UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

Redisseny i Refactorització d'una solució dintegració en el domini del transport. Una aplicació pràctica d'arquitectures dirigides per APIs

Treball Fi de Grau

Grau en Enginyeria Informàtica

AUTOR/A: Salido Ferrandis, Pau

Tutor/a: Fons Cors, Joan Josep

CURS ACADÈMIC: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Redisseny i refactorització d'una solució d'integració al domini del transport. Una aplicació pràctica del disseny d'arquitectures dirigit per APIs

TREBALL FI DE GRAU

Grau en Enginyeria Informàtica

Autor: Pau Salido Ferrandis

Tutor: Joan Josep Fons i Cors

Curs 2022-2023

Resum

Aquest treball presenta una proposta de solució basada en l'arquitectura d'integració d'API-led, centrada a millorar la comunicació i la gestió d'un projecte existent en el context de la gestió d'enviaments amb múltiples transportistes. En primer lloc, es presenta el context de la integració d'aplicacions empresarials en l'actualitat i es fa una anàlisi exhaustiu del projecte sobre el qual parteix aquest document, del que podem extraure que el principal problema és la falta d'escalabilitat i flexibilitat.

Després d'identificar les debilitats de l'arquitectura del projecte inicial, es formula una proposta de disseny emprant una arquitectura API-Led que soluciona els aspectes a millorar de la fase d'anàlisi. S'exposarà com ha sigut el procés d'implementació d'aquesta nova proposta i s'indicarà el procés necessari per a substituir l'anterior disseny amb el nou. Concretament, es farà servir Mulesoft i les ferramentes d'Anypoint Platform per al desenvolupament de la solució.

Per últim, es mostrarà com dur a terme proves end-to-end i proves de càrrega sobre el nou projecte fent ús de les ferramentes Postman i JMeter. Podem concloure que l'escalabilitat de l'arquitectura d'un projecte d'integració és fonamental per tal d'estalviar recursos de manteniment i per assegurar-ne el correcte funcionament.

Paraules clau: EAI, Integració d'aplicacions, disseny arquitectònic, arquitectures escalables, API-Led, middleware, Mulesoft

Resumen

Este trabajo presenta una propuesta de solución basada en la arquitectura de integración de API-Led, centrada en mejorar la comunicación y la gestión de un proyecto existente en el contexto de la gestión de envíos con múltiples transportistas. En primer lugar, se presenta el contexto de la integración de aplicaciones empresariales en la actualidad y se realiza un análisis exhaustivo del proyecto sobre el que parte este documento, de lo que podemos extraer que el principal problema es la falta de escalabilidad y flexibilidad.

Tras identificar las debilidades de la arquitectura del proyecto inicial, se formula una propuesta de diseño utilizando una arquitectura API-Led que soluciona los aspectos a mejorar de la fase de análisis. Se expondrá cómo ha sido el proceso de implementación de esta nueva propuesta y se indicará el proceso necesario para sustituir el anterior diseño con el nuevo. Concretamente, se utilizará Mulesoft y las herramientas de Anypoint Platform para el desarrollo de la solución.

Por último, se mostrará cómo llevar a cabo pruebas end-to-end y pruebas de carga sobre el nuevo proyecto haciendo uso de las herramientas Postman y JMeter. Podemos concluir que la escalabilidad de la arquitectura de un proyecto de integración es fundamental para ahorrar recursos de mantenimiento y asegurar su correcto funcionamiento.

Palabras clave: EAI, Integración de aplicaciones, diseño arquitectónico, arquitecturas escalables, API-Led, middleware, Mulesoft

Abstract

This work presents a proposed solution based on the API-Led integration architecture, focused on improving the communication and management of an existing project in the context of multi-carrier shipment management. First, the context of the integration of business applications today is presented and a comprehensive analysis is made of the project on which this document is based, from which we can extract that the main problem is the lack of scalability and flexibility .

After identifying the weaknesses of the initial project's architecture, a design proposal is formulated using an API-Led architecture that solves the aspects to be improved in the analysis phase. The implementation process of this new proposal will be explained and the process necessary to replace the previous design with the new one will be indicated. Specifically, Mulesoft and Anypoint Platform tools will be used for the development of the solution.

Finally, it will show how to perform end-to-end testing and load testing on the new project using the Postman and JMeter tools. We can conclude that the scalability of the architecture of an integration project is fundamental in order to save maintenance resources and to ensure its correct performance.

Key words: EAI, Application integration, architectural design, scalable architectures, API-Led, middleware, Mulesoft

Índex

Índex **v**

Índex de figures **vii**

Índex de taules **viii**

1 Introducció **1**
1.1 Motivació . . . 1
1.2 Objectius . . . 2
1.3 Estructura de la memòria . . . 2

2 Estat de l'art **5**
2.1 Tipus d'integracions . . . 6
2.1.1 Segons el propòsit . . . 6
2.1.2 Segons el model de comunicació . . . 6
2.2 Tipus de comunicacions . . . 9
2.3 Connectivitat dirigida per API . . . 10

3 Anàlisi del problema **13**
3.1 Debilitats d'un model hub-and-spoke . . . 13
3.2 Anàlisi del projecte inicial . . . 14
3.3 Possibles solucions . . . 16
3.3.1 Arquitectura Orientada a Serveis (SOA) . . . 16
3.3.2 Microserveis . . . 16
3.3.3 Enterprise Service Bus (ESB) basat en una arquitectura API-Led . . . 16
3.4 Solució proposada . . . 17
3.5 Metodologies de treball . . . 18

4 Disseny de la solució **21**
4.1 Arquitectura del sistema . . . 21
4.2 Disseny detallat . . . 22
4.3 Disseny d'APIS . . . 26
4.4 Model canònic de dades . . . 27
4.5 Transformació de dades . . . 28

5 Desenvolupament de la solució **33**
5.1 Tecnologia emprada . . . 33
5.1.1 Mulesoft . . . 33
5.1.2 Anypoint Platform . . . 34
5.1.3 Anypoint Studio . . . 35
5.1.4 DataWeave . . . 36
5.1.5 Cloudhub . . . 36

5.1.6 RAML . . . 37
5.2 Bones pràctiques . . . 37
5.3 Implementació comuna a totes les API . . . 38
5.3.1 Especificacions RAML en el Design Center . . . 39
5.3.2 Creació i estructuració de projectes a Anypoint Studio . . . 40
5.3.3 Estructura comuna en totes les API . . . 41
5.4 Desenvolupament de les API per capes . . . 43
5.4.1 Capa de sistema . . . 44
5.4.2 Capa de procés . . . 50
5.4.3 Capa d'experiència . . . 55

6 Implantació **57**
6.1 Entorns . . . 57
6.2 Desplegar entorns . . . 58

7 Proves **59**
7.1 Proves locals . . . 59
7.2 Proves end-to-end . . . 60
7.2.1 Postman . . . 60
7.3 Proves de càrrega . . . 61
7.3.1 JMeter . . . 62

8 Conclusions **63**
8.1 Treballs futurs . . . 63
8.2 Relació amb els estudis . . . 64

Bibliografia **67**

Apèndix

A Objetivos de desarrollo sostenible **69**

Índex de figures

2.1 Model punt a punt . . . 7
2.2 Model hub-and-spoke . . . 8
2.3 Model de bus de serveis empresarials . . . 8
2.4 Model de comunicació per transferència d'arxius . . . 9
2.5 Model de comunicació de base de dades compartida . . . 9
2.6 Model de comunicació per RPI . . . 10
2.7 Model de comunicació per missatgeria . . . 10
2.8 Representació d'una arquitectura amb connectivitat API-Led . . . 11

3.1 Esquema de funcionament del projecte inicial . . . 15
3.2 Funcionamient de la metodologia àgil Scrum . . . 19
3.3 Funcionamient del model en cascada . . . 20

4.1 Esquema de l'arquitectura proposada . . . 22
4.2 Diagrama de fluxos per a la creació de comandes mitjançant una crida a e-sistema-comandes . . . 24
4.3 Diagrama de fluxos per a eliminar de comandes mitjançant una crida a e-sistema-comandes . . . 24
4.4 Diagrama de fluxos per a la consulta de comandes mitjançant una crida a e-sistema-comandes . . . 25
4.5 Diagrama de fluxos per a la consulta de comandes mitjançant una crida a e-sistema-comandes . . . 26
4.6 Diferència entre integracions sense model canònic de dades i amb aquest. 28

5.1 Correspondència de quantitat de vCores amb memòria . . . 36
5.2 Especificació RAML de l'API s-gls . . . 39
5.3 Estructura del projecte s-envialia . . . 41
5.4 APIkit router i http listener autogenerats amb loggers i redirecció de fluxos 42
5.5 Gestió d'errors de l'API . . . 43
5.6 Implementació de s-gls . . . 44
5.7 Implementació de s-envialia . . . 45
5.8 Implementació de s-mrw . . . 47
5.9 Implementació de s-seur . . . 48
5.10 Implementació de s-victransa . . . 49
5.11 Implementació de s-db . . . 50
5.12 Fluxos que redirigeixen el fil d'execució al flux del transportista corresponent 51
5.13 Flux de transportista que només gestiona una comanda per crida . . . 52
5.14 Flux de transportista que gestiona diverses comandes per crida . . . 53

5.15 Flux de transportista que només gestiona una comanda per crida 53
5.16 Flux de transportista que gestiona diverses comandes per crida 54
5.17 Fluxos de consulta de comandes de la capa de procés del transportista Envialia 55
5.18 Implementació d'e-sistema-comandes 55

7.1 Estructura de peticions a l'esquerra i exemple de creació d'una comanda . 61

Índex de taules

4.1 Endpoints de les API 26
4.2 Mapeig entre el model canònic de dades i els models de dades de GLS i Envialia 29
4.3 Mapeig entre el model canònic de dades i els models de dades de MRW i Victransa 30
4.4 Mapeig entre el model canònic de dades i el model de dades de Seur . . . 31

CAPÍTOL 1

Introducció

En l'actualitat, són moltes les empreses que han d'enfrontar-se al repte que suposa integrar una àmplia varietat de serveis i processos per tal de poder garantir un servei de qualitat al client. Tanmateix, si la integració dels serveis no s'implementa de manera escalable, els costos de manteniment, actualització i inclús d'integració de noves aplicacions augmenten i l'eficiència dels serveis pot veure's afectada. És per açò que és important fer ús d'una arquitectura d'integració ben definida i flexible per tal de millorar l'eficiència del negoci i reduir els costos de l'empresa.

En aquest treball s'analitzarà el disseny d'una arquitectura d'integració basada en un middleware amb la finalitat d'identificar-ne les febleses de disseny. S'explicarà com aquestes febleses es poden solucionar fent ús d'una arquitectura API-Led seguint els principis REST i dividida en capes que permet la separació de responsabilitats de les API de manera jeràrquica fomentant la reutilització de codi. Per últim es proposarà una solució aplicant els coneixements adquirits durant l'etapa universitària i a les pràctiques d'empresa demostrant els beneficis que impliquen.

Per desenvolupar la nova solució es farà ús del programari Mulesoft, que ofereix una plataforma d'integració d'aplicacions basada en la creació d'un middleware que facilita tant la comunicació entre diferents serveis i aplicacions com la transformació de dades amb ajuda del llenguatge Dataweave 2.0. Una vegada proposada la solució s'integrarà un nou servei per tal de demostrar els beneficis obtinguts.

1.1 Motivació

La informàtica és omnipresent en el nostre dia a dia i és un component clau si no el més important per a la comunicació entre diferents serveis i indústries. Aquest paper que juga la informàtica no sempre està implementat d'una manera eficient, fet que pot afectar al rendiment d'un sistema major com pot ser una empresa.

A l'hora d'integrar aplicacions, la complexitat pot suposar un obstacle important per a les empreses que volen oferir un servei de qualitat al client. Concretament, un enfocament mal plantejat pot augmentar notablement els temps de resposta del servei i

incrementar els costos tant temporals com econòmics a l'hora d'integrar nous serveis, fer manteniment o actualitzar serveis que ja han estat integrats.

Durant la meua etapa com a estudiant en l'ETSINF, especialment al llarg de l'assignatura "Integració d'aplicacions" (IAP), i durant el transcurs de les meues práctiques universitàries a DISID CORPORATION S.L. m'he adonat de la importància que té el paper de la integració d'aplicacions a l'hora d'unir eficientment els diferents components d'un sistema major, permetent també la connexió a sistemes i plataformes de tercers que poden millorar la funcionalitat i l'experiència de l'usuari.

En la meua estança a Disid, vaig tindre la possiblitat de treballar en un projecte d'integració real. Tanmateix, el projecte no seguia les directrius estudiades en l'assignatura IAP ni respectava les bones pràctiques que em recomanaren a l'empresa. Tot i que una refactorització i redisseny aportaria moltes millores, segons les preferències i necessitats del client no era viable. És per això que la idea d'aquest TFG sorgeix de les dificultats que vaig trobar a l'hora d'integrar un servei en un projecte real que no havia sigut creat seguint els principis de la integració d'aplicacions.

1.2 Objectius

El propòsit d'aquest TFG és redissenyar i refactoritzar un projecte d'integració d'aplicacions existent d'una empresa seguint els principis d'arquitectures API-Led dividida en capes. El primer objectiu d'aquest document és comparar l'arquitectura present al projecte sobre el qual partim amb una arquitectura API-Led dividida en capes. Aquest objectiu inclou com a objectius secundaris l'anàlisi de l'arquitectura actual i la proposta d'una nova solució.

Un altre objectiu és demostrar els beneficis que pot suposar una arquitectura API-Led pel que respecta a l'escalabilitat, flexibilitat i reutilització de codi. Per a dur a terme aquest objectiu, trobem l'objectiu secundari d'integrar un nou servei a la solució proposada seguint els principis API-Led.

1.3 Estructura de la memòria

Aquest document està separat en diferents capítols i apartats per tal de fer més fàcil la seua comprensió. En primer lloc, trobem l' "Estat de l'art" (2), apartat en què s'estudia l'estat en què està el mercat de la integració de dades en l'actualitat i el context de Disid en aquest mercat. És en aquest apartat on es compararan les tecnologies emprades en aquest sector i on es justificarà l'ús de les que s'han escollit per a aquest projecte.

Una vegada presentat l'estat de l'art es presentarà una anàlisi de la problemàtica que es resoldrà en aquest TFG. Aleshores es presentaran i compararan diferents formes d'afrontar la situació, entre les quals s'escollirà una en concret. Aquesta informació es presentarà en el capítol "Anàlisi del problema" (3).

A partir de la informació presentada en el capítol mencionat s'elaborarà el disseny per a la solució i es justificaran l'arquitectura escollida i la tecnologia que s'emprarà. Tota aquesta informació es recull al capítol "Disseny de la solució" (4). És també en aquesta part on es presentarà la tecnologia de què farem ús per a la nostra solució.

El següent capítol és el "Desenvolupament de la solució" (5). És ací on es mostrarà la implementació de la solució. Com que el projecte del qual es parteix utilitza els serveis reals d'una empresa client, la implementació es farà sobre un servei de simulació.

A continuació trobem la part d' "Implantació" (6). En aquesta part s'explica quin és el procés a dur a terme per passar d'un entorn de desenvolupament a un de producció.

A l'apartat de "Proves" (7) es mostrarà com s'han realitzat les proves necessàries per a assegurar-se que el projecte desenvolupat té el funcionament correcte i esperat.

Per últim trobem les "Conclusions", on es reflexiona sobre el que s'ha aprés al llarg del treball. Aquest últim capítol serà on s'avalue si s'ha aconseguit complir amb els objectius que proposava el treball a la secció "Objectius" (1.2). És també en aquesta secció on es comentaran les dificultats trobades i com s'han solucionat.

CAPÍTOL 2

Estat de l'art

La integració d'aplicacions es defineix com el procés de connectar aplicacions i serveis de manera independent amb el propòsit de coordinar-los per tal de proveir una interfície unificada. Aquest procés es materialitza mitjançant l'ús de principis de l'arquitectura de sistemes que normalment, en aquest context, involucren el desenvolupament d'un middleware. [1][2][3]

El concepte d'integració d'aplicacions sorgeix sobre l'any 2000 per la necessitat emergent de les empreses de poder connectar-se amb aplicacions internes i externes. A causa del ràpid avanç de la tecnologia, les empreses incorporaren nous serveis i aplicacions que millorarien l'eficàcia i l'eficiència del sistema del qual disposaven. No obstant això, en molts casos no s'aconseguia la comunicació entre les diferents aplicacions. La integració d'aplicacions va aparèixer com a solució a aquest problema i a poc a poc va anar evolucionant fins al punt de convertir-se en un element clau per al funcionament i la transformació digital de les empreses. [3] [4]

El repte més gran de la integració d'aplicacions és proporcionar compatibilitat entre les tecnologies actuals i les noves. L'objectiu principal de tot projecte d'integració d'aplicacions és facilitar la comunicació entre dues o més aplicacions. Tot i això, també trobem l'objectiu de permetre i simplificar la incorporació de noves aplicacions, per la qual cosa és necessària la cooperació entre les aplicacions. La clau per aconseguir açò és que les aplicacions compartisquen les dades entre elles, fet que es coneix com a "democratització de les dades". Per aconseguir açò cada aplicació ha d'oferir documentació dels serveis que ofereix a la resta. [5]

En l'actualitat, són moltes les empreses que disposen d'integracions de diversos serveis i aplicacions que es complementen per oferir una funcionalitat major. Tanmateix, aquesta integració no sempre està plantejada sobre una arquitectura adequada. Aquest fet implica que les empreses poden trobar afectat el rendiment dels seus serveis. Per altra banda, una arquitectura que no siga escalable suposa un esforç addicional a l'hora de fer actualitzacions, manteniment o noves integracions a comparació d'una arquitectura flexible i ben plantejada.

Durant les pràctiques realitzades a l'empresa Disid he tingut l'oportunitat de treballar en diferents projectes i he comprovat la importància del paper que juga l'arquitectura

dels sistemes d'integració. He pogut comparar projectes d'integració d'aplicacions que tenien diferents arquitectures i he entés que no sols existeix una manera de plantejar una integració nova i que per a cada integració hi ha un tipus d'arquitectura adequada.

2.1 Tipus d'integracions

Fins ara hem estat parlant d'integració d'aplicacions en termes generals i hem marcat com a objectiu d'aquesta connectar diferents serveis entre sí i facilitar la comunicació entre aquests. Tanmateix, cal destacar que depenent d'alguns factors podem diferenciar certs tipus d'integracions.

2.1.1. Segons el propòsit

En primer lloc, podem destacar quatre tipus d'integracions segons el seu propòsit: [6]

- Integració de sistemes heretats: Aquest tipus d'integració té com a objectiu integrar aplicacions modernes en sistemes existents que han quedat obsolets que no poden ser canviats perquè, en alguns casos, són essencials per al funcionament d'una empresa. Aquestes integracions permeten emmascarar la tecnologia vella en una interfície més moderna, com podrien ser per exemple les API Rest de les quals parlaré més endavant.
- Integració d'aplicacions empresarials: Les empreses solen tindre diversos departaments i sistemes, com poden ser el de recursos humans, el sistema de comandes o el sistema de gestió empresarial, que requereixen cooperació i comunicació per al correcte funcionament de l'empresa. L'objectiu d'aquestes integracions és unificar els subsistemes dins un entorn empresarial oferint consistència en les dades i comunicació entre els diferents components de l'empresa.
- Integració de sistemes de tercers: Aquestes integracions permeten la incorporació de noves funcionalitats a un sistema existent. Existeixen ferramentes oferides per altres empreses, com poden ser el pagament en línia que ofereix PayPal. Sovint, les empreses prefereixen incorporar aquestes ferramentes al sistema intern en compte de desenvolupar el software per a un projecte en concret pel fet que s'estalvia temps i recursos.
- Integració B2B: La integració B2B (de l'anglés *business-to-business*) s'encarrega d'unir els sistemes de dues o més organitzacions permetent i automatitzant l'intercanvi de dades i documents entre les empreses.

2.1.2. Segons el model de comunicació

Per altra banda, cal comentar que quan es realitza una integració les aplicacions poden estar connectades entre si seguint diferents models de comunicació entre els quals destaquen els següents:

- Model punt a punt: Aquesta arquitectura consisteix a crear una connexió directa entre dues aplicacions per tal d'aconseguir que aquestes intercanvien informació. Això significa que per cada parella d'aplicacions existeix una connexió punt a punt. Per exemple, si volem integrar una nova aplicació a un servei amb 5 aplicacions ja integrades seguint el model punt a punt caldrà crear 5 noves connexions (una per aplicació suposant que la nova aplicació requerisca comunicar-se amb totes les altres). En el cas de fer canvis a una aplicació s'haurien de modificar també totes les connexions amb les aplicacions amb les quals intercanvia informació.

  Sabent açò, podríem concloure que aquest model podria ser viable per a projectes on s'integren unes poques aplicacions només. No obstant això, caldria tindre en compte que a mesura que s'incorporen noves aplicacions al projecte, el cost d'integració i manteniment augmenta de manera exponencial, per la qual cosa podríem dir que aquest model no és escalable. [6]

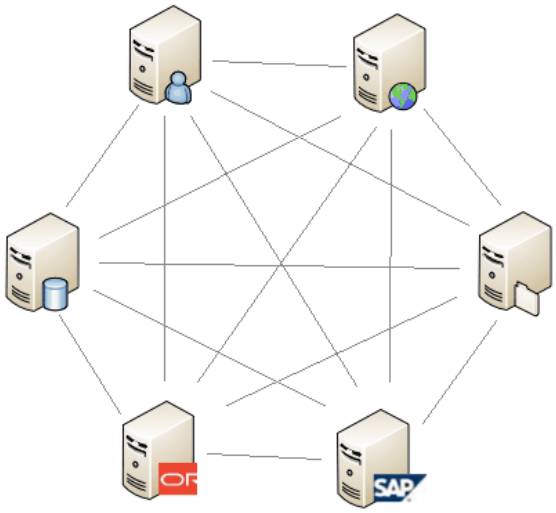

Figura 2.1: Model punt a punt

- Model hub-and-spoke: En aquest model existeix una aplicació que funciona com a nucli central connectant diverses aplicacions. Totes les comunicacions entre qualsevol de les aplicacions han de passar per aquest nucli central, que actua com a punt central de control i coordinació. El model hub-and-spoke ofereix diversos avantatges en termes de control, escalabilitat i manteniment. En primer lloc, permet gestionar de manera eficient les comunicacions entre les aplicacions, ja que el nucli central pot aplicar polítiques, transformacions de dades i lògica de negoci.

