**THAIANE ABREU SANTOS**

# MIGRAÇÃO DE SISTEMAS LEGADOS UTILIZANDO A *GOOGLE CLOUD PLATFORM*

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2019

**THAIANE ABREU SANTOS**

# MIGRAÇÃO DE SISTEMAS LEGADOS UTILIZANDO A *GOOGLE CLOUD PLATFORM*

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr. Gabriela Cabel Barbarán

São Paulo
2019

Catalogação-na-publicação

Santos, Thaiane
Migração De Sistemas Legados Para Nuvem Utilizando a Google Cloud Platform / T. Santos -- São Paulo, 2019.
50 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1. Ciência da Computação 2. Engenharia de Software 3. Reengenharia de Software I. Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

## DEDICATÓRIA

*Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida e a minha família que sempre apoiou e incentivou meu crescimento pessoal e profissional.*

## AGRADECIMENTOS

Ao PECE – Programa de Educação Continuada em Engenharia que através do curso de pós-graduação em Tecnologia de Software proporcionou ampliar o meu conhecimento em Engenharia de Software.

À minha família que sempre apoiou e incentivou o meu crescimento profissional e pessoal compreendendo com paciência a minha ausência dedicada aos estudos.

À professora orientadora Gabriela Cabel Barbarán que pacientemente me orientou na elaboração desse trabalho.

## RESUMO

Migração de sistemas legados para cloud implica na adaptação desses sistemas para diferentes paradigmas computacionais, bem como a alteração dos padrões de arquitetura. Com o interesse das grandes empresas na diminuição de custos de manutenção de sistemas legados e a consolidação de provedores de serviços de nuvem no mercado, comprovando maior flexibilidade e escalabilidade de sistemas, esse trabalho teve como objetivo a elaboração de um roteiro de migração de sistemas legados para a nuvem baseados na estratégia de reengenharia para SaaS. Para isso, foi feito um levantamento bibliográfico sobre computação em nuvem, sistemas legados, *domain driven design*, arquitetura baseada microserviços, *Google Cloud Platform* e estratégias de migração, a partir desse levantamento, foi criado e proposto um roteiro utilizando as ferramentas da *Google Cloud Platform* no qual foi aplicado num cenário real. Após a análise de cada passo do roteiro foi identificado os ganhos em diminuição de complexidade e a melhoria na qualidade da migração, bem como as dificuldades envolvidas no processo.



**Palavras-chave:** Migração de sistemas. Nuvem. *Google Cloud Platform*. Reengenharia *SaaS*.

## ABSTRACT

Migration of legacy systems to the cloud implies the adaptation of these systems to different computing paradigms, as well as the alteration of architectural patterns. With the interest of large companies in reducing maintenance costs of legacy systems and the consolidation of cloud service providers in the market, proving more flexibility and scalability of systems, this work had the objective of elaborating a script of migration of legacy systems to the cloud based on SaaS reengineering strategy. For this, a bibliographic survey was made on cloud computing, legacy systems, domain driven design, microservice based architecture, Google Cloud Platform and migration strategies. From this survey, a script was created and proposed using Google Cloud Platform tools in which it was applied in a real scenario. After analyzing each stage of the roadmap, the gains were identified in reducing complexity and improving the quality of migration, as well as the difficulties involved in the process.



**Keywords:** Systems migration. Cloud. Google Cloud Platform. SaaS Reengineering.

## LISTA DE ILUSTRAÇÕES

Pág.

# SUMÁRIO

Pág.

# 1. INTRODUÇÃO

Todas empresas praticamente dependem de sistemas de informação para suportar seu negócio. O que acontece é que a maioria desses sistemas foram construídos ao longo dos anos com tecnologias que hoje estão ultrapassadas, difíceis de serem alterados e atualizados, esses sistemas são considerados legados.
Atualmente as empresas sentem a necessidade de estruturar-se para atenderem as mudanças do mercado, relacionadas as pessoas, processos e novas tecnologias, o que torna os sistemas legados difíceis de se encaixar nesse cenário de constante mudanças.

## 1.1. Motivações

Visto as mudanças exponenciais que estamos vivenciando em relação ao uso da tecnologia no nosso dia-a-dia, a transformação digital é uma realidade que traz a necessidade de flexibilidade e agilidade das empresas em se adaptar as mudanças de forma rápida e simples, porém a maioria dos sistemas utilizados atualmente são legados, o que dificulta essa adaptação, uma vez que podem ser complexos, tecnologicamente obsoletos, entre outros fatores.
A motivação no presente trabalho é apresentar um roteiro de migração que sirva como um guia aos interessados pelo tema de migração de sistemas legados para nuvem, utilizando os recursos da *Google Cloud Platform*. Ajudando assim, os interessados pelo tema, a identificar se possuem o conhecimento necessário para realizar a migração de um sistema, quando ela é vantajosa, bem como o que é necessário conhecer e utilizar antes de entrar na jornada de migração.
Além disso, o autor deste trabalho está vivenciando esse processo de transformação digital e migração dos sistemas legados, o que será enriquecedor para os projetos em que ele atua.

## 1.2. Objetivo

Este trabalho apresentará um roteiro de migração de sistemas legados com base na estratégia de reengenharia para *SaaS* utilizando os recursos da *Google Cloud*

*Platform*, os pré-requisitos para migração, bem como os ganhos e dificuldades envolvidos, afim de guiar os primeiros passos de quem deseja entrar nessa jornada de transformação digital através da migração de sistemas legados para nuvem. Esse roteiro foi elaborado em consequência de um projeto piloto para evolução de sistemas legados.

Sendo assim, serão apresentados os assuntos de computação em nuvem, sistemas legados, paradigmas de programação, *domain driven design*, arquitetura baseada em microserviços, *Google Cloud Platform,* estratégias de migração de sistemas, e um roteiro migração utilizando os recursos da *Google Cloud Platform*.

## 1.3. Justificativas

Sistemas legados realizam tarefas úteis para a organização, mas foram desenvolvidos com tecnologias atualmente consideradas obsoletas (POWER, 2018), o que dificulta a evolução desses sistemas por falta de recursos capacitados, alta complexidade não conseguindo atender a rapidez e agilidade exigidas pelo mercado.

A nuvem é uma estrutura consolidada e comprovadamente flexível, escalável e sob demanda (MELL e GRANCE, 2011) o que possibilita o provisionamento e adaptação mais rápida a alterações atendendo melhor as necessidades do mercado.

Nesse sentido justifica-se a estratégia de migração de reengenharia para software como serviço, para obter os ganhos esperados com a melhor utilização dos recursos da nuvem pela aplicação, conforme apresentado por Zhao e Zhou (2014), essa estratégia de reengenharia do sistema legado para o modelo de software como serviço, é o mais difícil e custoso (ZHAO e ZHOU, 2014).

Por isso, a criação de um roteiro de migração com a finalidade de diminuir o grau de dificuldade encontrado nesse processo de migração, afim de guiar os interessados pelo tema a identificar o que migrar do sistema legado e como migrar afim de obter os ganhos esperados de escalabilidade, flexibilidade e agilidade.

Dentre as diferentes plataformas para nuvem, neste trabalho será adotada a Google Cloud Platform, que é um modelo de instalação de nuvem público. Por que o Google além de ser um provedor de nuvem, é uma empresa que está por trás de vários projetos e publicações de inovações tecnológicas, com a finalidade de colaborar e evoluir a comunidade de tecnologia, alguns exemplos desses projetos são o

*MapReduce* (DEAN e GHEMAWAT, 2004) e *Kubernetes* (WU, 2018), entre outros. Sendo assim, a *Google Cloud Platform* (GCP), serviço de nuvem disponibilizado pelo Google, contém uma vasta documentação própria e de parceiros publicadas para a utilização dos seus recursos, aplicações práticas que ajudarão no desenvolvimento do presente trabalho.

## 1.4. Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas e a estrutura do trabalho.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA apresenta os conceitos básicos relacionados ao trabalho, sendo eles, computação em nuvem, sistemas legados, paradigmas de programação, *Domain Driven Design*, arquitetura baseada em microserviços, os recursos da *Google Cloud Platform* e estratégias de migração para a nuvem.

O Capítulo 3 MIGRAÇÃO DE SISTEMAS LEGADOS PARA NUVEM apresenta os pré-requisitos para migração, um roteiro de migração utilizando os recursos da *Google Cloud Platform* e a aplicação desse roteiro num cenário real.

O Capítulo 4 ANÁLISE DOS RESULTADOS descreve os principais ganhos obtidos com a aplicação de cada passo do roteiro e as dificuldades encontradas no processo de migração.

O Capítulo 5 CONSIDERAÇÕES FINAIS descreve as conclusões deste trabalho e trabalhos futuros.

REFERÊNCIAS relaciona todos os artigos, livros e documentos utilizados na pesquisa para elaboração desse trabalho.

# 2. REVISÃO BIBLIOGRÁFICA

## 2.1. Computação em Nuvem

Computação em nuvem, segundo o NIST (Instituto Nacional de Padrões e Tecnologia), é um modelo para habilitar o acesso global através da rede, sob demanda onde é possível configurar um conjunto compartilhado de recursos como redes, servidores, armazenamento, aplicativos e serviços, que possam ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços. (MELL e GRANCE, 2011)

Esse modelo de nuvem originalmente foi composto por cinco características essenciais, três modelos de serviço e quatro modelos de instalação, descritos conforme a publicação especial do NIST "*The NIST Definition of Cloud Computing*".

### 2.1.1. Características essenciais

- **Autosserviço sob demanda**: o consumidor pode escolher por conta própria os recursos computacionais, que serão consumidos conforme a sua necessidade ou automaticamente, conforme configuração, sem exigir nesse caso a interação humana com o provedor de serviços.
- **Amplo acesso à rede:** os recursos estão disponíveis na rede para acesso de qualquer dispositivo conectado, independente da capacidade de processamento.
- **Agrupamento de recursos:** os recursos computacionais do provedor são agrupados para atender a vários consumidores usando um modelo de multi locação, com diferentes recursos físicos e virtuais atribuídos e reatribuídos dinamicamente de acordo com a demanda do consumidor. Há um senso de independência de localização em que o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos fornecidos, mas pode ser capaz de especificar a localização em um nível mais alto de abstração (por exemplo, país, estado ou datacenter). Exemplos de recursos incluem armazenamento, processamento, memória e largura de banda de rede.

- **Elasticidade rápida:** os recursos podem ser provisionados e liberados de forma elástica, em alguns casos automaticamente, para escalar rapidamente para fora e para dentro de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ilimitados e podem ser apropriados em qualquer quantidade e a qualquer momento.
- **Serviço medido:** os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos aproveitando um recurso de medição em algum nível de abstração apropriado ao tipo de serviço (por exemplo, armazenamento, processamento, largura de banda e contas de usuário ativas). O uso de recursos pode ser monitorado, controlado e relatado, proporcionando transparência tanto para o provedor quanto para o consumidor do serviço utilizado.

