NEURAL TECHNOLOGIES

Microservices Best Practice

Neural
Technologies

PAPER CONTENTS

INTRODUCTION

Image source: https://www.forbes.com/sites/sap/2017/03/14/digital-transformation-is-hard-but-microservices-are-easy/#2b85ee95c30b

Competition is always the most important differential across all industries. Competition in the Communications Service Providers (CSPs) domain, however, has never been more competitive than it is today. No CSP, regardless of size, can consider themselves immune to the disruptive influences from such fierce competition. To ensure survival, CSPs need to be able to innovate at the same pace, or faster, than their competition to avoid being left behind. What then, will the CSPs need to embrace to become a more agile, and more successful enterprise?

The CSP needs to begin with Digital Transformation & DevOps Culture: the development of smaller and more focused teams rapidly innovating on specific business areas, and then synchronising with other spaces within the CSP, and together delivering a total solution to the larger business problem. Think of it like all the musicians in the orchestra playing in harmony to produce a song. This Digitally Transformed DevOps Culture is the future of all CSPs.

To successfully lay the foundations for this continuous innovation and agility, means that CSPs will have had to embrace and adopt the Microservices architecture. This methodology enables the CSP's rapid-response to the never ending demands of business. In this whitepaper, we will explore the essential opportunities that the Microservices architecture provide.

WHY EMBRACE MICROSERVICES ARCHITECTURES?

Microservices are the result of the evolution of best-practice architectural principles. They shape the delivery of solutions to the business by combining multiple individual services together. All CSPs are constantly striving to deliver the ultimate customer experience within a landscape where customers are more demanding than ever and will leave a business that is too slow to respond. The IT department must therefore deliver solutions that can adapt rapidly to deliver a uniform experience to the customer across all channels, e.g. Web Portal, Physical Stores etc.

To achieve this, the business architecture should identify all of the individual digital assets which directly affect the core business capabilities of the CSP – this process will define the various Microservices for the business. These Microservices can be adapted for use in multiple contexts. The specific context is decided by the business processes and channels that your customer uses to consume your product and interact with your organisation.

Microservices in action look like this: Another customer access point (Web Portal) needs to be added to an existing subscription portal service (the current Mobile App portal). The use of Microservices architecture ensures that the only addition that needs to be made is the Web Portal Development, keeping the rest of the core business capabilities of the Subscription Portal Service intact.

What are the Benefits to the CSPs?

The aligning of the Microservices architecture with your business ensures that any changes or problems can be dealt with in an agile fashion. Business processes are automated by the Microservices architecture. When processes are changed, or new ones introduced, IT can rapidly respond by reorganising the Microservices into new compositions; as if a musician was simply changing a note in their song.

The ability to react to any business requirements will determine whether a CSP can change the 'song', or react to the business problem, quickly and with ease. If the business logic is composed with various Microservices, IT can match the pace of the business and react to business changes agilely. This agility is at the forefront of innovation – some required business changes may be:

1. The need for new revenue streams, where the CSPs would have to expose APIs to external partners
2. The need for new digital engagements with customers, where the CSP would add a Web Portal in addition to the Mobile App, or create entirely new products and services which may demand entirely new business processes

The common theme for the benefits of Microservices architecture is the "Speed of Change". CSPs need the tools to rapidly adopt a change in direction if the market requires it. IT can facilitate these changes by composing existing digital assets in the form of Microservices, into new business capabilities quickly. How do Microservices achieve this?

The IT departments of the CSP can deliver business solution logic in a decentralised manner since Microservices interact through the use of standardised APIs. Multiple teams from different IT business domains can implement Microservices with their own choice of technology while remaining interoperable, since Microservices can natively communicate with each other because of industry-wide adoption of standards such as TM Forum Open APIs. Think of a drummer in the orchestra with his own specific notes, as the Conductor brings the 'Microservice' drum sound into the song as a whole.