  En comparació del model punt a punt, el model hub-and-spoke ofereix una escalabilitat molt major pel fet que a l'incorporar una nova aplicació només cal establir la connexió amb el nucli central. No obstant això, trobem alguns inconvenients. Com totes les comunicacions depenen d'un nucli central, a mesura que augmenta la càrrega de treball la infraestructura dependrà del correcte funcionament d'una sola aplicació. Si aquest nucli falla, falla tot el sistema. Per altra banda, tot i que és més escalable que el model punt a punt, pot suposar una gran complexitat de gestió si s'integren moltes aplicacions.[6][7]

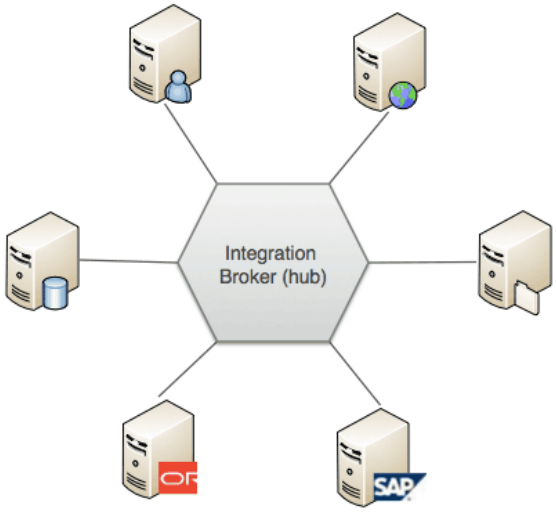

Figura 2.2: Model hub-and-spoke

- Bus de serveis empresarials: També conegut com model ESB (de l'anglés *Enterprise Service Bus*), el model de bus de serveis empresarials permet la comunicació i integració de diferents aplicacions i serveis dins d'una organització. L'ESB aconsegueix aquesta comunicació mitjançant el desenvolupament d'un middleware que connecta múltiples sistemes o aplicacions.

  El model ESB ofereix una major flexibilitat ja que presenta un adaptador o endpoint per a cada aplicació, a diferència del model hub-and-sopke que té un únic adaptador per a totes les aplicacions. Podríem dir que el model ESB és una versió millorada del model hub-and-spoke, ja que ofereix solucions més flexibles i escalables per a la integració d'aplicacions. Altre avantatge d'aquest model és que, al disposar d'un adaptador propi per a cada aplicació, aquestes poden acoplar-se i desacoplar-se al sistema sense afectar altres aplicacions. [6][7]

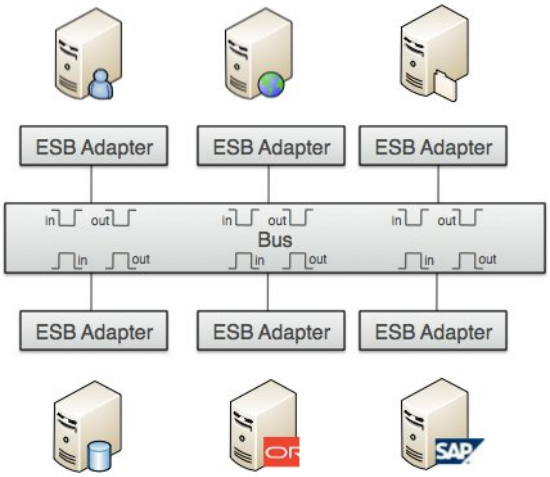

Figura 2.3: Model de bus de serveis empresarials

2.2 Tipus de comunicacions

Per últim, m'agradaria comentar que existeixen diverses maneres de comunicar dues aplicacions en una integració. Algunes de les més comunes són les següents: [8]

- Transferència d'arxius: És un model simple que permet la comunicació entre dues aplicacions mitjançant la transferència d'arxius, normalment emprant el protocol FTP (de l'anglés *File Transfer Protocol*) o SFTP (de l'anglés *Secure-FTP*). És senzill a l'hora de plantejar-lo, però cal tindre en compte que és el desenvolupador qui ha de configurar el procés d'exportació en l'aplicació que envia dades i el d'importació en l'aplicació que rep les dades. Per altra banda, no podem oblidar que el processament de fitxers pot suposar un cost temporal en cada comunicació.

Figura 2.4: Model de comunicació per transferència d'arxius

- Base de dades compartida: Aquest model de comunicació es basa en l'existència d'una base de dades comuna a totes les aplicacions. D'aquesta manera aconseguim una comunicació ràpida i consistència de dades. No obstant això, si dues o més aplicacions intenten modificar les mateixes dades simultàniament poden generar-se conflictes. Un repte d'aquest model és dissenyar un esquema per a la base de dades que s'adapte a totes les aplicacions. Per últim, cal remarcar que no totes les aplicacions seran sempre compatibles en aquest model. Per exemple, si volem realitzar una integració B2B (explicada en la secció 2.1.1) és probable que les dues organitzacions no vulguen crear una base de dades compartida.

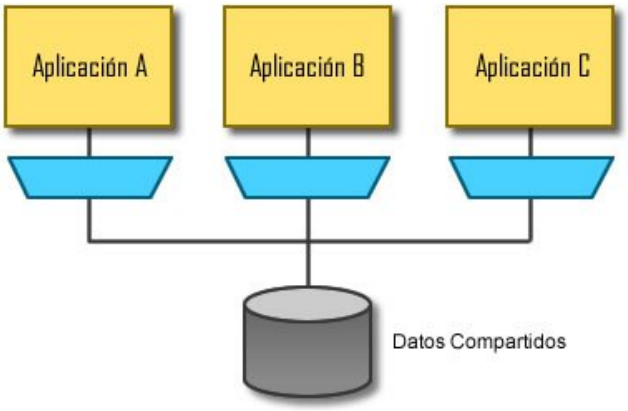

Figura 2.5: Model de comunicació de base de dades compartida

- Invocació remota de procediments: També conegut com a RPI (de l'anglés *Remote Procedure Invocation*), aquest model consisteix en la invocació de procediments o funcions de manera remota. Normalment, les aplicacions disposen d'una documentació breu que especifica quines són les funcions que ofereix. Alguns exemples d'aquest model de comunicació poden ser les API REST o els serveis SOAP. Un inconvenient d'aquest tipus de comunicació és que per tal de comunicar-se amb una aplicació, sol ser necessària la transformació de dades al model de dades que requereix l'aplicació.

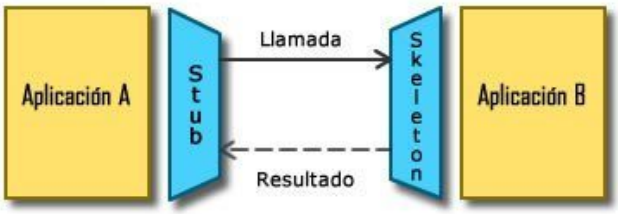

Figura 2.6: Model de comunicació per RPI

- Missatgeria: Aquest model proporciona un mecanisme de comunicació que permet enviar, rebre i gestionar els missatges de manera asíncrona. Per poder comunicar dues aplicacions existeix un gestor de cues que emmagatzema els missatges fins que s'entreguen a l'aplicació destinatària. El fet que la comunicació siga asíncrona evita que hi haja bloquejos i suposa que l'emissor i receptor no han d'estar en funcionament simultani. És per açò que aquest model pot millorar l'eficiència i flexibilitat respecte a altres models que hem vist depenent del projecte. Per altra banda, trobem alguns inconvenients, com poden ser la complexitat en el tractament d'errors o el control de l'ordre i la consistència dels missatges.

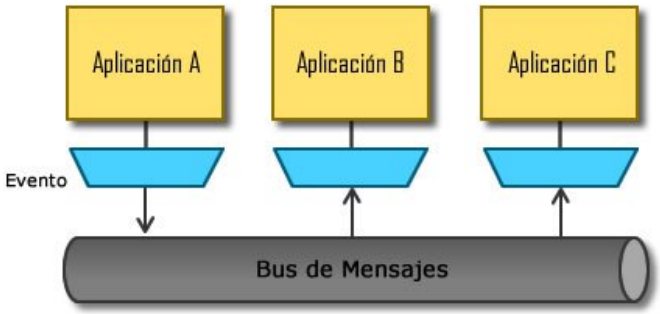

Figura 2.7: Model de comunicació per missatgeria

2.3 Connectivitat dirigida per API

La connectivitat dirigida per API, també com *API-Led connectivity* en anglés, «és una manera metòdica de connectar dades a aplicacions mitjançant API reutilitzables i proposades dins de l'ecosistema d'una organització» segons la descriu Prashant Choudhary

[9]. Aquesta metodologia ofereix una gran flexibilitat i reutilització de codi a l'hora de desenvolupar projectes dins una organització. A causa de la gran quantitat de components tecnològics dins una empresa i al constant creixement, les connexions punt a punt (explicades en el punt 2.1.2) ja no són una opció.

Per altra banda, la connectivitat dirigida per API està plantejada per a adoptar nous components de manera escalable. Aquest model parteix d'un ESB (explicat al punt 2.1.2) i inclou la comunicació per API d'una manera organitzada [10]. Concretament, el model API-Led dividit en capes presenta un mètode per a crear un ecosistema i incorporar-hi nous components. Existeix una divisió en capes que formen una arquitectura jeràrquica en què les API se situen en una única capa segons la funció que tinguen. Normalment trobem 3 capes: la capa de sistema, la capa de procés i la capa d'experiència. [9]

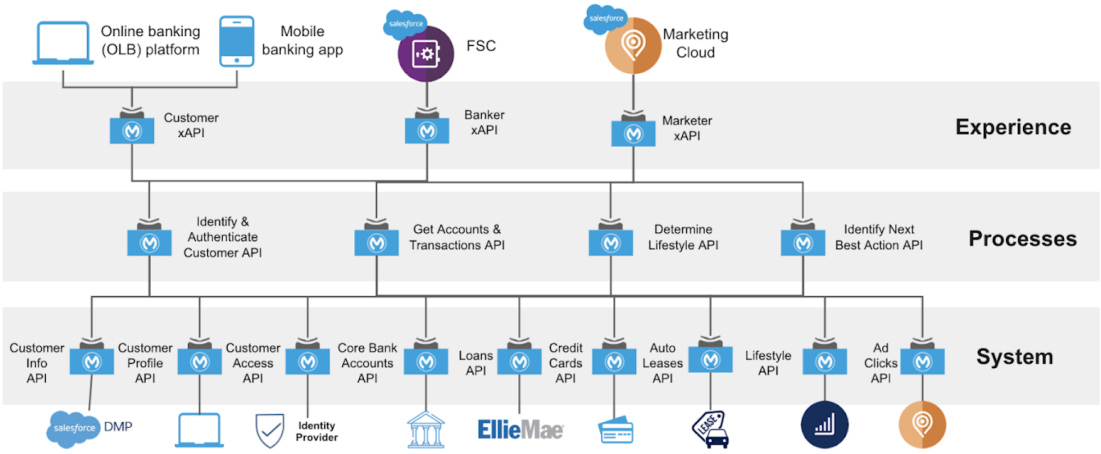

Figura 2.8: Representació d'una arquitectura amb connectivitat API-Led

- Capa de sistema: Les API d'aquesta capa s'encarreguen d'aïllar la complexitat i els canvis dels diferents sistemes que existeixen a l'empresa. Aconsegueixen açò oferint una interfície (API) amb els serveis del sistema. D'aquesta manera, no cal conèixer la tecnologia del sistema en qüestió, simplement fent una crida a l'API de sistema podem accedir als serveis desitjats. Aquestes API de sistema es poden reutilitzar en diversos projectes.

- Capa de procés: Les API de procés tenen com a funció principal orquestrar les funcionalitats que ofereixen les API de la capa de sistema i aplicar la lògica de negoci entre les diferents API. També és en aquesta capa on s'apliquen la majoria de les transformacions de dades i on solem trobar l'ús d'un model canònic de dades (que explicarem més endavant).

- Capa d'experiència: La finalitat de les API d'aquesta capa és proporcionar una interfície unificada i optimitzada per a les interaccions dels usuaris finals. La funció clau d'aquesta capa és l'agregació de dades de les API de procés necessàries per al servei que requereix l'usuari final.

La implantació d'una arquitectura API-Led pot aparentar la incorporació d'intermediaris que, com es pot intuir, podria afegir temps de processat i afectar el temps de resposta. No obstant això, les arquitectures amb connectivitat dirigida per API permeten mecanismes com el *caching* o les peticions de manera asíncrona que poden inclús reduir el temps de resposta inicial. [11]

CAPÍTOL 3

Anàlisi del problema

Aquest capítol té com a finalitat examinar de manera detallada el projecte existent del qual parteix aquest treball, identificant les seues funcionalitats, beneficis i, al mateix temps, les seues limitacions. Tot i que el projecte actual és funcional i ha estat implementat amb èxit, s'han detectat algunes debilitats que podrien minvar l'eficiència i la qualitat general del sistema.

Aquestes debilitats poden ser atribuïdes al disseny inicial del projecte i a la manca d'una visió més completa i escalable. En aquest capítol, s'exploraran a fons aquestes debilitats i es proposarà un canvi de disseny per abordar-les i millorar el rendiment, la flexibilitat i l'experiència de l'usuari final. L'objectiu és proporcionar una anàlisi exhaustiva del problema i establir les bases per a la proposta d'una solució més robusta i optimitzada.

3.1 Debilitats d'un model hub-and-spoke

El projecte inicial segueix una arquitectura hub-and-spoke, model que pot presentar algunes debilitats que podrien afectar el seu rendiment i eficiència. Algunes d'aquestes debilitats poden incloure la dependència excessiva del node central per a la comunicació entre els diferents nodes, la manca de flexibilitat i escalabilitat en l'afegit de nous nodes o la complexitat en la gestió i manteniment de la infraestructura.

Una debilitat associada a aquest model és la seua relativa manca d'autonomia en els nodes, ja que totes les comunicacions han de passar a través del node central. Això pot generar un punt únic de fallada i un possible efecte de coll de botella en el tràfic de dades, afectant la velocitat i l'eficiència del sistema en conjunt, podent crear també pèrdua de dades.

L'arquitectura hub-and-spoke pot presentar dificultats en l'afegiment de nous nodes a la xarxa, ja que s'ha de connectar cada nou node al node central. Aquest model és escalable només fins a cert punt pel que afegir un node nou pot suposar un procés complex en projectes que integren molts serveis diferents. A més, l'escalabilitat pot ser un desafiament en aquest tipus d'arquitectura, perquè l'afegiment de nous nodes pot requerir una reconfiguració o adaptació del node central.

D'altra banda, la gestió i manteniment de la infraestructura en una arquitectura hub-and-spoke pot resultar més complexa i requerir més recursos. És necessari garantir el funcionament adequat del node central, que actua com a punt central de control, i supervisar constantment les connexions amb els nodes per evitar interrupcions o problemes de connectivitat.

Tot i que l'arquitectura hub-and-spoke pot oferir avantatges en termes de centralització i facilitat de gestió, també pot presentar desafiaments notables en la coordinació del desenvolupament paral·lel i en la gestió de la infraestructura. Acò es deu al fet que només és una aplicació qui gestiona tots els processos intermediaris i de comunicació en les integracions.

3.2 Anàlisi del projecte inicial

L'aplicació central inicial de la qual parteix aquest projecte controla tota la integració en el context del transport i té la responsabilitat de coordinar les diferents operacions i interaccions entre els sistemes de comandes i gestions d'aquestes i els 14 transportistes integrats. Aquesta aplicació actua com a nucli central que facilita la comunicació i la transferència de dades entre les diferents parts involucrades en el procés logístic.

Una de les funcionalitats principals d'aquesta aplicació és gestionar les transformacions de dades. Com que cada transportista pot tenir formats i requisists específics, és essencial que les dades s'adapten i es transformen adequadament abans de ser enviades a cada sistema. Això implica processar i convertir les dades en els formats i protocols necessaris per a cada transportista, assegurant la coherència i la compatibilitat de la informació.

A més de les transformacions de dades, l'aplicació central també es responsabilitza de la sincronització de les comandes amb la base de dades pròpia de l'empresa. Això implica actualitzar i mantenir un registre precís de totes les comandes, assegurant que estiguen disponibles i siguen accessibles per a totes les parts implicades en el procés logístic. Aquesta sincronització permet la centralització de les dades en un únic punt. Per altra banda, pot resultar complicat mantindre sincronitzades la base de dades amb la realitat, ja que s'han de tindre en compte alguns casos concrets, com per exemple, si una comanda no s'ha pogut crear correctament en el transportista corresponent, la informació d'aquesta comanda no pot existir a la base de dades.

Una altra funcionalitat clau de l'aplicació és la gestió de les peticions i interaccions amb els sistemes dels 14 transportistes. Cada vegada que es genera una comanda, l'aplicació s'encarrega de transformar les dades d'un model canònic al model del sistema en qüestió, comunicar-se amb el sistema corresponent de cada transportista per enviar les dades i les instruccions pertinents i per últim elaborar una resposta per a la petició inicial. Això implica establir connexions, validar la informació i assegurar que tots els detalls rellevants siguen transmesos amb precisió i puntualitat. Tot açò se suma a què en cada petició s'ha de sincronitzar la base de dades. Com es pot observar, aquesta aplica-

ció central inclou molts processos, fet pel qual si la càrrega de peticions és gran podrien produir-se falles.

Per proporcionar una visió més clara i comprensible del funcionament del sistema, he inclòs un esquema general que respresenta el funcionament d'aquest projecte d'integració. Aquesta representació gràfica mostra els formats de les dades que s'intercanvien en cada connexió. En aquesta figura es pot observar que és la mateixa aplicació la que gestiona tots els processos necessaris per a dur a terme la integració.

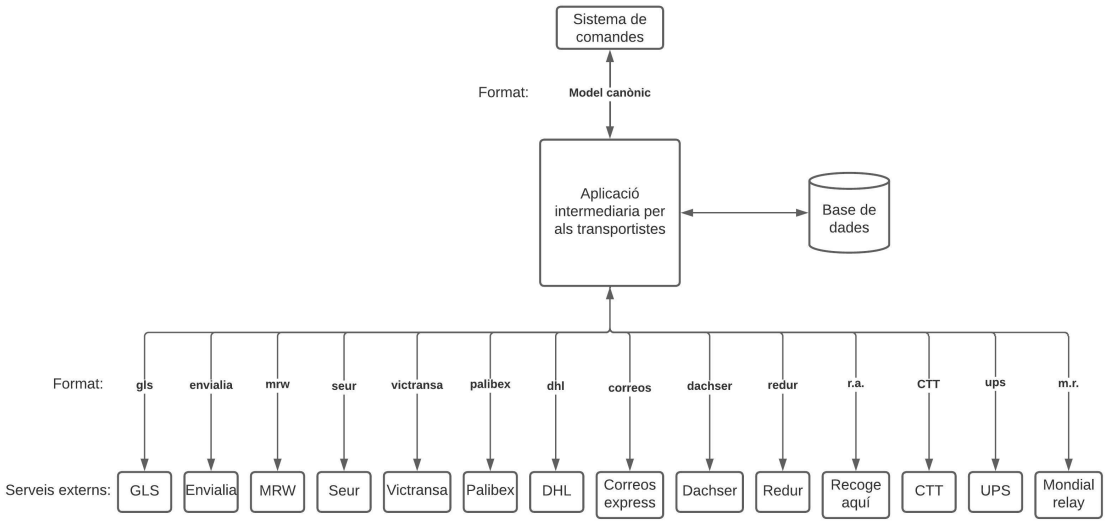

Figura 3.1: Esquema de funcionament del projecte inicial

No obstant això, malgrat la seva importància i funcionalitat crítica, l'aplicació central també planteja alguns desafiaments i problemàtiques. Un dels principals reptes és la gestió del desenvolupament en un entorn de treball paral·lel, especialment si el projecte involucra un equip d'informàtics. És necessari coordinar i sincronitzar els esforços de tots els membres de l'equip per garantir un desenvolupament coherent i cohesiu de l'aplicació, evitant conflictes i duplicacions de tasques.

Durant les meues pràctiques a DISID, vaig tindre l'oportunitat de treballar en la integració d'un dels transportistes juntament amb un altre membre de l'equip. No obstant això, vam trobar alguns desafiaments en la sincronització i la coordinació del nostre treball a causa de l'arquitectura plantejada. La naturalesa de centralitzar tots els procesos d'integració en una sola aplicació de l'arquitectura hub-and-spoke va limitar la nostra capacitat de treballar de manera paral·lela i avançar en la integració de forma més eficient. Aquesta limitació es va traduir en un desenvolupament més lent i va requerir un esforç addicional per coordinar-nos i assegurar-nos que els canvis i les tasques s'implementaven de manera coherent.

A més, la gestió i el manteniment de la infraestructura associada a l'aplicació central també pot resultar complexa i requerir recursos significatius. És crucial assegurar el correcte funcionament del node central i supervisar constantment les connexions amb els 14 transportistes integrats per evitar interrupcions o problemes de connectivitat. Això

implica garantir un entorn d'execució estable i escalable, gestionar les actualitzacions i respondre ràpidament a qualsevol incidència o error que puga sorgir en el procés d'integració, ja que un error en un transportista podria col·lapsar tota l'aplicació.

3.3 Possibles solucions

Existeixen diverses solucions alternatives a la proposta inicial d'integració d'aplicacions que podrien solventar alguns del problemes que sorgeixen al emprar el model hub-and-spoke. Considerant les necessitats i objectius del projecte, s'analitzaran tres alternatives populars: l'Arquitectura Orientada a Serveis (SOA), els microserveis i la creació d'un Enterprise Service Bus (ESB) utilitzant una arquitectura API-Led. Cada una d'aquestes alternatives té característiques úniques i pot oferir alguns avantatges concrets en termes d'escalabilitat, flexibilitat i eficiència en la integració d'aplicacions. [12]

3.3.1. Arquitectura Orientada a Serveis (SOA)

L'Arquitectura Orientada a Serveis (SOA) és un enfocament amplament adoptat per a la integració d'aplicacions. En el marc de SOA, els sistemes són dissenyats com a conjunt de serveis interoperables que es poden utilitzar per a diverses funcionalitats. Els serveis SOA són components autònoms que s'exposeixen a través d'interfícies i poden ser consumits per altres aplicacions mitjançant sol·licituds de servei. Aquest enfocament permet la reutilització i l'orquestació flexible dels serveis per satisfer les necessitats de diverses aplicacions. Aquest tiups d'arquitectures es basen en la idea de construir un catàleg de serveis que puguen ser utilitzats per múltiples aplicacions, proporcionant interoperabilitat i facilitant la integració mitjançant l'ús de protocols estàndard.

