INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Digital Transformation Integration & API Management Framework

JOÃO PEDRO GOMES PINTO
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Digital Transformation Integration & API Management Framework

João Pinto

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Software Engineering

Supervisor: Dr. Alexandre Bragança
External Supervisor: Ricardo Vila Nova & Pedro Richard Kretschmann

Porto, October 13, 2022

Dedicatory

To Johnnie. May his stubbornness live on.

Abstract

Digital transformation is revolutionizing the way organizations reuse legacy systems and connect them with the ever-increasing number of enterprise applications. These applications are usually hosted in the cloud with the legacy and other systems in the companies' own data centers. This means that hybrid integration architectures are becoming a crucial part of the day-to-day processes inside organizations. The problem with different platforms is that it produces challenges. The same data is spread across different data structures, with different names and formats.
This dissertation, devised with Deloitte, aims to design and implement a hybrid integration framework that connects and reuses the legacy systems with the cloud enterprise applications seamlessly to utilize the same data across all systems.
Firstly, the existing integration platforms and integration software that could support the development of the API framework will be presented and described. A set of architectural patterns will be presented and described with their respective advantages and disadvantages and the way they fit in the API framework that will be created.
The solution will then be evaluated with the Quantitative Evaluation Framework as well as by using system testing to see if the developed solution has quality.

Keywords: Hybrid Integration, Digital transformation, API Framework, Cloud platform

Resumo

A transformação digital está a revolucionar a maneira como as organizações reutilizam sistemas legacy pré-existentes, e os ligam com aplicações enterprise, que sofreram um aumento significativo nos últimos anos. Estas aplicações estão regularmente hospedadas em plataformas cloud com outros sistemas e os sistemas legacy hospedados em datacenters próprios da empresa. Isto significa que, atualmente, as arquiteturas de integração assumem um papel crucial nos processos de dia-a-dia das organizações. O problema com a utilização de várias plataformas diferentes é que criam desafios ao cliente como a diferença de mapeamentos e de formatos de dados que usam, criando dificulades em perceber como a mesma informação é armazenada e apresentada nas diferentes plataformas .
Esta dissertação, elaborada em contexto empresarial com a Deloitte, tem como objetivo o design e implementação de uma camada de integração híbrida que ligue e reutilize os sistemas legacy nos datacenters do cliente com as aplicações enterprise que o cliente usa de maneira ininterrupta.
Primeiramente, as plataformas de integração existentes e software de integração que podem suportar o desenvolvimento da framework de APIs, serão apresentados e descritos. Padrões arquiteturais que possam ser utilizados na framework de integração híbrida serão apresentados e descritos bem como as suas respetivas vantagens e desvantagens.
A solução será depois avaliada com a Quantitative Evaluation Framework e com a testagem do sistema para avaliar se o sistema satisfaz os requisitos.

Palavras-chave: Integração Híbrida, Transformação Digital, Framework de APIs, Plataforma Cloud

Acknowledgement

This is probably the hardest part to write of this entire dissertation. Firstly, because I have some people to thank, and secondly because it's going to be a long one.
The first people that I want to thank are my family and my girlfriend, Rita. This journey only got here because of their love and help. To my parents that raised me to be the person I am today, to Filipe that grew up to be a great companion, and to Rita that helped me endure the hardships that appeared along the treacherous path.
I also want to thank my friends. To Cláudio and to our therapy car rides that will forever be on my heart. To Mariana, my greatest academic inspiration. To Carvalho, my polyvalent friend. And to Teresa and Catarina, that helped me go through the first year unscathed and who shared my grievances.
A very special thanks to my second family in my second home in Vila Real. A lot of this dissertation was written there, and they helped me maintain the work-life balance as it should be. For Tozé and Fátima, Joana and João, this is also for you.
To my colleagues at Deloitte, that helped me start my professional life in the best way possible, that welcomed me with open arms and pushed me to other heights, and to my buddies in ISEP, that helped me reach this phase.
Finally, I'd like to thank my supervisors, both from ISEP and Deloitte, whose feedback and advice proved indispensable.

Contents

List of Figures

List of Tables

List of Acronyms

AHP	Analytic Hierarchy Process.
CPI	Cloud Platform Integration.
EC	Employee Central.
ECP	Employee Central Payroll.
ESB	Enterprise Service Bus.
FFE	Fuzzy Front End.
HCM	Human Capital Management.
HIP	Hybrid Integration Platform.
HR	Human Resources.
HTTPS	Hypertext Transfer Protocol Secure.
NCD	New Concept Development.
NPD	New Product Development.
QFD	Quality Function Development.
QoS	Quality of Service.
SaaS	Software as a Service.
SF	SuccessFactors.
SFTP	SSH File Transfer Protocol.
SIT	System Integration Testing.
SOA	Service Oriented Architecture.
UAT	User Acceptance Testing.

Chapter 1

Introduction

This chapter provides the context of the dissertation, a description of the problem, the objectives, and the document structure.

1.1 Context

In today's world, digital transformation is an important and prevalent topic for organizations so that they work more efficiently and faster. Digital transformation is the employment of newer technology to ameliorate the performance and day-to-day processes of an organization and, as a consequence, scale it [1]. It is normal to see companies using different platforms for differing departments with various ends from diverse vendors and their self-developed legacy systems. Older applications with older architectures that can slow down progress [2], are being renewed so that no time and money is lost in creating systems that are already implemented and no knowledge or data gets lost. This means that there needs to be a component that links these systems - an integration layer. Integration, combining systems to work as one single cohesive unit, is becoming pivotal for companies to operate these systems in an efficient and reliable manner [3] [4].
With digital transformation, as a consequence, a new shift in the industry began. Companies are changing their on-premise approach, where they previously had servers with their systems in-house, to cloud platforms and this means that integration also has to adapt to this and connect cloud applications or to also be available in the cloud. But, large companies with both cloud platforms and legacy on-premise systems that contain a lot of the business logic, would find that migrating these systems to the cloud would be too expensive or too complicated to do so. Then, a middle ground has to be achieved in order to use both legacy systems with the advantages of cloud platforms. In order to solve this, Hybrid Integration was developed[5]. It combines on-premises with cloud integrations running in integration platforms.
The purpose of this project is to create a hybrid integration framework that connects Deloitte's client's inherited legacy systems with their new cloud platform. An integration framework offers, as defined by Oracle, "a systematic, standards-based architecture for hosting application views, which are business-focused interfaces to corporate systems" [6]. Using MuleSoft, using an API-Led approach, a paradigm for Successfactors and the On-prem systems interact with each other will be created [7].
This client is a large retail company with a presence in various locations and it had a merger with another large retail company. A stage of this project was already been developed for the Netherlands, but two other stages are being developed - for the United States and for Europe, the region that this dissertation focuses on. For the Netherlands, a full integration framework was developed and is fully operational. It connects SuccessFactors to the

Netherlands' brands on-prem systems with a hybrid integration framework that is also housed on-premises.

1.2 Problem

The client invested in SAP SuccessFactors Employee Central for their Human Resources (HR) needs. This merger meant that a massive influx of employees to their systems and several legacy systems from the other various branches and brands of the merged retail company were inherited. With more than 300 thousand employees, this severely impacted the efficiency of their HR systems. If a change is made in an employee from Belgium, the HR team has to make a change in the Belgium's brand HR system and then make another change in the central repository for the main company, and there could be differences between what is submitted in Belgium and what is present in the main companies' HR system as in there could be specific fields that don't exist or have a country-specific meaning that may be lost. This method is unthinkable since there are so many different countries and different employees in the company.
The existing on-prem legacy systems need to be reused and integrated into their new ecosystem, so a custom hybrid integration solution needs to be developed to orchestrate the data between them and the new HR repository hosted in the cloud - Successfactors. It will be hybrid because both the HR repository and the API framework will be hosted in the cloud and the brands' systems on-prem. This has already been done for the Netherlands, but the rest of the European brands, encompassing several countries, still lacks this integration layer. This means that the new assets for these countries will be added to this existing code base that was developed for the previous iteration of the digital transformation to encompass the new requirements of these brands.
Connecting SAP SuccessFactors to the on-prem systems will have some caveats. The cloud platform limits the rate of calls made to it. By creating this custom layer, the orchestration of data will need to go around this problem and also reduce the reliance on other SAP products, since, to integrate with the on-prem systems, some of the brands have some SAP Products that integrate them with their own Successfactors instances will cease to exist, as there is a new Successfactors instance that will have all of the employees data. These products will be decommissioned so the new layer will have to replicate them, saving money, adding customization and a chance to scale the solution further if needed, since new products may be added to the client's ecosystem or new brands might be merged.

1.3 Objectives

To properly define the objective of this dissertation, a research hypothesis is required to perform a good research[8]. Firstly a question has to be posed in order to accurately present the hypothesis - "How can the cloud platform SuccessFactors be connected with the inherited systems in order to decouple and allow for the fast transaction of data and limit the rate of requests to SuccesFactors?", making the research hypothesis "A Hybrid Integration Framework like the one in this dissertation would orchestrate the data from Successfactors to the brands' on-prem systems in the appropriate time span and rate".
With the question presented, the objective of this dissertation is to create an integration layer that links the systems and is efficient, auditable, safe, reusable, and abstract.

1.4 Research Methodology

This dissertation is going to be orchestrated by researching the problem itself and possible solutions to then apply them in the design and development, so the Design Science Research method was used.
This method is "a qualitative research approach in which the object of study is the design process, i.e. it simultaneously generates knowledge about the method used to design an artifact and the design or the artifact itself"[9]. The stages of this methodology are presented in figure 1.1.

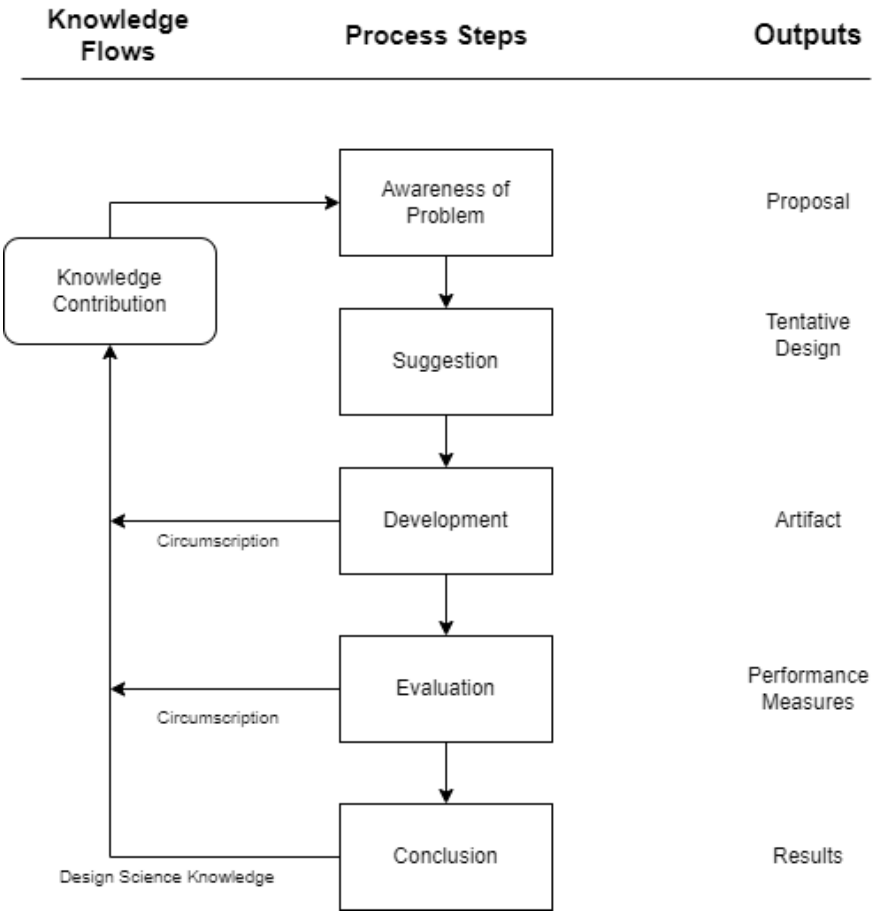


Figure 1.1: Example of the Design Science Research Methodology[10][11]

The research followed this flow by, first and foremost, identifying the problem - linking several systems on the cloud and on-prem in a way that removes the cloud platform perils. Then, the suggestion was to develop a hybrid integration layer that solves the problems of the client with the cloud platform. The development will be performed next, where the artifact is created to then be evaluated by system tests and by applying the QEF method.

1.5 Contributions

By analyzing the problem and the objectives of the dissertation, it is clear that, with the developed solution, it is expected that the client organization will use a new integration layer that connects cloud systems with the inherited legacy systems, with the exception of

certain systems that will be decommissioned, since the new solution will orchestrate the data flow in a different way that will make them useless. The new layer will consume data from SuccessFactors and feed the downstream systems, creating a seamless link between the systems and services across the organization.
The organization will have a safe and scalable solution that will be future-proof, and, in return, potentiate the business as a whole and provide efficiency that the company otherwise would not have with direct connections between systems and that will save them money in the long term.

1.6 Document Structure

This document is arranged into six chapters, which will now be introduced and properly described.

- Introduction - Provides an introductory view into the dissertation, and explains the context and problem that it proposes to solve
- State of the Art - In this chapter, a summary of already available hybrid integration platforms and tools are presented as well as integration approaches that are relevant to the problem.
- Value Analysis - The opportunity is identified and analyzed as well as the ideas that were materialized in order to solve the questions that were brought up to solve the problem. Then, the Analytic Hierarchy Process (AHP) method was used to choose an alternative to solve this problem. Lastly, the Quality Function Development (QFD) method was used and a Value Proposition was created.
- Design - Tools and design patterns were enumerated and explained as well as their advantages and drawbacks, the cloud deployment architecture is presented and the approach that was chosen to develop the framework.
- Experimentation and Evaluation - The evaluation method is explained and the methods, indicators, and hypotheses for the evaluation are laid out.

Chapter 2

State of the Art

In this chapter, a comprehensive analysis of the existing platforms that could be used to solve the problem and implement the system is performed. Firstly, iPaaS are described and examples are presented. Then, Hybrid Integration is presented and explained, and a reference architecture, as well as the building blocks of a Hybrid Integration Platform (HIP), are presented.

2.1 Integration

When applied to IT, integration is the process of connecting heterogeneous subsystems to create a larger, cohesive system with a unified language [12]. A major challenge that most businesses face is that information is isolated due to legacy systems in the enterprise's ecosystem while, simultaneously, businesses live surrounded by frequently shifting markets, constant emergence of new systems, and competition expanding. If old systems are to be removed or replaced, the organizations will suffer serious consequences in all areas [13].
The most simple and the very first integration pattern that was created was point-to-point integration, where the systems that needed to communicate with each other, were connected directly. With the disparity and complexity of systems that companies have nowadays, integration approaches evolved to tackle convoluted problems that complex integrations inherently bring.

2.2 Architectural Approaches to Integration

According to Gregor Hohpe, the architectural approaches to integration can be grouped in four general classes[14]:

- File Transfer
- Shared Database
- Remote Procedure Invocation
- Messaging

This section will explain these basic four approaches and the most recent patterns used in enterprise system integration.

File Transfer

File transfer is an integration approach where each respective system creates files that the integration layers transform into formats that other systems can consume and process [14]. File Transfer is somewhat of a simple method because "no extra tools or integration packages are needed", but, as a consequence, many tasks must be undertaken by the developers themselves. [14]. The systems must have common file naming conventions and agree on the directories they are stored in. The system that creates the file should assert that file names are unique and that no two systems are using the same file at the same moment [14]. Another problem with this approach is that updates are few and far between so some systems may be running with outdated data [14].

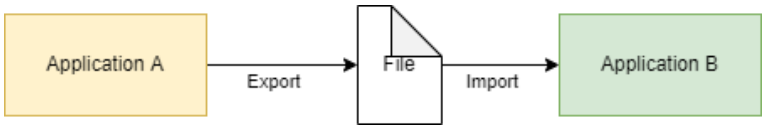


Figure 2.1: File Transfer [14]

Shared Database

As stated before, one of file transfer's shortfalls is immediacy. Modern systems need the most updated data as quickly as possible, but not so quickly that it may cause problems - "If an address is updated inconsistently in rapid succession, how do I decide which one is the true address? (...) I have to remember who is master for what data."[14]
When data is transferred through File Transfer, the formats of the data may not be enforced well enough and problems with integrating different formats of data can occur when There are always subtle business issues when data is viewed from different perspectives. Often these views are incompatible with each other [14].
This is where a shared database comes in. The systems use the same database so the data is always consistent throughout and it removes the problem of different data formats [14].
The problem with this approach is that developing a schema that works across multiple systems is difficult. Another issue is that systems may need to wait for other systems to finish using the database [14].

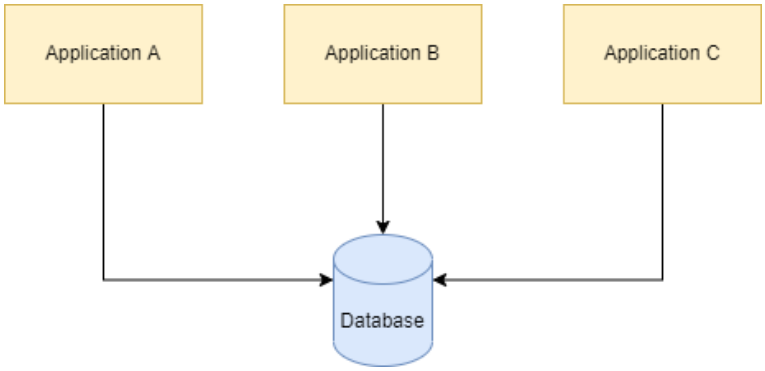


Figure 2.2: Shared Database [14]

Remote Procedure Invocation

The shared database approach is unwilling to change and the file transfer's process is haltered when changes are required. Some data requires certain specific transformations in certain cases and the former approaches aren't exactly open to certain transformations [14].
Applying the principle of encapsulation - "technique for minimizing inter-dependencies among separately-written modules" [15] - Remote Procedure Invocation allows for systems to call procedures in other systems to get or to change the called system's data. Another application will change the data that the first application is working on by calling it directly [14].
But, since the procedures are *remote*, they may be drawn out [14].
Another problem is that sequenced procedures are hard to change since, when one is changed, every single system in that sequence needs changing [14].

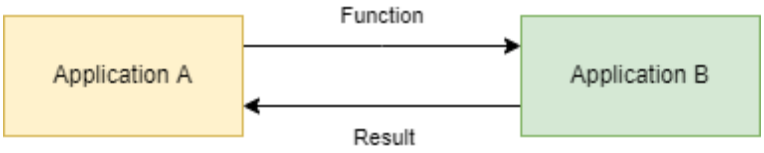


Figure 2.3: Remote Procedure Invocation [14]

Messaging

While the benefits of remote procedure invocation are great, it is slow and inherently a point-to-point integration method. If a system fails, the whole enterprise systems go down and each system knows information about other systems [14].
Messaging uses smaller packets for ease of consumption for the end systems, with an easily changeable schema, asynchronous, and with a retry mechanism [14]. The smaller packets make the transaction faster and more frequent, allowing the systems to be updated more regularly, each system may have very different conceptual models, it decouples the systems, and asynchronicity "does not require the systems to be functional at the same time" [14].

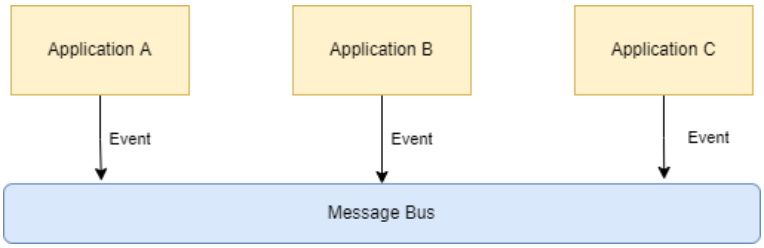


Figure 2.4: Messaging [14]

2.2.1 Enterprise Application Integration

Enterprise Application Integration (EAI) joins the systems in an enterprise ecosystem with an intermediary that "combines all applications, their databases, and processes running internally or externally" [16] acting as a "linking interface uniting multiple running applications in a company" [16].
Enterprise application systems integration should be at the forefront of every enterprise's goals to increase productivity, lower overall costs, or optimize business operations. The integration of internal and external information systems is critical to the success of any corporate project. It potentiates information exchange, which improves the value and quality

of information. It allows for the efficient amalgamation of various separate systems, data exchange, and data sharing between all processes of an enterprise, ensuring that all components of the firm operate over a database [16].
Inside organizations, there are three main categorizations of EAI: Horizontal, Vertical, and Three-dimensional management integration.

Horizontal Integration

Horizontal integration refers to intra-organization data integration, supply chain data integration, and customer data integration, all of which aim to reduce manufacturing lead times and costs while increasing consumer value. Its goal is to execute productive working division and system coordination through holistic cooperation of process and service based on the core of the process-driving technology. It may efficiently overcome consistency issues between data models in various systems, technology route differences in various systems, and diverse structural issues in both hardware and software platforms [16].

Vertical Integration

Vertical intra-organizational integration seeks to combine systems established at various administrative levels of a company. Despite incorporating functionality from several levels of an operational system into a single application is not unprecedented, it is more frequent to have different systems created to cover business operations at different managerial levels. Vertical integration removes the barriers that exist between supervisors and their staff. Because of strict hierarchical links, there is information asymmetry between superiors and their subordinates. Vertical integration aims to develop a healthy link between the flow and data exchange amongst superiors and subordinates in order to share business knowledge on business processes [16].

Three-dimensional management integration

As we live in an e-commerce era, enterprise information assets grow at an exponential rate, enterprises must confront the current issue that enterprise information resources must be allocated in a sensible and adequate manner to satisfy the requirements of online consumers, a concern that company executives place a high value on [16].
Enterprise's three-dimensional Information Resources Management philosophy is equal to achieving horizontal and vertical integration. Enterprise three-dimensional integration strives to break down relevant communication barriers across departments, remove intra-organizational information asymmetry, and improve intra-organizational information transparency. Horizontal and vertical integrations are the foundation and premise of three-dimensional integration, which will eventually execute reciprocal action of processes across organizations. Businesses attain a flat structure in intra-organizational communication and data network transmission through three-dimensional management integration. It can accomplish decentralization and timely decision-making in terms of judgment efficacy [16].

Service Oriented Architecture

Globalization needs an agile and adaptable architectural foundation. Service Oriented Architecture (SOA) is a significant topic in EAI since SOA responds to forthcoming corporate needs more quickly and efficiently [17]. SOA does not specify a particular implementation technology, but instead the architectural elements and principles used to represent a certain service in a specific environment [18].
SOA is "a methodology for developing and deploying distributed capabilities that may be

managed by multiple ownership domains" [19]. Its basic objective is to enable agility by bridging the 90-degree gap between IT applications and the business process layer [17].
Enterprise applications are large commercially complicated systems that must meet hundreds or even thousands of separate needs, installed on a variety of platforms of different ages and other difficult challenges such as distributed software, multiple platforms, application integration, diverse protocols, the Internet, multiple devices, and so on. As a result, enterprise applications necessitate interoperable solutions. Prior to SOA, however, apps did not enable open interoperable solutions and relied on proprietary APIs, necessitating a high degree of coordination[17].

Services
SOA is a collection of services, which may be broadly defined as a series of software components that perform business operations independently. A service ought to have platform-independent interfaces and self-contained functionality that is weakly connected with other services - loosely coupled - "we know what must go in and out but are unconcerned about how it accomplishes the translations" [20].

Objectives of SOA
SOA is fundamentally, at its heart, a "distributed architecture with a design focus on services", which are meant to serve business activities [20].
SOA needs to take a few concepts into consideration [21]:

1. Distribution - Systems are consisted of multiple software components that run on diverse networks utilizing distinct communication protocols [21].
2. Heterogeneity - The system components are located in remote heterogeneous (different from one another) settings, and engineers have really no control over the actual specifics [21].
3. Transparency - Services should be protocol agnostic, providing for transparency between providers and customers [21].
4. Dynamics - No decisions may be taken in advance so components must be configured at runtime [21].
5. Process-orientation - It's important to create loosely coupled but coordinated services for several business processes [21].

2.2.2 Enterprise Service Bus

An Enterprise Service Bus is a highly distributed, event-driven, integration-focused corporate SOA. It is a "standards-based integration platform that integrates messaging, Web services, data transformation, and intelligent routing" to link and orchestrate with transactional integrity the interaction of a large number of heterogeneous applications across extended companies. An Enterprise Service Bus (ESB) is appropriate for every project, regardless of its size [22].
An ESB serves as the SOA's implementation backbone. Thus, it offers an event-driven, loosely connected SOA with a highly dispersed universe of named routing destinations across a multiprotocol message bus. Applications (and integration components) in the ESB are abstractly detached from one another and connect over the bus as event-driven logical endpoints [23].
It may offer a framework that facilitates the addition of additional applications. All applications are linked to the ESB, and all apps interact through the ESB. An ESB layer handles the

challenges of implementing the integration logic. The ESB includes all types of integration functions, workbenches, and management environments out of the box, making it much simpler to develop a new integration flow between applications [22] [24].

The point of using ESB

Numerous point-to-point and EAI integration initiatives have resulted in the so-called unintended architecture. It consists of faulty point-to-point connections between closely connected applications on the one hand, and so-called integration islands on the other [24].
Point-to-point integration results in unreliable, insecure, unmonitorable, and in general unmanageable application communication paths. The apps are also closely connected, which requires the integrating application to be aware of the target application, the interface methods to call, the needed protocol to communicate, and the required data format to convey. Process and data transformation logic are contained inside the apps. Thus, each time an application undergoes a change, a new integration project must be initiated to restructure the dependent apps [24]. ESB solves this issue.

