NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Supply Chain (micro)TMS development

António Alberto Pessegueiro Gemelgo

Project presented as a partial requirement for the degree of Master of Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

SUPPLY CHAIN (MICRO)TMS DEVELOPMENT

by

António Alberto Pessegueiro Gemelgo

Project presented as a partial requirement for the degree of Master of Information Management, Specialization in Information Systems and Technologies Management

Supervisor: Vítor Santos

October 2021

ABSTRACT

The rise of technology across many verticals has necessitated the company's move to digitalization. Despite "XPTO" company a well know player on the retail and success on e-commerce internal market, they aimed at the strategy of continuous innovation to drive business growth and strengthen their position as a premium brand. They decided to move forward into digitalism inside cloud based solutions to get all the advantages of microservices architecture: optimize logistics and supply chain management, speed up the workflow and maximize service efficiency.

An agile organization is not achieved purely by shifting the focus from traditional functional/ technological oriented organizations. The new way to organize teams must reflect all the principles and right segregations of roles, which will be the most immediate and visible disruption and cutover from the traditional way of managing the IT.

In this project we aim to use agile framework with development based in house cloud microservice solution for a (micro)TMS solution/system that address the immediate needs imposed by the market in order to use it has competitive advantage.

KEYWORDS

AWS, Digitalism, Logistics, Microservices, Supply Chain Management, Transport Management System (TMS)

INDEX

TABLE OF FIGURES

LIST OF TABLES

LIST OF ANNEX

LIST OF ACRONYMS

AI - Artificial Inteligence

ALB - Application Load Balancer

AWS - Amazon Web Services

CD - Continuous Delivery

CI - Continuous Improvement

CTO - Chief Technology Officer

CTT - Correios e Telecomunicações de Portugal

ERP - Enterprise Resource Planning

HTTP - HyperText Transfer Protocol

IaaS - Infrastructure as a Service

IaC - Infrastructure as Code

IT - Information Technology

MVP - Minimum Value Product

PaaS - Platform as a Service

SC - Supply Chain

SCM - Supply Chain Management

TMS - Transportation Management System

UML - Unified Modelling Language

WhTF - Warehouse Transformation Framework

WMS - Warehouse Management System

XPTO - dummy company name

1. INTRODUCTION

Due of the 2020 pandemic context the online market gets a huge increase and importance on sales for “XPTO” company. This high demand puts the logistics platform in high pressure in terms of capacity in order to fulfil customer requests.

Some bottlenecks in terms of logistics operations were identified and in order to increase operational efficiency and more daily delivery capacity a (micro) Transports Management System solution was discussed and put in internal development to achieve customer satisfaction and therefore more business value.

1.1. BACKGROUND AND PROBLEM IDENTIFICATION

Almost all companies need to adapt the way they have implemented their business model and the companies on the retail area were not exception. In order to face the challenges of this too long pandemic period some problems need to be addressed in a matter of survival face of the unknown.

“XPTO” CTO was concerned about what needs to be changed or done, on a system management perspective, in order to the company be able to produce and deliver more goods to clients regarding the “online” boom during this pandemic period.

In an operations heavy environment such as in the retail industry, where financial cost and time taken are critical variables, such software development as microservices holds transformative potential. In such an ecosystem, which is always preoccupied with executing the processes fast, there is a tendency to take the path-of-least-resistance when managing information and keeping records. These shortcuts ultimately end up increasing the cost and time involved. Additionally, this increases the probability of loss of shipments — which causes loss of money, time to the customers and is thus a negative influence on the organizations relationship with its customers.

“XPTO” company wants to move completely into digitalism and in order to achieve that goal an internal evaluation was performed by external consulting entity and is ongoing with high focus on supply chain.

In terms of digital perspective, “XPTO” is view as e-commerce company, processes are simple by design, standardized and integrated for maximum efficiency, using real-time information for make insightful decision. Silo thinking will not be allowed and “XPTO” business must be ready to scale both in products, services provided and in volume. Must be agile on testing product and business models with minimal costs, every cost should be accountable and tracked in real time to ensure maximum value delivered to client and Information is at the core of decision making ensuring “XPTO” optimizes its processes and consistently exceeds customer expectations.

Business principles are enabled by IT principles that steer IT architecture design and build. The guidelines focus on sustainability through feasible projects, continuous development and operation, adopt security by design by considering implications of design beyond an immediate project and use of Design-Thinking to onboard user groups in planning, development, implementation and assessment for inventive solutions.

IT principles focus on customer experience and build client relationships, focus on end-to-end technical capabilities, leverage AI to make intelligent decisions, cloud-first and decomposable and reusable services. A new architecture requires a new compromise between IT and Business in order to leverage the architecture decoupling to achieve greater business speed where IT main rule is to provide platform where independent teams can develop capabilities with minimum impact between them.

New IT paradigm is driving fundamental change across the IT operating model & organization by Moving to Digitalism with Agile methodology:

- Growth mindset;
- Focus on speed-to-value and innovation (fail fast);
- Integrated, cross-functional teams (no barriers);
- Smaller, agile teams;
- Depth and breadth of experience/skills ("T-shaped").

Achieving a true agile and Business aligned technology function implies considering four key focus areas.

- Faster, iterative processes (Agile);
- Lean product management and small batching (Minimum Value Product (MVP));
- Integration focus;
- Automated coding, testing, deployment;
- Modern engineering (microservices, cloud, big data, APIs, containers, loosely coupled architectures);
- Collaborative decision-making.

Achieving a true agile and Business aligned technology function implies considering four key focus areas.

1) *Agile/ Flexible Operating Model* - From change-run-manage to delivering at right speed for the business

- Leverage multiple delivery approaches and methods that seamlessly coexist;
- Focus on service design and assembly rather than code development;
- Dev Ops, automated tools, AI, and analytics embedded across delivery and operations.

2) *Product / Services Based Organization* - From functional alignment to service orientation and innovation

- Organization structured around the services;
- Flatter organizations focused on dynamic business needs and skill alignment to enable collaboration and agility;
- Empowered co-located teams with shared objectives tied to customer journey.

3) *Tech + Business Governance* - From IT cost focus to Business value and transformation

- Driven by enterprise value; mindful of technology transformation / green field filling;
- New empowered, distributed decision-makers;
- Shift to flexible, consumption-based cost structure (Capex to Opex);
- With IT and the Business converging, there's a need for a new governance model.

49 *Talent Ecosystem* - From point-to-point provider management to ecosystem partnerships

- Tailored sourcing models; right sourcing mix for a liquid workforce;
- Forging of countless connections, IT as a talent orchestrator;
- Consolidated outsourcing shifts to ecosystem sourcing;
- Blurring lines between the organization and the ecosystem partners.

An agile organization is not achieved purely by shifting the focus from traditional functional / technological oriented organizations. The new way to organize teams must reflect all the principles and right segregations of roles, which will be the most immediate and visible disruption and cutover from the traditional way of managing the IT.

Refocus on flexibility and agility is key to shift and operate at the pace of the business and market. The organizational alignment and orientation will be key to internally organize all resources and capabilities. Shifting the focus of the technological organization from functional-process-technology to customer or service/product oriented is a decision that will profoundly impact the way IT delivers services.

New business trends arrive with the 21st century and with the continuous grow of digitalism solutions everything is at a "click". The change of consumer's behavior towards how they purchase goods increase the pressure on the logistics supply chain in order to fulfil the demand.

In this pandemic phase, since the 2nd trimester of 2020, the online segment gains new boost and the companies need to be able to adapt rapidly in order to be on the front line for gaining new businesses and competitive advantage in the market.

The traditional supply chain management approach is not able to address on a short time of period the required changes needed in order to exploit these new market opportunities.

1.2. STUDY RELEVANCE AND IMPORTANCE

This project will show clearly how companies can adapt themselves in order to reach more clients and faster. Changes on big companies tend to be slow because of the hierarchical structure implemented but this pandemic phase give business, operations, IT and administration a real opportunity to face the supply chain stresses due to the high demand and adapt new cloud based solutions in order to gain more productivity and delivery capacity of the goods.

The time is now and microservices based software are ideally for industries such as retail and others. It allows different entities to carry out their roles in the business processes with greater ease, promotes efficient coordination among them, helps lay out a clearer matrix of responsibility and reduces chances of conflict. It enables organizations to accomplish these ideals without requiring any extravagant additional cost or expertise.

Regardless of the size of the companies, replacing existing legacy systems with a suite of lightweight microservices or taking them up as an entirely new layer might turn some heads but might also be a turning point.

“XPTO” technical architecture blueprint has digital decoupling as a core design principle. Since we’ve modularized the application as UI and backend microservices (based on either features or functions) we’ve increased speed to market and improved developer productivity. We now have distinct, atomic units of work that enable small incremental changes to be made and rapidly deployed.

Microservices will be wrapped around common services that will standardize the overall architecture control, monitoring and security. The business layer of all microservices must be surrounded by an architecture framework, which will have the responsibility to abstract the programmer from any configuration and/or connection to the different tools that they are integrated, such as monitoring, logging, caching, database, etc...

The PaaS where microservices reside provides additional services as a baseline namely:

- Scaling Management;
- Container Management and Hot Configuration;
- API Gateway, Routing and Load Balancing (Lightweight);
- Service Discovery and Mashing;
- DevOps;
- Monitoring, Telemetry and Messaging.

Microservices will have independent development, deployment and maintenance. During the last months "XPTO" E-commerce is growing up. In order to make sure that all orders are delivered quickly, we need to find and integrate alternative carriers to do the home deliveries.

1.3. OBJECTIVES

Develop (micro)TMS and middleware solution with the focus on production efficiency and able to respond to high demand volume eliminating some constrains identified on supply chain operations.

CTT will be the first new carrier that we will work fully integrated with this new solution in order to address the below problem:

Right now the CTT carrier is already being used through a solution that is wasting very operational time and on a daily basis is having several problems that put in risk goods delivery. The actual solution implemented requires several manual steps in order to work properly.

CTT has one app which is used to generate a couple of documents/files when the user closes the shipping (cut off). This info needs to be consumed by some "XPTO" internal systems and also needs to be sent to CTT. In order to achieve that, below are all the manual steps required to do in order to goods leave the warehouse for CTT delivery:

- Manual Export file with all treated CTT orders;
- Import previous file to CTT application installed on team leaders operations PC;
- Scan code(s) returned from imported file to print CTT label;
- Place label on the package/volume;
- Export file from CTT application with orders and Tracking Numbers;
- Import previous file into "XPTO" systems.

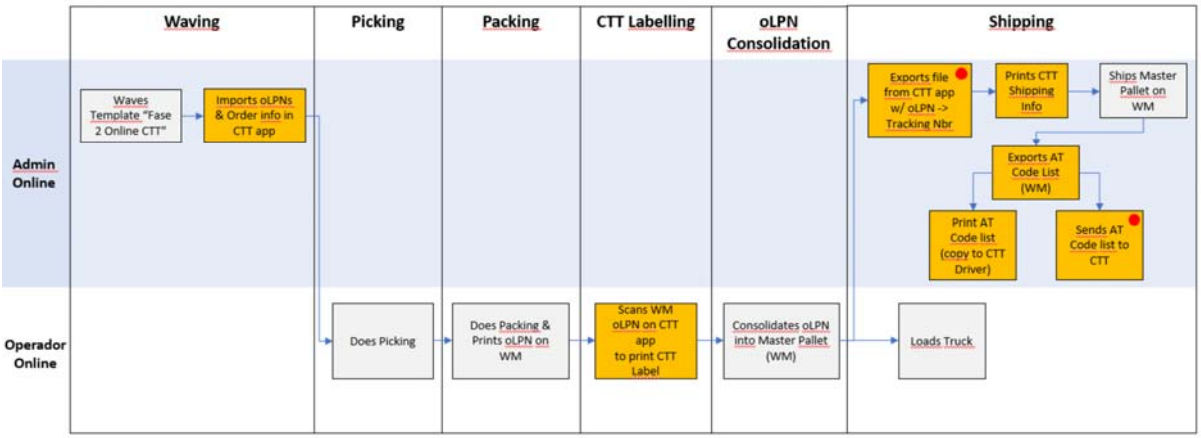


Figure 1 – CTT process (Source : own)

All the above orange steps will be replaced by the solution to be delivered in order to be fully automated and ready to receive more carriers if needed.

In terms of logistics operations this will be a great achievement because they only need to take care the logistic process of packing (label printing and tracking id generation) and expedition process (shipping flow) on a standard way independent of the carrier.

2. WORK PLAN

2.1. PROJECT DEFINITION

As the topic of the article is regarding a real project, before starting the design phase of the project some investigation was done in order to be able to do wise and aware decisions.

Some literature review aimed into identifying central aspects of supply chain management (SCM) into the cloud in order to conceptualize SCM and Cloud and identify key terms, a preliminary literature review was conducted by searching for the terms “Supply Chain Management”, “AWS”, “Digital Transformation” and “Transport Management System” in Google and Google Scholar. A review of journal articles, conference papers, books and edited volumes was also performed in order to search the most updated information available.

This analytical study is based on relevant information already available on the subject and analysis of relevant data. There are some limitations in such approach because this means that opinions or views not available during the research period will not be taking into account and the actual organization architecture was also taken in consideration regarding pros and cons during design phase. It was also taken into account the actual architecture blue print from "XPTO" and if it fit the requirements of the project.

2.2. RESOURCES & TOOLS

2.2.1. UML

According with UML. (n.d.)., Unified Modelling Language (UML) is a standardized modelling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems.

UML notation will be used to create all UML diagrams through *drawio* and *mermaid* tool. There are several important diagrams to represent the system:

- Use Case Diagram – Use cases enable the possibility to relate what we need from a system and how the system delivers on those needs.
- Activity Diagram– They are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.
- State Machine Diagram – They represent the permitted states, transitions and the events that effect these transitions.
- Sequence Diagrams – It shows how the objects interact with others in a particular scenario of a use case.

2.2.2. Cloud Resources

AWS cloud provides a broadest range of scalable, flexible infrastructure services that we can select to match our workloads and tasks. This gives us the ability to choose the most appropriate mix of resources for our specific applications.

2.2.2.1.AWS Application Load Balancer

A load balancer serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets in multiple Availability Zones. This increases the availability of the application. In order to achieve that is required to add one or more listeners to our load balancer.

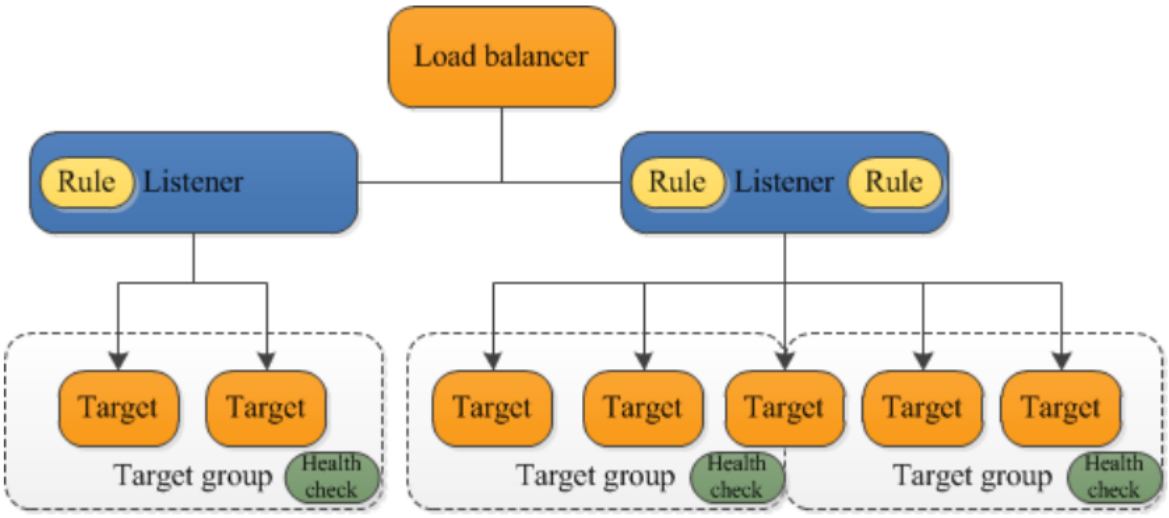


Figure 2 – ALB implementation (Source: What is an Application Load Balancer? - Elastic Load Balancing. (n.d.))

2.2.2.2.Kong API Gateway

Kong is one open source API gateway built on top of a lightweight proxy, the Kong Gateway delivers unparalleled latency performance and scalability for all our microservices applications regardless of where they run.

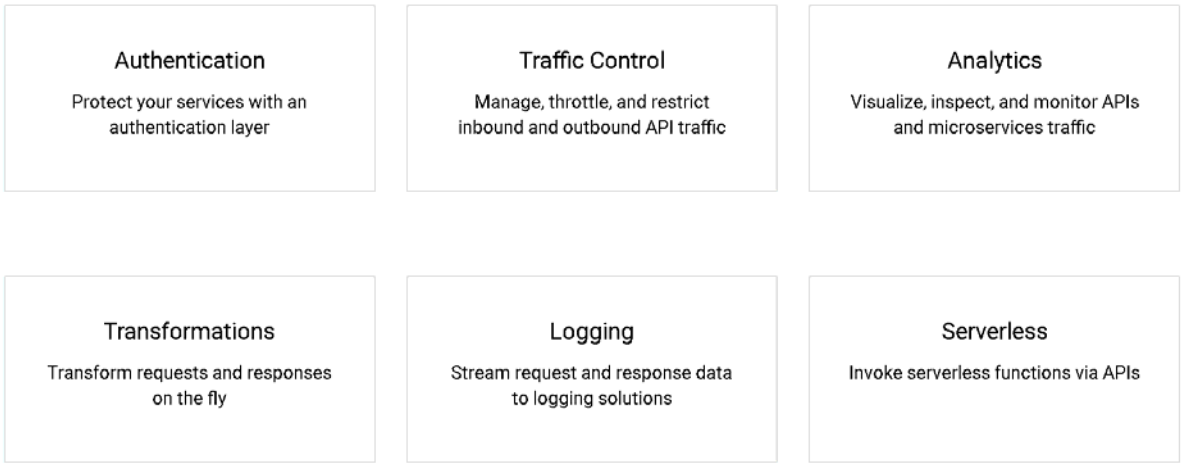


Figure 3 – Kong API key features (Source : Kong Inc.)

2.2.2.3.AWS DynamoDB

According with What Is Amazon DynamoDB? - Amazon DynamoDB. (n.d.), Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets us offload the administrative burdens of operating and scaling a distributed database and also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

NoSQL database systems use alternative models for data management, such as key-value pairs or document storage. In a NoSQL database such as DynamoDB, data can be queried efficiently in a limited number of ways, outside of which queries can be expensive and slow.

In DynamoDB, we design our schema specifically to make the most common and important queries as fast and as inexpensive as possible. Our data structures are tailored to the specific requirements of our business use cases.

2.2.2.4.Apache Kafka

According with What is Apache Kafka? | AWS. (n.d.), Apache Kafka is a distributed data store optimised for ingesting and processing streaming data in real-time. Streaming data is data that is continuously generated by thousands of data sources, which typically send the data records in simultaneously. A streaming platform needs to handle this constant influx of data and process the data sequentially and incrementally.

2.2.2.5.AWS Lambdas

According with What is AWS Lambda? - AWS Lambda. (n.d.), Lambda is a compute service that lets us run code without provisioning or managing servers. With AWS Lambda, we can run code for virtually any type of application or backend service - all with zero administration. We can use AWS Lambda to run the code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run the code in response to HTTP requests using Kong API Gateway, ALB or invoke our code using API calls made using AWS SDKs.

2.2.2.6.AWS State Machines and Step Functions

A state machine consists of a collection of states that can do work (Task states), determine to which states to transition next (Choice states), stop an execution with an error (Fail states), and so on.

Step Functions is based on state machines and tasks. A state machine is a workflow and a task is a state in a workflow that represents a single unit of work that another AWS service performs. Each step in a workflow is a state.

2.2.2.7.AWS CloudWatch

CloudWatch collects monitoring and operational data in the form of logs, metrics and events, providing a unified view of AWS resources, applications, and services that run on AWS.

2.2.2.8.Kibana

According with What is Kibana? – Amazon Web Services. (n.d.), Kibana is an open-source data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support. Also, it provides tight integration with Elasticsearch, a popular analytics and search engine, which makes Kibana the default choice for visualizing data stored in Elasticsearch.

2.2.2.9..NET Core

.NET Core is a Microsoft programming language where the developer writes, compiles and runs C# code in any operating system decoupled from the Microsoft ecosystem, in an open-source and cross-platform environment.

2.2.2.10.Grafana

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs and alerts for the web when connected to supported data sources.

2.2.2.11.Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit. According with Overview | Prometheus. (n.d.) the main features are:

- a multi-dimensional data model with time series data identified by metric name and key/value pairs;
- PromQL, a flexible query language to leverage this dimensionality;
- no reliance on distributed storage; single server nodes are autonomous;
- time series collection happens via a pull model over HTTP;
- pushing time series is supported via an intermediary gateway;
- targets are discovered via service discovery or static configuration;
- multiple modes of graphing and dashboarding support.

2.3. PROJECT SCHEDULING

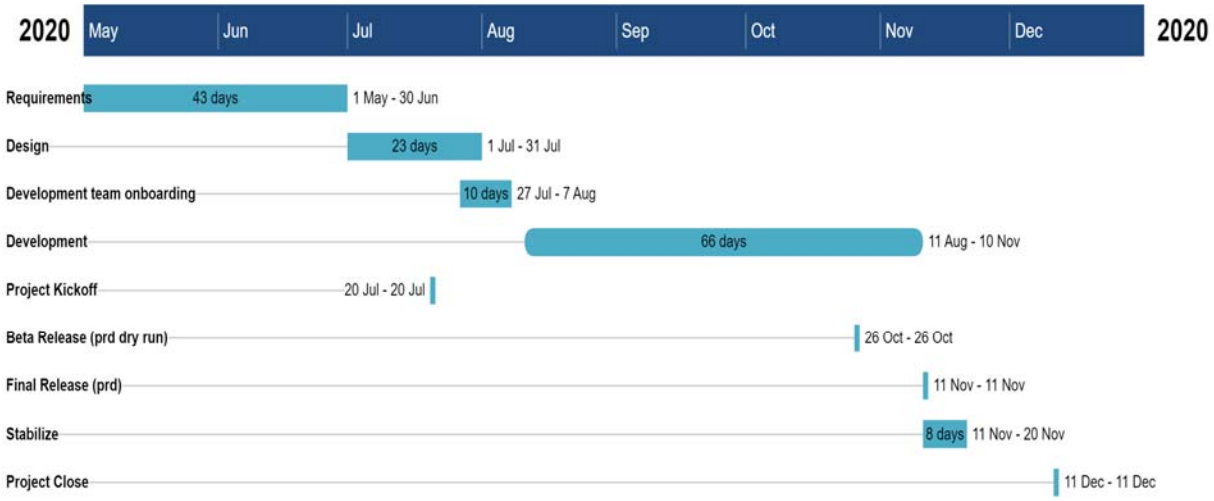


Figure 4 – Project chart (Source: own)

3. THEORECTICAL FRAMEWORK

3.1. DIGITAL TRANSFORMATION

Digital transformation is defined by Stolterman & Fors (2004), as “the changes associated with the application of digital technology in all aspects of human society”.

According with Gebayew et al (2018), Digital transformation is composed by “digital” and “transformation” terms where the first one refers in today’s world to the emerging technologies while latter one describes the new types of innovation and creativity enabled by the digital technologies. Furthermore, according with Bharadwaj et al (2013) digital transformation is defined as “an organizational strategy formulated and executed by leveraging digital resources to create differential value.”

Digital transformation involves using digital technologies to remake a process to become more efficient or effective. In order to achieve that many different technologies are used but the hottest topics right now are cloud computing, the Internet of Things, big data, and artificial intelligence.

Changing business processes and corporate culture are just as vital to the success of these initiatives. A genuine digital transformation project involves fundamentally rethinking business models and processes and keep in mind that true transformation is a journey, not a destination. Digital transformation remains a slippery concept that involves the delivery of value to the business and its customers in new ways.

Digital transformation as stated in What is digital transformation? - Digital Transformation Definition - Citrix. (n.d.) plays a key role in aspects such as:

Customer experience: Consumers today have more choices than ever, which means the stakes are high for businesses to not only deliver innovative products or services but to deliver meaningful interactions and experiences that delight customers and foster brand loyalty. In the digital marketplace, consumers are using the emerging technologies to be more informed about the product and service offerings and to decide whom to trust, what to buy and where to make the purchase (Berman, 2012).

Employee experience: Digital transformation can help organizations provide not only the tools that people need but also instant access to everything they need from anywhere.

Process optimization: Streamlined workflows, digital processes, and automated tasks are where organizations can always create efficiencies.

Product digitization: Digital transformation not only helps companies stay on the cutting edge of technology, but also creates an agile infrastructure necessary to constantly innovate and adapt to rapid change and consumer demands.

We can assume that the common challenges of digital transformation are:

Failure to lead with a strategy: Digital transformation should begin with a vision and it’s imperative to keep in mind business goals and objectives and how a digital transformation strategy can support them.

Lack of leadership buy-in: Any business transformation, digital or otherwise, is more likely to succeed when leadership is engaged and leadership is more likely to be engaged in initiatives that are directly aligned to the business strategy. When proposing any new project, it's important to demonstrate how it supports the business and will impact the bottom line. With leadership buy-in secured, you can avoid any budget challenges that will impact your project.

Planning in a silo: Leaders require insight into end-user experiences to ensure that the solution works properly and does not introduce new challenges and meet the company's need.

3.2. SUPPLY CHAIN MANAGEMENT

Today, individual firms no longer compete as independent entities with unique brand names, but rather as integral parts of a network. As such, the ultimate success of a firm will depend on its managerial ability to integrate and coordinate the intricate network of business relationships necessary to deliver products to customers (Lambert & Cooper, 2000).

In shorter words, Ballou (2004) states that the term supply chain refers to all those activities associated with the transformation and flow of goods and services, including their attendant information flows, from the sources of raw materials to end users.

According to Chopra & Meindl (2007), a supply chain consists of all parties involved, directly or indirectly, in fulfilling a customer request, as well as all the functions necessary in this task. The supply chain includes not only the manufacturer and suppliers, but also transporters, warehouses, retailers, and even customers themselves. A supply chain is dynamic and involves the constant flow of information, product, and funds between different stages.

Min (2015) refers to supply chain as an integrated system that synchronizes a series of interrelated business processes in order to: create demand for products; acquire raw materials and parts; transform these raw materials and parts into finished products; add value to these products; distribute and promote these products to either retailers or customers; facilitate information exchange among various business entities (e.g., suppliers, manufacturers, distributors, third-party logistics providers, and retailers). Furthermore, a supply chain is traditionally characterized by a forward flow of products and a backward flow of information.

While supply chains have existed for a long time, most organizations have only paid attention to what was happening within their boundaries. Few businesses understood, much less managed, the entire chain of activities that ultimately delivered products to the final customer. The result was disorganized and often ineffective supply chains. To capture the synergy of cross-functional and cross-organizational integration and coordination transversely to the supply chain and to subsequently make better strategic decisions, it is extremely important to plan, control, and design a supply chain as a whole.