3.3.2. Microserveis

Els microserveis són una arquitectura basada en la descomposició de les aplicacions en components petits i autònoms coneguts com a microserveis. Cada microservei és responsable d'una única funcionalitat específica i es pot desenvolupar, implementar i escalar de manera independent. Aquests microserveis són accessibles mitjançant interfícies de programació (APIs) i es poden connectar entre si per a dur a terme tasques més complexes. L'ús de microserveis permet una major agilitat, flexibilitat i mantenibilitat, ja que els canvis en un microservei no afecten la resta de l'arquitectura. A més, els microserveis faciliten la paral·lelització del desenvolupament, ja que diferents equips poden treballar en microserveis independents sense afectar el progrés global del projecte. Això pot accelerar els cicles de desenvolupament i millorar la productivitat de l'equip.

3.3.3. Enterprise Service Bus (ESB) basat en una arquitectura API-Led

Una altra alternativa és la creació d'un Enterprise Service Bus (ESB) utilitzant una arquitectura API-Led. En aquest enfocament, s'utilitza un ESB per a gestionar les comu-

nicacions i integracions entre les diferents aplicacions i serveis. L'arquitectura API-Led se centra en l'exposició d'APIs ben dissenyades i reutilitzables per a facilitar la integració. L'ESB actua com a punt central de control, gestionant les transformacions de dades, la sincronització de comandes amb la base de dades de l'empresa i les peticions als sistemes corresponents de cada transportista. L'ús d'una arquitectura API-Led permet una major flexibilitat i escalabilitat, ja que les APIs poden ser reutilitzades i escalades de manera independent, permetent canviar la quantitat de recursos que requereix cada component de l'ESB. A més, proporciona una millor visibilitat i control sobre les integracions, ja que l'ESB actua com a intermediari i facilita la supervisió i gestió centralitzada de les interaccions.

3.4 Solució proposada

Per tal de resoldre de manera efectiva els problemes que planteja el model hub-and-spoke exposats en la secció 3.1, propose la implementació d'un Enterprise Service Bus (ESB) amb una arquitectura API-Led, utilitzant APIs REST-ful per a gestionar les integracions. Aquesta solució busca superar les limitacions de dependència excessiva del node central, manca d'autonomia en els nodes, complexitat en l'afegiment de nous nodes i la gestió de la infraestructura. Mitjançant l'ús d'aquesta arquitectura, es busca millorar el rendiment, l'escalabilitat i la flexibilitat del sistema d'integració d'aplicacions.

En primer lloc, pel que fa al problema d'excessiva dependència del node central, cada aplicació o servei seria exposat com a una API REST-ful, permetent als nodes comunicar-se directament entre ells. Això elimina la necessitat que totes les comunicacions passen pel node central, evitant així la creació d'un punt únic de fallada i reduint el possible efecte de coll de botella en el tràfic de dades. Cada node pot ser autònom i comunicar-se directament amb altres nodes, millorant la velocitat i eficiència del sistema en conjunt. Al tractar-se de nodes independents, sorgeix la possibilitat de dedicar més recursos a un node que a altre. Açò permet una millor optimització dels recursos.

Quant a la millora de l'escalabilitat i afegiment de nous nodes, amb l'arquitectura API-Led l'afegiment de nous nodes resulta una tasca més senzilla. Cada nou node pot ser implementat com una nova API, que es pot connectar als altres nodes mitjançant les seues pròpies APIs REST-ful. Aquest enfocament permet una escalabilitat més flexible, ja que l'afegiment de nous nodes no requereix una reconfiguració o adaptació del node central. Cada node pot ser afegit de manera independent, agilitzant així el procés d'integració de nous serveis o aplicacions.

Un altre dels principals avantatges de l'arquitectura API-Led és la seua capacitat per facilitar el desenvolupament paral·lel. Cada API pot ser desenvolupada i mantinguda de manera independent, permetent als equips de desenvolupament treballar simultàniament en diferents integracions sense dependre d'un únic punt central. El desenvolupament paral·lel és possible ja que l'especificació de l'API descriu la funcionalitat de l'aplicació abans inclús que aquesta estiga implementada, pel que altres aplicacions que depenen d'aquesta saben com es comportarà i de quines funcionalitats disposarà, permetent

així el seu desenvolupament en paral·lel. Això millora la velocitat de desenvolupament i possibilita una millor coordinació entre els equips. [10] [13]

Finalment, m'agradaria anotar que he escollit aquesta com a la soluició a executar perquè solventaria totes les problemàtiques exposades en el punt 3.1, cosa que no vol dir que siga la única que les solvente. Per altra banda, aquest model és el que he més he treballat durant la meua etapa com a estudiant i també és el que empra DISID, l'empresa on he realitzat les pràctiques. És una solució flexible, escalable i lleugera que es pot adaptar a qualsevol tipus de projecte d'integració.

3.5 Metodologies de treball

La naturalesa de l'arquitectura de la solució proposada permet el desenvolupament del projecte mitjançant metodologies àgils, que són altament recomanables per aconseguir flexibilitat i adaptabilitat durant el procés de desenvolupament. Dins d'aquest context, es recomana l'ús de la metodologia àgil Scrum, ja que s'ajusta perfectament a les característiques del projecte.

Les metodologies àgils són formes de treballar en equips que ens permeten adaptarnos i ser flexibles en els projectes. En lloc de seguir un pla rígid, les metodologies àgils ens animen a treballar en iteracions curtes i col·laborar estretament amb el client. Això significa que podem lliurar resultats parcialment acabats més ràpidament i ajustar el que fem en funció del que necessite el client. Algunes de les metodologies àgils més conegudes són Scrum, Kanban i XP (*Extreme Programming*). [14]

Scrum és una metodologia àgil que es basa en un enfocament iteratiu i incremental per al desenvolupament de projectes. Es caracteritza per la seua flexibilitat, adaptabilitat i col·laboració en equip. Aquesta metodologia es basa en un conjunt de cerimònies i processos que permeten organitzar i gestionar el desenvolupament de manera eficient.

En la metodologia Scrum el client té un paper actiu durant tot el procés, participant en les reunions de planificació de sprints, les revisions de sprint i les retrospectives. Aquesta interacció constant permet un millor enteniment de les necessitats i expectatives del client, i ajuda a evitar desviacions i malentesos durant el desenvolupament.

En Scrum, el projecte es divideix en iteracions anomenades "sprints". Cada sprint té una durada fixa, generalment de dues a quatre setmanes, durant les quals l'equip de desenvolupament treballa en tasques prioritzades del producte. El producte és desenvolupat de manera incremental, amb l'entrega de funcionalitats funcionals al final de cada sprint.

Les cerimònies de Scrum inclouen la reunió de planificació de sprint, on s'identifiquen les tasques i es crea el backlog del sprint; les reunions diàries d'equip, on els membres comparteixen els seus progressos i identifiquen els possibles obstacles; la revisió de sprint, on es mostra el treball fet i es recull el *feedback* dels stakeholders; i la revisió de sprint, on l'equip reflexiona sobre el seu rendiment i busca millores per als propers sprints. [15]

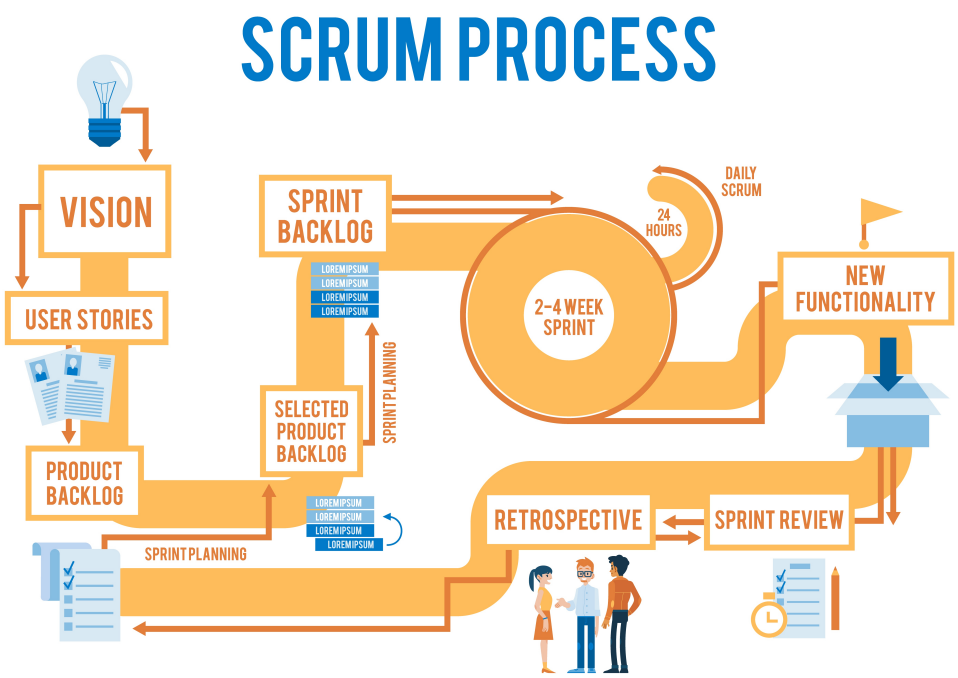

Figura 3.2: Funcionamient de la metodologia àgil Scrum

L'ús de Scrum en aquest projecte permetria una gestió efectiva del desenvolupament, amb entregues regulars i la possibilitat de respondre a canvis i requeriments emergents de manera àgil. La seua naturalesa col·laborativa i iterativa fomenta la comunicació i l'adaptabilitat, assegurant que el resultat final siga satisfactori per als stakeholders.

Tot i això, per al desenvolupament d'aquest treball s'ha emprat una metodologia de treball que es basa en el model en cascada. Aquest model és un enfocament seqüencial, en què cada fase del projecte s'inicia després que la fase anterior ha estat completada i aprovada. A diferència de mètodes àgils com Scrum, el model en cascada no permet canvis significatius una vegada que una fase ha començat [16]. Les fases principals d'aquest model solen ser: definició de requisits, anàlisi, disseny, implementació, prova, integració i manteniment.

La raó per la qual s'ha triat el model en cascada per a aquest treball s'ha desenvolupat de manera individual, i no hi han participat clients o stakeholders externs amb els quals col·laborar de manera regular. Això significa que no hi ha la necessitat d'adaptar-se a canvis o requeriments emergents com succeeix en mètodes àgils com Scrum, on la retroalimentació constant del client és crucial. En lloc d'això, el model en cascada permet una planificació i execució més estructurada de les diferents fases del projecte.

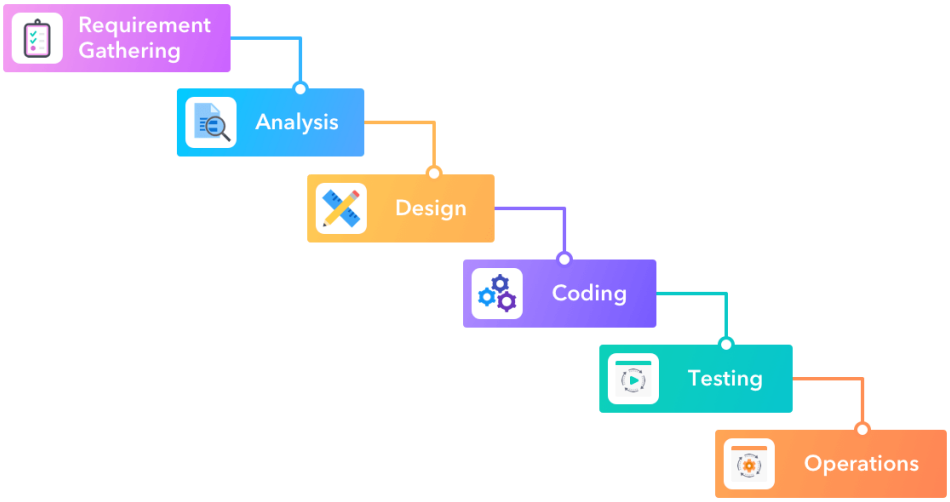

Figura 3.3: Funcionamient del model en cascada

Així doncs, en aquest cas, el model en cascada ha estat l'opció més adequada, ja que ha permés una gestió eficient de les diferents etapes del projecte de forma individual. No obstant això, és important assenyalar que, en altres contextos on la col·laboració amb stakeholders externs o la necessitat d'adaptació a canvis siguen més importants, es podria considerar l'ús de mètodes àgils com Scrum per a una gestió més àgil i flexible del projecte.

CAPÍTOL 4

Disseny de la solució

En aquesta secció del treball, ens centrarem en el disseny de la solució proposada per al projecte d'integració de diversos transportistes del que partim. L'objectiu principal és desenvolupar una arquitectura eficient i flexible que permeta la comunicació fluida entre el sistema de comandes i els diferents transportistes integrats. Optem per un model ESB amb comunicació API-Led, el qual ens permetrà organitzar les relacions entre les capes per a un intercanvi segur i eficient de dades. També incorporarem un model canònic per a facilitar la interoperabilitat entre els diferents formats de dades dels transportistes. Així, buscarem assegurar una integració coherent i optimitzada, oferint una interfície unificada per al sistema de comandes i garantint la gestió adequada de les comunicacions amb cada transportista amb el menor nombre de transformacions de dades possible.

NOTA: Cal remarcar que per tal de no fer un treball massa extens, només es mostrarà el disseny en detall de 5 transportistes: GLS, Envialia, MRW, Seur i Victransa.

4.1 Arquitectura del sistema

La solució proposada per al projecte consistirà en l'ús d'un ESB (Enterprise Service Bus) que segueix una arquitectura jeràrquica amb comunicació API-Led. API-Led Connectivity és un enfocament arquitectònic que posa les APIs en el centre de la integració d'aplicacions i sistemes. Consisteix en la creació d'APIs ben definides i reutilitzables per connectar els diferents components i serveis d'una organització.

Aquesta arquitectura està composta per tres capes principals: la capa de sistema, la capa de procés i la capa d'experiència.

La capa de sistema és la capa inferior i es refereix a les interaccions amb els sistemes subministrats pels diferents transportistes. Cada transportista disposarà d'un adaptador específic que facilitarà la comunicació amb el seu sistema. Aquest adaptador, que es materialitzarà en una API de sistema, garantirà que les dades s'intercanvien correctament i de manera estandarditzada amb el transportista corresponent. Les API d'aquesta capa seran accessibles per la capa de procés.

La capa de procés és la capa intermèdia entre la capa d'experiència i la capa de sistema, responsable de l'orquestració i coordinació de les diverses interaccions entre les diferents funcionalitats que ofereix la capa de sistema. És també en aquesta capa on es gestionarà la sincronització dels sistemes i la base de dades. Mitjançant l'ús de regles i lògica de negoci, aquesta capa garantirà que les dades es processin correctament, realitzant transformacions si és necessari i coordinant els fluxos de treball. Això permetrà gestionar eficientment les comandes i garantir la coherència i consistència de la informació.

La capa d'experiència és la capa més alta i interactua directament amb el sistema de comandes. Aquesta capa proporcionarà una interfície unificada a través de la qual el sistema de comandes podran interactuar amb els diferents transportistes. Mitjançant crides a APIs de la capa de procés, aquesta capa podrà intercambiar informació entre els sistemes de cada transportista i el sistema de comandes.

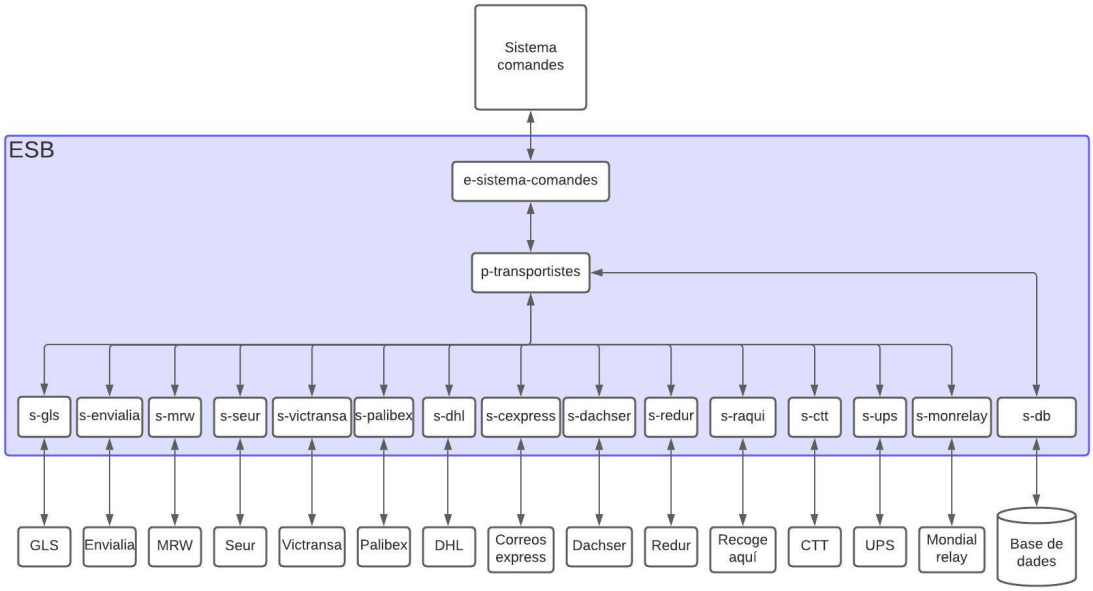

Figura 4.1: Esquema de l'arquitectura proposada

Com es pot observar en la figura 4.1, l'ESB que actua com a intermediari entre els sistemes dels transportistes i el sistema de comandes està compost per moltes aplicacions distribuides en tres capes. Cada aplicació comença amb la inicial de la capa a la que perteneix: "e" d'experiència, "p" de procés i "s" de sistema.

4.2 Disseny detallat

La solució proposada es basa en un flux de comunicació clarament definida entre les diferents capes de l'arquitectura, on cada capa farà peticions a la capa inmediatament inferior. Des del sistema de comandes, s'especifica el transportista i la capa d'experiència és l'encarregada de transmetre l'ordre de crear, eliminar o consultar una comanda a la capa de procés.

En aquest projecte faré ús d'un model canònic, concretament, el format en el que es comunica el sistema de comandes amb la capa d'experiència. Així, la capa de procés realitza la transformació del model canònic al model de dades específic de cada transportista. Després d'aquesta transformació, es fa la petició a l'API de sistema del transportista corresponent.

Arribats a aquest punt, l'API de sistema del transportista fa la crida al sistema corresponent i retorna els resultats a l'API de procés. En cas que no hi haja errors, l'API de procés crida a l'API de sistema de la base de dades per actualitzar i mantenir sincronitzada la base de dades amb la informació del transportista.

Finalment, l'API de procés transmet el resultat a la capa d'experiència, que és responsable de generar la resposta final per al sistema de comandes. Això permet una interfície unificada i estandarditzada per a la interacció amb els diferents transportistes integrats en el sistema.

Tots els transportistes ofereixen els serveis d'afegir, consultar i eliminar comandes, no obstant això, cada transportista ho fa d'una manera. Alguns, per exemple, afegeixen vàries comandes en cada petició i en altres només permeten afegir una per petició. El mateix passa a l'hora d'eliminar. Concretament en el model canònic de dades tant la creació de comandes com a l'hora d'eliminar-les es gestionen vàries comandes per petició. La capa encarregada de gestionar aquestes diferències és la capa de procés.

La creació de comandes mitjançant la funcionalitat oferida per la capa d'experiència resultarà en la creació de les comandes tant en el sistema del transportista seleccionat com en la base de dades de l'empresa.

Per altra banda, es poden eliminar comandes. De manera similar a la creació, quan fem ús de la capa d'experiència per eliminar diverses comandes d'una sola crida, aquestes s'eliminaran tant del sistema del transportista en qüestió com de la base dades.

Tant en la creació com a l'hora d'eliminar comandes, quan es tracte de vàries comandes en una sola crida a la capa d'experiència, totes les comandes seran creades o eliminades respectivament en el sistema del mateix transportista. És a dir, no es podran crear o eliminar comandes en transportistes diferents en una sola crida a la capa d'experiència.

Quant a la consulta de comandes, es consultarà una comanda per petició a la capa d'experiència. Caldrà especificar de quin transportista es vol consultar la comanda.