Characteristics of an ESB

According to Chappell [23], ESBs have multiple characteristics:

- **Pervasiveness**
  ESBs may serve as the central component of a ubiquitous grid. It has worldwide reach across departmental organizations, business divisions, and trade partners, and is capable of reaching an extended company. The flexible foundations of an ESB allow it to adapt to any sort of integration environment, making it suitable for localized integration initiatives [23].
  Apps are able to see and share data with other applications and services that are hooked into the bus as required. While web-services interfaces are an essential component of an ESB design, not all applications must be upgraded to become genuine web services in order to participate in the ESB. Multiple protocols, client API technologies, legacy message environments, and third-party application adapters are used to accomplish connectivity [23].

- **Standards-Based Integration**
  Using standard interfaces, an ESB offers an infrastructure that supports both industry-standard integration components and proprietary parts. These interfaces and components based on standards are assembled in a logical manner to form a connectable architecture [23].

- **Highly Distributed Integration and Selective Deployment**
  These integration capabilities are provided by the ESB as distinct services that may operate in a highly dispersed manner and grow independently of one another [23].

- **Distributed Data Transformation**
  The ability to transform data formats across applications is a crucial component of any integration strategy. Numerous programs do not describe equivalent data using the same format [23].
  In an ESB setup, data transformation is intrinsically a component of the bus. Transformation services that are tailored to the requirements of certain applications connected

into the bus may be discovered and used anywhere on the bus. Due to the fact that data transformation is a fundamental aspect of an ESB, it may be seen as a solution for "impedance mismatch between applications" [23].

- **Extensibility Through Layered Services**
  An ESB provides all the necessary fundamental features for nearly every integration project and may be supplemented with layered technologies for more specialized needs [23].

- **Event-Driven SOA**
  In an event-driven, ESB-enabled SOA, applications, and services are viewed as abstract service endpoints that can easily react to asynchronous events. The underlying connectivity and infrastructure are abstracted by Service Oriented Architecture. Services receive events and the ESB routes them wherever they must go and services are incognizant - their comprehension of protocols and the routing of the messages is unnecessary [23]
  . Custom integration services may be established, expanded, and reused as ESB functionality in an ESB SOA. With the objective of automating business operations in real-time, application endpoints that are accessible as services may be linked with specialized integration enablers to produce composite services and process flows that can be recombined and reused for diverse purposes [23]
  .

- **Process Flow**
  Process flow capabilities of an ESB span from "basic sequences of discrete steps to complex business process orchestration with parallel process execution pathways using conditional splits and joins. These may be managed via the use of basic message metadata or an orchestration language such as BPEL4WS" [23].
  The "process flow characteristics of the ESB" enable the design of departmental or business unit-specific business processes that can also live inside "a wider integration network" [23].
  ESB's process flow is able to route messages depending on their content [23].
  Because the process flow has its foundations in distributed SOA, it may traverse extremely dispersed networks without the need to be aware of the network or overall architecture [23].

- **Security and Reliability**
  Even amongst ESB nodes themselves, the security between applications and the ESB is capable of creating and maintaining the strictest authentication, credential management, and access control [23].

- **Autonomous but Federated Environment** Every single department or business unit must be autonomous. Nonetheless, they continue to depend on shared resources and accounting and reporting data that feed into a common business function. This creates a "technological and cultural barrier" since not all traffic will be routed to a single point of failure in the main office of a company. "Each entity should have its own IT function and not have to consider routing all of its message traffic or delegating control of its business rules and security domains via a centralized integration broker at one or the other location" [23].
  ESB's distributed properties are realized by decoupling endpoint definitions from the message bus. Physical deployment information includes underlying wire protocols, as

well as orchestration and data routing between endpoints. The federated qualities are realized by the ESB's capacity to separate and cross-application domains and security borders selectively [23].

- **Operational Awareness**
  ESB allows for the timely recording and dissemination of information. ESB provides the foundations necessary to stack new capabilities on top of the ESB in order to acquire real-time visibility into the business data. Auditing and tracking features that are a fundamental component of an ESB enable you to monitor and track the health of your business processes and the flow of messages across your SOA [23].

- **Incremental Adoption**
  ESB's flexibility to permit progressive adoption is one of its most distinguishing characteristics. There is a need to pursue broader strategic decisions that depend on the integration and reuse of current IT assets [23].
  The ESB's "federated/autonomous features" also add to the possibility to embrace an ESB project by project. ESB integration projects with incrementally phased deployments may deliver swift benefits and also connect to an existing infrastructure [23].

2.3 Modern Integration Solutions

In recent times, a surge of SaaS (Software as a Service) cloud products has appeared in the enterprise market and small, medium, and large organizations are migrating to these solutions to support their business processes and day-to-day activities. But no solution on the market fulfills the entirety of all of the requirements of a business and, as a result, multiple SaaS solutions are adopted to accomplish and satisfy these needs. That brings out the concern of integrating these systems with each other and with legacy applications that are on-premise, so integration strategies that can link seamlessly these systems are required [25][26].
When choosing an integration approach, there are multiple concerns that need to be considered, such as [26]:

- Security - Since there's the transfer of sensitive data between systems, the solution must be security-focused. Cloud platforms raise security concerns and the connection between the cloud platform and the legacy systems must have authentication and authorization in place.
- Scalability - The integration layer will need to be future-proof and be capable to handle the possibility of other systems being added.
- Flexibility - Certain complex integrations will be conducted and the prospect of future migrations of systems to the cloud needs to be taken into account.
- Monitorization - The solution must offer the client the tools to monitor and control the system, which sometimes are features that are lacking in current SaaS applications.

2.3.1 iPaaS

Integration Platform as a Service or iPaaS for short is a cloud-based integration solution to integrate various systems, private and public clouds [27].
Its objective is to integrate and potentiate the company's systems with a flexible solution

that can meet a wide range of integration scopes [28][29].

Since most small and medium-sized companies don't have the budget to hire skilled personnel to perform integrations, this led to the iPaaS rise. iPaaS eases the complexity of integration and it is increasingly being used for hybrid integrations [30].

iPaaS on the Market

iPaaS solutions have risen in popularity in recent times. A Gartner report states that in 2020 alone, iPaaS rose 38%, reaching $3.5 billion with North America having 40% of the market share, followed by Europe with 33% [31].
The market is divided into three tiers:

- Market Leader - Leading iPaaS solution from an established vendor that meets a broader set of use cases and that offers continuous support and new features. This is the tier that will be further explored below [30].
- Market Challenger - Has a set place in the market and is competitive. Offers a good variety of use cases, but does not display a clear comprehension of the market [30][32].
- Niche Player - Is used for specific integration use cases [30][32].
- Visionaries - These providers comprehend and have a good market understanding, but do not conduct a good execution [32].

Figure 1: Magic Quadrant for Enterprise Integration Platform as a Service

Source: Gartner (September 2021)

Figure 2.5: Gartner Magic Quadrant[32]

MuleSoft Cloudhub

MuleSoft's Cloudhub is the iPaaS integrant of MuleSoft's Anypoint platform. It is a cloud platform that is enterprise-oriented and it allows the connection of both on-prem and cloud systems. It also allows you to connect with MuleSoft's ESB. Furthermore, the existence of Anypoint Exchange allows users to use assets that other users created [33][30].
The Cloudhub platform offers standard connectors and integration models for popular SaaS, offering a quick way for companies to discard point-to-point integrations and create a solid, scalable solution [33].

Dell Boomi AtomSphere

Dell Boomi AtomSphere is a cloud integration platform that provides tools for hybrid integration. It considers itself the first iPaaS solution on the market [34][35]. It reduces time and effort in integrations since certain use cases with connections to Software as a Service (SaaS) systems are already covered with certain connectors, providing more value for the time of the customer [36].

Informatica Intelligent Cloud Services

This platform was identified by Gartner as the leading iPaaS solution for eight years in a row [32]. It is a cloud-native platform that also offers an artificial intelligence engine - Claire - to potentiate digital transformation [37].

2.4 Hybrid Integration Platform

The Hybrid Integration Platform is a rising capability framework that integrates both on-premises and cloud systems into a sole element seamlessly. It helps organizations by digitally transforming their legacy systems that otherwise would be hard and without its perils to replace [38].
It differs from an iPaaS because it is a capability framework - it creates a common language between systems and leverages the capabilities of an organization such as integration specialists and the users, while iPaaS is a "suite of cloud services".[39][40]. A hybrid Integration Platform is a custom-made solution built for the organization and an iPaaS is a service that a company provides and the organization buys it [39]. A Hybrid Integration Platform can be deployed in the cloud, on-premises, or hybrid [40].
It uses, as building blocks[38]:

- On-prem systems
- Integration Layers
- iPaaS
- iSaaS

Hybrid Integration Reference Architecture

By analyzing common patterns that can be identified in the solutions used by organizations when adopting hybrid integrations.[41] Several perspectives for hybrid integrations are available, such as application, data, and infrastructure.[41] In figure 2.6, a high-level reference

architecture from an application perspective can be seen, in which the integration of on-premises with cloud platforms is displayed [41].

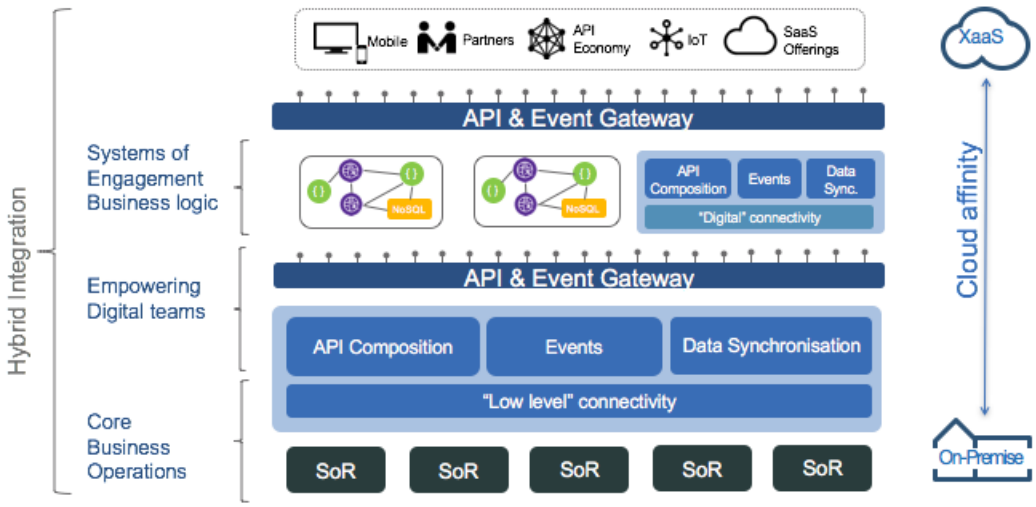


Figure 2.6: High-Level Integration Architecture[41]

- **Core Business Operations** - It is constituted by on-prem systems that offer business operations (systems of record - SoR) that have a lower level of connectivity [42].
- **Empowering Digital Teams** - This layer has APIs that connect to the downstream systems, with an event-based architecture. The APIs that are exposed with the gateway are called System APIs [42].
- **System of Engagement Business Logic** - This layer has microservices or other web apps to support business requirements [42].
- **Top** - Here are the cloud apps that the on-prem systems will connect to.

Event-Based Architecture

An event-based architecture is an architectural pattern where an event, "a change of state or an update"[43], is disseminated when something worthy of note happens to interested parties in the system that will handle and act accordingly [44]. It is an asynchronous process where the one who propagates the event has no knowledge of what happens after it propagates the event, making it an architecture that's inherently loosely coupled [44].
The event has two components: a header and a body. The header contains elements that identify the event and the body illustrates what happened [44]. An event can be processed in different ways, like [44]:

- **Simple Event Processing** - If a notable event is raised, another system downstream initiates an action. In figure 2.7, an example of a simple event processing is shown. A customer places an order via a service that changes the state from available to reserved and then checks if the amount of remaining books is above the threshold defined. If it is below this threshold, an event is raised and published to be consumed by a downstream system that buys more books [44].

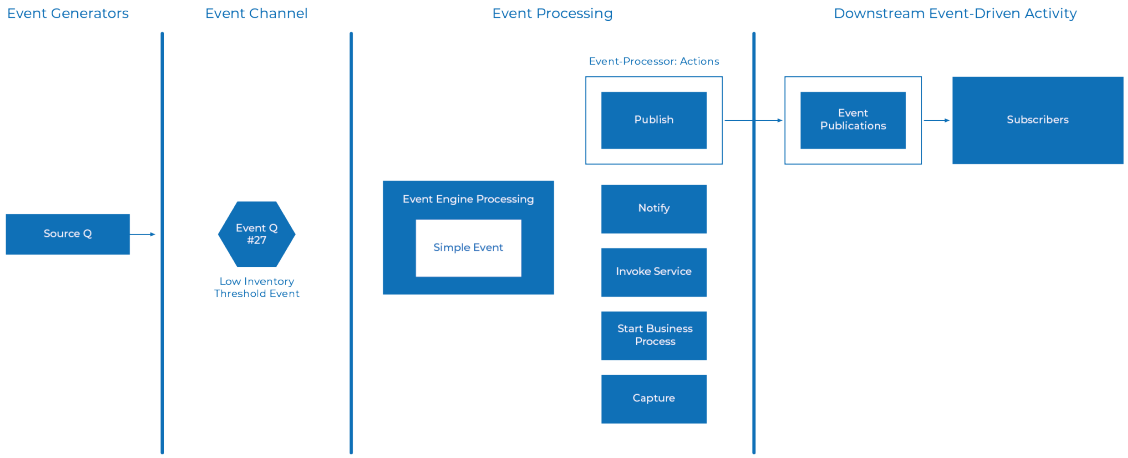


Figure 2.7: Simple Event Processing Example

- **Stream Event Processing** - In this processing style, there are notable and ordinary events. Ordinary events are filtered in order to understand their significance and then passed down to the subscribers, as seen in figure 2.8. Here, multiple events are created from two different sources, and a filter to screen items that are priced below 5000$ is in place. So, the event that is above this price is placed in the event channel to be processed by the simple event processing engine that publishes it. The other flows originate ordinary events and are stored in a data warehouse [44].

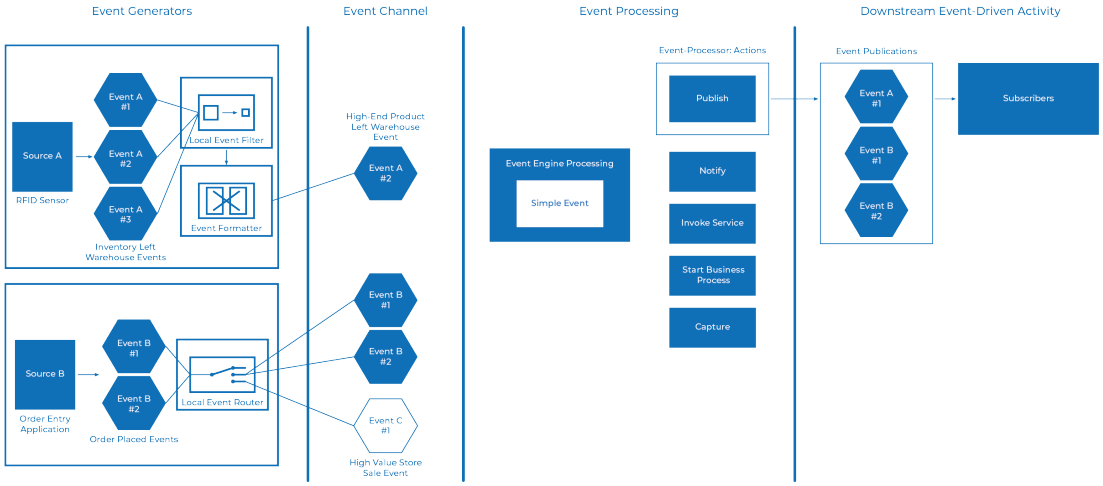


Figure 2.8: Stream Event Processing Flow Example

- **Complex Event Processing** - It is used to evaluate a conjunction of many events of both types - notable or ordinary. In figure 2.9, three complex processing flows are shown, where a system sends heartbeats every 15 minutes to show that is working. If the heartbeat isn't sent, the event processing actions stipulated are initiated and an event is received by the simple event engine and published to be subscribed by another system.
  The other flows are for fraud detection measures. If multiple events are raised by the same customer in a short period of time, an event is published. In the third flow, a suspicious purchase event is published if it is detected that the current purchase digresses more than 50% of other high-value past purchases [44].

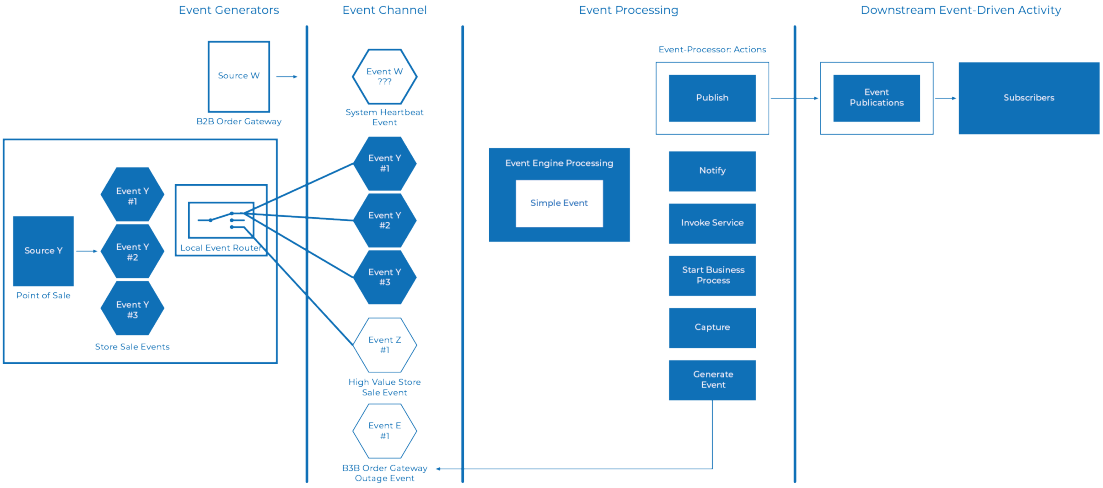


Figure 2.9: Complex Event Processing Flow Example

API-Led Approach

An API-Led approach separates APIs into layers, as seen in figure 2.10. The approach's objective is to promote the reuse of the integration flows inside of the integration solution and by events. This leads to reduced times to release the API and an opportunity to strengthen the security and efficiency by focusing on just one flow [45].

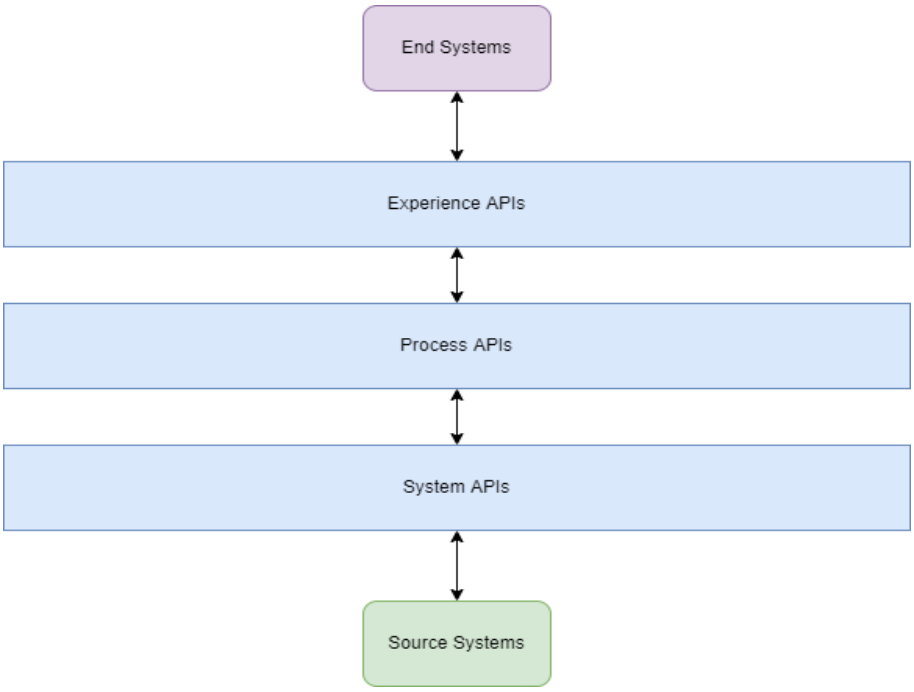


Figure 2.10: API-Led approach

- **System APIs** - They connect directly with the backend systems. They abstract the user from these complex systems and are then reused by process APIs [45].
- **Process APIs** - The business logic is situated in this layer. It links system APIs with experience APIs and transforms the messages to put them in the required format [45].

- **Experience APIs -** It is the entrance point to the solution. It is usually created with an API First approach specifically for the end-user [45].

It is considered an evolution of a Service Oriented Architecture (SOA) approach. It has defined layers with specific business logic and advocates for reusability. It aims to solve what is perceived as the problem with SOA approaches and that is that "heavyweight, top-down implementation approaches" are not suited for what is the modern digital transformation [46]. It defines ways to ensure decentralization of the connection to assets and builds an agile network of applications [46].

2.5 Company's Problem Context

Currently, the brands that were merged into the company use an on-premises approach to deploy and host their APIs. In contrast, SuccessFactors is hosted in the cloud. The diagram 2.11 illustrates this.

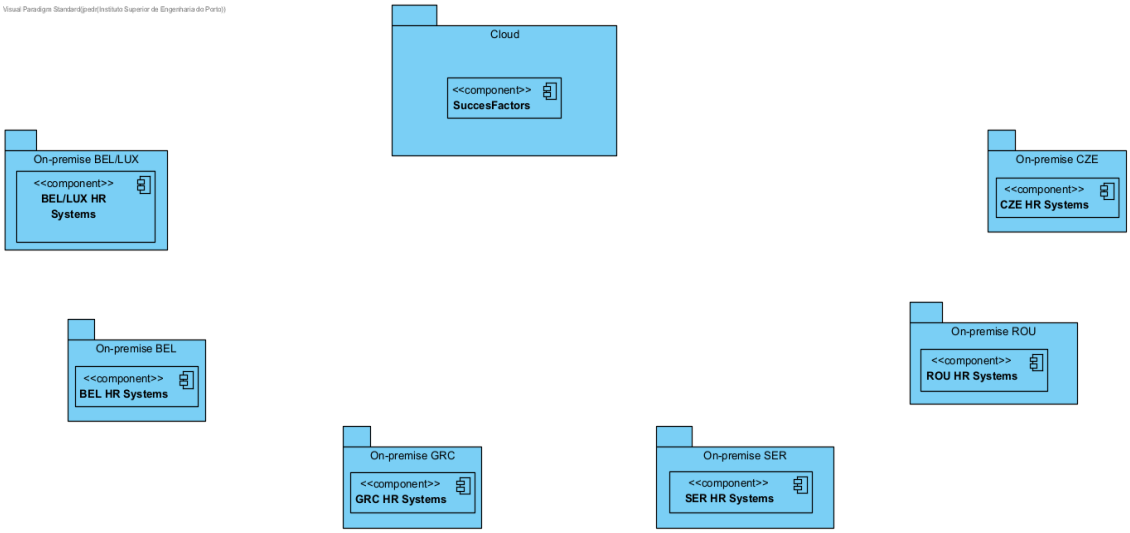


Figure 2.11: API-Led approach

The hybrid integration framework will connect the cloud platform with the on-premise system, two different hosting solutions, hence why it's considered hybrid.

Chapter 3

Value Analysis

Value analysis is a systematic, formal, and organized process of analysis and evaluation of a product or idea to maintain costs down without decreasing quality, while meeting the needs of the customer. The value analysis was done based on the Innovation Process using the New concept development (New Concept Development (NCD)) model[47], AHP, QFD, value of the solution for the customer, perceived value; definition of the value proposition; Business Canvas Model.

3.1 Innovation Process

With the rapidly ever-evolving technology world, new products come up and this, in turn, fuels the exigency and necessity of quality products to satisfy this demand. For product quality to be assured, emphasis on adding value to the organization or to the client is made, by defining a method with various steps that specify this innovation process.
The innovation process may be divided into three sequential stages: the fuzzy front end (Fuzzy Front End (FFE)), the new product development (New Product Development (NPD)) process, and commercialization[47].
The front-end part precedes a more formal and structured process that is the NPD, but the focus is being attributed to the former since it will set the base for the latter parts, albeit it is perceived as an experimental, uncertain, and chaotic phase. To remove the implication that it is mysterious, uncontrollable, and cannot be managed, a new term was coined - Front End Of Innovation". It is one of three stages, being the first one pre-work and it is where new opportunities are found, the second step is the scoping stage where marketing evaluations are made and the third and final step is where a business case is accomplished [47].
To address the lack of standardization in FFE, the NCD model was created. It provides insight and common terminology for the FFE[47].