In this sense, Supply Chain Management (SCM) is the set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that goods are produced and distributed at the right quantities, to the right locations, and at the right time, in order to minimize system wide costs while satisfying service level requirements (Simchi-Levi et al 2008). SCM represents a conscious effort by the supply chain firms to develop and run supply chains in the most effective and efficient ways

possible. The inherent activities cover everything from product development, sourcing, production, and logistics, as well as the information systems needed to coordinate these activities.

All definitions have in common that SCM refers to integration of internal and external key processes that goes from end-users through suppliers that provide the products, services and information in order to add value to customers and stakeholders.

As explained above, a Supply Chain (SC) is a network of interrelated entities that collaborate in order to provide products to customers. SCM is, then, the management of those entities in the best possible way, with the goal of obtaining significant benefits. In this sense, a brief description of what are the main goals of a SC and the potential of an efficient SCM is presented below.

The traditional objective of a Supply Chain, according to Shapiro (2006) is to minimize total supply chain cost to meet fixed and given demand. Thus, increasing the difference between what the final product is worth to the customer and the costs the supply chain incurs in filling the customer's request, which ultimately leads to higher profitability (Chopra & Meindl, 2007).

According with Handfield, R. (2020), SCM is the active management of supply chain activities to maximize customer value and achieve a sustainable competitive advantage. It represents a conscious effort by the supply chain firms to develop and run supply chains in the most effective and efficient ways possible. Supply chain activities cover everything from product development, sourcing, production, and logistics, as well as the information systems needed to coordinate these activities.

The organizations that make up the supply chain are “linked” together through physical flows and information flows. Physical flows involve the transformation, movement, and storage of goods and materials. They are the most visible piece of the supply chain. Information flows allow the various supply chain partners to coordinate their long-term plans, and to control the day-to-day flow of goods and materials up and down the supply chain.

Benefits of supply chain management

Supply chain management produces benefits such as new efficiencies, higher profits, lower costs and increased collaboration. SCM enables companies to better manage demand, carry the right amount of inventory, deal with disruptions, keep costs to a minimum and meet customer demand in the most effective way possible. These SCM benefits are achieved through choosing effective strategies and appropriate software to manage the growing complexity of today's supply chains.

Supply chain management activities can improve customer service. Done effectively, they can ensure customer satisfaction by making certain the necessary products are available at the correct location at the right time. By increasing customer satisfaction levels, enterprises can build and improve customer loyalty.

Two complex processes play important roles in most of the major steps of SCM: inventory management and logistics. Inventory management consists of various techniques and formulas for ensuring adequate supply for the least expenditure of time and resources. Manufacturers are faced

with a variety of inventory management issues, many of which involve coordinating demand planning with inventory at both ends of the production process.

Logistics is everything having to do with transporting and storing goods from the start of the supply chain, with delivery of parts and materials to manufacturers, to delivery of finished products to stores or direct to consumers and even beyond for product servicing, return and recycling -- a process called reverse logistics. Inventory management is threaded throughout the logistics process.

The most basic version of a supply chain includes a company, its suppliers and the customers of that company. The chain could look like this: raw material producer, manufacturer, distributor, retailer and retail customer.

Logistics vs. supply chain management

The terms supply chain management and logistics are often confused or used synonymously. However, logistics is just one component of supply chain management. It focuses on moving a product or material in the most efficient way, so it arrives at the right place at the right time. It manages activities such as packaging, transportation, distribution, warehousing and delivery.

In contrast, SCM involves a more expansive range of activities, such as strategic sourcing of raw materials, procuring the best prices on goods and materials and coordinating supply chain visibility efforts across the supply chain network of partners, to name just a few.

Technology is critical in managing todays supply chains, and every major supply chain management process has a software category dedicated to it. Commonly used SCM modules include a TMS for managing the transport and storage of goods, a WMS for all of the activities inside warehouses and distribution centers and an order management system, to handle processing of customer orders through WMS, ERP and TMS systems, at all stages of the supply chain.

3.3. CLOUD COMPUTING

3.3.1. Cloud vs on-premise

Small and medium-sized enterprises might want to keep their business small or to grow it. When they start to grow it may get harder to manage IT infrastructure. With a bigger company, more on-premise hardware is needed and it can grow gradually or exponentially and takes usually a long time to return on investment (Watson, L. & Mishler, C., 2014).

With cloud computing there is no need to take care of our infrastructure, it is provided from a cloud provider in a form of Infrastructure as a Service (IaaS) or Platform as a Service (PaaS). Cloud often offers to pay as you go, which means it does not involve large initial investment (Gannon, D. et al, 2017).

3.3.2. Cloud-native

There are many key ideas behind being Cloud-native, one of them is specific design patterns that became very successful while creating cloud applications. Most frequent arguments of cloud-native are:

- Cloud-native applications can operate on a global scale. The ordinary web application can be accessed anywhere in the world through the internet. Cloud-native application has replicas of servers and datacenters around the whole world so that accessing application results in minimal latencies. This approach creates very robust applications.
- Cloud-native applications have to scale well with many concurrent users. Assumption here that application can horizontally scale automatically. That approach requires careful observation of synchronization and consistency in distributed systems.
- Applications are built on assumption that infrastructure is unstable. Even though one zone of servers will crash down because of some natural disaster the application will still run in a different place, so the user does not realize that there is trouble.
- Upgrading or testing Cloud-native applications do not affect end users.
- Security must not be forgotten, cloud-native applications are built of many small components and these components cannot hold sensitive data. Access control needs to be managed at multiple levels.

At first to become cloud-native, IaaS replaces on-premise infrastructure with virtual machines running in the cloud. It was very difficult to engineer scalability and security at the same time with only on-premise solutions.

The first major design pattern for cloud-native applications was microservice architecture. This architecture relies on dividing application to small independent components and it easy to scale and reliable. All microservices should be designed for constant failure and recovery and through containerization it is possible to encapsulate each microservice instance so that it can be easily manipulated.

With all this, it is possible to create a well-developed cloud-native application based on microservice architecture (Gannon, D. et al, 2017).

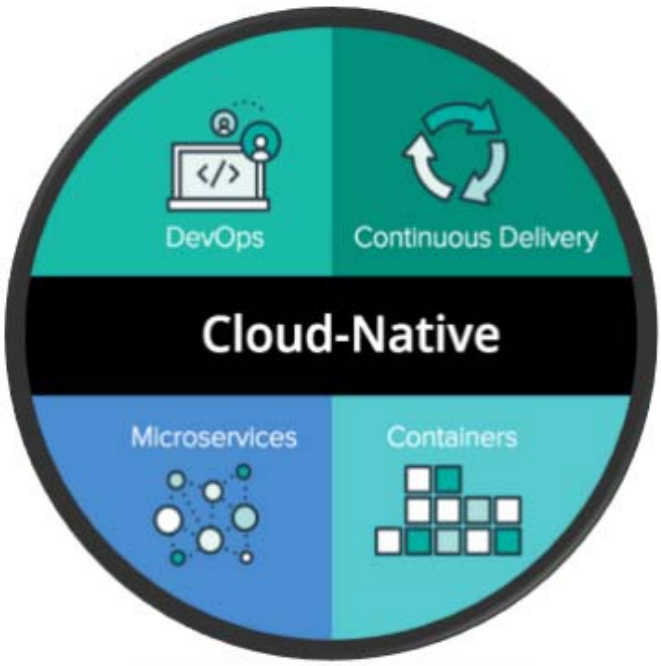


Figure 5 – Basic principles of cloud-native development (Source: Basic principles of cloud-native development. (n.d.))

3.3.2.1.Full stack example of Cloud-Native application

Before creating a new cloud-native application, it is good practice to choose proper tooling, there are a lot of tools for different parts of the application. Now the cloud market offers many solutions, most popular are Amazon web services (AWS), Microsoft Azure, Google cloud platform, VMware etc.

After the selection of cloud provider, it is required a usage of provisioning Infrastructure as Code (IaC) tool to create resources for a project. After choosing the IaC provisioning tool such as Terraform or CloudFormation, it is required to decide what will be the infrastructure configuration tools to use like Chef, Puppet or Ansible. After choosing the proper tools, infrastructure is prepared for serving a given purpose.

To be able to deploy to the cloud, developers tend to use runtime environments in which application runs. Those environments are usually created with Container Engines such as Docker, which allows to enclose application with needed components.

Orchestration and Management is the next step of being cloud-native, tools like Kubernetes or Docker Swarm are used to manage container clusters for easy orchestration across multiple hosts. They provide load balancing, scheduling of containers etc.

Many languages support microservice architecture and the one to be used it will be chosen by the development team. The code is being shipped to runtime services with Continuous Improvement/Continuous Deploy (CI/CD) tools such as Jenkins, Git Lab and others.

Cloud-native application must have monitoring and all modern monitoring tools support monitoring of containers and microservices (Jog, C.)

3.3.3. Cloud Providers

There are several cloud providers in the market like Amazon Web Services, Microsoft Azure and Google Cloud Platform and all of them provide similar options and differ a bit from each other's.

Amazon Web Services

Amazon Web Services (AWS) offer a vast number of services and tools to work regarding compute, storage, database, analytics, networking, mobile, IoT and more. AWS is the most mature provider focus on a public cloud rather than a hybrid cloud or private cloud but is weakness is the cost of resources. This implies that sometimes it is not the best choice for an enterprise customer (Carey, S.).

Microsoft Azure

Azure offers exceptional cloud infrastructure and believes that hybrid cloud is important, so it is supportive of private datacenters (Carey, S.).

Google Cloud Platform

The main benefits compared to other are expertise and industry-leading tools in deep learning and artificial intelligence, machine learning and data analytics. His potential lays in containers since Google developed the Kubernetes which is becoming an industry standard (Carey, S.).

3.4. TRANSPORT MANAGEMENT SYSTEM

Transportation in Supply Chain

Transportation means are fundamental in every society, transportation networks are essential for every supply chain and constitute the basis of any economic structure, by allowing the efficient distribution of goods.

One of logistics biggest points of focus is precisely the movement of physical material flows, whatever they are, across the network. Thus, the selection of transportation modes, the contracting of transportation services providers and the contractual management of those providers, whichever materials they are supposed to transport, is critical in terms of Logistics activities. Furthermore, the planning of transportation routes, the choice of vehicles and the slotting activities are also in the scope of the transportation logistics and transportation management (Crespo de Carvalho, 2010).

The goal of making the SC more agile also transfers a big share of the costs to the transportation sector, while trying to decrease costs in inventories and warehouses. With transportation being so essential in a SC, it cannot be managed in an isolated manner, since options about the mode and the type of transportation solutions have a significant impact in the cost structure and in the company's ability to react to demand, as well as noteworthy repercussions throughout the whole SC, which can improve or compromise, at last, customer satisfaction. Only a good coordination between each component would bring the benefits to a maximum (Tseng, 2005).

In sum, transportation is the glue that holds the supply chain together, and that allows the member organizations to operate efficiently and effectively as a system (Coyle et al, 2011).

Transportation Management System (TMS)

A TMS is specialized software for planning, executing and optimizing the shipment of goods. Users perform three main tasks on a TMS: Find and compare the rates (prices) and services of carriers available to ship a customer's order, book the shipment, then track its movement to delivery. The broader goals of using a TMS are to improve shipping efficiency, reduce costs, gain real-time supply chain visibility and ensure customer satisfaction.

Usually, orders come in automatically from ERP or Order Management Systems (OMS) that are integrated with the TMS. In addition, a TMS will sometimes be integrated with a warehouse management system (WMS) to enable better coordination of the tasks that occur at the interface of warehouses and freight shippers, such as palletization of goods, labor scheduling, yard management, load building and cross-docking.

The three main SCM systems -- ERP, WMS and TMS -- each have important but mostly distinct roles to play in processing orders.

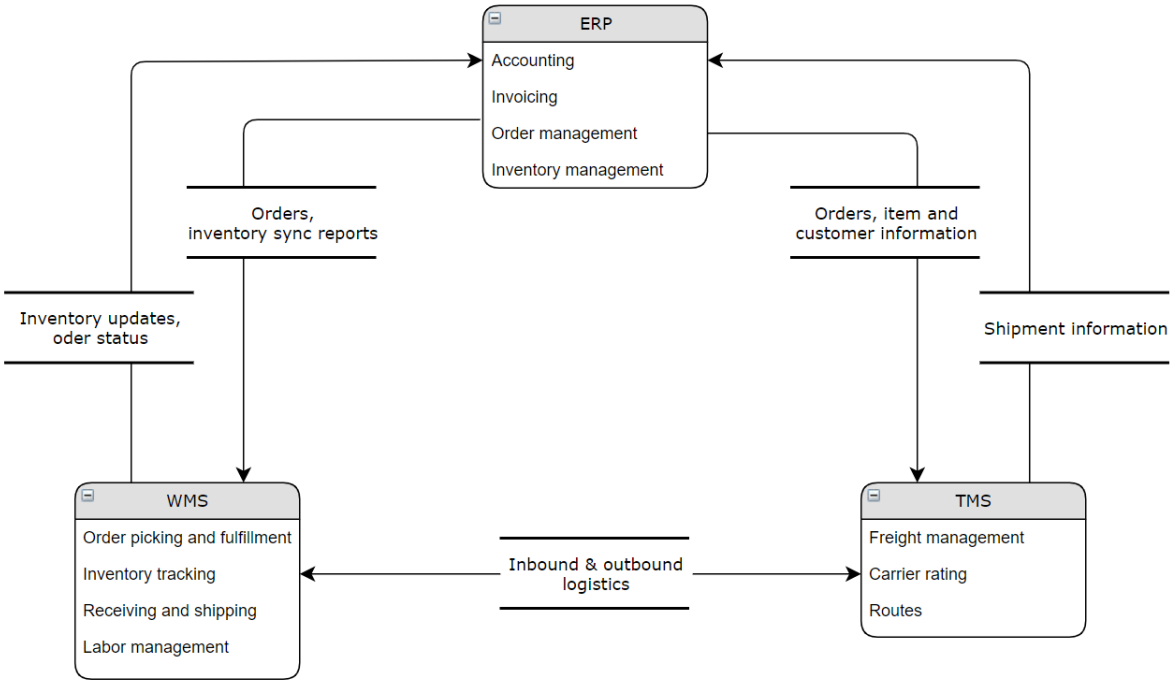


Figure 6 – SCM main systems (Source : own)

The ERP handles the accounting and most of the invoicing, order management and inventory management. The WMS role is to help users manage the fulfillment, shipping and receiving tasks in the warehouse or distribution center, such as "picking" goods from shelves for shipment or putting received goods away. Its role in inventory is to track the inventory data that comes in from barcode readers and RFID tags and update the inventory management module in the ERP system to ensure it has the latest information. Synchronizing the inventory data in both places is another important use of the integration link between the ERP and the WMS.

The ERP outputs the order information the TMS needs to prepare and execute shipments. Besides basics like customer name and address, the data stream from ERP also includes detailed item information to ensure the right products are shipped. The TMS returns the shipment details that the ERP needs for its accounting and order management functions, such as the tracking number, carrier name and costs.

Running in the cloud has obvious advantages in easing connectivity between TMS users, carriers, customers and supply chain partners and potential savings in IT labor and infrastructure.

A TMS acquires, stores and updates the rates that carriers charge for shipping, often over the internet in real time. The number of carriers in a TMS can reach the tens of thousands.

A TMS also enables users to execute the major actions of freight management, including booking of shipments with carriers. Real-time visibility into the movement of freight throughout the transportation network makes it possible to track the shipments and share that information with customers and suppliers.

While most TMS features focus on execution, much of the power of the system comes from the tools it provides for planning and optimizing the shipping process. It provides data and analytics on critical factors, such as price, service level and transit time, to enable users to choose the carriers and routes most likely to transport goods the fastest and cheapest.

Transportation and logistics management are inherently complex endeavors that require substantial paperwork for business-to-business (B2B) transactions, regulatory compliance and auditing. Accordingly, a TMS must have sufficient administrative features to support the documentation and financial reporting requirements.

A TMS must also handle settlement, a more complex process that requires documenting certain freight milestones and metrics before payment can be made, such as proof of delivery, pickup and time in transit. Data collected during the settlement process is fodder for the TMS performance management and optimization processes. TMS users can search settlement data for clues to customer demand and capacity utilization and to negotiate special pricing for factors such as loading speed and time of day.

Transportation management software bring benefits such as reduced distribution and warehouse costs through better fleet management, labor and space use, and coordination between the transportation and fulfillment functions; higher customer satisfaction from a more responsive shipping process; ability to track and monitor the lifecycle of orders and shipments in real time; reduced administrative costs and invoicing errors from automating freight payment and auditing processes.

3.5. SOFTWARE DEVELOPMENT

3.5.1. Concept

Software development goal is the creation of a high-quality software. This process is a set of planned tasks that will originate the final product, the denominated software (Pressman, 2010; Sommerville, 2011). On software development we have one important concept, that supports his development, denominated *framework.*

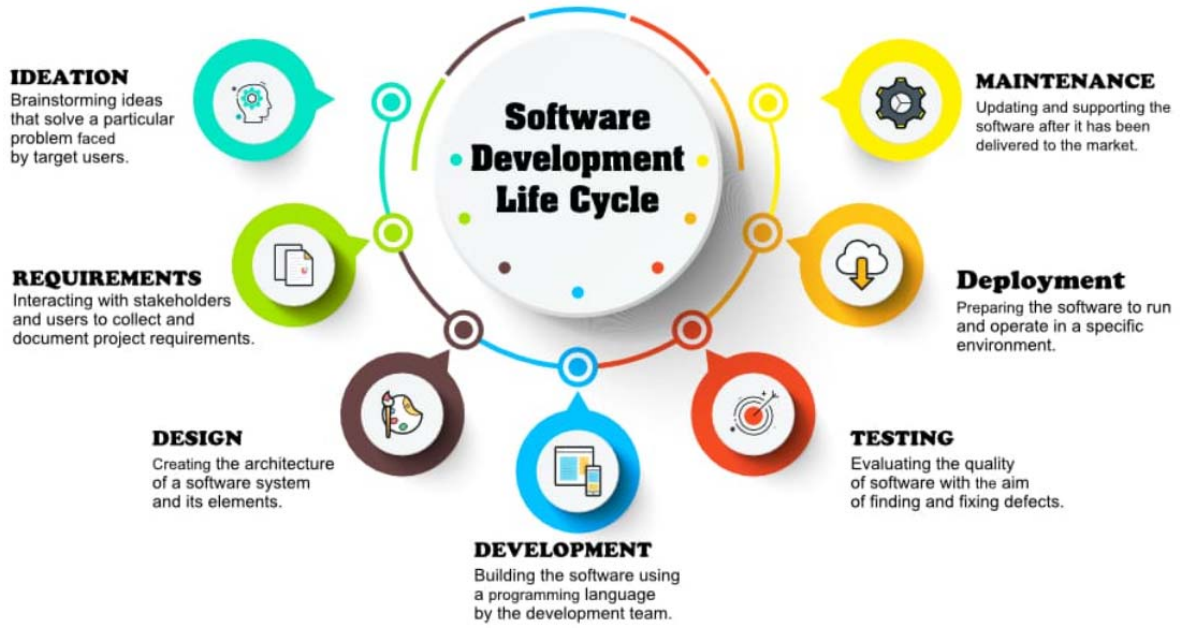


Figure 7 – Software Development Life Cycle phases (Source: Software Development Life Cycle phases. (n.d.))

According with Montoya, M. (2017), the SDLC phases can be described as:

- Requirements: Assessing scope of the system requirements and the overall project.
- Architecture and Design: Developing an understanding of the solution from a technical perspective, creating a high-level design of modular components and their interactions, and setting standards for how common technical issues should be resolved.
- Development: Producing code in an environment specific to the culture of the project. Tasks are assigned according to individual skills, and development continues until goals or milestones are reached.
- Testing, Delivery and Feedback: Testing of individual component should be ongoing, with application-level testing towards the end of the project — ideally, involving customers to confirm that requirements have been met, or to identify changes that must be made.

3.5.2. Software Development Methodology

3.5.2.1.Traditional Methodologies

Software development is a highly complex activity. It is characterized by variable requirements, the need for specialized and diverse skills, changeable and sophisticated technology used to develop and deploy software, and difficulty in management of the people who deal with such complexity every day.

Waterfall

Vijaya, D. (2013) identifies the essential nature of the traditional software development life cycle as follows:

• The goals are to thoroughly understand user needs, craft a solid design, develop software flawlessly, and implement a functional system that satisfies user needs.

• There is a heavy emphasis on thorough planning to deal with risks.

• It is based on the principles of hard-systems thinking — identifying alternate ways of reaching the desired state (S1) from the initial state (S0) and choosing the best way to achieve it (S0 – S1).

• Such an approach assumes that problems are well defined and that an optimum solution can be arrived at by extensive, up-front planning.

• It also assumes that the processes are predictable and can be optimized and made repeatable.

• It is also based on the assumption that processes can be adequately measured and that sources of variations can be identified and controlled during the development life cycle.

• In summary, the traditional software development life cycle is highly process-centric.

Based on the above understanding of systems development, organizations adopt a management style that is:

- Command-and-control-based, with a set hierarchy. Therefore, these are predominantly mechanistic organizations geared for high performance in a stable environment.
- Characterized by high formalization and standardization. People with different specializations are assigned roles for producing defined outcomes. In addition to this, they also produce a significant amount of documentation that explains the software and its technical and design specifications.
- Notable in that though customers play an important role, their participation is at maximum only during the specification and implementation stages.

The waterfall approach emphasizes a structured progression between defined phases. Each phase consists on a definite set of activities and deliverables that must be accomplished before the following phase can begin. The phases are always named differently but the basic idea is that the first phase tries to capture What the system will do, its system and software requirements, the second phase determines How it will be designed. The third stage is where the developers start writing the code, the fourth phase is the Testing of the system and the final phase is focused on Implementation tasks such as training and heavy documentation.

However, in engineering practice, the term waterfall is used as a generic name to all sequential software engineering methodology (Awad, 2005).

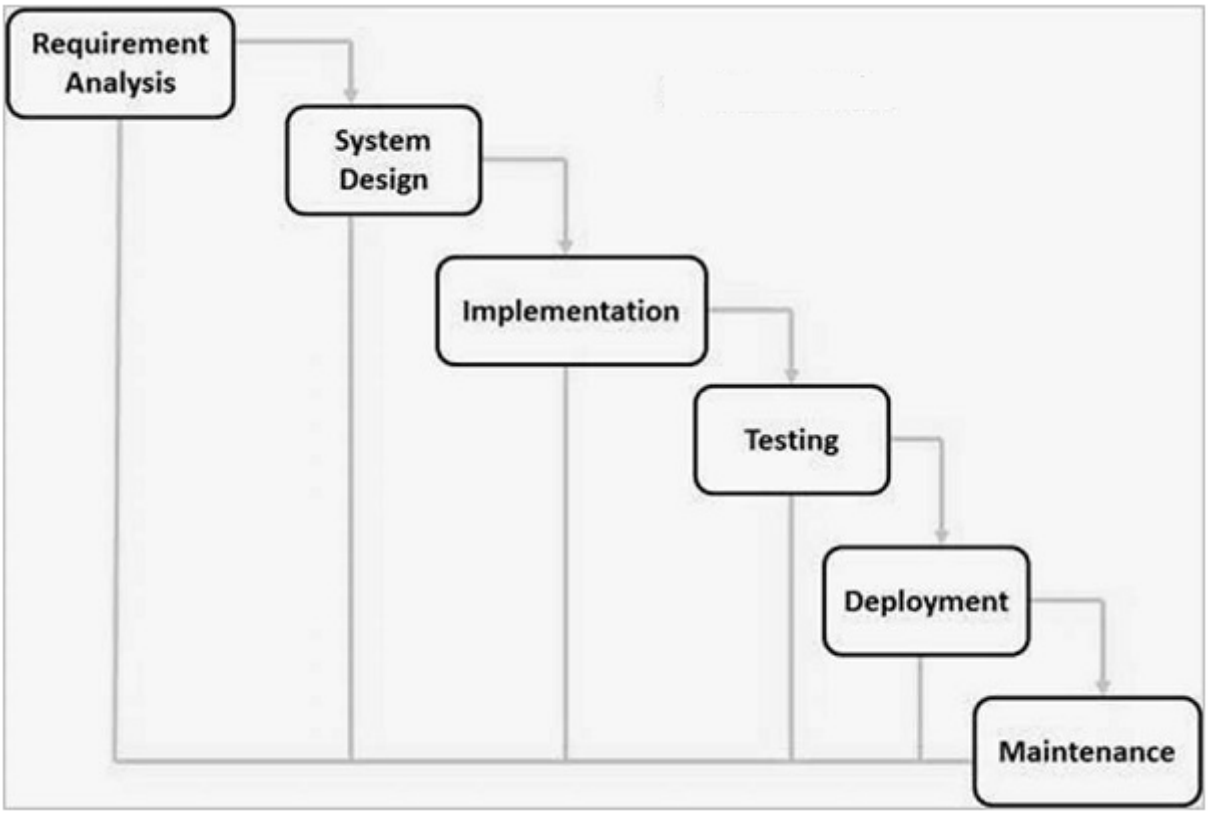


Figure 8 – Waterfall model (Source: Waterfall model. (n.d.))

Incremental Process

First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive versions are implemented and delivered to the customer until the desired system is released.

Spiral

Combines elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. The spiral model was defined by Barry Boehm, based on experience with various refinements of the waterfall model as applied to large software projects (Awad, 2005).

Awad (2005) defines the below life cycles:

- Objective setting – Specific objectives for the project phase are identified.
- Risk assessment and reduction – Key risks are identified, analyzed and information is obtained to reduce these risks.
- Development and Validation – An appropriate model is chosen for the next phase of development.
- Planning – The project is reviewed and plans are drawn up for the next round of spiral.

Relational Unified Process (RUP)

All efforts, including modelling, is organized into workflows in the Unified Process (UP) and is performed in an iterative and incremental manner. Some of the key features of the UP are as follows (Awad, 2005):

• It uses a component based architecture which creates a system that is easily extensible, promotes software reuse and intuitively understandable. The component commonly being used to coordinate object oriented programming projects.

• Uses visually modelling software such as UML – which represent its code as a diagrammatic notation to allow less technically competent individuals who may have a better understanding of the problem to have a greater input.

• Manage requirements using use-cases and scenarios have been found to be very effective at both capturing functional requirements and help in keeping sight of the anticipated behaviours of the system.

• Design is iterative and incremental – this helps reduce project risk profile, allows greater customer feedback and help developers stay focused.

• Verifying software quality is very important in a software project. UP assists in planning quality control and assessment built into the entire process involving all member of the team.

3.5.2.2.Agile Methodology

Agile development is one of many development methodologies. Agile refers to any process that aligns with the concepts of the Agile Manifesto (http://agilemanifesto.org/). This type of methodology is built on principles like simple design, continuous delivery, self-organizing teams, face-to-face communication and fast response. These principles are derived from four core agile values which are:

- Individuals and interactions over processes and tools;
- Working software over extensive documentation;
- Collaboration with customer over contract negotiation;
- Responding to change over following a plan.

According to the set of these values twelve agile principles were proposed. These principles enhance the importance of agility in software development.

Some principles derived from these values are improved by using a cloud environment for development, such as scalability, providing infrastructure (both hardware and software), fast delivery mechanisms, lowering cost and increasing software quality. In the bigger picture cloud computing affects agile software development with increasing prominence (Younas et al, 2018).