Per a mostrar de manera més gràfica el funcionament d'aquesta solució trobem els diagrames de fluxos de les figures 4.2, 4.3 i 4.4:

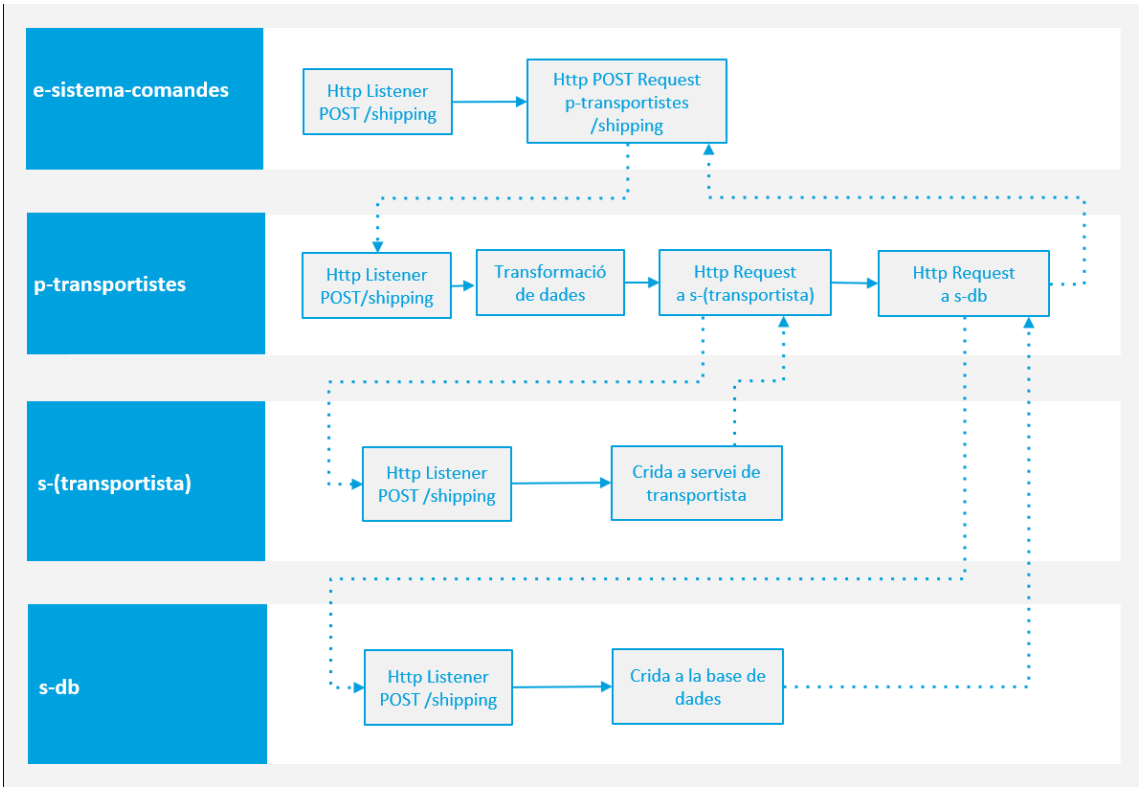

Figura 4.2: Diagrama de fluxos per a la creació de comandes mitjançant una crida a e-sistema-comandes

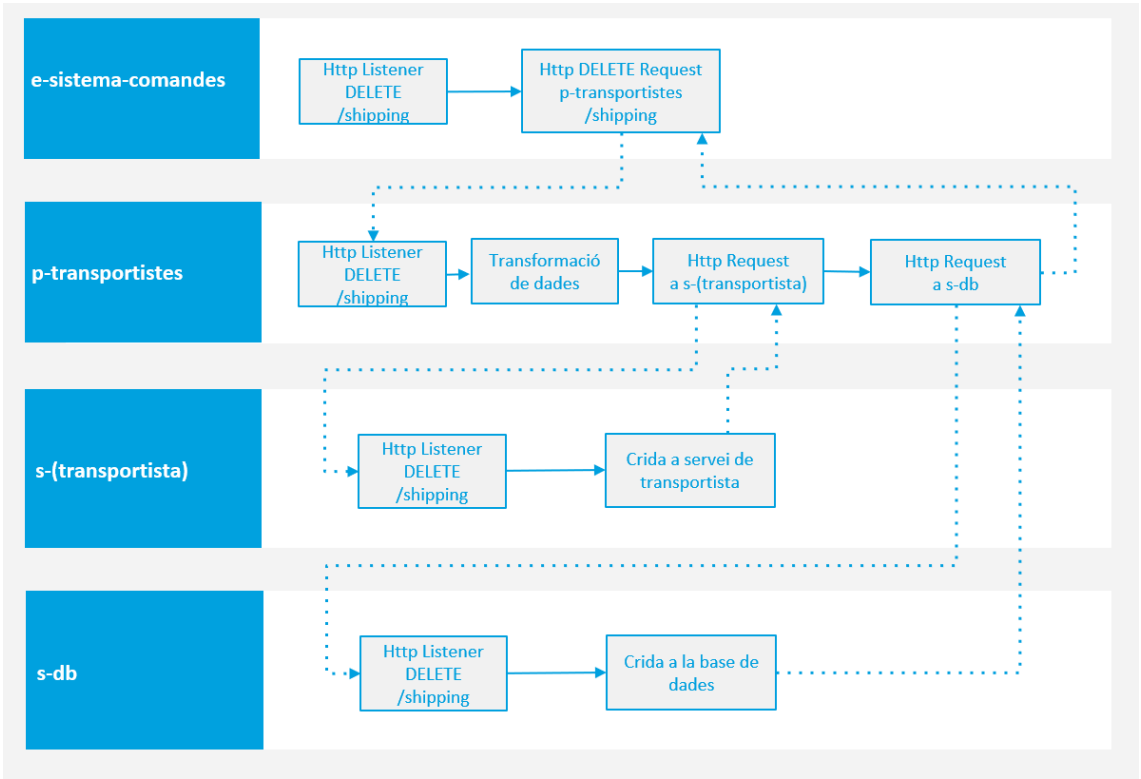

Figura 4.3: Diagrama de fluxos per a eliminar de comandes mitjançant una crida a e-sistema-comandes

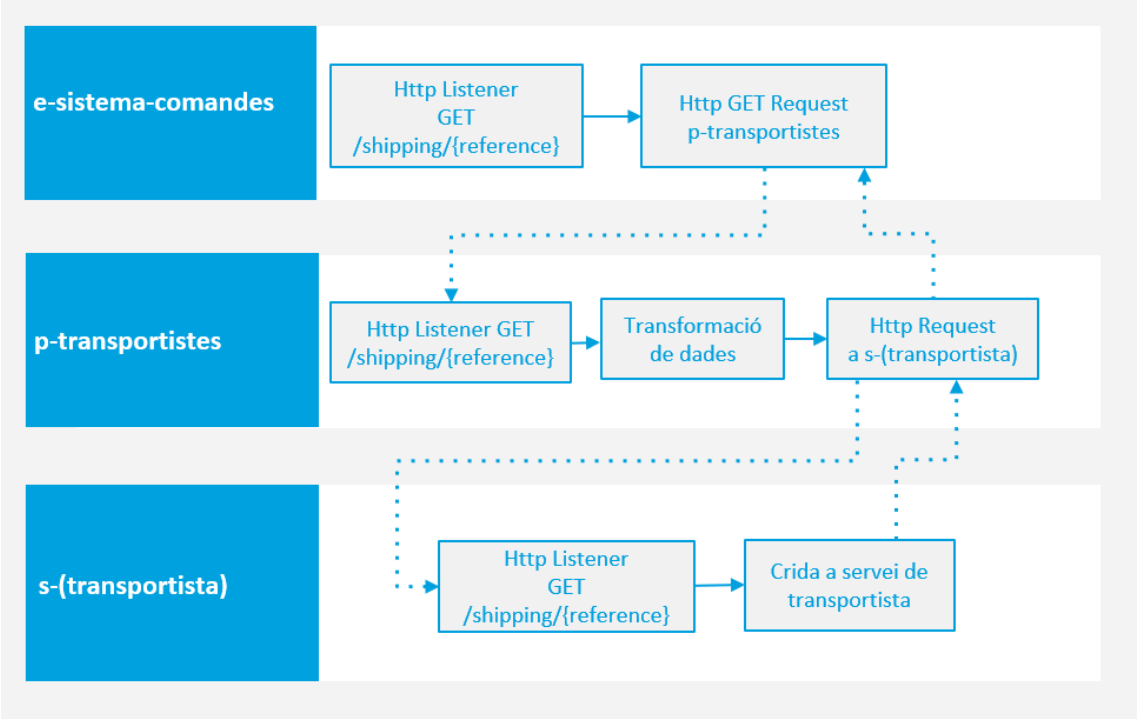

Figura 4.4: Diagrama de fluxos per a la consulta de comandes mitjançant una crida a e-sistema-comandes

Cal aclarir que en tant en el cas de la creació i eliminació de comandes (figures 4.2 i 4.3) la crida de l'aplicació *p-transportistes* a l'aplicació de sistema del transportista en qüestió (*s-(transportista)*) podria efectuar-se més d'una vegada depenent de si el transportista és compatible amb la gestió de diverses comandes en una sola crida.

A més de poder crear, consultar i eliminar comandes, el transportista *envialia* ofereix la possibilitat de consultar les comandes que tenen per destinació la mateixa empresa. Aquest servei també s'oferirà en la capa d'experiència, però caldrà especificar que només per a envialia. De la mateixa manera que amb els altres casos d'ús, en la figura 4.5 es pot observar el comportament de la solució davant una crida a e-sistema-comandes.

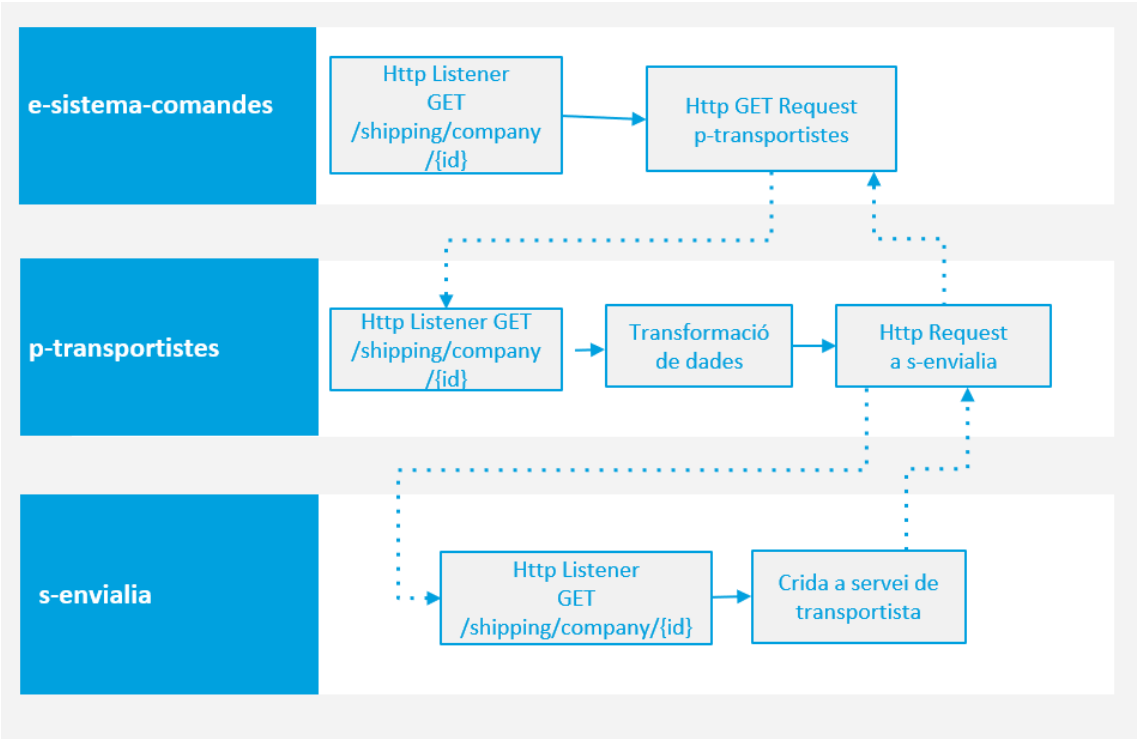

Figura 4.5: Diagrama de fluxos per a la consulta de comandes mitjançant una crida a e-sistema-comandes

4.3 Disseny d'APIS

En aquesta secció es mostraran els endpoints dels que diposa cada API i que facilitarà la comunicació amb la resta de APIs o sistemes. Per tal de mostrar aquesta informació trobem la taula 4.1.

API	**Endpoints**
e-sistema-comandes	/shipping /shipping/{reference} /empresas/{empresa}
p-transportistes	/shipping /shipping/{reference} /empresas/{empresa}
s-gls	/shipping /shipping/{reference}
s-envialia	/envios /envios/{referencia} /empresa/{empresa}
s-mrw	/envios /envios/{referencia}
s-victransa	/shipping /shipping/{referencia}

Taula 4.1: Endpoints de les API

4.4 Model canònic de dades

La utilització d'un model canònic és crucial en els projectes d'integració d'aplicacions, ja que permet estandarditzar les dades utilitzades pels diferents sistemes i aplicacions. Un model canònic actua com una representació comuna i acordada de les entitats i les seues relacions, independentment dels sistemes o aplicacions que les usen. Aquest model comú ens permet reduir les transformacions necessàries per a la integració de manera similar a com el model hub-and-spoke pot reduir comunicacions respecte a un model point-to-point. (2.1.2)

La importància del model canònic radica en diverses raons. En primer lloc, simplifica el procés d'integració, ja que elimina la necessitat de realitzar múltiples transformacions de dades entre els formats específics de cada sistema. Amb un model canònic, les dades es poden convertir de forma directa i eficient de l'origen al destí, evitant errors i incoherències. [17]

En segon lloc, el model canònic permet una comunicació més fluïda i efectiva entre els diferents sistemes. Al tenir una estructura de dades compartida i acordada, es facilita l'intercanvi d'informació i es redueixen els errors de comunicació. A més, les actualitzacions i canvis en el model es poden implementar de forma centralitzada, assegurant la consistència en tot el sistema d'integració.

El model canònic també ofereix flexibilitat respecte als canvis en els sistemes i les necessitats de negoci. En cas de canvis en els sistemes o les interfícies, només cal realitzar les adaptacions necessàries al model canònic, sense afectar directament les integracions existents. Aquest fet implica que si en aquest projecte s'incorporara altre sistema que fera ús de les funcionalitats dels transportistes de la mateixa manera que ho fa el sistema de comandes, però amb altre model de dades, només caldria incorporar una transformació de dades: del model del nou sistema al model canònic. En cas que no hi hagués un model canònic caldria implementar tantes transformacions com transportistes hi haja. Això permet una evolució més senzilla i eficient del sistema d'integració al llarg del temps. [8]

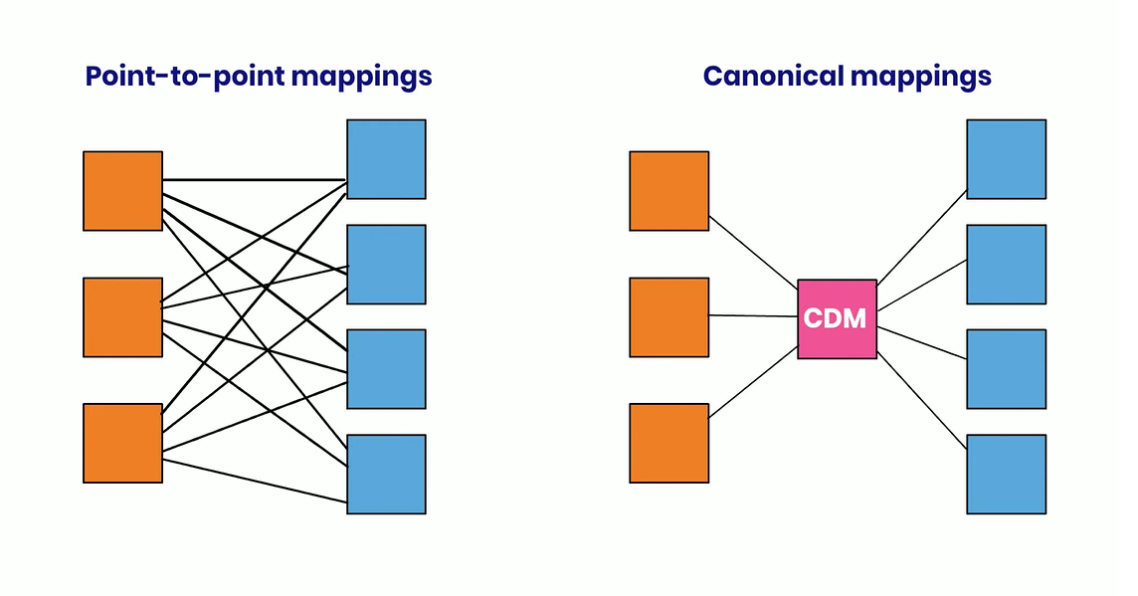

Figura 4.6: Diferència entre integracions sense model canònic de dades i amb aquest.

En la figura 4.6, les figures de color blau i taronja representen els models emprats pels sistemes a integrar. En el cas del nostre projecte aquests sistemes serien el sistema de comandes i els sistemes dels transportistes. Per altra banda, les línies que uneixen aquests sistemes representen transformacions de dades. Per altra banda, la caixa de color magenta amb les sigles *CDM* (de l'anglés *Canonical Data Model*) representen el model canònic de dades. Aquesta representació gràfica permet observar el paper que juga el model canònic de dades i els seus avantatges.

4.5 Transformació de dades

La transformació de dades és un pas clau en la integració d'aplicacions. Per a dur a terme una integració entre dos sistemes, sol ser requerida una transformació. Per tal de tindre clar quin camp d'un model correspon a altre de l'altre model se solen emprar taules de mapeig. Aquestes taules tenen un model de dades per columna i cada fila representa un camp semàntic. En aquest cas s'han fet servir els punts per a representar els nivells de profunditat dels camps. Per exemple, si el camp *envios* inclou altre camp anomenat *fecha*, aquest es representarà com a *envios.fecha*.

Els camps del model canònic han de ser suficients per a mapejar els altres models. És a dir, el model de dades dels sistemes sempre han de poder mapejar-se al complet a partir del model canònic. No obstant això, el contrari no té per què ser cert, alguns camps del model canònic poden quedar sense relacionar mentre es gestione de manera correcta. És per aquesta raó, que el model canònic és en aquest projecte el més extens. Tanmateix, s'ha procurat evitar camps duplicats o amb el mateix significat però amb diferent nom.

Per a les respostes de les funcionalitats de crear i eliminar el model canònic simplement contindrà el camp *referencias* en el que s'indiquen les referències creades o eliminades respectivament i el camp *mensaje* en el que s'inclourà un missatge d'èxit o d'error.

Per altra banda, els mapejos de creació i consulta de comandes són més extensos. Per tal de representar aquests mapejos s'han fet servir les taules anteriorment descrites. Concretament les taules 4.2, 4.3 i 4.4 podem observar com estan relacionats els camps dels models dels sistemes externs a l'empresa i el model canònic, aquest últim situat a la primera columna.

Model canònic	**GLS**	**Envialia**
envios.fecha	fecha	fecha
envios.empresa		empresa
envios.horario	horario	
envios.bultos	bultos	
envios.remitente.nombre	remitente.nombre	rNombre
envios.remitente.direccion1	remitente.direccion	rDireccion1
envios.remitente.poblacion	remitente.poblacion	rCiudad
envios.remitente.provincia	remitente.provincia	
envios.remitente.pais	remitente.pais	rPais
envios.remitente.cp	remitente.cp	rCP
envios.remitente.tlf1	remitente.tlf	rTlf
envios.remitente.tlf2		
envios.remitente.email	remitente.email	
envios.remitente.observaciones	remitente.observaciones	
envios.destinatario.nombre	destinatario.nombre	dNombre
envios.destinatario.direccion1	destinatario.direccion	dDireccion1
envios.destinatario.direccion2		dDireccion2
envios.destinatario.direccion3		dDireccion3
envios.destinatario.poblacion	destinatario.poblacion	dCiudad
envios.destinatario.provincia	destinatario.provincia	
envios.destinatario.pais	destinatario.pais	dPais
envios.destinatario.cp	destinatario.cp	dCP
envios.destinatario.tlf1	destinatario.tlf	dTlf
envios.destinatario.tlf1		
envios.destinatario.email	destinatario.email	
envios.destinatario.observaciones	destinatario.observaciones	
envios.importe	importe	
envios.prioridad		
envios.paquetes.medidas		paquetes.medidas
envios.paquetes.peso	peso	paquetes.peso
envios.paquetes.contenido		paquetes.contenido
envios.productos.nombre		
envios.productos.cantidad		
envios.productos.precio		
envios.productos.peso		

Taula 4.2: Mapeig entre el model canònic de dades i els models de dades de GLS i Envialia

Model Canònic	MRW	Victransa
envios.fecha		fecha
envios.empresa		
envios.horario		horario
envios.bultos		bultos
envios.remitente.nombre	remitente.nombre	remitente.nombre
envios.remitente.direccion1	remitente.direccion1	remitente.direccion1
envios.remitente.direccion2		remitente.direccion2
envios.remitente.direccion3		
envios.remitente.poblacion	remitente.ciudad	
envios.remitente.provincia	remitente.provincia	
envios.remitente.pais	remitente.pais	remitente.pais
envios.remitente.cp	remitente.codigoPostal	
envios.remitente.tlf1	remitente.telefono	remitente.telefono1
envios.remitente.tlf2		remitente.telefono2
envios.remitente.email	remitente.email	remitente.email
envios.remitente.observaciones		
envios.destinatario.nombre		destinatario.nombre
envios.destinatario.direccion1	direccionEnvio.direccion1	destinatario.direccion1
envios.destinatario.direccion2	direccionEnvio.direccion2	destinatario.direccion2
envios.destinatario.direccion3		
envios.destinatario.poblacion	direccionEnvio.ciudad	
envios.destinatario.provincia	direccionEnvio.provincia	
envios.destinatario.pais	direccionEnvio.pais	destinatario.pais
envios.destinatario.cp	direccionEnvio.codigoPostal	
envios.destinatario.tlf1		destinatario.telefono1
envios.destinatario.tlf2		destinatario.telefono2
envios.destinatario.email		destinatario.email
envios.destinatario.observaciones		
envios.importe		
envios.prioridad		
envios.paquetes.medidas	medidas	medidas
envios.paquetes.peso	peso	peso
envios.paquetes.contenido		
envios.productos.nombre	productos.nombre	
envios.productos.cantidad	productos.cantidad	
envios.productos.precio	productos.precio	
envios.productos.peso	productos.peso	

Taula 4.3: Mapeig entre el model canònic de dades i els models de dades de MRW i Victransa

Model Canònic	**Seur**
envios.fecha	fecha
envios.empresa	
envios.horario	
envios.bultos	
envios.remitente.nombre	remitente.nombre
envios.remitente.direccion1	remitente.(calle,numero,escalera)
envios.remitente.direccion2	
envios.remitente.direccion3	
envios.remitente.poblacion	remitente.ciudad
envios.remitente.provincia	
envios.remitente.pais	remitente.pais
envios.remitente.cp	remitente.codigoPostal
envios.remitente.tlf1	remitente.telefono
envios.remitente.tlf2	
envios.remitente.email	remitente.email
envios.remitente.observaciones	remitente.infoDireccion
envios.destinatario.nombre	destinatario.nombre
envios.destinatario.direccion1	destinatario.(calle,numero,escalera)
envios.destinatario.direccion2	
envios.destinatario.direccion3	
envios.destinatario.poblacion	destinatario.ciudad
envios.destinatario.provincia	
envios.destinatario.pais	destinatario.pais
envios.destinatario.cp	destinatario.codigoPostal
envios.destinatario.tlf1	destinatario.telefono
envios.destinatario.tlf2	
envios.destinatario.email	destinatario.email
envios.destinatario.observaciones	destinatario.infoDireccion
envios.importe	
envios.prioridad	prioridad
envios.paquetes.medidas	dimensiones
envios.paquetes.peso	peso
envios.paquetes.contenido	
envios.productos.nombre	producto
envios.productos.cantidad	
envios.productos.precio	
envios.productos.peso	

Taula 4.4: Mapeig entre el model canònic de dades i el model de dades de Seur

CAPÍTOL 5

Desenvolupament de la solució

En aquest capítol, abordarem el procés de desenvolupament de la solució proposada per a la integració d'aplicacions. Començarem explorant les tecnologies seleccionades per implementar la solució proposada, detallant les raons que les fan una bona opció per afrontar els reptes plantejats. A més, presentarem un conjunt de bones pràctiques que facilitaran a estandarizar el desenvolupament, assegurant la qualitat i la flexibilitat del projecte. Aquestes pràctiques ens permetran abordar amb eficàcia el treball en equip, l'evolució dels diferents components i la gestió d'errors de manera estructurada. Finalment, exposarem una visió detallada del procés de desenvolupament en sí, mostrant com s'han aplicat les tecnologies escollides per a dur a terme la posada en marxa del sistema d'integració d'aplicacions.