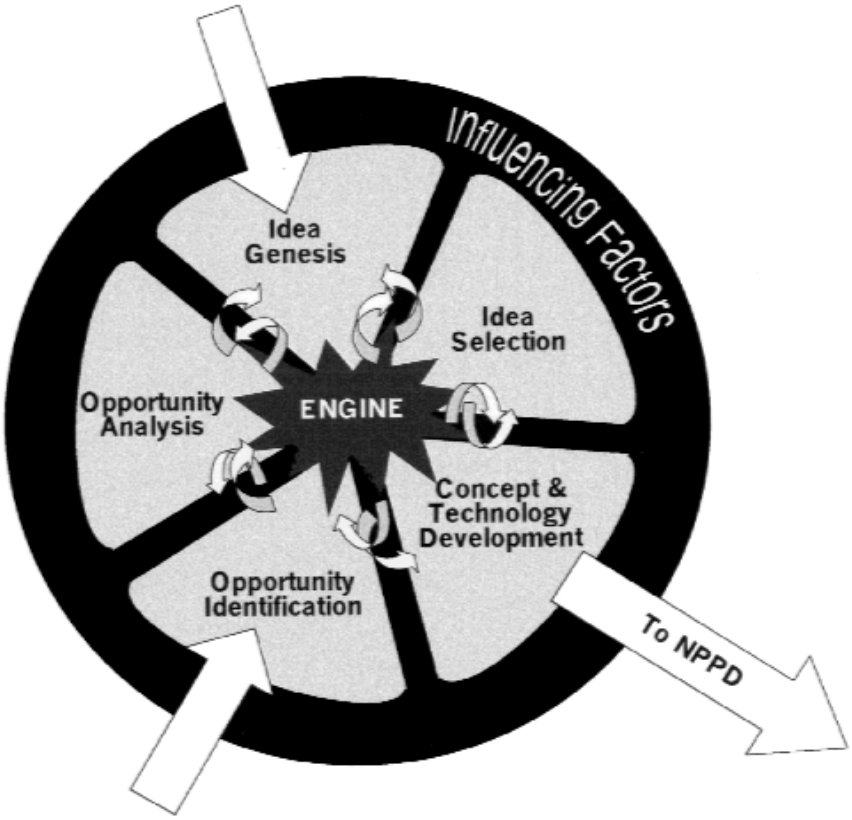


Figure 3.1: The NCD Model

The New Concept Development Model is a holistic framework that identifies "the most effective practices in managing the front end of innovation".
The NCD model is divided into three areas [48]:

- Engine – (center) contains the organizational attributes that drive the process such as vision and strategy.
- Wheel – (Inner) contains the five activities of the Front End: Opportunity Identification, Opportunity Analysis, Idea Generation Enrichment, Ideas Selection, and Concept Definition.
- Rim – (exterior) Includes all the external factors of the surroundings that influence the engine and the wheel's activities such as competitors, customers, trends, or regulations

3.1.1 Opportunity Identification

Opportunity identification is the process of identifying opportunities that typically are relevant to the company to pursue.
With the rising amounts of data creation and consumption, traditional integration approaches have become a bottleneck because they can no longer keep up with the amount of data requests and responses made[49]. With the rise of digitalization, the IDC predicts that, by 2025, the global data sphere will be around 175 zettabytes[26].

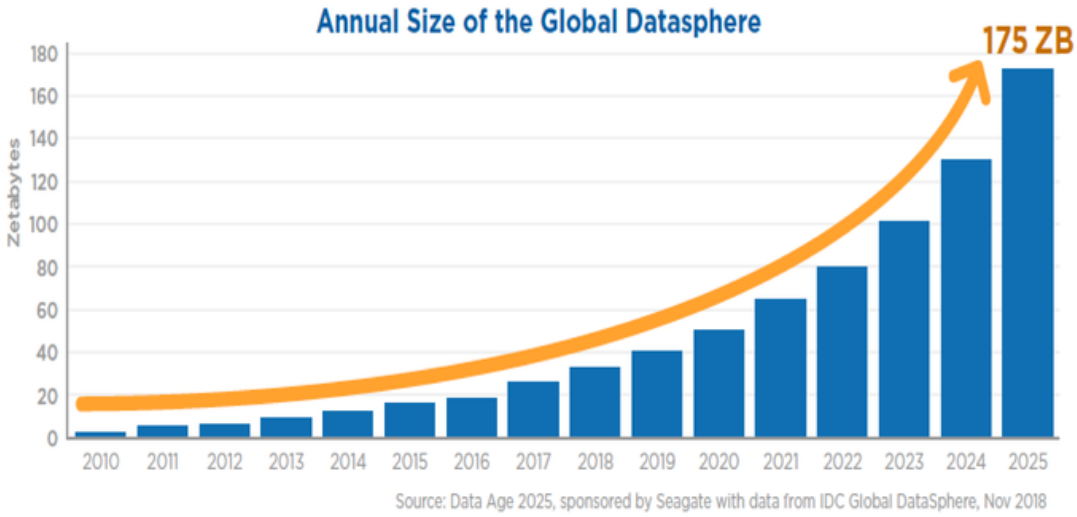


Figure 3.2: Datasphere size through the years[26]

Gartner predicts that, by 2022, 65% of large companies will have a Hybrid Integration Platform [40].
With more than 300 thousand employees, traditional integration systems are going to struggle and the transition from on-prem to the cloud can be time-consuming, expensive, and arduous. Not only that but, by inheriting several legacy HR systems in various countries, with systems both on-prem and in-cloud, an integration layer that integrates seamlessly on-prem systems, cloud applications, and existing API layers would be preferable since re-engineering the existing solutions wouldn't be the sensible approach, as they hold inherent value to the business [3].

3.1.2 Opportunity Analysis

In the opportunity analysis, the opportunity identified in the Opportunity Identification stage is analyzed and established to see if its development would be advantageous or not.
Due to the rise of cloud platforms and cloud integration, hybrid integration has too been rising in recent years. According to a Mordor Intelligence study[50], the Hybrid Integration Platform Market was valued at 22.56 billion dollars in 2020, with a predicted value of 44.56 billion dollars in 2026.

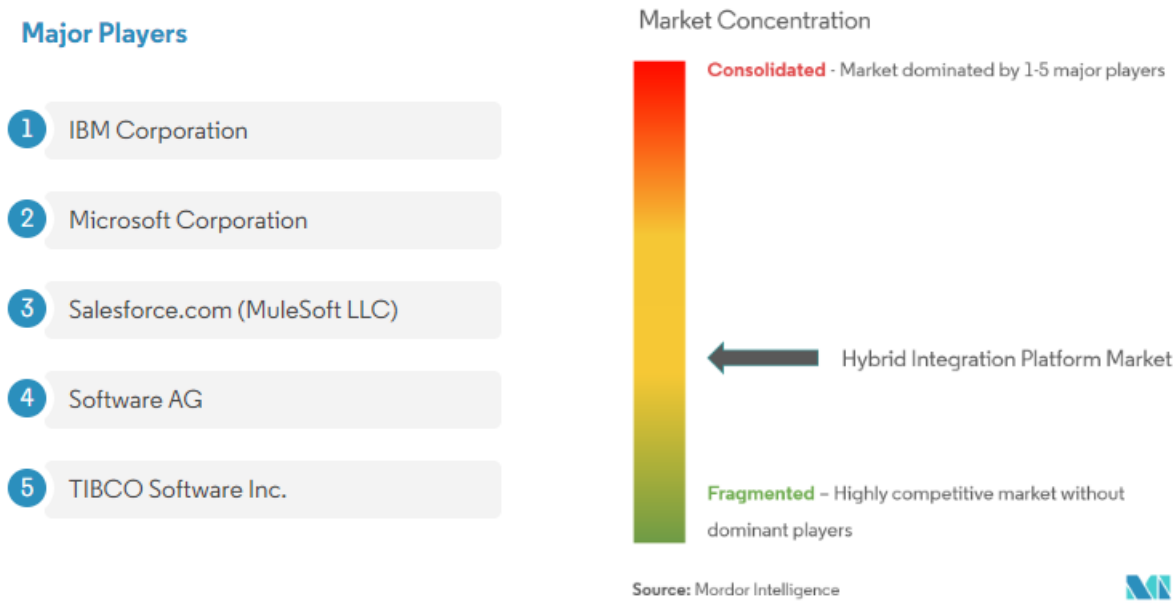


Figure 3.3: Market Share

Asia-Pacific is the Fastest Growing Region in the Global Hybrid Integration Platform Market, where 82% of organisations in Singapore have embraced hybrid cloud integration.

3.1.3 Idea Genesis

The element of idea generation and enrichment concerns the birth, development, and maturation of a concrete idea. Ideas suffer a lot of transformations during the NCD stages until it is completely consolidated. Ideas for possible solutions to this problem must be generated and narrowed down.
The ideas that will be conjured up must provide answers to the questions that follow:

- How can cloud and on-prem solutions be linked?
- How can the new solution decouple and efficiently connect the applications?
- How can the solution be reusable, monitored, safe, and easy to integrate with other systems?
- How can the solution solve the existing cloud platform problems?

The ideas that were generated were:

- Develop a custom API Framework
- Use an already existing Hybrid Integration solution on the market
- Migrate all systems On-prem

3.1.4 Idea Selection

The hard part about solving a problem is not generating ideas but choosing which ideas one should work towards.
Ideas must be allowed to mature even with reduced confidence[47], but to select the idea that will become the final solution, the questions in section 3.1.3 must be answered convincingly. The ideas were analyzed and their potential was assessed based on this and the ultimate idea was chosen, using the AHP method, created by Thomas L. Saaty.
The integration of cloud and existing on-prem solutions is of the utmost importance here, because the client wants to use the solutions already existent in the countries, to not disrupt the workflow. The second and fourth questions are also important since the whole point of the system is to remove the negative impact that this surge in employees created in the system and to remove the current limitations that the client feels with the existing cloud platform. Lastly, a quality framework is required because this is the heart and soul of the HR department and impacts almost half a million employees.
Next, the AHP method, created and proposed by L. Saaty [51], will be used to select one of the ideas identified in section 3.1.3, that will be considered as alternatives.

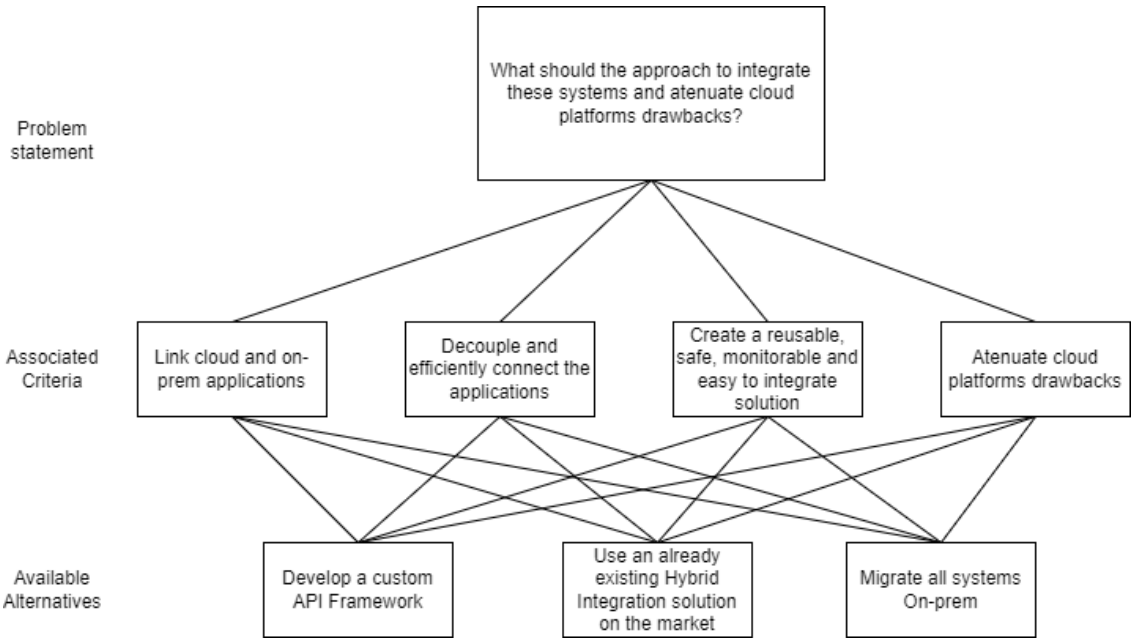


Figure 3.4: AHP Hierarchy Structure

After this exercise is done, the second phase of the AHP method begins, where a matrix to compare the criteria must be created. These criteria are going to be given values according to their importance, using the Saaty scale [51].

Importance Level	Definition	Description
1	Equal Importance	The two activities contribute equally to the objective
3	Weak importance	One activity is slightly more important than the other
5	Strong importance	One activity is strongly more important than the other
7	Demonstrated importance	One activity is strongly more important than the other and it is demonstrated in practice
9	Absolute Importance	The evidence favors one activity with the highest possible order of affirmation
2,4,6,8	Intermediate values	When a compromise is needed

Table 3.1: Saaty table

To create the matrix, these names will be taken into consideration:

- A: Link cloud and on-prem applications
- B: Decouple and efficiently connect the applications
- C: Create a reusable, safe, monitorable, and easy to integrate solution
- D: Circumvent cloud platforms limitations

	A	**B**	**C**	**D**
A	1	5	5	2
B	0,2	1	2	0,33
C	0,2	0,5	1	0,25
D	0,5	3	4	1

Table 3.2: Comparison Matrix

This matrix shows that criteria A - Link cloud and on-prem applications is the most important of them all, although being just slightly more important than criteria D - Circumvent cloud platforms limitations. Further analyzing the matrix, the latter is also more important than Decoupling and efficiently connect the applications and Creating a reusable, safe, monitorable, and easy to integrate solution, B and C respectively. At last, criteria B is slightly more important than criteria C.
The third AHP phase is done by calculating the relative priority of each individual criterion and this is made possible by normalizing the previous matrix and by calculating the average of the values of each line of the normalized matrix. This normalization is undertaken by dividing each value by the sum of its column.

	A	**B**	**C**	**D**	**Relative Priority**
A	0,5263	0,5263	0,4167	0,5581	0,506859445
B	0,1053	0,1053	0,1667	0,0930	0,11755406
C	0,1053	0,0526	0,0833	0,0698	0,077748878
D	0,2632	0,3158	0,3333	0,2791	0,297837617

Table 3.3: Normalized matrix and relative priority

The fourth stage is carried out by calculating the CR (Consistency Reason) to ensure that the consistency of the relative priorities is good. A value under 0.1 means that the relative priorities are consistent.

$$CR = \frac{IC}{IR}$$

The IR is selected by the values in table 3.2. Since the matrix's dimension is 4, the IR value is 0.9.

1	2	3	4
0,00	0,00	0,58	0,9

Table 3.4: IR Value Table

The IC value, on the other hand, is calculated using the following formula:

$$IC = \frac{\lambda_{max} - n}{n-1}$$

By multiplying the matrix in table 3.2 with the priority vector in 3.3, we get the following matrix:

A	2,0790
B	0,4737
C	0,3124
D	1,2149

Table 3.5: Priority vector matrix

λ_{max} is calculated by calculating the average of the matrix 3.3 divided by the priority vector.

$$\lambda_{max} = 4,057038182$$

With the λ_{max} calculated, the IC value can be calculated, being n = 4:

$$IC = \frac{4,057038182-4}{4-1} \; IC = 0,019012727$$

Dividing the IC value by the IR value for a table of size 4, the CR value is 0,02, which is below 0,1, so the relative priorities are consistent.

In the fifth stage, comparison matrices for each individual criteria are created to measure the importance of the alternatives.
To create the matrices as done before, these names will be taken into consideration:

- X: Develop a custom API Framework
- Y: Use an already existing Hybrid integration solution on the market
- Z: Migrate all systems On-prem

The matrices and the priority vectors were all calculated as done above. For the sake of brevity, only the comparison matrices, and their priority vectors will be shown.

	X	Y	Z
X	1	5	6
Y	0,2	1	3
Z	0,167	0,33	1

Table 3.6: Comparison matrix for Criteria A

	X	Y	Z
X	1	4	7
Y	0,25	1	4
Z	0,14	0,25	1

Table 3.7: Comparison matrix for Criteria B

	X	Y	Z
X	1	7	3
Y	0,14	1	0,2
Z	0,33	5	1

Table 3.8: Comparison matrix for Criteria C

	X	Y	Z
X	1	7	5
Y	0,14	1	0,2
Z	0,2	5	1

Table 3.9: Comparison matrix for Criteria D

By analyzing these matrices, we can clearly see that alternative X - Develop a custom API Framework is the strongest one by far for all of the criteria. Alternative Y - Use an already existing Hybrid integration solution on the market and Z - Migrate all systems On-prem both take two criteria against one another.

In the sixth stage, we calculate the composed priorities of the alternatives. This is done by multiplying the priority matrix that is created by merging the priory vectors of each of the above matrices with the relative priority vector in table 3.3. The result is:

	A	B	C	D
X	0,53029525	0,515796703	0,482541652	0,522398501
Y	0,15105905	0,175824176	0,05532908	0,05389103
Z	0,0686457	0,058379121	0,212129269	0,173710469

Table 3.10: Composed priority matrix

By evaluating the matrix 3.11, the alternative to be chosen is to Develop a custom API Framework.

X	0,522526149
Y	0,117587102
Z	0,109886749

Table 3.11: Result matrix

3.2 Functional Analysis

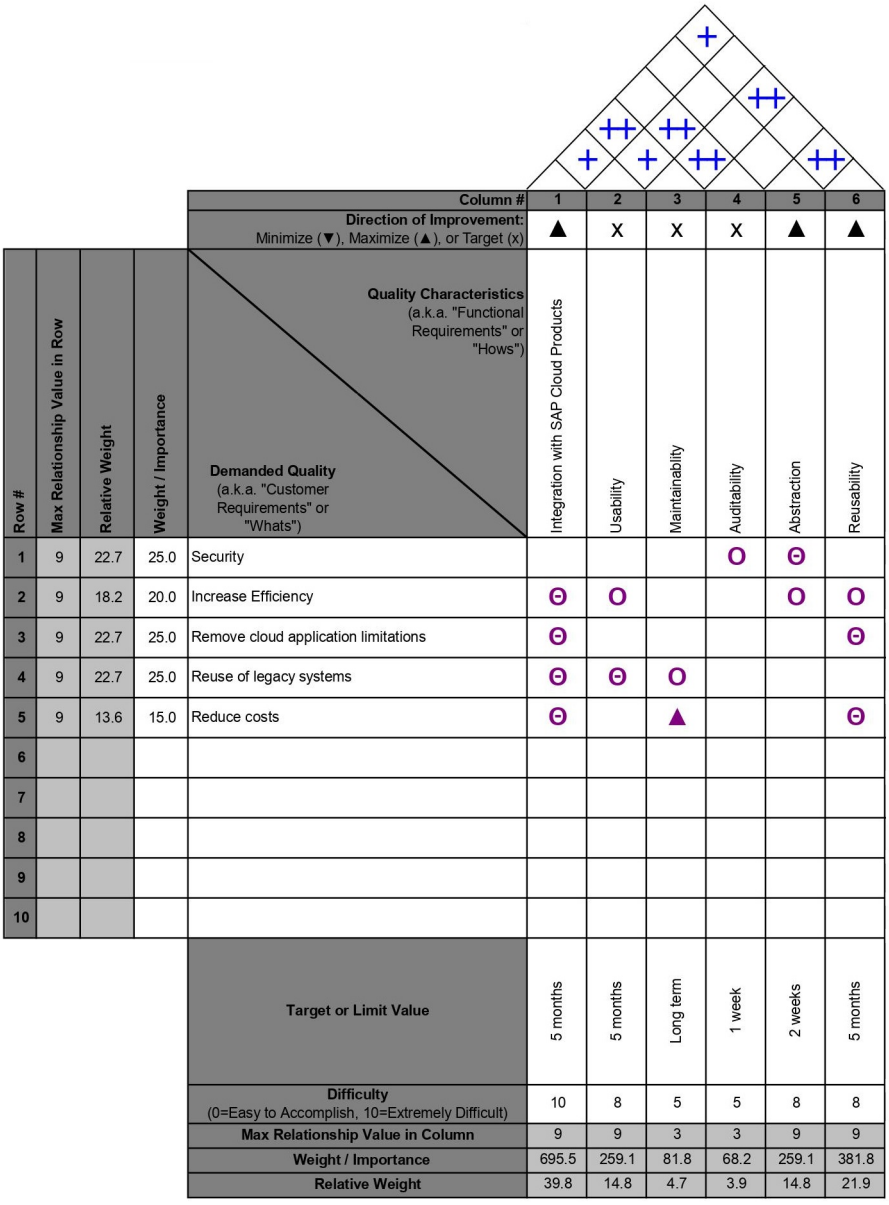
Originated in the late 60s in Japan, QFD (Quality Function Deployment) is a proposal to define requirements and transform them into something of use for the development process [52] [53]. To better illustrate this, a house of quality [54] was used, as shown in figure 3.5. The following customer requirements and their respective weights were identified:

- Security - 20
- Increase Efficiency - 20
- Remove cloud application limitations - 25
- Reuse of legacy systems - 25
- Reduce costs - 10

As for functional requirements, the following were identified:

- Integration with SAP Cloud Products
- Usability
- Maintainability
- Auditability
- Abstraction
- Reusability

Since this is a custom solution for specific client needs, this house of quality doesn't have competitive analysis since there are no real competitor solutions.



Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats") \ Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")	Integration with SAP Cloud Products	Usability	Maintainability	Auditability	Abstraction	Reusability
				Column #	1	2	3	4	5	6
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	▲	x	x	x	▲	▲
1	9	22.7	25.0	Security				O	Θ	
2	9	18.2	20.0	Increase Efficiency	Θ	O			O	O
3	9	22.7	25.0	Remove cloud application limitations	Θ					Θ
4	9	22.7	25.0	Reuse of legacy systems	Θ	Θ	O			
5	9	13.6	15.0	Reduce costs	Θ		▲			Θ
6										
7										
8										
9										
10										
				Target or Limit Value	5 months	5 months	Long term	1 week	2 weeks	5 months
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	10	8	5	5	8	8
				Max Relationship Value in Column	9	9	3	3	9	9
				Weight / Importance	695.5	259.1	81.8	68.2	259.1	381.8
				Relative Weight	39.8	14.8	4.7	3.9	14.8	21.9

Figure 3.5: House of Quality

3.3 Value

Value can be perceived as a need, desire, interest, standard /criteria, beliefs, attitudes, and preferences[55]. But value differs depending on the person or the situation.
Zeithmal in 1988 [56], suggested that “perceived value is the consumer’s overall assessment of the utility of a product based on perceptions of what is received and what is given”. Different customers can perceive value about the same product or service differently[57]. If what is left after the combination of sacrifices and benefits given to the client is positive, then the solution is of value to the customer and that’s customer value.
This solution provides value to the customer by integrating all hr systems inside of the same environment, syncing the various inherited hr systems with the new systems, in a fast and

clean solution that removes the issues of traditional integrations and the cost and difficulty of migrating to a cloud solution.

Benefits	**Sacrifices**
Integration of the new System with legacy systems, reusing them	Cost of maintaining systems and implementation
All company employee data comes from a single source	Cost of the cloud platform and its drawbacks

Table 3.12: Benefits and Sacrifices

3.3.1 Value Proposition Canvas

Value Proposition Canvas is an "overall view of a company's bundle of products and services that are of value to the customer."[58]. It presents the product or service, what it aims to solve and what benefits the client should expect from it. To describe the value for the customer, a tool called Value Proposition Canvas [59], shown below, was created.

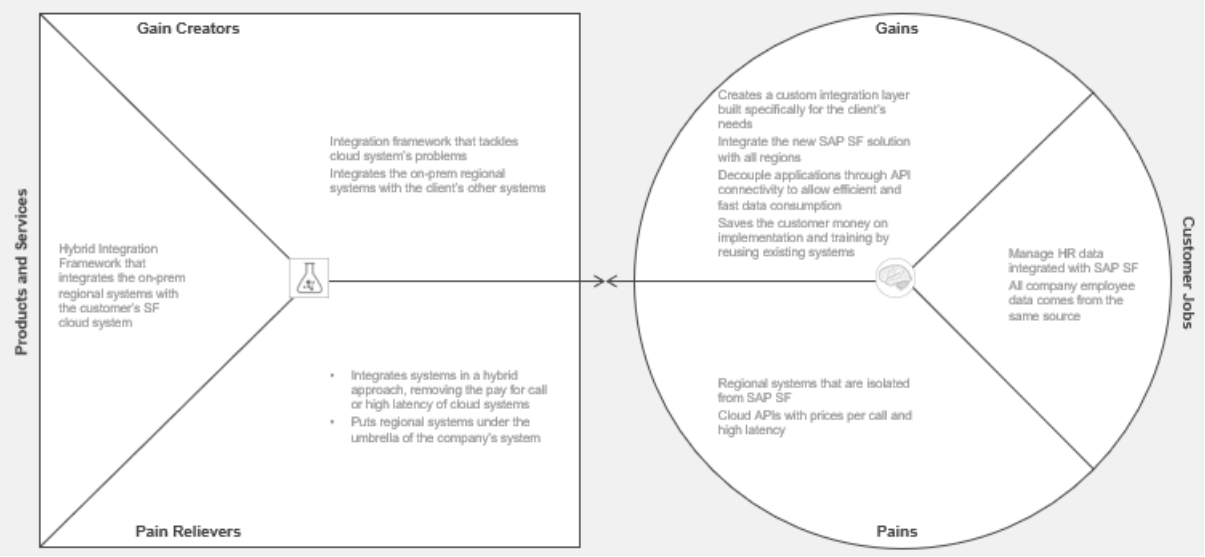


Figure 3.6: Value Proposition Canvas

Chapter 4

Analysis and Design

The current chapter describes and demonstrates an architectural proposal for the hybrid integration framework and the identified integration use cases.

4.1 Analysis of current architecture

This section presents the architectural design pattern to be used in this system and alternatives.
Architectural design is an abstract representation of the system that identifies the subsystems and applications that make up the system. It is done in the earlier stages of the project to describe the software architecture.[60]

4.1.1 Current Landscape

Currently, the HR systems for each country aren't connected to the centralized HR solution and are still using their isolated solution with their own data format and their mapping formats. This HR data from each system is being migrated to the new centralized solution that is SuccessFactors as part of the overall project, but it is not the subject nor the problem of this dissertation.

Belgium/Luxembourg

This region currently has a central repository in the cloud, with the employee data that feeds a Human Capital Management (HCM) System that then propagates the data to a Retail System and an orchestrator, that controls the flow with a database.