Agile software development has various methods and since general talk about it may not give a clear idea of how agile development works, Scrum is stated as the most popular agile framework.

3.5.2.3.Scrum

Scrum is defined as a flexible, holistic product, a development strategy where developers work as a unit to reach a common goal. One development cycle is called Sprint. Sprints are usually no long-term plans that have an elected amount of features that are implemented in one development cycle. After every sprint, Sprint Planning is arranged to prioritize the features. Sprints are created from Sprint Backlogs which works as a to-do list. In Scrum daily meetings are held. Each team member should be prepared and share answers to three basic questions.

- What did the member yesterday do that contributed to sprint goal?
- What does the member plans to do today?
- Are there any difficulties that can prevent the member from contributing?

After each iteration, team members are part of a Retrospective meeting where they share and identify lessons and improvements for the next sprints (Younas et al, 2018). With this simple example of how scrum works, it is safe to say that in the modern world agile development is a great way to work on projects for customers that are driven by fast-changing demand on the market as agile offers solutions for certain problems.

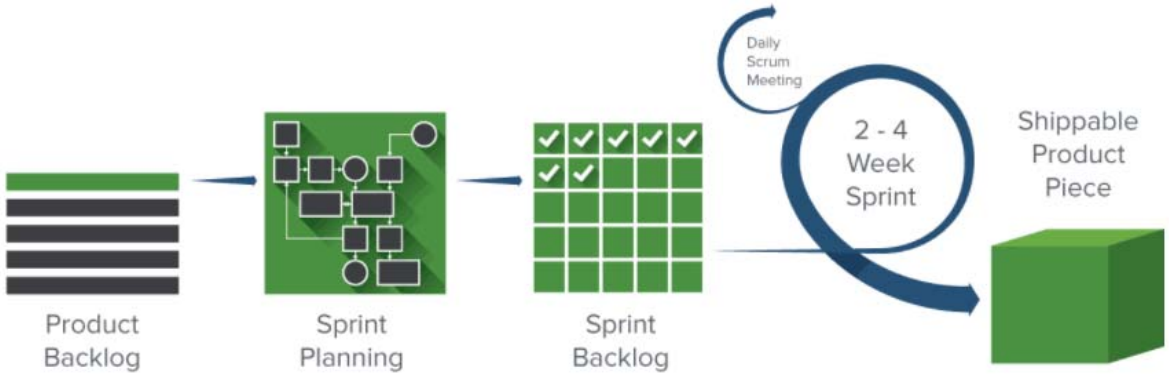


Figure 9 – Steps in the Scrum Process (Source: What's the Difference? Agile vs Scrum vs Waterfall vs Kanban. (n.d.))

3.5.2.4.Kanban

Kanban is a way to improve flow and provoke system improvement through visualization and controlling work in progress. Kanban primarily aims to safeguard the team from the unending tasks appropriated by management, with an attempt to realize ongoing development pace and adaption similar to other Agile methods, where little to no resistance to change arises (Sjøberg, 2012; Bolaji, 2015).

The major principles of Kanban are workflow visualization through the Kanban board, limiting work in progress by minimizing the number of features to be implemented, management and measurement of flow, making clear policies, implementing feedback, looping and enhancing collaboration in a continuous manner (Anderson, 2010).

Kanban practices definition are:

1) *Visualize the Flow of Work*. Use cards or software to visualize the process activities on swim lanes.

2) *Limit Work in Progress (WIP)*. Encourage your team to complete work at hand first before taking up new work. The team pulls in new work only when they have capacity to handle it.

3) *Manage Flow.* Observe the work as it flows through the swim lanes. Address any bottlenecks.

4) *Make Process Policies Explicit*. Visually diagram the process rules and guidelines for managing the flow of work.

5) *Implement Feedback Loops*. Throughout the work process, incorporate regular reviews with the team and customers to gather and incorporate feedback.

6) *Improve Collaboratively, Evolve Experimentally*. As a team, look for and incorporate improvement initiatives, including through safe-to-fail experiments.

The Kanban method makes use of visual Kanban board to improve software development through the display of different phases of the process of development.

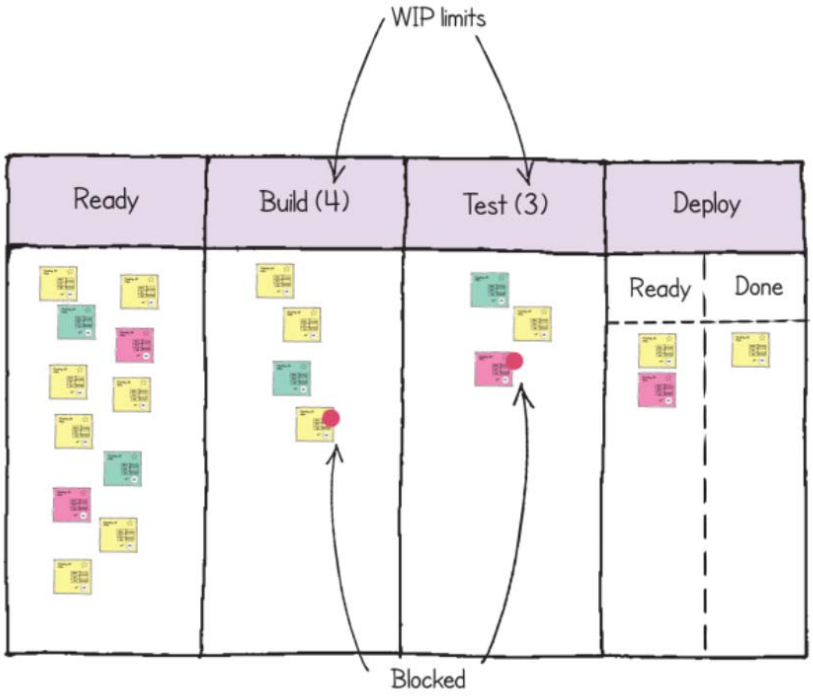


Figure 10 – Kanban board (Source: Mathenge, J. (n.d.).)

3.5.2.5.Adaptative Software Development (ASD)

It aims to enable teams to quickly and effectively adapt to changing requirements or market needs by evolving their products with lightweight planning and continuous learning. The ASD approach encourages teams to develop according to a three-phase process: speculate, collaborate, learn.

3.5.2.6.Lean Software Development

Lean Software Development (LSD) is an agile framework based on optimizing development time and resources, eliminating waste, and ultimately delivering only what the product needs. The Lean approach is also often referred to as the Minimum Viable Product (MVP) strategy, in which a team releases a bare-minimum version of its product to the market, learns from users what they like, don't like and want to be added, and then iterates based on this feedback.

3.5.2.7.Dynamic Systems Development Method (DSDM)

The fundamental idea behind DSDM is to fix time and resources, and then adjust the amount of functionality accordingly rather than fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality.

3.5.3. DevOps

Why is DevOps important? Current IT market is dominated by the speed of releasing products. This can be seen by the popularity of agile techniques to shorten development cycles. And when development cycles are fast enough, there is a bigger need to correctly create space where that product can be placed and regularly updated. With DevOps, it is safer to make changes more often because of automated pipelines of the whole deployment (Artac et al, 2017).

DevOps work usually lays in increasing automation and faster deployment process. First DevOps task is to create an automated deployment mechanism. Deployment strategy is mostly based on deployment scripts or some continuous delivery system, which is triggered by the Continuous Improvement (CI) system. Strategies to deploy to different environments such as development or production may differ. While the development environment is usually automated. Deployment to production often needs manual triggering.

Infrastructure as code, provisioning and configuring environments repeatedly and reliably is part of DevOps expertise and can be part of the CI/CD pipeline. Tools such as Terraform, Chef or Puppet are used for this purpose.

Developers and operators actively monitor applications and services that were developed, both in production or other environments. Monitoring is done for various purposes, such as providing visibility over failures of deployment or quality of provided services. With proper monitoring faster response to bugs and anomalies is achieved which leads to greater customer satisfaction (Lwakatare et al, 2019).

3.5.3.1.Continuous Integration and Delivery (CI/CD)

Continuous integration (CI) and continuous delivery (CD) embody a culture, set of operating principles and collection of practices that enable application development teams to deliver code changes more

frequently and reliably. The implementation is also known as the CI/CD pipeline and is one of the best practices for DevOps teams to implement (Sacolick, 2018).

Continuous integration

Continuous integration is a philosophy that supports rapid software development. Operating principles are based on that philosophy and they help to achieve delivering of new code frequently and reliably. Using this method, it is easier to detect bugs in code sooner than in large additions of code less often.

Teams that want to implement CI/CD to their business often start with version control systems. Code checking can be done frequently for smaller features but also for longer time frames. Development teams are using different strategies for different cases and define how code is merged into production environments.

There are many techniques like version-control branching, which is based on creating a branch for each environment where software is running. One branch is development, for the newest features. The second branch is created for testing, where the testing is done and after all the needed steps are done, code is merged to the production branch which represents the code used in the latest version of the production system.

The second strategy could be feature flags. This mechanism is built around turning on or off features at run time. A production system is using master branch code to run. Newest features are flagged and until they are tested, they cannot be flagged as production-ready so neither be deployed.

Continuous delivery

Continuous delivery is part of CI/CD that delivers software to its desired environments. Usually, teams have more environments such as development, testing and production. Each of those environments should have same configurations but are for different purposes.

The objective of continuous integration is to gather code at one place to be handed to continuous delivery. After everything is set up, a continuous delivery process could look like this:

First, the code is pulled from a version control system and starts a build of an application. Then the infrastructure as code tool is executed to change required infrastructure in a given environment. This step is more important for a cloud environment as they are more mutable. Next step is moving a built application to the target environment and configuring environment variables dependent on the environment that is being used. After everything is set up, the application is pushed to their appropriate services, such as web servers, API services. Then an application is deployed, the last thing to do is execute any steps required to restart services that are needed for new code to take effect. At the moment when is application successfully deployed, continuous tests are executed, if tests fail rollback will be applied.

More and different steps could be part of continuous delivery. Those which are mentioned here should give a good understanding of a given problematic (Sacolick, 2018).

Testing in CI/CD

The vast part of CI/CD is testing. The optimal case is to deliver new versions of software as quickly as possible. Also, quality assurance is very important. This means that the CI/CD pipeline should have included various types of tests to be executed in process of delivering new versions, and in case tests will find an error in code or delivery process, a rescue plan should exist. That rescue plan might be a rollback to the previous version.

However, the best practice in testing is before continuous delivery is executed. Before releasing a new feature, developers should run unit tests, functional tests and regression tests on their local environment. This leads to correct code in version control systems after committing a new portion of code without breaking the working environment.

Testing code is the first part of the testing of the whole software. There are more like performance testing, API testing, security testing, all these can be also automated. The key to automating these tests is the ability to trigger them some easy way such as the command line.

When all testing is automated, it can be integrated into the CI/CD pipeline. Raw code testing can be done in CI while committing or merging with the master branch. Other tests like performance testing could be done only after deploying the new version to the target environment and if those fails, rollback can be executed (Sacolick, 2018).

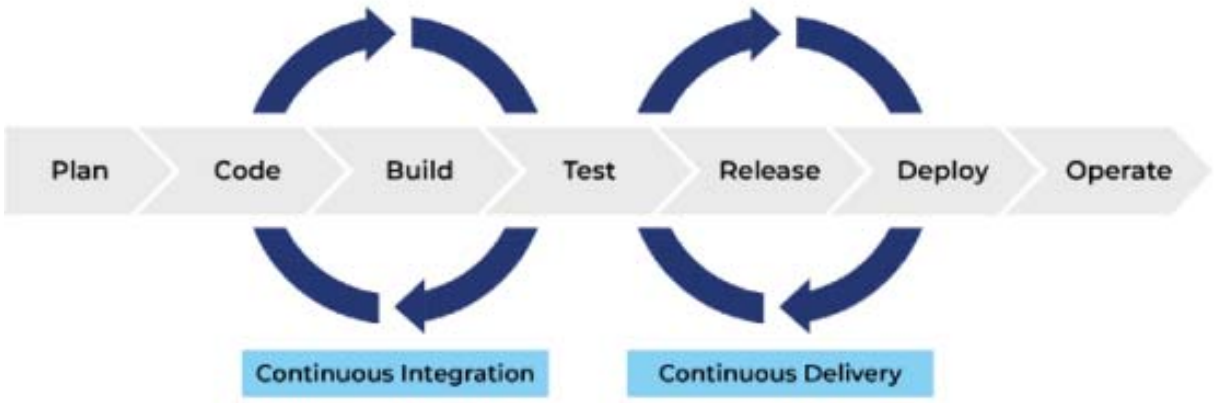


Figure 11 – CI/CD process (Source: Eneh, T.)

3.5.3.2.Infrastructure as Code (IaC)

Automation of infrastructure is a key DevOps practice. The philosophy behind this is that infrastructure gets a new level of abstraction, infrastructure becomes part of code which describes the desired infrastructure configuration in definition files. That means we can treat infrastructure as another part of the software.

Why is infrastructure as code important? The answer is quite simple. It saves time. It reduces the time spent on doing repetitive things such as patching infrastructure. IaC allowed to create definition files of configuration and so it reduces propensity on errors.

When infrastructure is automated, it is easier to test software in a sandbox environment. It is easy to spin up a new environment, test the new version of the application and then tear down infrastructure in minutes.

In the “cloud age” it is even easier to run these configurations because there is no need to buy physicals servers. It is easier to set up infrastructure on cloud servers which are provided by many companies. On the cloud, there is no need to be afraid of the local server having enough memory because it can be easily added by requesting the provider for more (Johann, 2017).

However, once the infrastructure is managed by IaC tool it might be troublesome to manually debug problems. That is because the IaC tool usually holds a current state which is the last snapshot of applied infrastructure. When is that infrastructure manually changed it may corrupt given state and prevent the tool from working correctly (HashiCorp).

3.5.3.3.Existing Infrastructure as Code Tools

Terraform

Terraform is a server provisioning tool. This means it is responsible for server creation in opposite to IaC tools that install and configure an existing server (Danek). It is a universal IaC tool that is cloud-agnostic and helps to manage large infrastructure for different kind of applications (Chan). Its automation is often different. Some teams run Terraform locally but use the consistent working directory for Terraform to run in. Another approach could use different orchestration tools such as Jenkins to run Terraform (Nallamala).

Terraform uses its domain-specific language. Infrastructure is described using high-level configuration syntax.

The core usage of Terraform is built on planning and applying configuration files. Before creating or updating infrastructure Terraform informs the user about changes in a plan that contains exact information about resource changes in existing infrastructure to increase safety and reduce human error. After inspecting changes, the plan can be applied and infrastructure is updated in a moment (HashiCorp).

Chef

Chef is a popular IaC configuration management tool among CI/CD practitioners. Chef handles installation and management of software on existing servers. Chef uses Ruby-based DSL to create its recipes and cookbooks. It has versioning system that allows maintaining a consistent configuration. Each cookbook should relate to a single task, but it can deliver different server configurations based on resource definition. Chef uses a procedural approach to its configuration, as describing procedure is necessary to get the desired state. Thanks to its support for cloud provisioning APIs, Chef works well with other IaC server provisioning tools. Chef is cloud-agnostic and works with many cloud service providers (Danek; Nallamala).

Puppet

Similar to Chef, Puppet is a popular configuration management tool that helps with continuous delivery of software. It has Ruby-based DSL. Puppet which uses a declarative approach to its configuration. Defining the desired state of your infrastructure causes Puppet to automatically enforce the desired state and fixes any incorrect changes. This approach is mainly directed toward system administrators. It can be integrated with the leading cloud providers such as AWS, Azure, Google Cloud and VMware (Chan; Nallamala; Danek).

Ansible

Ansible is an open-source infrastructure configuration tool. Ansible forms infrastructure by describing relations between system components as opposed to others which manage systems independently. It describes its configurations in YAML in form of Ansible Playbooks, because of that configurations are easy to understand and deploy. Its functionality can be extended by writing new modules and plugins (Chan).

Atlantis

Atlantis is an open-source tool that allows improved collaboration on projects where Terraform is used. Its review and applying system is done in pull requests created in version control system. It allows automation of infrastructure creation (Atlantis).

Atlantis is a good tool to add to the CI/CD pipeline while using Terraform. However, it does not support control of concurrent access to infrastructure changes.

4. PROJECT

4.1. ASSUMPTIONS

The proposed architecture must meet the below business requirements:

- Software must handle 10.000 shipconfirms messages and perspective to scale to 30.000 messages/daily for Black Friday and Christmas (peak periods)
- Handle 5000 Tracking ID's daily and should be able to scale to 30.000/day
- Multithread and ability to handle more than 100.000 messages/day
- 20.000 Shipconfirm/Tracking ID day for all carrier
- Integration with Apache Kafka, Metis backoffice, log patterns and CTT
- Autoscale based on demand
- Event-Driven
- Error recover capabilities
- Resilience to failures or infrastructure
- High availability
- Fast response
- Microservices design

4.2. DEVELOPMENT METHODOLOGY

The development methodology defined to be used was decided based on the traditional and agile methodologies analysed. According to the project characteristics it was decided to use the Scrum framework because we focus to adopt agile methodology aligned with "XPTO" view in order to add value on the shortest time period possible.

According with Scrum we adopt the current Sprint Calendar template:

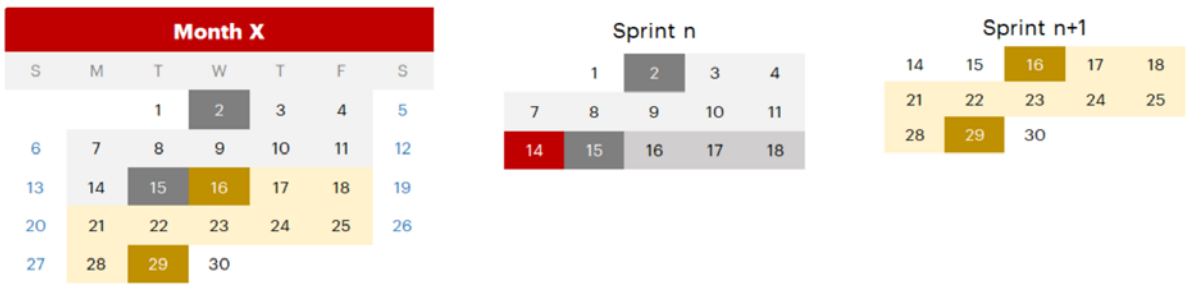


Figure 12 – Sprint Calendar template (Source: own)

1. The sprint starts on Wednesday as a duration of two weeks and ends on a Tuesday.
2. The last deployment to Dev environment should happen until 12am, on the last Monday of the sprint (highlighted in red above). Issues reported during the QA phase might warrant deployments after this deadline.
3. Local development is still available to developers.
4. Last Tuesday, all Tasks with a status of completed are presented in DEV environment during the Sprint Review.
5. Three days after the end of the sprint the "XPTO" team can test in Dev and approve the deployment to the Production environment in the example above 16-18th days.

The Scrum characteristics allows to:

- Measure progress through value added delivery
- Adaptive solutions to complex problems
- Continuous delivery of value in short periods
- Pillars: transparency, inspection and adaption

Team was composed by:

- Project Manager
- Product Owner – Authority to define final product, incrementally. Grants and maximizes value added from development team deliveries
- Business Owner
- Solution Designer
- Scrum Master – handle blockers and facilitate iteration between team members
- 7 developers – able to build the software end-to-end and with quality

With this framework we are able to build a MVP and execute all tests required in order to grant the quality and deliver new code releases at the end of each sprint.

4.3. FUNCTIONAL REQUIREMENTS ANALYSIS

Functional requirements analysis is fundamental for software development because they are considered as the needs that we need to address and grant. Well defined requirements will allow a good software development in order to achieve project deadline.

4.3.1. Functional Requirements

Functional requirements focus on existing problems and new operational needs according with operational and business key users.

The functional requirements were classified with priority and all of them were taking into account for the MVP delivery on Black Friday period.

ID	Title	Description	Priority
FR1.0	Save counter(s) values for tracking number generation	Create, for each carrier, counters that will be used to generate the values of the tracking number, according to the shipping type (Home Delivery or Pick Up In Store).	**HIGH**
FR2.0	Tracking number logic	The tracking number generation logic, must be guaranteed in order to produce different values, regarding the pair "carrier"/"shipping type" and in accordance to the carrier instructions.	**HIGH**
FR3.0	Generate WM shipping labels according to carrier layout	For each pair "carrier"/"shipping type", there must be printed a shipping label, according to the carrier shared layout and with all the relevant information required by the carrier, to guarantee a correct delivery.	**HIGH**
FR4.0*	Execute Packing to invoice and print/stick WMS label	Pack the items and make: • OMS call; • AT call; • print label (shipping or exception depending on the OMS feedback) The layout of this label depends on the shipVia for the pack that has been assigned on the Waving. If the shipVia corresponds to the one created to the alternative carrier, than the printed label should be according to the defined carrier layout (FR3.0).	**MEDIUM**
FR5.0*	Sort the packages	At the end of the "packing convey belt" all the packages should be scanned and sorted to be directed to a specific zone ("sorter logic" to be updated)	**HIGH**
FR6.0*	Scan packages at PTL ONLINE	The PTL ONLINE system splits the packages to carton boxes, according to the store ID or group of stores. To achieve this PTL ONLINE has configured the different store ID's linked to the display addresses.	**LOW**
FR7.0	Print "aggregated box" label on PTL ONLINE	If the "delivery type" is PUIS, the PTL ONLINE system must be prepared to communicate with WM to print a "aggregation box" label that has a tracking number according to the counter and logic previously referred on FR1.0 and FR2.0 respectively as well as the layout referred on FR3.0 .	**HIGH**

FR8.0	Reprint Shipping Label	If there's the need to reprint a Shipping Label there are currently two options: 1. via RF Menu - option "Print OB Docs"; 2. via UI - on the "oLPN Detail Menu" → "More" → "Print Shipping Label". For both of this options the tracking number logic is triggered and a new tracking number is generated for this new shipping label (works as designed). This logic must be maintained and the new tracking numbers must respect the the logic previously referred on FR1.0 and FR2.0, as well as the layout referred on FR3.0 .	**HIGH**
FR9.0*	Reprint"aggregation box" label	If there's the need to reprint a "aggregation box" label the current procedure is to (using the box ID): • via UI - on the "oLPN Detail Menu" → "More" → "Print Shipping Label".	**LOW**
FR10.0*	Palletize or Anchor package	After the sorting there's the need to group the packages on a pallet so that they can be easily loaded on the truck. To fulfill this actions the oLPN of the package needs to be scanned.	**LOW**
FR11.0*	Load pallet	Load the pallets with all the packages on the truck, by scanning the pallet ID.	**LOW**
FR12.0	Send shipconfirm message to external systems	A shipconfirm message is created to be sent to the external systems (RETEK, OMS, WeCare & *"carrier"*) in order to align stock, close related documents, allow tracking and inform the carrier that a new package is to be delivered by them. The "carrier" message component must be generated and sent to the correspondent carrier and not to any other carrier. The current logic for *dpd* message component is triggered if the shipVia has the "CH##" mask.	**HIGH**
FR13.0*	Guarantee that FOR EACH ORDER LINE the tracking number is on external systems (i.e OMS and/or WeCare)	Generate on WM and propagated trough the AS IS flow, to the external systems (OMS and WeCare), the corresponding order line tracking number to allow track and trace of the packages and therefore provide to final costumer information on the the different package status.	**HIGH**

FR14.0*	Provide ALL RELEVANT and NECESSARY information to BI Teams	Using standard systems and procedures, allow access, by BI Teams, to ALL RELEVANT and NECESSARY information, so that they can accomplish their needs.	**HIGH**

Table 1 – Functional Requirements (Source: own)

4.3.2. Diagrams

4.3.2.1. Use Case Diagram – CTT process AS-IS

The use case diagram as the function of describing the CTT shipping orders flow and allows to have the general vision of the flow itself as it is implemented right now.

Currently, the Alternative Carrier for CTT has many additional manual steps (orange boxes on figure 14):

- Export file after waving with a list of oLPNs;
- Import previous file to CTT local system;
- Scan oLPN after WM packing, to print CTT label;
- Place CTT label on the package;
- Export file from CTT App with oLPNs and Tracking Numbers;
- Import previous file into CTT Wrapper system.

The operation because of the above manual extra steps, in order to grant the sequence flow has to perform 8 additional activities (see figure 13) that impact operational efficiency and can lead to human error mistakes.

By the below diagram we can see that we have several manual activities that are required to be performed in order to ship goods and all of them need to be achieved otherwise the agreed Lead Time with the client will not be achieve.

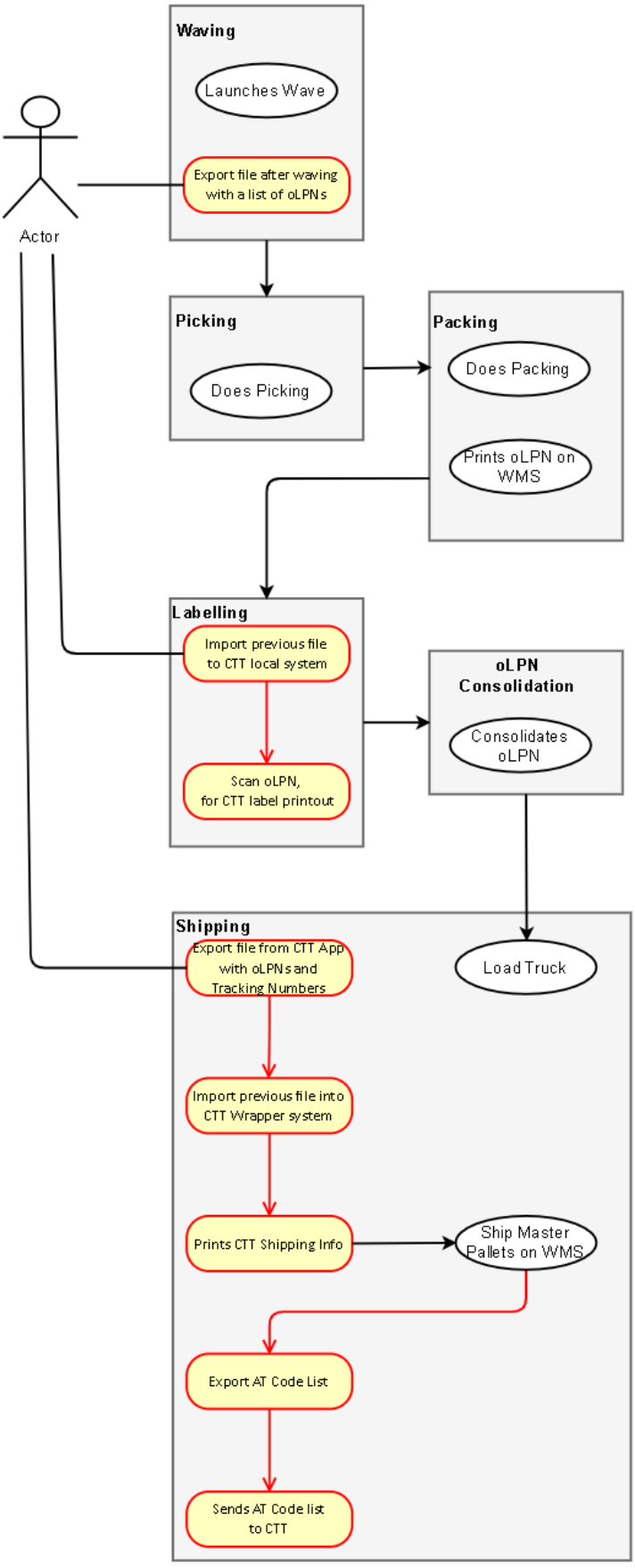


Figure 13 – CTT manual steps (Source: own)

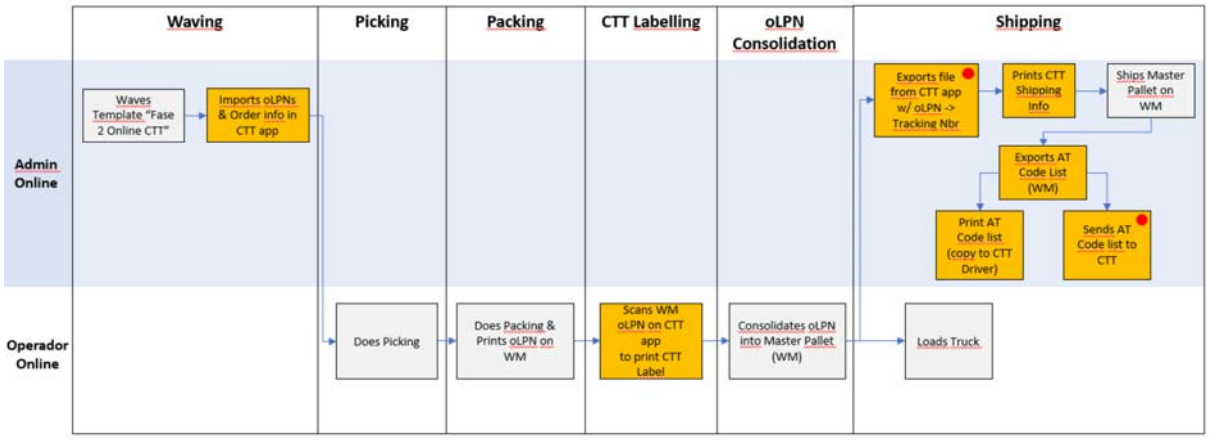


Figure 14 – CTT shipping orders flow (AS-IS) (Source: own)

4.3.2.2.Use Case Diagram – CTT process TO-BE

The below use case diagram describes the new CTT shipping orders flow according with the functional requirements used for the software development.