### 2.1.2. Modelos de serviços

- **Software como Serviço** *(SaaS - Software as a Service)*: o recurso fornecido ao consumidor é o uso de aplicações do fornecedor executando em uma infraestrutura na nuvem. As aplicações podem ser acessadas através da internet. O consumidor não gerencia nem controla a infraestrutura na nuvem, somente consome as aplicações fornecidas pelo provedor.
- **Plataforma como Serviço** *(PaaS – Platform as a Service):* o recurso fornecido ao consumidor é uma plataforma que permite a criação, hospedagem e gerenciamento seus aplicativos criados ou adquiridos na infraestrutura na nuvem, desenvolvidos com linguagens de programação, bibliotecas, serviços e ferramentas suportados pelo fornecedor da nuvem ou compatíveis. O consumidor tem controle sobre as aplicações desenvolvidas e instaladas e as configurações do ambiente de hospedagem de aplicações.
- **Infraestrutura como Serviço** *(IaaS – Infrastructure as a Service)*: o recurso fornecido ao consumidor é o provisionamento de processamento, armazenamento, comunicação de rede e outros recursos de computação fundamentais nos quais é permitido ao consumidor a instalação e execução de softwares em geral, incluindo sistemas operacionais, aplicativos,

armazenamento, e aplicativos instalados, e possivelmente um controle limitado de alguns componentes de rede (como firewalls).

### 2.1.3. Modelos de instalação

- **Nuvem privada**: a infraestrutura de nuvem é provisionada para uso exclusivo por uma única organização que inclui vários consumidores (por exemplo, unidades de negócios). Ele pode ser de propriedade, gerenciado e operado pela organização, por terceiros ou por alguma combinação deles, e pode existir dentro ou fora das instalações. (MELL e GRANCE, 2011)
- **Nuvem da comunidade:** a infraestrutura de nuvem é provisionada para uso exclusivo por uma comunidade específica de consumidores de organizações que compartilham preocupações (por exemplo, missão, requisitos de segurança, políticas e considerações de conformidade). Ele pode ser de propriedade, gerenciado e operado por uma ou mais das organizações da comunidade, um terceiro ou uma combinação deles, e pode existir dentro ou fora das instalações. (MELL e GRANCE, 2011)
- **Nuvem pública:** a infraestrutura de nuvem é provisionada para uso aberto pelo público em geral. Ele pode ser de propriedade, gerenciado e operado por uma organização comercial, acadêmica ou governamental ou por alguma combinação deles. Existe nas instalações do provedor de nuvem.
- **Nuvem híbrida**: a infraestrutura de nuvem é uma composição de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública) que permanecem como entidades exclusivas, mas unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicativos (por exemplo, estouro de nuvem para balanceamento de carga entre nuvens). (MELL e GRANCE, 2011)

## 2.2. Sistemas Legados

Segundo Bennet (1995) “sistemas legados realizam tarefas úteis para a organização, mas foram desenvolvidos com tecnologias atualmente consideradas

obsoletas” (BENNETT, 1994). Para Umar (1997) “são sistemas com valor crítico para o negócio das empresas desenvolvidos há cinco ou mais anos". (UMAR, 1997)
Um mapeamento realizado pelos estudantes Alex Severo Chervenski e Daniele Martins, apresentam as características mais citadas em artigos do ano de 2015, dentre as quais é importante destacar: desenvolvido com tecnologias obsoletas, vital para a organização, documentação ausente ou desatualizada, difícil manutenção, desenvolvido há muito tempo, difícil ou impossível de estender e integrar, arquitetura rígida e inflexível, ausência de pessoal especializado, requer muita manutenção, manutenção cara e lenta. (CHERVENSKI e MARTINS e BORDIN, 2016)
Em conclusão, podemos entender que sistemas legados são sistemas antigos de alta importância para serem descontinuados, difícil e caros de serem mantidos e lentos para adequação a mudanças necessárias para o negócio.

## 2.3. Paradigmas de Programação

O que é um paradigma de programação? Eles existem desde as primeiras discussões sobre computação, Hailpern em janeiro de 1986 publicou um artigo sobre linguagens multi paradigma, que traz uma definição simples de entender.
Segundo Hailpern, paradigma computacional é “uma maneira de abordar um problema de programação. Uma maneira de restringir o conjunto de soluções. ” (HAILPERN, 1986)
Nesse trabalho serão abordados os paradigmas de programação estruturado e orientado a objetos, devido a aplicação do roteiro apresentado no trabalho ter sido realizada em sistemas legados do mainframe, escritos na linguagem Cobol, ou seja, estruturados. Já o paradigma orientado a objetos porque ajuda no desenvolvimento de sistemas distribuídos, que será a estrutura dos sistemas migrados para a *Google Cloud Platform* nesse trabalho.
As linguagens estruturadas isolam e abstraem as preocupações em procedimentos e funções, enquanto as linguagens orientadas a objetos abstraem as preocupações por meio de classes e objetos. (SIMMONDS, 2012)

## 2.4. *Domain Driven Design*

O conceito de desenho orientado a domínio (*domain driven design*) parte do princípio de que o sistema de software é uma automação desenvolvida para a solucionar problemas de negócios do mundo real. Esse negócio do mundo real, é o domínio do seu sistema. Devemos entender desde o início que o software é originado e profundamente relacionado a este domínio. (AVRAM e MARINESCU, 2006)

Para começar é necessário entender profundamente o domínio do seu sistema com os especialistas de negócios. Após isso deve-se ter em mente uma planta de onde deseja-se chegar com o sistema, inicialmente essa planta será incompleta, por isso é um trabalho que deve ser aprimorado com o tempo. É necessário abstrair esse conhecimento em forma de um modelo de domínio. De acordo com Eric Evans, um modelo de domínio não é um diagrama particular; é a ideia de que o diagrama se destina a transmitir. Não é apenas o conhecimento na cabeça de um especialista no domínio; isto é uma abstração rigorosamente organizada e seletiva daquele conhecimento. (AVRAM e MARINESCU, 2006)

Um modelo é uma parte essencial no desenho de sistemas. Ele é necessário para poder lidar com a complexidade do sistema. Todo o pensamento do processo sobre o domínio é sintetizado neste modelo. Esse modelo deve ser compartilhado com todos os envolvidos no processo de desenvolvimento, como os especialistas de negócio, desenvolvedores, designers. Existem diferentes formas de fazer um modelo de domínio, podendo elas serem: gráficas: diagramas, casos de uso, desenhos, imagens, etc.; escritas: é escrita a visão do time sobre o domínio; através de uma linguagem: é permitido e devido criar uma linguagem para comunicar problemas específicos sobre o domínio. (AVRAM e MARINESCU, 2006)

Tendo um modelo expresso, é iniciado o desenho do código. O desenho de código tem foco nos detalhes, e é muito importante para a qualidade do sistema. Aqui os padrões de desenho de código são úteis, e devem ser aplicados quando necessário. Boas técnicas de codificação ajudam a criar código limpo e sustentável. (AVRAM e MARINESCU, 2006)

Desenho orientado a domínio combina prática de design e desenvolvimento e mostra como design e desenvolvimento podem trabalhar juntos para criar uma melhor solução. Um bom desenho acelerará o desenvolvimento, enquanto o feedback proveniente do processo de desenvolvimento melhorará o desenho. (AVRAM e MARINESCU, 2006)

## 2.5. Arquitetura Baseada em Microserviços

O termo "Arquitetura de Microserviços" surgiu nos últimos anos para descrever um modo particular de projetar aplicativos de software como conjuntos de serviços de implementação independente. (RICHARD, 2016)

As organizações de desenvolvimento de software usam a arquitetura de microserviços para obter uma entrega mais rápida e maior segurança à medida que a escala de seus sistemas aumenta. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)

Mas o que são microserviços? No livro *Building Microservices*, Sam Newman descreve resumidamente como: “Microserviços são pequenos serviços autônomos que trabalham juntos”. (NEWMAN, 2015)

### 2.5.1. Características dos Microserviços

O que faz do microserviço ser mais rápido a mudanças? Justamente a forma como ele deve ser construído. A seguir estão descritas as características principais de como um microserviço deve ser construído, com base nos livros “*Building Microservices*” de Sam Newman e “*Microservice Architecture*” dos autores Irakli Nadareishvili, Ronnie Mitra, Matt McLarty e ike Amundsen.

- **Pequeno em tamanho**, com foco em fazer uma coisa bem-feita, não há uma definição clara sobre o quão pequeno deve ser, ele descreve no livro que quanto maior a maturidade em gerir a complexidade da separação, menor seu microserviço será. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)
- **Autônomo, desacoplado, independente implantável**, cada microserviço é uma entidade separada, podendo ser implementado como um serviço isolado em uma PaaS, o ideal é cada serviço estar em uma máquina separada. A regra de ouro do microserviço é o desacoplamento, o microserviço deve ser capaz de ser alterado sem afetar os demais que consomem suas informações, para isso deve ser projetado com a preocupação do que está expondo e abstraindo para os serviços consumidores. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)

- **Mensagens,** sua comunicação com outros serviços deve ser realizada através de chamadas de rede (mensagens/ APIs, a comunicação é um ponto de atenção importante no sentido do desacoplamento, devem ser projetadas de forma que não seja necessário alterar a comunicação toda vez que o serviço for alterado. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)
- **Limitado por contextos**, cada serviço é definido por domínios de negócio, para realizar essa divisão em microserviços, é imprescindível conhecer sobre, "*Domain-driven-design*", que ajudarão na hora de limitar o contexto dos seus microserviços. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)
- **Construído e liberado com processos automatizados**, os microserviços devem ser desenvolvidos e implantados, via ferramentas automatizadas de entrega e integração contínua. (NADAREISHVILI e MITRA e AMUNDSEN, 2016)

### 2.5.2. Benefícios dos Microserviços

- Tecnologia heterogênea, cada serviço pode ser construído em uma linguagem diferente conforme necessidade, pois cada serviço é independente e permite a experimentação de novas tecnologias que se interligam através de contratos de comunicação estabelecidos entre os serviços, utilizando assim o recurso que melhor atenda a sua necessidade. (NEWMAN, 2015)
- Resiliência, por ser responsável por uma parte e não o todo, se um serviço falhar ele não afetará toda a aplicação, podendo ser tratado individualmente para o correto funcionamento ou descontinuidade conforme necessidade. (NEWMAN, 2015)
- Escalabilidade, cada serviço pode ser dimensionado conforme sua necessidade de processamento, sem gerar ociosidade ou capacitação desnecessária. (NEWMAN, 2015)
- Fácil de implantar, com microserviços podemos fazer uma alteração em um único serviço e implantá-lo independentemente do resto do sistema, permitindo rapidez na implantação seja de novas funcionalidades, correções ou melhorias. (NEWMAN, 2015)

- Alinhado organizacionalmente, microserviços nos permitem alinhar melhor a arquitetura com a organização, ajudando a minimizar o número de pessoas que trabalham em qualquer base de código para atingir o ponto ideal de tamanho e produtividade da equipe, podendo também transferir a propriedade de serviços entre as equipes para tentar manter as pessoas trabalhando em um único serviço. (NEWMAN, 2015)
- Composível, o microserviço deve ser desenhado e construído de forma que seja possível a reutilização por diferentes consumidores de diferentes maneiras para diferentes propósitos. (NEWMAN, 2015)
- Otimiza a substituição, uma vez que é uma funcionalidade isolada, alterá-la ou substituí-la não implica em alterar a aplicação toda, somente a parte necessária. (NEWMAN, 2015)

## 2.6. *Google Cloud Platform*

O *Google Cloud Platform* (GCP) é o serviço de nuvem fornecido pela empresa Google.

