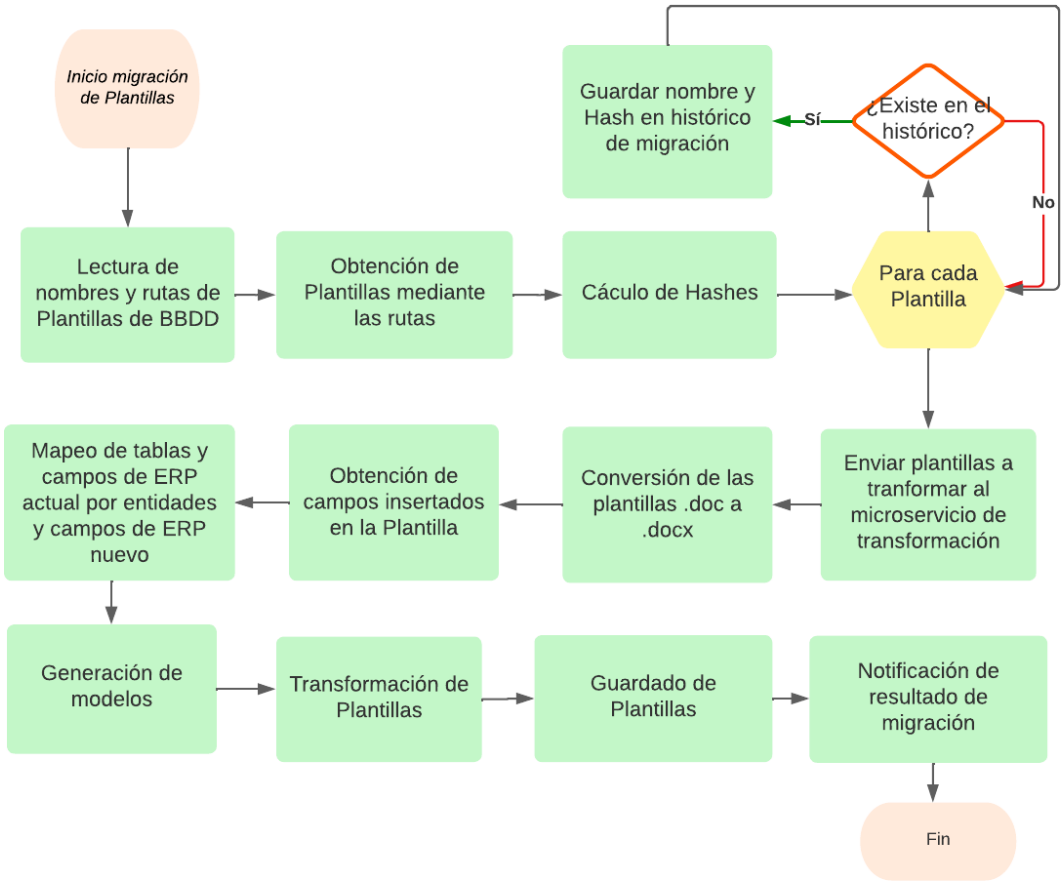

Figura 16. Diagrama de flujos del proceso de migración de plantillas

5.5 Programación

En este apartado, se presentarán y discutirán diversos desafíos y aspectos de interés vinculados a la programación realizada. Esta exploración es fundamental para entender las dificultades inherentes al proceso de desarrollo, las estrategias empleadas para superar obstáculos, y los factores que impulsan la innovación en este campo. La discusión se centrará no solo en los aspectos técnicos, sino también en el impacto y las implicaciones que estas decisiones de programación pueden tener en la eficacia, eficiencia y evolución de los sistemas desarrollados.

5.5.1 Mejora realizada

Una de las mejoras significativas que se han implementado en este proyecto es la creación de un analizador de migración. Este componente se encarga de crear el mapeo de las tablas y sus campos del ERP actual a las entidades y sus campos en la nueva

versión del ERP. Lo que hace especial a este analizador es que conoce todos los modelos de conversión.

Antes de esta mejora, cada vez que se generaba un modelo, se buscaba y cargaba el modelo que contenía la conversión con el mapeo. Este proceso podía ser ineficiente, especialmente cuando se trataba de un gran número de modelos.

Con la introducción del analizador, este problema se ha resuelto. El analizador conoce todos los modelos de conversión desde el principio, lo que significa que ya no es necesario buscar y cargar el modelo de conversión cada vez que se genera un modelo. Esto ha mejorado la eficiencia del proceso de generación de modelos y ha hecho que el proceso de migración sea más rápido y fluido.

Esta mejora es un ejemplo de cómo un cambio aparentemente pequeño puede tener un impacto significativo en la eficiencia y la efectividad de un sistema. Al optimizar el proceso de generación de modelos, se ha mejorado la velocidad y la fiabilidad de todo el proceso de migración.