The main objective of this initiative is to find solutions in order to minimize or even remove all the manual processes that are currently being used on the ONLINE flow regarding the shipping trough the Alternative Carrier CTT.

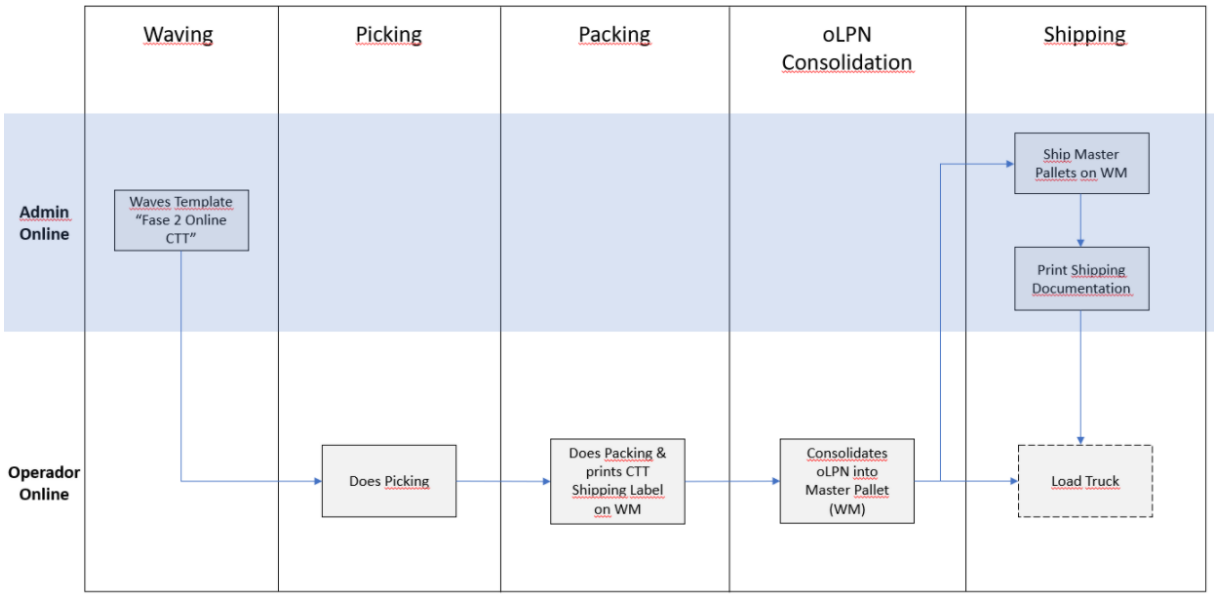


Figure 15 – CTT shipping orders flow (Source: own)

CTT Packing process will be achieved through tracking id & label flow with message integration as follows:

1. When "XPTO" operations pack the Order, that action triggers the Tracking ID & Label request to WhTF API via HTTPS.
2. WhTF validates the message and routes it to microTMS.
3. MicroTMS will generate tracking number and label and sends Tracking ID & Label message to WhTF.
4. WhTF will deliver Tracking ID & Label message to WMS API and store it into DynamoDB.

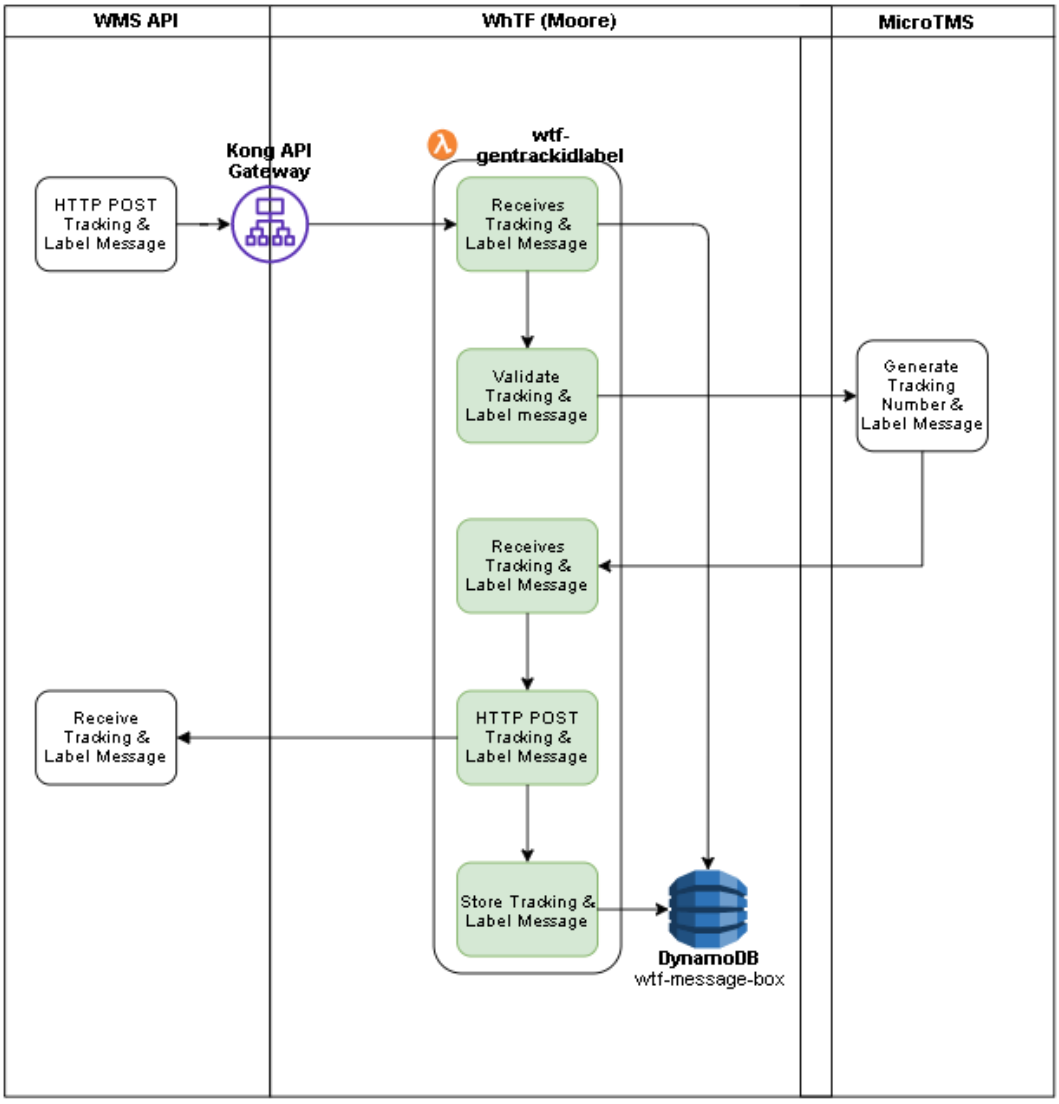


Figure 16 – CTT TrackingId & Label flow end-to-end (Source: own)

CTT Shipping process (Load Truck) will be achieved through shipconfirm flow with message integration is as follows:

1. WMS sends shipment confirmations through HTTP to WhTF API.
2. WhTF proceeds to save every ShipConfirm message into kafka private topic whtf-ship-confirms.

3. Amazon MSK event source mapping reads from kafka private topic whtf-ship-confirms and validate ShipConfirm message.

4. Transform ShipConfirm message into STDShipConfirm message.

5. Transform STDShipConfirm message according with WOSA envelope and publish it into topic whtf-std-ship-confirms.

6. MicroTMS will consume topic whtf-std-ship-confirms and transform it according to CTT needs.

7. MicroTMS will sent CTT ShipConfirm message to CTT carrier through SFTP.

8. MicroTMS will sent CTT Shipconfirm message to WOSA topic tms-carrier-updates for Order Management System (OMS).

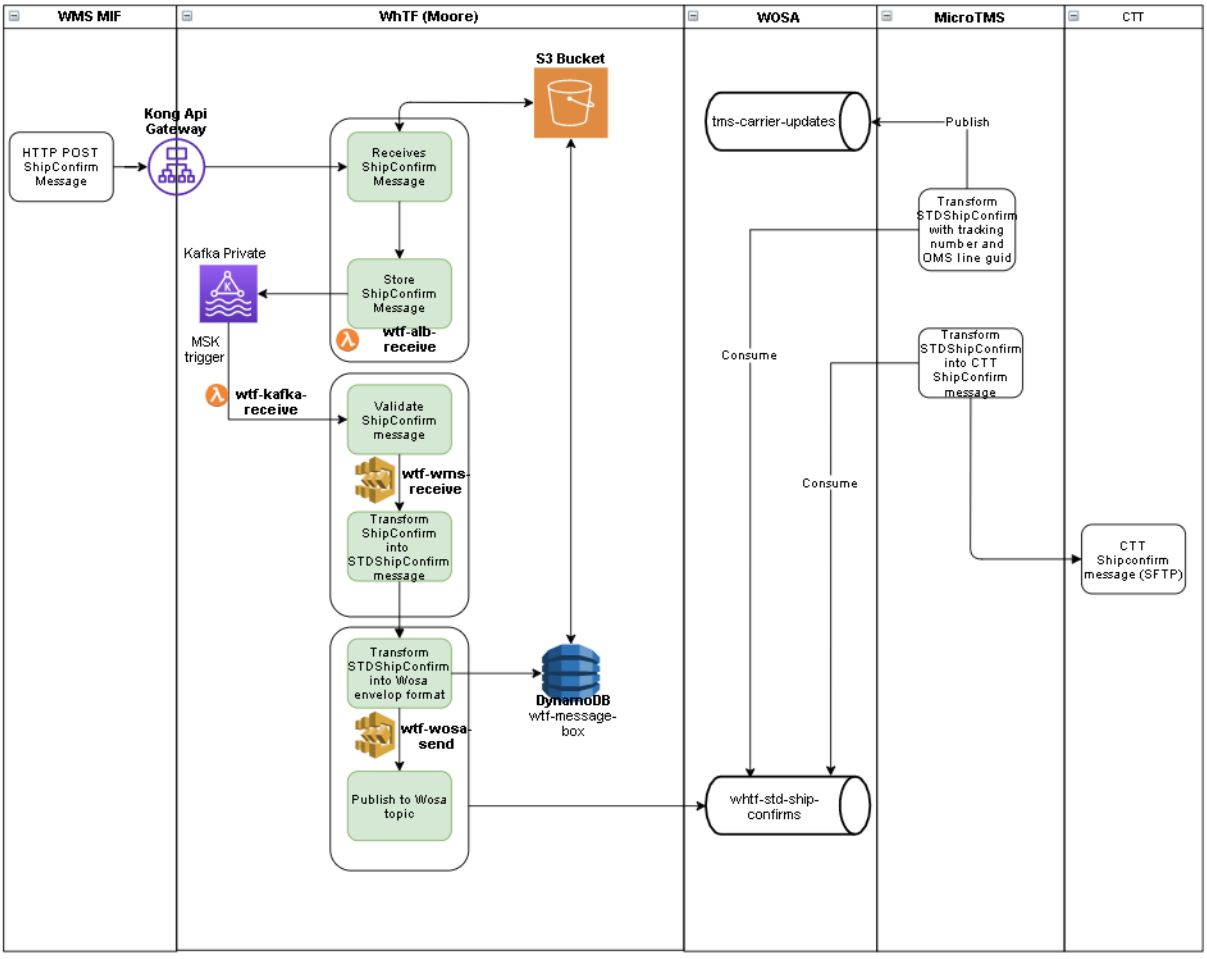


Figure 17 – CTT shipconfirm flow end-to-end (Source: own)

4.3.2.3.Shipconfirm activity diagram

On the shipconfirm activity diagram it's possible to visualize and understand CTT shipping orders flow inside the new development and how all processes relate between them.

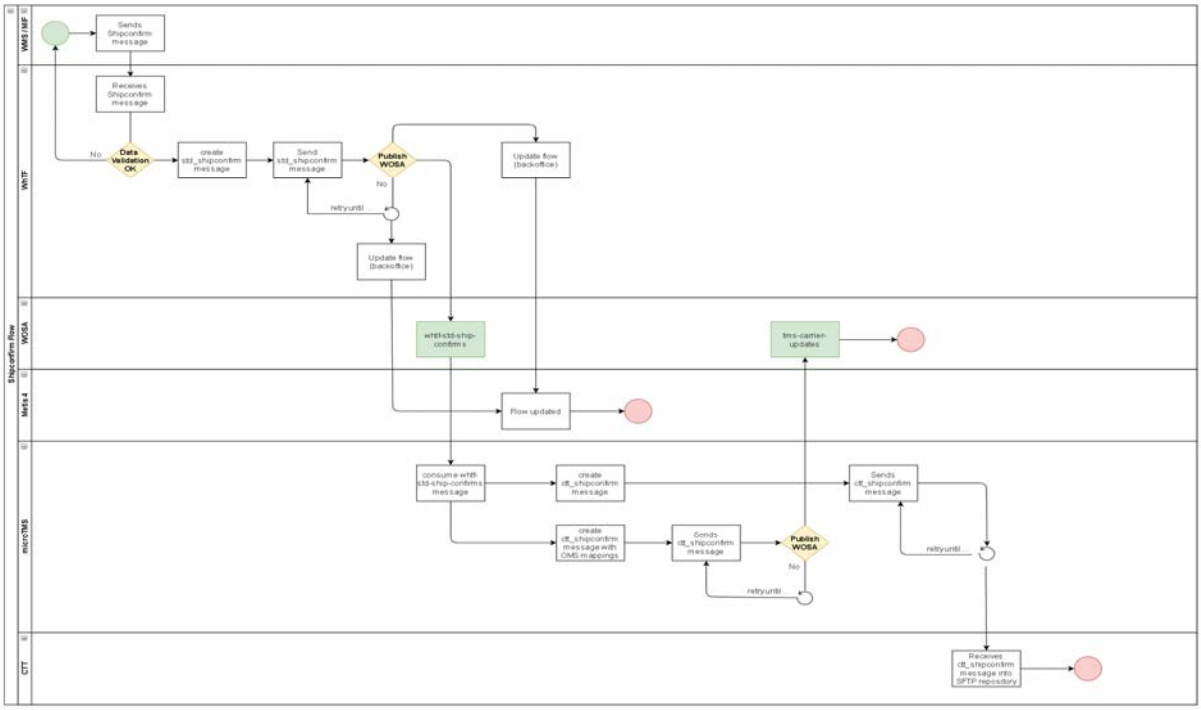


Figure 18 – Shipconfirm activity diagram (Source: own)

4.3.2.4.Shipconfirm sequence flow diagram

It shows how the objects interact with others in this particular scenario of a use case.

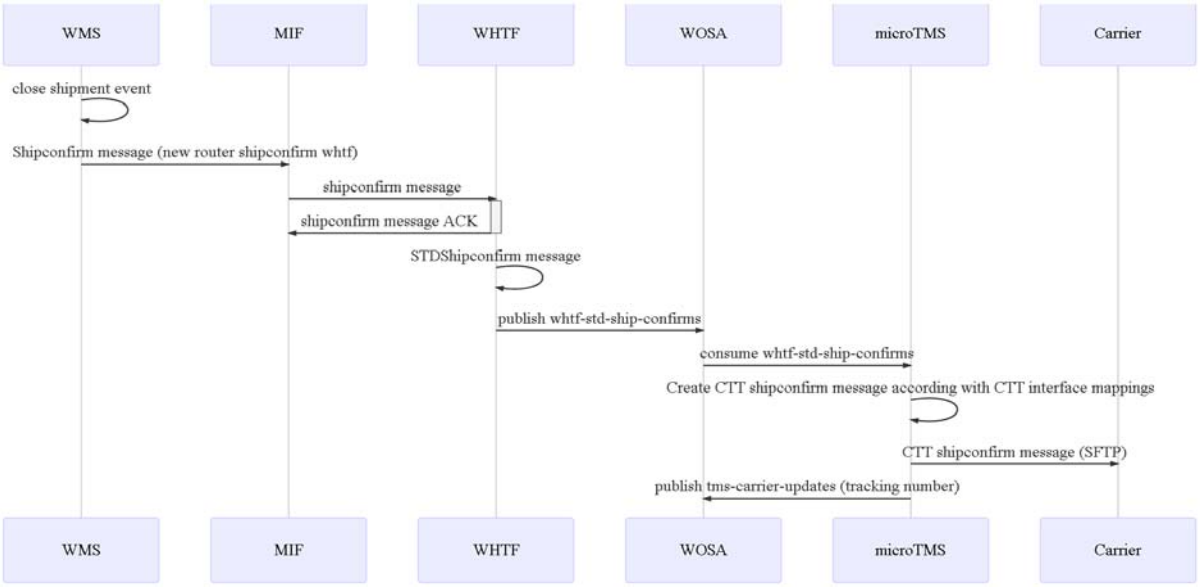


Figure 19 – Shipconfirm sequence diagram (Source: own)

4.3.2.5.Warehouse Transformation Framework - Shipconfirm flow State Machine diagram

State diagrams depict the permitted states and transitions as well as the events that effect these transitions. In case of fallbacks, retry mechanism is configured in order to help to overcome some issues that may arise during the message flow.

Definition

```
{
  "Comment": "Wtf State Machine",
  "StartAt": "Receive",
  "States": {
    "Receive": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:[redacted]:function:moore-wtf-wms-receive-dev",
      "Next": "Send",
      "Retry": [ {
        "ErrorEquals": [ "RetriableException" ],
        "IntervalSeconds": 1,
        "BackoffRate": 2.0,
        "MaxAttempts": 5
      } ],
      "Catch": [ {
          "ErrorEquals": [ "ShipConfirmValidationException" ],
          "Next": "Fallback"
      } ]
    },
    "Send": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:[redacted]:function:moore-wtf-wosa-send-dev",
      "Retry": [ {
        "ErrorEquals": [ "RetriableException" ],
        "IntervalSeconds": 1,
        "BackoffRate": 2.0,
        "MaxAttempts": 5
      } ],
      "End": true
    },
    "Fallback": {
       "Type": "Pass",
       "End": true
    }
  }
}
```

Figure 20 – Shipconfirm state machine diagram (WhTF) (Source: own)

4.3.2.6.microTMS - Shipconfirm flow State Machine diagram

The same logic explained before is applied here.

Definition

```
{
  "Comment": "Tms State Machine",
  "StartAt": "Receive",
  "States": {
    "Receive": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:[redacted]:function:moore-tms-wosa-receive-dev",
      "Next": "Transform",
      "Retry": [ {
        "ErrorEquals": [ "RetriableException" ],
        "IntervalSeconds": 1,
        "BackoffRate": 2.0,
        "MaxAttempts": 5
      } ],
      "Catch": [ {
          "ErrorEquals": [ "ShipConfirmValidationException" ],
          "Next": "Fallback"
      } ]
    },
    "Transform": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:[redacted]:function:moore-tms-wosa-transform-dev",
      "Retry": [ {
        "ErrorEquals": [ "RetriableException" ],
        "IntervalSeconds": 1,
        "BackoffRate": 2.0,
        "MaxAttempts": 5
      } ],
      "End": true
    },
    "Fallback": {
       "Type": "Pass",
       "End": true
    }
  }
}
```

Figure 21 – Shipconfirm state machine diagram (microTMS) (Source: own)

4.3.2.7.Tracking Id & Label activity diagram

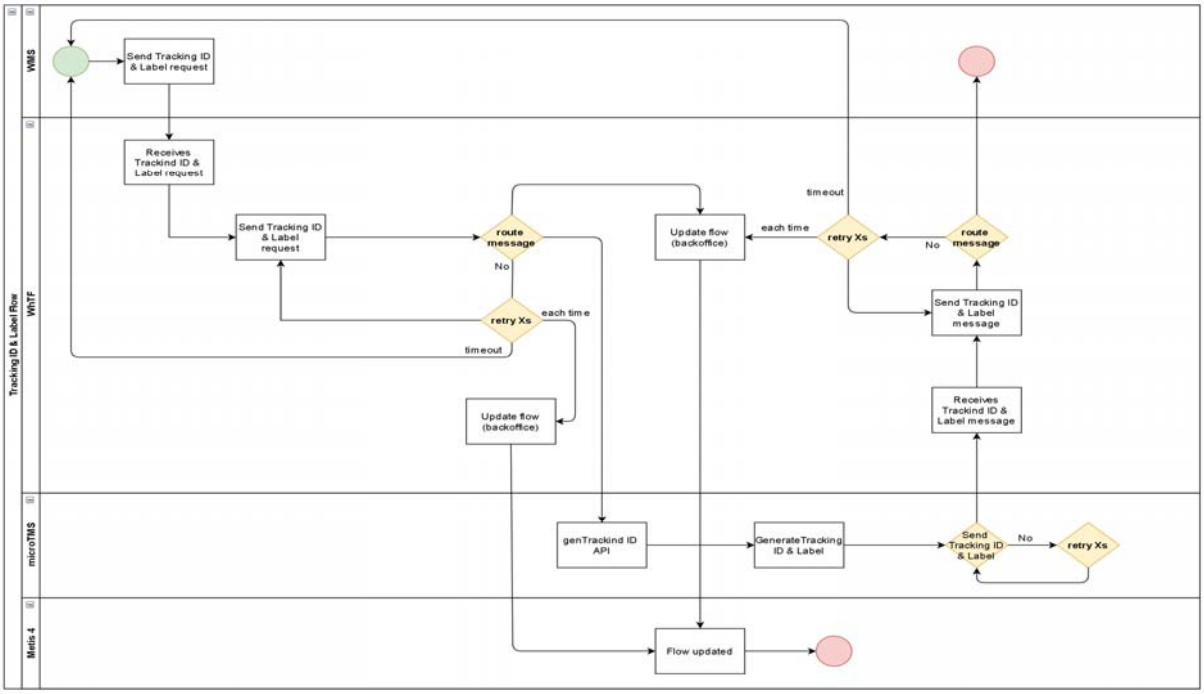


Figure 22 – Tracking Id & Label activity diagram (Source: own)

4.3.2.8.Tracking Id & Label sequence flow diagram

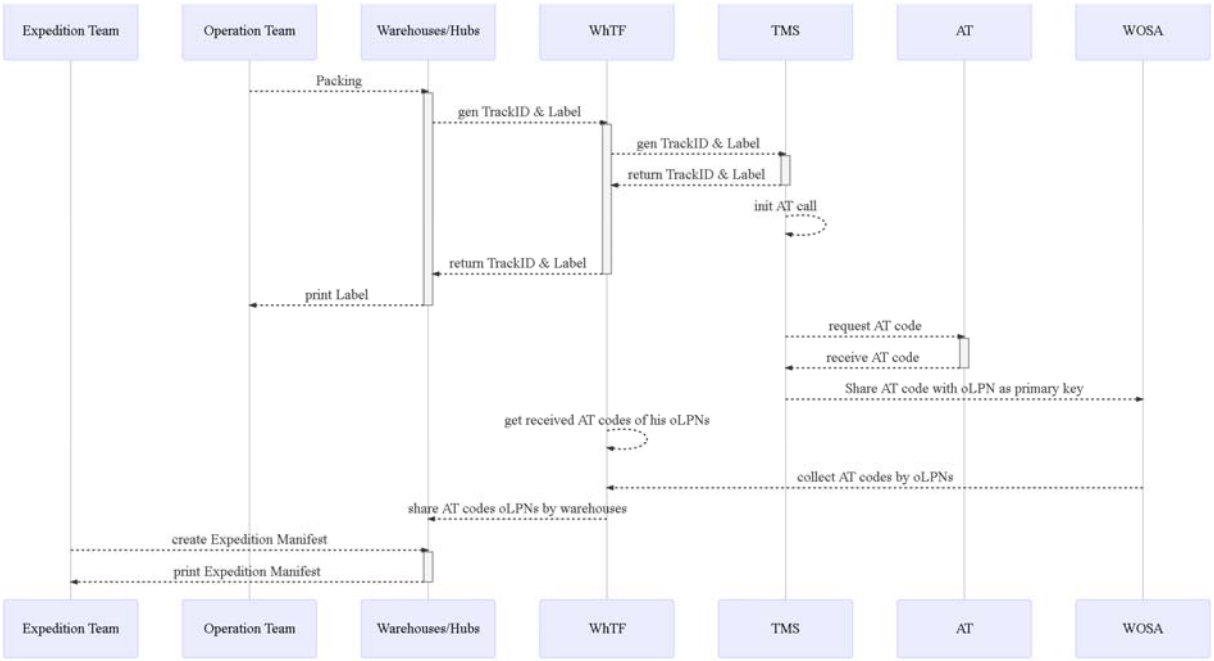


Figure 23 – Tracking Id & Label sequence diagram (Source: own)

4.3.2.9.Shipconfirm

WMS sends shipconfirm messages after the preparation of all orders assigned to a shipment. Those messages will be received in xml format and converted into shipconfirm standard message (json format) for microTMS consume.

WhTF transforms every carrier shipconfirm message on the standard format, according with the mappings on Annex 1, but microTMS will only consume CTT messages.