O GCP consiste em um conjunto de ativos físicos, como computadores e unidades de disco rígido, e recursos virtuais, como máquinas virtuais (VMs), localizados nos centros de dados do Google em todo o mundo. Cada local de data center está em uma região global. Entre as regiões estão a central dos EUA, a Europa Ocidental o Leste da Ásia. Cada região é uma coleção de zonas, isoladas entre si dentro da região. Cada zona é identificada por um nome que combina um identificador de letra com o nome da região. Por exemplo, a zona a na região do Leste da Ásia se chama asia-east1-a. (GOOGLE 12, 2018)

Essa distribuição de recursos oferece diversas vantagens, inclusive redundância em caso de falha e latência reduzida localizando recursos mais próximos dos clientes. Essa distribuição também introduz regras sobre como recursos podem ser usados juntos. (GOOGLE 12, 2018)

### *2.6.1. Google Compute Engine*

O *Compute Engine* é o serviço de máquinas virtuais da GCP, onde é possível criar a sua infraestrutura de nuvem, configurando as instâncias, tipos de máquinas

utilizadas, processamento, imagens de sistemas operacionais, migração ao vivo, máquinas virtuais preventivas, máquinas virtuais blindadas, modelos para instâncias, grupo de instâncias, armazenamento, redes *Virtual Private Cloud* (VPC), balanceamento de carga e escalonamento, regiões e zonas, controle de acesso e contêineres. (GOOGLE 6, 2018)

Os contêineres podem ser utilizados com as imagens públicas de Linux, Windows Server 2016 ou a imagem de sistema operacional otimizada para contêineres da Google. Contêineres permitem que os aplicativos sejam executados com menos dependências na máquina virtual do host e sejam executados independentemente de outros aplicativos em contêiner que você implanta na mesma instância de máquina virtual. Sendo assim, os aplicativos em contêiner tornam-se mais portáteis, mais fáceis de implantar e mais fáceis de manter em escala. (GOOGLE 7, 2018)

### *2.6.2. Google Kubernetes Engine*

O *Google Kubernetes Engine* (GKE) fornece um ambiente gerenciado para implantar, gerenciar e dimensionar aplicativos em contêineres usando a infraestrutura do Google. Esse ambiente fornecido pelo GKE consiste em várias máquinas (especificamente, instâncias do Google Compute Engine) agrupadas para formar um cluster. (GOOGLE 8, 2018)

“Os clusters GKE são ativados pelo sistema de gerenciamento de clusters de código aberto do Kubernetes. O Kubernetes fornece os mecanismos através dos quais você interage com o cluster. Você usa os comandos e recursos do Kubernetes para implantar e gerenciar seus aplicativos, executar tarefas de administração e definir políticas, além de monitorar a integridade de suas cargas de trabalho implementadas. ” (GOOGLE 8, 2018)

### *2.6.3. Google App Engine*

O *App Engine* é uma plataforma como serviços da GCP para desenvolvimento de software. Utilizado para criar e escalonar aplicações automaticamente na nuvem da Google, fazer gerenciamento de infraestrutura e com suporte as linguagens: Node.js, Java, Ruby, C#, Go, Python ou PHP em seu ambiente ou a utilização de um

ambiente próprio de execução no qual pode ser utilizado qualquer outra linguagem, possui uma grande variedade de ferramentas e permite a integração de qualquer biblioteca através da conexão de um contêiner do Docker. (GOOGLE 9, 2018)

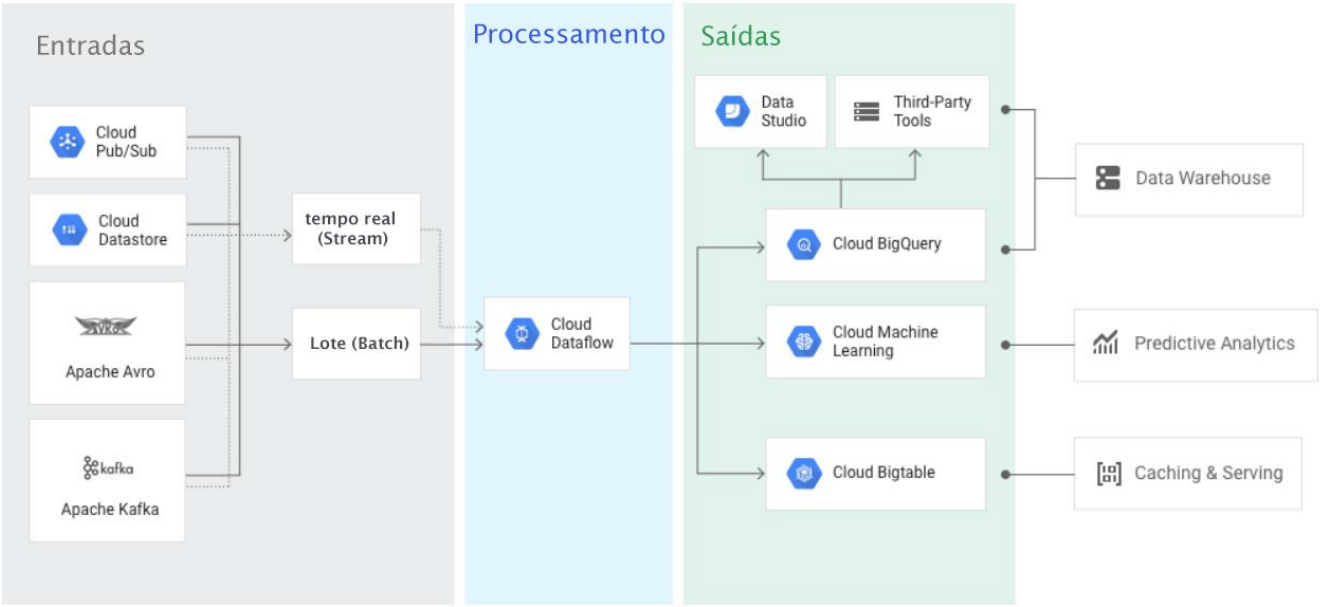
O *App Engine* permite o encaminhamento de solicitações recebidas para versões diferentes do app, execução de testes A/B e a permissão de implementação de recursos adicionais, bem como definição de regras de acesso através de seu firewall e utilização dos certificados SSL/TLS* gerenciados por padrão no seu domínio personalizado sem custo adicional. (GOOGLE 9, 2018) Além disso o App Engine possui uma série de APIs prontas para utilização nas linguagens suportadas. (GOOGLE 9, 2018)

O *App Engine* oferece duas opções de ambientes para os aplicativos sendo eles: o ambiente padrão e o ambiente flexível. Ambos os ambientes têm o mesmo fluxo de trabalho do desenvolvedor centrado em código, dimensionam de maneira rápida e eficiente para lidar com a demanda crescente e permitem que a utilização da tecnologia de serviço comprovada do Google para criar aplicativos Web, móveis e *IoT* rapidamente e com mínima sobrecarga operacional. Embora os ambientes disponíveis tenham muito em comum, eles diferem de algumas maneiras importantes, porém ambos podem ser utilizados em conjunto permitindo assim que seus serviços aproveitem os benefícios individuais de cada um. (GOOGLE 11, 2018)

### *2.6.4. Cloud Dataflow*

O *Cloud Dataflow* é um serviço gerenciado para transformar e enriquecer dados em modos de fluxo (tempo real) e em lote (histórico) com igual confiabilidade e expressividade, ou seja, um recurso de extração, transformação e carga (ETL) disponibilizado na GCP, a figura 1 apresenta de forma gráfica a transformação dos dados com o Dataflow.

**Figura 1- Transformação de Dados utilizando o *Cloud Dataflow***




**Fonte: Google 2, 2018**

O *Dataflow* possui uma abordagem sem servidor de provisionamento e gerenciamento de recursos, com acesso à uma capacidade praticamente ilimitada permitindo assim a solução dos maiores desafios de processamento de dados. (GOOGLE 2, 2018)

### *2.6.5. Cloud Pub Sub*

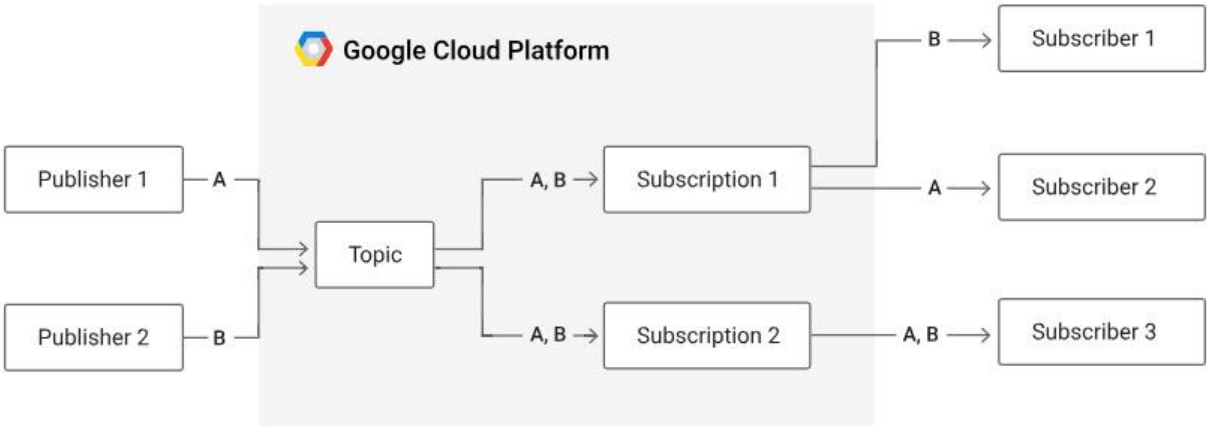
O *Cloud Pub Sub* é o serviço de transmissão de mensagens em tempo real, escalável disponível na Google Cloud Platform, totalmente gerenciado e assíncrono, que permite enviar e receber mensagens entre aplicativos independentes, por trabalhar de forma desacoplada dos remetentes e destinatários. (GOOGLE 4, 2018)

Alguns conceitos chave são necessários para entender um serviço de mensageria sendo eles:

- Mensagem: conteúdo transmitido, são os dados que se movem através do serviço.
- Tópico: onde são publicadas as mensagens a serem distribuídas
- Assinatura: entidade que representa o interesse em receber mensagens sobre um determinado tópico.
- Publisher (produtor): cria as mensagens e envia (publica) para o serviço de mensagens em um tópico especificado.
- Assinante (consumidor): recebe mensagens do tópico assinado.

O fluxo básico de mensagens pelo *Cloud Pub Sub* pode ser resumido na figura 2, extraída do artigo "*A Google-Scale Messaging Service*" publicado pelo Google.