La figura 17 ilustra que la clase “ReportsGenerator” encargada de la generación de modelo incluye el analizador denominado “MigrationParser”, el cual tiene conocimiento de los modelos y genera un “RPToRPQMapper”. Este mapeador, esencial en el proceso de transición entre sistemas ERP, almacena y vincula el nombre de la tabla en la versión actual del ERP con el nombre de la entidad en la nueva versión del ERP, y correlaciona igualmente los nombres de los campos correspondientes en ambos sistemas.

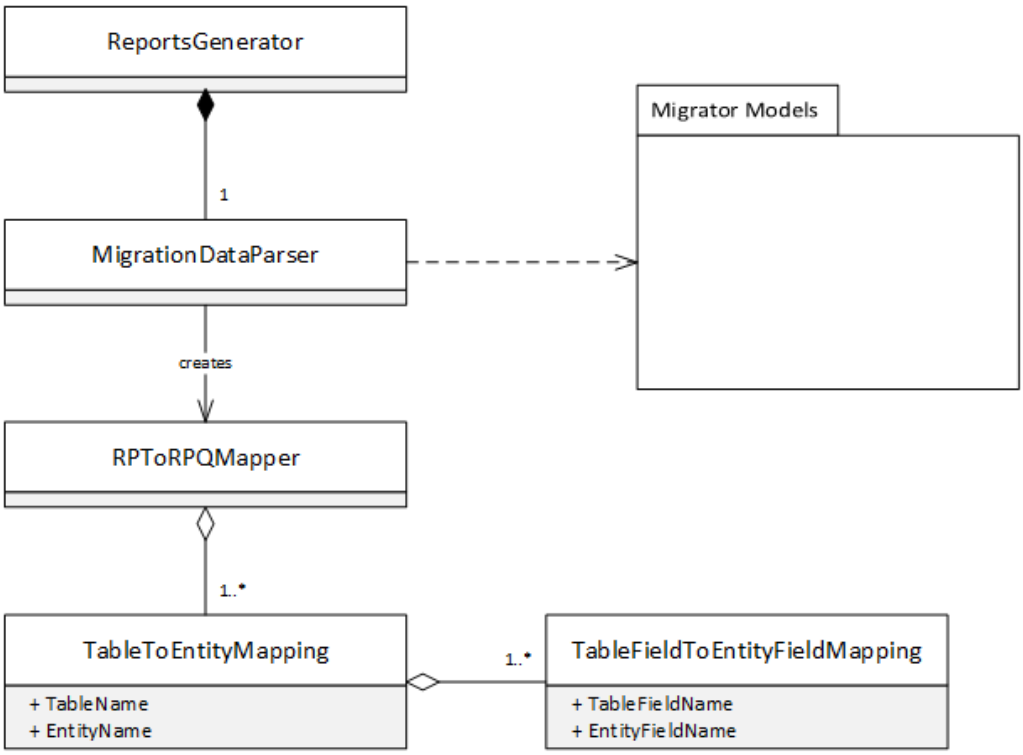

Figura 17. Diagrama de la clase de mapeo

5.5.2 Patrón utilizado

El patrón de estrategia ha sido fundamental para manejar las diferencias entre las plantillas .doc y .docx en tu proyecto. Este patrón de diseño permite seleccionar un algoritmo en tiempo de ejecución, lo que es ideal para situaciones en las que se necesita manejar diferentes tipos de datos o realizar diferentes tipos de operaciones dependiendo de las circunstancias.

En este caso, se han implementado dos estrategias: una para las plantillas .doc y otra para las plantillas .docx. Cada estrategia encapsula el código necesario para leer el tipo de plantilla correspondiente y calcular su hash.

Cuando se necesita procesar una plantilla, el sistema determina el tipo de plantilla y selecciona la estrategia correspondiente. Esta estrategia se encarga entonces de leer la plantilla y calcular su hash.

El uso del patrón de estrategia ha proporcionado una gran flexibilidad al sistema, permitiendo manejar fácilmente las diferencias entre las plantillas .doc y .docx y adaptarse a diferentes requisitos de cálculo de hash. Esta flexibilidad ha sido crucial para el éxito del proceso de migración.

Aunque solo hay dos formatos de archivo (.doc y .docx) en este caso, el patrón de estrategia aún puede ser beneficioso ya que las operaciones para manejar estos dos formatos son significativamente diferentes. Además de que garantiza flexibilidad para cambios futuros y código más limpio y mantenible.

5.5.3 Desafíos

A lo largo del desarrollo, se han encontrado varios desafíos que han requerido soluciones innovadoras y adaptativas.