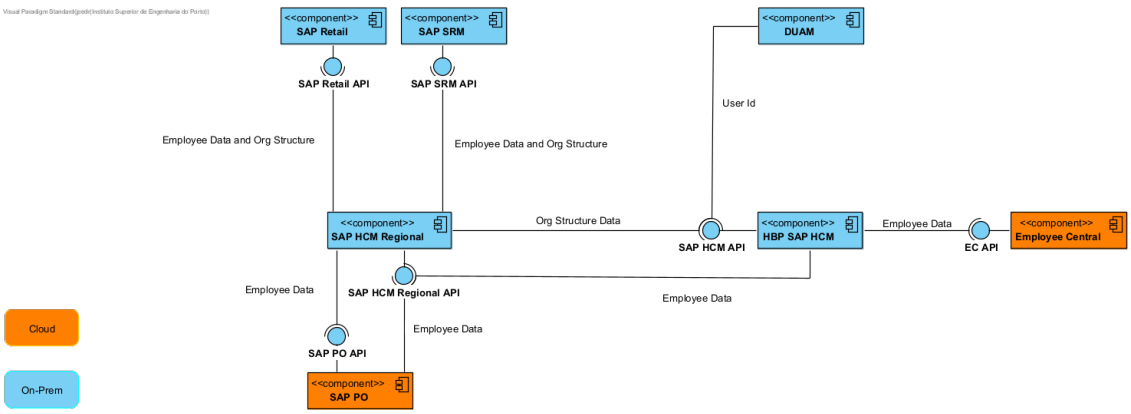


Figure 4.1: BEL/LUX Architecture As-Is

Czech Republic

Czech Republic controls the data flow with a middleware that links the systems. They currently have an HR system and an Expenses System situated on the cloud and databases that will need Employee and Organisational Structure Data.

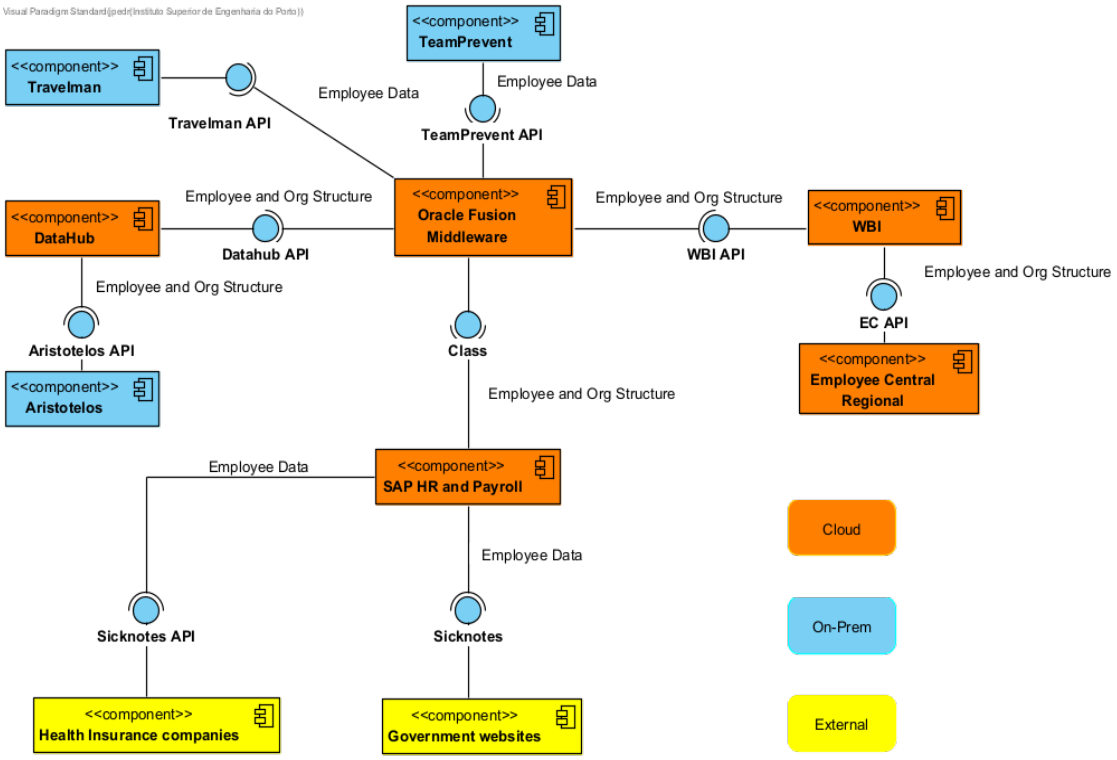


Figure 4.2: CZE Architecture As-Is

Greece

Similar to Czech Republic, Greece uses an orchestrator to control the data flow between the Retail and HR system that they possess, all on-prem.

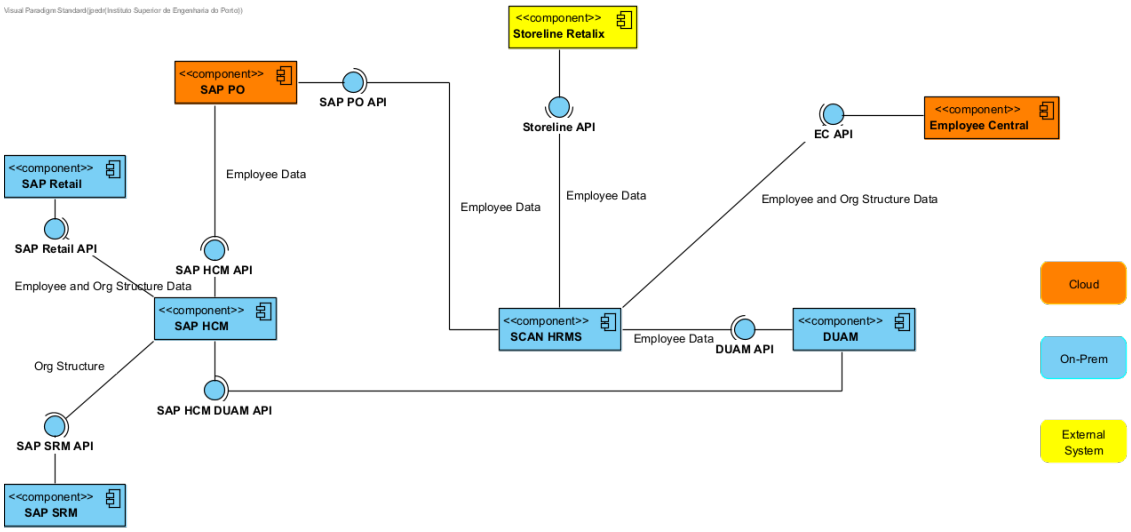


Figure 4.3: GRE Architecture As-Is

Romania

Similar to Czech Republic, Greece uses an orchestrator to control the data flow between the Retail and HR system that they possess, all on-prem.

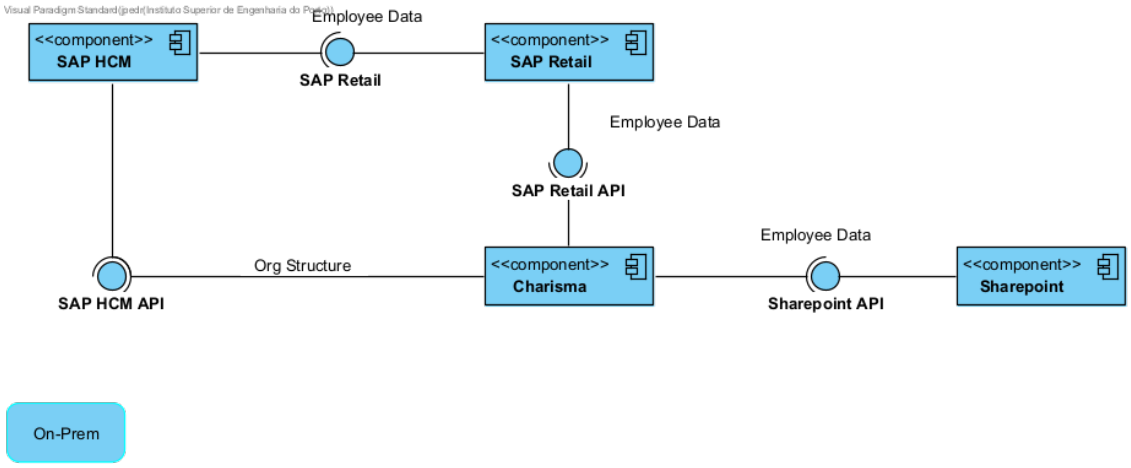


Figure 4.4: ROU Architecture As-Is

Serbia

Serbia has a cloud repository that feeds an HCM system. This system then propagates data to a Retail System.

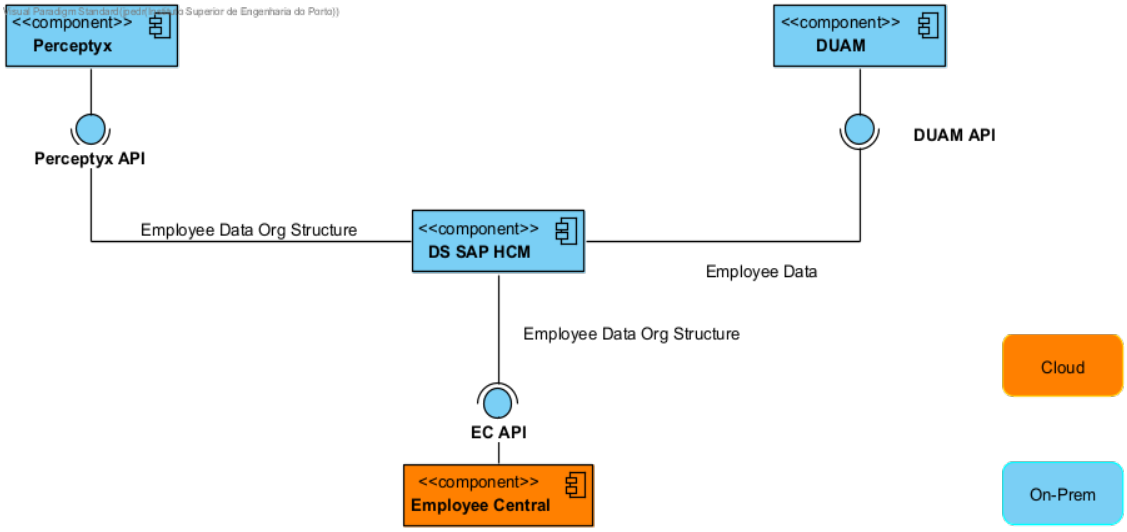


Figure 4.5: SER Architecture As-Is

4.2 Requirements

IEEE defines requirement as a "condition or capability needed by a user to solve a problem or achieve an objective" [61].
For this project, the functional requirements are as follows:

- Employee Data to HR systems
- UserID to HR Systems
- Organisational Data to HR systems
- Job Requisition data to HR Systems
- Job Requisition data to GetNoticed
- Application Data to HR Systems
- Application Data to GetNoticed
- Application Data from HR Systems to SF
- Application Data from GetNoticed to SF
- Candidate Data to HR Systems
- Candidate Data to GetNoticed
- Candidate Data from HR Systems to SF
- Candidate Data from GetNoticed to SF
- Sicknotes to ECP

- APIs need to produce logs
- APIs need to have error handling
- APIs need to support authentication

The non-functional requirements are:

- Reliability
  - The APIs need to be well-documented and well-versioned.
  - Good software practices need to be used in order for the maintenance teams to better comprehend the code.
- Scalability
  - APIs that will have high rates of requests need to have asynchronous methods.
  - The APIs need to be efficient.
  - Cloud deployment will benefit scalability.
- Security
  - All APIs will use the Hypertext Transfer Protocol Secure (HTTPS) protocol and require authentication.

4.3 Solution Design

The objective of this dissertation is to design an API-Led integration layer implemented in MuleSoft and deployed in Microsoft AKS in containers, that connects the new SuccessFactors central HR system to the legacy systems located in each country, based on the following factors:

- Secure
- Auditable
- Monitorable
- Allows decoupling of applications and efficient data consumption

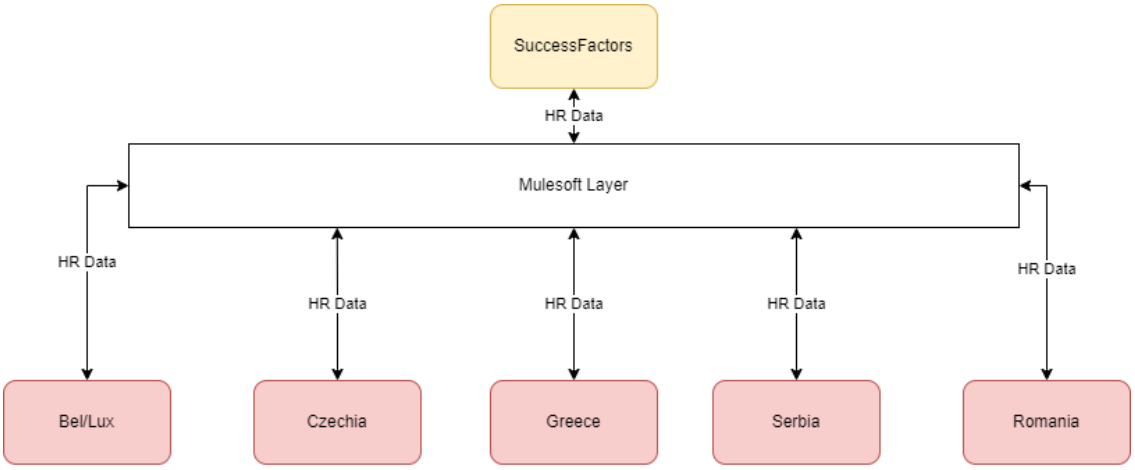


Figure 4.6: To-Be Landscape

4.3.1 Cloud Deployment

The cloud provides on-demand access and the capacity to grow the infrastructure based on real requirements. A software project consists of a large number of distinct executables, each having unique platform and library requirements. They need the deployment and configuration of various components before and during execution. It is vital that these deployment and setup processes be repeatable, otherwise, establishing processes soon loses their utility [62].
Tools, such as Docker swarm and Kubernetes, have emerged to manage the deployment and setup of containers in the cloud. This design, containerization, orchestration, and public cloud computing resources combination has drastically decreased deployment times. Nevertheless, contemporary deployments are very dynamic and adaptable. Regular and quick modifications to the code, and creates downtime when they're applied. Zero-downtime deployments should not result in service interruptions for end users. The old version will continue to operate until the new one is available [63].
In this project, the deployments and hosting of the APIs using Microsoft AKS, as other iterations of the project were running on-prem, being this the pioneer project running on the cloud.
The cloud architecture of this project is pictured in 4.7.

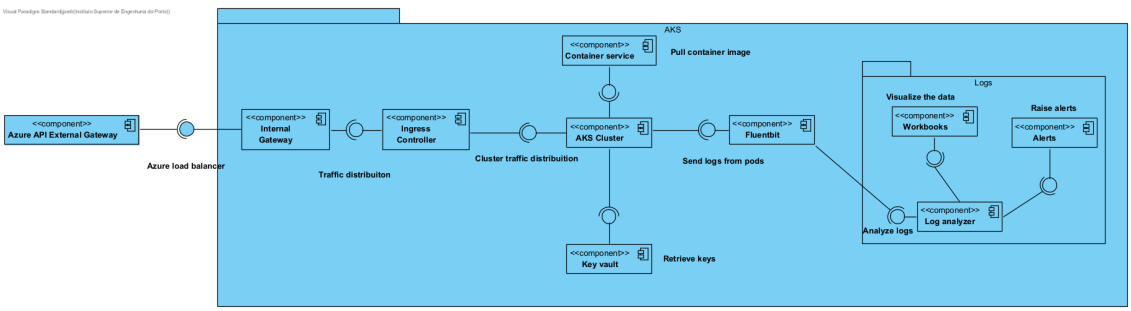


Figure 4.7: Cloud Architecture

- **External Application Gateway** - The entry point to the APIs. It's only applied to the outbound client requests.

- **Internal Application Gateway -** An internal gateway is used for inbound and outbound requests. For the internal routing configuration, nginx is used.
- **AKS (Azure Kubernetes Service) -** AKS is the container orchestrator platform
- **Fluentbit -** Forwards all log data from the containers to the log analyzer
- **Key Vault -** Stores all secure values that otherwise would be defined in the configuration files on an API level. The API fetches these values during the build phase.
- **ACR (Azure Container Service) -** Where software images are stored.
- **Log Analyzer -** Tool to analyze the logs returned from the containers.
- **Workbooks -** Visualization tool for the logs for easier comprehension and analysis.
- **Alerts -** Tool that triggers emails, and messages or creates tickets in other services. It charges per alert.

4.3.2 Integrations

Based on the concepts presented above, the solution will follow the subsequent architecture, presented in diagram 4.8.

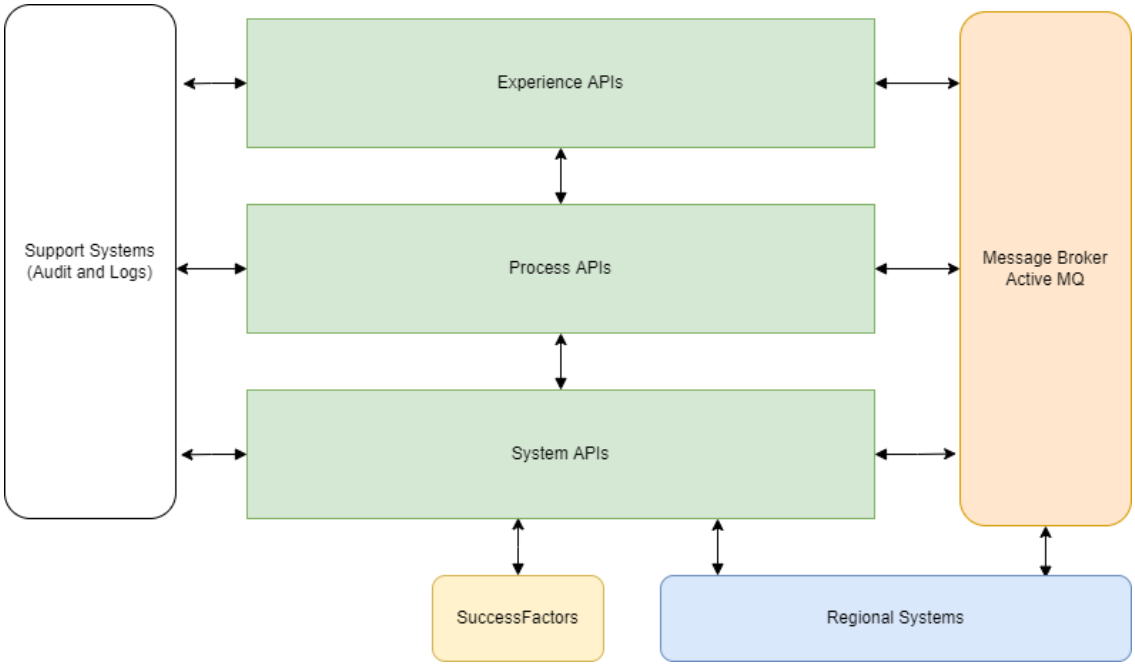


Figure 4.8: System Architecture

Each layer has a purpose, as explained in section 2.4 of chapter 2. The support systems, present in diagram 4.8 will be used by the APIs in these layers while the Message Broker will be used by the regional systems as well as the APIs - the APIs publish an event to then be consumed by the regional systems.

Belgium/Luxembourg

SuccessFactors will connect to the regional systems as presented in Appendix A.
Recruitment API and GetNoticed API will handle data from SuccessFactors Recruitment and On-boarding. Persons API, Departments API, Positions API, Jobs API will handle Employee, Departments, Positions, and Job data from SuccessFactors. SAP Retail Process API will replicate the SAP Product that integrated the regional SF instance with SAP Retail. The Scheduler will trigger HR Synch Publisher API on an hourly basis in order to synchronize the data in the downstream systems with SF and to not exceed the rate of requests that SF has. ITIM API will handle the UserID data stored in the Identity Manager System.

Czech Republic

The to-be landscape of the Czech Republic region will follow the architecture presented in diagram 4.9.
Recruitment API API will handle data from SuccessFactors Recruitment and Onboarding. Persons API, Departments API, Positions API, Jobs API will handle Employee, Departments, Positions, and Job data from SuccessFactors. CSSZ Process API will replicate the SAP Product that integrated the regional SF instance with the czech social security website to retrieve employee sicknotes. Coupa Persons API will handle the data from Coupa, a travel expenses system. The Scheduler will trigger HR Synch Publisher API on an hourly basis in order to synchronize the data in the downstream systems with SF and to not exceed the rate of requests that SF has. ITIM API will handle the UserID data stored in the Identity Manager System.

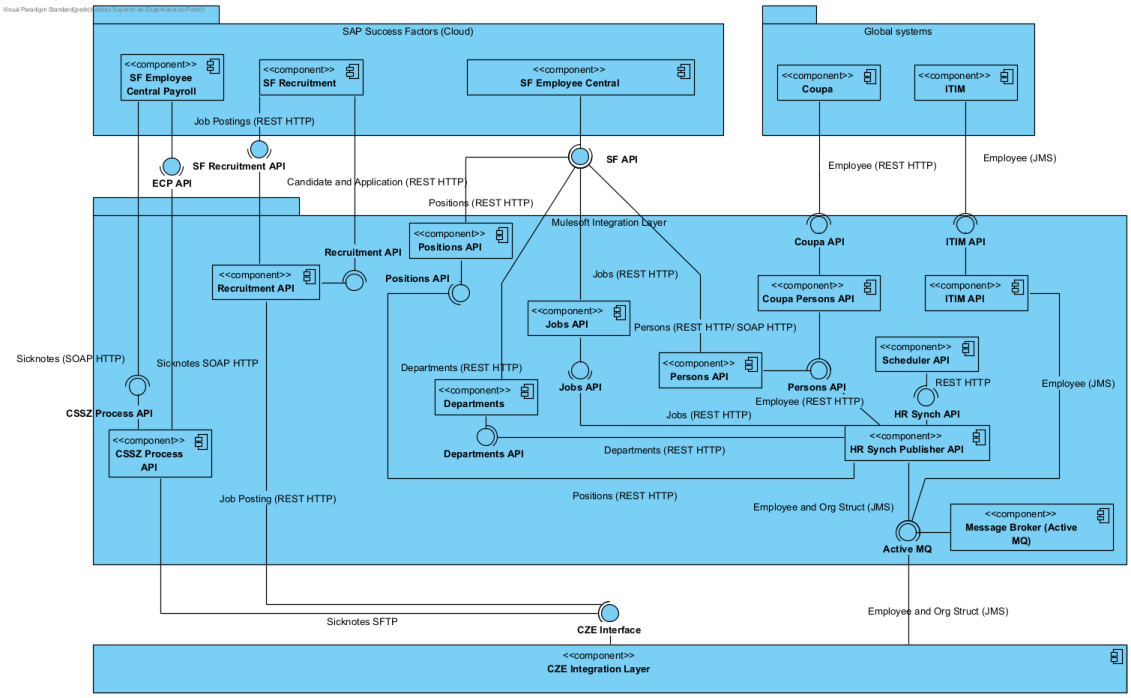


Figure 4.9: To-Be Landscape CZE

Greece

For this country, an approach similar to Belgium/Luxembourg will be used. The only difference is that a new API will be used to receive a trigger from SuccessFactors when an involuntary termination is issued. This employee update has to be made in real-time, while most data will be updated hourly.

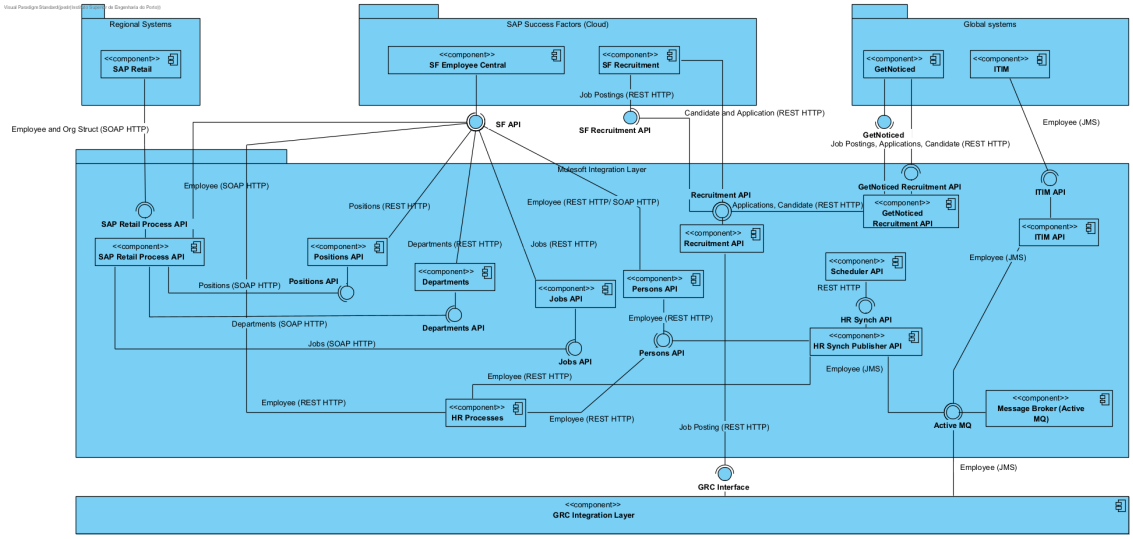


Figure 4.10: To-Be Landscape GRC

Romania

Similar to Belgium/Luxembourg, the following architecture will be used in the Romania region 4.11

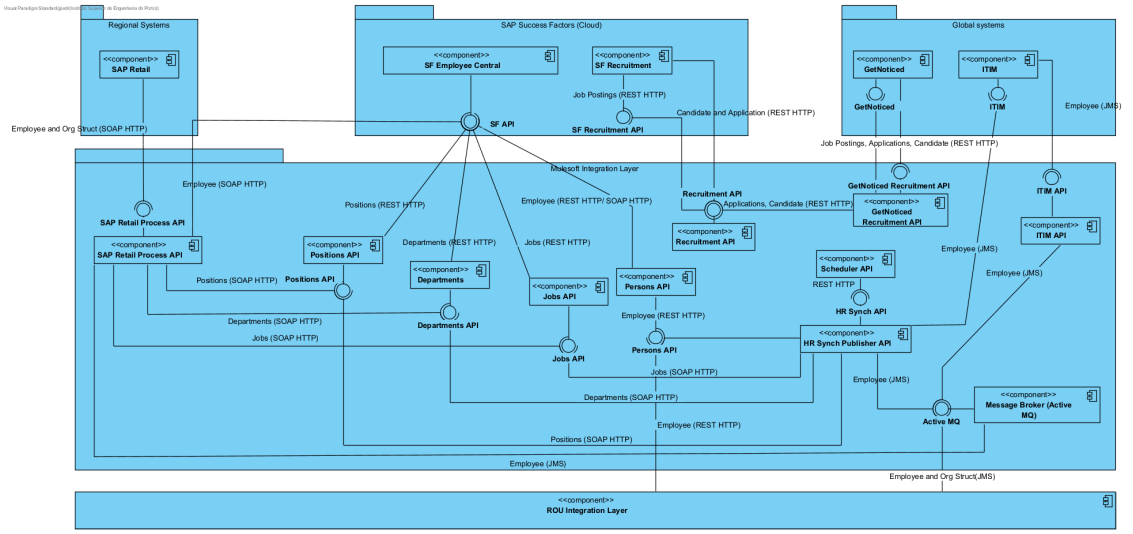


Figure 4.11: To-Be Landscape ROU

Serbia

For Serbia, the following architecture will be used.