The message wil be published on the kafka topic whtf-std-ship-confirms with the following envelope structure:

Message

Kafka Header	Attribute	Type	Mandatory	Unique	Description
id	id	"XPTO "ID	YES	YES	A unique ID for this message, to act as an idempotence identifier. a Base64-encoded string of an array of 16 bytes.

Stream

Kafka Header	Attribute	Type	Mandatory	Unique	Description
sn	name	String	YES	YES	Stream the message is related to.
sk	key	String	YES	YES	Unique identifier of the entity inside the Stream. This usually maps to the primary key of the entity in the source database system.
si	intent	String	YES	NO	The intent of this message. This can describe an action or an event that occurred.

sv	offset	String	YES	NO	Offset of this message inside the scope of the <Name,Key> tuple. Example: "4th message for the Order with OrderID X". You should consider this field as the "version" of the entity value. When you create an entity, the version would be 1. When you update it afterwards, the version would be 2. The ideal representation is an integer that starts at 1 and is incremented for each update you make. This allows consumers to quickly discard old messages (just track the version of the entity and discard messages with versions below the last you recorded), but also to detect if you lost ant messages (if you recorded version 5 and you get a message for version 7, you can determine that you lost 1 message - if that is important or not, it depends on the use case). The fallback representation (because it is easier and usually available) is to use the millisecond precision last-updated-at timestamp for the entity. Please note that if you use this representation, you will lose the ability to detect missing messages. Again, the ideal representation of the previous paragraph is strongly recommended.

Tracing

Kafka Header	Attribute	Type	Mandatory	Unique	Description
tg	request_id	"XPTO "ID	YES	NO	Context/Span/Trace ID of the **global** event or command that triggered this message. Establishes a "parenting" relationship with the event or command with that "XPTO"ID.

tc	tracing_id	"XPTO "ID	YES	NO	Context/Span/Trace ID of the **caller** event or command that triggered the chain of events that lead to this message. Whenever the event or command that caused this message to be sent has a Context/Span/Trace ID, that Context/Span/Trace ID should be copied into this field. This allows for the correlation of all the activity related to a single "root cause" event or command. This field is mandatory.

Payload

Kafka Header	Attribute	Type	Mandatory	Unique	Description
ps	schema	String	YES	NO	The schema of the Raw message field, defining the serialization method and the version. Examples: "json#1", "json#1.1".
pa.<key>	annotations	Set of pairs Key.	NO	NO	A set of pairs Key, Value that annotates the message, allowing for possible filtering of messages. The Key may repeat itself allowing for annotating the message with multiple values for the same key (e.g. the several SKUs present in an order).
None. This is the message content itself	raw	Array of bytes	YES	NO	The serialized data of the actual message.

Backwards Compability

Kafka Header	Attribute	Type	Mandatory	Unique	Description
ev	envelope_version	String	NO	NO	"0" for the inline envelope (deprecated version) "1" for the current version of the Kafka Headers envelope If ev doesn't exist it's assumed the envelope is inline (old deprecated version)

Shipconfirm kafka envelope message example

KAFKA Headers

"id" = "AXERr2d/mMlKatwA+ELS1Q=="
"sn" = "whtf-std-ship-confirm"
"sk" = "000085389"
"si" = "ShipConfirm"
"sv" = "1595611980"
"tg" = "AXERr2d/vqwSQJtIOYFniA=="
"tc" = "AXERr2d/vqwSQJtIOYFniA=="
"ps" = "json#1"
"pa.shipvia" = "CTHD"
"pa.shippeddttm" = "7/24/20 17:33"
"ev" = "1"

KAFKA Message

complete payload message


```
{
  "Header": {
    "Source": "MANH_wms",
    "Message_Type": "ShipConfirm"
  },
  "Message": {
    "ShipConfirm": {
      "ShipConfirmSummary": {
        "ShipConfirmHeaderInfo": {
          "ShippedDttm": "7/24/20 17:33",
          "CreatedDate": "7/24/20 17:33",
          "ProNbr": "01989898"
        }
      },
      "ShipConfirmDetails": {
        "Orders": {
          "BillAcctNbr": "GAPR 2020.70819/303092",
          "BillFacilityAliasId": "1460",
          "DoType": "Customer Order",
          "MajorOrderGrpAttr": "1100500865",
          "OrderType": "ONL",
          "DoStatus": "Shipped",
          "StoreNbr": "9610004275",
          "DistroNumber": "80f3588c-3057-4529-856e-c36c7eef4d39",
          "OrderBillToInfo": {
            "BillToEmail": "nd",
            "BillToPhoneNumber": "915322079"
          },
          "OrderDestInfo": {
            "DestAddress1": "R Particular de Sto. Antonio",
            "DestAddress2": "AJUDA",
            "DestCity": "LISBOA",
            "DestCountryCode": "PT",
            "DestFacilityAliasId": "513",
            "DestName": "ANTONIO GEMELGO",
            "DestPostalCode": "1300000"
          },
          "OrderLineItem": {
            "ItemName": "7054195",
            "OrderQty": "1",
            "ShippedQty": "1"
```

```
        },
        "Lpn": {
          "ShipVia": "CTHD",
          "TcLpnId": "256019880033984442",
          "TcOrderId": "99733906",
          "TcShipmentId": "000085389",
          "TrackingNbr": "1100500865_001",
          "ReturnTrackingNbr": [],
          "Weight": "0.2",
          "TaxID": "9612270797",
          "LpnDetail": {
            "ItemName": "7054195",
            "TcOrderLineId": "1",
            "LpnItemAttributes": {
              "ProductStatus": "02"
            },
            "LpnSerialNumber": {
              "Quantity": "1",
              "SeqNbr": "1",
              "SrlNbr": "uyhchchchgccgh"
            },
          "OrderQty": "1",
          "ShippedQty": "1"
          }
        }
      }
    }
  }
}
```

In case the message is for CTT carrier then data validation is done on lambda wtf-wms-receive for mandatory fields, namely:

WMS Shipconfirm Message	Description	DataType Values	Mandator	DataType
Header.Source	Source system	"MANH_wms" - (default value)	Y	String
Header.Message_Type	Message Type	"ShipConfirm" - (default value)	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.ShippedDttm	Shipped Date Time	example: 7/24/20 17:33	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.DateCreated		example: 7/24/20 17:33	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.ProNbr			Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.BillAcctNbr	Bill account number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.BillFacilityAliasId	Bill Facility Alias		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DoType	Distribution Order Type	"Customer Order" - default value	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.MajorOrderGrpAttr	Major Order grouping Attribute		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderType	Order Types	"ONL" - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DoStatus	Distribution Order Status	"Shipped" - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.StoreNbr	AT Code for ONL Orders		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DistroNumber	OMS Channel Order Guide		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestAddress1	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCity	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCountryCode	Destination Address Information	PT - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestFacilityAliasId	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestName	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestPostalCode	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.ItemName	SKU ID / Item Name		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.OrderQty	Order Quantity		Y	Decimal
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.ShippedQty	Shipped Quantity		Y	Decimal
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.ShipVia	Carrier Identification for goods transportation		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcLpnId	Tc Lpn Id		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcOrderId	Order ID		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId	Shipoment Number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TrackingNbr	Carrier Tracking Number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.Weight	Total Package Weight (kg)		Y	Decimal
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.ItemName	SKU ID / Item name		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.TcOrderLineId			Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.LpnItemAttributes.ProductStatus	Warehouse ID		Y	String

Figure 24 – CTT Shipconfirm mandatory data (Source: own)

4.3.2.10.CTT Shipconfirm file

For every "XPTO" CTT shipment it is required to generate a shipconfirm file and deliver it by SFTP to CTT for integration (seemappings on Annex 4).

The name of the file should be something like “xol085389.exp”.

- “xol” → This is an ID defined by CTT to us.
- “085389” → This value comes from the Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId on the WOSA Topic (whtf-std-ship-confirms). The same one used on the XML Tags “INTREF“ & “MESREF” & “NRG”
- .exp → This is the extension used by CTT. EXP stands for Symbols Export File.

Below are the details how to pass from Standard Shipconfirm message to CTT specific one for integration.

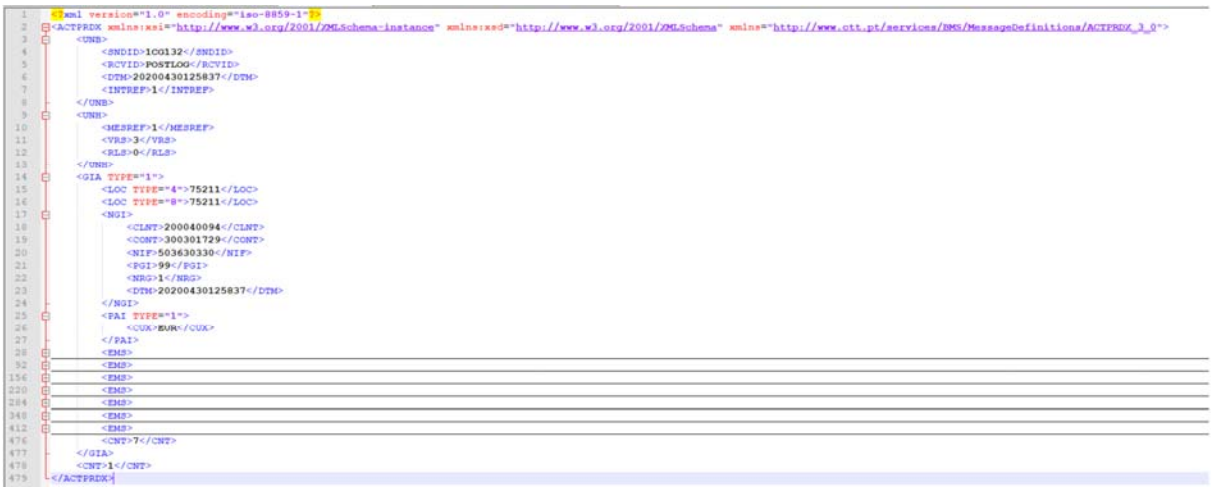


Figure 25 – CTT file Header (Source: own)

Figure 26 – CTT file details (Source: own)

4.3.2.11.CTT Labels

CTT labels requests by "XPTO" operations will originate 2 types of labels: one good and one bad also known as “Exception” label which indicates to the operation that additional actions are required because something is not accurate.

This example shows a regular request (for a Tracking Id & Label) from WMS. The request will always be on JSON format.


```
{
	"labelRequest": {
		"distributionOrderId": "7445324",
		"tcLpnId": "256019880034758868",
		"shipmentNumberOMS": "33283676_001",
		"shipVia": "CTHD",
		"orderType": "ONL",
		"containerType": "CTO",
		"containerSize": "E02",
		"invoiceNumber": "FR 19L1460/349386",
		"originFacilityId": "01",
		"originFacilityName": "XPTO",
		"destinationName": "Antonio Gemelgo",
		"destinationAddress1": "Rua Tomas Pilar 1",
		"destinationAddress2": null,
		"destinationAddress3": null,
		"destinationCity": "Lisboa",
		"destinationPostalCode": "1300258",
		"destinationCountry": "PT",
		"billAddress1": "Rua Tomas Pilar 1",
		"billAddress2": null,
		"billAddress3": null,
		"billCity": "Lisboa",
		"billPostalCode": "1300258",
		"billCountry": "PT",
		"billToName": "Antonio Gemelgo",
		"billPhoneNumber": "911234567",
		"billEmail": "dummy@live.com.pt",
		"destinationFacilityId": "1460",
		"destinationFacilityName": "XPTO ONLINE PT",
		"packedDateTime": "2020-10-02 22:31:44:000000",
		"trackingNumber": null,
		"EAN128": "24250650969881192123785662O",
		"weight": "4,1200",
		"serviceText1": "D",
		"serviceText2": "AXXX",
		"routingText": "0959-LX1",
		"customOrderPrty": "1",
		"refNum4": "5,00000",
		"items": [
			{
				"distributionOrderLine": "52434737",
				"sku": "7194624",
				"ean": "5601988457319",
				"quantity": 1,
				"oms_guid": "066e5c56-dc1a-11ea-bafd-42f3dbb8130e"
			},
			{
				"distributionOrderLine": "52437440",
				"sku": "7220905",
				"ean": "4020628715717",
```

```
                                    "quantity": 1,
                                    "oms_guid": "1fb9b4f2-e08a-11ea-9c1c-57ab20de6f81"
                                }
                            ]
                        }
}
```


Field name	Parent node	Mandatory	Type and Size
distributionOrderId	labelRequest	Yes	String
tcLpnId	labelRequest	Yes	String
shipmentNumberOMS	labelRequest	Yes	String
shipVia	labelRequest	Yes	String
orderType	labelRequest	Yes	String
containerType	labelRequest	No	String
containerSize	labelRequest	No	String
invoiceNumber	labelRequest	No	String
originFacilityId	labelRequest	Yes	String
originFacilityName	labelRequest	No	String
destinationName	labelRequest	No	String
destinationAddress1	labelRequest	Yes	String
destinationAddress2	labelRequest	No	String
destinationAddress3	labelRequest	No	String
destinationCity	labelRequest	Yes	String
destinationPostalCode	labelRequest	Yes	String
destinationCountry	labelRequest	Yes	String
billAddress1	labelRequest	Yes	String
billAddress2	labelRequest	No	String
billAddress3	labelRequest	No	String
billCity	labelRequest	Yes	String
billPostalCode	labelRequest	Yes	String
billCountry	labelRequest	Yes	String
billToName	labelRequest	Yes	String
billPhoneNumber	labelRequest	Yes	String
billEmail	labelRequest	No	String
destinationFacilityId	labelRequest	Yes	String
destinationFacilityName	labelRequest	No	String
packedDateTime	labelRequest	Yes	Date
trackingNumber	labelRequest	No	String
EAN128	labelRequest	No	String
weight	labelRequest	Yes	Numeric
serviceText1	labelRequest	No	String
serviceText2	labelRequest	No	String

routingText	labelRequest	No	String
customOrderPrty	labelRequest	No	String
refNum4	labelRequest	Yes	Numeric
items	labelRequest	Yes	Array
distributionOrderLine	items	Yes	String
sku	items	Yes	String
ean	items	No	String
quantity	items	Yes	Number
oms_guid	item	No	String

Table 2 – CTT Label request mappings (Source: own)

This example shows a regular OK response (return code = HTTP 200 OK) with a generated Tracking Id & Label from microTMS. The response will always be a message on JSON format.

Field name	Parent node	Mandatory	Type and Size
trackingNbr	label	Yes	String
returnTrackingNbr	label	Yes	String
fileContents	label	Yes	String

Table 3 – CTT Label response (Source: own)

CTT Label response output example:

{ "label": {
"trackingNbr": "FA272000371PT",
"returnTrackingNbr": "FA272000371PT",
"fileContents":
"^XA\n^SZ2^MCY~TA0~JSN^LT0^MFN,N^JZY^PW806^PR6,6,6^PMN^JMA^LH0,0^LRN^XZ\n^XA^DFR:SSF0.ZPL\n^CI28,36,21\n^LH0,0\n^JMA\n^PW816\n^LL\n\n^FO705,170^GFA,2296,2296,14,,:::::O0JF8,N0LFC,M0NFC,L03OF,K03PFE,K07QF,K0RFC,J03RFE,J07SF,J0TFC,I01TFE,I03UF,I07UF8,I0VF8,001VFC,003VFE,:003WF,007WF,00XF8,:00XFC,01MF8I07LFC,01LF8K0LFE,01KFEL03KFE,03KF8M07JFE,03JFEN03KF,03JFCO0KF,07JF8O07JF,07JFP03JF,07IFEP03JF,07IFEP01JF8,07IFCP01JF8,07IFCQ0JF8,07IF8Q07IF8,:::0JFR07IF8,::07IFR03IF8,:07IFR07IF8,::::07IF8Q07IF,:::03IF8Q07IF,03IF8Q0JF,03IFCQ0JF,03IFCQ0IFE,01IFCP01IFE,:01IFEP03IFC,00IF8Q0IFC,00FET03FC,008,,::::::::::::L0WFC,I01YFE,I07YFE,001gGF,007gGF8,00gHF8,00gHFC,01gHFC,01gHFE,03gHFE,03gIF,07gIF,07gIF8,:07gIFC,:07gIFE,:07gJF,0gKF,0gKF8,07JFQ0JF,07IFCQ0JF,07IF8Q0JF,:07IFR0JF,::07FFER0JF,::07IFR0JF,:::,::::::::::::J03XFE,I07YFE,001gGF,003gGF,007gGF8,00gHF8,00gHFC,01gHFC,03gHFE,:03gIF,07gIF,07gIF8,:07gIFC,:07gIFE,:0gKF,0gKF8,07gJF8,07JFQ0JF,07IFCQ0JF,07IF8Q0JF,:07IFR0JF,::07FFER0JF,::07IFR0JF,:::^FS\n\n^FX
********** Shipment Zone **********\n^FO15,25^A0N,25,15^FDCTT Expresso | Av. D. João II, 13 1999-001 LISBOA | Alvará: 654940 | NIPC: 503630330^FS\n^FO0,0^GB816,15,15^FS\n^FO0,300^GB510,0,3^FS\n^FO0,480^GB816,0,3^FS\n^FO300,560^GB516,0,3^FS\n^FO0,600^GB816,0,3^FS\n^FO0,736^GB816,0,3^FS\n\n^FO0,50^GB816,0,3^FS\n^FO300,300^GB0,300,3^FS\n^FO510,50^GB0,430,3^FS\n^FO700,50^GB0,430,3^FS\n\n\n\n^FX ********** Delivery information **********\n^FO30,70^A0N,25,25^FDDestinatário:^FS\n^FO30,100^A0N,35,35^FDAntonio Gemelgo^FS\n^FO30,140^A0N,35,35^FDRua Tomas Pilar 1 ^FS\n^FO30,180^A0N,35,35^FD^FS\n^FO30,220^A0N,35,35^FD1300258^FS\n^FO30,260^A0N,35,35^FDLisboa^FS\n\n^FX
********** Customer information **********\n^FO20,320^A0N,25,25^FDContacto:^FS\n^FO20,350^A0N,25,25^FDAntonio Gemelgo^FS\n^FO20,420^A0N,25,25^FDTelf:^FS\n^FO20,450^A0N,25,25^FD965453980^FS\n\n^FX ********** PUISor HD **********\n\n^FO20,500^A0N,25,25^FD"XPTO" ONLINE PT^FS\n^FO20,540^A0N,70,70^FD1460 CTT^FS\n\n\n^FX ********** oLPN physical properties **********\n^FO320,330^A0N,25,25^FDVolumes:^FS\n^FO320,365^A0N,25,25^FD2^FS\n^FO320,400^A0N,25,25^FDPeso:^FS\n^FO320,435^A0N,25,25^FD4.1200 Kg^FS\n\n^FX ********** Sender information **********\n\n^FO670,80^A0R,25,25^FDRemetente:^FS\n\n^FO515,60^A0R,25,25^FDContrato:^FS\n^FO515,155^A0R,25,25^FD300301729^FS\n\n^FO515,285^A0R,25,25^FDCliente:^FS\n^FO515,365^A0R,25,25^FD200040094^FS\n\n^FO640,80^A0R,25,25^FD"XPTO"^FS\n^FO610,80^A0R,25,25^FD2050-999 OTA ^FS\n^FO580,80^A0R,25,25^FDTelf:^FS\n^FO580,140^A0R,25,25^FD808 100007^FS\n\n^FO550,80^A0R,25,25^FD^FS\n\n^FX ********** Content description **********\n\n\n^FX ********** WMOS License Plate **********\n^FO50,610^BY2^BC,100,Y,N,N^FD256019880034758868^FS\n\n\n^FX ********** Shipment numbre OMS

**********\n^FO320,500^BCN,50,N,N,N^FD33283676_001^FS\n^FO320,570^A0N,25,25^FD33283676_001^FS\n\n^FX ********** Routing Zone **********\n\n ^FO550,750^A0N,35,35^FDProduto 19^FS\n\n ^FO600,795^A0N,45,45^FD 02/10/20^FS\n\n\t\t^FO70,755^A0N,30,30^FDOBJ ID: FA272000371PT^FS\n\t\t^FO110,795^A0N,30,30^FDCP7: 1300258 ^FS\n ^FO150,920^BY3^BCN,200,N,N,N^FDFA272000371PT^FS\n ^FO370,1130^A0N,15,15^FDFA272000371PT^FS\n\n \n^XZ\n^XA^XFR:SSF0.ZPL^PQ1,0,1,Y^XZ\n^XA^IDR:SSF0.ZPL^XZ\n"
}

The CTT label has the below requirements in order to be accepted by CTT platform for delivery:

1) CTT Label Specific Properties

White paper, size 10x15cm

Font type – Courrier

CTT Expresso logo

Bar Code: CODE39

2) CTT Label Required Elements

Carrier Identification (Logo)

3) CTT Label Fields

1. Header – Mandatory
    a. Logo
    b. CTT Product Type ***(a)***
        i. Fixed value: 19
    c. Number of volumes shipped
        i. Fixed value: 1
    d. Shipped date
2. Sender - name, address, phone number – Mandatory **(a)**
    a. XPTO
    b. EN 1
    c. OTA
    d. 2050-999 OTA
    e. Portugal
3. Recipient - name, address, phone number – Mandatory
    a. Client name
    b. Client phone number
    c. Client address
    d. Client city
    e. Client postal code - city
    f. Client country
4. Observations – "XPTO" internal code of order status

5. Contact
6. Tracking Number & Bar Code – Mandatory
7. Client Nº – Mandatory **(a)**
8. Contract Nº – Mandatory **(a)**
9. Shipped volumes – weight -30Kg – Mandatory
10. Fixed Sentence – Mandatory **(a)**
    a. "CTT Expresso | Av. D. João II, 13 1999-001 LISBOA | Alvará: 654940 | NIPC: 504520296"

(a) All these values need to be saved on a parametrization table on moore-tms-configurations-*. This approach will be used to keep this fields more flexible in case of future changes.

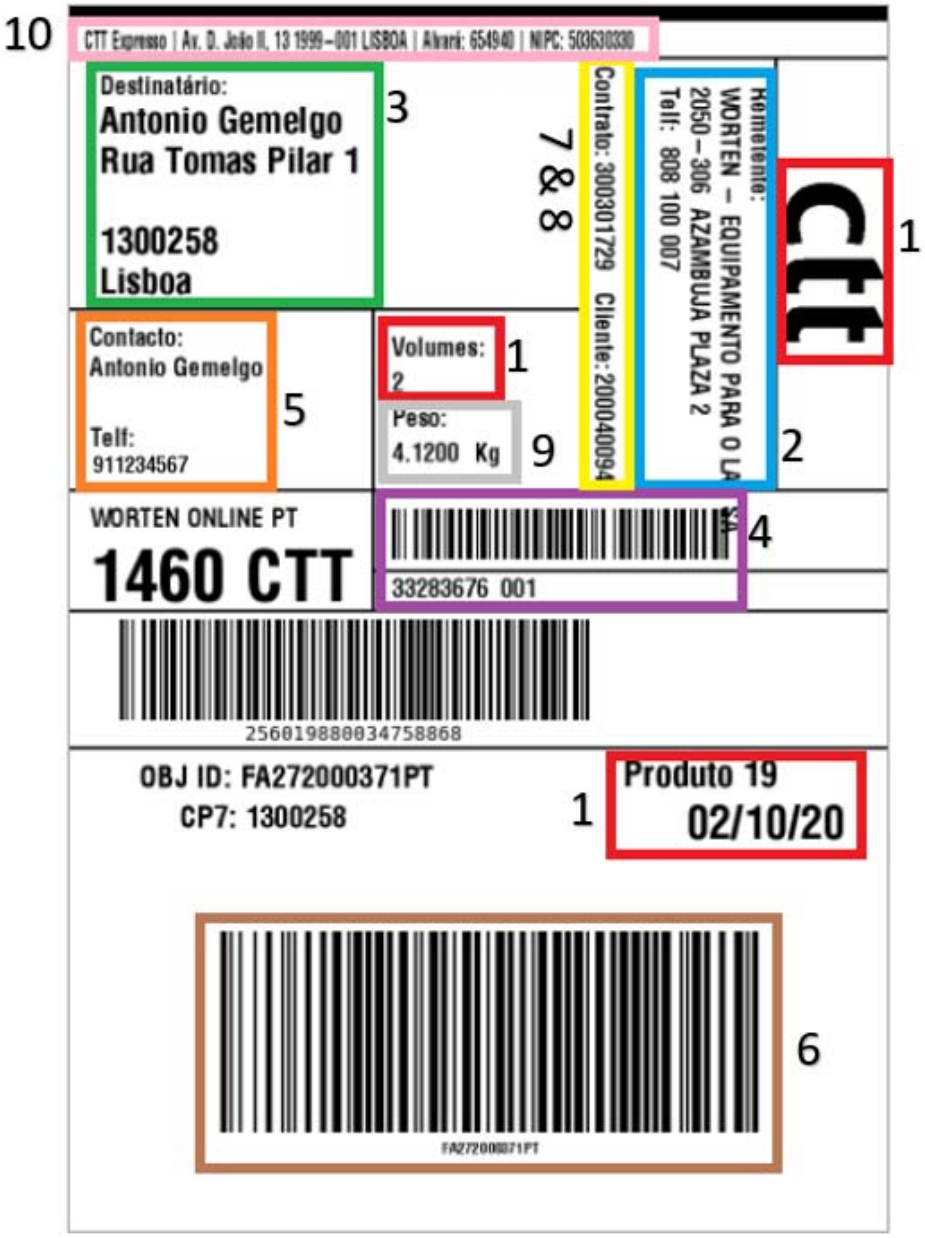


Figure 27 – CTT label (to-be)

This example shows a regular NOK response (HTTP 500 Bad Request) with an Error Code and related Description from microTMS. The response will always be a message on JSON format.

```
{
  "Status": {
    "errorCode": "500",
    "errorDescription": "InternalServerError",
  }
}
```

Field name	Parent node	Mandatory	Type and Size
errorCode	status	Yes	String
errorDescription	status	Yes	String

Table 4 – CTT Label NOK error message

The CTT exception label will be generated with the errorCode and errorDescription, see figure 39, sent within the Bad Request response in order to facility operations troubleshooting or corrective actions.

Figure 28 – CTT Exception label printout

4.3.2.12.CTT Tracking Numbers

In order to keep track of the deliveries it is required to use valid CTT Tracking numbers ranges explained below.

Example: ***FA8302500616PT***

FA*_____** → Type of CTT product used (Fixed Value ***(a)).

__830_____ → ID specifically assigned by CTT to "XPTO" (Fixed Value ***(a)***).

____50061___ → Sequence for "XPTO" deliveries. Range of 99999 possible values, from 00001 to 99999. ***(b)***

_____6__ → Check Digit. Documentation about this on the next topic (How to calculate CTT Check Digit from a Tracking Number).

_____PT → Fixed value for CTT Deliveries (national and international) (Fixed Value ***(a)***).

(a) All these values need to be saved on a parametrization table on microTMS DB. This approach will be used to keep this fields more flexible in case of future changes.

(b) *830 until 839* represents an ID specifically assigned by CTT to "XPTO" on microTMS only.