Uno de los primeros desafíos fue la lectura de las plantillas de los clientes, que pueden estar en formatos .doc o .docx. En la nueva versión del ERP, se trabaja con el SDK de OpenXML, que se intentó utilizar para leer ambos formatos de la misma manera. Sin embargo, esto resultó ser un desafío ya que .doc y .docx son formatos diferentes y no pudieron ser leídos de la misma manera con el SDK de OpenXML.

El formato .doc es el formato de archivo de Word más antiguo, que se utilizó por defecto en las versiones de Word desde Word 97 hasta Word 2003. Este formato es un formato binario, lo que significa que la información en el archivo se almacena en una serie de unos y ceros. Esto puede hacer que los archivos .doc sean más difíciles de manejar para los programas que no son Microsoft Word.

Por otro lado, el formato .docx es el formato de archivo predeterminado para las versiones de Word desde Word 2007 en adelante. Este formato es un formato basado en XML, lo que significa que la información en el archivo se almacena en un formato de texto legible por humanos. Esto hace que los archivos .docx sean más fáciles de

manejar para una variedad de programas, incluyendo el SDK de OpenXML que se utiliza en este proyecto.

Otro desafío significativo fue el cálculo del hash de un documento. Se descubrió que el hash de un documento, que se calcula a partir de todo su contenido, nunca es el mismo debido a los metadatos del documento. Los metadatos son datos sobre los datos, que en el caso de un documento pueden incluir información como la fecha de creación, el autor, y otros detalles que pueden cambiar cada vez que se abre o se lee el documento. Esto presentó un desafío para el proceso de migración, ya que se necesitaba una forma de calcular un hash consistente que no se viera afectado por los cambios en los metadatos.

Finalmente, se encontró un desafío en la generación de modelos. La generación de modelos se realiza en .NET 4.7.2, mientras que el microservicio trabaja con .NET 6.0 Estas dos versiones de .NET no son compatibles entre sí, por lo que la generación de modelos tuvo que ser implementada como un proceso separado.

A pesar de estos desafíos, se han encontrado soluciones efectivas y el proyecto ha avanzado con éxito. Estos desafíos han proporcionado valiosas oportunidades de aprendizaje y han contribuido a la mejora continua del proceso de desarrollo.

5.6 Pruebas

Las pruebas son una parte esencial del desarrollo de software, ya que garantizan que el sistema funciona correctamente y cumple con los requisitos establecidos. En el contexto de este proyecto, se han realizado tanto pruebas automatizadas como manuales. Las pruebas automatizadas son útiles para verificar rápidamente la funcionalidad y la integridad del código, especialmente para tareas repetitivas y procesos que pueden ser fácilmente automatizados. Sin embargo, no todos los aspectos del sistema pueden ser probados automáticamente. Por ejemplo, la generación de modelos, que es una parte crucial del proceso de migración de plantillas, requiere una revisión manual para asegurar que los modelos se generan correctamente y son adecuados para su uso en el nuevo sistema ERP. Por lo tanto, las pruebas manuales también han sido una parte importante del proceso de prueba, complementando las pruebas automatizadas y asegurando una cobertura de prueba completa.

5.6.1 Pruebas automatizadas

En el contexto de este proyecto, se han realizado pruebas unitarias automatizadas para cada funcionalidad. Las pruebas unitarias se centran en probar unidades individuales de código, como funciones o métodos, de manera aislada. Estas pruebas ayudan a asegurar que cada parte del sistema funciona correctamente por sí misma, lo que es fundamental para la fiabilidad general del sistema.

Para facilitar la creación de estas pruebas unitarias, se han utilizado las DSL Tools, herramientas de modelado mencionadas anteriormente. En este caso, las DSL Tools han permitido crear pruebas unitarias de manera más eficiente y precisa y sin necesidad de implementar código.

Un ejemplo de modelado sería los tests de la figura 18 que pertenecen a la acción de obtener plantillas desde la ruta guardada en la base de datos que se prueba la correcta obtención de los archivos de plantillas ya que pueden estar guardadas o en local o en el servidor del ERP actual:

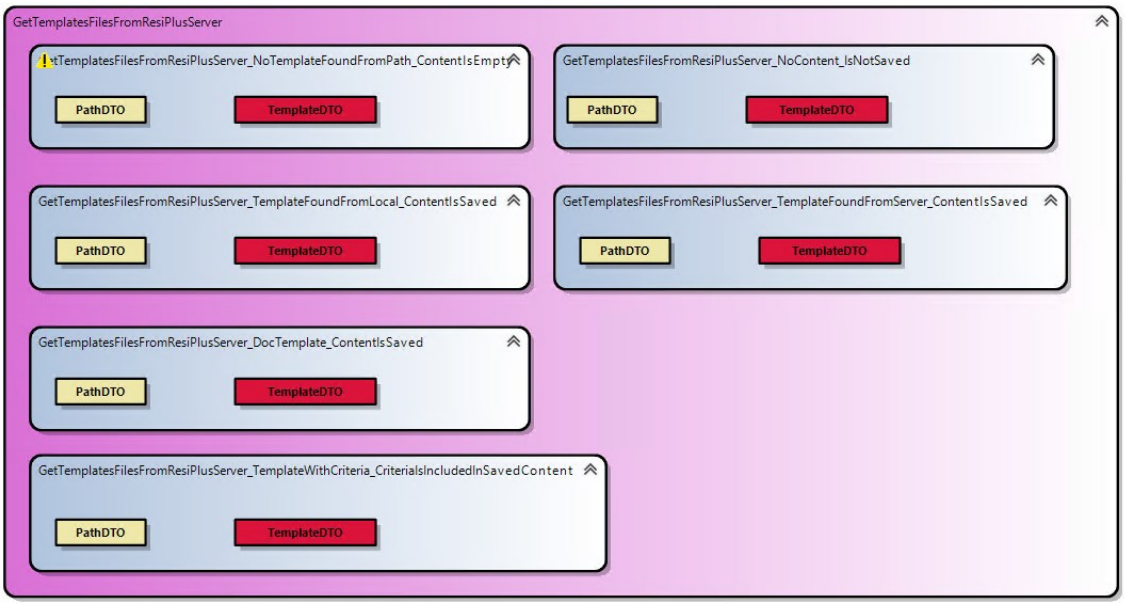

Figura 18. Modelo de tests de obtención de plantillas

Para las pruebas creadas con DSL Tools, se establece una convención de colores para ayudar a identificar diferentes elementos. Según esta convención:

- **Contenedores Morados**: Representan la clase de test. Estos contenedores agrupan un conjunto de pruebas relacionadas que se ejecutarán juntas. Pueden contener múltiples contenedores azules de prueba y también pueden tener cajas amarillas y rojas de DTO de entrada y salida respectivamente.

- **Contenedores Azules**: Representan las pruebas individuales. Estos contenedores contienen lógica de prueba específica y se ejecutan de forma aislada. Cada contenedor azul de prueba puede tener una o varias cajas amarillas de DTO de entrada y cajas rojas de DTO de salida.

- **Cajas Amarillas**: Representan los DTO de entrada. Estas cajas contienen los datos que se utilizarán como entrada en las pruebas. Los DTO de entrada se pueden vincular a los contenedores azules de prueba correspondientes para simular diferentes escenarios de entrada.

- **Cajas Rojas**: Representan los DTO de salida. Estas cajas contienen los datos esperados o resultantes que se verificarán durante la ejecución de las pruebas. Los DTO de salida se utilizan para validar los resultados obtenidos en las pruebas y se vinculan a los contenedores azules de prueba asociados.

En resumen, los contenedores morados representan la clase de test, los contenedores azules son las pruebas individuales, las cajas amarillas son los DTO de entrada y las cajas rojas son los DTO de salida. Esta convención de colores facilita la comprensión y organización de las pruebas creadas con DSL Tools.

Cabe destacar también los tests que realizan interacción con las bases de datos del microservicio, como podemos ver en la figura 19:

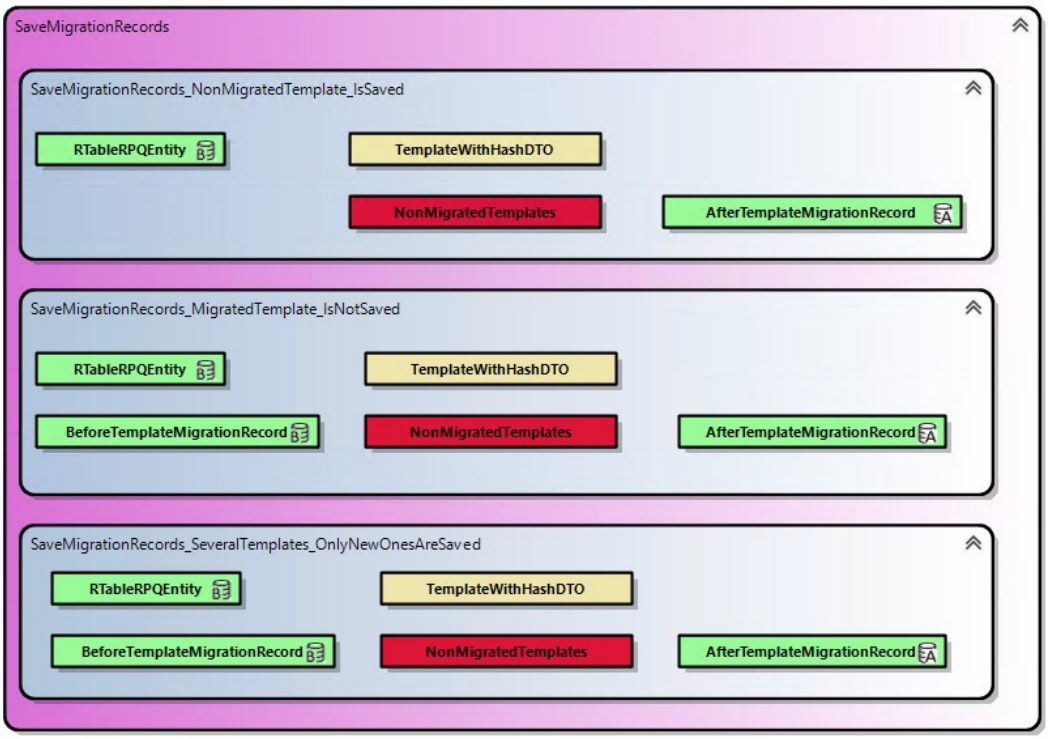

Figura 19. Modelo de tests de guardado de histórico de migración

Para completar la información, se pueden utilizar cajas verdes para representar los elementos presentes en la base de datos del microservicio antes de la migración y para representar los datos esperados después de haber ejecutado el proceso de migración.

- **Cajas Verdes de la Izquierda**: Simbolizan los elementos que ya existen en la base de datos del microservicio previo a la migración. Estas cajas contienen los datos y estructuras que se encuentran en la base de datos actual del sistema. Pueden incluir tablas, columnas, registros y otros componentes de la base de datos.

- **Cajas Verdes de la Derecha**: Simbolizan la información que se prevé que la base de datos del microservicio de migración contenga una vez que el proceso de migración de los datos leídos se haya completado. Estas cajas contienen los datos y estructuras que se esperan como resultado de la migración. Pueden incluir cambios en la estructura de la base de datos, nuevos registros, actualizaciones y cualquier otro cambio esperado en los datos almacenados.

Al utilizar cajas verdes para representar los elementos de la base de datos, se puede visualizar y comparar fácilmente los datos antes y después de la migración, lo que facilita la comprensión del proceso y la validación de los resultados obtenidos.

5.6.2 Pruebas manuales

Las pruebas manuales son una parte esencial del proceso de prueba, especialmente para las partes del sistema que pueden ser difíciles de evaluar adecuadamente a través de pruebas automatizadas. En este proyecto, las pruebas manuales han sido particularmente importantes para la generación de modelos.

La generación de modelos es un proceso crucial en la migración de plantillas, ya que los modelos sirven como plantillas para la creación de nuevos informes en el nuevo sistema ERP. Sin embargo, este proceso no puede ser automatizado con las DSL Tools debido a las diferencias entre las versiones de .NET utilizadas por el microservicio y la generación de modelos. Mientras que el microservicio utiliza .NET 6.0, la generación de modelos utiliza .NET 4.7.2.

Por lo tanto, se han realizado pruebas manuales para verificar que los modelos se generan correctamente. Estas pruebas implican revisar cada modelo generado para asegurar que refleja correctamente la estructura y el contenido de la plantilla original, y que es compatible con la nueva versión del ERP. Además, se verifica que cada modelo pueda ser utilizado para generar un informe correctamente en el nuevo sistema.

Aunque las pruebas manuales pueden ser más laboriosas que las pruebas automatizadas, son esenciales para asegurar la calidad y la fiabilidad del sistema, especialmente en áreas como la generación de modelos que no pueden ser probadas automáticamente.

Un ejemplo de modelo generado de plantillas tras ejecutar una prueba manual es la figura 20 donde tenemos una plantilla con su nombre relacionado con un DTO con los campos *FormalFullName* y *Surname1*:

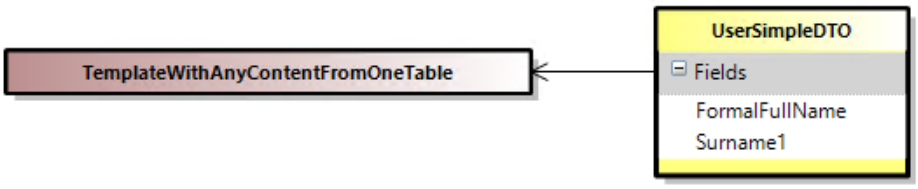

Figura 20. Ejemplo modelo generado

5.6.3 Resultado de pruebas

La implementación de pruebas, tanto automatizadas como manuales, ha sido fundamental para el éxito de este proyecto. Las pruebas han proporcionado una serie de beneficios significativos, incluyendo la identificación temprana de problemas, la garantía de la calidad del código y la validación de la funcionalidad del sistema.