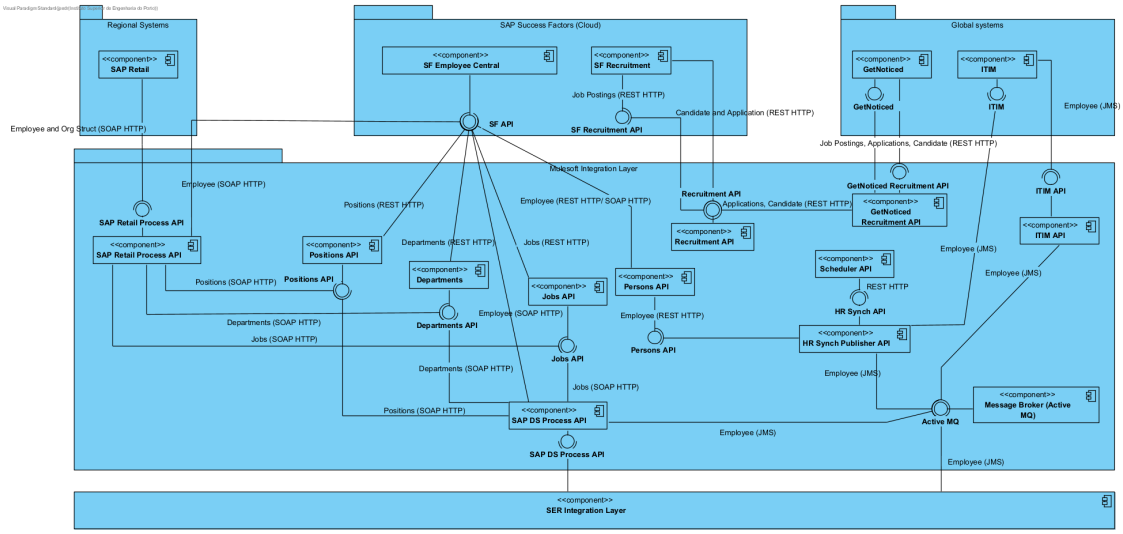


Figure 4.12: To-Be Landscape SER

Belgium

For Belgium, the following architecture will be used. For this integration, an API to cross-reference data and translate it from the Belgium systems to Successfactors will be used.

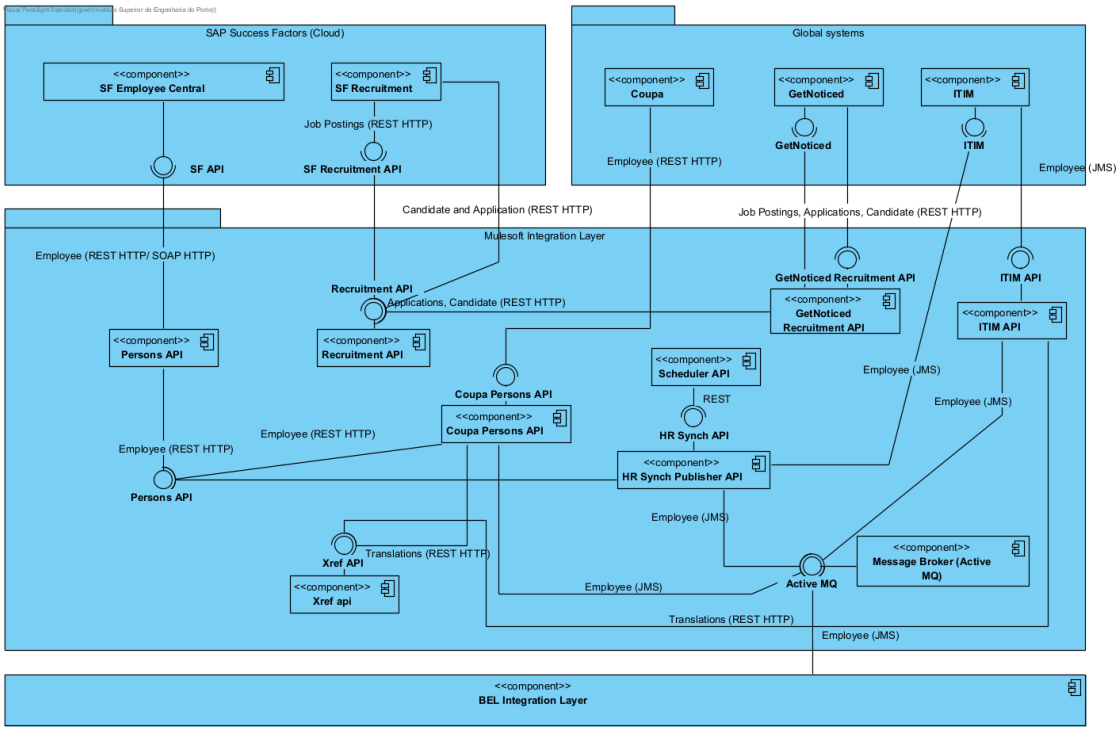


Figure 4.13: To-Be Landscape BEL

Synchronization

The developed integration layer will also have a basis on data synchronization, using two patterns: near-real-time synchronization and file-based batch.

Near-Real-Time Synchronization

With such a large dataset and with so many requests being made, a real-time approach wouldn't warrant consistency and availability, since SuccessFactors limits the rate of requests [64]. An event-driven approach with a publisher/subscriber model was chosen, with hourly requests to Successfactors (deltas), using ActiveMQ as a message broker[44].

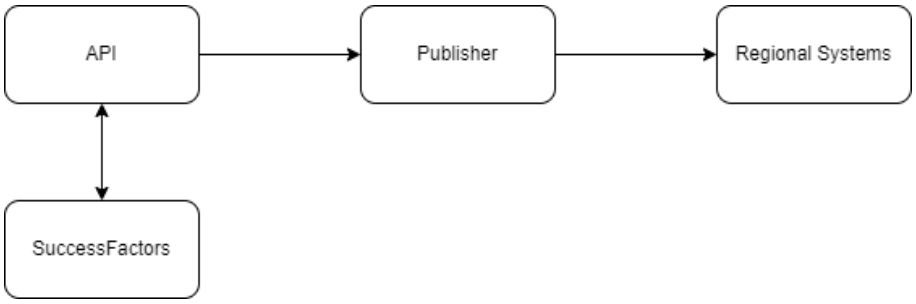


Figure 4.14: Near-Real Time Synchronization

This approach will provide the ability to:

- Support each system's Quality of Service (QoS) by configuring the message broker
- Filter messages by the message broker

Real-Time Synchronization

In a particular case for Greece, a real time approach was needed for involuntary terminations. When an involuntary termination is issued, SF triggers an API that updates the employee's status so all access is denied.

File-based Batch Synchronization

Certain systems may not need a data delivery in a near-real-time approach and, since legacy systems are being reused, the final system may not have an exposed interface, needing a file-based approach [65].

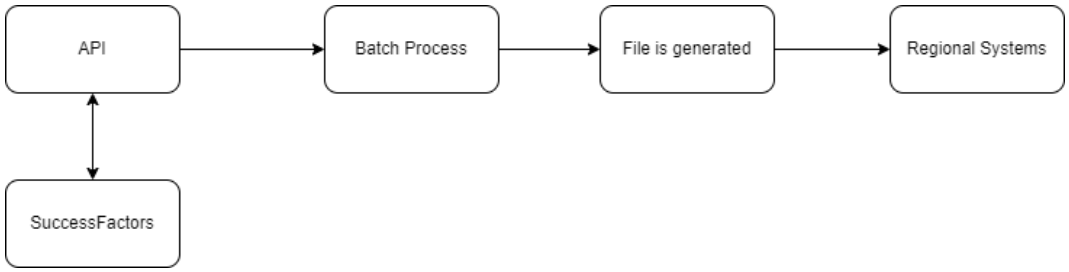


Figure 4.15: File-Based Batch Synchronization

Alternatives

Kafka

Kafka is another open-source publish/subscribe system used for event-based architectures. Although they are similar, they have their architectural differences. ActiveMQ routes messages that end up on a queue for consumers to consume. Kafka, on the other hand, publishes messages to a topic-specific append log [66].
It uses batching to improve throughput outstandingly and it doesn't wait for acknowledgements from the message brokers, giving the responsibility to the consumer [66].
Ultimately, ActiveMQ was chosen because the system is following a near-real-time approach and it can't abide for data loss. It is also the technology that was used in the previous releases of the project for other regions, and making this solution homogeneous across all regions would be advantageous.

Deploy APIs On-Prem

In previous iterations of the project, APIs were deployed on-premise with Openshift, a docker container "orchestration platform optimized for web applications" [67].
The decision to deploy on-premise was made because the client had the licensing, the know-how and the equipment to use the Openshift platform efficiently on-premise.
The biggest issue with this approach is that it lacks the needed elasticity and scalability that cloud deployments offer.
A decision was made by the client to fully deploy this project on Azure.

Use other integration platforms

Other integration platforms could be used to develop and maintain the APIs. WSO2, Apian, and TIBCO are technologies that Deloitte uses in other projects. This would mean that all APIs that are created in MuleSoft would be developed utilizing other tools.
Ultimately, since Deloitte is a partner of MuleSoft and the client supported the decision to use MuleSoft, other tools were discarded.

4.3.3 Integration Use Cases

In this section, the identified integration use cases for each country connected by this integration layer will be presented.
For all the countries, two use cases in common were identified: employee data and organisational structure data to the downstream systems.

Employee Data

This section contains the use cases identified that orchestrate employee data between SF and regional HR systems.

Employee Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail

HBP SAP HCM, DS SAP HCM, and SAP Retail are regional systems that manage information of employees for Belgium/Luxembourg, Serbia, and Greece. Previously, these systems used an integration system - SAP Cloud Platform Integration - to integrate with the regional SF solution, and the SAP Cloud Platform Integration (CPI) and the regional SF platforms will be decommissioned. Since these HR systems have little to no customization, CPI will be replicated by a MuleSoft API - SAP HBP Process API.

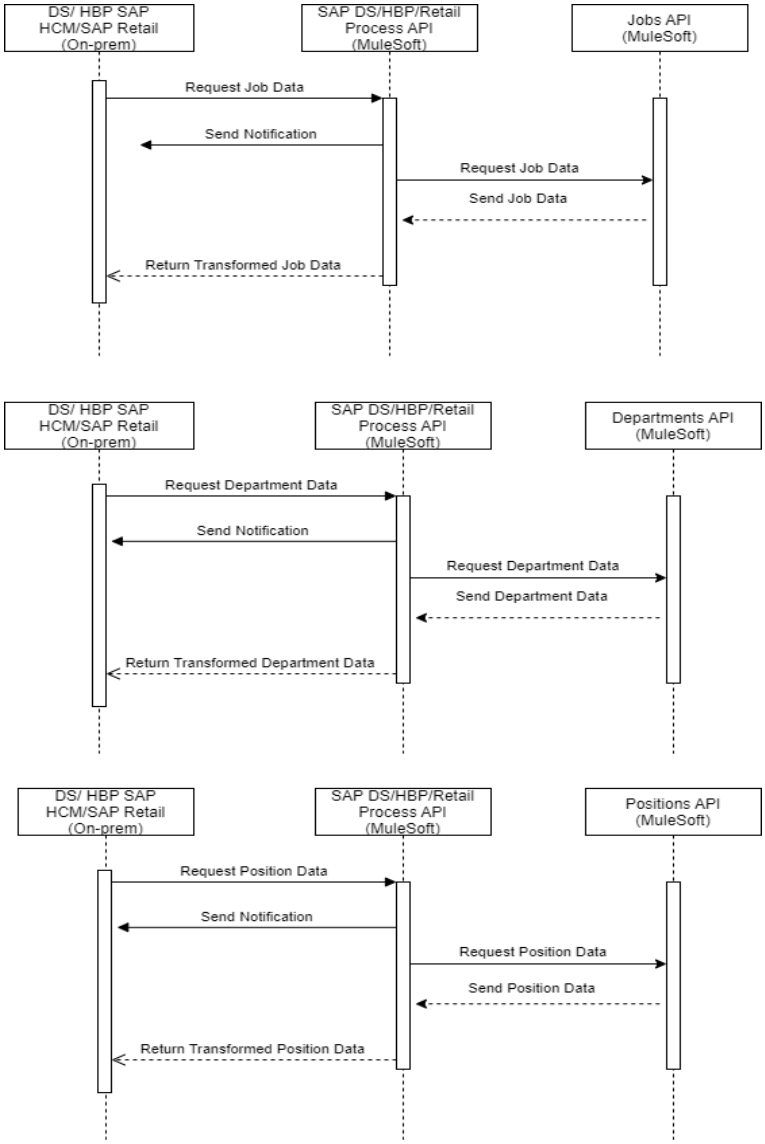


Figure 4.16: Employee Data do Regional Systems Sequence Diagram

The regional system sends a request to an API in the MuleSoft layer that notifies the system that it is sending the request. The request is then transformed and sent to SuccessFactors via SOAP. After receiving the data, it is transformed into the regional layer's required mappings in the API. The API then notifies the HR system that the data was sent and the process is ended.

Employee Data to Identity Management System

The Identity Management System is the global system responsible to manage the accesses and permissions of the users of the client's systems. The Employee data from all the countries will be migrated to Employee Central within SuccessFactors, which means that data changes synchronization with the Identity Management System will be adapted and automated.

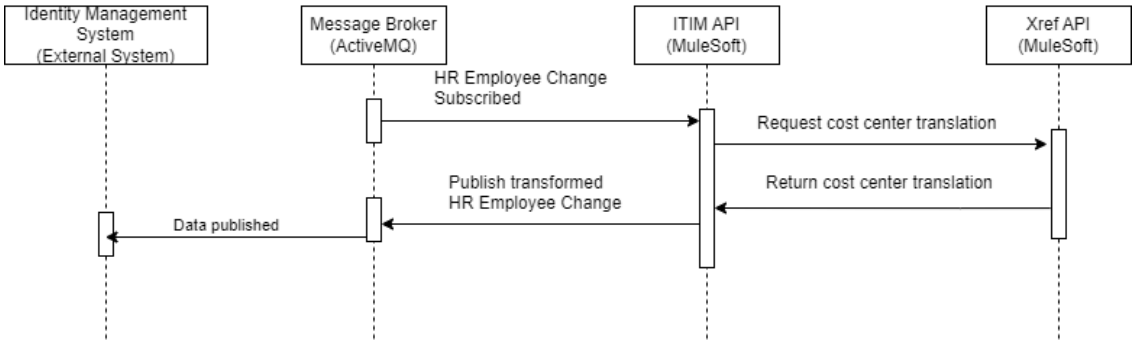


Figure 4.17: Employee Data to Identity Management System Sequence Diagram

A message subscriber processor within the MuleSoft API starts a new process upon receiving the message. The Identity Management System uses a legacy field that needs to be translated in another API - Xref API.
The data is then transformed into the required format and published to the queue.

UserID to HBP SAP HCM/ DS SAP HCM/ SAP Retail

The Identity Management System holds Identity data for employees whereas Employee Central (SuccessFactors) holds Employee data. The data regarding Identity needs to be propagated to different systems as it will not be mastered in Employee Central. This project's MuleSoft APIs will be responsible to propagate the UserID data changes from the Identity Management System so it can be propagated in real-time to the downstream systems.

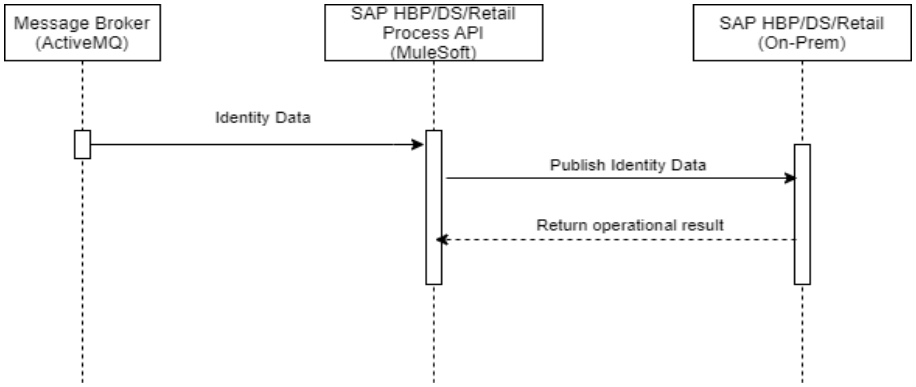


Figure 4.18: UserID to HBP SAP HCM/ DS SAP HCM/ SAP Retail Sequence Diagram

The API subscribes the Identity data from ActiveMQ queues. The focus is to publish the Identity data to the downstream systems

Employee Data to HR systems

Czech Republic, Romania, Serbia, Greece and Belgium have systems that manage information of employees on a regional level. Employee information changes on SuccessFactors need to be synchronized with these systems.

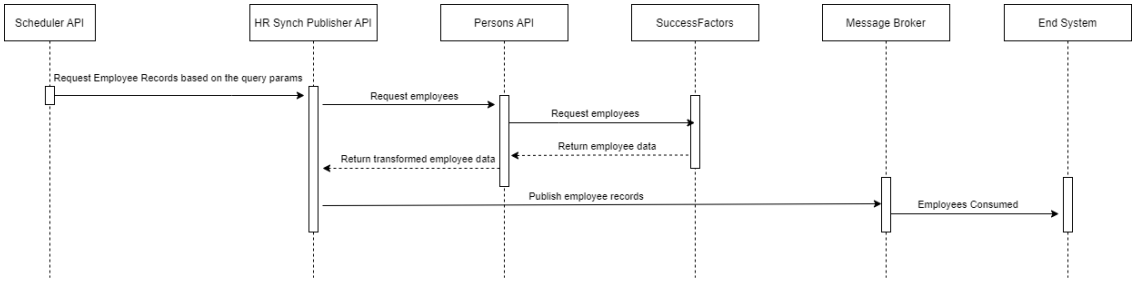


Figure 4.19: Czech Republic, Romania, Serbia, Greece, and Belgium Employee Sequence Diagram

A scheduler triggers an API hourly that triggers different APIs based on the type of record needed. In the case of employees, it triggers Persons API that retrieves the employee data from SuccessFactors. These records are published in ActiveMQ to be consumed by the end systems.

Employee Data to Coupa

Coupa is a Cloud Platform for Travel & Expenses Management, being responsible for the global expense management activities in the client. The data regarding Employee needs to be propagated to different systems, which means that data changes synchronization with Coupa will be adapted based on SuccessFactors.

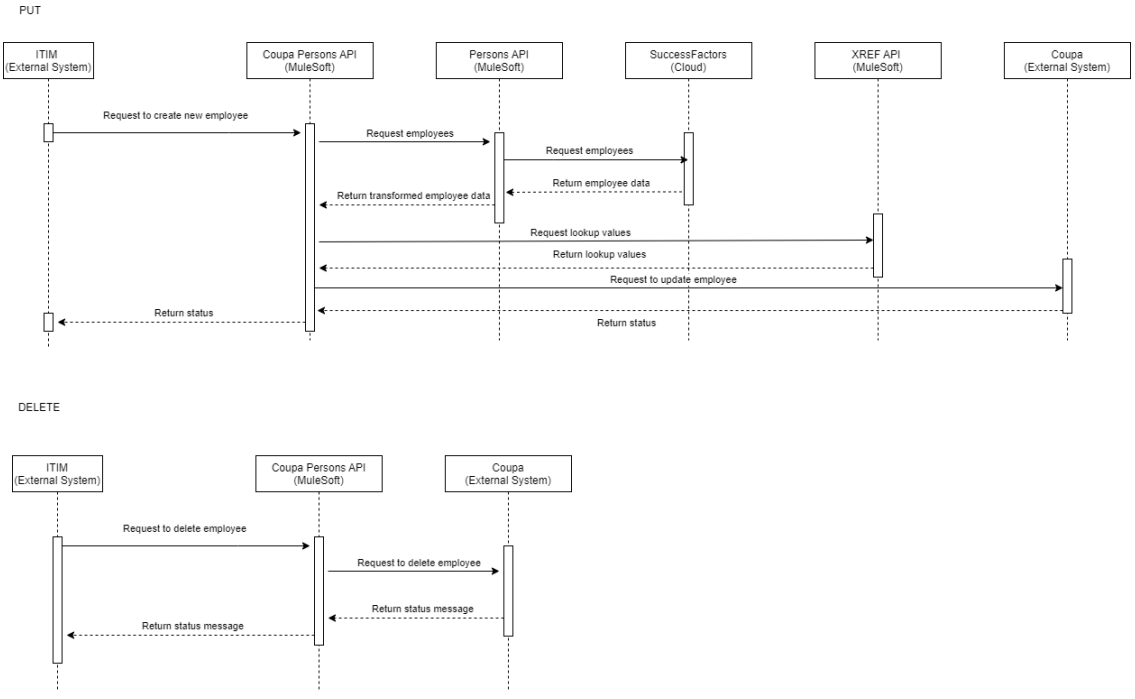


Figure 4.20: Employee Data from Belgium and Czech Republic to Coupa Sequence Diagram

For the PUT flow, an employee retrieved from Persons API will be updated or created in Coupa with enrichment from legacy values from Xref API.

As for the delete flow, a delete request with the employee Id is made to Coupa directly.

Employee Data to HR Systems On Demand

For the Belgium brand, the message broker-based solution was discarded on the brand's wish. So, for this particular integration, the HR System directly calls Persons API to get employee data.

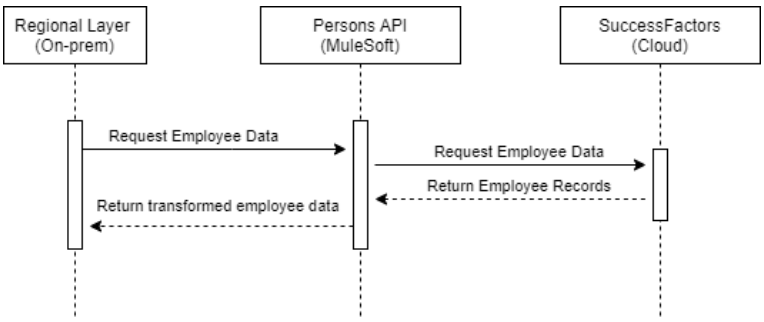


Figure 4.21: Employee Data to HR Systems On Demand Sequence Diagram

The HR System requests employee data directly to Persons API which retrieves and transforms the data in Employee Central.

Employee Data to HR Systems in Real-Time

The Greece brand needed to have a real-time integration to deal with involuntary employee termination. A SAP SuccessFactors Intelligent Service Center integration was created to trigger an API - HR Processes - that then triggers the subsequent integrations to update the employee in the downstream systems.

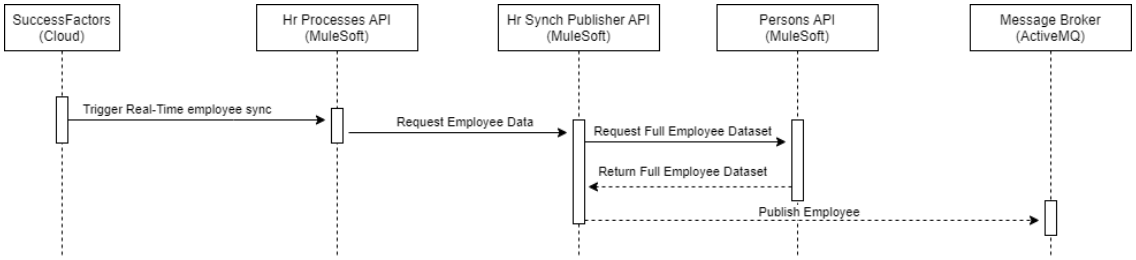


Figure 4.22: Employee Data to HR Systems in Real-Time

Employee Central triggers HR Processes API when an involuntary termination is issued. HR Processes triggers HR Synch API that pushes the employee, enriched by Persons API, into the correct queue.

Organisational Structure Data

This section contains the use cases identified for Organisational Structure Data - Positions, Jobs and Departments.

Organisational Structure Data Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail

As previously stated, HBP SAP HCM, DS SAP HCM, and SAP Retail are regional systems that manage information of employees for Belgium/Luxembourg, Serbia, and Greece, managing also Organisational Structure Data. The integration with SAP CPI will also be decommissioned and replicated with MuleSoft.
The queries made from these systems will be translated and sent to other APIs of this project and mapped to the required mappings that these systems need.

