UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PAULO GUILHERME SANTIAGO

MIGPLAN: PROCESSO APLICADO NA RENTABILIDADE EMPRESARIAL ATRAVÉS DE REENGENHARIA DE SISTEMAS.

DOIS VIZINHOS

2022

PAULO GUILHERME SANTIAGO

MIGPLAN: PROCESSO APLICADO NA RENTABILIDADE EMPRESARIAL ATRAVÉS DE REENGENHARIA DE SISTEMAS.

MIGPLAN: PROCESS APPLIED TO BUSINESS PROFITABILITY THROUGH LEGACY SYSTEMS REENGINEERING

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Software do Curso de Bacharelado em Engenharia de Software da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Gustavo Jansen de Souza Santos

DOIS VIZINHOS
2022

PAULO GUILHERME SANTIAGO

MIGPLAN: PROCESSO APLICADO NA RENTABILIDADE EMPRESARIAL ATRAVÉS DE REENGENHARIA DE SISTEMAS.

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Software do Curso de Bacharelado em Engenharia de Software da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 24/junho/2022

Gustavo Jansen de Souza Santos
doutorado
Universidade Tecnológica Federal do Paraná

André Roberto Ortoncelli
doutorado
Universidade Tecnológica Federal do Paraná

Rafael Alves Paes de Oliveira
doutorado
Universidade Tecnológica Federal do Paraná

DOIS VIZINHOS
2022

Ao meu querido e eterno Pai.

AGRADECIMENTOS

Agradeço imensamente a minha família, aos meus amigos, ao meu orientador e a todos os demais que participaram de alguma forma para a minha formação.

Obrigado por tudo.

RESUMO

Sistemas legados sofrem de diversos problemas nos cenários de produção atual. Por exemplo, o alto custo de manutenção de servidores e a necessidade de crescimento no mercado. Estratégias de reengenharia e modernização desses sistemas são necessárias. O objetivo deste trabalho de conclusão de curso foi a criação de um processo para melhorar a rentabilidade de empresas que utilizam sistemas legados por meio de reengenharia da arquitetura desses sistemas. Para atingir esse objetivo, o processo definiu uma nova arquitetura que utiliza os conceitos de computação em nuvem. Com isso, foi efetuado o uso de um novo fluxo de transmissão de dados que possibilita a operabilidade em contigência em casos de erros ou desligamento do servidor. Para validar o processo, foi realizado um estudo de caso que executou a migração de uma funcionalidade legada para uma funcionalidade moderna por meio de abordagens e técnicas de reengenharia. As abordagens estruturaram como deveria ocorrer a migração, visando agilidade, segurança e performance. As técnicas foram utilizadas para extração do conhecimento necessário para realizar a migração e colocar em prática todas as abordagens. Como resultado, neste trabalho, foi obtido que o processo reduziu o custo de inatividade para o usuário final da empresa.

Palavras-chave: reengenharia de software; microsserviços; computação em nuvem; migração.

ABSTRACT

Legacy systems suffer from several problems in current production scenarios. For example, the high cost of server maintenance and the need for growth in the market. Strategies to reengineer and modernize these systems are necessary. The objective of this work was to create a process to improve the profitability of companies that use legacy systems through the reengineering of the architecture of these systems. To achieve this goal, the process defined a new architecture that uses concepts of cloud computing. This new architecture provides the use of a new data transmission stream that enables operability in a contingency in case of errors or server shutdown. To validate the process, a case study was proposed, with the goal to perform the migration of a legacy functionality for modern functionality through approaches and techniques. The approaches structured how the migration should occur, aiming at agility, security, and performance. The techniques were used to extract the knowledge necessary to carry out the migration and put all approaches into practice. As a result, in this work, we concluded that the process reduced the cost of downtime for the business end user.

Keywords: software reengineering; microservices; cloud computing; migration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

SUMÁRIO

1 INTRODUÇÃO

De acordo com Fowler (2014), os serviços são aplicações acopladas com objetivos distintos e independentes que são requisitados conforme as solicitações de uma Interface de Programação de Aplicações (do inglês, *Application Programming Interface, API*); estes métodos e serviços são utilizados para construção de funcionalidades entre um único ou demais sistemas.

Com o passar do tempo, houve a necessidade de padronizar a arquitetura desses serviços, pois diversos pesquisadores e profissionais começaram a interpretar os serviços de diferentes modos, e a propor novos meios de desenvolvimento de acordo com suas próprias abordagens de uso, deturpando o conceito e o contexto pelo quais os serviços são implementados (RAZAVIAN; LAGO, 2015). Desta necessidade, surgiu a Arquitetura Orientada a Serviços (do inglês, *Service-Oriented Architecture, SOA*) que permite o planejamento de estratégias do projeto, desde a fase inicial dos requisitos, do desenvolvimento, das validações e da implantação de serviços que auxiliam aplicativos a ter maior flexibilidade e mutabilidade (CIGANEK; HAINES; HASEMAN, 2005; KUMAR; DAKSHINAMOORTHY; KRISHNAN, 2007; RAZAVIAN; LAGO, 2015).

A adoção do SOA permite que os serviços atuem em sistemas legados, em outras palavras, sistemas antigos, onde tem-se tecnologias obsoletas e de dificil manutenção, necessitando modernizar suas funcionalidades devido ao avanço tecnológico. Dessa forma, reutilizando as funcionalidades existentes dos sistemas para construções de aplicações com maior agilidade e com menor custo operacional (RAZAVIAN; LAGO, 2015).