Se han establecido aproximadamente 30 pruebas, de las cuales todas las pruebas relacionadas con acciones ad hoc sí han sido automatizadas. Cada una de estas pruebas se ha creado después de la implementación de cada método correspondiente y se ha ejecutado posteriormente. Se han aplicado en forma regresiva, es decir, cada vez que se modifica el código asociado, estas pruebas se ejecutan de nuevo para garantizar la estabilidad y funcionalidad del sistema. Asimismo, se han llevado a cabo estas pruebas en cada etapa de integración de los sprints, reafirmando su funcionamiento correcto y asegurando la calidad y efectividad del código a medida que se va desarrollando el proyecto.

Uno de los beneficios más importantes de las pruebas ha sido la identificación temprana de problemas. Por ejemplo, gracias a las pruebas, se descubrió que el SDK de OpenXML no puede leer documentos .doc. Este hallazgo fue crucial, ya que permitió buscar e implementar una solución alternativa a tiempo, evitando posibles retrasos y problemas en la migración de las plantillas.

Otro hallazgo importante que se hizo gracias a las pruebas fue que el hash calculado para un documento es cambiante, ya que cada vez que se lee o se abre un documento, cambian los metadatos. Este descubrimiento fue esencial para el diseño del proceso de migración, ya que implicó que se necesitaba una forma de calcular el hash que no se viera afectada por los cambios en los metadatos.

En resumen, la implementación de pruebas ha sido esencial para el desarrollo de este proyecto. Las pruebas han permitido identificar y resolver problemas temprano, asegurar la calidad del código y validar la funcionalidad del sistema, contribuyendo en gran medida al éxito del proyecto.

5.7 Cronología del proyecto

El desarrollo de este proyecto se ha llevado a cabo en varias etapas, siguiendo una metodología ágil con sprints de 1 a 2 semanas.

En el inicio del proyecto, durante el **Sprint 0**, se planteó el problema, se diseñaron los requisitos y se especificaron en detalle. Además, se crearon los modelos de dominio y se definieron las acciones a realizar en el microservicio "Migrador".

A continuación, en el **Sprint 1**, se desarrolló la funcionalidad para la lectura de la base de datos, la obtención de las plantillas y el cálculo del hash de las mismas.

El siguiente paso, durante **el Sprint 2**, se centró en la corrección del cálculo del hash y en la creación de pruebas para los métodos implementados, asegurando así la calidad y la fiabilidad del código.

En el **Sprint 3**, se implementó el guardado del histórico de migración y se preparó el envío de las plantillas al microservicio "Transformer". Se crearon pruebas para estas nuevas funcionalidades, garantizando su correcto funcionamiento.

El **Sprint 4** se dedicó a la transformación de las plantillas. Dada la complejidad de algunas plantillas, se comenzó con la casuística de plantillas formados por campos concretos. Se implementó la lectura de los campos insertados, la generación del modelo y el mapeo de las tablas por entidades. También se crearon pruebas para estas nuevas funcionalidades.

Durante el **Sprint 5**, se realizó la transformación de las plantillas y su posterior guardado en la base de datos. Se crearon pruebas para estas nuevas funcionalidades, asegurando su correcta implementación.

Finalmente, en el **Sprint 6**, se está trabajando en la notificación del resultado de la migración, la conversión de documentos de formato .doc a .docx y una nueva casuística de plantilla con tablas.

Cada sprint ha sido una etapa crucial en el desarrollo del proyecto, permitiendo un avance constante y estructurado hacia la finalización del TFG.

Además de las tareas específicas de cada sprint, a lo largo de todo el proyecto se ha llevado a cabo la redacción de la memoria del TFG. Esta tarea ha implicado documentar de manera continua todas las fases del proyecto, desde la planificación inicial hasta la implementación final, incluyendo los desafíos encontrados y cómo se superaron. Esta documentación ha sido esencial para mantener un registro detallado del trabajo realizado y para facilitar la revisión y evaluación del proyecto.

En la figura 21 se puede observar una representación visual de los sprints mencionados anteriormente:

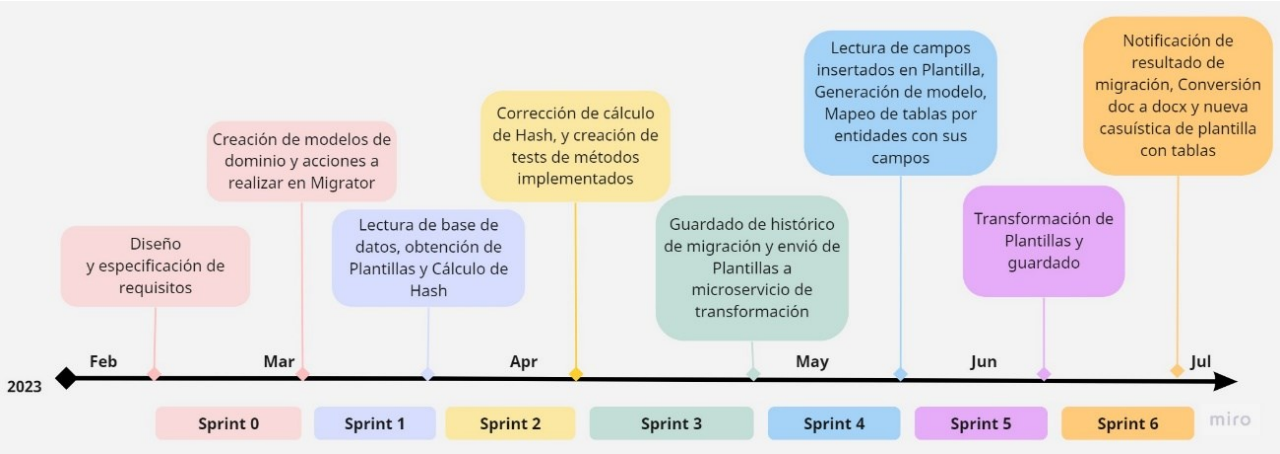

Figura 21. Línea de tiempo

Por último, cabe mencionar que este proyecto ha requerido un esfuerzo significativo, resultando en aproximadamente 3000 líneas de código escritas. Sin embargo, gracias al uso de las DSL Tools, se ha optimizado el proceso de desarrollo, evitando una cantidad aún mayor de código. La implementación del proceso de migración en sí mismo ha consumido alrededor de 330 horas, abarcando desde la planificación inicial hasta la implementación final, incluyendo las pruebas y la corrección de errores.

CAPÍTULO 6

Conclusiones y trabajo futuro

El desarrollo de este Trabajo de Fin de Grado ha sido una experiencia enriquecedora tanto a nivel personal como profesional, ha supuesto un reto y una oportunidad de crecimiento. Ha permitido aplicar los conocimientos adquiridos durante la carrera, especialmente en las asignaturas de Proceso de software, Calidad de software, Diseño de software y Análisis y Especificación de requisitos. Estas asignaturas han proporcionado las bases teóricas y prácticas necesarias para enfrentar los desafíos del proyecto.

Los objetivos planteados al inicio del proyecto se han cumplido de manera satisfactoria, lo que demuestra la efectividad de las estrategias y técnicas implementadas. Se ha desarrollado un mecanismo de control robusto que evita la migración de plantillas que ya han sido migradas previamente. Este mecanismo, basado en el cálculo de hash de las plantillas, ha demostrado ser eficaz y ha optimizado el proceso de migración al evitar redundancias innecesarias.

Además, se han realizado pruebas exhaustivas para garantizar la funcionalidad y eficiencia del sistema. Estas pruebas han permitido identificar y corregir posibles fallos, asegurando así la calidad del software desarrollado.

En cuanto a la estrategia de migración incremental, se ha logrado un progreso significativo. Se ha comenzado con las plantillas con estructuras de campos más sencillas, avanzando progresivamente hacia estructuras de campos más complejas. Actualmente, el sistema ya es capaz de migrar plantillas con estructuras de campos simples, lo que demuestra el éxito de la estrategia implementada.

A pesar de los logros alcanzados, aún queda trabajo por hacer. En el futuro, planeamos abordar casuísticas más complejas, como las tablas anidadas. Este será un desafío importante, ya que requiere una comprensión más profunda de las estructuras de datos y una mayor capacidad de adaptación del sistema.

Además, hay varios cambios relacionados con la base de datos de las dos ERP que requerirán un estudio más profundo. Algunos campos de la base de datos del ERP actual ya no existen en la nueva versión del ERP, o se han convertido en tablas que serán referenciadas. Este es un aspecto crucial que debe ser abordado para garantizar la correcta migración de los datos.

Finalmente, en términos de seguridad, se planea realizar un análisis de seguridad de las plantillas antes de ser migradas para prevenir cualquier posible malware (RNF04). Este será un aspecto crucial para garantizar la integridad y seguridad de todo el sistema de migración. La seguridad es un aspecto fundamental en cualquier sistema de software, y este proyecto no es una excepción. En el futuro, se realizarán análisis de

seguridad más exhaustivos para garantizar que las plantillas migradas no contengan malware y que el sistema de migración sea seguro y confiable.

Referencias

[1] «Talend Data Migration,» [En línea]. Available: https://www.talend.com/resources/understanding-data-migration-strategies-best-practices/. [Último acceso: 24 Mayo 2023].

[2] «Talend Data Transformation,» [En línea]. Available: https://www.talend.com/resources/data-transformation-defined/. [Último acceso: 24 Mayo 2024].