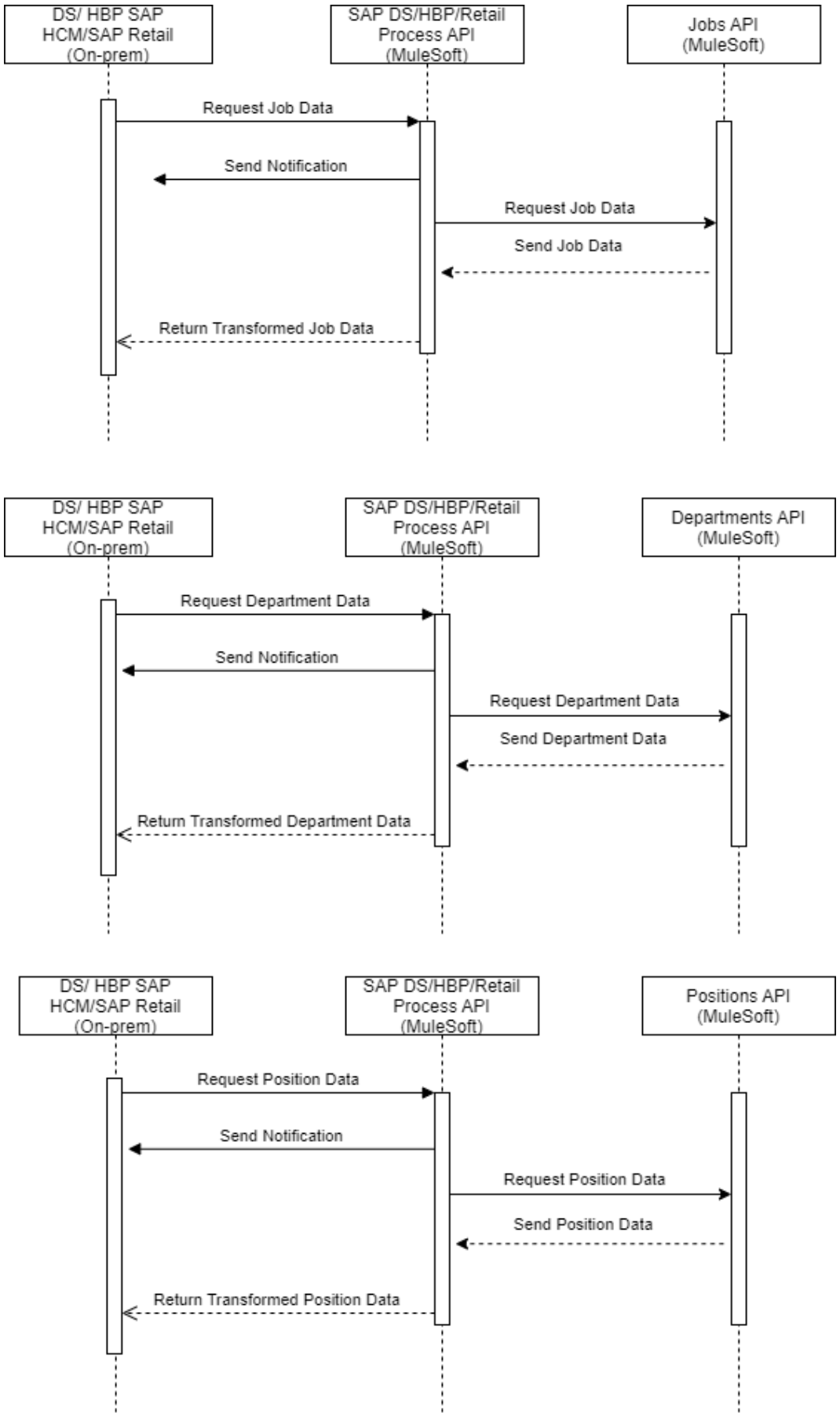


Figure 4.23: Organisational Structure to HBP SAP HCM/ DS SAP HCM/ SAP Retail Sequence Diagram

The regional system sends a request to an API in the MuleSoft layer that notifies the system that it is sending the request. The request is then transformed and sent to Positions, Jobs, and Departments APIs via a REST HTTP request. After receiving the data, it is transformed into the regional layer's required mappings in the API and sent to the end systems.

Organisational Structure Data to Regional HR Systems

The Regional HR Systems manage Organisational Structure Data information for Czech Republic, Greece, Romania, and Belgium brands. Data changes synchronization with this system will be adapted and automated.

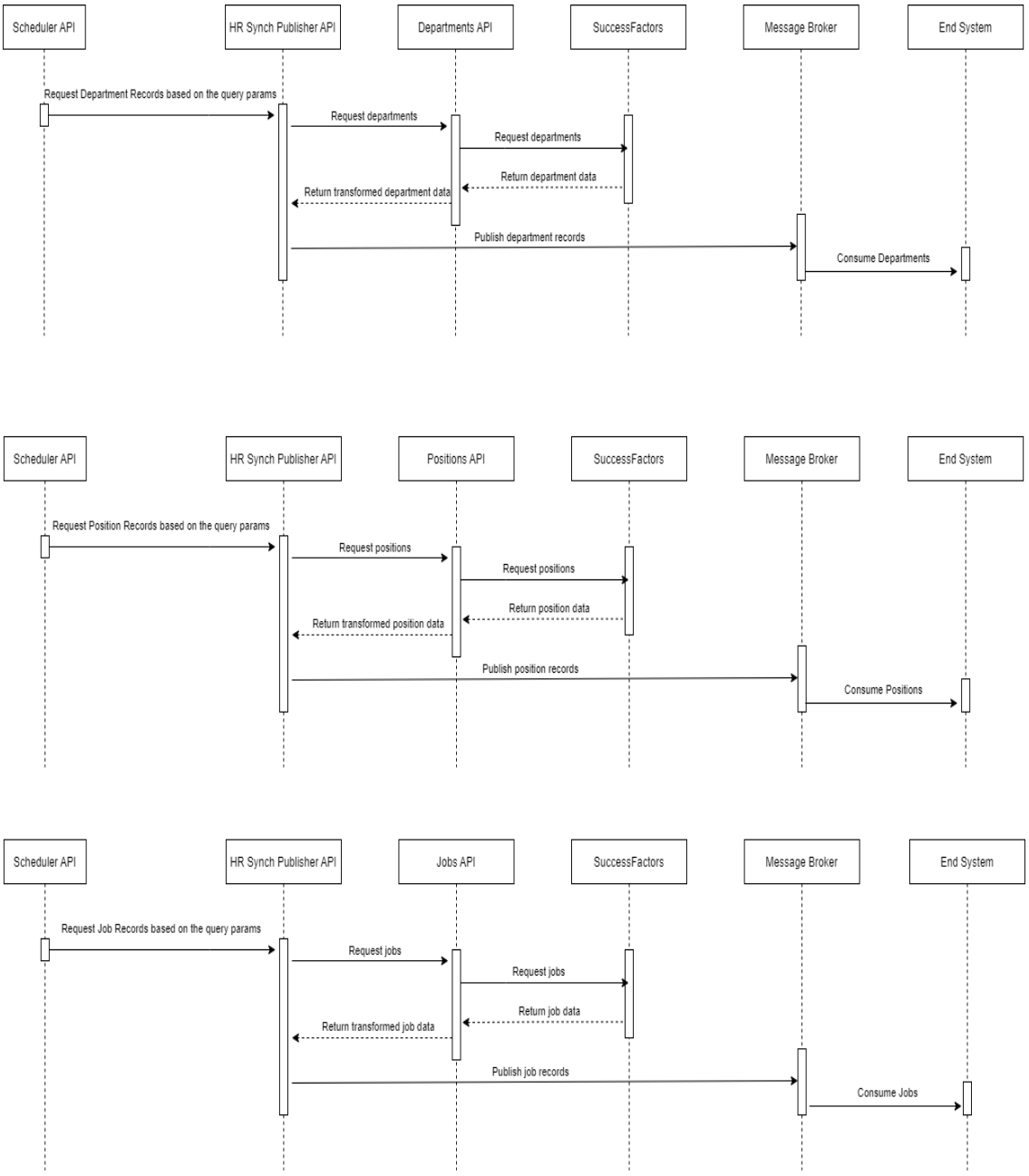


Figure 4.24: Organisational Structure Data to HR Systems Sequence Diagram

A scheduler triggers an API hourly that triggers different APIs based on the type of record needed. For organisational structures, Positions, Jobs, and Departments APIs are triggered. These records are published in ActiveMQ to be consumed by the end systems.

Job Requisition Data

Job Requisition Data to GetNoticed

GetNoticed is a digital recruitment agency. The Job Requisition data relates to the job vacancies that the client intends to post externally in career websites for new hiring purposes. Recruitment API, and a process MuleSoft API, GetNoticed Recruitment API, will be responsible to get Job Requisition Posting data from SF Recruitment in order to make it available to external systems such as GetNoticed.

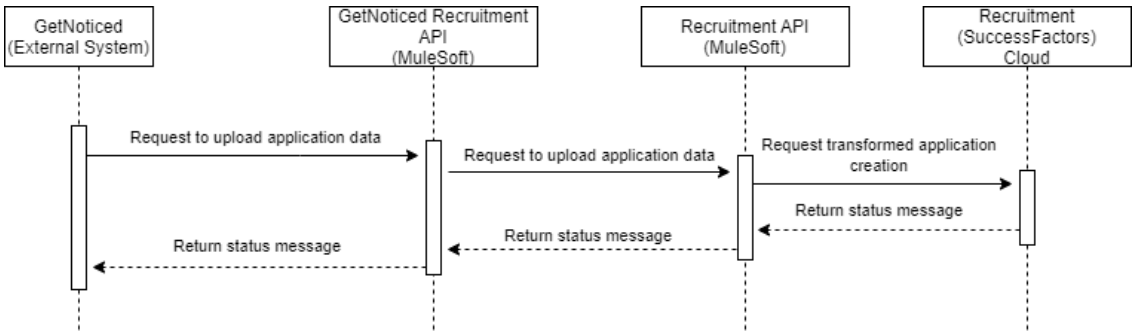


Figure 4.25: Job Requisition Data to GetNoticed Sequence Diagram

GetNoticed requests Job Posting data to GetNoticed Recruitment API that gets transformed data from Recruitment API, retrieved from Recruitment in SuccessFactors.

Job Requisition Data to HR Systems

The job vacancies also need to be posted externally in other systems that the brands desire.

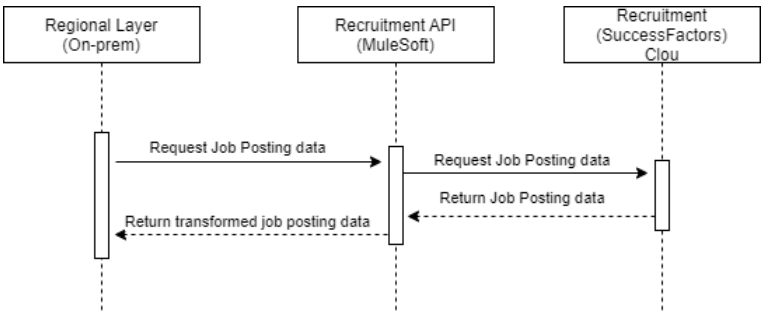


Figure 4.26: Job Requisition Data to HR Systems Sequence Diagram

The external regional layer requests the SuccessFactors Recruitment transformed recruitment data from Recruitment API.

Application Data

Application Data to GetNoticed

The Application data relates to a Candidate's application details when applying to a specific Job Requisition Posting from a job vacancy. Recruitment API and GetNoticed Recruitment API will be responsible to perform CRUD operations in SF Recruitment in order to make this data available to external systems such as GetNoticed.

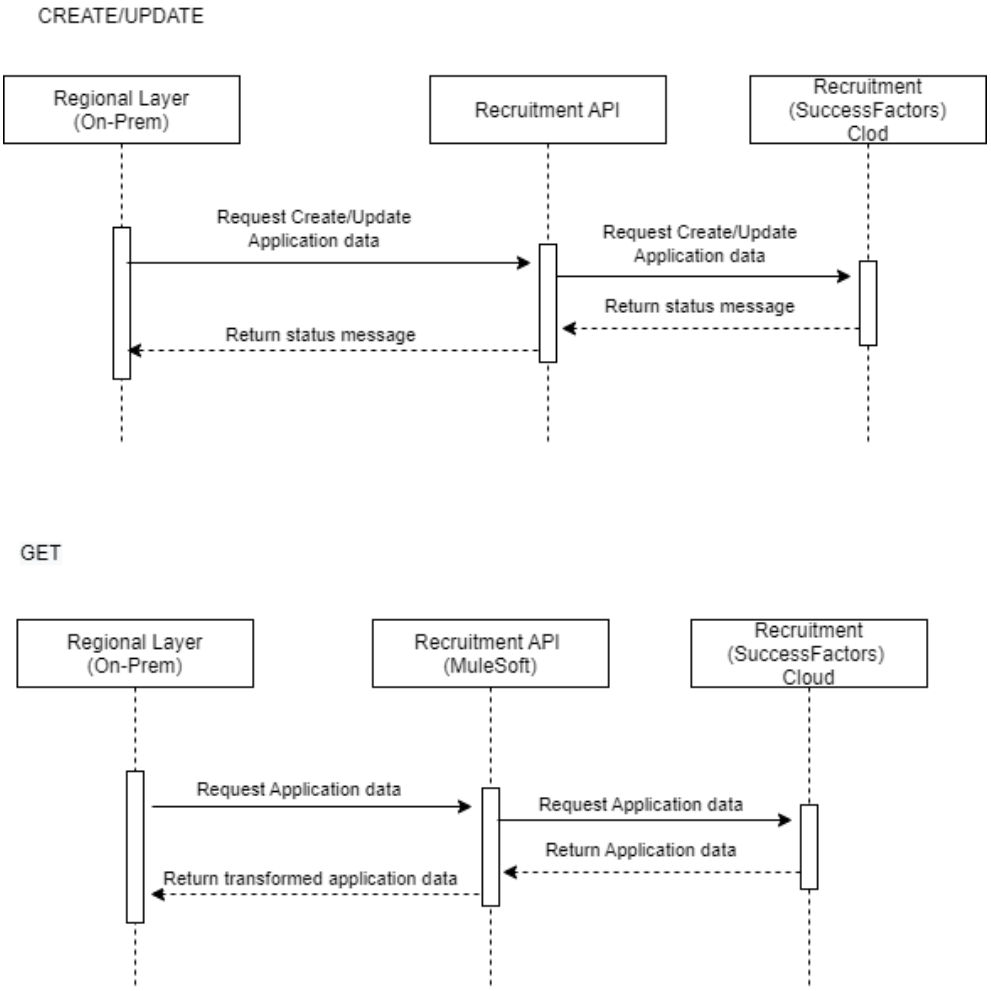


Figure 4.27: Application Data to GetNoticed Sequence Diagram

Application Data to HR Systems

The application data also needs to be posted externally in other systems that the brands desire.

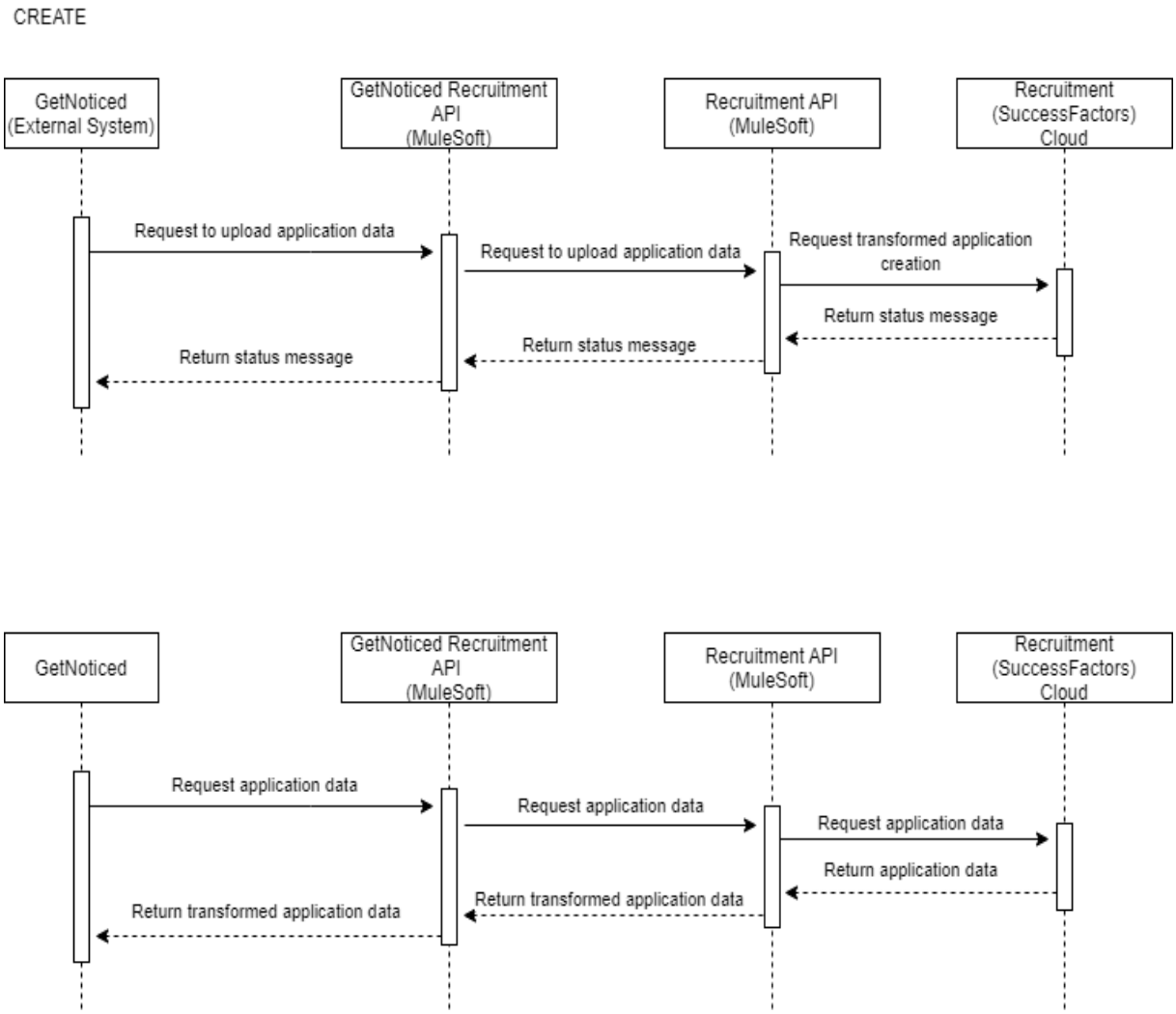


Figure 4.28: Application Data to HR Systems Sequence Diagram

Candidate Data

Candidate Data to GetNoticed

The Candidate data relates to the information from an individual, external to the client, that intends to submit an Application for a Job Requisition Posting (job vacancy). Recruitment API and GetNoticed Recruitment API will be responsible to perform CRUD operations in SF Recruitment in order to make this data available to external systems such as GetNoticed.

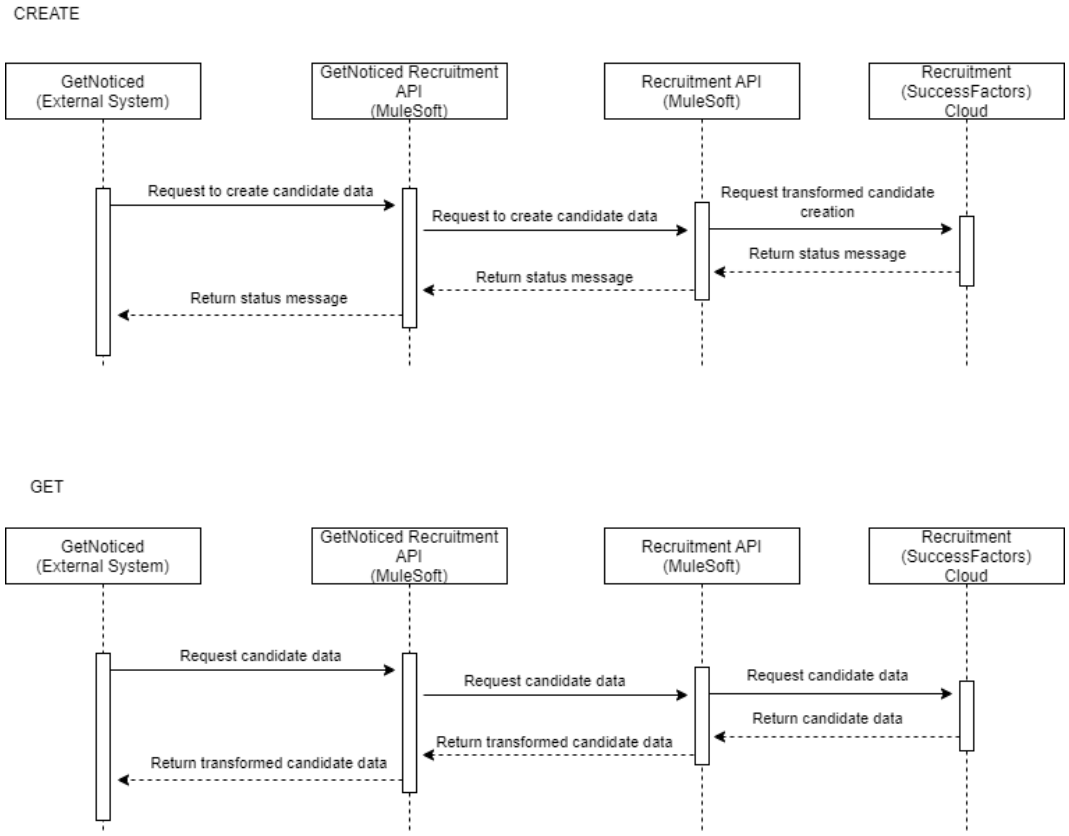


Figure 4.29: Candidate Data to GetNoticed Sequence Diagram

Application Data to HR Systems

The application data also needs to be posted externally in other systems that the brands desire.

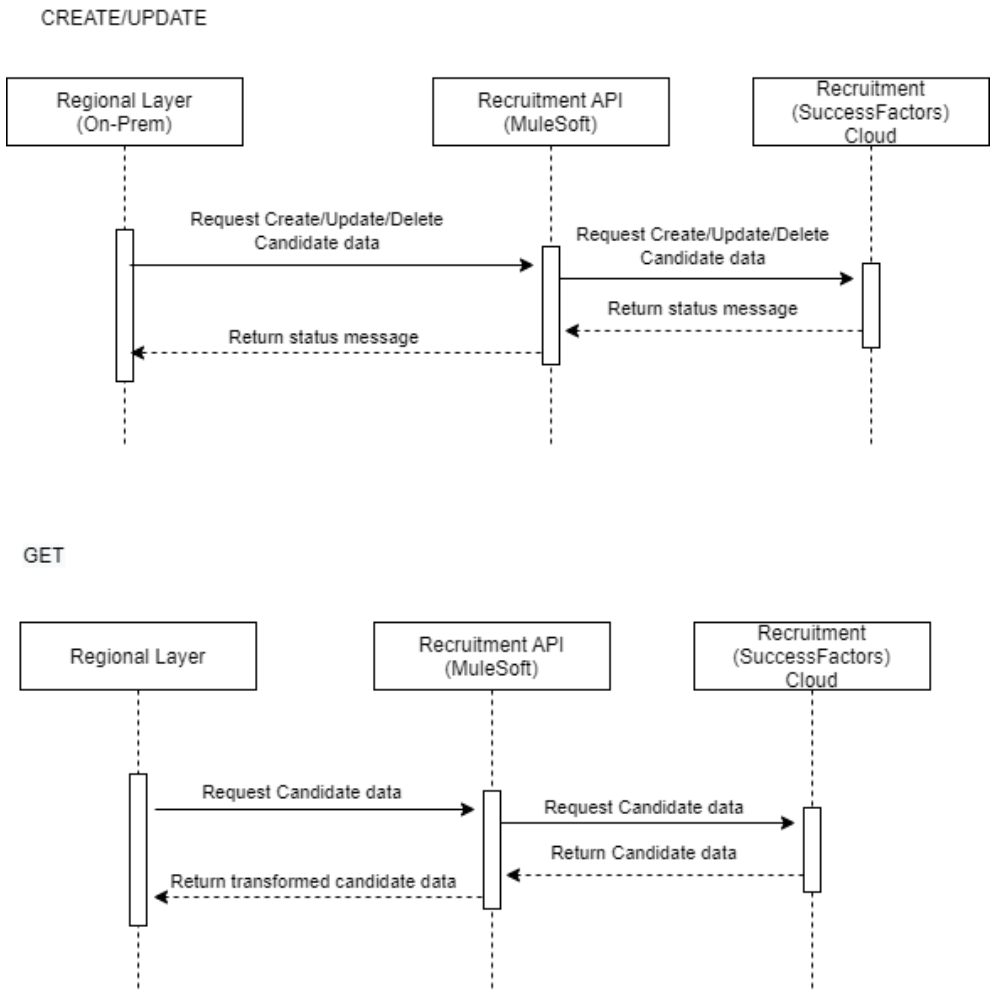


Figure 4.30: Candidate Data to HR Systems Sequence Diagram

Sicknotes

Sicknotes to ECP

Sicknotes are files downloaded from Czech Republic's Social Security website and they have information related to Czech employees' sick leaves. Employee Central Payroll had a CPI integration, using SOAP Protocol, that requested and returned these files. This solution will be decommissioned.
The sicknote files will be downloaded and placed on a SSH File Transfer Protocol (SFTP) server. When Employee Central Payroll (ECP) triggers the MuleSoft API, it will read and return the sicknotes to ECP, replicating all the operations and services from the CPI integration.