**Figura 2 - Fluxo básico de mensagens Cloud Pub Sub**




**Fonte: Google 4, 2018**

Nesse cenário, há dois produtores publicando mensagens distintas em um único tópico. Há duas assinaturas para o tópico, onde a primeira assinatura tem dois assinantes e a segunda assinatura tem um assinante. As letras A e B representam mensagens. A mensagem A vem do produtor 1 e é enviada para o Assinante 2 via Assinatura 1 e para o Assinante 3 via Assinatura 2. A Mensagem B vem do Publicador 2 e é enviada para o Assinante 1 via Assinatura 1 e para o Assinante 3 via Assinatura 2.

Muito importante para esse tipo de serviço de mensageria, é o monitoramento, pois caso alguma máquina onde esteja qualquer dessas entidades (produtores, tópicos e assinantes) ficar indisponível por algum motivo, é necessário ter ações para que

mensagens não sejam perdidas, o serviço *Cloud Pub Sub,* possui esse monitoramento que permite identificar automaticamente se há algo errado através dos acordos de níveis de serviço (SLAs).

### *2.6.6. Cloud Functions*

O *Google Cloud Functions* é um ambiente de execução sem servidor para criar e conectar serviços em nuvem. Os serviços criados são conectados a eventos emitidos a partir de sua infraestrutura e serviços em nuvem, evitando assim a necessidade de provisionamento de infraestrutura ou preocupação em gerenciar qualquer servidor.
O *Cloud Functions* pode ser escrito e executado em: JavaScript (Node.js 6.14.0, Node.js 8.11.1) ou Python (Python 3.7.1) no Google Cloud Platform, facilitando a portabilidade e o teste local. (GOOGLE 3, 2018)

### *2.6.7. Cloud Storage*

O *Google Cloud Storage* é a ferramenta de armazenamento disponibilizada pela Google, onde é possível armazenar e recuperar a qualquer momento o conteúdo armazenado independente de sua localização e quantidade de dados. O *Cloud Storage* pode ser utilizado para uma variedade de cenários, incluindo conteúdo de sites, armazenamento de dados para arquivamento e recuperação de desastres ou distribuição de grandes objetos de dados para usuários e sistemas por meio de download direto. (GOOGLE 5, 2018)

### 2.6.8. Istio

O Istio é uma malha de serviço (*service mesh*) independente de código aberto que fornece os fundamentos necessários para executar com sucesso uma arquitetura de microserviço distribuída. O Istio reduz a complexidade de gerenciamento de implantações de microserviços pois fornece uma maneira uniforme de conectar, proteger e monitorar microserviços, além de ser configurável o controle de acesso do

Istio, as regras de roteamento e assim por diante, usando uma API personalizada do Kubernetes, através do kubectl ou da ferramenta de linha de comando Istti istioctl, que fornece validação extra.

O Istio disponibiliza as funcionalidades de: balanceamento automático de carga para tráfego HTTP, gRPC, WebSocket, MongoDB e TCP; controle refinado do comportamento do tráfego com regras avançadas de roteamento, novas tentativas, recuperação de falhas e injeção de falhas; uma camada de política configurável e API que suporta controles de acesso, limites de taxa e cotas; métricas, logs e rastreamentos automáticos para todo o tráfego dentro de um cluster, incluindo entrada e saída do cluster; comunicação segura de serviço a serviço em um cluster com autenticação e autorização baseadas em identidade forte. (GOOGLE 10, 2018)

## 2.7. Estratégias de Migração para a Nuvem

Há várias formas de migrar uma aplicação para nuvem, seja ela uma migração somente de infraestrutura, migração para adequação a uma plataforma como serviço, substituição da aplicação para um software como serviço quando aplicável, revisão da aplicação para torná-la um serviço, entre inúmeras outras, nesse trabalho a proposta é de migração, que pode ser classificada na estratégia de reengenharia para *SaaS*.

Zhao e Zhou (2014) apresentam em seu artigo “*Strategies and Methods for Cloud Migration*” algumas das estratégias de migração para nuvem mais conhecidas e praticadas apresentando uma análise comparativa entre elas classificando por carga de trabalho, complexidade, nível de modificação e efeitos da migração conforme apresenta a tabela 1.

**Tabela 1 - Comparação entre as estratégias de Migração**

| | *Migração para IaaS* | *Migração para PaaS* | *Substituição para SaaS* | *Revisão com base em SaaS* | *Reengenharia para SaaS* |
|---|---|---|---|---|---|
| Carga de trabalho | Pouco | Moderado | Pouco | Moderado | Muito |
| Nível de complexidade | Fácil | Moderado | Fácil | Moderado | Difícil |
| Nível de modificação | Sem necessidade de modificação | Modificar a aplicação para ser compatível | Sem necessidade de modificação | Integração de serviços e dados, composição de serviços | Engenharia reversa, redesenho de estrutura, visão futura de engenharia |
| Efeitos | Economizar com manutenção de infra | Livre de gerenciamento de recursos | Preços flexíveis, mecanismo, conveniente manutenção | Preços flexíveis, mecanismo, conveniente manutenção, reutilização | Preços flexíveis, mecanismo, conveniente manutenção, reutilização, escalabilidade |

**Fonte: (Zhao e Zhou, 2014)**

Através da Tabela 1 é possível identificar que somente a reengenharia para SaaS, atenderá todos os objetivos esperados na proposta de migração apresentada nesse trabalho que visa atingir os seguintes requisitos após sua implementação: velocidade de desenvolvimento e entrega, escalabilidade, reutilização e facilidade de manutenção com menor impacto possível.

Na reengenharia de um sistema legado para *SaaS*, os sistemas legados serão reprojetados para o serviço em nuvem, o que é desafiador pela complexidade agregada, pois exige uma engenharia reversa do legado, redesenho de estrutura, geração de serviço, etc. (ZHAO e ZHOU, 2014)

Além das estratégias de migração para nuvem, há também as estratégias de migração que levam em consideração o escopo da migração se total, parcial, incremental ou encapsulamento, que normalmente são selecionadas através de uma análise de riscos, custos e tempo versus o atendimento das necessidades de negócio e requisitos. (ALTHANI e KHADDAJ, 2017)

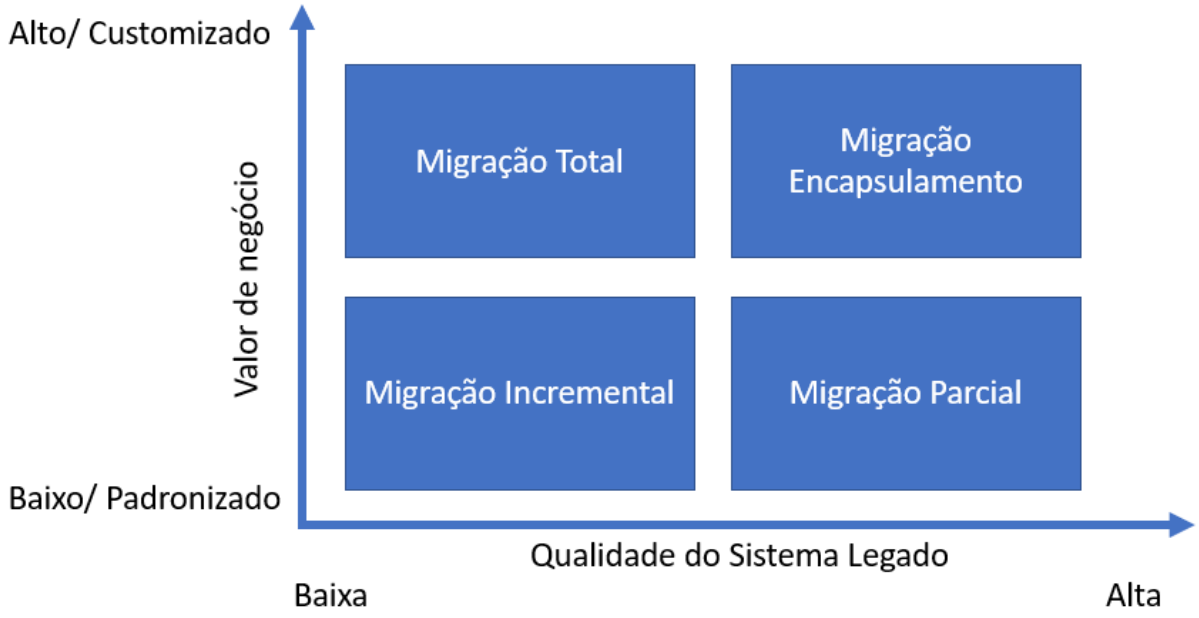
- **Migração total:** consiste em reconstruir o sistema do zero com o objetivo de adequação a uma infraestrutura mais moderna (EMBLEY e THALHEIM, 2011), e de substituição total do sistema legado, desativando-o assim que o novo sistema estiver com todos os dados migrados e operando produtivamente (BISBAL et. al., 1999)
- **Migração parcial:** consiste em substituir parte do sistema legado, algumas aplicações, componentes para um ambiente específico, usando por exemplo

uma aplicação de mercado genérica que atenda as necessidades de negócio. (ALTHANI e KHADDAJ, 2017)

- **Migração incremental:** consiste em reconstruir o sistema de forma incremental e evolutiva, um exemplo desse tipo de migração é o método renascentista. (WARREN e RANSOM, 2002)
- **Migração por encapsulamento:** consiste em encapsular o código existente em uma interface moderna, mantendo o legado como sistema principal. (ALTHANI e KHADDAJ, 2017) (SHAIEF, 2011)

Althani e Khaddaj através de sua revisão sistemática apresentam de forma comparativa as vantagens e desvantagens de cada estratégia levando em consideração o aspecto de qualidade do sistema legado relacionando a escolha do método de migração mais adequado para atendimento do valor de negócio esperado, conforme apresenta a Figura 3.

**Figura 3 - Comparação entre abordagens de migração**




**Fonte: (ALTHANI e KHADDAJ, 2017)**

# 3. MIGRAÇÃO DE SISTEMAS LEGADOS PARA NUVEM

## 3.1. Proposta de Migração Utilizando GCP

A proposta de migração desse trabalho é baseada na estratégia de reengenharia para SaaS, pois a estrutura arquitetural encontrada no sistema legado utilizado para a elaboração do roteiro, estar em uma arquitetura monolítica, onde um único programa é responsável por inúmeras funcionalidades. O que torna difícil sua manutenção ou alteração devida a alta complexidade de entendimento e testes regressivos para garantia de que o que foi alterado não impactou o restante do sistema.

Além da estratégia de migração para SaaS, também será utilizada a abordagem de migração parcial, com o modelo de instalação em nuvem pública *Google Cloud Platform*.

No capítulo 2 desse trabalho, foram apresentados alguns conceitos básicos que serão utilizados nesse roteiro de migração utilizando a GCP.

### 3.1.1. Pré requisitos para migração de um sistema

Avaliar a necessidade de migração, o sistema a ser migrado deve ser avaliado afim de identificar o retorno sobre um investimento alto como o de migração. Conforme apresentado na seção 2.7, é necessário avaliar qual abordagem de migração a ser adotada. Essa abordagem de migração pode ser uma combinação entre as abordagens afim de obter o resultado esperado.