Microservices can also simplify the scaling process in the business solution flow for a CSP, since the business solution itself is composed of multiple Microservices. If a Microservice is becoming a bottleneck within a business solution due to slow execution, then that Microservice can be easily moved to a more powerful hardware for increased performance; or IT can run multiple instances of the Microservice across different machines to process data elements in parallel.

When we compare the ease of Microservices scalability with monolithic systems, where scaling is non-trivial, there is only one winner. If a module has a slow internal piece of code, there is no way to make that individual piece of code run faster. To scale a monolithic system, one must run a copy of the complete system on a different machine, but even doing that does not resolve the bottleneck of a slow internal-step within the monolith.

WHAT DOES THE "MICRO" IN A MICROSERVICE STAND FOR?

Traditionally, CSPs have delivered business solutions in siloes, after being developed from the isolated demands of individual departments. Consequentially, software products were then developed or purchased only in regard to these limited scopes. Yet, the customer facing business processes typically span multiple departments and are not isolated to one. The challenge that the traditional method has created over time is a lack of alignment across departments, which in turn has led to duplicated efforts or worse: missing/inaccurate information on various services leading to a poor customer experience.

The concept of Microservices evolved from these issues across business processes – CSPs needed to modify their existing processes or create entirely new ones so that they could respond agilely. Rather than building or purchasing "business solutions" in the traditional sense, you can develop Microservices as the building block of business logic. This logic can then be orchestrated to address any requirements of the business.

Though, what classifies as a Microservice?

The Scope of Responsibility of a "Micro"-Service

The most obvious scope of responsibility of a Microservice is whether the business entity is easily identifiable with a glance at the business process. It is critical that the Scope of Responsibility can be easily carved out so that the Microservices are relevant in many contexts across both the process layer and the experience layer. "Distinct" is the best acronym of "micro" when discussing the scope of responsibility for a Microservice. The Microservices must be agnostic to any business process so that it can be used in multiple business solutions. Ideally, however, a Microservice will be an "atomic" service that you can not dismantle into further Microservices.

The Team behind the development of a "Micro"-Service

When developing a Microservice, the team should focus entirely on a single business entity of the business process, and be sure to include domain experts of that entity so that they can effectively market the Microservice.

Moreover, the team ownership and life-cylce management of the Microservice will include everything from design to development to roadmap. They will need to add any APIs as required for the Microservice. This should all be completed by one team, with no handoff to other teams.

Effort needed to develop the "Micro"-Service

Microservices that address the needs of specific business entities will recognise that certain business concepts should only exist within the relevant business entity, for example a Rating Engine. Microservices should not be developed to model a universal solution unless the business entity it serves is naturally shared across the entire organisation.

How cost-effective can the "Micro"-Service be deployed

The costs of development and support for the individual Microservice can be reduced by the combination of smaller teams with complete ownership and life-cycle responsibility of that Microservice.

MICROSERVICES VS. SINGLE MONOLITHIC BUSINESS SOLUTION DEVELOPMENT

Image source: https://www.coscale.com/blog/comparing-microservices-and-monolithic-applications-from-the-perspective-of-monitoring

Microservices mark a fundamental shift for how IT will approach business solution development in the future. The traditional approach for business solution development has been to assign large teams to work on a single, monolithic application. Project managers, developers and operational staff of such monolithic application development could build (with varying degrees of success) an implementable business solution. There are, however, some clearing issues with the traditional approach:

- Limited to no re-use of monolithic business solutions. Monolithic business solutions, by definition, hide their internals. Whilst some reusability might be realised by API's at the "edge" of the monolith, the chance for the re-use of internal components is limited. When re-use is achieved it's usually through shared libraries which foster tight coupling.
- Monolithic business solutions can regularly become unmanageable as the size and complexity of the business solution grows. This creates issues where no single developer understands the entirety of the business solution.
- Scaling monolithic business solutions can be challenging: identifying and tuning specific aspects of business solution functionality for performance in isolation of other aspects is usually impossible since all functionality is bundled and cannot be split across multiple servers

A Microservices architecture, in combination with industry standard based APIs, provides an approach to development that avoids the delivery of monolithic business solutions. The monolith is instead "broken up" into a set of independent Microservices that are developed, deployed and maintained separately. This has the following advantages:

- Microservices are encouraged to be small, ideally built by a handful, or even a single developer, so that there is a full understanding of a single Microservice.
- Microservices can exist as independent section of a business solution, ready to be scaled independently of any other Microservices within the same business solution.
- As Microservices expose their interfaces through standard based APIs, they can be consumed and re-used by other services and business solutions.