How to calculate CTT Check Digit from a Tracking Number

Example: ***FA83050061_PT*** → ***FA830500616PT***

1º: (8*8)+(3*6)+(0*4)+(5*2)+(0*3)+(0*5)+(6*9)+(1*7) = 64+18+0+10+0+0+54+7 = ***143***

2º: MOD(143/11) = 12,***45*** → ~= ***5***

3º Checkdigit = 11-5 = ***6***

4.4. ARCHITECTURE AND SOLUTION DESIGN

4.4.1. Architecture

The Architecture design was aligned according with AWS services available and which best suited an Event Oriented Architecture based on Lambda Functions.

Communication between WMS and WhTF endpoints will be done via HTTPS with SSL certificate through API Gateway, called Kong, which is responsible to route all the requests to the internal services responsible for handling those requests which in turn triggers a Lambda function that validates the request.

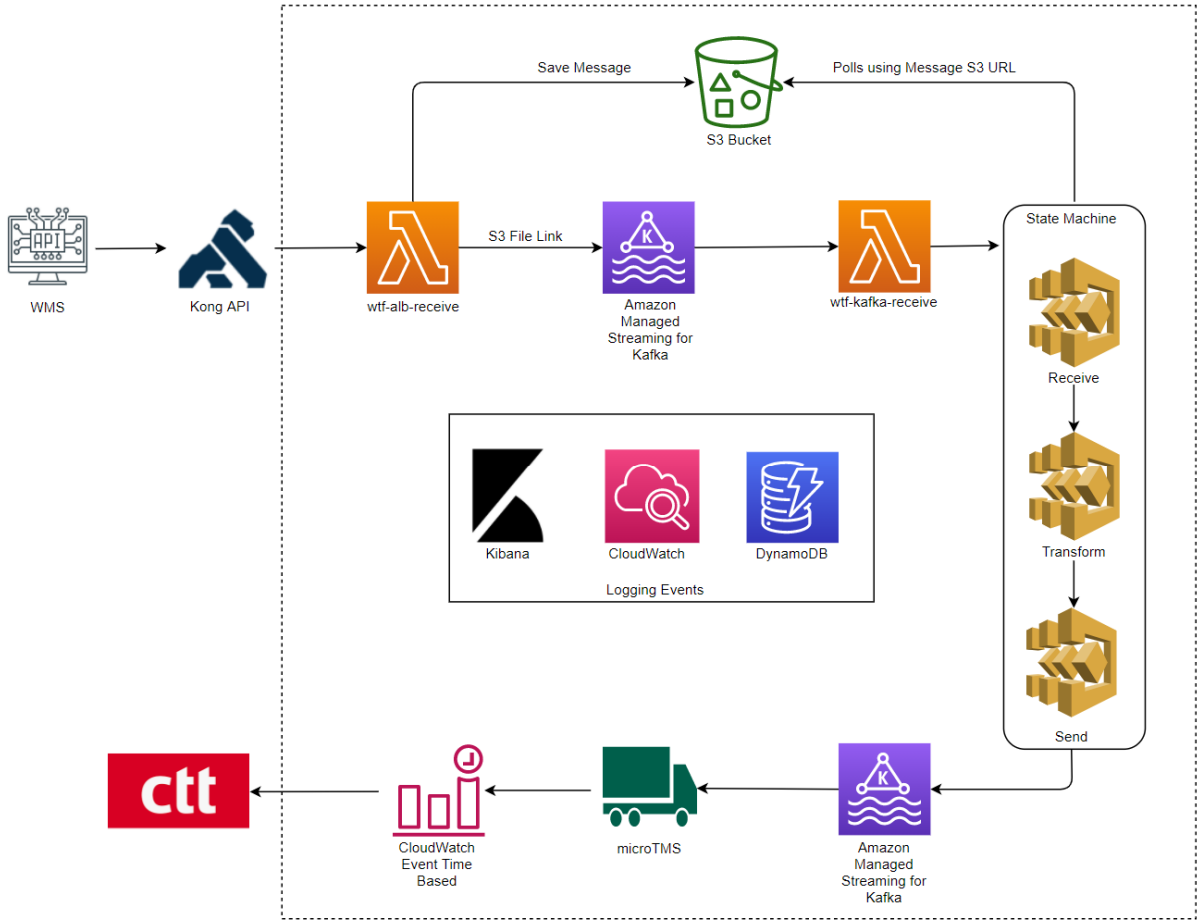


Figure 29 – Architecture Design – Event Oriented Architecture based on Lambda Function (Source: own)

Because of time constraints and based on Scrum methodology a MVP version was established for delivery in order to grant value added to business.

WhTF MVP focus on handling Shipconfirm and Trackind Id & Label messages from WMS <-> microTMS.

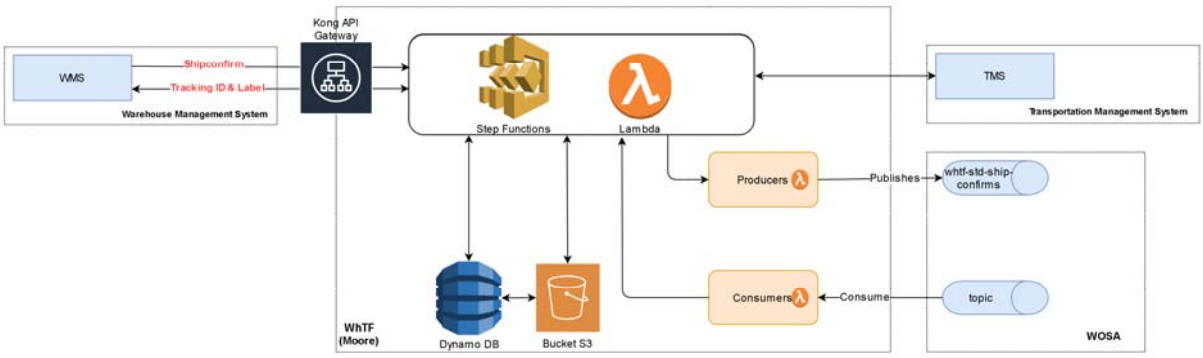


Figure 30 – WhTF MVP Solution Design (source: own)

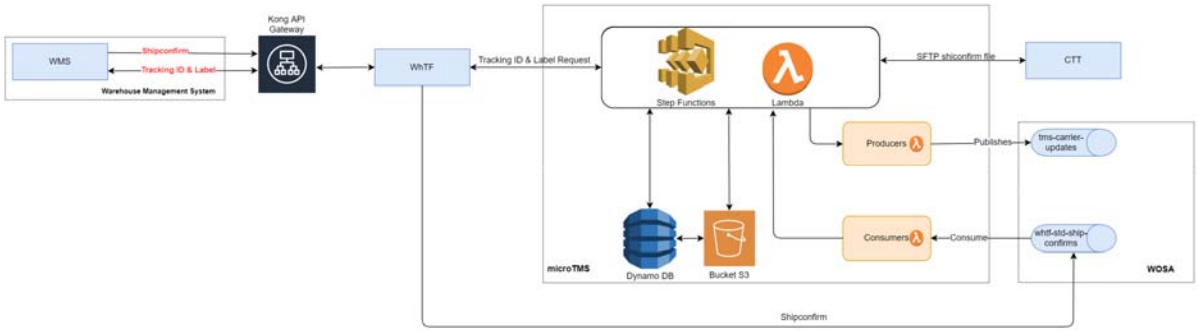


Figure 31 – microTMS MVP Solution Design (Source: own)

4.4.2. Interfaces

4.4.3. Data Model

The DynamoDB data model in Amazon DynamoDB table is a collection of items and each item is a collection of attributes. Each attribute is a name value pair and can contain a single value but Json document or a set of values.

When we create a table in addition to the table name we must specify a primary key to use on the table as another database's a primary key in DynamoDB uniquely identifies each item in the table.

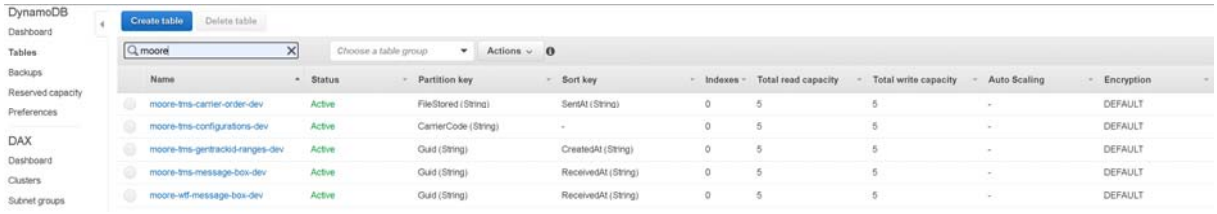


Figure 32 – DynamoDB tables (Source: own)

The type of primary key is a composite key using both a partition key and a sort key. The first attribute is the partition key and the second attribute is a sort key again DynamoDB uses the partition key value as input to an internal hash function.

The output from the hash function determines the partition where the item is stored. All items with the same partition key are stored together in a sorted order by the sort key value.

The *moore-wtf-message-box-** table will have all WhTF messages handled and the output for each column is:

Guid: S3 ID from original message received

ReceivedAt: timestamp of message received

FlowId: message flow Id

FlowId	Meaning	Obs
0	ShipConfirm	Indicates the ShipConfirm flow
1	GenTrackId	Indicates the Generate TrackId flow

Figure 33 – WhTF Flow Id (Source: own)

FlowStep: message Flow Step

FlowStep	Meaning	Flow	Obs
0	ReceivedInAlb	Initial	When The ShipConfirm is received from ALB Proxy
1	ValidationPassed	Mid - Success	When The ShipConfirm received in WhTF, received Lambda has fields full compliance
2	Error	Mid - Error	When The ShipConfirm received in WhTF, received Lambda has any field(s) not compliance
3	Transformed	Mid - Success	When The ShipConfirm received in WhTF, transform Lambda has fields full compliance and has been transformed
4	Sent	Final	When The ShipConfirm received in WhTF, send Lambda and sent successfully to WOSA

Figure 34 – WhTF Flow Step (Source: own)

MessageIn: Standard message conversion

MessageOut: Standard message published into kafka

SentAt: timestamp of kafka publishing

ReceiveBy: Lambda name handling

SentBy: Lambda name handling

DequeuedAt: timestamp of kafka private handling

DeQueuedBy: Lambda name handling

QueueDeliveryResult: kafka private output

StepFunctionARN: state machine handling information

ValidateBy: Lambda name handling

ValidateAt: timestamp of lambda message validation

OriginalMessageS3Name:

Error: Error description

ErrorAt: timestamp of error occurrence

When we create or update a table we specify how much provision throughput capacity we need for reads and writes and Amazon DynamoDB will automatically allocate the necessary machine resources to meet our throughput needs while also ensuring consistent low latency performance throughput is measured in capacity units a unit of read capacity represents one strongly consistent read per second or eventually consistent reads per second for any item as large as 4 kilobytes. A unit of write capacity represents one right per second for items as large as one kilobyte DynamoDB supports two operations.

The *moore-tms-message-box-** table will have all microTMS messages handled and the columns explanation is as follows:

Guid:

ReceivedAt: timestamp of message received

FlowId: message flow Id

FlowId	Meaning	Obs
0	ShipConfirm	Indicates the ShipConfirm flow
1	GenTrackId	Indicates the Generate TrackId flow

Figure 35 – microTMS Flow Id (Source: own)

FlowStep: message Flow Step

FlowStep	Meaning	Flow	Obs
0	Received	Initial	The message box has this state when microTMS Received Lambda function put it in database
1	Converted	Mid - Success	The message change to this flowstep when the microTMS transform Lambda has converted the message and modify flowstep in database to '2'
2	SentToCarrier	Mid - Success	The message has this step in database when all messages of the day were sent to CTT (this process is made at the end of the day)
3	Error	Mid - Error	To address failures in sending, validating or processing the message
4	Sent	Mid - Success	When the message has been sent to WOSA without issues
5	GenTrackIdLabeL	Final	When microTMS received message from WhTF and generates Track Id and Label
6	Dequeued	Mid - Success	The message box has this state when the message is dequeued from kafka

Figure 36 – microTMS Flow Step (Source: own)

DequeuedAt: timestamp of kafka handling

DeQueuedBy: Lambda name handling

StepFunctionARN: state machine handling information

CarrierId:

MessageReceiveInJson: Kafka message received

ReceivedBy: Lambda name handling

SentAt: timestamp of kafka publishing for OMS

MessageOut: response message of trackingId

FileStored: CTT file name generated

SentBy: Lambda name handling

SingleMessageConverted: CTT messages conversion

SingleMessageSentToWosa: CTT tracking numbers message sent to OMS

TcShipmentId: WMS shipment

TransformedAt: timestamp of lambda message transformation

TransformedBy: Lambda name handling

Because of Tracking Number generation, table *moore-tms-gentrackid-ranges-** was created where all the attributes required are maintained and accessed through software coding namely:

- Valid range numbers;
- Active ranges;
- Date of start and finish of range;
- Sequence in use.

The below columns have the valid ranges for tracking numbers attribution regarding CTT carrier.

Guid	CreatedAt	CarrierCode	CountryCode	FlagActive	LastValue	ProductTypeCo	ShipperId	StartDate
2519215f-c4c6-4c20-b8b8-d179ee18a20e	2020-11-10T20:23:20.398Z	CTT	PT	false	0	FA	836	
41bee8d6-fb9c-45ee-992f-2ba7fda15a7d	2020-11-10T20:23:20.348Z	CTT	PT	false	0	FA	830	
5d801639-3e71-40e5-bc5b-7cd31e8746c9	2020-11-10T20:23:20.388Z	CTT	PT	false	0	FA	834	
5da83a3c-3a10-4d3f-8bf9-a164821e26a6	2020-11-10T20:23:20.408Z	CTT	PT	false	0	FA	838	
77cd8a6e-2587-4edc-9603-4594213c849b	2020-11-10T20:23:20.393Z	CTT	PT	false	0	FA	835	
9fa5d45b-f7e4-4138-bcf9-4a04a5de9725	2020-11-10T20:23:20.377Z	CTT	PT	false	0	FA	832	
aa4c0cdb-a505-4b29-a9d0-42abc1a61493	2020-11-10T20:23:20.403Z	CTT	PT	false	0	FA	837	
e797b437-f079-4ec0-b49c-36aef585a073	2020-11-10T20:23:20.383Z	CTT	PT	false	0	FA	833	
f86a0d52-7e04-4d23-948c-7600403513e4	2020-11-10T20:23:20.372Z	CTT	PT	true	28462	FA	831	2020-11-12T15:32:15.517Z
fa5ed07a-b6af-4bc4-9e5d-52ac7d87f63	2020-11-10T20:23:20.414Z	CTT	PT	false	0	FA	839	

Figure 37 – CTT tracking numbers (Source: own)

Since microTMS will handle several carriers and everyone will for sure have distinct requirements and configurations regarding labels printout it was decided that those definitions will be keep not hardcoded but instead they will be stored on a DynamoDB table, *moore-tms-configurations-**, that will be invoked to pass the required fixed values to populate the label template.

Figure 38 – Table moore tms-configurations-dev definitions (Source: own)

The *moore-tms-carrier-order-** table has the “metadata” for ctt file generation and information to be sent to internal systems.

FileStored: File sent to CTT SFTP

SentAt: timestamp of CTT SFTP publishing

CarrierId:

CountEMS: total of messages treated

FlowStep: message Flow Step

FlowStep	Meaning	Flow	Obs
0	Sent	Initial	The CTT file was generated and sent to CTT SFTP
1	Processed	Final	The CTT file was processed successfully
2	Error	Mid - Error	The CTT file was sent and had processing errors

Figure 39– microTMS Flow Step (Source: own)

MessageSent: S3 Id

SentBy: Lambda name handling

WosaConfirmationMessage: kafka confirmation message of all messages integration

Error: Error description

ErrorAt: timestamp of error occurrence

DynamoDB is a fast, highly scalable NoSQL database which best fits our needs namely:

- Large number of small writes and reads
- Simple data models, transactions, simple updates
- Performance – automatically optimized by the system
- Reliability and availability
- Durability
- Low cost

4.4.4. Coding

As we all know, all projects start out with just a handful of files. After working for some time on the project, the codebase increases. In the end, the project becomes huge. Maintenance is hard. And we haven’t even talked about adding new features — which is painful, at best. Even making small changes takes hours. Bug fixing seems to be a never-ending story. Fixing one bug always introduces at least one new bug.

To prevent the situation described above from happening, it's important that no more code than is intended gets written. Don't create extra abstractions to support some feature that might get implemented someday. You should value simplicity over feature-richness. Only implement things when you actually need them, not when you just foresee you may need them. This is called the "You aren't gonna need it" (YAGNI) principle.

Writing in 2 lines of code what is written in 10 lines it's easier to read and understand. Remember that good code can be read "by a 3 years old baby"! Keep it simple!

4.4.4.1.Branching Strategy to use in code

The branching strategy is GitFlow.

We follow the basic Gitflow model but with a couple of changes:

- our "develop" branch is an actual developers-only integration branch;
- we have a "next" branch that contains the feature branches that the developer is ready to move to testing and acceptance;
- master is merged into release branches before each build of the release branch.

Branches

A brief explanation on the columns:

- Branch type, the type of branch. Branches of the same type share semantic significance in the process;
- Availability: indicates if the branch type is always present or if it is an optional branch. It also indicates if there is one or more than one branch of this type;
- Branch name: full name or name template for the branch;
- Description: high level description of the objective of this branch.

Branch type	Availability	Branch name	Description
Production	single, always present	master	The master branch contains the deployed code, builds that were approved by business and passed regression testing

Release	multiple, but only one active. Usually present	release-<fix-version>	The release-* branches represent a specific JIRA fixVersion. This is branched from master, and we merge "next" and feature branches into it. **No development allowed on release branches**, we can merge more feature or bug fixing branches, but we don't actively develop on release branches. Focus of a release branch is to make sure that testing and acceptance phase pass
Hot fix	multiple, hopefully never present	hotfix-<ticket-number>	Branch of master, to fix a regression or blocker bug in **production**. Mostly behaves like a feature branch, but has an abbreviated test/approval cycle. **Do not use** for regular bugs, see feature.
Next Release	single, always present	next	A staging area for tickets (stories, bug fixes) that are considered "Ready to Deploy" by developers. It is assumed that branches only merge into "next" after reasonably tested on integration branches and TST environment by the developer
Feature	multiple, hopefully always present	<ticket-number>-<optional description>	Each story or normal bug fix is developed in a separate branch. Branch is started from "next".

Table 5 – Branch Types (Source: own)

The "master" branch holds all the code that was approved by QA and Business and passed regression testing. You can have multiple builds in master, but not deploy them. The current deployed build has a git tag named "live" pointing to it. All the code in master is ready to deploy at any moment, but business considerations can (and often will) postpone a release. This is due mostly to risk control, based on the current level of stability and coverage of our automated and manual test suites.

Most of the fun happens in "next". This is branch where all the feature branches that are ready to be tested, accepted and pushed into the next release will be merged by developers. This is also the branch that is used by developers as the root of their feature branches. "master" might be several commits behind "next". Please note that "next" is a long lived branch, never reset, never recreated. You should see "next" as an holding area for branches that are ready to be tested, but not in the currently active "release" branch, only on the next one.

The "feature-*" branches is where the work is done by developers. They branch of the latest "next", and will remain here until code review is approved. Developers are free and encouraged to merge often with "develop" branch, for integration testing. The feature branch should be merged to "next" when and only when:

- a successful merge with "develop" was done;
- this successful merge was successful deployed to TST environment;
- basic tests by the developer for the feature were performed in the TST environment with this successful deployment of the successful merge.

If these conditions hold, the developer is free to merge to "next". If a release is already being tested, the release manager can also ask a developer to merge the "feature" branch directly to the "release" branch. This is used when a bug fixes on feature branches: deploy directly to the active "release" branch and "next".

Usually at the start of a sprint, a new "release" branch is created. It is created from master and immediately receives a merge from "next" with all the pending branches ready to test. From this branch, builds for testing and acceptance are created.

Stories that should still make the open release branch and were not in the next branch are free to merge

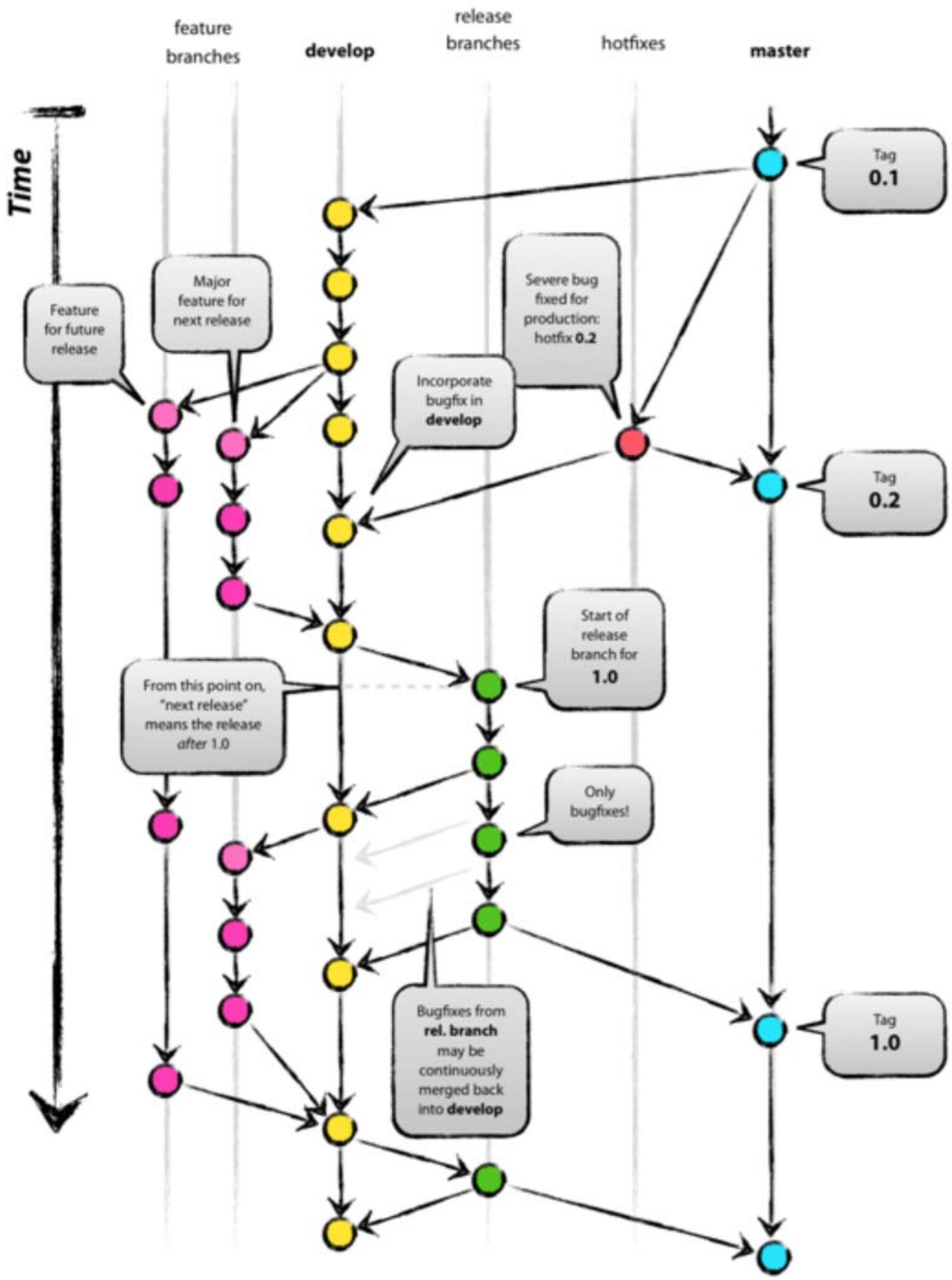


Figure 40 – Branching Strategy to use in code (Source: Introduction to GitLab Flow | GitLab. (n.d.))

4.4.4.2.Deployment Process

User Story/Task, Bug Workflow and Lifecycle in Jira

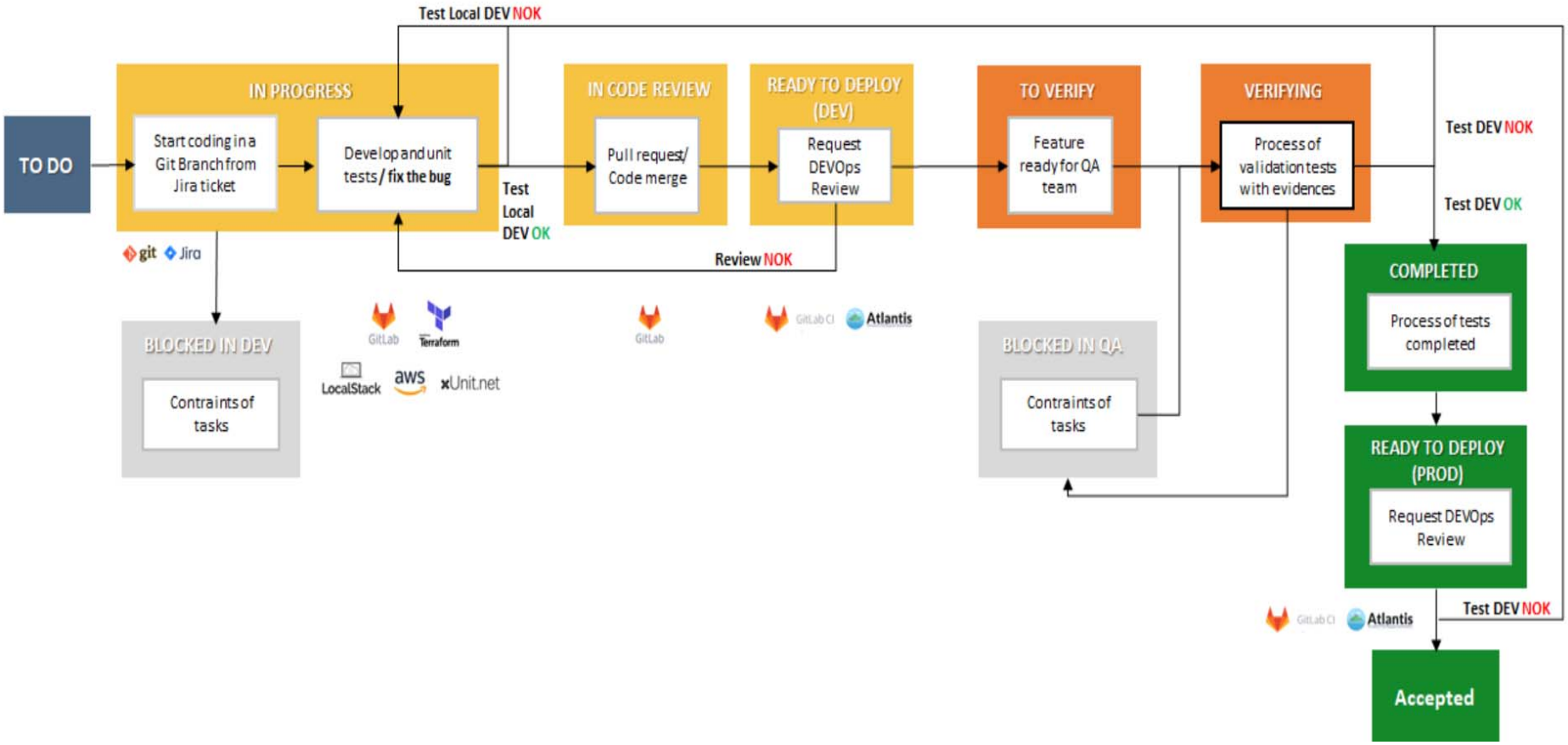


Figure 41 – Deployment Process Flow (Source: own)

The process is divided in 3 major development stages, with several statuses characterizing each, broadly speaking:

1. **To Do:** a task as been analyzed and included in a sprint, in the case of features (User Story or Task), it's been groomed
2. **In Development:** the developer as started working on the User Story/Task/Bug
    a. In Progress: the developer creates an appropriately named branch and starts development of the feature/analyses the issue, and creates any unit tests necessary
    b. In Code Review: when a Pull request is created and the rest of the development team is asked to review the code
    c. Ready to Deploy (DEV): a request to the DevOps Team is made using the Team Channel, to review any terraform/infrastructure changes
    d. To Verify: the code is ready for QA (Quality Assurance) team
    e. Verifying: a member of the QA team as started the QA process
3. **Development completed:**
    a. Completed: QA team tests successful (with evidences) the feature/bug is ready to deploy to production
    b. Ready to deploy: a request to the DevOps Team is made using the Team Channel, to review any terraform/infrastructure changes, a precautionary step since the code should not have changed from step "2. c)" above
    c. Accepted: the internal "XPTO" team validates the feature/bug fix the days following the sprint review process

The blocked statuses (in Dev and QA) should be used sparingly, if a new dependency was identified only during the development stage for example, or some part of the infrastructure/development environment is made unavailable for an extended period, these should be exceedingly rare.