5.1 Tecnologia emprada

Tot i que existeixen diverses alternatives, per al desenvolupament de la solució s'ha decidit treballar amb les ferramentes de Mulesoft, Dataweave, RAML, Anypoint Studio i CloudHub per diverses raons, però sobretot per la compatibilitat que ofereixen i com es complementen unes a altres. Aquestes plataformes ofereixen una ampla gamma de funcionalitats i eines per a la integració d'aplicacions, facilitant la connectivitat, la gestió i la sincronització entre els diferents sistemes.

5.1.1. Mulesoft

MuleSoft és una plataforma d'integració d'aplicacions i serveis que s'ha guanyat una gran reputació com a líder en el mercat de la integració empresarial. Concretament, ha estat anomenat 7 vegades consecutives en els últims anys líder per Gartner [18]. La seua funció principal és facilitar la connexió i interacció entre diverses aplicacions, sistemes i fonts de dades, tant dins com fora d'una organització. Mitjançant l'ús de MuleSoft, les empreses poden integrar de manera eficient les seues aplicacions i dades, permetent una millor sincronització entre els seus diferents sistemes.

Un tret característic de MuleSoft és la seua arquitectura basada en ESB 2.1.2. Aquesta arquitectura permet crear una infraestructura flexible i modular que connecta les diferents aplicacions i sistemes mitjançant un canal central de comunicació. Aquest canal actua com un conductor que transmet les dades i missatges entre els diferents nodes de la xarxa, facilitant la integració i l'orquestració de les diverses funcionalitats.

Mulesoft està format per molts components i ferramentes, cadascun amb una funció diferent. Els més destacables, i dels que parlaré a continuació, són Anypoint Platform, Anypoint Studio, Dataweave i CloudHub.

5.1.2. Anypoint Platform

Anypoint Platform és la plataforma per a la gestió d'integracions empresarials proporcionada per MuleSoft, la qual ofereix un conjunt complet d'eines per a facilitar la integració d'aplicacions i serveis en entorns complexos. Aquesta plataforma està dissenyada per millorar la comunicació entre sistemes i aplicacions, permetent una interconnexió més àgil i eficient entre diferents components tecnològics.

Entre les diferents ferramentes que Anypoint Platform proporciona, les més destacables per al desenvolupament dut a terme en aquest treball són les següents: [19]

- **Design Center:** Aquesta eina permet als desenvolupadors crear, dissenyar i gestionar APIs (Interfaces de Programació d'Aplicacions) amb facilitat. L'Anypoint Design Center ofereix una interfície gràfica i intuitiva que permet definir els recursos, mètodes i paràmetres de les APIs, així com generar la documentació associada per facilitar el seu ús per part d'altres desenvolupadors.

  Design Center ofereix també un sistema de branques similar a Git [20] que facilita el desenvolupament en equip agilitzant així el procés de noves integracions.

- **Exchange:** És un entorn on els desenvolupadors poden trobar, compartir i reutilitzar connectors, plantilles i altres recursos d'integració. A través de l'Anypoint Exchange, les organitzacions poden fomentar la col·laboració interna i externa, així com aprofitar components predefinits que acceleren el desenvolupament i optimitzen la qualitat de les integracions. És ací on es publiquen les versions de les APIs dissenyades en el Design Center i altres recursos desenvolupats sent visibles per als usaris dins de l'organització o amb permisos.

- **API Manager:** Aquest component de Mulesoft ofereix un control centralitzat per a la gestió d'APIs. Permet definir polítiques de seguretat, controlar els accessos i permisos, monitoritzar el rendiment de les API i protegir-les amb autenticació i autorització avançades. L'Anypoint API Manager és essencial per garantir la seguretat, la visibilitat i el control de les interaccions amb les API dins de l'organització.

- **Runtime Manager:** Aquesta eina ofereix un entorn de gestió per a la implementació, monitorització i administració de les integracions. Amb Anypoint Runtime Manager, els usuaris poden executar i escalar les seues integracions en entorns locals o al núvol (CloudHub), monitoritzant-ne l'estat i la salut, i gestionar-ne les ver-

sions i actualitzacions. És també en el Runtime Manager on trobem la ixida de la consola de les aplicacions, on es mostren errors, logs i informació de desplegament.

A banda d'aquestes ferramentes, trobem altres com API Visualizer, Access Management o API Governance entre altres. Totes elles aporten funcionalitat o donen soport al procés de desenvolupament d'un ESB format per diferents aplicacions. És per això que aquesta plataforma d'adapta perfectament a la solució proposada.

5.1.3. Anypoint Studio

Anypoint Studio és l'entorn de desenvolupament integrat (IDE) proporcionat per MuleSoft. És una eina que permet als desenvolupadors crear, dissenyar i implementar integracions amb facilitat i eficiència. Aquesta eina està estretament relacionada amb MuleSoft i Anypoint Platform, ja que forma part de l'ecosistema integrat d'eines que ofereix la plataforma per a la gestió de les integracions empresarials. [19]

Anypoint Studio està dissenyat per treballar específicament amb la plataforma d'integració d'Anypoint. Amb l'ajuda d'aquesta eina, els desenvolupadors poden crear "fluxos"d'integració mitjançant la definició visual d'una seqüència de passos que manipulen i transformen les dades entre les diferents aplicacions i serveis. Cada un d'aquests passos està dut a terme per un connector. Els fluxos són una concatenació de connectors que duen a terme una funcionalitat concreta, i cada aplicació està formada per un conjunt de fluxos. Aquests fluxos estan basats en la tecnologia Mule, que és la base de MuleSoft i proporciona un motor d'integració lleuger i altament escalable.

Una de les característiques més destacades d'Anypoint Studio és la seua interfície gràfica i de tipus drag-and-drop, que permet als desenvolupadors crear rutes d'integració sense haver de codificar manualment cada pas. Això fa que el procés de desenvolupament siga més senzill, ja que els desenvolupadors poden centrar-se en el disseny de la lògica de negoci sense haver de preocupar-se pels detalls tècnics de baix nivell. Per altra banda, el fet de poder observar l'estructura dels fluxos de manera gràfica permet una millor comprensió del codi a simple vista.

Anypoint Studio està estretament relacionat amb Anypoint Platform. Quan es desenvolupen les integracions amb Anypoint Studio, es poden publicar fàcilment com a APIs a l'Anypoint Exchange (encara que aquesta funció també la permet el Design Center) perquè altres desenvolupadors les puguen trobar i reutilitzar. A més, les API desenvolupades en Anypoint Studio poden funcionar en CloudHub (una plataforma en el núvol on es poden desplegar les API) mitjançant Runtime Manager. De la mateixa manera es pot fer servir la funcionalitat de l'Anypoint API Manager des del mateix IDE, la qual cosa permet als desenvolupadors definir polítiques de seguretat, monitorar el rendiment i gestionar l'accés a les APIs directament des de l'entorn d'Anypoint Studio.

5.1.4. DataWeave

DataWeave és un llenguatge de transformació de dades que ofereix MuleSoft. És una part essencial de la plataforma d'integració d'Anypoint, que inclou Anypoint Studio i Anypoint Platform. DataWeave permet als desenvolupadors transformar i manipular dades entre diferents formats i estructures, així com permet incorporar lògica als fluxos. D'aquesta manera es facilita la integració i comunicació entre les diferents aplicacions i sistemes.

Amb DataWeave, els desenvolupadors poden realitzar transformacions de dades complexes amb una sintaxi simple. Aquest llenguatge ofereix suport per a diversos tipus de dades, com ara XML, JSON, CSV, objectes Java i altres formats d'intercanvi de dades i s'encarrega de oferir transparència al desenvolupador en gran part del procés de transformació de dades. A més, proporciona funcions i operadors que permeten realitzar diverses operacions de manipulació de dades, com ara filtrar, mapejar, agrupar, unir i fer càlculs.

5.1.5. Cloudhub

CloudHub és una plataforma d'integració com a servei (iPaaS) que permet, en conjunt amb Anypoint Platform, connectar, gestionar i orquestrar aplicacions de manera senzilla i eficient. CloudHub està dissenyat per funcionar totalment en el núvol, eliminant la necessitat d'invertir en infraestructures físiques o servidors, ja que tot el processament es realitza en el núvol.

CloudHub presenta una gran flexibilitat, ja que pot adaptar-se a les demandes variables de les empreses sense requerir esforços addicionals d'administració. Els usuaris poden desplegar integracions fiablement sense preocupar-se per la gestió de la infraestructura, ja que això és manejat per MuleSoft. De la mateixa manera, els usuaris poden canviar la quantitat de recursos utilitzats per cada aplicació segons la demanda que aquestes tinguen mitjançant els vCores.

Worker Size	Heap Memory	Storage
0.1 vCores	500 MB	8 GB
0.2 vCores	1 GB	8 GB
1 vCore	1.5 GB	12 GB
2 vCores	3.5 GB	40 GB
4 vCores	7.5 GB	88 GB
8 vCores	15 GB	168 GB
16 vCores	32 GB	328 GB

Figura 5.1: Correspondència de quantitat de vCores amb memòria

Un tret interessant de CloudHub és el desplegament "zero downtime", el qual permet desplegar una nova versió d'una aplicació de manera que fins que la nova versió no estiga accessible la versió anterior si que serà accessible, cosa per la què la funcionalitat de l'aplicació no es veu interrompuda per a altres aplicacions consumidores. Gràcies a aquesta funció els processos d'actualització no afecten al funcionament d'altres aplicacions que requereixen de serveis de l'aplicació a actualitzar.

5.1.6. RAML

RAML (RESTful API Modeling Language) és un llenguatge de modelatge dissenyat per a descriure interfícies d'API RESTful. Aquest llenguatge proporciona una manera senzilla i elegant de documentar, dissenyar i provar API RESTful, facilitant la comunicació clara i eficient entre equips de desenvolupament i usuaris. La seua sintaxi lleugera i intuïtiva permet definir els recursos, mètodes, paràmetres, respostes i altres elements d'una API, aportant coherència al disseny de les integracions.

RAML juga un paper clau en la plataforma d'integració d'Anypoint de MuleSoft. Aquesta relació estreta és el que permet a les empreses obtenir el màxim valor de les seves integracions. Quan es crea un disseny d'API utilitzant RAML, aquest pot ser utilitzat per importar directament les especificacions de l'API a Anypoint Studio i crear una estructura inicial sobre la que implementar la funcionalitat de l'aplicació. Això significa que els desenvolupadors poden començar a treballar amb una base ben definida, sense necessitat de escriure tot el codi des de zero.

Més enllà del desenvolupament, RAML també es relaciona amb la fase de prova i documentació d'API. Amb les especificacions de RAML, les proves d'API poden ser generades automàticament mitjançant una interfície gràfica per assegurar que totes les funcions són correctes i s'ajusten als estàndards. A més, RAML permet crear documentació clara i detallada per a les APIs, la qual cosa resulta essencial perquè altres equips o desenvolupadors externs entenguin fàcilment com interactuar amb els diferents serveis.

5.2 Bones pràctiques

Les bones pràctiques de desenvolupament són essencials en un projecte MuleSoft, tot i que la part de desenvolupament de la tecnologia és de "low code". Aquestes directrius són especialment importants, ja que, en la majoria dels casos, els projectes es duen a terme en equip. Mantenir el codi net i ordenat afavoreix la qualitat del projecte i facilita la comprensió d'aquest, especialment tenint en compte que el resultat del desenvolupament és molt visual. Això permet als desenvolupadors comprendre fàcilment el funcionament dels fluxos i futurs desenvolupadors poden mantenir o modificar el codi amb facilitat, fins i tot si no són els autors originals. Així, desenvolupar amb una visió de futur, sabent que altres persones poden treballar en el codi en qualsevol moment, és fonamental per a una millor col·laboració i manteniment del projecte.

A continuació trobem una llista amb les bones pràctiques més rellevants que he aplicat durant el meu període de pràctiques a l'empresa DISID:

- Renombrar elements amb noms descriptius: Utilitzar noms descriptius pels components del flux facilita la comprensió del funcionament del flux sense necessitat d'entrar en cada component o revisar el codi XML.
- Utilitzar variables descriptives: Emprar noms representatius per a les variables millora la comprensió del codi. Evitar noms genèrics com "var1" o "payload" facilita la lectura i el manteniment del codi.
- Afegir logs informatius, però no excessivament grans: Incorporar logs amb la informació necessària per a revisar incidències. Evitar logs excessivament grans o massa xicotets per a no afectar el rendiment de l'aplicació..
- Afegir loggers a l'inici i final de cada flux: Col·locar loggers a l'inici i final de cada flux permet conéixer quan comença i acaba, i proporcionar informació relacionada amb cada flux.
- Encriptar propietats amb informació sensible: Encriptar propietats que continguin informació confidencial, com contrasenyes o tokens, per a protegir les dades sensibles.
- Separar interfícies i implementacions: Emprar arxius d'interfície per a definir totes les entrades de l'aplicació i tenir arxius separats per a les implementacions d'eixes interfícies, per a millorar l'organització del codi.
- Agrupar totes les configuracions en un arxiu: Centralitzar totes les configuracions en un arxiu global.xml per a facilitar la cerca i actualització de configuracions.
- Tenir arxius de propietats per entorn: Crear arxius .yaml de propietats per a cada entorn utilitzat, assegurant configuracions específiques per a cada context d'implementació.

5.3 Implementació comuna a totes les API

Per dur a terme la implementació de la solució definida en el capítol de disseny 4 s'han seguit una sèrie de passos per al desenvolupament de cada API. En primer lloc, s'ha detallat l'especificació RAML de cada API fent ús del Design Center. Una vegada acabada, aquesta s'ha publicat a Exchange, d'aquesta manera, l'especificació RAML és accessible des de l'Anypoint Studio i a més es poden importar connectors que faciliten la crida a les APIs. Una vegada es crea el projecte en Anypoint Studio, cal estructurar-lo seguint les bones pràctiques. Arribats a aquest punt, s'implementa la funcionalitat mitjançant fluxos. En els següents apartats podem trobar aquests passos descrits amb més detall.

5.3.1. Especificacions RAML en el Design Center

En aquesta fase de la implementació s'han desenvolupat les especificacions RAML seguint les directrius de disseny establertes en el capítol anterior. Aquest procés ha permés crear un "esquelet" sobre el qual començar la implementació de la funcionalitat de cada API més avant.

Un aspecte rellevant d'aquesta etapa és el desenvolupament d'un mòdul comú en totes les API de la solució que reuneix tots els tipus de dades diferents que es troben en el projecte així com un exemple de cada tipus. Aquest mòdul ha estat dissenyat amb l'objectiu de reutilitzar codi i centralitzar tots els tipus de dades utilitzats pel projecte.

```
1  #%RAML 1.0
2  title: s-tfg-gls
3
4  uses:
5    tfg-datatypes: exchange_modules/2bc8556c-3282-48b9-b09c-44758ae90f43/tfg-datatypes/1.0.16/tfg-datatypes.raml
6
7  /shipping:
8    post:
9      body:
10       application/json:
11         type: tfg-datatypes.GLS-CrearEnvio
12     responses:
13       201:
14         body:
15           application/json:
16             type: tfg-datatypes.GLS-RespuestaCrearEnvio
17
18
19   /{reference}:
20     get:
21       responses:
22         200:
23           body:
24             application/json:
25               type: tfg-datatypes.GLS-RespuestaConsultaEnvio
26     delete:
27       responses:
28         200:
29           body:
30             application/json:
31               type: tfg-datatypes.GLS-RespuestaEliminarEnvio
32
```

Figura 5.2: Especificació RAML de l'API s-gls

Com es pot observar en la figura 5.2, en les especificacions RAML es descriuen els *endpoints* que seran accessibles per a una API en concret així com el tipus de la resposta i el cos de la petició (en cas de requerir-ne). Per altra banda, es pot observar també en la línia 5 de la figura com s'importa el mòdul que conté els tipus de dades una vegada aquest ha estat publicat a Exchange.

Una vegada acabades les especificacions de les API, aquestes han sigut publicades a Exchange. Aquest pas permet lligar l'especificació de l'API amb Anypoint Studio de manera que, en cas de publicar una nova versió de l'especificació, els nous canvis es poden aplicar sobre el projecte d'Anypoint Studio. A més, per tal de facilitar la crida a altres API de capes inferiors, cada API pot importar connectors generats automàticament quan es publiquen les especificacions de les API en qüestió a Exchange.

5.3.2. Creació i estructuració de projectes a Anypoint Studio

Per a cada API del projecte, s'ha generat un projecte específic a Anypoint Studio, seguint les bones pràctiques comentades en la secció 5.2. En el desenvolupament de cada API, s'ha donat una especial atenció a l'assignació de noms als recursos i fluxos de cada projecte. S'han utilitzat noms descriptius i significatius que reflecteixen la funcionalitat de cada element facilitant la comprensió i el manteniment del codi. De la mateixa manera, s'han fet servir variables amb noms descriptius per a les dades que contenen.

Per altra banda, s'ha implementat una estratègia de logging coherent que proporciona informació de començament i final de cada flux. S'han afegit logs amb una quantitat d'informació adequada tant al principi com al final de cada flux que permeten saber quin flux s'ha executat junt amb algunes dades importants. A més a més, s'han inclòs logs en els casos d'error mostrant informació representativa sobre l'error a la consola.

Quan es crea un projecte a Anypoint Studio a partir d'una especificació d'API d'Exchange, l'eina genera automàticament un arxiu amb un listener que té la funció d'enrutar cada petició rebuda al flux corresponent de l'endpoint especificat. Aquest listener actua com a punt d'entrada de l'API, captura les sol·licituds i les redirigeix cap als fluxos (també generats automàticament a l'arxiu) que s'encarreguen de processar-les. Tot i que la funcionalitat podria incloure's directament a aquests fluxos, convé crear-ne de nous que aporten funcionalitat i referenciar-los.

Pel que fa a la concentració de configuracions en un sol arxiu, s'ha optat per utilitzar un arxiu "global.xml" on es recullen les configuracions dels diferents connectors per a cada API. Aquest enfocament centralitza les configuracions, permetent una millor gestió i manteniment dels paràmetres de cada integració.

Finalment, s'han creat arxius de propietats per a cada entorn (com ara local, test, producció, etc.) amb les configuracions específiques per a cada cas. A més, s'ha definit una propietat que determina quin entorn està fent servir l'aplicació en cada moment així com de quin arxiu s'han d'emprar les propietats. Aquesta pràctica permet gestionar de forma eficaç els canvis d'entorn i assegurar la correcta configuració en cada ambient, evitant errors tipogràfics a l'hora de canviar els valors de les propietats i agilitzant el canvi d'entorn.

Per tal d'il·lustrar millor la estructura final dels projectes, trobem la figura 5.3, on podem observar les modificacions que hem fet sobre un projecte nou a partir d'una especificació d'Exchange.

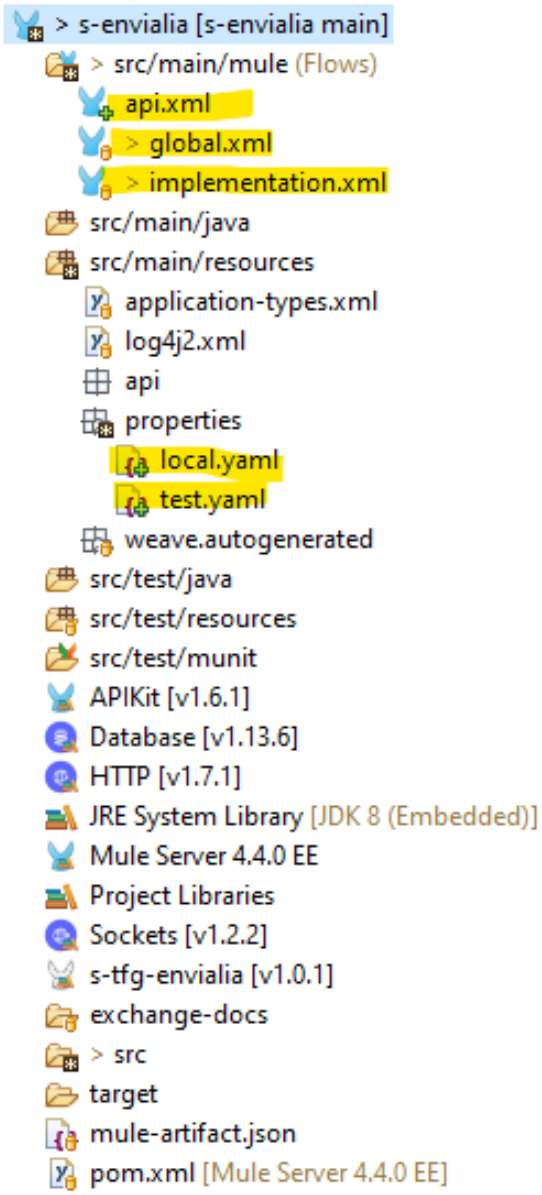

Figura 5.3: Estructura del projecte s-envialia

Per a tots els projectes s'ha seguit la mateixa estructura a excepció de p-transportistes, que a més inclou un fitxer per a cada transportista on podem trobar fluxos necessaris per a la comunicació amb les diferents APIs de la capa de sistema.

5.3.3. Estructura comuna en totes les API

L'arxiu autogenerat a partir de l'especificació RAML inclou un flux per a cada mètode de cada endpoint a més del listener HTTP i el component "Apikit router" que enruta les peticions al flux corresponent.

Aquest arxiu té l'estructura que es pot observar en la figura 5.4, on el primer flux que comença amb el listener passa la petició al connector "APIkit Router", que traspassa la petició al flux corresponent. Cada un dels fluxos dels endpoints comença i acaba amb un logger que dona informació del flux al que es crida i, en els que tenen paràmetres

en l'URL, també es mostra el valor d'aquests. En cas d'haver-hi paràmetres, aquests es guarden en una variable. Per tal d'evitar perdre aquesta informació quan es realitzen altres crides més avant. Cada un d'aquests fluxos crida a un flux d'implementació on trobarem la funcionalitat.

Figura 5.4: APIkit router i http listener autogenerats amb loggers i redirecció de fluxos

Al genererar-se automàticament aquests fluxos a partir de l'especificació RAML, també s'hi genera un gestor d'errors que ens permet elaborar una resposta alternativa en els diferents casos d'error com es pot observar en la figura 5.5.