However, managing distributed Microservices at scale does come with its own challenges that need to be carefully taken into consideration:

However, managing distributed Microservices at scale does come with its own challenges that need to be carefully taken into consideration:

- When developing Microservices, the individual development teams need to make sure they are re-usable as much as possible. These re-usable Microservices should provide clear documentation, test consoles, etc. so that re-using is an easy option, rather than building from scratch and costing the organisation further time and resources.
- The close dependencies between each of the Microservices needs to be well understood when orchestrating them together to form a total business solution. The impact to the total business solution if any of the Microservices have outages or require upgrades could have a cascading effect to the detriment of the organisation; therefore they need to be closely monitored and analysed.

MICROSERVICE DESIGN – THE CORE PRINCIPLES

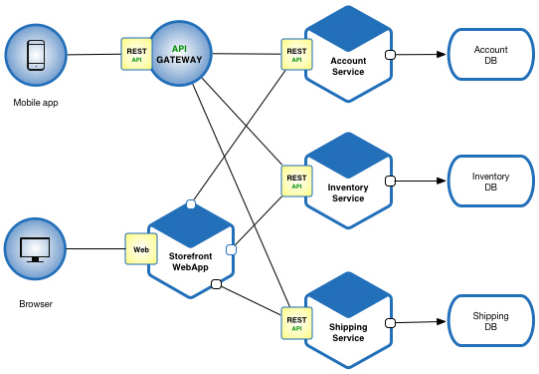


Image source: http://microservices.io/patterns/microservices.html

Microservice capabilities are exposed though a set of well-defined standard based APIs. They encapsulate a core business entity and as such are assets to the business. The implementation of the Microservice, which may involve integrations with various data lakes, is completely hidden since the exposed standard based APIs are purely defined in business terms.

Loose Coupling

Moreover, the dependencies between Microservices and their consumers are minimised by applying the principle of loose coupling. This is achieved by only exposing the standard based APIs, so that any changes in the implementation of the Microservice will be completely transparent. This allows the Microservice owner to change the implementation of that Microservice without any upstream impact, for example improving performance or changing to data source. This, however, assumes the use of an underlying standardised data model that is understandable by any services for the definition of the payload.

Autonomy

Autonomy is an important factor when developing the Microservice. Autonomy is the measure of control that the Microservice has over its runtime environment and database schema during implementation. This enhances the performance and reliability of the Microservice and ensures a higher quality of service. Coupled with statelessness, autonomy also contributes to the overall availability and scalability of the Microservice. Each Microservice has its own inbuilt fault tolerance so that it will have minimal impact on its own Service Level Agreement (SLA) when collaborating with other Microservices. By virtue of being independent of each other, Microservices have the opportunity to cut off communication to a failed Microservice that it is collaborating with. This technique stops individual Microservice failures from propagating through the larger business solution.

Data Lakes vs. Virtualised Data Lakes

The autonomy principle of Microservice design means that Microservices are often built with their own independent data lake. This could potentially create problems depending on the type of data records the individual Microservice needs to store and/or consume to operate.

A solution is to combine these individual data lakes together in a virtual plain, so that all Microservices always have access to a common set of data records by defining a standardised data model. This is achieved by extracting the individual data lakes' data structure into a common metadata dictionary, and then converting the individual data lake query language into a unified query language, all independent of the individual data lakes' technical setup. This will also ease the development of Microservices as they only need to talk one database query language and the choice of database (e.g. Hadoop vs. ODBC) can still be chosen as per the need of the Microservice but hidden from the development of it. Instead, the matching of database technology to satisfy the need of the Microservice is configured in the metadata dictionary.