[3] «ODI Understanding Migration Process,» [En línea]. Available: https://docs.oracle.com/middleware/1221/odi/install-migrate-owb-odi/understanding.htm#ODIMG107. [Último acceso: 24 Mayo 2023].

[4] «Características SSIS,» [En línea]. Available: https://learn.microsoft.com/es-es/sql/integration-services/integration-services-features-supported-by-the-editions-of-sql-server?view=sql-server-ver16. [Último acceso: 24 Mayo 2023].

[5] «Documentación SSIS,» [En línea]. Available: https://learn.microsoft.com/es-es/sql/integration-services/integration-services-developer-documentation?view=sql-server-ver16. [Último acceso: 24 Mayo 2023].

[6] «Modeling SDK for Visual Studio - Domain-Specific Languages,» [En línea]. Available: https://learn.microsoft.com/en-us/visualstudio/modeling/modeling-sdk-for-visual-studio-domain-specific-languages?view=vs-2022. [Último acceso: 22 Mayo 2023].

[7] «Beneficios de MDD,» [En línea]. Available: https://www.onetool.ph/2019/04/23/6-key-benefits-of-model-driven-development/#:~:text=6%20Key%20Benefits%20of%20Model-Driven%20Development%201%201.,6%206.%20You%20keep%20your%20Corporate%20Knowledge%20. [Último acceso: 26 Mayo 2023].

[8] «Unit Testing with Moq,» [En línea]. Available: https://methodpoet.com/unit-testing-with-moq/. [Último acceso: 21 Mayo 2023].

[9] «Newtonsoft,» [En línea]. Available: https://www.newtonsoft.com/json/help/html/Introduction.htm. [Último acceso: 29 Mayo 2023].

[10] «Estructura WordProcessingML,» [En línea]. Available: https://learn.microsoft.com/es-es/office/open-xml/structure-of-a-wordprocessingml-document. [Último acceso: 22 Mayo 2023].

[11] «Open XML SDK,» [En línea]. Available: https://learn.microsoft.com/es-es/office/open-xml/open-xml-sdk. [Último acceso: 22 Mayo 2023].

[12] «Open XML PowerTools,» [En línea]. Available: http://www.ericwhite.com/blog/powertools-for-open-xml-expanded/. [Último acceso: 22 Mayo 2023].

[13] «What are microservices?,» [En línea]. Available: https://microservices.io/. [Último acceso: 20 Mayo 2023].

[14] «Microservices article,» [En línea]. Available: https://martinfowler.com/articles/microservices.html. [Último acceso: 20 mayo 2023].

[15] «Diseño arquitectura de microservicios,» [En línea]. Available: https://learn.microsoft.com/es-es/azure/architecture/microservices/. [Último acceso: 6 Junio 2023].

[16] «Beneficios arquitectura de microservicios,» [En línea]. Available: https://www.hiberus.com/crecemos-contigo/cuales-son-los-beneficios-de-una-arquitectura-microservicios/. [Último acceso: 6 Junio 2023].

[17] «ISO/IEC 25010,» [En línea]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010. [Último acceso: 29 Mayo 2023].

APÉNDICE A

Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

El Trabajo de Fin de Grado (TFG) que estamos desarrollando tiene una relación directa e importante con dos de los Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas: el ODS 3 "Salud y bienestar" y el ODS 9 "Industria, innovación e infraestructuras".

ODS 3: Salud y bienestar

El presente TFG se enmarca en el contexto de una empresa del sector sociosanitario. Este sector juega un papel crucial en la promoción de la salud y el bienestar de la población, que es precisamente el objetivo del ODS 3. Al desarrollar una nueva plataforma ERP basada en microservicios para la migración y transformación de datos, estamos contribuyendo a mejorar la eficiencia y eficacia de los servicios sociosanitarios. Esto, a su vez, puede tener un impacto positivo en la calidad de la atención y los servicios que reciben los usuarios, lo que contribuye a su salud y bienestar general.

ODS 9: Industria, innovación e infraestructuras

El ODS 9 se centra en la construcción de infraestructuras resilientes, la promoción de la industrialización inclusiva y sostenible, y el fomento de la innovación. Este TFG se alinea con este objetivo en varios aspectos. Estamos trabajando en la creación de una infraestructura de software resiliente y escalable mediante el uso de una arquitectura basada en microservicios. Además, estamos promoviendo la innovación al desarrollar soluciones personalizadas para la migración y transformación de datos, que van más allá de las capacidades de las herramientas existentes.

En resumen, este TFG tiene una relación directa y significativa con los ODS 3 y 9. A través de nuestro trabajo, estamos contribuyendo a la promoción de la salud y el bienestar, y al mismo tiempo fomentando la innovación y el desarrollo de infraestructuras resilientes en el sector sociosanitario.