Uma amostra de sucesso de implementação do SOA é o banco ABN AMRO Brasil que, após realizar a sua migração para o SOA, percebeu como resultado, aplicações menos complexas e com maior otimização de tempo (SORDI; MARINHO; NAGY, 2006). Conforme descreve Bielski (2005), adotar essa estratégia fez com que a ABN AMRO Brasil tivesse evoluções tecnológicas obtendo como produto final um sistema mais amplo para novas integrações de sistemas de software na nova arquitetura desenvolvida.

Entretanto, manter serviços rodando em servidores na escala 24x7 (24 horas do dia durante toda a semana) afeta diretamente a rentabilidade das empresas, pois manter um *Data Center* funcionando gasta em média 38% do custo total em energia das empresas; todos esses fatores ocorrendo em um lugar físico da empresa que precisa de manutenção, agregando assim custos adicionais (PONEMON, 2016).

Tradicionalmente, diante deste alto custo, é realizado o desligamento dos servidores do *Data Center* após o horário comercial para fins de economia de energia. A partir desse problema surgiu um novo: toda e qualquer aplicação é encerrada juntamente ao desligamento do servidor. Com tais problemas, as empresas não conseguem acompanhar a grande demanda no mercado, isto é, diferentes nichos de público alvo que poderiam ser explorados para realização de novos produtos são descartados pela falta de tecnologia ou investimento necessário (PONEMON, 2016).

Neste trabalho é proposta uma estratégia de reengenharia de aplicações tendo como finalidade a criação ou reuso de funcionalidades das empresas sem precisar de conexão direta com seus servidores. Pretende-se amenizar problemas de custo, manutenção e investimento em *Data Center*, possibilitando a criação de sistemas com transação de dados em tempo real e a disponibilidade da aplicação *on-line* na escala 24x7 para ser acessada e utilizada de qualquer lugar. Para a demonstração prática de que os desafios referentes à utilização desta estratégia foram sanados, será construído um aplicativo *Mobile* que envolve a integração de tecnologias e conceitos recentes, sendo elas a Computação em Nuvem (do inglês, *Cloud Computing, CC*) e a Computação em Nuvem Móvel (do inglês, *Mobile Cloud Computing, MCC*).

1.1 Motivação

A motivação deste trabalho teve início em uma necessidade real, envolvendo uma empresa do sudoeste do Paraná; um dos fatores pelo qual a mesma não consegue evoluir e acompanhar o crescimento do mercado é o sistema legado. Esse fator acontece pela arquitetura de sistema atual implantada, pelos custos de energia para se manter servidores em execução na escala 24x7 e pelas aplicações que necessitam da execução desses servidores para funcionar.

Esses problemas impactam em dois lados, na empresa de software que perde clientes por não ter tecnologias que atenda as demandas solicitadas e os clientes da empresa de software, isto é, os usuários do software, que não conseguem ter tecnologias necessárias para expandir sua lucratividade.

Outro agravante são os erros que ocorrem nos servidores da empresa de software e nos servidores do cliente da empresa, que podem resultar em inatividade de ambos. Portanto, esse trabalho é um esforço da academia em resolver problemas da indústria, mais especificamente as fábricas de software regionais, mas que tem uma aplicação para empresas que passem pelo mesmo problema.

1.2 Objetivos

Conforme a UNISC (2020), a rentabilidade é o retorno sobre o investimento. Este trabalho tem como objetivo geral a criação de um processo focado em melhorar a rentabilidade de empresas que desenvolvem e de empresas que utilizam sistemas legados através de reengenharia desses sistemas. Dentre os pontos de rentabilidade abordados neste trabalho, estão:

- **O retorno sobre investimento em servidores das empresas fornecedoras de software**, que aborda os custos gerais na utilização de servidores nas empresas.
- **O retorno sobre investimento das empresas clientes nos softwares da empresa fornecedora**, que aborda os custos que o usuário final enfrenta devido a erros inesperados ou desligamentos nos servidores.
- **O retorno sobre investimento após o uso do processo do MIGPLAN**, que aborda os custos que foram reduzidos após o uso do MIGPLAN.

Como objetivos específicos, é proposta a construção de um aplicativo *Mobile* que utiliza este processo, respectivamente, o levantamento de tecnologias atuais para essa aplicação. Para concluir tais objetivos foram designadas as seguintes atividades:

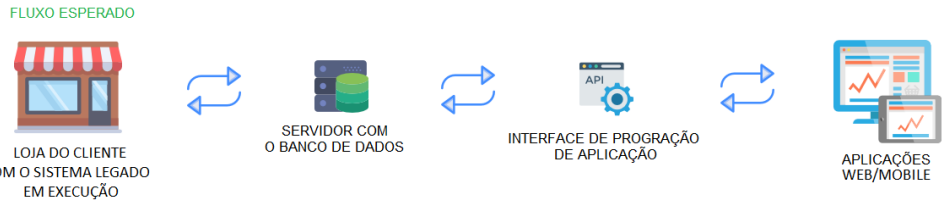
- **Seleção de metodologia para migração**.
- **Seleção de biblioteca para a Computação em Nuvem**.
- **Seleção de biblioteca *Mobile***.
- **Seleção de abordagens para migração**.
- **Elaboração da arquitetura**.
- **Desenvolvimento de serviços e microsserviços**.
- **Desenvolvimento da aplicação *Mobile***.
- **Verificação e Validação da aplicação**.
- **Documentação Arquitetural**.
- **Documentação do *processo***.
- **Avaliação Experimental**.