Event-Driven Architecture

Alternatively, it is possible to use an event-driven architecture. In this architecture a Microservice publishes an event when something notable happens. Other Microservices subscribe to those events and when a Microservice receives an event it can update accordingly, which might lead to more events being published.

You can use events to implement business transactions that span multiple solutions across the Lines of Business (LOBs). A transaction consists of a series of steps, and each step consists of a Microservice updating a business entity and publishing an event that triggers the next step.

Provided that each service automatically updates the database and publishes these events at least once, you can implement business transactions that span multiple services. It is important to note that these are not ACID (Atomicity, Consistency, Isolation and Durability) transactions. They offer much weaker guarantees, such as eventual consistency. This transaction model has been referred to as the BASE (Basically Available, Soft state, Eventual consistency) model.

You can also use events to maintain materialised views that pre-join data owned by multiple Microservices. The service that maintains the view subscribes to the relevant events and updates the view. For example, the Customer Order View Updater Service that maintains a Customer Orders view, could subscribe to the events published by the Customer Service and Order Service.

In a Microservices architecture, a service can subscribe to event notifications published by other services as triggers for action.

An event-driven Microservice architecture has several benefits:

- It enables the implementation of transactions that span multiple services and provide eventual consistency.
- It enables an application to maintain materialised views.

An event-driven Microservice architecture also has several drawbacks:

- The programming model is more complex than when using ACID transactions. Often you must implement compensating transactions to recover from application-level failures; for example, you must cancel an order if the credit check fails. Also, applications must deal with inconsistent data because of changes made by in-flight transactions that are visible.
- Subscribers must detect and ignore duplicate events.

Self-Service Consumption of Microservices through Standardised APIs

Every business entity of a CSP needs technology to build new business solutions for their specific function/customer. Therefore, the IT of a CSP needs to digital transform itself from the traditional function of the sole technology provider, into an organisation that can keep up with the pace of the digital era. This transformation can only occur, however, if IT transforms itself into a business enabler rather than a centralised technology function. Let's explore this requirement.

Being a business enabler means that IT has to decentralise business solution development though the use of Microservices, including data lake access to the different CSPs' Lines of Business (LoBs) and external business partners. This way, IT can concentrate on a strategy business partnership with the business rather than remaining purely technical.

Service proliferation, however, is a trade-off incurred by this approach. Managing these Microservices at scale raises a number of challenges:

- Service Discovery and Documentation
- Fault tolerance
- Quality of Service
- Security and Privacy
- Request traceability
- Failure triage

It is therefore imperative that you manage your Microservices in a fashion that will facilitate self-service access across all the lines of business. This can be achieved by exposing a common set of Standardised APIs per Microservice that will allow the CSP to do the following:

- Publish your Standardised APIs so that the consuming party has everything they need to self- serve including a well-defined purpose and scope of your Microservice.
- Adapt your Standardised APIs with Policies of Governance, covering everything from Security, Auditing, to Dynamic Data Filtering and more
- Monitor your Standardised APIs so that you ensure scalability according to traffic levels and over-usage patterns by the business.
- Tailor your Standardised APIs to the specific needs of the different LOBs so that API management becomes a decentralised/federated exercise in collaboration. The adoption of TM Forum Open APIs would be highly recommended for this purpose.

The Dangers of New

New architectural trends always pose a danger as they are often seen as silver bullets for IT's problems and are often introduced without consideration for all related dependencies such as infrastructure, processes and developer skillsets.

Adopting a Microservices approach to solution development should be a carefully measured approach to ensure that the benefits can be maximised. It is therefore imperative when designing & developing Microservices that they only encapsulate business entity capabilities for each of the business domains. The risk of not doing this, is that you will end up building a Microservice "look alike" that is actually a monolith business solution containing all the downfalls that come with it.