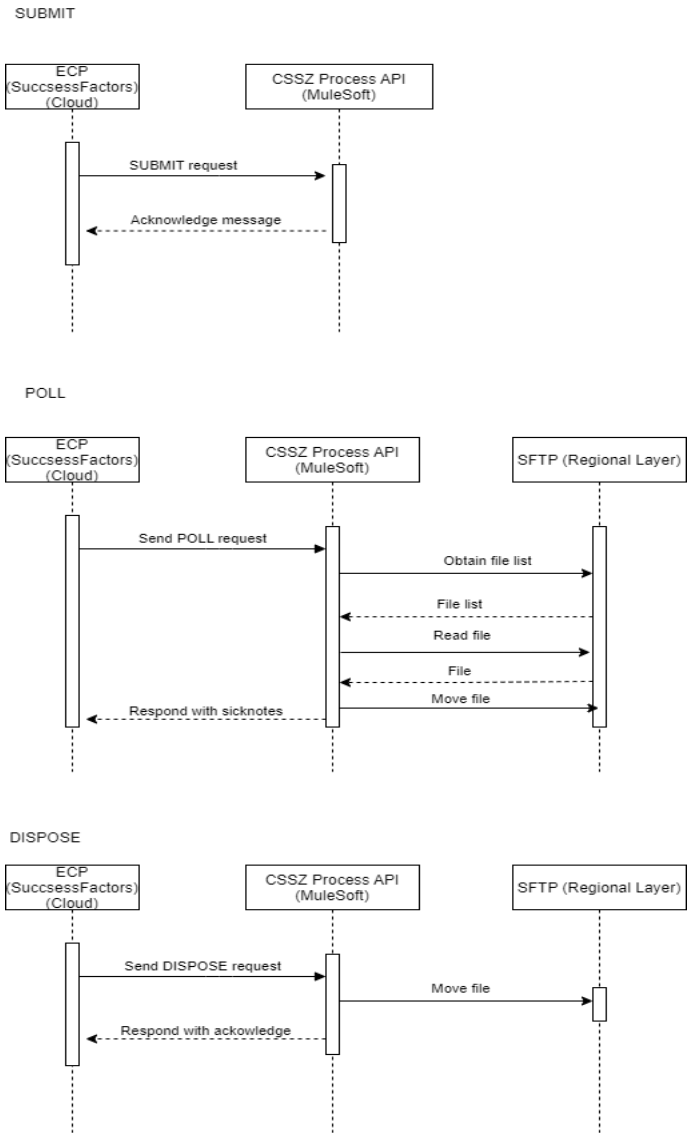


Figure 4.31: Sicknotes to ECP Sequence Diagram

ECP sends a SUBMIT request to start the process of the sicknotes retrieval. CSSZ Process API sends an acknowledge message. ECP then sends the POLL request to CSSZ Process API.

The API gets the file list from the SFTP server and reads them individually. After reading the files, it moves them from the Data folder to the Processing folder and returns the response to ECP.
When ECP wants to finish the transaction, it sends a DISPOSE message and the files will be moved from the processing folder to the archive folder.

Chapter 5

Implementation

This chapter defines and discusses the execution of the solution designed in chapter 4. The methodology of the implementation will be explained, the requirements-gathering process and the implementation of the APIs will be described.

5.1 Methodology

To ensure regular, consistent delivery of high-quality deliverables, an Agile approach was implemented, in which the sprints have a duration of two weeks. To implement the Kanban framework, Microsoft's Planner was used to control the cards that describe the tasks at hand. The categories for the cards were Backlog, To Do, On Hold, In Progress, Pending Review, and Reviewer Approved.
The methodology of development was grounded on the best practices of Continuous Integration and Continuous Delivery, using GitHub as the git repository and Azure DevOps Pipelines to automate the deployment process. Since there were four teams - the EU, NL, US, and maintenance team - essentially working on the same APIs and, evidently, on the same repositories, a thorough approach had to be met so that no accidents would happen.

5.2 Requirements Gathering

To gather requirements, multiple meetings, focus groups and workshops were performed where:

- An agenda was delineated before the meeting so that everyone was prepared on what topics the meeting would touch upon
- A facilitator was always present that would take notes and later provide a document where decisions and important themes were described

After that, requirements would be delineated and validated with the stakeholders of the project. Decisions were made based on the time, effort, and monetary expense of the tasks and negotiations between the brands, developers, and project managers were performed in order to assess what requirements were to be developed and what could constitute as changes to be done after the main requirements were fulfilled.

5.3 API Development

The APIs in this project were layered as described in chapter 4, in an API-led approach, with system, process, and experience APIs that perform different functions in the integration

layer.
The MuleSoft projects could be developed in a way where there are three main project files that contain the application logic:

- API file - Contains the flows that define the methods of the API
- Implementation - Contains the flows that hold the application logic
- Globals - Contains the configurations that the processors of the flows of the API and implementation files use

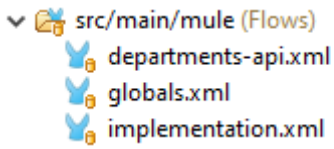


Figure 5.1: API Files

5.3.1 System APIs

Persons API

Employee Central (SuccessFactors) holds data for the client's Employees. It is required to have a single point of access to Employee Central system in order to manage Employee objects, i.e. to query multiple or individual entries.
The interaction with the Employee Central for this particular data is achieved through the Persons API using standard REST protocol for all requests except getting only employee changes, that is done using the SOAP protocol.

Method	Protocol	Description
GET /person-details	REST/SOAP	Retrieves Employee data (Full Dataset) from Employee Central (SuccessFactors) based on query parameters
GET /persons/personId	REST	Retrieves data from a single employee from Employee Central (SuccessFactors) based on personId (GemplID)
GET /persons/personId/history	REST	Retrieves historical Employee's data changes from Employee Central (SuccessFactors) based on personId (GemplID)

Table 5.1: Persons API methods

Recruitment API

Recruitment (SuccessFactors) holds Job Posting data. It is required to have a single point of access to Recruitment system in order to manage Job Posting objects, i.e. to query multiple or individual entries within the client's Job Posting data.

Considering the identified use cases and requirements, the Recruitment API was created to support the following functionalities and definitions:

- List Job Postings' records when triggered

- Perform CRUD operations to Candidates' records when triggered
- Perform CRUD operations to Applications' records when triggered

Method	Protocol	Description
GET/job-postings	REST	Retrieves Job Posting data from Recruitment (SuccessFactors) based on query parameters.
GET/candidates	REST	Retrieves Candidate data from Recruitment (SuccessFactors) based on query parameters.
POST/candidates	REST	Triggers the creation of a new candidate in Recruitment (SuccessFactors).
GET/candidates/candidateId	REST	Retrieves data from a single candidate from Recruitment (SuccessFactors) based on candidateId.
PATCH/candidates/candidateId	REST	Upsert the candidate information of an Candidate record in Recruitment (SuccessFactors) based on candidateId.
POST/candidates/candidateId/attachments	REST	Triggers the creation of attachments to an existing candidate in Recruitment (SuccessFactors).
POST/applications	REST	Triggers the creation of a new application in Recruitment (SuccessFactors).
GET/applications/applicationId	REST	Retrieves data from a single application from Recruitment (SuccessFactors) based on applicationId.
PATCH/applications/applicationId	REST	Upsert the application information of an Job Application record in Recruitment (SuccessFactors) based on applicationId.
POST/applications/applicationId/attachments	REST	Triggers the creation of attachments to an existing application in Recruitment (SuccessFactors).

Table 5.2: Recruitment API methods

Departments API

The client's department data is held in Employee Central (SuccessFactors). It is required a single point of access to Employee Central system in order to manage Department related

records, i.e. to query multiple or individual entries from different data domains within organisational structure data.
Considering the identified use cases and requirements, the Departments API was created to support the following functionalities and definitions:

- List Department's records when triggered by the HR Synch Publisher API as part of the Department's data synchronization process.

Method	Protocol	Description
GET /departments	REST	Retrieves Department data from Employee Central (SuccessFactors) based on query parameters.

Table 5.3: Departments API methods

Positions API

It is required a single point of access to Employee Central system in order to manage Position objects, i.e. to query multiple or individual entries within the client's Position data.
Considering the identified use cases and requirements, the Positions API was created to support the following functionalities and definitions:

- List Position's records when triggered by the HR Synch Publisher API, as part of the Position's data synchronization process.
- List Position organisation records when triggered by external systems, as part of the Positions's data synchronization process.

Method	Protocol	Description
GET /positions	REST	Retrieves Position data from Employee Central (SuccessFactors) based on query parameters.
GET /organisation	REST	Retrieves Position organisation data from Employee Central (SuccessFactors) based on query parameters.

Table 5.4: Positions API methods

Jobs API

Considering the identified use cases and requirements, the Jobs API was created to retrieve data from SuccessFactors and support the following functionalities and definitions:

- List Job's records when triggered by an external party, as part of the Job's data synchronization process.

Method	Protocol	Description
GET /jobs	REST	Retrieves Job data from Employee Central (SuccessFactors) based on query parameters.

Table 5.5: Jobs API methods

ITIM API

Employee Central (SuccessFactors) holds employee data that needs to be translated before being delivered to ITIM.
Hence, considering the API Led integration approach, the orchestration of ITIM data is performed through the ITIM API.
Considering the identified use cases and requirements, the ITIM API was created to support the following functionalities and definitions:

- Consume employee data from ActiveMQ, orchestrates and republish the data to ITIM ActiveMQ's queues.

Method	Protocol	Description
POST /synch-persons	REST	Triggers the propagation of Employee data to an SFTP location by invoking Persons API based on query parameters.
POST /async/synch-persons	REST	Triggers asynchronously the propagation of Employee data to an SFTP location by invoking Persons API based on query parameters.

Table 5.6: ITIM API methods

Coupa Persons API

Coupa system holds Travel Expenses data for employees. It is required a single point of access to the Coupa system in order to manage Employee and Expenses data, i.e. to create, update and query multiple or individual entries.
Considering the identified use cases and requirements, the Coupa Persons API was created to support the following functionalities and definitions:

- Employee synchronization from Employee Central to Coupa system
  - Each Employee's record information in Coupa must be coherent with its respective Employee data from HR (SuccessFactors). Meaning that if the Employee is present in both realities, any update performed in HR data must be propagated to Coupa.
  - The ITIM system is responsible to provision User information in Coupa, and this is achieved through the Coupa Persons API which allows to create, update fields or deactivate employees in Coupa.

– Considering that an employee record in Coupa is a combination of what is defined as HR data (Person) and User data (User), if a CRUD operation is to be provisioned by ITIM, the Coupa Persons API takes advantage of the existent Persons API to enrich the employee record (User) with the remaining fields of HR data (Person) before provisioning that record to Coupa system.

Method	Protocol	Description
GET /persons	REST	Retrieves Employee data from Coupa by invoking the Coupa API based on query parameters.
POST /persons	REST	Triggers the creation of a new Employee in Coupa by invoking the Coupa API.
GET /persons/personId	REST	Retrieves Employee data from Coupa by invoking the Coupa API based on HR personId (GlemplID).
PUT /persons/personId	REST	Updates an existing Employee record in Coupa by invoking the Coupa API based on HR personId (GlemplID).
DELETE /persons/personId	REST	Deactivates an Employee in Coupa by invoking the Coupa API based on HR personId (GlemplID).

Table 5.7: Coupa Persons API methods

Xref API

Throughout the integration with all the systems on the scope of the project, some transformations are required to translate fields from one system's language to another's. Xref was created for that purpose. This API translates identifiers or lists of values between systems. To perform that, the list of values or identifiers need to be stored in a specific format in a PostgresDB database. Xref will then be responsible to read from this database and value translation.

Method	Protocol	Description
POST /instance-xref	REST	Method that inserts a number of identifiers and binds them with a canonical ID.
GET /instance-xref	REST	Method to retrieve the translation of identifiers between systems as part of the Instance Cross Reference.
GET /lov-xref	REST	Method to retrieve the direct translation of entity values between systems as part of the List of Values (LoV) Cross Reference.
POST /lov-xref	REST	Method to insert a number of lovs and bind them with a canonical ID.
GET /lov-xref/canonicalId	REST	Method to retrieve a specific list of Values (LoV) Cross Reference component for a canonicalId.

Table 5.8: Xref API methods

CSSZ Process API

CSSZ is Czech Republic's Social Security and it holds employee's sicknotes.
This API orchestrates requests from SAP ECP to retrieve sicknotes that were downloaded from CSSZ's e-Portal and uploaded into an SFTP file server to be processed and returned by CSSZ Process API.

Method	Protocol	Description
POST /SUBMIT	SOAP	SAP ECP sends a SUBMIT request to start the process.
POST /POLL	SOAP	Method to retrieve the files uploaded to the SFTP file server.
POST /DISPOSE	SOAP	Method to end the transaction.

Table 5.9: CSSZ Process API methods

5.3.2 Process APIs

GetNoticed Recruitment API

GetNoticed Recruitment API is a process API that connects to Recruitment API to support the following functionalities and definitions:

- Triggers Recruitment API to retrieve Job Postings' records when invoked by GetNoticed, as part of the Job Postings' data synchronization process.
- Triggers Recruitment API to perform CRUD operations to Candidates' records when invoked by GetNoticed, as part of the Candidates' data synchronization process.

- Triggers Recruitment API to perform CRUD operations to Applications' records when invoked by GetNoticed, as part of the Applications' data synchronization process.

Method	Protocol	Description
GET/job-postings	REST	Retrieves Job Posting data from Recruitment API based on query parameters.
GET/candidates	REST	Retrieves Candidate data from Recruitment API based on query parameters.
POST/candidates	REST	Triggers the creation of a new candidate in Recruitment API.
POST/applications	REST	Triggers the creation of a new application in Recruitment API.
POST/applications/applicationId/attachments	REST	Triggers the creation of attachments to an existing application in Recruitment API.

Table 5.10: GetNoticed Recruitment API methods

SAB HBP/DS/Retail Process API

Employee Central holds data for Employee and Organisational Structure Data.
This API orchestrates requests from the SAP On-Prem systems HBP SAP HCM, SAP DS HCM, and SAP Retail and requests Employee data to Employee Central and Organisational Structure Data to the existing System APIs.

Method	Protocol	Description
POST/ EMPLOYEE_UPDATE_REQUEST	SOAP	The SAP On-prem systems send a request to synchronize employee data.
POST/ EMPLOYEE_UPDATE_CONFIRMATION	SOAP	The SAP On-prem systems send a confirmation to confirm that it received and updated employee data
POST/ ORGSTRUCT_UPDATE_REQUEST	SOAP	The SAP On-Prem systems send a request to return organisational structure data (Jobs, Position, and Departments)
POST/applications/applicationId/attachments	REST	Triggers the creation of attachments to an existing application in Recruitment API.

Table 5.11: SAP HBP/DS/Retail Process API methods

5.3.3 Experience APIs

HR Synch Publisher API

Employee Central holds Employee, Department, Location, Position, and Job related data. The interaction with Employee Central is achieved through the System APIs such as: Persons API, Departments API, Positions API and Jobs API.
Considering the identified use cases and requirements, the HR Synch Publisher API was created to support the following functionalities and definitions:

- Each data change regarding Employee data, Department data, Position data and Job data that occurs in Employee Central must be coherent with the data available to be subscribed in the Message Broker. Meaning that the data is present in both realities, and that data must be propagated to ActiveMQ topics/queues.
- When triggered by the Scheduler API, retrieve delta data changes of Employee, Department, Location, Position, and Job data by invoking Persons API, Departments API, Positions API, and Jobs API in order to publish records to the Message Broker (ActiveMQ) as part of the SuccessFactors Employee Central (EC) related data synchronization process.

Method	Protocol	Description
POST /synch-persons	REST	Triggers the propagation of Employee data to ActiveMQ by invoking Persons API based on query parameters.
POST/synch-departments	REST	Triggers the propagation of Department data to ActiveMQ by invoking Departments API based on query parameters.
POST/synch-positions	REST	Triggers the propagation of Position data to ActiveMQ by invoking Positions API based on query parameters.
POST/synch-jobs	REST	Triggers the propagation of Job data to ActiveMQ by invoking Jobs API based on query parameters.
POST /synch-identity	REST	Get triggered from IAM to propagate Identity Data to ActiveMQ based on query parameters.
POST /async/synch-persons	REST	It is triggered asynchronously to perform the propagation of Employee data to ActiveMQ by invoking Persons API based on query parameters.
POST/async/synch-departments	REST	It is triggered to perform the propagation of Department data to ActiveMQ by invoking Departments API based on query parameters.

POST/async/synch-positions	REST	It is triggered asynchronously to perform the propagation of Position data to ActiveMQ by invoking Positions API based on query parameters.
POST/async/synch-jobs	REST	It is triggered asynchronously to perform the propagation of Job data to ActiveMQ by invoking Jobs API based on query parameters.
POST /async/synch-identity	REST	Get triggered from IAM to propagate Identity Data to ActiveMQ based on query parameters.

Table 5.12: HR Synch Publisher API methods

HR Processes API

This API triggers HR Synch Publisher API when SuccessFactors EC triggers real-time data changes that need to be propagated to the end systems.

Method	Protocol	Description
POST /persons/personId/synch-person	REST	Triggers the synchronization of employee changes within Blade's landscape by invoking HR Synch Publisher API, when triggered by SuccessFactors (SF) Employee Central.

Table 5.13: HR Processes API methods

Chapter 6

Experimentation and Evaluation

This chapter describes the process of evaluating the developed solution. Firstly, the hypotheses are identified, then the indicators to evaluate the solution are presented, and lastly, the explanation of the method applied to evaluate the solution is described.

6.1 Hypotheses

A hypothesis is a conjecture that provides a verifiable statement that explains what is going to be tested and what's expected to find in relation to the set variables [68].
The objective of this project is to create an integration solution that's auditable, safe, reusable, and abstract that removes the perils of cloud platforms. Keeping that in mind, to evaluate the dissertation, a research hypothesis was created to evaluate the indicators presented above and is as follows:

- The proposed hybrid integration framework will efficiently connect the HR systems to SuccessFactors and use the same data for all the systems and circumvent cloud platform's limitations.

Further hypotheses are required to create a solid and usable system:

- The proposed hybrid integration framework is auditable
- The proposed hybrid integration framework is reusable
- The proposed hybrid integration framework is reliable
- The proposed hybrid integration platform is secure

6.2 Evaluation Indicators and Sources of Information

The evaluation indicators are markers that measure if something was accomplished or if progress was made towards achieving a specific output in the solution.[69]
The considered evaluation indicators were:

- **Integrations**: The integrations between legacy systems and SuccessFactors need to be evaluated
- **Quality and Reliability**: Understand if the integration layer is reliable
- **Audit and monitorability**: It is intended to evaluate if whether or not the integration layer is auditable and monitorable

- **Reusability**: An evaluation of the level of reusability of the APIs needs to be performed
- **Security**: It is intended to evaluate the security of the integration layer
- **Documentation**: Documentation of the APIs needs to be evaluated

To evaluate if the integrations were successful, unit, integration, and acceptance testing is going to be performed. If the integration is accepted by the client, the project manager marks the integration as 100% and that percentage is going to be used for the evaluation. For reliability, monitorability, reusability, and documentation, the system architects and the cloud and maintenance teams of the client need to provide feedback and evaluate if whether or not the goal was achieved.
For security, all APIs need to have authentication in place.

6.3 Evaluation

To evaluate quality, criteria and ways to measure quality have to be defined to quantify and accurately appraise it. When quality characteristics can't be measured, a further breakdown of these characteristics has to be done.[70]
To evaluate this solution, the Quantitative Evaluation Framework, proposed by Escudeiro & Bidarra in 2008 [70] was used as well as software tests.

6.3.1 Quantitative Evaluation Framework

The Quantitative Evaluation Framework allows to correspond requirements to quality characteristics. This framework needs the definition of several elements such as dimensions, factors, requirements, and metrics.[71]
The first phase of the QEF is to define dimensions. In this specific case, these are the characteristics that the solution must have. Then, the factors are defined for each dimension. These are accumulations of requirements with a given percentage of the weight of the dimension it is inserted in. The third phase is the definition of the requirements of the factors, that have a given weight in the number between 0 and 10, with increments of 2:

- 10 - Fundamental
- 8 - Very Important
- 6 - Important
- 4 - Necessary
- 2 - Optional
- 0 - Irrelevant

Lastly, the metrics are set for each requirement.[70]

Calculation

To calculate the value of each dimension, the value of each factor must be set, being n the number of factors for the calculated dimension[70]:

$$Factor = \sum_{m}(pr_m * pc_m)$$

$$Dimension_i = \sum_{n}(p_n * factor), \quad \sum_{n}(p_n) = 1 \quad and \quad p_n \in [0,1]$$

Each factor is calculated with the following formula, with m being the number of criteria for the factor, pr_m the weight of criteria m, and pc_m the percentage of completeness of the m criteria:

$$Factor_n = \frac{1}{\sum_m pr_m} * \sum_{m}(pr_m * pc_m)$$

The global system deviation, the euclidean distance, is given by:

$$D = \sqrt{\sum_{j}(1 - \frac{Dim_j}{100})^2}$$

In the end, the system quality is calculated by:

$$q = (1 - \frac{D}{\sqrt{n}}) * 100, \quad q \in [0,100]$$

Calculation of the quality of the system

Integrations

- Factors
  - Integrations with legacy systems
    - ☐ Employee Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail
    - ☐ Employee Data to Identity Management System
    - ☐ UserID to HBP SAP HCM/ DS SAP HCM/ SAP Retail
    - ☐ Employee Data to HR systems
    - ☐ Employee Data to Coupa
    - ☐ Employee Data to HR Systems On Demand
    - ☐ Employee Data to HR Systems in Real-Time
    - ☐ Organisational Structure Data Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail
    - ☐ Organisational Structure Data to Regional HR Systems
    - ☐ Job Requisition Data to GetNoticed
    - ☐ Job Requisition Data to HR Systems
    - ☐ Application Data to GetNoticed
    - ☐ Application Data to HR Systems
    - ☐ Candidate Data to GetNoticed

        - ☐ Application Data to HR Systems
        - ☐ Sicknotes to ECP

Quality and reliability

- Factors
    - Integration Reliability
        - ☐ Fault detection
        - ☐ Logging

Audit and monitorability

- Factors
    - Auditing and tracing
        - ☐ Logging
        - ☐ Exception/Error tracking
        - ☐ Metrics
        - ☐ Health checks

API Reusability

- Factors
    - Reusability
        - ☐ Abstraction
        - ☐ API-led approach

API Security

- Factors
    - Security
        - ☐ API authentication

API Documentation

- Factors
    - Documentation
        - ☐ Architecture Documentation
        - ☐ Technical Documentation
        - ☐ API Documentation

6.3.2 System Testing

System testing tests the whole system in regard to its requirements. It aggregates several tests such as functional testing, performance testing, and unit tests.[72]
In this project, the two main testing approaches were System Integration Testing (SIT) and User Acceptance Testing (UAT), performed by the brands' Quality Assurance team.

Unit Testing

A unit test executes a single "unit" and compares the actual and anticipated outcomes. A single or several class methods are tested to provide observable outcomes that are automatically checked [73].
In this project, all APIs were tested with unit testing, where a mocked SuccessFactors payload was consumed and the output of the API was compared to what the outcome was supposed to be. A minimum code coverage of 90% was needed for an API to move on to the next phase of testing.
Unit tests were developed at the same time as the APIs were developed and the API was only deployed if all the tests had passed.

System Integration Testing

SITs are tests performed on a comprehensive, integrated system to see whether it complies with its set requirements [74].
A task performed by the system is a test, but a series of tasks is a possible test case since the transition between two tasks could be a significant context shift [75].
The QA testers would request data from the APIs in the Development and Testing environments with mock data and analyze if the data was consumed correctly by their end systems. A good communication basis is key between the teams. If any error or defect appeared, the logs would be analyzed to see in which end the defect lied. When the defect was fixed, testing would resume.
This phase of testing started in July and ended in October.

User Acceptance Testing

Before a system is delivered to the end users, it is crucial to do user acceptance testing. A client should be able to decide whether to accept the system based on the results of an acceptance test [76].
Consequently, it gives the client confidence that the application satisfies all the conditions given and the manner in which they behave as it was anticipated [76].
The code would be merged into the acceptance branches and it would be performed with data that is similar to production. If the integration was accepted by the client, a 100% would be considered for the Integrations with Legacy Systems factor requirements.

6.4 Results

The results of the Quality evaluation framework give a score of 93 out of 100. It is important to mention that the values in the tables were inserted by the author. To have a higher degree of veracity, an external individual could have evaluated and inserted the values, but that wasn't possible for this project.
The integrations were almost completely achieved but the reusability aspect of the solution

was lacking. Quality was almost fully achieved, but it lacks in fault tolerance. Exception and error tracking were not completely achieved for the audit and monitorability dimension. All APIs have authentication and have RAML specifications, technical, and architecture documentation.
The global deviance of the system (D) value was 0,33, and the global quality of the system (q) was 93% so we can conclude that the framework achieves its requirements. Attachment 1 provides the complete QEF implementation.

Chapter 7

Conclusion

This dissertation presented and explained how an integration framework to orchestrate requests between on-prem systems and the new SuccessFactors HR repository.
The first phase of this dissertation studies the current state of the art of integration approaches and modern solutions that could solve the problem presented. A value analysis was then performed in Chapter 3 in order to identify and choose the best solution to this problem.
A hybrid integration framework was designed in Chapter 4, following approaches studied in Chapter 2. Chapter 5 refers how the implementation of this solution was performed. The solution was evaluated in Chapter 6 to verify if the objectives of the solution were achieved.

7.1 Research Questions

In Chapter 1, a research question was posed - "How can the cloud platform SuccessFactors be connected with the inherited systems in order to decouple and allow for the fast transaction of data?". After concluding this dissertation, it's possible to conclude that a custom, API-Led integration framework hosted in Microsoft AKS allowed for the orchestration of data between SF and the on-prem systems.

7.2 Limitations and Future Work

Due to the lack of information about SAP CPI that the Employee Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail, Organisational Structure Data Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail and Sicknotes to ECP are replacing, these integrations weren't completed to their full extent, meaning that certain operations were replicated, but there are some niggles and errors that still occur that weren't fully ironed out.
Another point that will be addressed in the future is the decommission of ActiveMQ. ActiveMQ will be replaced by Kafka in the future.
UATs weren't fully completed in this project, meaning that there could be issues that weren't fully discovered.

7.3 Final Considerations

Companies are investing more and more in integration layers and hybrid integration frameworks will still be on the rise for years to come. The reutilization of legacy systems connected with newer platforms saves companies' money and helps them cherry-pick what system is

better for a particular function, letting the integration layer do all the hard work of connecting the systems.
This was a challenging project that made the student re-use the knowledge that was accumulated through the academic years. The ever-changing project day-to-day life where complex problems are solved and several key-players are involved and stakes are high is stimulating and gratifying.