Com base nos problemas a serem abordados foi modelado um fluxo de dados, visando solucionar tais falhas de arquitetura, sem grande impacto de alterações nas empresas, isto por meio do levantamento e análise de técnicas para realizar a migração entre aplicações e as abordagens necessárias para a migração.

2 CONTEXTO

Para demonstrar o fluxo esperado em uma arquitetura que depende de um servidor integrado pode ser visualizado a Figura 1, onde este fluxo baseia-se em uma comunicação sem erro, onde tem-se a loja do cliente que utiliza o sistema principal, como exemplo, um Sistema Integrado de Gestão Empresarial (do inglês, *Enterprise Resource Planning, ERP*) bastante utilizado em empresas de gestão comercial e conhecido como software de retaguarda.

Figura 1 – Fluxo esperado da transmissão de dados entre aplicações de sistemas legados



Fonte: Autoria própria.

O ERP é conectado com um servidor que armazena o banco de dados da loja do cliente, isto é, todos os dados que são gerados pelo cliente durantes dias, meses ou anos, como cadastros gerais, movimentações financeiras e notas fiscais, são armazenados em um único lugar. Uma vez que o cliente decida expandir suas vendas para aumentar seu lucro, este deve recorrer a vendas *on-line* em plataformas *Web* ou *Mobile* por razões de custo-benefício em relação às lojas físicas.

Tradicionalmente, sistemas legados não são capazes de suportar e processar estruturas *on-line*. Para resolução de tal problema, na maioria dos casos, uma ponte de conexão para comunicação é construída. Essa ponte é conhecida *API*, utilizada para conectar o sistema principal a outras aplicações de diferentes plataformas.

Como exemplo, uma transação comum de dados que ocorre nessa arquitetura é a venda de produtos; ou seja, uma pessoa compra produtos na loja *on-line* do cliente, essa venda é contabilizada no banco de dados por meio da API, e é gerada automaticamente a nota fiscal do produto no sistema principal.

No entanto, erros podem ocorrer no servidor que mantêm o banco de dados, ou a loja do cliente decide desligar o servidor para fins de economia de energia. Esses fatores acarretam o fluxo não esperado, representado na Figura 2, onde a API irá perder a conexão com os dados da loja, não realizando operações de comunicação, tais como *request* e *response*; consequentemente, ocasionando erros nas aplicações *on-line* até a inatividade das mesmas.

Portanto, uma solução para estes problemas é a reengenharia dessa arquitetura de sistemas legados por meio de uma arquitetura orientada a serviços, ou seja, um fluxo para uma saída estretégica que possibilita a reutilização dos códigos existentes e de fácil manutenção, para melhor entedimento pode ser visualizado o Capítulo 5 que demonstra o estudo de caso do trabalho.

Figura 2 – Fluxo não esperado da transmissão de dados entre aplicações de sistemas legados

Fonte: Autoria própria.

3 ASPECTOS CONCEITUAIS

Este capítulo apresenta os conceitos fundamentais para o entendimento do *processo* proposto neste trabalho, contextualizando as principais caracteristicas para seu funcionamento e sua implementação.

3.1 Estrutura de custos

Como citado no Capítulo 2, os problemas enfrentados dos erros inesperados e de desligamento do servidor são apenas exemplos de uma gama de custos gerados. Desde 2011, o Instituto Ponemon (2016) realiza estudos sobre os custos gerais da utilização de *Data Center* nas empresas. Esses estudos revelaram que o custo médio para o uso dos servidores aumentou de R$ 2.106.224,63 em 2010 para R$ 3.084.771,48 em 2016.

O levantamento dos dados foi realizado a partir de pesquisas de custo em áreas problemáticas, tais como o impacto do tempo de inatividade sobre as aplicações, o custo para detectar e corrigir processos não esperados e os danos em equipamentos e outros ativos do *Data Center*. Após analisar os custos de atividades de cada área problemática, e baseado nas ações de resposta e seus gastos associados, obteve-se uma estrutura de custos, como pode ser observado abaixo:

- **Detecção**: Tem como custo de atividade o tempo até a descoberta inicial e investigação sobre o erro ocorrido no servidor.
- **Contenção**: Tem como custo de atividade o tempo para realizar uma estratégia de contenção para controle do erro, impedindo que ele se espalhe, agrave ou cause maiores pertubações.
- **Recuperação**: Tem como custo de atividade o tempo para reorganização das redes e sistemas principais para o estado estável após o erro ter ocorrido.
- **Resposta**: Tem como custo de atividade o tempo de resposta das redes e sistemas principais para o estado estável após o erro ter ocorrido.
- **Equipamento**: Tem como custo de atividade o gasto de compras e reparos de novos equipamentos, incluindo reformas.
- **Produtividade (TI)**: Tem como custo de atividade o tempo de inatividade e as despesas relacionadas aos erros e manutenções no servidor.
- **Produtividade (Usuário final)**: Tem como custo de atividade o tempo de inatividade do usuário final perante ao não funcionamento do servidor.

- **Terceiros**: Tem como custo de atividade o gasto com contratados, consultores, auditores e outros especialistas engajados para ajudar a resolver os problemas não planejados.

3.2 Reengenharia de software

A reengenharia de software é um processo de mudança, que busca novos métodos e processos para serem utilizados pelas empresas e é caracterizada por agrupar uma série de atividades e eliminar outras que não agregam valor. Isto ocorre quando produtos ou serviços não se adequeam às necessidades dos clientes. As grandes empresas são as que mais têm interesse pelos processos de reengenharia, pois essa necessidade ocorre quando existe algum nível de estrutura estabelecida (FERNANDES; BERTOLLO, 1999).