It is also highly recommended that you put in place a strict DevOps-style team coordination and automation as much as possible otherwise your Microservices initiative could bring more pain than benefits. Additionally, the orchestration of those Microservices and the automation required for digitalisation is also mandated to ensure that you can collect statistics and apply Artificial Intelligence & Machine Learning for both maintenance and efficiency improvements.

All of these design principles contribute to the principle of orchestration which allows the Microservices to deliver value to the business in different contexts. By orchestrating you can combine a Microservice with other Microservices to form an aggregate, and this effectively becomes the new form of business solution development. The aim of discoverability is to communicate to all interested parties a clear understanding of the business purpose and the standard based APIs of the individual Microservice so that it can easily be consumed when needed. Thus, the Microservice must be published in a way that ensures that developers of total business solutions have everything they need to easily understand how to consume it.

EXAMPLES OF MICROSERVICES IN THE WORLD TODAY

Image source: https://medium.com/refraction-tech-everything/how-netflix-works-the-hugely-simplified-complex-stuff-that-happens-every-time-you-hit-play-3a40c9be254b

Netflix - Hundreds of Microservices, one giant service

Several companies have adopted the Microservice architecture and in this section, I will highlight a few of these examples and the benefits they have gained as a consequence.

Netflix ushered in a technological revolution around ten years ago by rewriting the applications that run the entire service to fit into a Microservices architecture — which means that each application, (or Microservice's code and resources) are its very own. It will not share any of its code/resources with any other app by nature. When two applications do need to talk to each other, they use an application programming interface (API) — a tightly-controlled set of rules that both programs can handle. Developers can now make many changes, small or huge, to each application as long as they ensure that it interacts with the API. Moreover, since the one program knows the other's API properly, no change will break the exchange of information.

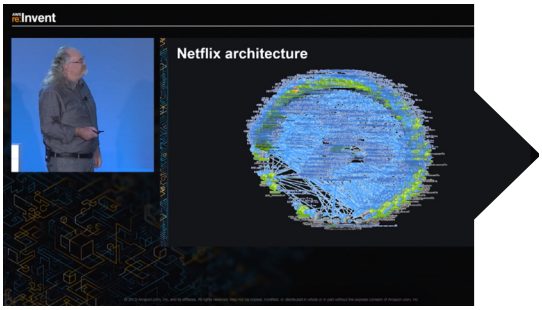


Image source: https://medium.com/refraction-tech-everything/how-netflix-works-the-hugely-simplified-complex-stuff-that-happens-every-time-you-hit-play-3a40c9be254b

Netflix estimates that it uses around 700 Microservices to control each of the many moving parts that make up the entire Netflix service: one Microservice stores what shows you have watched, one deducts the monthly fee from your credit card, one provides your device with the correct video files that it can play, one takes a look at your watching history and uses algorithms to guess a list of movies that you will like, and one will provide the names and images of these movies to be shown in a list on the main menu. And that is just the tip of the iceberg. Netflix engineers can make changes to any part of the application and can introduce new changes rapidly while ensuring that nothing else in the entire service breaks down.

So, in a nutshell, what happens when you press the Netflix play Button:

- Hundreds of Microservices (i.e. tiny independent programs) work together to make one large Netflix service.
- Content legally acquired or licensed is converted into a size that fits your screen, and is protected from being copied.
- Servers across the world make a copy of it and store it so that the closest one to your device can be delivered at maximum quality and speed.
- When you select a show, your Netflix app cherry picks which of these servers will it load the video from.
- You are now gripped by Frank Underwood's chilling tactics, given depression by BoJack Horseman's rollercoaster life, tickled by Dev in Master of None and made phobic to the future of technology by the stories in Black Mirror.

Walmart successfully revitalised its failing architecture with Microservices

Image source: http://www.techiexpert.com/five-ways-walmart-uses-big-data/

Content Reference: https://apiumhub.com/tech-blog-barcelona/microservices-architecture-implementation/

This is a good example of what should be done when aging architecture begins to negatively affect a business. This is the multi-million dollar question which the IT Department of Walmart Canada had to address after they were failing on Black Fridays for two years in a row.