Afim de identificar a abordagem a ser utilizada, é necessário avaliar a qualidade do sistema legado e o valor de negócio agregado. Conforme apresentado na seção 2.2 algumas questões podem ser consideradas afim de auxiliar na identificação da necessidade de migração do sistema legado: É um sistema que necessita de muita manutenção, e qual sua complexidade? São requisitadas novas funcionalidades frequentemente ou é um sistema relativamente estável que está atendendo da forma que foi construído? Está construído numa tecnologia onde há facilidade de encontrar recursos capacitados? Necessita de escalabilidade? Tem funcionalidades que poderiam ser reutilizadas por outros sistemas? Se as respostas forem, muita manutenção, alta complexidade, dificuldade de evolução, e falta de recursos

capacitados, é um grande sinal de que a migração deve ser aplicada para esse sistema.

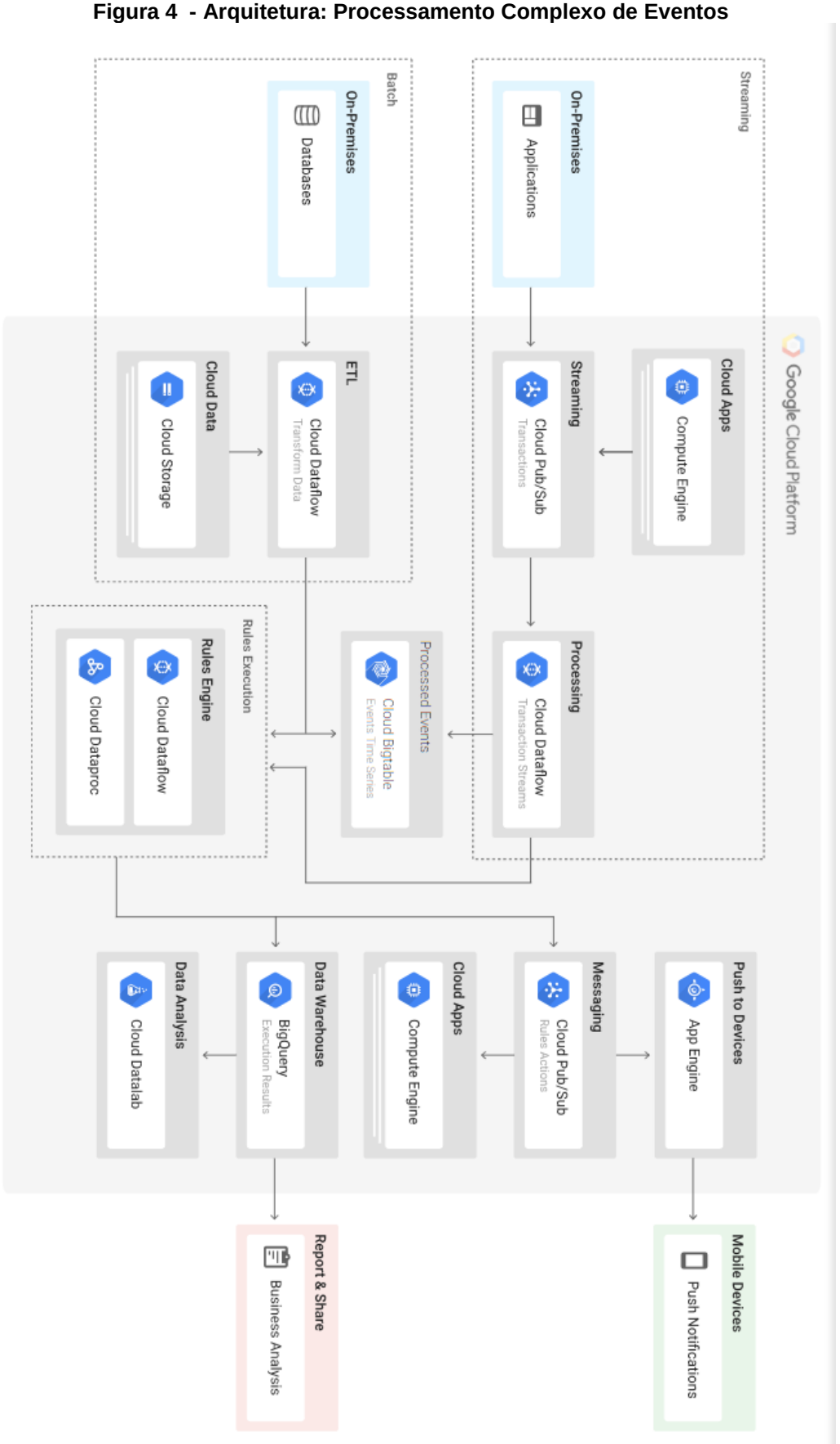
### 3.1.2. Roteiro de Migração

1. **Seleção do time de migração:** Definido que a migração é a melhor escolha, conforme a seção 2.4, é necessário reunir um time multidisciplinar que conheça as plataformas que serão trabalhadas no cenário proposto, analistas e desenvolvedores cobol com conhecimento funcional e técnico do sistema legado, analistas e desenvolvedores com conhecimentos de *Domain Driven Design* e linguagem java, arquitetura baseada em microserviços, analistas de negócio do sistema a ser migrado, e eventualmente analistas especialistas em GCP para dar suporte ao time de migração visto que foi a plataforma escolhida para esse trabalho.
2. **Definição da estratégia de migração**: O próximo passo após a definição do time, é a avaliação da estratégia de migração a ser utilizada com base no sistema legado e nos objetivos almejados.
3. **Definição do domínio do sistema legado**: Definida a estratégia de migração, devem ser iniciadas as discussões acerca do domínio do sistema legado a ser migrado, quais contextos serão considerados para migração e quais serão mantidos na estrutura antiga, utilizando DDD, deve ser elaborado o modelo de domínio do sistema legado pensando também na visão futura desse sistema, conforme apresentado na seção 2.4.
4. **Abstração dos módulos:** Com o modelo de domínio definido, conforme apresentado na seção 2.4, é necessário realizar uma análise afim de abstrair os módulos de acordo com os contextos, separando o que pode ser acessado por cada contexto, esse é um passo importante para atender a característica de desacoplamento dos microserviços apresentada na seção 2.5.1, que serão desenvolvidos a partir dessa definição.
5. **Definição de arquitetura:** Avaliados o contexto do sistema legado que será recriado em microserviços nesse trabalho, e as funcionalidades do sistema legado que continuarão em funcionamento, é necessário avaliar a comunicação entre o sistema legado e os serviços desenvolvidos na

plataforma. Será gerado um arquivo? Serão publicadas mensagens de forma online em um fila? Será disponibilizada uma base de dados para acesso? Tudo vai depender da necessidade funcional identificada, questões como performance, segurança e disponibilidade são importantes de considerar. Para a opção do arquivo é possível subir o arquivo para o recurso de *cloud storage* (2.6.7), e posteriormente ser transformado pelo *cloud dataflow* (2.6.4). Para a opção de mensagem online pode utilizado o recurso pub sub (2.6.5).

A Figura 4, apresenta um exemplo de arquitetura onde as aplicações legadas estão no ambiente *“On-Premisses”,* na parte “S*treaming*” é apresentado um fluxo online de tratamento dos eventos e disponibilização para as aplicações, e na parte “*Batch*” é apresentado um fluxo de transformação do lote de informações provenientes do legado. Portanto, conforme a necessidade da funcionalidade que está sendo criada, é possível utilizar ambos recursos da GCP, não limitando as possibilidades de desenvolvimento.

**Figura 4 - Arquitetura: Processamento Complexo de Eventos**




**Fonte – Google 1, 2018**

6. **Configuração do ambiente e recursos:** Após a definição da construção das novas funcionalidades e integração com as existentes, é necessário realizar a configuração do ambiente da nuvem bem como os seus recursos adicionais que serão utilizados com base na análise realizada nos itens 3, 4 e 5 desse roteiro.
7. **Definição da comunicação:** Seguindo o padrão de arquitetura orientado a microserviços conforme apresentado na seção 2.5, os serviços devem ser construídos de forma independente e desacoplada, dessa forma a comunicação deve ser realizada através de mensagens. A comunicação por mensagens na *Google Cloud Platform* é realizada através do recurso de *Cloud Pub Sub*.
8. **Construção de microserviços:** Além da definição de como será realizada a transmissão das informações do legado para a nuvem e vice-versa, é necessário construir os microserviços com base nas funcionalidades extraídas do modelo de domínio. Para a construção e execução dos microserviços na GCP, é utilizado o recurso do *Cloud Functions* (2.6.6).
9. **Configuração do monitoramento:** Uma vez implantados os microserviços e conectados através das mensagens, é necessário, obter as informações de monitoramento desses serviços e filas, bem como do ambiente, para detecção de falhas, o Google disponibiliza o recurso de *Stackdriver* para verificar tudo que está acontecendo no projeto criado, que deve ser devidamente configurado para obter os melhores ganhos.
10. **Gerenciamento de microserviços**: Todos os microserviços desenvolvidos, devem ser gerenciados conforme sua utilização. Integrando o monitoramento realizado pelo Stackdriver ao serviço Istio apresentado na seção 2.6.8, que deve ser configurado no Compute Engine, ou Kubernetes Engine é possível realizar as configurações da malha de serviços, de forma segura, portabilizável conectando os microserviços com menos complexidade.

## 3.2. Aplicação do Roteiro

Neste item será apresentada a aplicação de roteiro em um caso real, e por questões de sigilo, os nomes utilizados nesse trabalho são fictícios, bem como detalhes de

funcionalidades e implementação não serão abordados, sendo apresentados de forma macro a migração realizada com foco nas ferramentas utilizadas do *Google Cloud Platform.*

### 3.2.1. Pré requisitos para Migração do Sistema

O contexto da necessidade de migração de uma aplicação legada (Mainframe / Cobol), justifica-se pela necessidade de novas funcionalidades que não eram possíveis de serem agregadas na tecnologia existente, por questões de aumento de processamento, que acarretariam em aumento de custos, além de falta de recursos capacitados na tecnologia Cobol.

A migração realizada utilizando esse roteiro durou em torno de 9 meses para implantação, devido a alta complexidade do sistema legado, bem como as necessidades de negócios e processos internos obrigatórios.