A reengenharia de software e engenharia reversa são as maneiras que muitas empresas buscam para reconstruir ou manter seus softwares, conseguindo solucionar problemas como ter que fazer manutenções complexas e da inutilização de suas estruturas. Entretando, conforme Pressman e Maxim (2016) a reengenharia é um processo que ocupa tempo, tem um custo significativo em dinheiro e absorve recursos que poderiam ser aplicados em outras necessidades, por estes motivos a reengenharia não é realizada em alguns meses, é um processo que ocorrerá por muitos anos.

3.3 Engenharia reversa

De um modo geral, a engenharia reversa é o processo de descobrir informações sobre um hardware, software ou sistema por meio da análise de sua funcionalidade, estrutura, entre outros. Geralmente a engenharia reversa envolve desmontar e analisar algum artefato, estudando em detalhes sobre o seu funcionamento e então tentar recriar esse mesmo artefato, mesmo sem entender os processos envolvidos no desenvolvimento do qualquer parte do original.

Portanto, a engenharia reversa para software é o processo para analisar um programa na tentativa de criar uma representação do programa em um nível mais alto de abstração. Como nenhuma especificação foi desenvolvida ou está desatualizada, é responsabilidade desta engenharia entender os processos e assim obter a recuperação do projeto (PRESSMAN; MAXIM, 2016).

Uma derivação desta engenharia é a engenharia reversa dos dados que ocorre em diferentes níveis de abstração. Em nível de programa, as estruturas internas de dados repassam diversas vezes pela engenharia reversa para obter um reengenharia total do programa. Em nível de sistema, as estruturas de dados globais, (e.g., arquivos, bases de dados) repassam muitas vezes pela reengenharia reversa para obter novos paradigmas (PRESSMAN; MAXIM, 2016).

3.4 Serviços e Microsserviços

Os microsserviços têm como abordagem o desenvolvimento de aplicações utilizando um conjunto de pequenos serviços, onde cada um executa seu próprio processo e se comunicando na maioria das vezes por uma API. Esses serviços são construídos em torno das regras de negócio de cada empresa, e devem ter o mínimo possível de gerenciamento centralizado, pois os mesmos podem ser escritos em diferentes linguagens de programação e utilizar diferentes tecnologias de armazenamento de dados (FOWLER; LEWIS, 2015).

O serviço em si pode ser implantado, dimensionado e testado de forma independente. Um serviço possui apenas uma única responsibilidade, isto é, realizar um único processo, para assim, garantir integridade e fácil manutenção (THÖNES, 2015). Um exemplo de execução de um serviço ou de microsserviços é um processador de filas, no qual um único serviço está lendo uma mensagem de uma fila, executando uma pequena parte da lógica de negócios e depois transmitindo-a. Nesta organização, estes processos podem ser alocados em pequenos serviços (microsserviços). Entretanto, os processos devem sempre atender aos requisitos, sejam eles funcionais, não-funcionais ou multifuncionais (THÖNES, 2015).

3.5 Arquitetura Orientada a Serviços

Conforme Bass, Clements e Kazman (2003), o padrão SOA descreve um conjunto de componentes que fornecem e consomem serviços. Em uma SOA, os componentes e serviços do provedor de serviços e os componentes do consumidor podem usar diferentes linguagens e plataformas de implementação. Os serviços são altamente autônomos, e.g., provedores de serviços e consumidores de serviços, e geralmente são implantados de forma independente e pertencem a diferentes sistemas ou até diferentes organizações.

Os componentes possuem interfaces que descrevem os serviços que os mesmos solicitam de outros componentes. A qualidade de um serviço pode ser especificada e garantida com um Acordo de Nível de Serviço (do inglês, *Service Level Agreement, SLA*). Os componentes conseguem sua computação solicitando serviços um do outro.

Além do provedor de serviços e dos componentes do consumidor de serviços, um aplicação que utiliza a SOA pode usar componentes especializados que atuam como intermediários e fornecem serviços de infraestrutura, como pode ser visualizado abaixo:

- A chamada de serviço pode ser mediada por um Barramento de Serviço da Empresa (do inglês, *Enterprise Service Bus, ESB*). Rotas ESB conduzem as mensagens entre consumidores e prestadores de serviços. Além disso, um ESB pode converter mensagens de um protocolo ou tecnologia para outro, executar várias transformações de dados, (e.g., formato, conteúdo, divisão, mesclagem) e realizar as verificações de se-

gurança e gerenciar transações. Quando não existe um ESB, os prestadores de serviços e os consumidores se comunicam uns com os outros de maneira ponto a ponto.

- Para melhorar a independência dos provedores de serviços, um registro de serviço pode ser usado na SOA. O registro é um componente que permite que os serviços sejam registrados em tempo de execução. Este permite a descoberta de serviços em tempo de execução, o que aumenta a flexibilidade do sistema ocultando a localização e identidade do provedor de serviços. Um registro pode até permitir várias versões em tempo real do mesmo serviço.

Os tipos básicos de conexão usado em SOA são os seguintes:

- O *SOAP* é protocolo padrão para comunicação na tecnologia de serviços da web. Tais serviços, consumidores e provedores interagem trocando mensagens XML de solicitação e de resposta.
- No *REST*, um consumidor de serviço envia HTTP sem bloqueio de solicitações. Essas solicitações dependem dos quatro comandos HTTP básicos (POST, GET, PUT, DELETE) para instruir o provedor de serviços a criar, recuperar, atualizar ou excluir um recurso.
- Mensagens assíncronas: Fazem uma troca de informações. Os componentes não precisam aguardar um aviso de recebimento, porque presume-se que a infraestrutura tenha entregue a mensagem com sucesso.