The following table summarizes what should happen in each step before progressing to the next:

ENVIRONMENT	TRANSITION	RESPONSIBLE	DESCRIPTION	JIRA COMMENT	COMMENT EXAMPLE ON JIRA
LOCAL/DEV	TO DO → IN PROGRESS	DEV	Implementation started. The issue must be assigned to the developer in Jira. In this phase is defined a new branch in GitLab according with the number of the task.	Not required	feature/ASC-XX-name-of-task
	IN PROGRESS → IN CODE REVIEW		The ticket was implemented and tested **SUCCESSFUL** by the developer locally. In this transition the code is merged into DEV	Not required	Hi @###, Tested successfully in DEV environment. Thanks, XXXX
	IN CODE REVIEW → READY TO DEPLOY		Request DevOps Review to deploy for DEV. Use the following Team Channel to ask the DevOps team for aproval of the Terraform elements in the Merge Request.	Mandatory	The MR to Dev is waiting DevOps team review.
	READY TO DEPLOY → TO VERIFY	QA	In this phase the feature/bug is ready to be tested by the QA team.	Mandatory	Hi @QA Team, Can you please test this feature/bug
	IN PROGRESS → BLOCKED IN DEV	DEV	The ticket is blocked by a dependency in DEV. The ticket is assigned to the responsible for the dependency.		Hi @###, This task is being blocked due to: (...)
DEV/ PROD	TO VERIFY → VERIFYING	QA	The tests are assigned to a member of QA .	Mandatory	
	VERIFYING → COMPLETED	DevOps / QA	If tested **SUCCESSFULLY** by the QA team in DEV environment and is ready to be the next phase "Completed". The test **FAILS** in DEV environment by the QA team. The tester must assign the ticket again to the developer and update the ticket status to DEV in progress. The QA Team should add evidences of the working feature/fixed bug in the comment Request a review to DevOps to deploy for PROD.	Mandatory	If **FAILS**: Hi @###, There was an error in this feature (...) Following this steps: (...) 1. (...) 2. (...) The following occurred: add evidences of the issue The expected result: If **SUCCESSFULLY**: Hi, QA completed with the following evidence: (images). The MR to Prod is waiting DevOps team review.
	COMPLETED → ACCEPTED	Solution Designer/Project Manager/ Product Owner	The client should validate the acceptance criteria in the days following the Sprint Review and after testing the status should be changed to "Accepted".	Not mandatory but advisable	
	IN PROGRESS → BLOCKED IN QA	QA Team	The ticket is blocked by a dependency. The ticket is assigned to the responsible for the dependency.		Hi @###, This task is being blocked due to: (...)

Figure 42 – Deployment Process Stages (Source: own)

4.4.4.3.Project structure

Application using AWS Stack and .Net Core 3.1 with CI/CD in GitLab and IaC in Terraform and Automated Regression Tests included in the CI/CD Pipeline.

Gitlab-ci.yml

This file, see annex 2, configures the pipeline of the project. It defines what to execute and what to do when a condition is met, i.e. when to succeed or fail. The project is multi-staged, each stage contains scripts and is defined globally for the pipeline. Jobs of a stage are run in parallel and each stage is run only after the previous stage's jobs are successfully completed.

The “Run Integration Tests” stage uses a dockerfile to define the application's image and a docker-compose to define the full service environment required to tests it. These file types and their use in this project are covered in the next section, for now it's important to notice only the "docker-compose up" and "docker-compose down" lines, these will deploy all images defined in docker-compose and then removes the containers that were created for the services defined in the same file.

Docker-compose

Docker-compose was used to define a multi-container Docker application environment to run integration testing before deployment.

The docker-compose.yml, annex 3, file configures the application's services, then creates all the defined images in the same host. Moore depends on Localstack, Kafka, Zookeeper, and the DynamoDB at this point in development, and the docker-compose.yml file reflects this need.

Adding services to docker-compose is a matter of either creating a custom dockerfile or finding a public image of the service and either following the example for localstack above of the "web" app example, and simply placing the new dockerfile in either a new directory or using a different name, since there should never be more than one dockerfile in the root directory of the application.

4.4.5. Testing

Regression Testing

We need to define a methodology so that we can reduce to a minimum the probability of new features or bug fixes breaking existing working features.

This is achieved with regression testing that is run at the end of each sprint. Since this is a backend project, all the regression tests can be automated, saving us the cost of paying to a QA team, and allowing the regression tests to be executed in a few minutes.

In a large system, achieving 100% code coverage is generally not cost effective. For example, some test cases are expensive to reproduce but are highly improbable. The cost to benefit ratio does not justify repeating these tests simply to record the code coverage.

4.4.6. Monitoring

The software is monitored through the collection of AWS and application metrics. These metrics will be handled by Grafana and represent the data through dashboards.

It is assumed that the Elasticsearch will be used as storage and full text search engine as is suits the use case.

For ease of use and familiarity, the Kibana application and related tools and plugins will be used as log exploring and visualization interface.

Log aggregation will be done at Node/Instance level via logstash.

Elasticsearch indices will be used to separate log messages by systems and/or components.

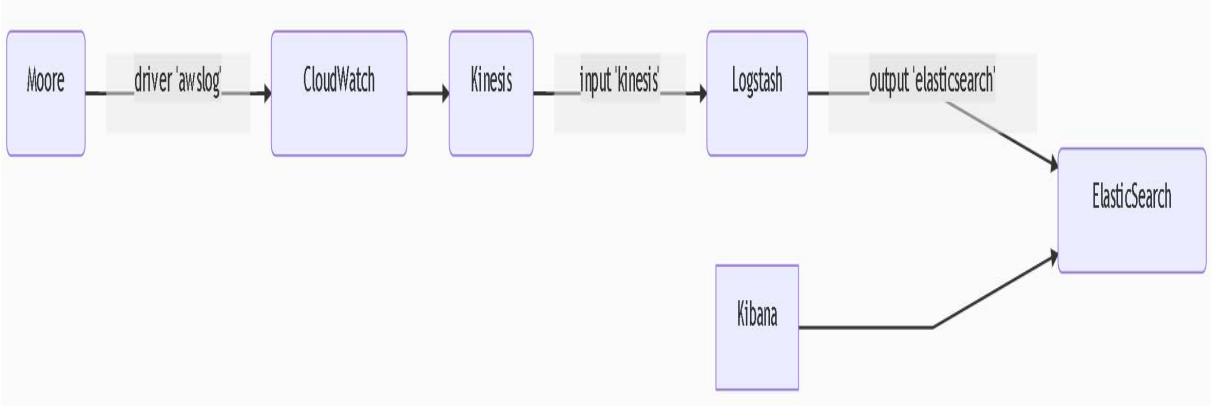


Figure 43 – Data Flow for monitoring (Source: own)

Follows a non-exhaustive list of the current architecture properties:

- Some degree of latency introduced by the Moore->Cloudwatch->Kinesis flow This isn't controlled by us.
- Current messaging format and routing requires a complex (and possibly evergrowing) list of logstash filters.
- Configurable retention on Cloudwatch.
- awslog driver simply forwards stdout to Cloudwatch but appends three bits of information: timestamp, timestamp of ingestion and logstream prefix which contains almost all the needed metadata.

5. CONCLUSIONS

The creation of this project team from the scratch raises some difficulties regarding the scrum implementation and development in AWS services because not all members were familiarized with it and this compromises on the first sprints team performance because a lot of discovery was necessary regarding AWS topics and the development of implementation tasks. Team start to gain speed with the pass of the weeks and the project went live on the 13th of November 2020 and was celebrated by "XPTO" as a huge success on the period of most demanding for operations regarding Black Friday month and Christmas period.

The adoption of microservices allow to address the need of standardize operational processes independent of the carrier for Online flow and with this we were able to mitigate risks regarding operational errors and increase the delivery performance being able to deliver more goods than before.

This project in terms of operations benefits allows to reduce 2 FTE/month and increases goods delivery by day in 1000% to CTT carrier which allows to reduce costs and deliver goods on a more expedite manner.

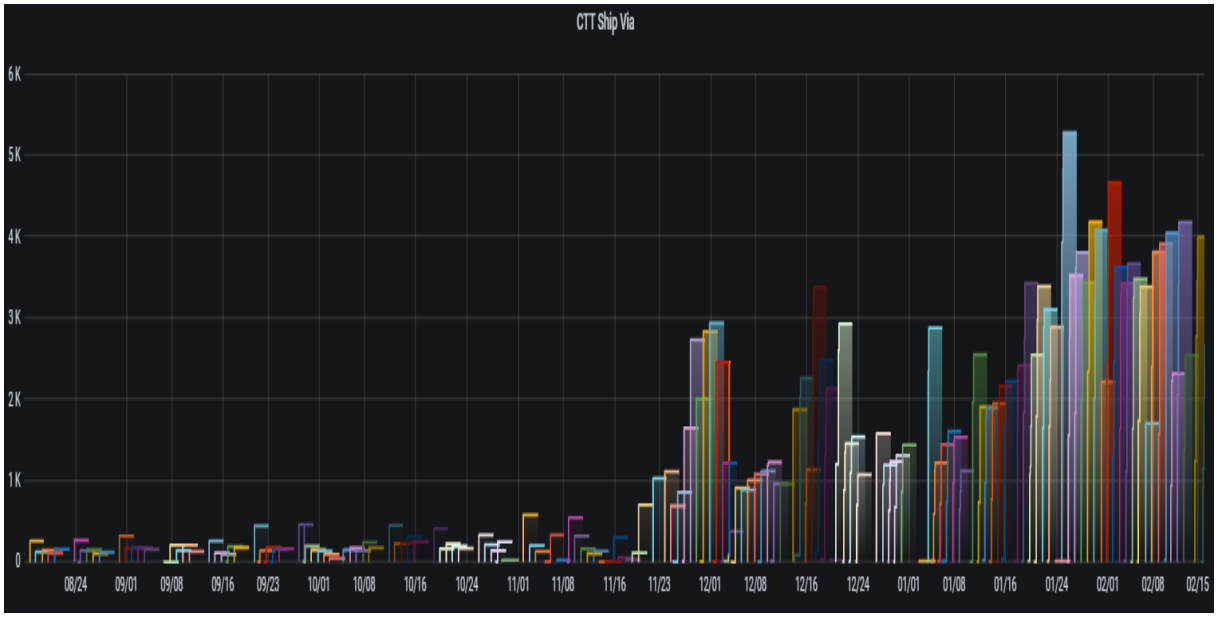


Figure 44 – CTT daily delivery

The implementation as a microservice allows to speed up developments since it is decoupled from other internal systems and gain deployment flexibility.

5.1. LIMITATIONS

This project MVP due to time constrains is focused on the CTT carrier although the middleware (WhTF) already processes all messages for all other carriers that are working with "XPTO", the microTMS development only focus on CTT for Home Delivery.

This delivery doesn't contempt a backoffice frontend for support team for incident resolution or microTMS configuration management interface for development teams but during the project some tools were developed in order to help the support team to overcome some potential incidents that may arise namely:

- Ctt file(s) not generated;
- Ctt information not sent to "XPTO" internal systems;
- Dynamodb tables maintenance (whtf and microtms);
- Reprocess error messages (whtf and microtms);

5.2. FUTURE WORKS

Develop microTMS in order to be able to process data for all current carriers and new ones including delivery modes namely:

- CTT pick up in store by the client;
- DPD home delivery and pickup in store;
- Luis Simões;
- Total Media.

For CTT and DPD labeling process must be also taken care and integrations with Tax authority should be the nexts milestones for Iberian B2C market.

In order to integrate all this carriers the dynamodb approach used must be revised because a relational model is more suitable for all this relations and dependencies and will be also extremely useful for the backoffice interface with the management interface which will allow to configure new carriers, carriers attributes, label templates, working days and others.

Dynamodb fetch of data approach must be revised to replace SCAN by QUERY in order to gain performance. Time To Live (TTL) must also be implemented in order to grant automatic maintenance on the tables for performance and save money.

BIBLIOGRAPHY

Anderson, D.J. (2010). Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press (2010)

Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M. and Tamburri, D. A. DevOps: Introducing Infrastructure-as-Code. IEEE. 2017, p. 497–498.

Atlantis. What Is Atlantis? TEXu [online]. [cit. 2020-21-07]. Available at: https://www.runatlantis.io/guide/#getting-started.

Awad, M. A. (2005). A Comparison between Agile and Traditional Software Development Methodologies, School of Computer Science and software Engineering, The University of Western Australia.

Ballou, R. H. (2004). Business logistics – supply chain management planning, organizing and controlling the supply chain. 5th ed: - New Jersey : Pearson / Prentice-Hall

Basic principles of cloud-native development. (n.d.). [Image]. https://medium.com/velotio-perspectives/cloud-native-applications-the-why-the-what-the-how-9b2d31897496

Berman, S. J. (2012). Digital transformation: opportunities to create new business models. Strategy & Leadership, 40(2), 16–24.

Bharadwaj, A., Omar, A. E. S., Pavlou, P. A. & Venkatraman, N. (2013). Digital Business Strategy: Toward a Next Generation of Insights. MIS Quarterly, 37(2), 471–482.

Bolaji, A. (2015). A cross-disciplinary systematic literature review on Kanban, Master's Thesis. University of Oulu. 62 p. Available at: http://jultika.oulu.fi/files/nbnfioulu-201502111073.pdf.

Carey, S. AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise? TEXu [online]. [cit. 2020-15-07]. Available at: https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whatsthe-best-cloud-platform-for-enterprise.html.

Chan, M. 15 Infrastructure as Code tools you can use to automate your deployments TEXu [online]. [cit. 2020-15-01]. Available at: https://www.thorntech.com/2018/04/15-infrastructure-as-code-tools/.

Chopra, S., & Meindl, P. (2007). Supply Chain Management: Strategy, planning and operations. 6 th Global Edition. Pearson Education Limited.

Crespo de Carvalho, J. (Coordinator) (2010). Logística e Gestão na Cadeia de Abastecimento. Portugal: Edições Silabo.

Coyle, J., Novack, R., Gibson, B. & Bardi, E. (2011). Transportation: A supply chain perspective. 7th edition. South-Western College Pub.

Danek, B. Why Choose Terraform Over Chef, Puppet, Ansible, SaltStack And CloudFormation? TEXu [online]. [cit. 2020-14-07]. Available at: https://selleo.com/blog/why-choose-terraform-over-chef-puppet-ansiblesaltstack-and-cloudformation.

Eneh, T. Most popular CI/CD pipelines and tools TEXu [online]. [cit. 2020-10-07]. Available at: https://medium.com/faun/most-popular-ci-cd-pipelines-and-tools-ccfdce42986.

Fitzgerald, M., Kruschwitz, N., Bonnet, D. & Welch, M. (2014). Embracing Digital Technology: A New Strategic Imperative. MIT Sloan Management Review, 55(2), 1.

Gannon, D., Barga, R. and Sundaresan, N. Cloud-Native Applications. IEEE Cloud Computing. IEEE. 2017, vol. 4, no. 5, p. 16–21. ISSN 2325-6095.

Handfield, R. (2020). What is Supply Chain Management (SCM)? Supply Chain Resource Cooperative. https://scm.ncsu.edu/scm-articles/article/what-is-supply-chain-management-scm

HashiCorp. Introduction to Terraform TEXu [online]. [cit. 2020-15-01]. Available at: https://www.terraform.io/intro/index.html.

Gebayew, C., Hardini, I. R., Panjaitan, G. H. A., Kurniawan, N. B. & Suhardi. (2018, 22-26 Oct. 2018). A Systematic Literature Review on Digital Transformation. Bandung - Padang, Indonesia: 2018 International Conference on Information Technology Systems and Innovation (ICITSI).

Introduction to GitLab Flow | GitLab. (n.d.). Introduction to GitLab Flow. https://docs.gitlab.com/ee/topics/gitlab_flow.html

Jog, C. Cloud Native Applications — The Why, The What The How. TEXu [online]. [cit. 2020-16-07]. Available at: https://medium.com/velotio-perspectives/cloudnative-applications-the-why-the-what-the-how-9b2d31897496.

Johann, S. Kief Morris on Infrastructure as Code. IEEE Software. IEEE. 2017, vol. 34, no. 1, p. 117–120. ISSN 0740-7459.

Kong Inc. (2020, September 9). Open Source API Gateway | Kong Microservices API Gateway. KongHQ. https://konghq.com/kong/

Lambert, D. M., & Cooper, M. C. (2000). Issues in Supply Chain Management, 83, 65–83

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V. et al. DevOps in practice: A multiple case study of five companies. Elsevier B.V. 2019, vol. 114, p. 217–230. ISSN 0950-5849

Mathenge, J. (n.d.). Scrum vs Kanban: A Comparison of Agile Methodologies. BMC Blogs. Retrieved January 30, 2021, from https://www.bmc.com/blogs/scrum-vs-kanban/

Min, H. (2015). The Essentials of Supply Chain Management (53). Pearson Education.

Montoya, M. (2017). Project Management & Agile Methodologies. Cprime. https://www.cprime.com/resources/blog/project-management-agile-methodologies/

Nallamala, N. The Top 7 Infrastructure As Code Tools For Automation TEXu [online]. [cit. 2020-14-07]. Available at: https://www.ibexlabs.com/top-7-infrastructure-as-code-tools/.

Overview | Prometheus. (n.d.). Prometheus. Retrieved January 24, 2021, from https://prometheus.io/docs/introduction/overview/

Pressman, R. (2010). Software engineering: a practitioner's approach. New York: McGraw-Hill companies.

Sacolick, I. (2018). What is CI/CD? Continuous integration and continuous delivery explained. InfoWorld.com. San Mateo: Infoworld Media Group.

Shapiro, J. F. (2006). Modeling the supply chain. Pacific Grove, CA: Brooks/Cole-Thomson Learning.

Simchi-levi, D., Kaminsky, P., & Simchi-Levi, E. (2008). Designing and Managing the Supply Chain. USA: McGraw-Hill.

Sjøberg, D. I & Solberg, J. (2012). Quantifying the effect of using kanban versus SCRUM: A case study, IEEE software, vol. 29, pp. 47-53.

Software Development Life Cycle phases. (n.d.). [Image]. https://ncube.com/blog/software-development-life-cycle-guide

Sommerville, I. (2011). Software Engineering, 9th Edition. Pearson Addison Wesiley.

Stolterman, E. & Fors, A. (2004). Information Technology and the Good Life IFIP International Federation for Information Processing (143). Boston, Massachusetts: Springer.

Tseng, Y. (2005). The role of transportation in logistics chain. Eastern Asia Society for Transportation Studies, 5, 1657–1672.

UML. (n.d.). What Is Unified Modeling Language (UML). Retrieved January 24, 2021, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

Vijaya, D., Traditional and Agile Methods: An Interpretation, 23 January 2013, http://www.scrumalliance.org/community/articles/2013/january/traditional-and-agile-methodsan-interpretation.

Watson, L. & Mishler, C. (2014). From On-Premise Applications to the Cloud. Strategic Finance. Montvale: Institute of Management Accountants, vol. 96, no. 2, p. 80–81. Available at: http://search.proquest.com/docview/1552717174/. ISSN 1524833X.

Waterfall model. (n.d.). [Image]. https://www.researchgate.net/figure/Stages-of-waterfall-model-Source-tutorials-point-2017_fig1_331736364

What Is Amazon DynamoDB? - Amazon DynamoDB. (n.d.). AWS DynamoDB. Retrieved January 24, 2021, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

What is Apache Kafka? | AWS. (n.d.). Amazon Web Services, Inc. Retrieved January 24, 2021, from https://aws.amazon.com/msk/what-is-kafka/

What is AWS Lambda? - AWS Lambda. (n.d.). AWS. Retrieved January 24, 2021, from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

What is digital transformation? - Digital Transformation Definition - Citrix. (n.d.). Citrix.Com. https://www.citrix.com/glossary/what-is-digital-transformation.html

What is Kibana? – Amazon Web Services. (n.d.). Amazon Web Services, Inc. Retrieved January 24, 2021, from https://aws.amazon.com/pt/elasticsearch-service/the-elk-stack/kibana/

What's the Difference? Agile vs Scrum vs Waterfall vs Kanban. (n.d.). Smartsheet. Retrieved January 30, 2021, from https://pt.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban

Yoo, Y., Henfridsson, O., Lyytinen, K. & Viktoria. (2010). The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. Information Systems Research, 21(4), 724.

Younas, M., Jawawi, D. N., Ghani, I., Fries, T. & Kazmi, R. (2018). Agile development in the cloud computing environment: A systematic review. Information and Software Technology. Elsevier B.V., vol. 103, p. 142–158. ISSN 1214-0716.

ANNEX

WMS Shipconfirm Message	Description	DataType Values	Mandatory	DataType
Header.Source	Source system	“MANH_wms” - (default value)	Y	String
Header.Message_Type	Message Type	“ShipConfirm” - (default value)	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.ShippedDttm	Shipped Date Time	example: 7/24/20 17:33	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.DateCreated		example: 7/24/20 17:33	Y	String
Message.ShipConfirm.ShipConfirmSummary.ShipConfirmHeaderInfo.ProNbr			Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.BillAcctNbr	Bill account number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.BillFacilityAliasId	Bill Facility Alias		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DoType	Distribution Order Type	“Customer Order” - default value	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.MajorOrderGrpAttr	Major Order grouping Attribute		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderType	Order Types	“ONL” - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DoStatus	Distribution Order Status	“Shipped” - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.StoreNbr	AT Code for ONL Orders		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.DistroNumber	OMS Channel Order Guide		Y	String

Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.ExtPurchaseOrder	OMS Channel Order Guide		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderBillToInfo.BillToEmail	Bill to Address Information		N	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderBillToInfo.BillToPhoneNumber	Bill To Phone Number		N	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestAddress1	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestAddress2	Destination Address Information		N	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCity	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCountryCode	Destination Address Information	PT - (default value)	Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestFacilityAliasId	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestName	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestPostalCode	Destination Address Information		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.ItemName	SKU ID / Item Name		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.OrderQty	Order Quantity		Y	Decimal
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderLineItem.ShippedQty	Shipped Quantity		Y	Decimal

Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.ShipVia	Carrier Identification for goods transportation		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcLpnId	Tc Lpn Id		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcOrderId	Order ID		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId	Shipoment Number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TrackingNbr	Carrier Tracking Number		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.ReturnTrackingNbr	Tracking Number		N	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.Weight	Total Package Weight (kg)		Y	Decimal
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TaxID	AT Code for Retail Orders (WWM07 TAX ID)		N	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.ItemName	SKU ID / Item name		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.TcOrderLineId			Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.LpnItemAttributes.ProductStatus	Warehouse ID		Y	String
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.LpnSerialNumber.Quantity	Quantity		N	Integer

Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.LpnSerialNumber.SeqNbr	Sequence Number		N	Integer
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.LpnDetail.LpnSerialNumber.SrlNbr	Serial Number		N	String