Figura 5.5: Gestió d'errors de l'API

Com que tant l'estructura dels fluxos és autogenerada i els gestors d'errors es repeteixen per a totes les API, només es mostraran els exemples anteriors per tal de no mostrar informació pràcticament idèntica.

5.4 Desenvolupament de les API per capes

En aquest punt del treball, es mostra breument com ha sigut implementada la funcionalitat en les diferents APIs en cada capa. S'han implementat un total de 5 transportistes, tal com es concreta al principi del capítol de disseny (4), per mantenir una extensió adequada del treball. Per tal d'englobar el màxim nombre d'escenaris possibles s'ha procurat integrar transportistes amb sistemes diferents, incloent-hi una base de dades, un servidor FTP, cues de missatgeria, una altra API REST i un web service.

5.4.1. Capa de sistema

Es mostrarà la implementació de les API de la capa de sistema. Concretament es mostraran les API s-gls, s-envialia, s-mrw, s-seur i s-victransa, que integren un web service, una base de dades, una API Rest, un servidor FTP i cues de missatgeria respectivament. A més, es mostrarà també la implementació de s-db que permet l'accés a través d'una API REST a la base de dades interna de l'empresa.

Funcionalitat de s-gls

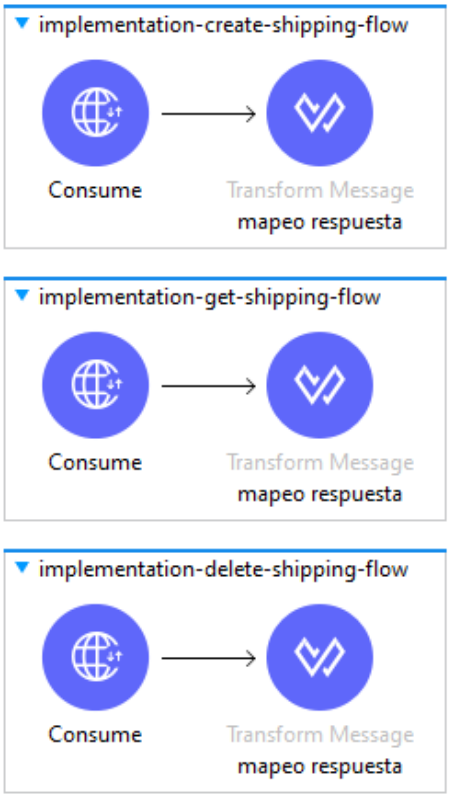

Figura 5.6: Implementació de s-gls

Aquesta API de la capa de sistema es connecta de manera síncrona a un *web service consume* [21]. El sistema de GLS processa la petició de crear, consultar o eliminar una comanda en cada cas i envia una resposta. Com que el model de dades que rep aquesta API és el model que empra el sistema de GLS, la petició al web service es fa a través del primer connector de cada un dels fluxos, tant per a la creació, la consulta i per eliminar comandes.

El segon connector de cada flux té la funció de transformar la informació de la qual disposa en el model de dades de la resposta corresponent de GLS.

Funcionalitat de s-envialia

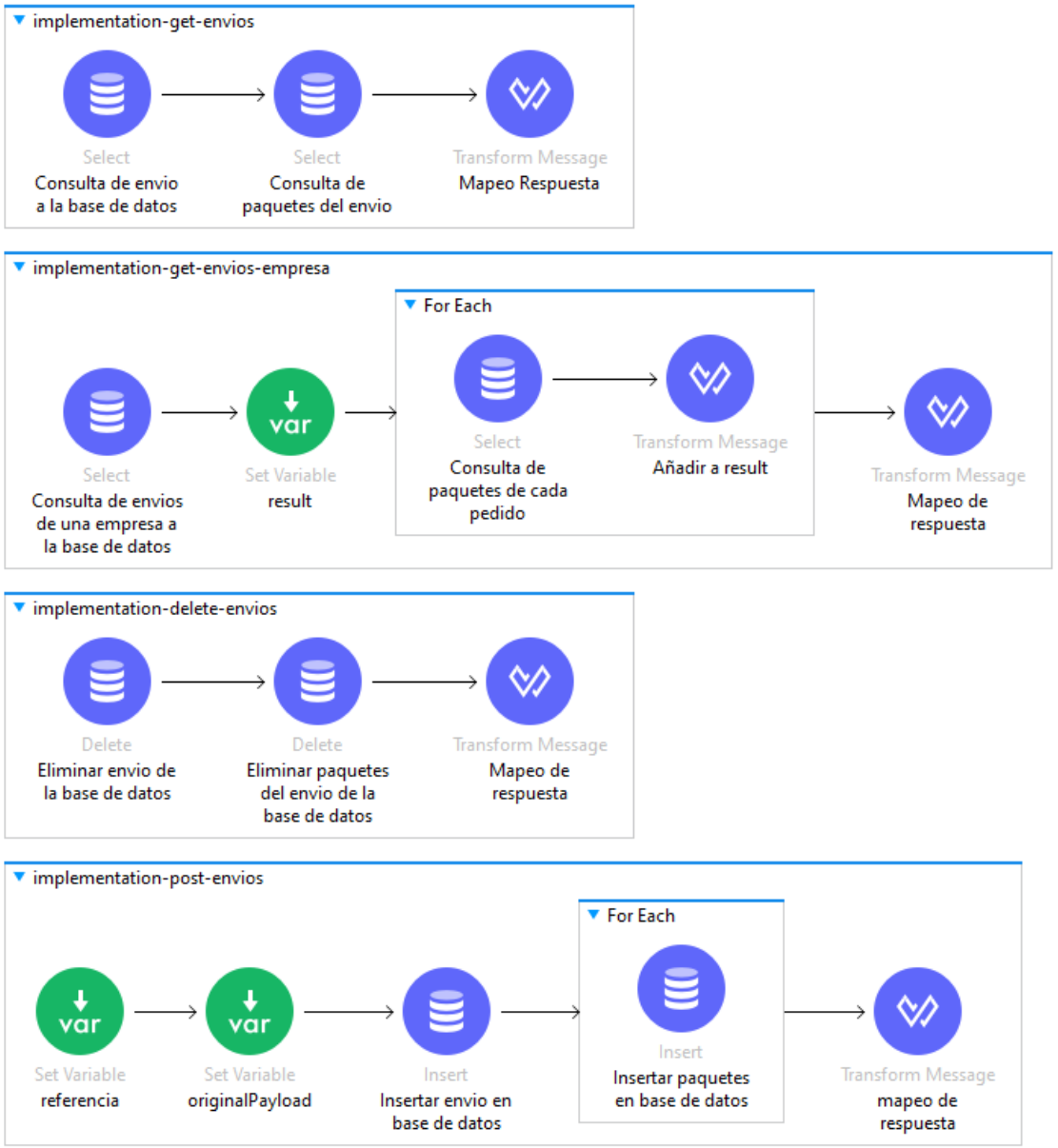

Figura 5.7: Implementació de s-envialia

Aquesta API de sistema té un flux més ja que, a part de crear, consultar i eliminar comandes, ofereix la funcionalitat adicional de consultar les comandes d'una empresa en concret. L'API s-envialia es connecta amb la base de dades d'envialia. Concretament, es fan servir les taules "envíos" i "paquetes", que contenen informació de les comandes i els paquets de les comandes respectivament. En el següent llistat trobem la implementació visible a la figura 5.7 detallada.

- **implementation-get-envios:** Aquest és el flux que ens permet la consulta de comandes i està compost per 3 connectors amb funcions diferents. El primer connector consulta les dades de la comanda de la base de dades a excepció de la informació

respectiva als paquets i guarda la informació en una variable. Per altra banda, el segon connector consulta la informació dels paquets que pertanyen a la comanda que es vol consultar. El resultat d'aquesta operació es guarda en una altra variable.

Finalment, el connector final elabora la resposta amb la informació disponible a les dos variables creades pels connectors anteriors. Per a realitzar aquesta funció es fa servir un script de Dataweave 2.0 que insereix la informació dels paquets de la comanda (la segona variable) en la informació general de la comanda (la primera variable).

- **implementation-get-envios-empresa:** Aquest flux ens permet consultar les comandes realitzades per una empresa en concret. El primer connector ens permet fer una consulta a la base de dades que retorna informació sobre totes les comandes de l'empresa en qüestió. Aquest connector ens proporciona un *array* amb una entrada per cada comanda de l'empresa. El segon connector crea una variable amb nom "result" que conté un *array* buit.

  A continuació hi trobem un bucle que, per a cada comanda de l'empresa, consulta la informació dels paquets que conté amb ajuda del primer connector del bucle "for each". El segon connector del bucle té com a funció combinar la informació de la comanda amb la dels seus respectius paquets i afegir el resultat a la variable "result". Al final d'aquest bucle, en la variable result podem trobar totes les comandes de l'empresa incloent-hi la informació dels paquets.

  En últim lloc, trobem un connector del tipus "transform message" que ens permet elaborar la resposta seguint el model de dades concretat a l'especificació RAML.

- **implementation-delete-envios:** Aquest flux té com a finalitat eliminar una comanda de la base de dades d'Envialia. Per tal de realitzar aquesta funció, requereix els tres connectors que conté. En primer lloc, trobem una petició a la base de dades que elimina la informació de la comanda en qüestió de la base de dades. Per altra banda, el segon connector elimina la informació relativa als paquets que pertanyien a aquesta comanda.

  Finalment, trobem el connector que elabora la resposta mostrant un missatge d'èxit o error i la referència de la comanda eliminada en cas d'exit.

- **implementation-post-envios:** Aquest flux és el que ens permet crear noves comandes al sistema d'Envialia. En primer lloc, trobem un connector que crea la variable referència, la qual no es repetirà en diferents iteracions d'aquest flux. El següent connector ens permet guardar la informació que disposem a la payload a una variable anomenada "originalPayload". Acò es deu al fet que si l'eixida dels connectors de la base de dades sobreescriuen el valor del "payload" si no es dirigeix l'eixida a una variable.

  El tercer connector del flux té com a funció inserir la informació general de la comanda a la base de dades. Una vegada realitzada aquesta inserció, trobem un bucle que recorre els paquets de la payload original disponible a la variable "originalPayload" i que insereix cada un dels paquets a la base de dades.

Finalment, de la mateixa manera que en els fluxos anteriors trobem un connector que elabora una resposta segons el model del transportista. En aquest cas, mostra un missatge d'èxit o error i la referència de la comanda creada en cas d'èxit.

Funcionalitat de s-mrw

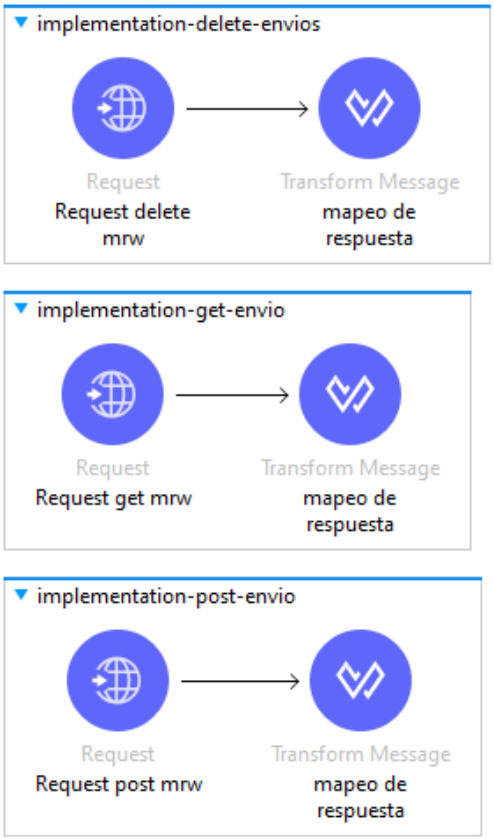

Figura 5.8: Implementació de s-mrw

Aquesta API de la capa de sistema s'integra amb el sistema de MRW, al qual accedeix mitjançant una API REST. S-mrw consta de tres funcions, les quals ens permeten eliminar, consultar i crear comandes en el sistema de MRW.

En la figura 5.8 podem observar com estan implementats els diferents fluxos. Cada flux té dos connectors. El primer realitza una petició a l'API REST del sistema a integrar de MRW. Aquesta petició no necessita cap transformació de dades anterior perquè l'api rep les dades en el format sol·licitat per MRW. El segon connector s'encarrega d'elaborar la resposta d'acord amb l'especificació RAML.

Funcionalitat de s-seur

Figura 5.9: Implementació de s-seur

L'API s-seur integra el sistema del transportista SEUR. Concretament, aquest transportista empra un servidor FTP i es fa servir de la transferència d'arxius per a gestionar les comandes. Com que el model de dades que rep s-seur és el requerit pel sistema de SEUR, no cal realitzar cap transformació de dades prèvia. La configuració del connector que es connecta al servidor FTP la podríem trobar a l'arxiu global.xml i és comuna per als connectors dels diferents fluxos, ja que es tracta del mateix servidor FTP.

Aquests fluxos engloben els seus connectors en un connector "try", la qual cosa ens permet gestionar els errors en cas de haver-ne. S'ha pres aquesta decisió pel fet que el servidor FTP no torna un missatge en cas de no poder fer l'operació, sinó que torna un error. És per això que en cas d'haver-hi error, el connector amb nom "Respuesta de error" elaborarà una resposta d'error seguint el model de dades de la resposta declarat a l'especificació RAML.

En cas que la connexió amb el servidor FTP tinga èxit, el connector següent a l'FTP elaborarà una resposta segons el model de dades especificat al RAML.

Funcionalitat de s-victransa

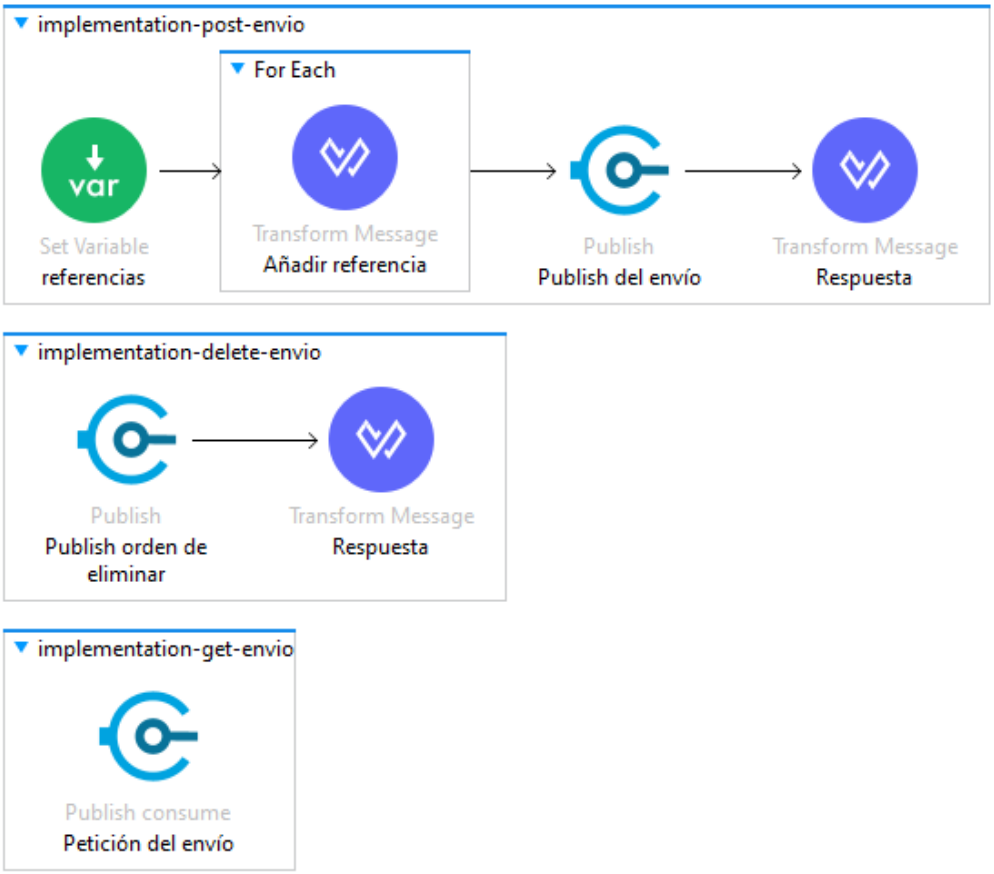

Figura 5.10: Implementació de s-victransa

L'API s-victransa integra un sistema que es comunica mitjançant cues. S-victransa ens ofereix la possibilitat de crear una sèrie de comandes amb una única petició, eliminar una comanda i consultar una comanda. En el següent llistat trobem una explicació detallada de la implementació visible a la figura 5.10:

- **implementation-post-envio:** Aquest flux té com a finalitat crear una sèrie de comandes al sistema de Victransa. El primer connector que trobem al flux crea la variable *referencias*, la qual contindrà un *array* buit. El següent bucle és qui afegirà valors a la variable *referencias* creant una variable única per a cada comanda. El tercer connector d'aquest flux publicarà les comandes junt amb les referències a una cua del sistema de Victransa de manera asíncrona. Finalment, l'últim connector elaborarà la resposta incloent-hi un missatge i les referències de les comandes creades.
- **implementation-delete-envio:** Aquest flux ofereix la possibilitat d'eliminar una comanda del sistema de Victransa. Per a aconseguir aquest objectiu, el primer con-

nector publica una ordre d'eliminar en una cua del sistema de Victransa de manera asíncrona i aquesta s'encarrega d'eliminar la comanda. El segon connector elabora una resposta de manera que l'API s-victransa retorne el model de dades especificat en el RAML.

- **implementation-get-envio:** Finalment, trobem el flux que ens permet consultar una comanda del sistema de Victransa. El connector que trobem al flux publica un missatge en una cua del sistema de Victransa amb la referència de la comanda que es vol consultar i espera una resposta. Aquesta és l'única operació síncrona de s-victransa. Per altra banda, com que el model de dades de la resposta coincideix amb l'especificació RAML no cal afegir cap transformació a les dades.

Funcionalitat de s-db

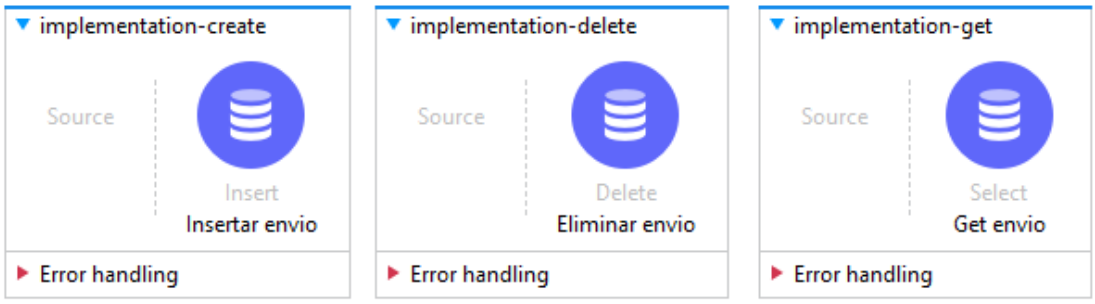

Figura 5.11: Implementació de s-db

Aquesta API proporciona la possibilitat de crear, eliminar i consultar una comanda de la base de dades de la propia empresa. Els fluxos d'implementació que presenta són simples ja que no requereix de cap transformació de dades. Simplement amb la query SQL de cada connector es pot realitzar el mapeig de camps per tal d'inserir, consultar o eliminar una comanda.

5.4.2. Capa de procés

Aquesta capa està formada íntegrament per l'API p-transportistes. Concretament, aquesta capa s'encarrega de fer la crida a l'API de sistema del transportista indicat en la petició que rep en p-transportistes i, per altra banda, mantindre una sincronització de les comandes en la base de dades interna de l'empresa. P-transportistes disposa d'uns fluxos inicials que redirigeixen el fil d'execució al flux corresponent del transportista indicat en la petició.

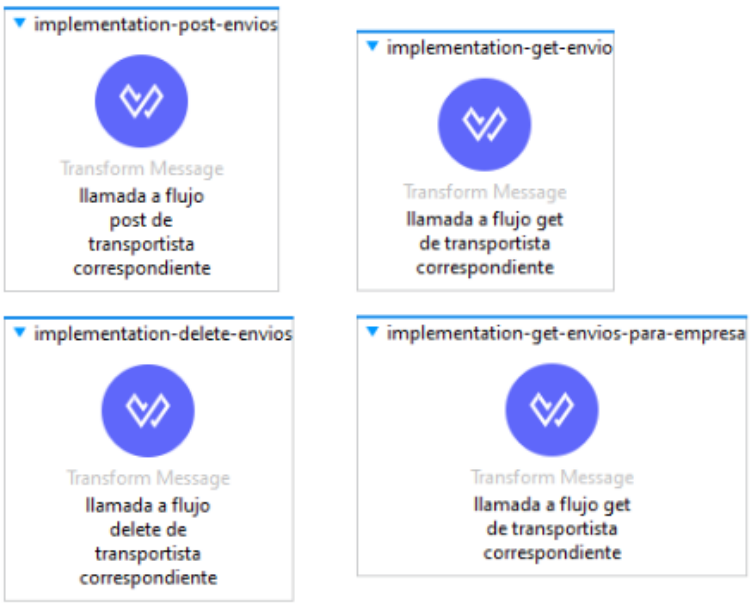

Figura 5.12: Fluxos que redirigeixen el fil d'execució al flux del transportista corresponent

Els fluxos de la figura 5.12 fan servir un script de DataWeave 2.0 per tal de fer la crida al flux "transportista"-"mètode"-flow, on "transportista" és el paràmetre que requereix la petició a p-transportistes segons està definit a l'especificació RAML i "mètode" és el mètode en què es crida a l'API, és a dir, post, get o delete.

Per altra banda, el connector del flux implementation-get-envios-para-empresa fa una crida a "transportista"-get-envios-para-empresa-flow. De la mateixa manera que abans, "transportista" cobra el valor del paràmetre que inclou la petició que en aquest cas només pot ser "envialia".

Creació de comandes

Quant a la implementació dels fluxos que gestionen la creació de comandes hi trobem dos tipus: els fluxos de transportistes que donen suport a la creació de només una comanda per crida i els que poden gestionar diverses comandes per crida.

Tenint en compte que el model canònic està dissenyat per tal que es puguen gestionar diverses comandes per crida, aquests fluxos deuen gestionar diverses comandes als transportistes que només en gestionen una comanda per crida.