3.6 Computação em Nuvem

O CC oferece um meio para acesso de servidores, armazenamento de arquivos, banco de dados e outros demais serviços de aplicativos, via Internet. Cada serviço em nuvem e método de implantação do mesmo oferece diversos diferentes níveis de controle, que oferecem flexibilidade e gerencimento (AMAZON, 2019).

Uma aplicação baseada em nuvem é totalmente implantada na nuvem, contendo todos os aspectos da aplicação. Para realizar essa aplicação existem diferentes estratégias de implantação que podem ser utilizadas para propor o conjunto de serviços ideal. Tais estratégias são:

- **Infraestrutura como um serviço (IaaS)**: Basicamente formada por componentes básicos em nuvem, permite acesso a recursos de rede e computadores, assim como, armazenamento de dados.
- **Plataforma como um serviço (PaaS)**: Gerencia a infraestrutura da empresa, isto é, o hardware e os sistemas operacionais, permitindo foco da empresa somente na implantação e no gerenciamento das aplicações.

- **Software como um serviço (SaaS)**: Dentre as estratégias, a mais completa, pois executa e gerencia pelo provedor de serviços, que na maioria dos casos, são as aplicações para o usuário final.

Portanto, o CC permite que os usuários utilizem os recursos sob demanda. Como resultado, os aplicativos podem ser rapidamente fornecidos e liberados com o mínimo esforço de gerenciamento e de interações do provedor de serviços (FERNANDO; LOKE; RAHAYU, 2013).

3.7 Computação em Nuvem Móvel

Com o surgimento dos aplicativos móveis e o suporte base da CC, há uma variedade de serviços para usuários móveis. Como exemplo, a MCC é introduzida como uma integração do CC para o ambiente móvel. Isto reflete a novos tipos de serviços e instalações para usuários móveis para aproveitar ao máximo a CC.

Como citado por Fernando, Loke e Rahayu (2013), os dispositivos móveis estão se tornando cada vez mais uma parte essencial da vida humana como as ferramentas de comunicação mais eficazes e convenientes, não limitadas por tempo e local. Esses usuários móveis acumulam uma rica experiência de vários serviços de aplicativos móveis, que são executados nos dispositivos ou em servidores remotos por meio de redes sem fio. Este rápido progresso de evolução se torna uma grande tendência de desenvolvimento para a área tecnológica, bem como para os campos do comércio e da indústria.

4 REVISÃO NARRATIVA

Com a finalidade de fundamentar a base teórica e conceitual proposto neste trabalho, foi desenvolvida uma Revisão Narrativa para realizar o levantamento e análise de técnicas para migração entre aplicações e respectivamente, a migração dos dados existentes. A Revisão Narrativa é apropriada para descrever o desenvolvimento de um determinado assunto, utilizando o ponto de vista teórico ou contextual (ROTHER, 2007).

Este capítulo foi constituído no seguinte modelo: Na Seção 4.1 é relatado o contexto de como foi desenvolvida a revisão e levantados os conteúdos da base teórica e conceitual do trabalho, na Seção 4.2 é relatada as abordagens para migração de dados encontradas, na Seção 4.3 foram relatado o levantamento das técnicas encontradas, na Seção 4.4 foram descritos os resultados obtidos e na Seção 4.5 foram discutidas as considerações finais.

4.1 Contexto

Para critério de estudo e avaliação foi utilizada a revisão sistemática desenvolvida por Razavian e Lago (2015), que por meio de 75 estudos primários sobre estratégias de migração para o SOA obtiveram oito categorias de abordagens de migração, 21 categorias de elementos de conhecimento que moldam e conduzem a migração e dez categorias de conversões de conhecimento.

Para foco de desenvolvimento foi realizado um filtro aos assuntos de interesse correspondentes ao que é proposto por este trabalho, isto é, contruir a base teórica e conceitual do *processo*. Dentre as oito categorias de abordagens foram filtradas somente cinco, formando assim, as abordagens para migração que podem ser visualizadas na Seção 4.2. De 21 categorias de elementos de conhecimento foram filtradas somente duas e das dez categorias de conversões de conhecimento foi filtrada apenas uma; juntas formaram as técnicas para migração que podem ser visualizadas na Seção 4.3.

Também obteve-se como resultado duas visões de estado, isto é, a condição atual e os objetivos da empresa. A primeira visão é o estado ou condição atual da empresa e que corresponde ao nível de conhecimento que está disponível sobre a arquitetura atual do sistema legado. Por exemplo, a Figura 1 mostrada no Capítulo 2, demonstra este estado. A segunda é visão é referente ao estado ou condição do futuro e que corresponde ao objetivo da realização da migração à longo prazo. Como exemplo, a Figura 6 que será apresentada no Capítulo 5, demonstrando a nova organização do sistema utilizando serviços e computação em nuvem.

4.2 Abordagens para migração

As abordagens para migração são essenciais às empresas que precisam se manter competitivas no mercado. As abordagens estruturam como deve acontecer a migração, garantindo a agilidade e seguranças das informações e mantendo a alta performance das novas aplicações. No contexto no qual este trabalho é aplicado, nestas mesmas abordagens, respectivamente é abordada a migração dos dados em segundo plano, isto é, os dados existentes da empresa serão transitados e capturados entre as aplicações pela nova arquitetura baseada em serviços.