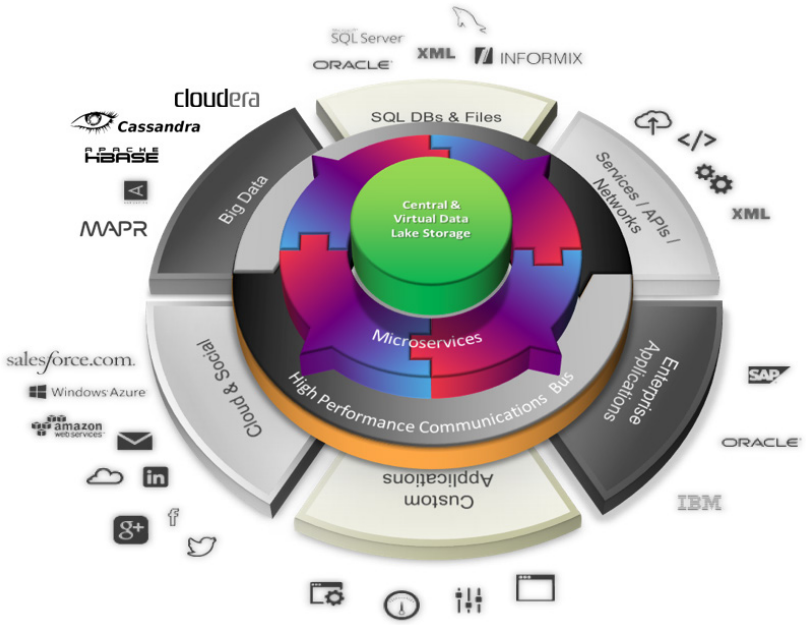
The problem was that IT couldn't handle 6 million page views per minute and this made it impossible for them to maintain any kind of remotely positive user experience. Before embracing Microservices, Walmart had an architecture for the internet of 2005, designed around desktops, laptops and monoliths. The company decided to re-platform its old legacy system in 2012 since it was unable to scale for 6 million page views per minute and was actually down for most of the day during peak events. They wanted to prepare for the world by 2020, with 4 billion people connected, 25+ million apps available. So, Walmart re-platformed to a Microservices architecture with the intention of achieving close to 100% availability with reasonable costs.

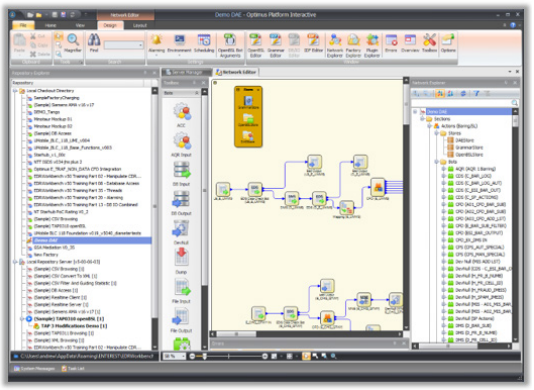
Migrating to Microservices actually brought immediate results:

- Conversions were up by 20% overnight
- Mobile orders were up by 98% instantly
- No downtime on Black Friday or Boxing Day
- The operational savings were significant since the company moved off of its expensive hardware onto commodity hardware
- They saved 40% of the computing power and experienced 20-50% cost savings overall

WHAT NEURAL TECHNOLOGIES OFFERS

Neural Technologies have been using the Microservice architecture as part of its Optimus Platform business solution development product since 2003.





Initially, when Neural Technologies launched the first version of its Optimus Platform, the term Microservices had not been adopted by the industry. Neural Technologies developers made up their own term and named their Microservice offerings as BotPacs (Robot Packages). Today Neural Technologies has an extensive list of Microservices (BotPacs) that their various customers integrate directly into the multiple purposely built business solutions that have been developed on top of the Optimus Platform by utilising the Dynamic Workflow Orchestration tool, Optimus Interactive, that comes together with the Optimus Platform.