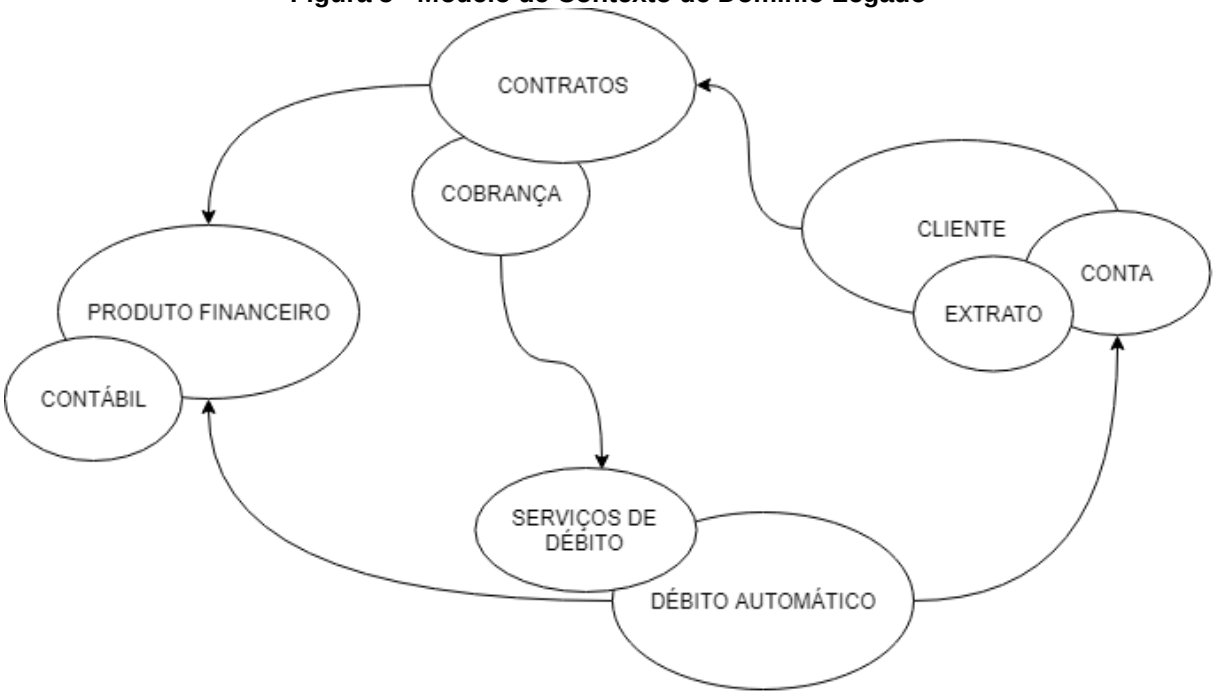
### 3.2.2. Passos do roteiro:

1. **Seleção do time de migração:** Foram selecionados, 2 engenheiros de sistemas com conhecimento funcional e técnico da aplicação legada, sendo eles um sênior e um pleno, 1 desenvolvedor pleno com conhecimento em linguagem Cobol, 2 engenheiros de sistemas com conhecimento em aplicações distribuídas, sendo 1 sênior e um pleno, 1 arquiteto de sistemas sênior com conhecimento em modelagem orientada a domínio, arquitetura microserviços, 2 desenvolvedores juniores com conhecimento em linguagem java, 1 desenvolvedor pleno com conhecimento em linguagem java e aplicações *cloud native* e 1 consultor GCP. Além do time de migração, foi necessário o constante alinhamento com os analistas de produto para garantir o escopo das novas funcionalidades solicitadas.
2. **Definição da estratégia de migração:** O time após analisar o tamanho e valor de negócio da aplicação legada, bem como os riscos envolvidos na indisponibilização dessa aplicação, identificou a vantagem na utilização da estratégia de migração parcial, desenvolvendo somente as novas funcionalidades na nuvem e integrando-as com a plataforma legada, uma vez que o sistema legado possui muitas integrações com outros sistemas legados o que necessitaria envolvimento de outras áreas inviabilizando a migração.

Conforme as análises comparativas apresentadas na seção 2.7 deste trabalho, as estratégias escolhidas pelo time ao aplicar esse roteiro foram: migração parcial, reengenharia para SaaS utilizando o modelo de instalação híbrido, onde o legado foi mantido no ambiente de infraestrutura interno (*On-Premisses*) e as novas funcionalidades conectadas ao legado na nuvem pública da *Google Cloud Platform*.

3. **Definição do domínio do sistema legado:** O domínio extraído do sistema legado, está no contexto de serviço financeiro de pagamento automático em conta conforme apresentado na Figura 5. Muitos produtos financeiros, e clientes utilizam a modalidade de débito automático para realizar os pagamentos em dia.

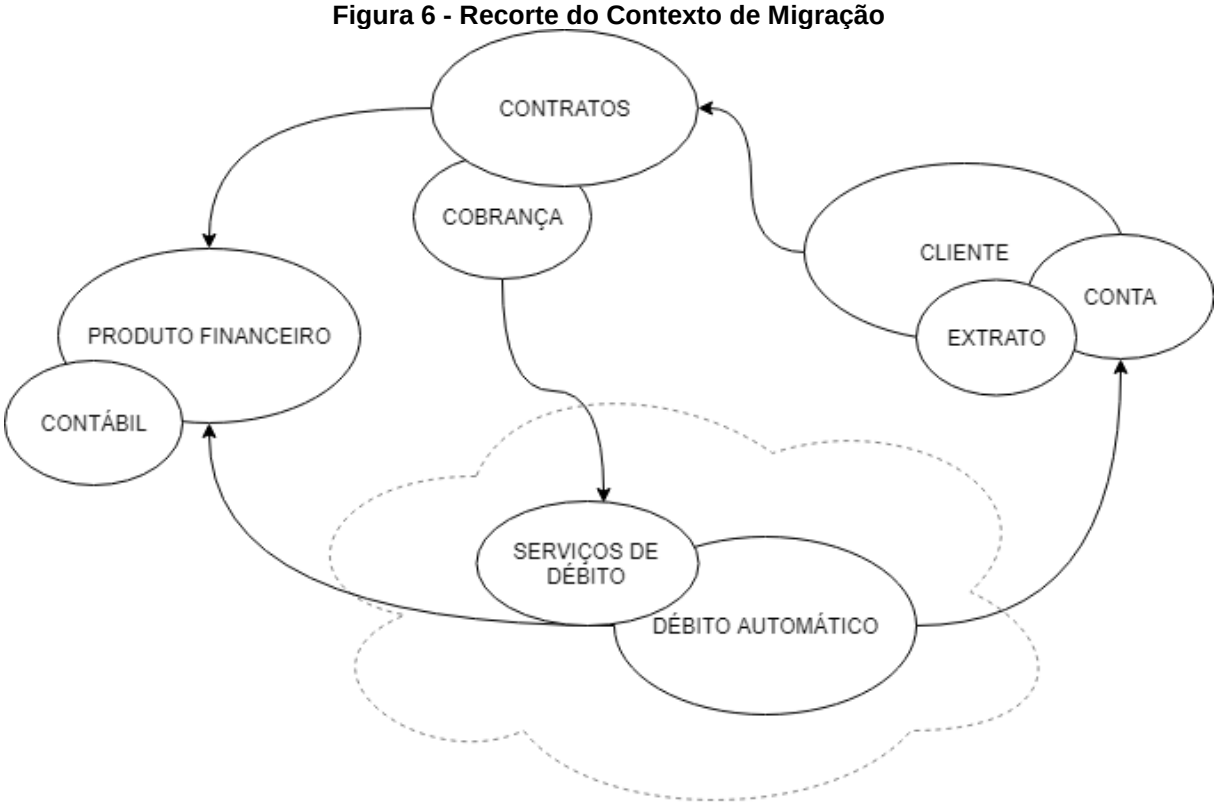
**Figura 5 - Modelo de Contexto de Domínio Legado**




**Fonte: Elaborado pelo autor.**

4. **Abstração dos módulos:** Após identificado o contexto do sistema legado, foi selecionado para migração o contexto de débito automático para fins de migração parcial conforme apresentado na Figura 6, sendo assim parte do débito automático continuará no legado e as novas funcionalidades desenvolvidas na nuvem.

**Figura 6 - Recorte do Contexto de Migração**

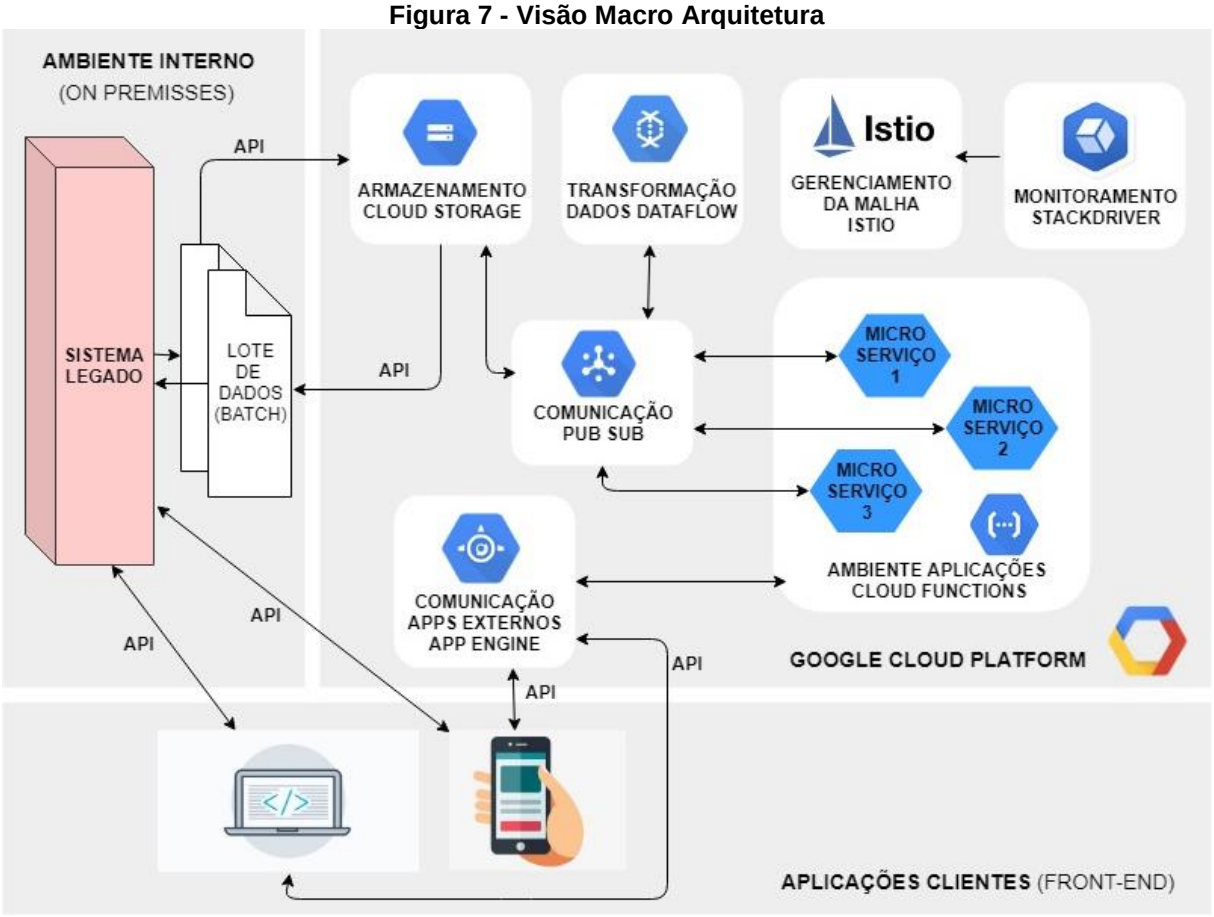



**Fonte: Elaborado pelo autor**

5. **Definição de arquitetura:** Após selecionado o contexto para migração, foram analisadas as necessidades de negócio e requisitos não funcionais do sistema. A performance entre a convivência do legado com as novas aplicações, ex.: velocidade de transferência e processamento de grande quantidade de registros (ex.:20 milhões de registros) do legado

Para atender as necessidades funcionais e técnicas, foi definido que a migração seria realizada através da troca de arquivos texto entre o sistema legado e a plataforma na nuvem utilizando a API (Interface de Programação de Aplicação) do recurso de cloud storage da Google, conforme apresentado na Figura 7.