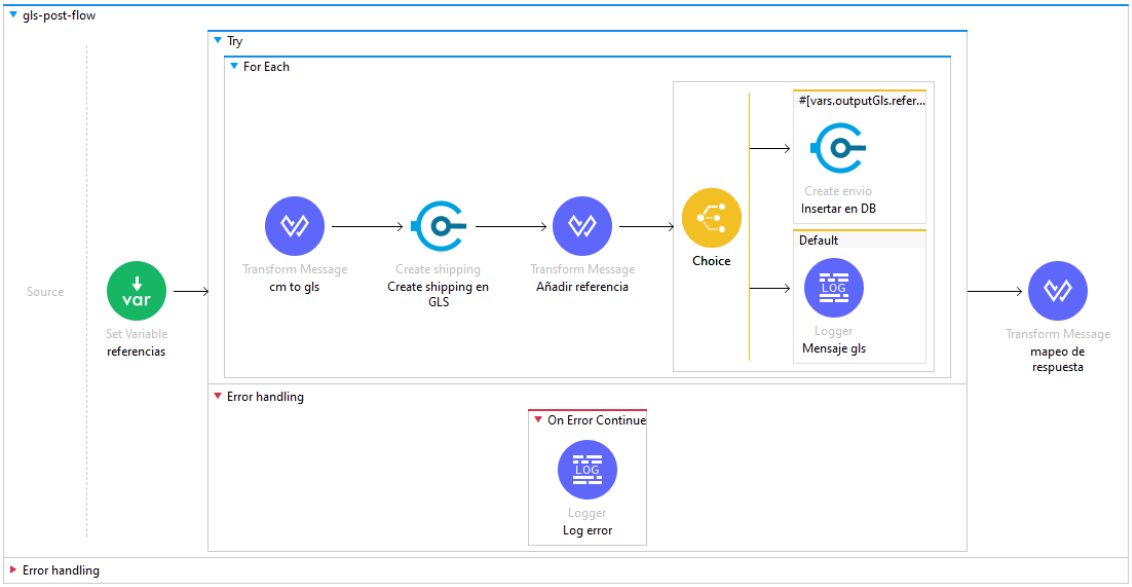

Figura 5.13: Flux de transportista que només gestiona una comanda per crida

El flux que podem observar a la figura 5.13 mostra com p-transportistes orquestra la creació de comandes al transportista GLS, no obstant això, els fluxos per als transportistes Envialia, MRW i Seur presenten la mateixa estructra, ja que tots aquests transportistes gestionen només una comanda per crida.

El primer connector d'aquest flux crea la variable *referencias* que incorporarà els valors de les referències de les comandes que es creen correctament. A continuació trobem un espai "try" que inclou un bucle. Cada iteració del bucle transforma del model canònic al model de GLS una comanda de la petició i l'envia a l'API del transportista. Segons si la resposta indica èxit o no, s'afegeix la referència a la variable *referencias* o no respectivament.

Després, també depenent de si la creació de la comanda ha tingut èxit o no en l'API del transportista, es farà una crida a s-db per tal d'inserir la comanda a la base de dades interna de l'empresa o es mostrarà per consola un missatge d'error indicant per què no ha sigut possible la creació de la comanda. En cas d'ocórrer cap error, es mostrarà l'error per consola.

Finalment, trobem un connector que elaborarà la resposta seguint el model de dades especificat al RAML.

Per altra banda, trobem el transportista Victransa que sí que dona suport a la gestió de diverses comandes per crida. Podem observar la implementació del flux que permet la creació de comandes en Victransa en la figura 5.14

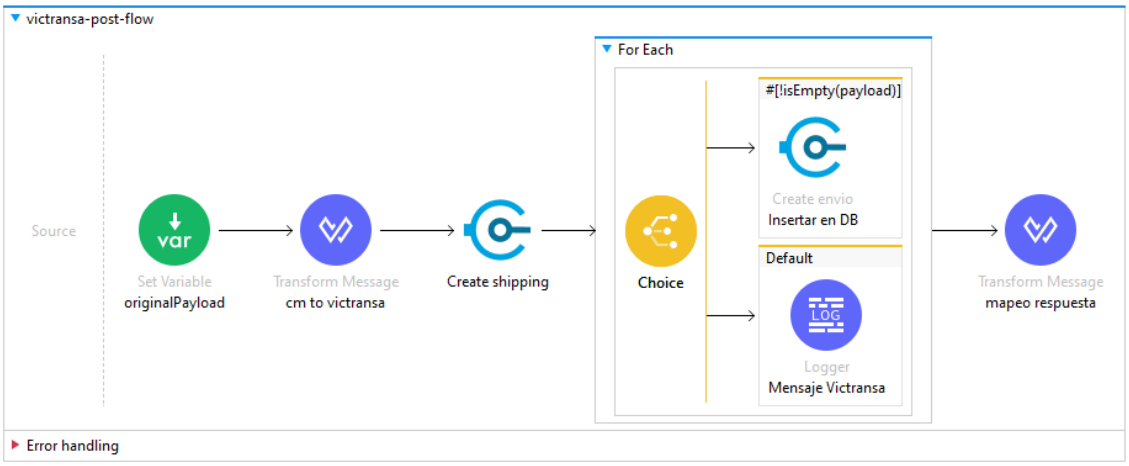

Figura 5.14: Flux de transportista que gestiona diverses comandes per crida

En primer lloc, aquest flux crea una variable *originalPayload* que ens servirà per a evitar sobreescriure les dades que rep el flux i perdre-les. Després, es transformen les dades d'entrada al model de l'API s-victransa i es realitza la petició a s-victransa amb el connector "Create shipping". El bucle que trobem a continuació recorre l'*array* de referències retornat per s-victransa i depenent de si el valor és una String buida ("") o no, crea la comanda també en la base de dades interna de l'empresa o mostra un missatge per la consola d'error. Per últim, s'elabora una resposta amb el model de dades especificat al RAML.

Eliminació de comandes

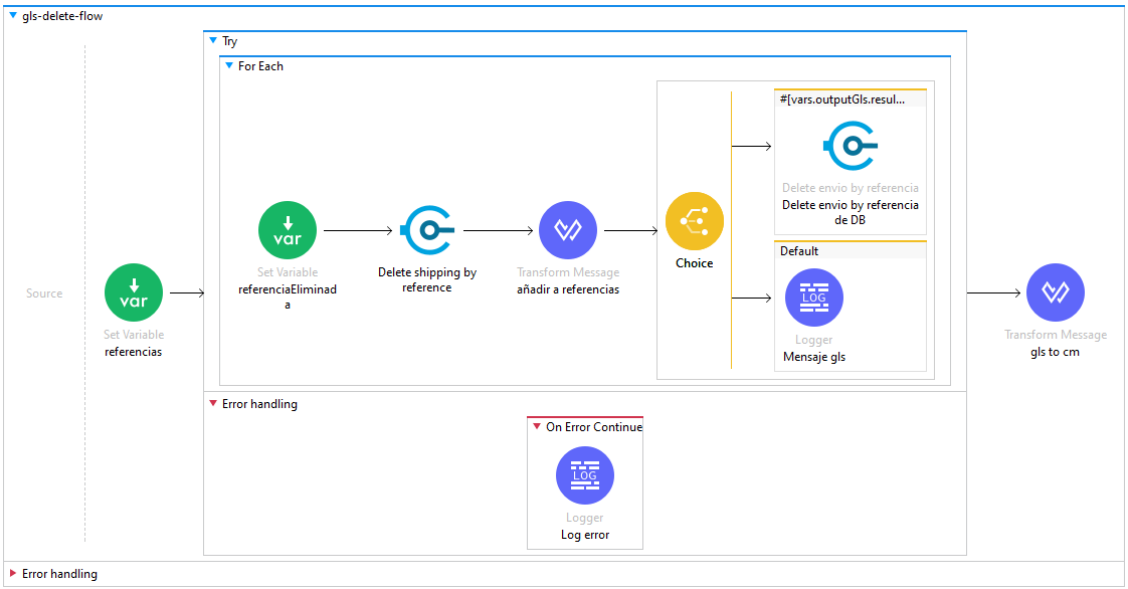

Figura 5.15: Flux de transportista que només gestiona una comanda per crida

El flux que podem observar en la figura 5.15 ens mostra el procés que realitza p-transportistes per tal d'eliminar diverses comandes en el transportista GLS, no obstant

això, els fluxos per als transportistes Envialia, Seur i Victransa també presenten aquesta estructura.

En primer lloc, es crea la variable *referencias*. De manera similar al flux de la figura 5.13, aquesta variable inclourà les referències de les comandes que s'eliminen correctament. A continuació tenim un bucle. En cada iteració del bucle es guarda la referència a eliminar en la variable *referenciaEliminada*. Aplegats a aquest punt, es fa la petició per a eliminar la comanda a l'API de sistema del transportista corresponent i s'afegeix a la variable *referencias*, en cas d'èxit, la referència de la comanda eliminada. En cas contrari s'afegeix una String buida.

Després segons la resposta de l'API de la capa de sistema s'elimina de la base de dades interna de l'empresa o es mostra un missatge explicant per què no s'ha pogut eliminar la comanda. Finalment, tenim un connector que elabora la resposta.

D'altra banda, trobem el transportista MRW que sí que dona suport a la gestió de diverses comandes per crida. Podem observar la implementació del flux que permet la creació de comandes en Victransa en la figura 5.16.

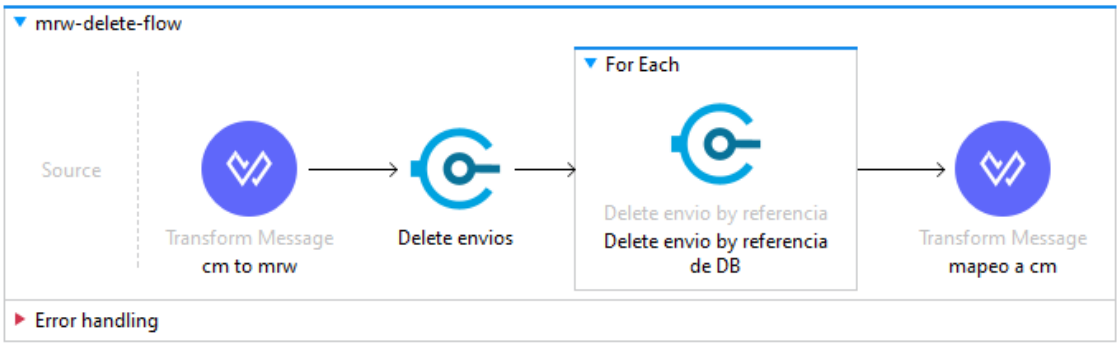

Figura 5.16: Flux de transportista que gestiona diverses comandes per crida

Aquest flux comença transformant la informació d'entrada al model de dades de s-mrw. Una vegada fet açò, es fa la petició a s-mrw, que retornarà un *array* amb les referències de les comandes que s'hagen eliminat correctament. En el bucle següent s'eliminaran de la base de dades interna de l'empresa les comandes corresponents a les referències de la resposta de s-mrw. Finalment, s'elaborarà una resposta mitjançant l'últim connector.

Consulta de comandes

Per tal de consultar les comandes tots els fluxos per als diferents transportistes presenten la mateixa estructura. En primer lloc, una crida a l'API del transportista corresponent que inclou la referència de la comanda a consultar i una transformació de dades per tal d'adaptar la informació retornada per la capa de sistema al model de dades canònic.

A més dels fluxos de consulta d'una comanda donada una referència, el transportista Envialia inclou la funcionalitat de consultar les comandes associades a una empresa en concret. Aquest flux comparteix la mateixa estructura que els altres fluxos de consulta,

amb la diferència que en compte de retornar una única comanda retorna un *array* de comandes.

En la següent figura podem observar, en la part superior, el flux que consulta una comanda i, en la part inferior, el flux que consulta les comandes d'una empresa en concret.

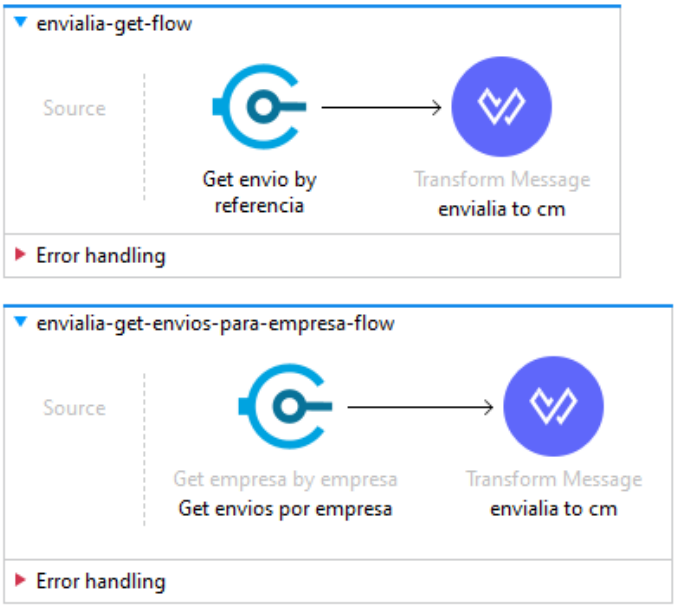

Figura 5.17: Fluxos de consulta de comandes de la capa de procés del transportista Envialia

5.4.3. Capa d'experiència

Aquesta capa és la que exposarà tota la funcionalitat del projecte d'integració al sistema de comandes de l'empresa. L'especificació del RAML és molt similar a la de p-transportistes, ja que presenta els mateixos endpoints i, a més, els mateixos tipus de dades. És per aquesta raó que la implementació dels fluxos que aporten funcionalitat als endpoints són molt senzills, simplement consten d'una crida a l'API de la capa de procés, és a dir, a p-transportistes.

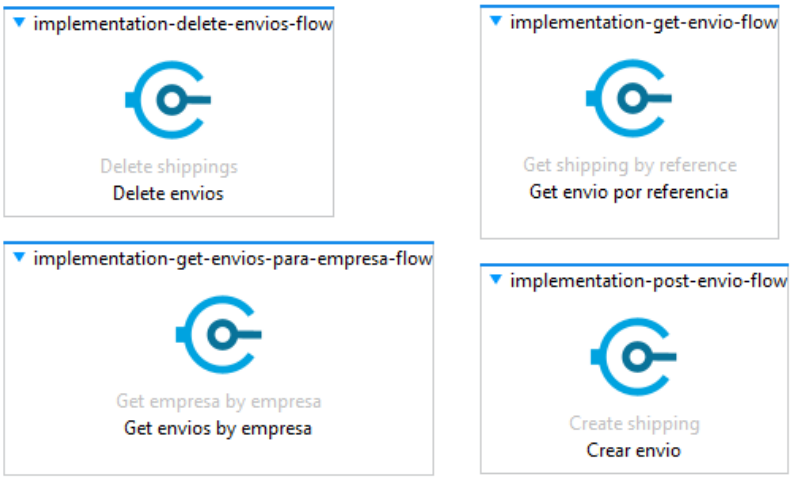

Figura 5.18: Implementació d'e-sistema-comandes

CAPÍTOL 6

Implantació

Una vegada la solució ja està completament implementada, ens trobem amb el repte d'implantar la nova solució on abans trobàvem el projecte d'integració sobre el qual partíem. Cal destacar que aquest escenari és fictici, ja que aquest projecte és extern a l'empresa i no cap la possibilitat d'accedir als components de l'empresa. A més, s'ha de tenir en compte que les diferents APIs que integren la solució es poden desplegar en àmbit local. No obstant això, en aquest capítol s'explica com seria el procés de migració d'una solució a altra.

6.1 Entorns

Els entorns de desplegament són espais separats on les aplicacions poden ser provades i executades en diferents etapes del cicle de vida del desenvolupament. Aquests entorns permeten als desenvolupadors provar les seues aplicacions en diferents contextos abans de posar-les en producció [22]. D'aquesta manera, es poden identificar i solucionar problemes abans que arriben als usuaris finals i es pot assegurar que l'aplicació es comporte de manera esperada en diferents circumstàncies. Tot i que existeixen múltiples entorns diferents, per a aquest tipus de projecte destaquen l'entorn de prova o test i l'entorn de producció.

L'entorn de prova és un entorn on els desenvolupadors poden fer proves exhaustives de l'aplicació abans que aquesta siga posada en producció. És ací on es poden realitzar diverses proves, com proves d'integració, proves de rendiment i proves de seguretat, per assegurar-se que l'aplicació funciona correctament i compleix amb els requisits establerts. En aquest entorn, es poden simular diferents escenaris i situacions per identificar possibles problemes i ajustar l'aplicació abans que arribe als usuaris finals. Per tal d'assegurar que aquestes proves són tan fiables com siga possible, cal que els recursos sobre els quals es despleguen les aplicacions siga el més paregut possible amb l'entorn de producció.

L'entorn de producció, d'altra banda, és l'entorn on l'aplicació ja està en funcionament i és utilitzada pels usuaris finals. En aquest entorn, l'objectiu principal és assegurar-se que l'aplicació siga estable, escalable i responga de manera adequada a la càrrega real

dels usuaris. És essencial que l'entorn de producció tinga la màxima disponibilitat i que siguen minimitzades les interrupcions o caigudes del sistema.

Anypoint Platform proporciona suport per als diferents entorns de desplegament, incloent-hi l'entorn de prova i l'entorn de producció. Amb CloudHub, és possible desplegar la mateixa aplicació en ambdós entorns, però amb configuracions diferents. Per exemple, en l'entorn de prova es pot configurar l'aplicació perquè faça referència als serveis de prova proporcionats per l'empresa i els components a integrar, mentre que en l'entorn de producció es pot configurar per fer ús dels serveis finals. Per exemple, hi ha empreses que disposen d'una base de dades per a l'entorn de prova i altra per a l'entorn de producció. Això permet realitzar proves sense afectar la producció i assegurar-se que l'aplicació funciona adequadament abans de ser llançada als usuaris finals.

Cal anotar que a causa de les limitacions de la versió de prova gratuïta del compte amb què s'ha desenvolupat la solució, no ha sigut possible desplegar la solució en ambdós entorns de manera completa. Malgrat això, aquest enfocament hauria de ser utilitzat en situacions on es disposa d'accés als diferents entorns i es pot gestionar la configuració d'acord amb les necessitats del projecte.

6.2 Desplegar entorns

Per poder desplegar l'aplicació als entorns de prova o de producció, es requereix un procés que combina la configuració i la implementació en l'Anypoint Studio juntament amb l'ús de CloudHub.

En primer lloc, s'ha de configurar l'aplicació a l'Anypoint Studio de manera que les propietats estiguen adaptades als entorns de prova o de producció, depenent en quin entorn volem desplegar l'apliació. Això implica ajustar els paràmetres que determinen com l'aplicació es connectarà amb altres sistemes i recursos. Normalment, aquest procés es duu a terme a través d'arxius de propietats, de manera que fent referència a un arxiu o a un altre es pot canviar d'entorn ràpidament.

Després de configurar les propietats, es procedeix a desplegar l'aplicació a CloudHub. Aquí és on es decideix en quin entorn s'ha de fer el desplegament, siga prova o producció. Aquesta elecció és crucial, ja que afectarà directament l'experiència de l'usuari final. En aquest punt, es pujarà un arxiu .jar que conté tot el codi al núvol, permetent que l'aplicació siga executada en aquest entorn remot.

Aquest procés de desplegament assegura que l'aplicació estiga configurada de manera adequada i que funcione correctament en l'entorn desitjat. La combinació d'Anypoint Studio i CloudHub permet als desenvolupadors gestionar i controlar de manera eficient la implementació i l'execució de les seues aplicacions en entorns diferents, adaptant-les a les necessitats de prova o producció.

CAPÍTOL 7

Proves

Després de desplegar la solució en l'entorn de prova, cal dur a terme una sèrie de proves per tal d'assegurar el correcte funcionament. Es posen a prova totes les funcions de l'aplicació. Es tracta de verificar que tot funcione bé i de manera precisa, assegurant-se que l'aplicació és capaç de tractar les seues tasques amb èxit. Aquesta etapa de proves en l'entorn de test és com un pas previ a l'exposició de la solució a l'usuari final on es corregeixen els problemes abans que puguen arribar als usuaris. Una vegada que tot està provat i validat en aquest entorn de proves, només queda el pas final: dur-ho a l'entorn de producció. En aquest capítol s'explicarà quina sèrie de proves s'han dut a terme per tal de comprovar que la solució implementada funciona.

7.1 Proves locals

Durant la fase de desenvolupament, s'han realitzat proves locals per a cada component de la solució. Aquestes proves s'han dut a terme en un entorn local utilitzant Anypoint Studio. Aquest enfocament ha permés desplegar diverses APIs de la solució en una mateixa màquina per a verificar el funcionament correcte de cada component de manera individual abans de la seua integració en la solució global.

Per a aquestes proves locals, s'ha aprofitat la funcionalitat de consola de APIs proporcionada per Anypoint Studio, una interfície que ens permet fer consultes a l'API desplegada i, a més, ens ofereix documentació sobre aquesta generada automàticament a través de l'especificació RAML. Mitjançant aquesta eina, s'han pogut simular i executar peticions d'API de manera interactiva i observar les respostes en temps real. Això ha permés verificar la coherència de les dades processades per cada API individual, identificar possibles problemes i assegurar-se que els fluxos específics de cada component funcionen correctament. Aquestes comprovacions s'han pogut dur a terme mitjançant la ferramenta Mule Debbugger, una eina que ens permet executar les peticions connector per connector podent veure els valors de variables i altres elements del fil d'execució.

L'enfocament de proves locals ha estat essencial per garantir que cada part de la solució estigués implementada correctament i complís amb les especificacions de disseny. L'ús de l'entorn local i les eines proporcionades per Anypoint Studio ha permés iden-

tificar i resoldre problemes individualment abans de passar a les proves més àmplies i desplegaments als diferents entorns. Això ha contribuït a assolir una solució coherent en el desenvolupament global del projecte.

7.2 Proves end-to-end

Les proves end-to-end, també conegudes com a proves E2E, representen un pas crucial en la validació i assegurament de la qualitat d'una aplicació o sistema. Aquest tipus de proves es caracteritza per simular i validar el flux de treball complet d'un usuari des del principi fins al final a través de tota l'aplicació. En altres paraules, aquestes proves abasten totes les parts de l'aplicació i se centren a avaluar com les diferents parts interactuen hi col·laboren quan treballen juntes com un sol sistema.

La raó principal d'executar proves end-to-end és assegurar-se que totes les diverses funcions, mòduls i components d'una aplicació funcionen adequadament quan es posen a treballar conjuntament. Aquest tipus de proves és una simulació completament realista de com els usuaris interactuarien amb l'aplicació en una situació real [23]. Això significa que s'avaluen aspectes com l'entrada i sortida de dades, les comunicacions entre sistemes interns i externs, així com la gestió d'errors i altres escenaris possibles.