Como na maioria dos sistemas legados, existem diversas funcionalidades que permitem ao usuário final realizar operações em um único software; isto faz com que o banco de dados do cliente seja muito valioso. Como citado no Capítulo 2, é no banco de dados que são armazenados todos os dados que são gerados durante a utilização do sistema pelo usuário final. Portanto, para não se ter a perda de todas estas funcionalidades e dados gerados, durante a fase inicial da migração até após o fim da mesma, não é necessario descontinuar o sistema legado existente.

Após realizado o filtro das abordagens pertinentes a este trabalho, obteve-se como resultado as seguintes abordagens para migração de dados: Transformação de código (A1); Identificação de serviço (A2); Transformação de elementos de design (A3); Engenharia direta (A4) e Composição de padrões de projeto (A5).

- **A1**: Limitada à transformação em nível do sistema. O código legado existente é transformado em implantações baseadas em serviços, a migração propõe mover o sistema legado como um todo para um serviço, plataforma ou tecnologia. Foram identificadas duas principais categorias, A1.1 e A1.2:

  A1.1: Consiste em reconstruir inteiramente o sistema legado para uma aplicação web através de implantações de diversos serviços, solucionando o problema da necessidade de entender todo um código legado para realizar a migração. Esta abordagem é bastante similar à abordagem Big-Bang de Bass, Clements e Kazman (2003), isto é, ambas propõem reescrever todo o sistema.

  A1.2: Consiste em fazer adaptações para um sistema web ou mobile utilizando as pequenas implantações de serviços e encapsulamento das principais funcionalidades do sistema legado. Esta abordagem é bastante similar à abordagem Incremental de Bass, Clements e Kazman (2003), ou seja, o processo é realizado a partir de pequenos incrementos.

- **A2**: A migração limita-se à identificação dos serviços candidatos no sistema legado existente. Essa abordagem utiliza atividades de engenharia reversa para identificar os elementos que são candidatos à migração para SOA, isto é, este é um processo de identificação e modelagem de serviços e não para efetuar alguma modificação da aplicação existente.

- **A3**: O processo de transformação ocorre apenas no nível conceitual do elemento básico de design, e.g, classes e interfaces. A migração desta abordagem limita-se a reformular os elementos legados existentes para elementos baseados em serviço.

- **A4**: Esta abordagem abrange totalmente o processo de engenharia direta, ou seja, a implantação dos serviços a partir de um modelo ou conceito de alto nível, com foco no desenvolvimento dos serviços através das regras de negócio.

- **A5**: A abordagem tem como característica usar uma composição dos padrões de projetos específicos da SOA para transformar a existente arquitetura para arquitetura baseada em serviços. Esses padrões são soluções reutilizáveis para extrair serviços ou facilitar transformações de elementos herdados em serviços. Como exemplo, a migração de um determinado componente do sistema legado para um padrão *REST*.

4.3 Técnicas para migração

A análise das técnicas são necessárias para colocar em prática as abordagens de migração de dados, e consistem basicamente na extração do conhecimento necessário para a realizar a migração. Após realizado o filtro obteve-se como resultado as seguintes ténicas: Conhecimento relacionado à solução (T1); Conhecimento relacionado a problemas (T2) e Conversões de conhecimento (T3).

- **T1**: Abrange todo o conhecimento da implementação de sistemas existentes, e é categorizada no conhecimento relacionado ao código (T1.1) e ao design (T1.2). Essa técnica é relacionada à primeira visão de estado ou condição; portanto, é necessario o conhecimento da arquitetura e do código-fonte e eventualmente conhecer todas as funcionalidades do sistema a ser migrado.

  T1.1: Utiliza o conhecimento sobre a implementação, i.e., do código-fonte do sistema existente para transforma-lo em novos serviços.

  T1.2: Utiliza o conhecimento para selecionar os elementos de design legados a serem migrados como um serviço.

- **T2**: Esta técnica pertence ao conhecimento do domínio sobre os possíveis problemas que vão surgir, e normalmente é interligado aos processos de negócios, regras de negócios, requisitos e cálculos de custo-benefício. Esta técnica é categorizada em um tipo principal de conhecimento, sendo ele: o domínio estrutural e comportamental (T2.1).

  T2.1: Obter os requisitos e atributos de domínio necessário para combater o problema, abordando sua funcionalidade e o seu comportamento. Para combater os problemas, é necessário ter o conhecimento dos processos de negócios, dos fluxos de tra-

balhos e de outros cenários que implicam na liberação das funcionalidades do sistema e na sua estrutura final.

- **T3**: Nessa técnica são investigados os conhecimentos coletados nas técnicas anteriores **T1** e **T2**, tendo como objetivo converter tais conhecimentos para elaborar uma estratégia alvo, isto é, a estratégia de migração.

  Após elaborada a estratégia, basicamente formada por problemas e soluções, é decidido como a mesma será priorizada e como será distribuída para as equipes envolvidas durante o processo de migração.

4.4 Resultados

Após indentificadas as abordagens e técnicas necessárias, foi construido um plano de migração específico para este trabalho, que consiste em quatro fases para realizar a migração de forma eficiente, segura e fluída. As fases são: Processo de definição do estado (F1); Processo de agregação (F2); Processo de desenvolvimento (F3) e o Processo de documentação (F4). A Figura 3 representa os principais pontos de ação de cada fase.