**Figura 7 - Visão Macro Arquitetura**




**Fonte: Elaborado pelo autor.**

6. **Configuração do ambiente e recursos**: A configuração da nova infraestrutura foi realizada pelo consultor GCP, e os recursos utilizados nessa migração foram: *Google Compute Engine, Google App Engine, Cloud Dataflow*, *Cloud Pub Sub, Cloud Functions, Cloud Storage, Stackdriver* e o serviço de malha Istio.
7. **Definição da comunicação:** Para a comunicação dos microserviços construídos foi utilizado o recurso Pub Sub, através das assinaturas nos tópicos as informações são recebidas e armazenadas em bases de dados próprias de cada microserviço.

Na comunicação entre o legado e os microserviços além do recurso de pub sub, também foram utilizados os recursos de cloud storage para armazenar os arquivos texto de entrada e saída, e o dataflow para transformar o conteúdo dos arquivos em mensagens a serem publicadas no pub sub e enviadas aos microserviços conforme as assinaturas definidas.

8. **Construção de microserviços:** Os microserviços foram construídos em java com framework Spring, utilizando o cloud functions, cada microserviço com uma base de dados própria, alguns com base *Cloud Spanner* (banco de dados relacional/ SQL), outros com uma base *Cloud Bigtable* (banco de dados não relacional / NoSQL), conforme a necessidade de negócio.
9. **Configuração monitoramento:** O monitoramento foi realizado através do recurso do *Stackdriver,* através do monitoramento de logs, erros e métricas monitoramento de tempo de execução, uso de memória, quantidade de execuções integradas aos recursos de cloud functions e pub sub.
10. **Gerenciamento de microserviços:** O gerenciamento da malha de serviços da aplicação foi configurado no recurso de *Compute Engine*, utilizando o Istio integrado ao monitoramento via *Stackdriver* das máquinas, microserviços integrando ao *Cloud Functions* e a comunicação através do recurso *Pub Sub*.

## 3.3. Validação do Roteiro

Para identificar os passos do roteiro onde há maior facilidade, dificuldade e complexidade, foi elaborado um questionário voltado aos engenheiros e desenvolvedores da empresa que trabalham com projetos de migração.
O questionário e as respostas obtidas podem ser consultados na seção de apêndices ao final desse trabalho.

## 3.4. Considerações sobre o Capítulo

Nesse capítulo foram apresentados, os pré-requisitos para a migração de um sistema legado, um roteiro de migração na forma de um roteiro de implementação, a aplicação do roteiro em um caso real, num sistema legado que foi migrado utilizando a *Google Cloud Platform* e um questionário de validação do roteiro apresentado para identificar os passos de maior facilidade, dificuldade e complexidade existentes no roteiro.

# 4. ANÁLISE DOS RESULTADOS

## 4.1. Ganhos na aplicação do Roteiro

Com a aplicação do roteiro apresentado nesse trabalho, foi possível identificar a diminuição da complexidade no processo de migração, devido a organização dos passos a serem executados, bem como a completude e qualidade, uma vez que todos os pontos principais necessários para executar a migração foram elencados, de forma simples e objetiva.

- **Passo 1 - Seleção do time**: com a seleção de um time multidisciplinar a sinergia dos conhecimentos de tecnologia e negócio, garantem que a aplicação seguirá os melhores padrões de tecnologia sem perder as características cruciais de negócio, sendo projetada e desenvolvida uma aplicação completa no momento da migração com base em reengenharia, sem essa seleção a visão dos analistas responsáveis pela migração correm o risco de deixar a aplicação na melhor tecnologia e perder valor de negócio, ou de difícil manutenção por dar ênfase na funcionalidade sem a visão da tecnologia.
- **Passo 2 – Definição da estratégia de migração:** a definição da estratégia adotada pelo time é o que servirá de guia para o início de desenho do projeto de migração, sem uma estratégia bem definida do que será migrado e como, é fácil perder tempo e dinheiro por realizar uma migração sem planejamento, de forma que não obterá os ganhos esperados.
- **Passo 3 – Definição do domínio do sistema legado:** o primeiro passo em uma migração com estratégia de reengenharia para SaaS, que é a apresentada nesse trabalho, é ter bem definido o domínio de negócio do sistema existente, para que o novo sistema seja desenvolvido de forma aderente ao negócio do sistema sem perder o valor de negócio e de forma estruturada com a complexidade de uma aplicação com característica de software como serviço.
- **Passo 4 - Abstração dos módulos:** a abstração dos módulos, é o que garantirá a agilidade na manutenção e inclusão de novas funcionalidades (agilidade), uma vez que as aplicações serão menores e desconectadas, as

alterações se tornam menos complexas, além dos ganhos em escalabilidade, se um módulo da aplicação necessita de mais memória, mais disco, somente ela terá alterações de configuração de ambiente, evitando falhas por sobrecarga na infraestrutura e custos desnecessários por ociosidade de recursos de infraestrutura.

- **Passo 5 - Definição da arquitetura**: sem a seleção de uma arquitetura adequada de integração, é praticamente impossível gerenciar uma aplicação com característica de software como serviço, devido alta complexidade dessa estratégia de migração, um sistema sem uma arquitetura definida, não estará estruturado para obter os ganhos que o ambiente da nuvem fornece, podendo ter na nuvem uma aplicação não escalável, de difícil manutenção e alteração.
- **Passo 6 - Configuração do ambiente e recursos:** sem uma configuração adequada a aplicação desenvolvida pode sofrer falhas de funcionamento, comunicação, desde indisponibilidade parcial ou total, acarretando uma série de problemas não só sistêmicos, mas também de negócio.
- **Passo 7 - Definição da comunicação:** essa definição é importante para que todas as informações sejam trafegadas conforme necessidade entre as aplicações, se há necessidade de acesso online ou batch, questões de disponibilidade e perda de informações, podem ocorrer quando não é realizada uma definição adequada de comunicação.
- **Passo 8 - Construção de microserviços:** a preocupação da nova aplicação construída no ambiente da nuvem ser baseada na arquitetura de microserviços é principalmente devida ao fato de que se estruturada conforme a literatura apresentada na seção 2.5, é o que garantirá a melhor utilização dos recursos oferecidos pela nuvem, de flexibilidade, agilidade, escalabilidade, pois cada serviço tendo isoladamente sua função é possível ser medido e provisionado o ambiente conforme demanda (escalabilidade), bem como manutenções e correções serão direcionadas para os serviços sem impactar os demais (flexibilidade, manutenibilidade e agilidade).
- **Passo 9 - Configuração do monitoramento:** não somente para aplicações em nuvem, mas principalmente para elas, é imprescindível ter tudo que está construído sendo monitorado afim de evitar falhas, corrigi-las tempestivamente quando ocorrem e ter os insumos necessários de utilização

de recursos computacionais por cada serviço (aplicação), que fornecerá os dados de base para alterações de configurações da nuvem conforme a necessidade, garantindo a saúde da aplicação, sem monitoramento nada disso seria possível ser obtido de forma simples e rápida.

- **Passo 10 - Gerenciamento de microserviços:** sem o gerenciamento da malha de microserviços questões como balanceamento de carga, recuperação automática em casos de falhas, ficaria de responsabilidade do desenvolvedor, quando já existem ferramentas disponíveis para utilização, facilitando o gerenciamento e monitoramento dos serviços, garantindo a flexibilidade e escalabilidade automática a partir de um monitoramento ativo dos serviços.

Os detalhes de configuração da GCP não foram abordados nesse trabalho, pois a Google além de fornecer o serviço de configuração, tem documentado o roteiro de utilização de cada recurso citado nesse trabalho no seu repositório de documentação da *Google Cloud Platform*, disponível online de forma aberta e atualizada para consulta.

Os detalhes das funcionalidades existentes no sistema legado migrado parcialmente e as novas funcionalidades construídas não foram descritas por questões sigilosas, o intuito da apresentação da aplicação do roteiro em um caso real foi apresentar os recursos utilizados para que a migração seja efetiva em termos de flexibilidade, agilidade, escalabilidade, reutilizável e de fácil manutenção.

## 4.2. Dificuldades encontradas no processo de Migração

O roteiro apresentado é um guia para o processo prático de implementação de uma migração para nuvem utilizando a estratégia de reengenharia para SaaS, que elenca os passos a serem realizados na jornada de migração, bem como as ferramentas GCP, porém sem detalhes na construção das aplicações novas, como frameworks de desenvolvimento, modelagem de dados, e esteira de entrega contínua.

A mudança de paradigmas de programação implica num processo de desconstrução do que existe no legado para recriação em um ambiente totalmente diferente, por isso é necessário muito cuidado para não construir as novas funcionalidades na nuvem conforme foram desenvolvidas no legado, pois se fosse dessa forma os ganhos obtidos seriam somente de ambiente e não no comportamento do sistema,

portanto é de grande importância ter o time multidisciplinar que domine as novas tecnologias e negócio.

### 4.3. Considerações do Capítulo

Esse capítulo apresentou os principais ganhos e dificuldades encontrados no processo de migração com a aplicação do roteiro proposto nesse trabalho.

# 5. CONSIDERAÇÕES FINAIS

## 5.1. Contribuições do Trabalho

Migrações de sistemas legados utilizando a estratégia de reengenharia para SaaS são altamente complexas, porém não impossíveis, a contribuição do roteiro apresentada é em tornar o processo menos complexo, uma vez que os principais passos dessa estratégia de migração foram abordados num formato de roteiro, garantindo a completude e qualidade nesse tipo migração.

A aplicação do roteiro ajuda na garantia de qualidade da migração devido a completude dos itens que devem ser considerados para garantir que a migração para nuvem realmente seja realizada de forma adequada, medida e monitorada, garantindo a aderência aos ganhos esperados pela nuvem, de escalabilidade, flexibilidade, agilidade de alterações e inclusão de novas funcionalidades conforme apresentado na seção 4.1 deste trabalho.

O roteiro desenvolvido e apresentado nesse trabalho, servirá como um guia para os interessados em relação a migrações onde envolvem processos de reengenharia de sistema afim de adaptação ao ambiente em nuvem, utilizando os recursos da *Google Cloud Platform*, que apesar de serem específicos da Google, podem ser encontrados em outros provedores de nuvem recursos similares nos quais esse roteiro pode ser utilizado também como base a qualquer migração envolvendo processos de reengenharia SaaS.

## 5.2. Trabalhos Futuros

Como trabalhos futuros há oportunidade em desenvolver um detalhamento do roteiro quanto ao processo de desenvolvimento em ambientes de nuvem, incluindo os frameworks de desenvolvimento de aplicações na nuvem, configuração de ambiente de desenvolvimento e entrega contínua, conteinerização, portabilidade de aplicações em nuvem e automação de testes em ambientes de nuvem.