Annex 1 – WhTF Shipconfirm mappings

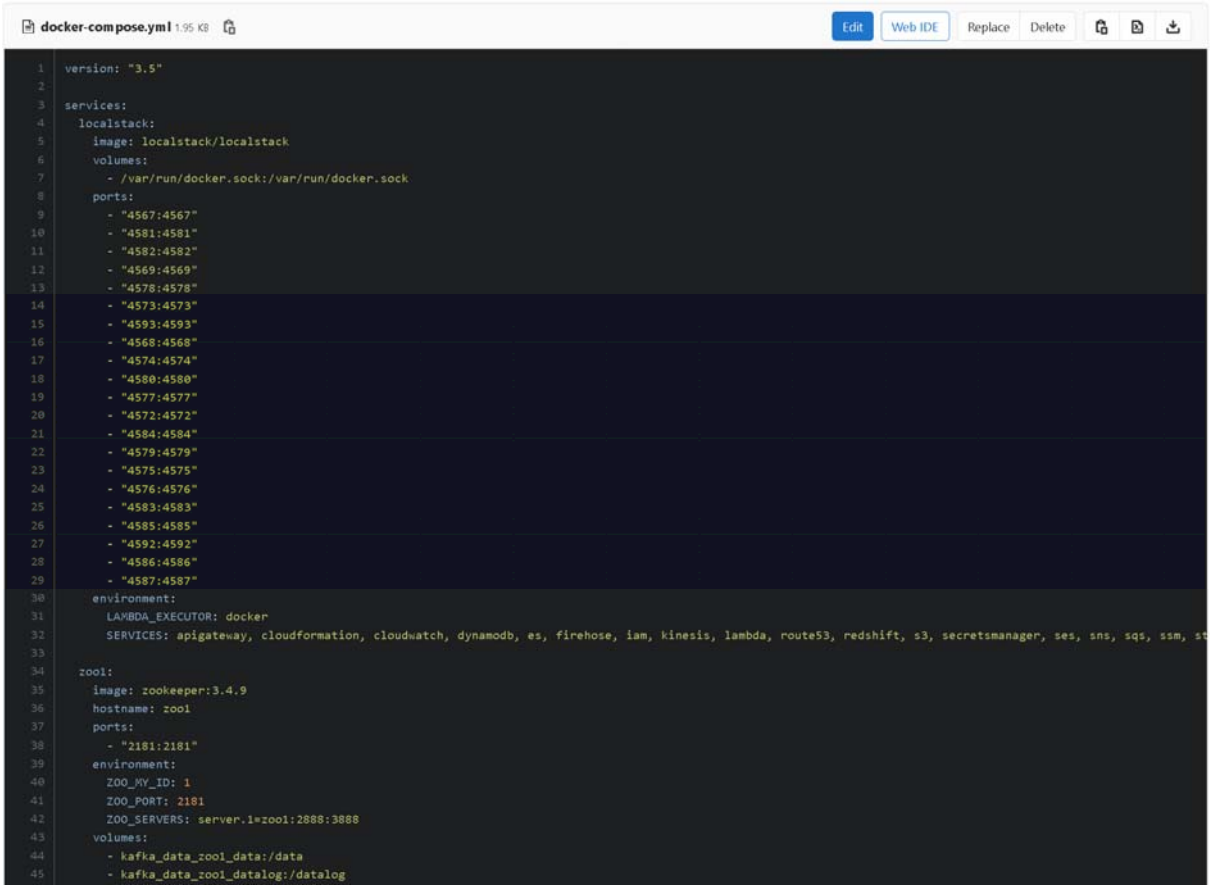
.gitlab-ci.yml 4.79 KB

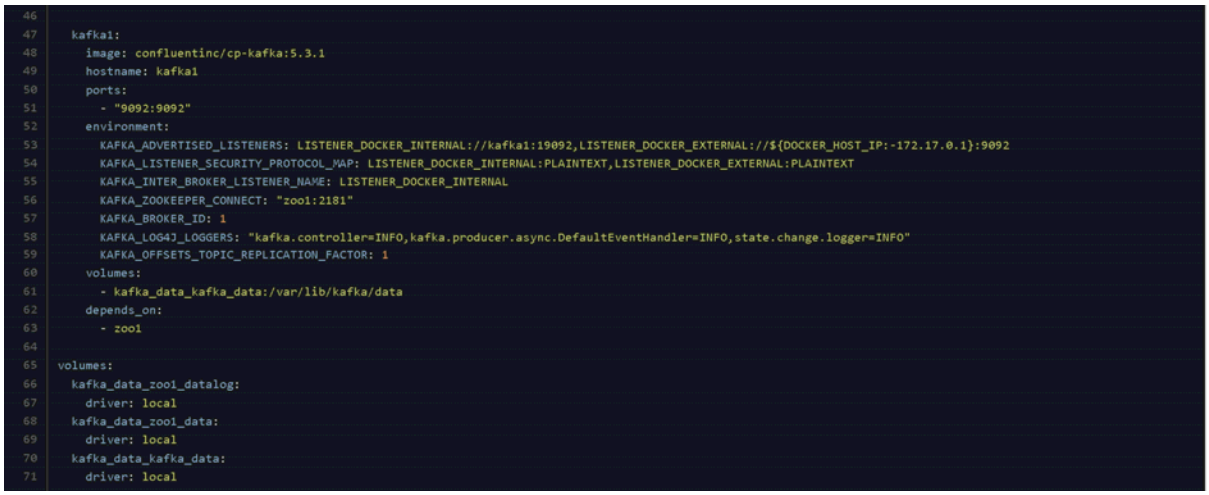
Edit | Web IDE | Replace | Delete

```yaml
image: mcr.microsoft.com/dotnet/core/sdk:3.1

stages:
  - 'Build Solution'
  - 'Deploy'
  - 'Cleanup'
  - 'Run Integration Tests'

build-solution:
  stage: 'Build Solution'
  script:
    - dotnet build

integration-tests:
  stage: 'Run Integration Tests'
  script:
    - dotnet test Tests/Moore.IntegrationTest -c Release
  when: manual

build-image-dev:
  extends: build-image
  variables:
    ENVIRONMENT: dev
    ReleaseEnv: ""
    roleId: "894824635875"
    role: arn:aws:iam::${roleId}:role/moore-ci-${ENVIRONMENT}
  allow_failure: true

build-image-prd:
  extends: build-image
  variables:
    ENVIRONMENT: prd
    ReleaseEnv: "Prod"
    roleId: "622664898613"
    role: arn:aws:iam::${roleId}:role/moore-ci-${ENVIRONMENT}
  allow_failure: true
  only:
    - /^hotfix\/+$/
    - /^release\/+$/

build-image:
  stage: 'Deploy'
  variables:
    AWS_CLI_DEFAULT_REGION: eu-west-3
    rolename: moore-ci-${ENVIRONMENT}
  script:
    - echo ENVIRONMENT = $ENVIRONMENT
    - echo ReleaseEnv =  $ReleaseEnv
    - echo roleId =  $roleId
    - echo role =  $role
    - export PATH="$PATH:/root/.dotnet/tools"
    - dotnet tool install -g Amazon.Lambda.Tools
    - apt-get update
    - apt-get -y install sudo
    - curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
    - sudo apt-get install unzip
    - sudo apt-get install zip
    - unzip awscliv2.zip
    - sudo ./aws/install
    - KST=(`aws sts assume-role --role-arn ${role} --role-session-name ${rolename} --query '[Credentials.AccessKeyId,Credentials.SecretAccessKey,Credentials.Sess
```

```
    - unset AWS_SECURITY_TOKEN
    - export AWS_DEFAULT_REGION=${AWS_CLI_DEFAULT_REGION}
    - export AWS_ACCESS_KEY_ID=${KST[0]}
    - export AWS_SECRET_ACCESS_KEY=${KST[1]}
    - export AWS_SESSION_TOKEN=${KST[2]}
    - export AWS_SECURITY_TOKEN=${KST[2]}

    - dotnet test Tms/Configurations/Tms.Configs -c Release${ReleaseEnv}

    - dotnet-lambda package Wtf.Alb.Proxy.zip -pl Wtf/LambdaFunctions/Wtf.Alb.Proxy -c Release${ReleaseEnv}
    - dotnet-lambda package Wtf.Kafka.Proxy.zip -pl Wtf/LambdaFunctions/Wtf.Kafka.Proxy -c Release${ReleaseEnv}
    - dotnet-lambda package Wtf.Wms.Receive.zip -pl Wtf/LambdaFunctions/Wtf.Wms.Receive -c Release${ReleaseEnv}
    - dotnet-lambda package Wtf.Wosa.Send.zip -pl Wtf/LambdaFunctions/Wtf.Wosa.Send -c Release${ReleaseEnv}
    - dotnet-lambda package Wtf.GenTrackId.Label.zip -pl Wtf/LambdaFunctions/Wtf.GenTrackIdLabel -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.Kafka.Proxy.zip -pl Tms/LambdaFunctions/Tms.Kafka.Proxy -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.Wosa.Receive.zip -pl Tms/LambdaFunctions/Tms.Wosa.Receive -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.Wosa.Transform.zip -pl Tms/LambdaFunctions/Tms.Wosa.Transform -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.Ctt.Send.zip -pl Tms/LambdaFunctions/Tms.Ctt.Send -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.Wosa.Send.zip -pl Tms/LambdaFunctions/Tms.Wosa.Send -c Release${ReleaseEnv}
    - dotnet-lambda package Tms.GenTrackId.Label.zip -pl Tms/LambdaFunctions/Tms.GenTrackIdLabel -c Release${ReleaseEnv}

    - aws lambda update-function-code --function-name moore-wtf-alb-receive-${ENVIRONMENT} --zip-file fileb://Wtf.Alb.Proxy.zip
    - aws lambda update-function-code --function-name moore-wtf-kafka-receive-${ENVIRONMENT} --zip-file fileb://Wtf.Kafka.Proxy.zip
    - aws lambda update-function-code --function-name moore-wtf-wms-receive-${ENVIRONMENT} --zip-file fileb://Wtf.Wms.Receive.zip
    - aws lambda update-function-code --function-name moore-wtf-wosa-send-${ENVIRONMENT} --zip-file fileb://Wtf.Wosa.Send.zip
    - aws lambda update-function-code --function-name moore-wtf-gentrackidlabel-${ENVIRONMENT} --zip-file fileb://Wtf.GenTrackId.Label.zip
    - aws lambda update-function-code --function-name moore-tms-kafka-receive-${ENVIRONMENT} --zip-file fileb://Tms.Kafka.Proxy.zip
    - aws lambda update-function-code --function-name moore-tms-wosa-receive-${ENVIRONMENT} --zip-file fileb://Tms.Wosa.Receive.zip
    - aws lambda update-function-code --function-name moore-tms-wosa-transform-${ENVIRONMENT} --zip-file fileb://Tms.Wosa.Transform.zip
    - aws lambda update-function-code --function-name moore-tms-ctt-send-${ENVIRONMENT} --zip-file fileb://Tms.Ctt.Send.zip
    - aws lambda update-function-code --function-name moore-tms-wosa-send-${ENVIRONMENT} --zip-file fileb://Tms.Wosa.Send.zip
    - aws lambda update-function-code --function-name moore-tms-gentrackidlabel-${ENVIRONMENT} --zip-file fileb://Tms.GenTrackId.Label.zip
  when: manual
  # artifacts:
  #   paths:
  #     - Wtf.Alb.Proxy.zip
  #   expire_in: 1 week

clean:
  stage: 'Cleanup'
  script:
    - dotnet clean --configuration Debug
    - dotnet clean --configuration Release
    - dotnet clean --configuration ReleaseProd
  when:
    manual
```

Annex 2 – gitlab-cy yaml





```
  kafka1:
    image: confluentinc/cp-kafka:5.3.1
    hostname: kafka1
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka1:19092,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-172.17.0.1}:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
      KAFKA_ZOOKEEPER_CONNECT: "zoo1:2181"
      KAFKA_BROKER_ID: 1
      KAFKA_LOG4J_LOGGERS: "kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    volumes:
      - kafka_data_kafka_data:/var/lib/kafka/data
    depends_on:
      - zoo1

volumes:
  kafka_data_zoo1_datalog:
    driver: local
  kafka_data_zoo1_data:
    driver: local
  kafka_data_kafka_data:
    driver: local
```

Annex 3 – docker-compose yaml

Shipconfirm Standard Structure (JSON)	CTT Structure (XML)
Header TMS Parameter table -> Carrier.Ctt.ShipmentParameters	
N/A	<!-- header (***Start***) --> <!ELEMENT UNB (SNDID, RCVID, DTM, INTREF)> ***<UNB>***
IssuerId	<!ELEMENT SNDID (#PCDATA)> ***<SNDID>1CJ241</SNDID>*** **Note:** This Issuer Id will be kept in table moore-tmt-configurations-*
ReceiverId	<!ELEMENT RCVID (#PCDATA)> ***<RCVID>POSTLOG</RCVID>*** **Note:** This Receiver Id will be kept in table moore-tms-configurations-*
N/A	<!ELEMENT DTM (#PCDATA)> ***<DTM>20200430125837</DTM>*** **Nota:** It is a SYSDATE.
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId	<!ELEMENT INTREF (#PCDATA)> This Tag has the Shipment Id from WMS truncated to 6 digits:***<INTREF>085389</INTREF>***
N/A	<!-- header (***End***) --> <!ELEMENT UNB (SNDID, RCVID, DTM, INTREF)> ***</UNB>***
N/A	<!-- message (***Start***) --> <!ELEMENT UNH (MESREF, VRS, RLS)> ***<UNH>***
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId	<!ELEMENT MESREF (#PCDATA)> This Tag has the Shipment Id from WMS truncated to 6 digits:***<MESREF>085389</MESREF>***
MessageVersion	<!ELEMENT VRS (#PCDATA)> ***<VRS>3</VRS>*** **Note:** This Id will be kept in moore-tms-configurations-*

ReleaseId	<!ELEMENT RLS (#PCDATA)> ***<RLS>0</RLS>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!—message (***End***) --> <!ELEMENT UNH (MESREF, VRS, RLS)> ***</UNH>***
AcceptanceGuideId	<!-- Acceptance Guide (***Start***) --> <!-- 1: Contratual --> <!-- 2: Local --> <!-- 3: Ocasional --> <!ELEMENT GIA (LOC+, NGI, PNA, PAI, EMS+, CNT)> <!ATTLIST GIA TYPE (1|2|3) #REQUIRED> <!-- Local / Nó --> <!-- 1: Estação de Correios --> <!-- 2: Centro de Distribuição Postal --> <!-- 3: Centro Operacional --> <!-- 4: Cliente --> <!-- 5: Nó Origem --> <!-- 6: Nó Destino --> <!-- 7: Nó Tratamento --> <!-- 8: Nó Encaminhamento --> <!-- 9: Nó Sistema Externo --> ***<GIA TYPE="1">*** **Note:** This Id will be kept in moore-tms-configurations-*
LocationId	<!ELEMENT LOC (DESC, IDLOC, Value)> <!ATTLIST LOC TYPE (1|2|3|4|5|6|7|8|9) #REQUIRED> ***<LOC TYPE="4">72101</LOC>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!—Acceptance Guide number (***Start***) --> <!ELEMENT NGI (CLNT, CONT, NIF, PGI, NRG, DTM, TPADQ, REGIVA)> ***<NGI>***
ClientId	<!ELEMENT CLNT (#PCDATA)> ***<CLNT>200040186</CLNT>*** **Note:** This Client Id will be kept in moore-tms-configurations-*

ContractId	<!ELEMENT CONT (#PCDATA)> ***<CONT>300301729</CONT>*** **Note:** This Contract Id will be kept in moore-tms-configurations-*
NIF	<!ELEMENT NIF (#PCDATA)> ***<NIF>503630330</NIF>*** **Note:** This Id will be kept in moore-tms-configurations-*
ProductTypeId	<!ELEMENT PGI (#PCDATA)> ***<PGI>99</PGI>*** **Note:** This Id will be kept in moore-tms-configurations-*
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcShipmentId	<!ELEMENT NRG (#PCDATA)> This Tag has the Shipment Id from WMS truncated to 6 digits: ***<NRG>085389</NRG>***
N/A	<!ELEMENT DTM (#PCDATA)> ***<DTM>20200430125837</DTM>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!-- Acceptance Guide (***End***) --> <!ELEMENT NGI (CLNT, CONT, NIF, PGI, NRG, DTM, TPADQ, REGIVA)> ***</NGI>***
PaymentTypeId	<!-- Payment Type (***Start***) --> <!-- 1: Crédito --> <!-- 2: Pronto Pagamento --> <!-- 99: Não se aplica --> <!ELEMENT PAI (CUX, MOA?, CPO?)> <!ATTLIST PAI TYPE (1|2|99) #REQUIRED> ***<PAI TYPE="1">*** **Note:** This Id will be kept in moore-tms-configurations-*
CoinCode	<!ELEMENT CUX (#PCDATA)> ***<CUX>EUR</CUX>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!—Payment Type (***End***) --> <!-- 1: Crédito --> <!-- 2: Pronto Pagamento --> <!-- 99: Não se aplica --> <!ELEMENT PAI (CUX, MOA?, CPO?)> <!ATTLIST PAI TYPE (1|2|99) #REQUIRED> ***</PAI>***

Detail(s) = Shipment Manifest	
N/A	<!--Shipment Manifest(***Start***) --> <!-- 1: Internacional --> <!-- 2: Nacional --> <!ELEMENT EMS (GIN, PIA, PIADESC, PIAASSOC, ITL, QTY, MEA, ZTX, CND?, DPC?, TIN?, PNA*, NAD, TRL?, DHL?, IIO+, SEP*, AGN?, PIN?, REC?, NBACKS?, QTDDA?, INSDDA?, CDAT?, OBSC?, LOCAV?, INRM, LOCAVTP?, DCTP?, REF?, LANG?, HOUSEID?, DTVAL?, CBE?, TPPRD?, TPPRDDESC?, TPEXP?, TPEXPDESC?, OBJBACK?, DTDISTR?, JANDISTR?, INRECEQUIP?, DTPREVENTR?, DESCDETCONT?, EWT?, LOC*, CUSTOMS?, RGJU?, EQUIP*, INDMORDEV, GINEXT?, ENTCJ?, AGRUPCODCLI?, NUMTENT?)> ***<EMS>***
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TrackingNbr	<!ELEMENT GIN (#PCDATA)> Tracking Number: ***<GIN>FA830500015PT</GIN>*** **Note:** Tracking Number generated by the flow Tracking Id & Label
SubProductId	<!ELEMENT PIA (#PCDATA)> ***<PIA>ENCF005.01</PIA>*** **Note:** This Id will be kept in moore-tms-configurations-*
SubProductDescId	<!ELEMENT PIADESC (#PCDATA)> ***<PIADESC>19</PIADESC>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!ELEMENT QTY (#PCDATA)> ***<QTY>1</QTY>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!-- Weight (***Start***) --> <!ELEMENT MEA (WTR?, WTD?, VOL?, WTV?)> ***<MEA>***
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.Weight	<!ELEMENT WTR (#PCDATA)> <!ATTLIST WTR UNIT (GRM) #REQUIRED> ***<WTR UNIT="GRM">12000</WTR>***
N/A	<!-- Weight (***End***) --> <!ELEMENT MEA (WTR?, WTD?, VOL?, WTV?)> ***</MEA>***

TaxZoneId	<!ELEMENT ZTX (#PCDATA)> ***<ZTX>0001</ZTX>*** **Note:** This Id will be kept in moore-tms-configurations-*.
Remetente ("XPTO")	
PartnerType	<!--Partner Type (***Start***) --> <!-- 1: Remetente --> <!-- 2: Destinatario --> <!-- 3: Devolução --> <!ELEMENT PNA (NAD)> <!ATTLIST PNA TYPE (1|2|3) #REQUIRED> ***<PNA TYPE="1">*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!—Name & Address (***Start***) --> <!ELEMENT NAD (NME, ADR, CPL, CTR, TEL?, CTA?, RFF?, TLM?, EMAIL?, ADR2L?, CODDIV?, CTRDESC?, LOCCARGA?)> ***<NAD>***
Name	<!ELEMENT NME (#PCDATA)> ***<NME>"XPTO" </NME>*** **Note:** This Id will be kept in moore-tms-configurations-*
Address	<!ELEMENT ADR (#PCDATA)> ***<ADR>EN 1 </ADR>*** **Note:** This Id will be keept in moore-tms-configurations-*
N/A	<!-- Postal Code (***Start***) --> <!ELEMENT CPL (PTC, CTY)> ***<CPL>***
ZipCode	<!ELEMENT PTC (#PCDATA)> ***<PTC>2050-999</PTC>*** **Note:** This Id will be kept in moore-tms-configurations-*.
City	<!ELEMENT CTY (#PCDATA)> ***<CTY>OTA </CTY>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!-- Postal Code (***End***) --> <!ELEMENT CPL (PTC, CTY)> ***</CPL>***

CountryCode	<!ELEMENT CTR (#PCDATA)> ***<CTR>PT</CTR>*** **Note:** This Id will be kept in moore-tms-configurations-*
Telefone	<!ELEMENT TEL (#PCDATA)> ***<TEL>939457144</TEL>*** **Note:** This Id will be kept in moore-tms-configurations-*
Name	<!ELEMENT CTA (#PCDATA)> ***<CTA>"XPTO" </CTA>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!ELEMENT TLM (#PCDATA)> ***<TLM />***
N/A	<!ELEMENT EMAIL (#PCDATA)> ***<EMAIL />***
City	<!ELEMENT CTY (#PCDATA)> ***<CTY>OTA </CTY>*** **Note:** This Id will be kept in moore-tms-configurations-*
N/A	<!-- Name & Address (***End***) --> <!ELEMENT NAD (NME, ADR, CPL, CTR, TEL?, CTA?, RFF?, TLM?, EMAIL?, ADR2L?, CODDIV?, CTRDESC?, LOCCARGA?)> ***</NAD>***
N/A	<!-- Partner Type (***End***) --> <!-- 1: Remetente --> <!-- 2: Destinatario --> <!-- 3: Devolução --> <!ELEMENT PNA (NAD)> <!ATTLIST PNA TYPE (1|2|3) #REQUIRED> ***</PNA>***
Client	
PartnerType	<!-- Partner Type (***Start***) --> <!-- 1: Remetente --> <!-- 2: Destinatario --> <!-- 3: Devolução --> <!ELEMENT PNA (NAD)> <!ATTLIST PNA TYPE (1|2|3) #REQUIRED> ***<PNA TYPE="2">*** **Note:** This Id will be kept in moore-tms-configurations-*

N/A	<!—Name & Address (***Start***) --> <!ELEMENT NAD (NME, ADR, CPL, CTR, TEL?, CTA?, RFF?, TLM?, EMAIL?, ADR2L?, CODDIV?, CTRDESC?, LOCCARGA?)> ***<NAD>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestName	<!ELEMENT NME (#PCDATA)> ***<NME>Carla Junqueira</NME>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestAddress1	<!ELEMENT ADR (#PCDATA)> ***<ADR>Rua Quinta da cortegaca n?7 3E</ADR>***
N/A	<!-- Postal Code / City (***Start***) --> <!ELEMENT CPL (PTC, CTY)> ***<CPL>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestPostalCode	<!ELEMENT PTC (#PCDATA)> ***<PTC>2840-046</PTC>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCity	<!ELEMENT CTY (#PCDATA)> ***<CTY>ALDEIA DE PAIO PIRES</CTY>***
N/A	<!-- Postal Code / City (***End***) --> <!ELEMENT CPL (PTC, CTY)> ***</CPL>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCountryCode	<!ELEMENT CTR (#PCDATA)> ***<CTR>PT</CTR>***
N/A	<!ELEMENT TEL (#PCDATA)> ***<TEL />***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestName	<!ELEMENT CTA (#PCDATA)> ***<CTA>Carla Junqueira</CTA>***
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TcLpnId	<!ELEMENT RFF (#PCDATA)> ***<RFF>2560198800287836 92</RFF>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderBillToInfo.BillToPhoneNumber	<!ELEMENT TLM (#PCDATA)> ***<TLM>+351969814111</TLM>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderBillToInfo.BillToEmail	<!ELEMENT EMAIL (#PCDATA)> Esta Tag é o Email do cliente: ***<EMAIL>bbbbbbb@hotmail.com</EMAIL>***
Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestAddress2	<!ELEMENT ADR2L (#PCDATA)> ***<ADR2L />***

Message.ShipConfirm.ShipConfirmDetails.Orders.OrderDestInfo.DestCity	<!ELEMENT CTY (#PCDATA)> ***<CTY>Paio Rios</CTY>***
N/A	<!-- Name & Address (***End***) --> <!ELEMENT NAD (NME, ADR, CPL, CTR, TEL?, CTA?, RFF?, TLM?, EMAIL?, ADR2L?, CODDIV?, CTRDESC?, LOCCARGA?)> ***</NAD>***
N/A	<!-- Partner Type (***End***) --> <!-- 1: Remetente --> <!-- 2: Destinatario --> <!-- 3: Devolução --> <!ELEMENT PNA (NAD)> <!ATTLIST PNA TYPE (1|2|3) #REQUIRED> ***</PNA>***
N/A	<!-- Package (***Start***) --> <!ELEMENT IIO (OID, OIDEXT?, MEA?, INIMP?, BOB?, BOBD?, BOBR?, REFERENCES?)> ***<IIO>***
Message.ShipConfirm.ShipConfirmDetails.Orders.Lpn.TrackingNbr	<!ELEMENT OID (#PCDATA)> ***<OID>FA272500015PT</OID>***
N/A	***<NUM_ORDEM />***
N/A	<!-- Package (***End***) --> <!ELEMENT IIO (OID, OIDEXT?, MEA?, INIMP?, BOB?, BOBD?, BOBR?, REFERENCES?)> ***</IIO>***
N/A	<!-- Special Services (***Start***) --> <!ELEMENT SEP (SEPID, PAI?, SEPDESC)> ***<SEP>***
SpecialServiceCode	<!ELEMENT SEPID (#PCDATA)> ***<SEPID>ENCE030.01.05</SEPID>*** **Note:** This Id will be keept in moore-tms-configurations-*

PaymentType	<!—Payment Type (***Start***) --> <!-- 1: Crédito --> <!-- 2: Pronto Pagamento --> <!-- 99: Não se aplica --> <!ELEMENT PAI (CUX, MOA?, CPO?)> ***<PAI TYPE="99">*** **Note:** This Id will be keept in moore-tms-configurations-*
CoinCode	<!ELEMENT CUX (#PCDATA)> ***<CUX>EUR</CUX>*** **Note:** This Id will be keept in moore-tms-configurations-*
ShippingCosts	<!ELEMENT MOA (#PCDATA)> ***<MOA>0</MOA>*** **Note:** This Id will be keept in moore-tms-configurations-*
N/A	<!ELEMENT CPO (#PCDATA)> ***<CPO />***
N/A	<!—Payment Type (***Start***) --> <!-- 1: Crédito --> <!-- 2: Pronto Pagamento --> <!-- 99: Não se aplica --> <!ELEMENT PAI (CUX, MOA?, CPO?)> ***</PAI>***
N/A	<!-- Special Services (***End***) --> <!ELEMENT SEP (SEPID, PAI?, SEPDESC)> ***</SEP>***
Message.ShipConfirm.ShipConfirmDetails.Orders.StoreNbr	<!ELEMENT CDAT (#PCDATA)> ***<CDAT />*** **Note:** Tax Authority Code.
Message.ShipConfirm.ShipConfirmDetails.Orders.DistroNumber	<!ELEMENT OBSC (#PCDATA)> ***<OBSC>5a0f7544-dade-11ea-982c-b10e0d24e537</OBSC>***
N/A	<!ELEMENT LOCAV (#PCDATA)> ***<LOCAV />***
ShippingId	<!ELEMENT INRM (#PCDATA)> ***<INRM>N</INRM>*** **Note:** This Id will be keept in moore-tms-configurations-*
N/A	<!ELEMENT DCTP (#PCDATA)> ***<DCTP />***

N/A	<!ELEMENT LANG (#PCDATA)> ***<LANG />***
N/A	<!ELEMENT TPPRD (#PCDATA)> ***<TPPRD />***
N/A	<!ELEMENT TPEXP (#PCDATA)> ***<TPEXP />***
N/A	<!-- Shipment Manifest (***End***) --> <!-- 1: Internacional --> <!-- 2: Nacional --> <!ELEMENT EMS (GIN, PIA, PIADESC, PIAASSOC, ITL, QTY, MEA, ZTX, CND?, DPC?, TIN?, PNA*, NAD, TRL?, DHL?, IIO+, SEP*, AGN?, PIN?, REC?, NBACKS?, QTDDA?, INSDDA?, CDAT?, OBSC?, LOCAV?, INRM, LOCAVTP?, DCTP?, REF?, LANG?, HOUSEID?, DTVAL?, CBE?, TPPRD?, TPPRDDESC?, TPEXP?, TPEXPDESC?, OBJBACK?, DTDISTR?, JANDISTR?, INRECEQUIP?, DTPREVENTR?, DESCDETCONT?, EWT?, LOC*, CUSTOMS?, RGJU?, EQUIP*, INDMORDEV, GINEXT?, ENTCJ?, AGRUPCODCLI?, NUMTENT?)> ***</EMS>***
N/A	<!ELEMENT CNT (#PCDATA)> ***<CNT>7</CNT>*** **Nota:** Este ***Numero de Elementos,*** é um contador de elementos (Tags EMS) presentes no Shipconfirm.
N/A	<!-- Acceptance Guide (***End***) --> <!-- 1: Contratual --> <!-- 2: Local --> <!-- 3: Ocasional --> <!ELEMENT GIA (LOC+, NGI, PNA, PAI, EMS+, CNT)> <!ATTLIST GIA TYPE (1|2|3) #REQUIRED> <!-- Local / Nó --> <!-- 1: Estação de Correios --> <!-- 2: Centro de Distribuição Postal --> <!-- 3: Centro Operacional --> <!-- 4: Cliente --> <!-- 5: Nó Origem --> <!-- 6: Nó Destino --> <!-- 7: Nó Tratamento --> <!-- 8: Nó Encaminhamento --> <!-- 9: Nó Sistema Externo --> ***</GIA>***

N/A	<!ELEMENT CNT (#PCDATA)> ***<CNT>1</CNT>*** **Note:** fixed value

Annex 4 – CTT mapping matrix (Source: own)