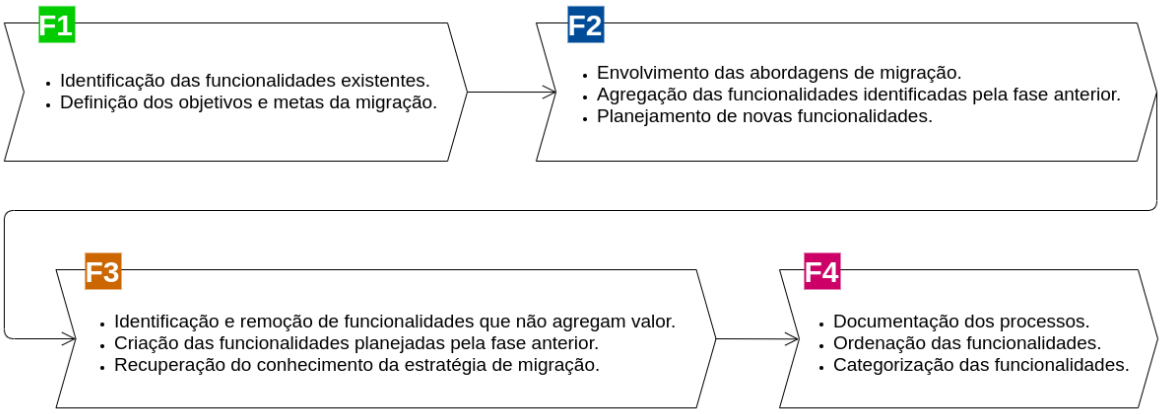
F1: Essa fase é responsável por definir o estado ou condição da arquitetura atual, visando reunir todas as funcionalidades existentes para assim realizar a migração. Outra responsabilidade é definir o estado ou condição do futuro, isto é, quais são os objetivos da empresa a longo prazo e definir metas a serem alcançadas pela migração.

F2: Nesta fase as abordagens são envolvidas na migração, recomenda-se para ao que é proposto neste trabalho utilizar a categoria da abordagem **A1.2**. São agregadas as funcionalidades identificadas pela fase **F1** e ao surgir algum novo requisito não previsto, é planejada a criação de novas funcionalidades. Isso ocorre porque inicialmente se uma funcionalidade que representa um requisito não existe é porque houve erro na recuperação da arquitetura, pois não considerou a transformação da funcionalidade.

F3: Essa fase é responsável por identificar e remover as funcionalidades que não agregam valor ao usuário final. As novas funcionalidades planejadas pela fase **F2** devem seguir a abordagem **A4** para sanar qualquer requisito pendente; para isto, é necessário realizar novamente a técnica **T3**, isto é, a recuperação do conhecimento da estratégia de migração, que se baseia nas técnicas **T1** e **T2**.

F4: É uma fase documental sobre todos os processos realizados nas fases anteriores, assim garantindo a integridade da migração. Esta fase é responsável por relatar a ordem em que as funcionalidades foram identificadas, criadas ou removidas, mantendo um histórico dos processos que foram realizados na migração. Outra responsabilidade desta fase é relacionar cada funcionalidade com uma categoria, *e.g.*, uma funcionalidade específica está na categoria de controle de estoque.

Figura 3 – Fases de Migração



Fonte: Autoria própria.

4.5 Considerações finais

Dentre os maiores desafios enfrentados nessa revisão narrativa está o entendimento sobre todos os processos necessários para a migração e o filtro aplicado aos assuntos de interesse correspondentes ao que é proposto neste trabalho. Entretanto, tais desafios irão auxiliar na elaboração da arquitetura final e na migração proposta nesse trabalho.

Portanto, a revisão narrativa proporcionou o conhecimento às abordagens necessárias para a migração e que também serão utilizadas para a transação e captação dos dados. Além disso, obteve-se a metodologia necessária para a migração por meio do levantamento e análise de técnicas aplicadas no estado da arte.

5 ESTUDO DE CASO

A reengenharia de software neste trabalho parte do objetivo de melhorar a rentabilidade empresarial, como relatado no Capítulo 1, dispondo como exemplo os problemas da empresa sob estudo como custos de energia para manter servidores em execução 24x7 e aplicações que necessitam da execução desses servidores para funcionar. Com este propósito, foi elaborado um estudo de caso com dois objetivos principais.

O primeiro objetivo, descrito no Capítulo 4, foi a elaboração de um plano de migração que definiu um processo para auxiliar as migrações de funcionalidades de sistemas legados, transformando-as em funcionalidades modernas ou até mesmo novas aplicações. O plano possui quatro fases distintas, sendo tais: o processo de definição de estado **(F1)**; o processo de agregação **(F2)**; o processo de desenvolvimento **(F3)** e o processo de documentação **(F4)**.