# REFERÊNCIAS

**ALTHANI, B.; KHADDAJ S. *"Systematic review of legacy system migration". Kingston University***, 2017

**AVRAM, A.; MARINESCU, F.;** *"Domain-Driven Design Quickly"*. 1. ed. [S.l.]: InfoQ, 2006. 104 p.

**BENNETT, K.; "***Legacy Systems: Coping with Success*". *IEEE*, 1994.

**BISBAL, J.; LAWLESS D.; WU, B.; GRIMSON, J**. "*Legacy Information System Migration: A Brief Review of Problems, Solutions and Research Issues". Computer Science Department,* 1999*.*

**CHERVENSKI, A.; MARTINS, D.; BORDIN, A.** "Mapeamento de Características de Sistemas Legados". Universidade Federal do Pampa, 2016.

**DEAN, J.; GHEMAWAT S. "***MapReduce: Simplified Data Processing on Large Clusters*". *Google, Inc.,* 2004

**EMBLEY, D. W.; THALHEIM, B.** "*Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges." Springer*, 2011.

**GOOGLE 1.** "Documentação *Google Cloud Platform: Architecture: Complex Event Processing"*. Google, 2018. Disponível em: https://cloud.google.com/solutions/architecture/complex-event-processing.

**GOOGLE 2.** "Documentação *Google Cloud Platform: Cloud Dataflow*". Google, 2018. Disponível em: https://cloud.google.com/dataflow/.

**GOOGLE 3.** "Documentação *Google Cloud Platform: Cloud Functions Overview*". Google, 2019. Disponível em: https://cloud.google.com/docs/overview/?hl=pt-br.

**GOOGLE 4.** "Documentação *Google Cloud Platform: Cloud Pub/Sub: A Google-Scale Messaging Service*". Google, 2018. Disponível em: https://cloud.google.com/pubsub/architecture.

**GOOGLE 5.** "Documentação *Google Cloud Platform: Cloud Storage documentation*". Google, 2018. Disponível em: https://cloud.google.com/storage/docs/.

**GOOGLE 6.** "Documentação *Google Cloud Platform: Compute Engine - Concepts*". Google, 2018. Disponível em: https://cloud.google.com/compute/docs/concepts.

**GOOGLE 7.** "Documentação *Google Cloud Platform: Containers on Compute Engine"*. Google, 2018. Disponível em: https://cloud.google.com/compute/docs/containers/.

**GOOGLE 8.** “Documentação *Google Cloud Platform: GKE Overview*”. Google, 2018. Disponível em: https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview.

**GOOGLE 9.** “Documentação *Google Cloud Platform: Google App Engine*”. Google. Disponível em: https://cloud.google.com/appengine/?hl=pt-br.

**GOOGLE 10.** “Documentação *Google Cloud Platform: Istio on GKE*”. Google, 2018. Disponível em: https://cloud.google.com/istio/docs/istio-on-gke/overview

**GOOGLE 11.** “Documentação *Google Cloud Platform: Java on Google App Engine*”. Google, 2018. Disponível em: https://cloud.google.com/appengine/docs/java/.

**GOOGLE 12.** “Documentação *Google Cloud Platform: Visão Geral da Plataforma*”. Google, 2018. Disponível em: https://cloud.google.com/docs/overview/?hl=pt-br.

**HAILPERN, B.** “*Multiparadigm Languages*”. IEEE, 1986.

**MELL, P.; GRANCE, T.;** “*The NIST Definition of Cloud Computing”. NIST Special Publication, 2011.*

**NADAREISHVILI, I.; MITRA, R.; AMUNDSEN, M.** “*Microservice Architecture*”. O’Reilly, 2016.

**NEWMAN, S.** “*Building Microservices*”. O’Reilly, 2015.

**POWER, B.** “*Digital Transformation Through SaaS Multiclouds”. IEEE Cloud Computing*, 2018.

**RICHARD, M. “***Microservices vs. service-oriented architecture*”. O’Reilly, 2016. Disponível em: https://www.oreilly.com/learning/microservices-vs-service-oriented-architecture.

**SHAIEF, A.** “*Critical Success Factors for the Migration of Legacy Information Systems to SOA". Delft University of Technology*, 2011.

**SIMMONDS, D.** “*The Programming Paradigm Evolution*”. IEEE, 2012.

**UMAR, A.** “*Application Reengineering – Building Web-Based Applications and Dealing With Legacy”. Prentice Hall*, 1997.

**WARREN, I.; RANSOM, J**. “*Renaissance: A Method to Support Software System Evolution*”. IEEE, 2002.

**WU, A.** “*Running Your Modern .NET Application on Kubernetes”*, 2018

**ZHAO, JF. ; ZHOU, JT.** *“Strategies and Methods for Cloud Migration”. International Journal of Automation and Computing*, 2014.

# APÊNDICE 1 – Questionário de Validação do Roteiro

Este apêndice foi desenvolvido pelo autor, utilizando a plataforma Google Formulários.

Questionário - Roteiro de Migração de Sistemas para a Nuvem utilizando a Google Cloud Platform

Esse questionário será utilizado no Trabalho de Conclusão de Curso de MBA da USP para identificar quais os passos do Roteiro de Migração proposto, são mais fáceis/difíceis e complexos. O resultado do questionário será incluído nesse trabalho de forma sigilosa, portanto nenhum dado pessoal será divulgado.

*Obrigatório

1. Dados os passos do Roteiro em sua opinião quais encontrou maior dificuldade para implementação? * *Marque todas que se aplicam.*

   1- Seleção do time de migração
   2- Definição da estratégia de migração
   3- Definição do Domínio do legado
   4- Abstração dos Módulos
   5- Definição da Arquitetura
   6- Configuração do Ambiente e Recursos
   7- Definição da Comunicação (filas/tópicos/arquivos)
   8- Construção das aplicações (microserviços)
   9- Configuração do Monitoramento
   10- Gerenciamento da aplicação em nuvem (microserviços)

2. Quais encontrou maior facilidade? * *Marque todas que se aplicam.*

   1- Seleção do time de migração
   2 - Definição da estratégia de migração
   3 - Definição do Domínio do legado
   4 - Abstração dos Módulos
   5 - Definição da Arquitetura
   6 - Configuração do Ambiente e Recursos
   7 - Definição da Comunicação (filas/tópicos/arquivos)

8 - Construção das aplicações (microserviços)

9 - Configuração do Monitoramento

10 - Gerenciamento da aplicação em nuvem (microserviços)

3. Dos passos onde encontrou maior dificuldade, classifique de 0 a 5, onde 0 indica baixa complexidade e 5 alta complexidade * *Marcar apenas uma opção.*

0 1 2 3 4 5

4. Há algum passo que gostaria de acrescentar em relação a migração de sistemas legados para a nuvem? Descreva abaixo:

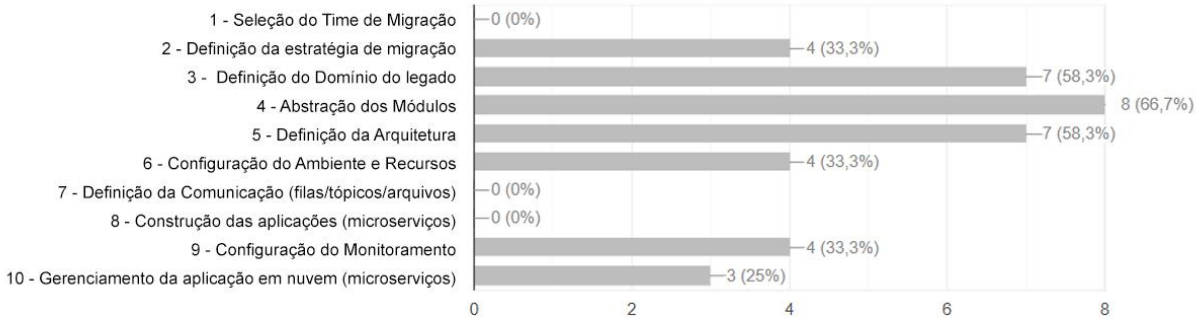
5. Na sua opinião, o roteiro apresentado ajuda a guiar equipes que não conhecem sobre migração de sistemas para nuvem? * *Marcar apenas uma opção.*

- ◯ Sim
- ◯ Não

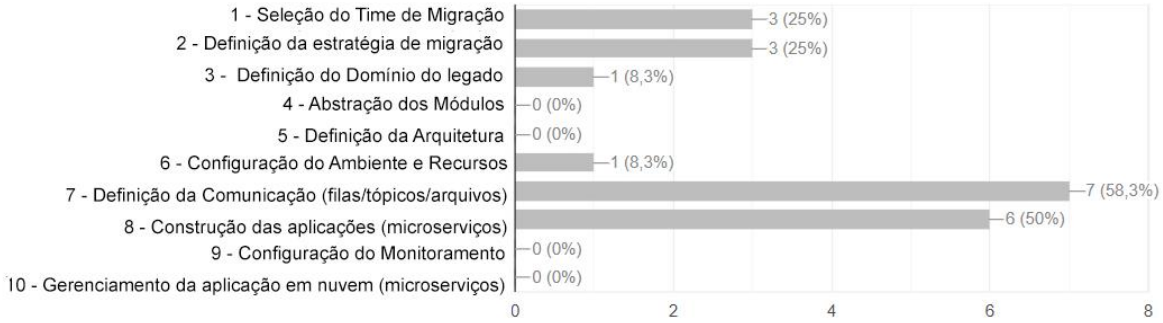
# APÊNDICE 2 - Respostas do Questionário de Validação do Roteiro

Este apêndice foi elaborado pelo autor.

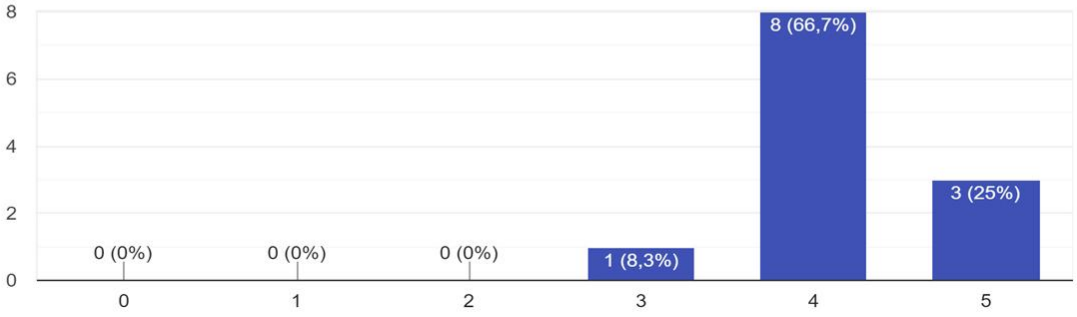
Dados os passos do Roteiro em sua opinião quais encontrou maior dificuldade para implementação?




Quais encontrou maior facilidade?




Dos passos onde encontrou maior dificuldade, classifique de 0 a 5, onde 0 indica baixa complexidade e 5 alta complexidade



Há algum passo que gostaria de acrescentar em relação a migração de sistemas legados para a nuvem? Descreva abaixo:

| |
|---|
| Automação de Testes Regressivos, Funcionais e Homologação |
| Testes |
| Dependendo da migração, o ponto mais complicado é a estratégia de convivência do software migrado com o legado restante |
| Análise de viabilidade |

Na sua opinião, o roteiro apresentado ajuda a guiar equipes que não conhecem sobre migração de sistemas para nuvem?

12 respostas