Les proves E2E també són útils per identificar possibles problemes que podrien passar per alt en proves més petites o amb una regió del sistema aïllada. Quan els diversos components de l'aplicació estan totalment interconnectats, poden aparéixer problemes que no es manifesten en proves unitàries o en petits grups de components. Aquestes proves globalment integrades poden revelar interaccions inesperades, problemes de rendiment, bloquejos de flux i altres qüestions que puguen afectar l'experiència de l'usuari final.

7.2.1. Postman

Postman és una plataforma de desenvolupament d'APIs que proporciona eines per a la creació, prova i documentació d'interfícies de programació d'aplicacions. Postman presenta una interfície senzilla que permet als desenvolupadors interactuar amb les APIs mitjançant peticions HTTP/S i avaluar les respostes en temps real. Aquesta eina és àmpliament utilitzada en el desenvolupament d'aplicacions i la integració de sistemes per garantir el correcte funcionament i la coherència de les interaccions d'API [24].

En aquest projecte, Postman ha sigut una ferramenta clau per a realitzar proves dels serveis desplegats a CloudHub. Ha permés emular diferents escenaris d'ús, des de crides simples fins a proves més complexes end-to-end simulant la interacció real del sistema de comandes amb els serveis dels transportistes. El seu entorn proporciona una àmplia gamma de funcionalitats, inclosa la possibilitat de configurar autenticació, establir capçaleres personalitzades i analitzar les respostes per identificar possibles errors.

Utilitzant Postman, s'ha realitzat una validació detallada de cada capa de la solució. Les crides a les APIs de les capes superiors i als sistemes dels transportistes s'han simulat

amb detall, permetent observar com les dades flueixen a través del sistema i com interactuen amb altres components. Aquest enfocament de proves ha estat clau per a identificar i abordar de manera efectiva qualsevol problema, incoherència o comportament inesperat abans de la posada en marxa de la solució en un entorn de producció.

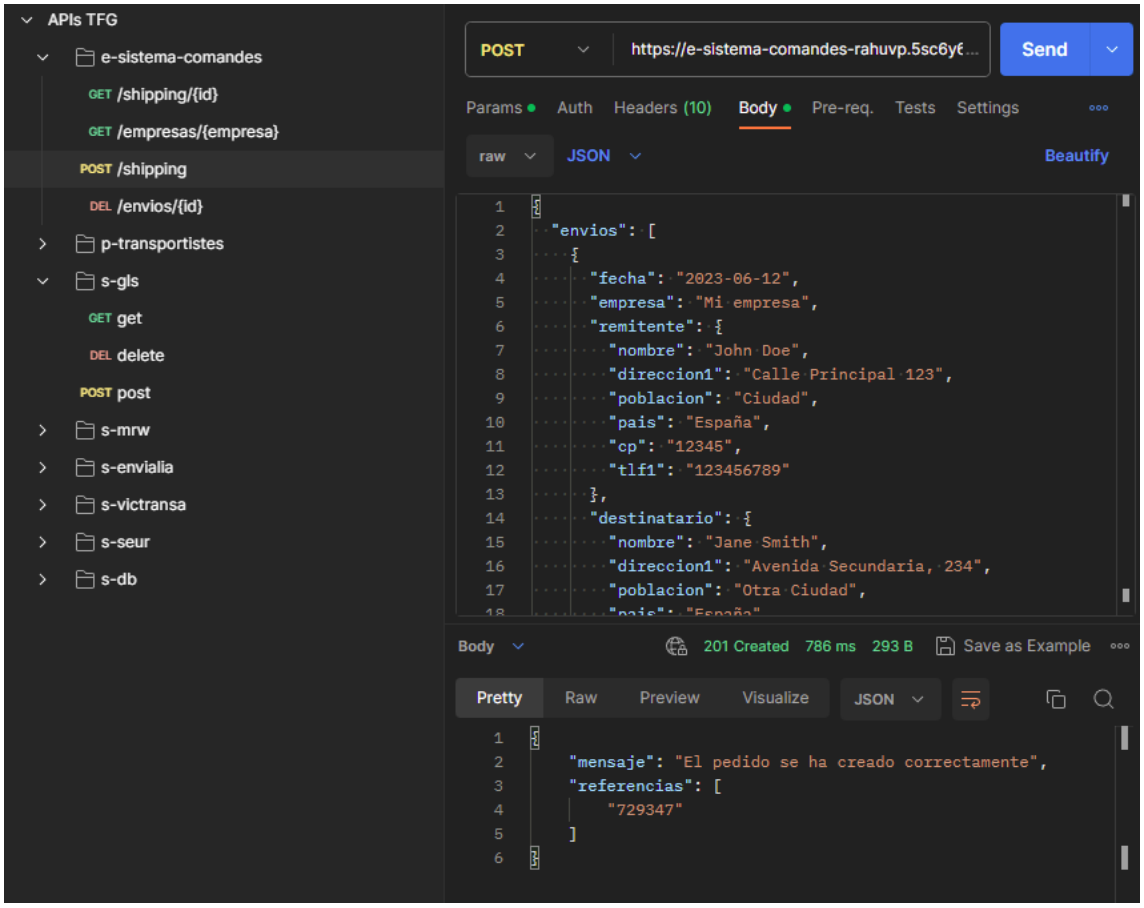

Figura 7.1: Estructura de peticions a l'esquerra i exemple de creació d'una comanda

En la figura 7.1 podem veure l'exemple de prova E2E de la creació d'una comanda mitjançant una crida a l'API e-sistema-comandes. A l'esquerra podem observar que s'han creat peticions per tal de fer proves en cada una de les APIs que s'han implementat.

7.3 Proves de càrrega

Les proves de càrrega, també conegudes com proves d'estrés, són un tipus de proves de software que tenen com a objectiu avaluar el comportament i el rendiment d'una aplicació sota condicions de càrrega intensa o situacions límit. Aquest tipus de proves es realitzen per determinar els límits operatius i identificar com es comporta el sistema quan s'enfronta a una demanda excepcionalment alta o a circumstàncies extremes. L'objectiu principal de les proves d'estrés és identificar les debilitats, els punts febles i les possibles fallades de l'aplicació quan està sotmesa a una càrrega excessiva.

En aquestes proves, se simula una gran quantitat de demanda, ja siga amb un nombre significativament gran d'usuaris simultanis, peticions concurrents o volums de dades

més alts del que l'aplicació normalment processaria. A mesura que augmenta la càrrega, es monitoritzen factors com el temps de resposta, i la tassa d'error en les respostes. Això permet identificar els punts on el rendiment comença a decaure, les possibles fallades i els efectes secundaris que podrien sorgir sota condicions extremes.

Les proves d'estrés són essencials per a assegurar que una aplicació puga manejar situacions de pic de demanda o situacions excepcionals sense col·lapsar. Aquestes proves ajuden els desenvolupadors i els enginyers a prendre decisions informades sobre com optimitzar i dimensionar l'aplicació per garantir un rendiment adequat en totes les circumstàncies. En el context d'aquest projecte, les proves d'estrés han estat utilitzades per avaluar la capacitat del sistema per processar un gran volum de transaccions i sol·licituds simultànies, assegurant que el sistema siga robust i estable fins i tot en situacions de càrrega extrema.

Aquest tipus de prova és essencial quan es desenvolupa un projecte d'integració, ja que és habitual que en aquest tipus de projectes existisquen pics de demanda de serveis i cal estar-ne preparat. Cal destacar que en aquest projecte no s'han fet servir els sistemes de l'empresa per a no fer ús de recursos que pogueren afectar el rendiment d'aquesta. Açò es deu al fet que aquesta empresa no comptava amb sistemes de prova i sols disposava dels sistemes de producció, és a dir, els que empren els usuaris finals.

7.3.1. JMeter

JMeter és una ferramenta de codi obert amplament utilitzada per dur a terme proves de càrrega i d'estrés en aplicacions web i altres serveis. La seua funcionalitat principal és simular un gran nombre d'usuaris i peticions per avaluar el rendiment i la capacitat d'una aplicació sota condicions de càrrega intensa [25].

JMeter ofereix una interfície gràfica intuïtiva que permet als usuaris configurar escenaris de proves amb facilitat. Les proves s'estructuren mitjançant plans de prova que inclouen diferents elements com peticions HTTP, grups d'usuaris, fitxers de dades, monitors i controladors. L'usuari pot configurar el nombre d'usuaris virtuals, les peticions que realitzen i les pauses entre les accions.

Una vegada configurat el pla de prova, JMeter simula les accions d'aquests usuaris virtuals, enviant peticions als servidors objectiu. A mesura que s'executen les proves, JMeter recopila dades sobre el rendiment, incloent-hi temps de resposta, latència, càrrega del sistema i altres mètriques rellevants. Aquestes dades es presenten en forma de taules, gràfics i informes que permeten als usuaris analitzar i avaluar el comportament de l'aplicació en diferents nivells de càrrega.

La implicació de JMeter en proves d'estrés és completa durant tot el procés. Permet als desenvolupadors i enginyers comprendre com es comportarà l'aplicació en condicions de càrrega intensa, identificar els possibles punts febles i les àrees de millora. Les proves d'estrés amb JMeter ajuden a dimensionar l'aplicació, optimitzar-la per al rendiment i assegurar que siga capaç de gestionar situacions de demanda elevada sense col·lapsar.

CAPÍTOL 8
Conclusions

Una vegada fet el treball, ha sigut visible la importància que té la integració d'aplicacions, no només pel que fa al rendiment, sinó també pel manteniment i el desenvolupament dins del marc dels projectes d'integració. El desenvolupament d'un projecte d'integració emprant una arquitectura apropiada des d'un principi no implica un esforç addicional. A més, si l'arquitectura compta amb diferents components, com ha sigut el cas en aquest treball, per la naturalesa d'aquesta es pot dur a terme el desenvolupament en paral·lel.

Un altre fet que cal destacar després d'aquest treball és la relació que trobem entre una arquitectura flexible i escalable i el temps de desenvolupament necessari en cas d'incorporacions o canvis al projecte. Durant el meu període de pràctiques vaig comprovar de primera mà el cost d'incorporar un nou transportista al projecte original i identificarne els aspectes a millorar exposats en aquest document, especialment quan es tracta d'un projecte gran.

Per resumir, aquest treball destaca la importància de l'adopció d'un enfocament de desenvolupament adaptable i escalable en els projectes. Establir una arquitectura adequada des de les primeres fases pot evitar ineficiències i dificultats futures, tant en rendiment com en desenvolupament. Ajornar la implementació d'una arquitectura apropiada pot derivar en majors esforços per a modificar-la posteriorment, fins al punt que aquest canvi podria ser contraproduent. Així doncs, podríem concloure que establir una arquitectura planificada i ben fonamentada des del principi és una decisió clau per assolir l'èxit i l'eficàcia en els projectes d'integració d'aplicacions.

8.1 Treballs futurs

El treball desenvolupat en aquest projecte és funcional i eficaç en l'àmbit de la gestió d'enviaments i la integració de múltiples transportistes. Aquesta solució ha estat dissenyada amb una arquitectura adaptable i escalable, la qual cosa ha permès no només satisfer els objectius inicials sinó també obrir les portes a futurs projectes que poden complementar i ampliar encara més els seus beneficis.

La implementació d'aquesta solució no només millora l'eficiència en la gestió d'enviaments, sinó que també ofereix una base sòlida per a futures ampliacions i millores.

La flexibilitat en l'arquitectura d'integració API-Led permet afegir nous components i funcionalitats de manera àgil i sense complicacions excessives.

A més, la naturalesa modular de la solució fa que siga fàcilment adaptable a les necessitats canviant de l'empresa, obrint la porta a la inclusió de noves funcions o la millora de les existents. Això significa que, a mesura que l'empresa evoluciona i s'adapta a nous reptes i oportunitats, aquesta solució pot acompanyar i facilitar aquest procés.

Alguns exemples de treballs futurs que es podrien dur a terme són els següents:

- Ampliació de la solució: Una possibilitat és continuar expandint la solució de gestió d'enviaments i integrar més transportistes i sistemes. Aquesta ampliació podria incloure l'adopció de més serveis d'API de tercers i la integració d'altres funcionalitats relacionades amb la logística i l'enviament.
- Seguretat i protecció de dades: Donada la importància de la seguretat i la protecció de dades en qualsevol sistema que gestione informació confidencial, seria convenient avaluar la seguretat de la solució actual i implementar mesures addicionals de seguretat per a garantir la integritat i la confidencialitat de les dades. Un exemple de millora de seguretat podria ser la implantació d'un mètode d'autenticació en cada crida amb un client-id i un client-secret. Aquests camps haurien de ser vàlids per tal que la crida es duguera a terme correctament.
- Desenvolupament de Front-end: Aquest projecte se centra principalment en la part de backend i en les integracions. Una línia de treball futur podria ser el desenvolupament d'una interfície d'usuari que facilitara la gestió i el seguiment dels enviaments per part dels usuaris finals o administradors.
- Inclusió d'altres funcions de l'empresa: Es podria considerar l'expansió de la solució per a incloure altres funcions o processos interns de l'empresa que puguen beneficiar-se de la integració i l'automatització, com ara la gestió de magatzems o la facturació.

8.2 Relació amb els estudis

"Redes de Computadores" m'ha servit per comprendre com es comuniquen els ordinadors a través de xarxes, i m'ha permès conèixer protocols àmpliament emprats com poden ser HTTP i HTTPS entre altres. Aquest coneixement ha estat de vital importància per al desenvolupament del projecte, ja que les comunicacions entre sistemes són la base de la integració.

"Gestió de Projectes" m'ha dotat d'eines i coneixements que han estat essencials per gestionar eficaçment el desenvolupament d'aquest treball. Les metodologies de treball i la planificació m'han servit per poder dur a terme un desenvolupament organitzat i tindre un marc de treball en equip per a projectes similars.

"Ingenieria del Software" m'ha introduït al concepte de desenvolupament de projectes dividits en capes i m'ha fet comprendre la importància d'aquest enfocament en l'arquitectura d'aplicacions.

"Bases de Dades y Sistemes d'Informació" i "Tecnologia de Bases de Dades" m'han proporcionat coneixements sobre la creació de consultes i la gestió de la informació. Això ha estat essencial per aprendre a integrar sistemes que operen amb bases de dades i per garantir un accés eficient a la informació necessària per a la gestió d'enviaments.

"Integración de aplicaciones" ha tingut un impacte directe en la realització del projecte, ja que m'ha introduït als conceptes fonamentals de la integració de sistemes. Les diferents formes d'integració, les tecnologies utilitzades i els reptes associats han sigut àmpliament abordats en aquesta matèria, i aquest coneixement ha estat aplicat directament en el treball. Concretament, aquesta assignatura m'ha permés consolidar el concepte de projectes dividits en capes i conèixer moltes maneres d'integrar sistemes, com poden ser els arxius, les bases de dades, el correu electrònic o els serveis SOAP.

Agraïments

En aquesta secció desitge expressar el meu sincer agraïment a totes aquelles persones i institucions que han contribuït en la realització d'aquest treball de fi de grau.

En primer lloc, vull agrair als meus estimats companys de classe, amb qui he compartit moments d'aprenentatge i superació al llarg d'aquests anys. La seua col·laboració i suport han estat inestimables, fent que aquesta etapa siga molt més enriquidora i emocionant.

Un agraïment especial a la meua família pel seu suport incondicional. Les seues paraules d'encoratjament han estat el motor que m'ha mantingut avançant i superant els reptes.

Tinc una profunda gratitud pel meu tutor de treball de fi de grau Joan Josep Fons i Cors, qui m'ha guiat amb paciència i coneixement al llarg del desenvolupament d'aquest projecte. Les seues orientacions i suggeriments han estat fonamentals per a la seua finalització amb èxit.

Vull expressar el meu agraïment a DISID, l'empresa on vaig realitzar les pràctiques, per brindar-me l'oportunitat d'aplicar els coneixements adquirits en un entorn real. La seua col·laboració i suport han estat essencials per a l'èxit d'aquest treball.

Finalment, vull reconéixer la Universitat Politècnica de València i tot el seu personal docent i administratiu. Els recursos, l'entorn i l'atmosfera d'aprenentatge proporcionats han estat fonamentals per al meu creixement personal i professional.

A tots ells, moltes gràcies per haver format part d'aquesta etapa i per haver contribuït a la consecució d'aquest treball. El vostre suport ha estat vital i sempre ho recordaré amb gratitud.

Bibliografia

[1] Wikipedia. Integración de aplicaciones para empresas, Consulta: 15/06/2023, url: https://es.wikipedia.org/w/index.php?title=Integraci%C3%B3n_de_aplicaciones_para_empresas&oldid=122203777.

[2] SAP. ¿Qué es la integración de aplicaciones? `https://www.sap.com/latinamerica/products/technology-platform/integration-suite/application-integration.html`.

[3] Patricia Alejandra Bazán. *Integración de aplicaciones*. Editorial de la Universidad Nacional de La Plata (EDULP), ISBN 978-950-34-2020-1, 2021.

[4] David S Linthicum. *Enterprise application integration*. Addison-Wesley Professional, ISBN 978-0201615838, 2000.

[5] Manuel Navarro. Los desafíos que plantea la integración IT. *Revista Byte*, 2022.

[6] CIS. ¿Cuáles son los tipos de servicios de integración?, 2022.

[7] Ross Mason. ESB and hub n' spoke architectures – ESB or not to ESB revisited part 2. *Mulesoft blog*, 2011.

[8] Gregor Hohpe and Bobby Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional ISBN 978-0321200686, 2004.

[9] Prashant Choudhary. What is API-Led connectivity? unlock business agility. *Salesforce - The 360 Blog*, 2023.

[10] Vinay Kumar. ESB and API-led architecture – enemies or friends. *Oracle Developers*, 2019.

[11] Brian Labelle. The pros and cons of API-led connectivity. *Linkedin*, 2023.

[12] Chloe Henderson. What is enterprise application integration (EAI)? pros vs. cons. *AnyConnector*, 2020.

[13] Marco Palladino. How API gateways complement ESBs. *NEW TECH FORUM*, 2022.

[14] Sandra Garrido Sotomayor. Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. *IEBS*, 2021.

[15] Redacciones APD. Cómo aplicar la metodología scrum y qué es el método scrum. *APD*, 2022.

[16] Anastasia Stsepanets. Modelo de cascada (waterfall): Qué es y cuándo conviene usarlo. *Ganttpro*, 2023.

[17] G. Robinson and M.J. Zhou. Utility applications should be integrated with an interface based on a canonical data model, not directly with each other. In *IEEE PES Power Systems Conference and Exposition, 2004.*, pages 1592–1596 vol.3, 2004.

[18] DISID. Por sexta vez consecutiva Mulesoft es nombrado líder por gartner, Consulta: 18/07/2023. url: https://www.disid.com/blog/mulesoft-lider-gartner-2021/.

[19] Divya. What is Mulesoft, and what is Mulesoft Anypoint Platform? *Cloud Foundation*, 2021.

[20] Wikipedia. Git, Consulta: 20/07/2023, url: https://es.wikipedia.org/w/index.php?title=Git&oldid=151585023.

[21] Wikipedia. Servicio web, Consulta: 20/07/2023, url: https://es.wikipedia.org/w/index.php?title=Servicio_web&oldid=151607451.

[22] Microsoft. Planificación de los entornos de desarrollo, pruebas, ensayo y producción, 2023.

[23] QAlified. Pruebas end-to-end: definición, ejemplos, herramientas, y más, 2022.

[24] Assembler Institute. ¿Qué es Postman? Características y Ventajas, 2022.

[25] Satish Sheti. Apache Jmeter: Todo lo que necesitas saber. *Geekflare*, 2022.

APÈNDIX A

Objetivos de desarrollo sostenible

Grau de relació del treball amb els Objectius de Desenvolupament Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**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 desenvolupament sostenible és una qüestió de gran importància i el coneixement de com els nostres projectes poden contribuir a aquesta causa és fonamental. En aquest context, el present treball de final de grau ofereix una oportunitat per avaluar com el projecte pot alinear-se amb els Objectius de Desenvolupament Sostenible (ODS) establerts per les Nacions Unides.

En la taula de relació del treball amb els ODS presentada, podem observar com algunes connexions són més clares que d'altres. En primer lloc, cal reconéixer que hi ha ODS en els quals el treball no té una relació directa o rellevant. Aquest és el cas dels ODS 1, 2, 3, 5, 6, 7, 10, 12, 13, 14, 15, 16 i 17, els quals no estan directament implicats en el contingut o les activitats abordades en aquest projecte.

No obstant això, hi ha ODS amb els quals aquest treball està més estretament relacionat. El treball encaixa d'alguna manera amb l'ODS 4, Educació de qualitat, ja que implica la integració i millora de processos empresarials que poden conduir a un millor funcionament i, per tant, a una educació de qualitat en l'àmbit del transport i la logística.

Els ODS 8 i 9, Treball digne i creixement econòmic, i Indústria, innovació i infraestructures, tenen una alta relació amb aquest treball. Aquest projecte aborda la millora de la comunicació i la gestió d'un sistema d'enviaments amb múltiples transportistes mitjançant l'ús d'una arquitectura d'integració d'API-Led. Aquesta millora pot tenir un impacte directe en l'eficiència i la productivitat de les operacions, contribuint a un creixement econòmic sostenible i a la millora de les condicions de treball per als empleats implicats en aquestes operacions.

A més, l'ODS 11, Ciutats i comunitats sostenibles, també està relacionat en certa manera amb el treball. La millora de la comunicació i la gestió en els processos d'enviaments pot influir en la millora de la qualitat de vida en les comunitats involucrades en aquestes operacions logístiques.

Si bé hi ha ODS amb els quals aquest treball està més alineat, també és important admetre que alguns d'ells podrien tenir una relació més tangencial o indirecta. Aquesta relació podria ser més evident si consideràrem les implicacions socials, econòmiques i mediambientals del projecte a llarg termini.

En conclusió, mentre que el treball no està estretament vinculat amb tots els ODS, és evident que té una relació significativa amb diversos d'ells. L'ús d'una arquitectura d'integració d'API-Led per millorar la comunicació i la gestió dels processos d'enviaments amb múltiples transportistes pot tenir un impacte positiu en l'eficiència, la productivitat i la sostenibilitat, i per tant contribuir als objectius de desenvolupament sostenible en diferents àmbits. Aquesta anàlisi ens recorda la importància de considerar l'impacte dels nostres projectes en la cerca de solucions que promoguen un futur sostenible per a tothom.