References

[1] David Goerzig and Thomas Bauernhansl. "Enterprise Architectures for the Digital Transformation in Small and Medium-sized Enterprises". In: vol. 67. Elsevier B.V., 2018, pp. 540–545. doi: 10.1016/j.procir.2017.12.257.

[2] Daniele Wolfart et al. "Modernizing Legacy Systems with Microservices: A Roadmap; Modernizing Legacy Systems with Microservices: A Roadmap". In: (). doi: 10.1145/3463274.3463334. url: https://doi.org/10.1145/3463274.3463334.

[3] Sonja M. Hyrynsalmi et al. "Towards the utilization of cloud-based integration platforms". In: Institute of Electrical and Electronics Engineers Inc., June 2021. isbn: 9781665449632. doi: 10.1109/ICE/ITMC52061.2021.9570235.

[4] Abdullah Al Malaise Al-Ghamdi and Farrukh Saleem. "Enterprise application integration as a middleware: Modification in data process layer". In: Institute of Electrical and Electronics Engineers Inc., Oct. 2014, pp. 698–701. isbn: 9780989319317. doi: 10.1109/SAI.2014.6918263.

[5] Shanmugasundaram Palanimalai and Ilango Paramasivamb. "AN enterprise oriented view on the cloud integration approaches -hybrid cloud and big data". In: vol. 50. Elsevier B.V., 2015, pp. 163–168. doi: 10.1016/j.procs.2015.04.079.

[6] BEA Systems. *Understanding the Integration Framework*. 2001.

[7] Jon Pierce. *Integration Frameworks and Enterprise Integration Patterns*. Jan. 2014.

[8] Alexander H. Toledo, Robert Flikkema, and Luis H. Toledo-Pereyra. "Developing the Research Hypothesis". In: *Journal of Investigative Surgery* 24.5 (2011), pp. 191–194. doi: 10.3109/08941939.2011.609449. eprint: https://doi.org/10.3109/08941939.2011.609449. url: https://doi.org/10.3109/08941939.2011.609449.

[9] Anna-Karin Carstensen and Jonte Bernhard. "European Journal of Engineering Education Design science research-a powerful tool for improving methods in engineering education research". In: (2018). issn: 1469-5898. doi: 10.1080/03043797.2018.1498459. url: https://www.tandfonline.com/action/journalInformation?journalCode=ceee20.

[10] Julius Quarshie Azasoo and Kwame Osei Boateng. "A Retrofit Design Science Methodology for Smart Metering Design in Developing Countries". In: Institute of Electrical and Electronics Engineers Inc., July 2015, pp. 1–7. isbn: 9781467373678. doi: 10.1109/ICCSA.2015.23.

[11] Vijay Vaishnavi and Bill Kuechler. *DESIGN SCIENCE RESEARCH IN INFORMATION SYSTEMS*. 2017.

[12] Tim Ehrens. *What is integration? - Definition from WhatIs.com*. 2015. url: https://www.techtarget.com/searchcustomerexperience/definition/integration.

[13] Song Xudong, Wang Xueping, and Liu Xiaobing. "Research on enterprise application integration architecture and development approach". In: 2008, pp. 215–218. isbn: 9780769535050. doi: 10.1109/IITA.Workshops.2008.243.

[14] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns*. 2003.

[15] Alan Snyder. *Encapsulation and Inheritance in Object-Oriented Programming Languages*. 1986.

[16] Abdullah Al Malaise Al-Ghamdi and Farrukh Saleem. "Enterprise application integration as a middleware: Modification in data process layer". In: Institute of Electrical and Electronics Engineers Inc., Oct. 2014, pp. 698–701. isbn: 9780989319317. doi: 10.1109/SAI.2014.6918263.

[17] Meeraa Shahibo and Faathima Fayaza. *Service Oriented Architecture in Enterprise Integration*.

[18] Hassan. Rajaei et al. *Proceedings of the 2008 Spring simulation multiconference.* Society for Computer Simulation International, 2008, p. 880. isbn: 1565553195.

[19] Amelia Maurizio et al. "Service Oriented Architecture: Challenges for Business and Academia". In: *Hawaii International Conference on System Sciences* 0 (Jan. 2008), p. 315. doi: 10.1109/HICSS.2008.387.

[20] Derek T Sanders JA Hamilton and Richard A MacDonald. *Supporting A Service-Oriented Architecture*. 2008.

[21] Michael Stal. *Using Architectural Patterns and Blueprints for Service-Oriented Architecture*. 2006. url: www.ieee.org/portal/pages/about/documentation/copyright/polilink.html.www.computer.org/software.

[22] Jieming Wu and Xiaoli Tao. "Research of enterprise application integration based-on ESB". In: vol. 5. 2010, pp. 90–93. isbn: 9781424458462. doi: 10.1109/ICACC.2010.5487292.

[23] David A Chappell. *Enterprise Service Bus*. 1st ed. O'Reilly Media, Inc., 2004. isbn: 9780596006754.

[24] Martin Breest. *An Introduction to the Enterprise Service Bus*.

[25] Qing Li et al. "Enterprise Information Systems Applications integration in a hybrid cloud computing environment: modelling and platform) Applications integration in a hybrid cloud computing environment: modelling and platform Applications integration in a hybrid cloud computing environment: modelling and platform". In: (2013). doi: 10.1080/17517575.2012.677479. url: https://www.tandfonline.com/action/journalInformation?journalCode=teis20.

[26] David Reinsel, John Gantz, and John Rydning. *The Digitization of the World From Edge to Core*. 2018.

[27] Rafael Huang Cestari, Sebastien Ducos, and Ernesto Exposito. "iPaaS in Agriculture 4.0: An Industrial Case". In: *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. 2020, pp. 48–53. doi: 10.1109/WETICE49692.2020.00018.

[28] Saurabh Sharma. "Ovum Decision Matrix iPaaS 2015". In: ().

[29] *What is an Integration Platform as a Service (iPaaS)? | MuleSoft*. url: https://www.mulesoft.com/resources/cloudhub/what-is-ipaas-gartner-provides-reference-model.

[30] Saurabh Sharma. "Ovum Decision Matrix: Selecting an Integration PaaS (iPaaS) Solution, 2015-2016". In: (2015).

[31] Bindi Bhullar et al. *Market Share Analysis: Integration Platform as a Service, 2020*. 2021. url: https://www.gartner.com/en/documents/4002917-market-share-analysis-integration-platform-as-a-service-.

[32] Eric Thoo et al. *Gartner Magic Quadrant for Enterprise Integration Platform as a Service*. 2020. url: https://www.gartner.com/en/documents/3990698/magic-quadrant-for-enterprise-integration-platform-as-a-.

[33] *CloudHub | MuleSoft Documentation*. url: https://docs.mulesoft.com/runtime-manager/cloudhub.

[34] Dell Boomi. "Connect Once-Integrate Everywhere®". In: (). url: www.boomi.com.

[35] *What is Dell Boomi? - Definition from WhatIs.com*. url: https://www.techtarget.com/searchcloudcomputing/definition/Dell-Boomi.

[36] *Case Study: Adoption of Dell Boomi AtomSphere iPaaS by Novartis Simplifying integration and realizing TCO savings and faster time-to-value with iPaaS*. 2013.

[37] *Informatica Named a Leader in Gartner Magic Quadrant for EiPaaS for Seven Years in a Row*. url: https://www.prnewswire.com/news-releases/informatica-named-a-leader-in-gartner-magic-quadrant-for-eipaas-for-seven-years-in-a-row-301164629.html.

[38] *What a Hybrid Integration Platform Actually is? | LinkedIn*. url: https://www.linkedin.com/pulse/what-hybrid-integration-platform-actually-massimo-pezzini/.

[39] *iPaaS and hybrid integration platform | Learn how it's great to be HIP*. url: https://blog.axway.com/amplify-products/ipaas-hybrid-integration-platform.

[40] Rob van der Meulen. *Use a Hybrid Integration Approach to Empower Digital Transformation*. url: https://www.gartner.com/smarterwithgartner/use-a-hybrid-integration-approach-to-empower-digital-transformation.

[41] Cloud Standards Customer Council. "Cloud Customer Architecture for Hybrid Integration Executive Overview". In: (2017).

[42] IBM. *Hybrid Cloud Integration reference architecture - Hybrid Integration Reference Architecture*. url: https://ibm-cloud-architecture.github.io/refarch-integration/hybrid-ref-arch/.

[43] *Event-Driven Architecture*. url: https://aws.amazon.com/pt/event-driven-architecture/.

[44] Brenda M Michelson and Patricia Seybold. *Event-Driven Architecture Overview*. 2011.

[45] Mayur Raj, Singh Chouhan, and Sujatapriyambada Mishra. *API-led Connectivity Using Mulesoft For Healthcare*.

[46] *API-led Connectivity vs. SOA | MuleSoft Blog*. url: https://blogs.mulesoft.com/learn-apis/api-led-connectivity/api-led-connectivity-vs-soa/.

[47] Peter A Koen et al. *FuzzyFrontEnd: Effective Methods, Tools, and Techniques LITERATURE REVIEW AND RATIONALE FOR DEVELOPING THE NCD MODEL*.

[48] Peter Koen et al. "Providing Clarity and A Common Language to the "Fuzzy Front End"". In: *Research-Technology Management* 44.2 (2001), pp. 46–55. doi: 10.1080/08956308.2001.11671418.

[49] Pete McCaffrey. *What is a hybrid integration platform (HIP)? - Cloud computing news*. 2019. url: https://www.ibm.com/blogs/cloud-computing/2019/07/16/what-is-hybrid-integration-platform/.

[50] *Hybrid Integration Platform Market | 2021 - 26 | Industry Share, Size, Growth - Mordor Intelligence*. url: https://www.mordorintelligence.com/industry-reports/hybrid-integration-platform-market.

[51] T.L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. Advanced book program. McGraw-Hill International Book Company, 1980. isbn: 9780070543713. url: https://books.google.pt/books?id=Xxi7AAAAIAAJ.

[52] D.R. Kiran. "Chapter 30 - Quality Function Deployment". In: *Total Quality Management*. Ed. by D.R. Kiran. Butterworth-Heinemann, 2017, pp. 425–437. isbn: 978-0-12-811035-5. doi: https://doi.org/10.1016/B978-0-12-811035-5.00030-1. url: https://www.sciencedirect.com/science/article/pii/B9780128110355000301.

[53] Joseph Berk and Susan Berk. "Chapter 12 - Quality Function Deployment: Understanding and satisfying customer expectations..." In: *Quality Management for the*

Technology Sector. Ed. by Joseph Berk and Susan Berk. Woburn: Butterworth-Heinemann, 2000, pp. 124–134. isbn: 978-0-7506-7316-7. doi: https://doi.org/10.1016/B978-075067316-7/50012-7. url: https://www.sciencedirect.com/science/article/pii/B9780750673167500127.

[54] *QFD Online*. url: http://www.qfdonline.com/.

[55] SUSANA NICOLA, EDUARDA PINTO FERREIRA, and J. J. PINTO FERREIRA. "A NOVEL FRAMEWORK FOR MODELING VALUE FOR THE CUSTOMER, AN ESSAY ON NEGOTIATION". In: *International Journal of Information Technology & Decision Making* 11.03 (2012), pp. 661–703. doi: 10.1142/S0219622012500162. eprint: https://doi.org/10.1142/S0219622012500162. url: https://doi.org/10.1142/S0219622012500162.

[56] Valarie A. Zeithaml. "Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence". In: *Journal of Marketing* 52.3 (1988), pp. 2–22. issn: 00222429. url: http://www.jstor.org/stable/1251446.

[57] Chee Liew, Song Lian, and Muthaloo Subramaniam. "Relationship Value and Relationship Quality: An Exploration of Its Antecedents on Customer Loyalty". In: *Asian Social Science* 13 (Dec. 2017), p. 51. doi: 10.5539/ass.v13n12p51.

[58] Alexander Osterwalder. "The business model ontology a proposition in a design science approach". In: 2004.

[59] *Value Proposition Canvas – Download the Official Template*. url: https://www.strategyzer.com/canvas/value-proposition-canvas.

[60] Ian Sommerville. *Software engineering*. Pearson, 2011, p. 773. isbn: 9780137035151.

[61] *IEEE Standard Glossary of Software Engineering Terminology*. 1990. doi: 10.1109/IEEESTD.1990.101064.

[62] Rawaa Qasha, Jacek Cala, and Paul Watson. "Towards Automated Workflow Deployment in the Cloud Using TOSCA". In: Institute of Electrical and Electronics Engineers Inc., Aug. 2015, pp. 1037–1040. isbn: 9781467372879. doi: 10.1109/CLOUD.2015.146.

[63] Chaitanya K. Rudrabhatla. "Comparison of zero downtime based deployment techniques in public cloud infrastructure". In: Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 1082–1086. isbn: 9781728154640. doi: 10.1109/I-SMAC49090.2020.9243605.

[64] Friedrich Laux et al. "The Second International Workshop on Large-scale Graph Storage and Management". In: (2015).

[65] Jillian Hufford. *Why CSV File-based Integration Can Be Better than API-based Integration | nChannel Blog*. url: https://www.nchannel.com/blog/csv-file-based-integration-vs-api/.

[66] Philippe Dobbelaere and Kyumars Sheykh Esmaili. "Industry Paper: Kaaa versus RabbitMQ". In: 12 (2017). doi: 10.1145/3093742.3093908.

[67] Jiujiu Yu et al. "PaaS for web applications with OpenShift Origin You may also like Direct print technology with Qz Tray for web application transaction I M A D S Atmaja and I N G A Astawa-Exploration on Web Testing of Website PaaS for web applications with OpenShift Origin". In: *IOP Conf. Series: Journal of Physics: Conf. Series* 898 (2017), p. 82037. doi: 10.1088/1742-6596/898/8/082037.

[68] O'Leary Zina. *The Essential Guide to Doing Your Research Project*. 2017.

[69] CDC. *Developing Evaluation Indicators*. 2007. url: http://www.cdc.gov/std/program/pupestd.htm.

[70] Paula Escudeiro and José Bidarra. "Quantitative Evaluation Framework (QEF)". In: *Conselho Editorial/Consejo Editorial* (Jan. 2008), p. 16.

[71] Farideh Heidari and Pericles Loucopoulos. "Quality evaluation framework (QEF): Modeling and evaluating quality of business processes". In: *International Journal of Accounting Information Systems* 15.3 (2014). Business Process Modeling, pp. 193–223. issn: 1467-0895. doi: https://doi.org/10.1016/j.accinf.2013.09.002. url: https://www.sciencedirect.com/science/article/pii/S1467089513000389.

[72] Lionel Briand and Yvan Labiche. "A UML-Based Approach to System Testing". In: *Software and Systems Modeling* 1 (1 Sept. 2002), pp. 10–42. issn: 1619-1366. doi: 10.1007/s10270-002-0004-8.

[73] Michael Olan. *UNIT TESTING: TEST EARLY, TEST OFTEN **. 2003.

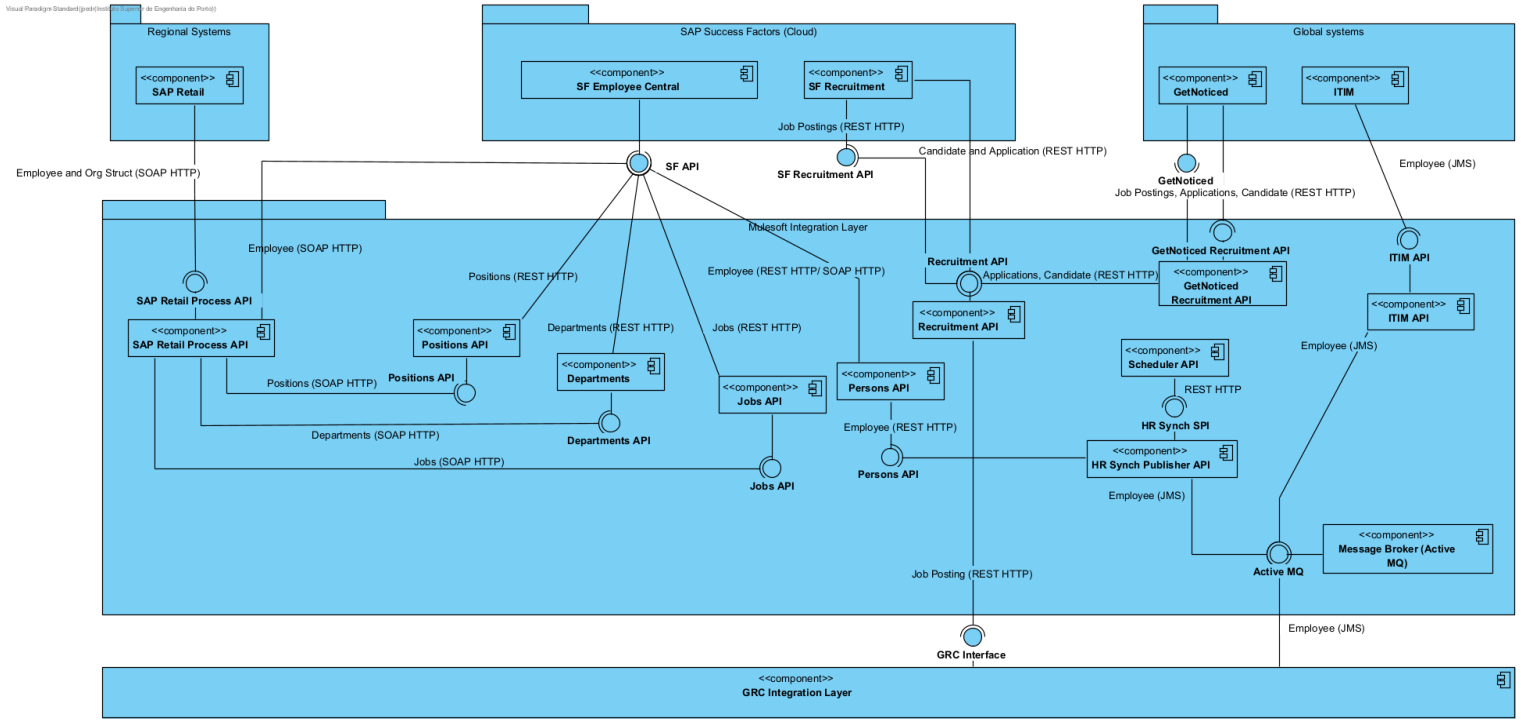
[74] Thomas Hamilton. *What is System Integration Testing (SIT) with Example*.

[75] England) International Symposium on Software Testing and Analysis (2007 : London. *Proceedings of the 3rd International Workshop on Advances in Model-Based Testing : 2007, London, United Kingdom, July 09-12, 2007*. ACM, 2007, p. 125. isbn: 9781595938503.

[76] Rozana Suman and Shamsul Sahibuddin. "User acceptance testing in mobile health applications: An overview and the Challenges". In: vol. Part F148384. Association for Computing Machinery, 2019, pp. 145–149. isbn: 9781450361033. doi: 10.1145/3322645.3322670.

Appendix A

BEL/LUX To-Be Landscape



Appendix B

Application of the Quality Evaluation Framework

q	D	Q_j	Dimension	Q_j	W_{ij} (Factor Weight *j* in Dim *i*) [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*) [0,100]
93%	0,33	78,0137	Integrations	78,0137	1,00	Integrations with Legacy Systems	10	II01 - Employee Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail	70
							10	II02 - Employee Data to Identity Management System	85
							10	II03 - UserID to HBP SAP HCM/ DS SAP HCM/ SAP Retail	85
							8	II04 - Employee Data to HR systems	70
							10	II05 - Employee Data to Coupa	85
							8	II06 - Employee Data to HR Systems On Demand	85
							10	II07 - Employee Data to HR Systems in Real-Time	85
							10	II08 - Organisational Structure Data Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail	50
							10	II09 - Organisational Structure Data to Regional HR Systems	80
							8	II10 - Job Requisition Data to GetNoticed	85
							8	II11 - Job Requisition Data to HR System	85
							8	II12 - Application Data to GetNoticed	85
							8	II13 - Application Data to HR System	85
							10	II14 - Candidate Data to GetNoticed	85
							8	II15 - Application Data to HR Systems	85
							10	II16 - Sicknotes to ECP	50
		95,3846	Quality and reliability	95,3846	1,00	Integration Reliability	6	IR01 - Fault Tolerance	80
							10	IR02 - Unit Testing	100
							10	IR03 - Logging	100
		93,75	Audit and monitorability	93,75	1,00	Audit and Tracing	10	AA01 - Logging	100
							10	AA02 - Exception/Error tracking	80
							2	AA03 - Metrics	100
							10	AA05 - Health checks	100
		76,6667	API Reusability	76,6667	1,00	Reusability	8	AR01 - Abstraction	60
							10	AR02 - API-Led Approach	90
		100	API Security	100	1,00	Security	10	AS01 - API Authentication	100
		100	API Documentation	100	1,00	Documentation	10	AD01 - Architecture documentation	100
							10	AD02 - Technical documentation	100
							10	AD03 - API Documentation	100

Appendix C

Quality Evaluation Framework Criteria

Dimension	Integrations			
Factor	Integrations with Legacy Systems			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
II01 - Employee Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail	Employee Data is correctly orchestrated between SF and SAP HBP/DS/Retail systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II02 - Employee Data to Identity Management System	Employee Data is correctly orchestrated between SF and the Identity Management System	No data was integrated	Partial integration was accomplished	Data was fully integrated
II03 - UserID to HBP SAP HCM/ DS SAP HCM/ SAP Retail	The User ID is synched with SAP HBP/DS/Retail systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II04 - Employee Data to HR systems	Employee Data is correctly orchestrated between SF and On-Prem HR systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II05 - Employee Data to Coupa	Employee Data is correctly orchestrated between SF and Coupa	No data was integrated	Partial integration was accomplished	Data was fully integrated
II06 - Employee Data to HR Systems On Demand	Employee Data is synched on-demand between SF and HR Systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II07 - Employee Data to HR Systems in Real-Time	Employee Data is synched in real time between SF and HR Systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II08 - Organisational Structure Data Data to HBP SAP HCM/ DS SAP HCM/ SAP Retail	Organisational Structure Data is correctly orchestrated between SF and SAP HBP/DS/Retail systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II09 - Organisational Structure Data to Regional HR Systems	Organisational Structure Data is correctly orchestrated between SF and On-Prem HR systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II10 - Job Requisition Data to GetNoticed	Job requisitions are orchestrated between SF Recruitment and GetNoticed	No data was integrated	Partial integration was accomplished	Data was fully integrated
II11 - Job Requisition Data to HR Systems	Job requisitions are orchestrated between SF Recruitment and regional HR Systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II12 - Application Data to GetNoticed	Job Applications are orchestrated between SF Recruitment and GetNoticed	No data was integrated	Partial integration was accomplished	Data was fully integrated
II13 - Application Data to HR Systems	Job Applications are orchestrated between SF Recruitment and regional HR Systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II14 - Candidate Data to GetNoticed	Candidate Data is orchestrated between SF Recruitment and GetNoticed	No data was integrated	Partial integration was accomplished	Data was fully integrated
II15 - Candidate Data to HR Systems	Candidate Data is orchestrated between SF Recruitment and regional HR Systems	No data was integrated	Partial integration was accomplished	Data was fully integrated
II16 - Sicknotes to ECP	Czech Employee Sicknotes are sent to ECP	No data was integrated	Partial integration was accomplished	Data was fully integrated

Appendix D

Quality Evaluation Framework Criteria

Dimension	Quality and reliability			
Factor	Integration Reliability			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
IR01 - Fault Tolerance	All APIs must have error handlers to control, propagate and handle errors that may appear accordingly.	No error handlers were implemented	Error handlers were implemented partially	Error handlers were implemented fully
IR02 - Unit testing	Unit tests to test if the expected operation is performed by the flow	No unit tests were implemented	Unit tests were implemented partially	Unit tests were implemented fully
IR03 - Naming conventions	Recommended naming convention for flows, sub-flows, variables, and properties are followed	No naming convention was followed	Naming conventions were partially followed	All naming conventions were correctly followed

Appendix E

Quality Evaluation Framework Criteria

Dimension	Audit and monitorability			
Factor	Audit and Tracing			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
AA01 - Logging	All APIs must have loggers in every flow, including error flows, and every request.	No loggers were implemented	Loggers were implemented partially	Loggers were implemented fully
AA02 - Exception/Error tracking	Errors are raised and correctly logged	No error loggers are implemented	Some error loggers were implemented	All intended error loggers were implemented
AA03 - Metrics	APIs produce metrics to ensure proper monitoring	No API produce metrics	Some APIs produce metrics	All APIs produce metrics
AA05 - Health checks	All APIs must have na endpoint that returns a response to chck if the API is reachable	No API has a healthcheck mechanism	Some APIs have a healthcheck	All APIs have a healthcheck mechanism

Appendix F

Quality Evaluation Framework Criteria

Dimension	API Reusability			
Factor	Reusability			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
AR01 - Abstraction	The API's code is not repeated to perform similiar functions	No	-	Yes
AR02 - API-Led Approach	APIs are created purposefully to handle specific data	No	-	Yes

Appendix G

Quality Evaluation Framework Criteria

Dimension	API Security			
Factor	Security			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
AS01 - API Authentication	All APIs use require client-id and secret, basic authentication or another auth orization method in order to be used	No API has authenticati on	Some APIs have authentication	All APIs have authentication

Appendix H

Quality Evaluation Framework Criteria

Dimension	API Documentation			
Factor	Documentation			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
AD01 - Architecture documentation	The framework architecture is documented	Not documented at all	Partially documented	Fully documented
AD02 - Technical documentation	All APIs are documented	Not documented at all	Partially documented	Fully documented
AD03 - API Documentation	All APIs have RAML specifications when applicable	No API has a RAML specification	Not all APIs have a RAML pecification	All APIs have a RAML specification