Optimus Interactive not only orchestrates the Microservices into a total business solution offering, it also enables the solution developers to modify such Microservices on the fly to suit their own specific business requirements through its purpose built programming language, openBSL (open Bot Scripting Language), as well as a full featured compiler, debugger and simulator to ensure that the business solution developed directly meets the needs to the business.

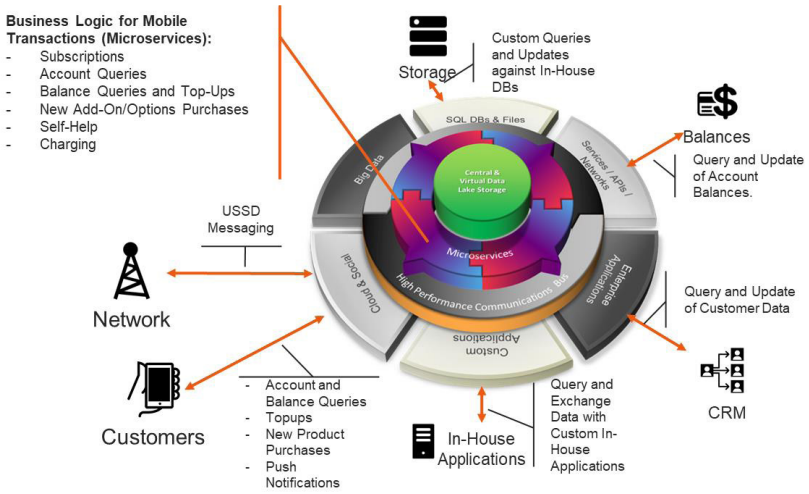
Putting Microservices into Practice – A Real-Life Example

In this section we will look at one of Neural Technologies customers who had identified the need to offer a more consistent customer experience as part of its digital transformation strategy.

The digital transformation requirements identified as part of the success criteria for the project were:

- Multi-Touch points seamless integration (Frontend, Backend)
- Enhanced customer experience through customised workflows per customer
- Centralised business logic
- Fast time-to-market
- Commercial open sourced and flexible to allow quick business changes
- High Scalability, Availability, Maintainability

What was delivered to address the business requirements:



Technical Deliverables:

- Orchestration of transactions and all involved components for End2End full transactional security
- Custom Configured Business Logic
- Extensible without Product Modification

Results:

- Connected to
  - 11 Frontend touch points
  - 7 Backend platforms
  - 120 Functional APIs
- Enabling more than 20 services
- 10 million transactions per day

Timeline:

- ~ 6 Months

CONCLUSION

Microservices are clearly an important and welcoming trend in the software development industry and have many advantages over previous architectural approaches. There are, however, various concerns to be aware of when instituting a Microservices architecture. CSPs need to implement Microservices because of the ease of deployment and agile nature, but if it is not managed and implemented properly, this architecture can create disorganisation and lack of governance. Therefore, instituting Microservices in a way that will create competitive advantage and help the CSPs innovate faster goes beyond a mere selection of product partner. You must also consider the people, process, and culture within the organisation.

Neural Technologies recommends a holistic, platform approach to Microservices, cantered around API-led connectivity. Not only does API-led connectivity create the integration component so crucial to the proper function of your technology stack, but it will also allow developers inside and outside the central IT team to create new solutions in a manageable, re-usable, and governed fashion thus eliminating the concerns of too many business solutions that the CSP can not control. In addition, Neural Technologies Optimus Platform approach provides a unique ability to allow the individual LOBs (in coordination with IT) to build, innovate, and deliver new solutions via dynamic workflow orchestration of and development of Microservices though its innovative Optimus Interactive graphical user interface to address any business needed as they arise throughout the CSP.

In today's hyper-competitive business environment, it is crucial to stand out and provide a customised and unique experience for customers and partners. Microservices are the way for a CSP to achieve this. If done in a holistic, manageable fashion, the Microservices architecture will become the de facto solution development standard for CSPs going forward.

Contact us at: info@neuralt.com to learn more.