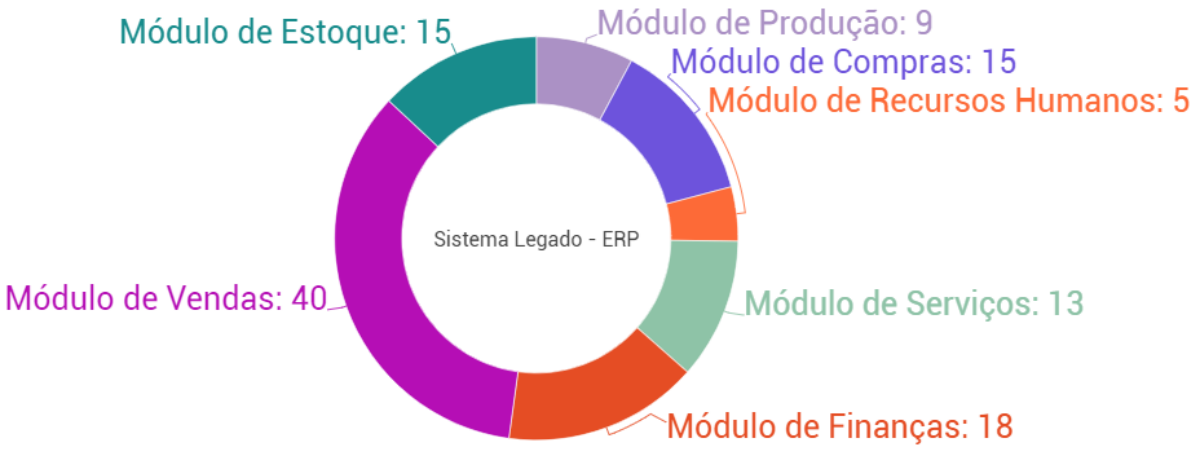
O segundo objetivo foi a criação de um fluxo que realiza a transmissão de dados entre o sistema legado e as aplicações modernas, sendo motivado pelo fluxo legado descrito no Capítulo 2 que justificou a reengenharia necessária no sistema.

5.1 Processo de definição de estado (F1)

Essa fase realizou as ações de identificação das funcionalidades existentes e a definição dos objetivos e metas da migração. Partindo deste contexto, foram reunidos todos os módulos do sistema legado para identificação das suas respectivas funcionalidades, conforme pode ser visualizado na Figura 4.

O sistema é dividido entre os módulos de produção, compras, recursos humanos, serviços, finanças, estoque e vendas. Cada um dos módulos contém a quantidade de funcionalidades existentes e mais utilizadas no sistema com base nas informações disponibilizadas pela empresa.

Figura 4 – Módulos do sistema legado



Fonte: Autoria própria.

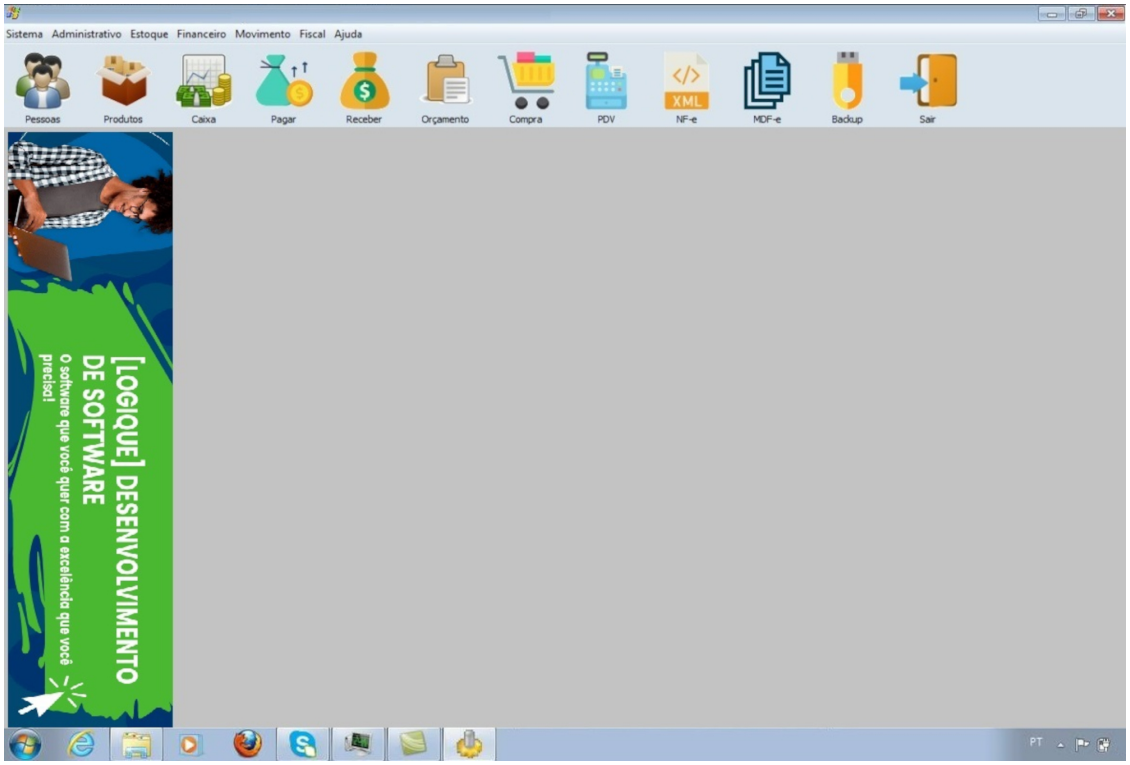
- **Módulo de Produção**: Representando 9 das 115 funcionalidades do sistema, é essencial para otimizar o desempenho de empresas industriais, reduzindo custos, perdas e prazos de entregas, contendo em sua grade de funcionalidade a supervisão dos processos de criação dos materiais, qualidade dos produtos, engenharia necessária, ferramentaria requirida, planejamento e controle de custos.
- **Módulo de Compras**: Representando 15 das 115 funcionalidades do sistema, é essencial para controle do fluxo de compras, contando com requisições internas e externas de materiais, pedidos de compras, notas fiscais, baixa em estoque e orçamentos.
- **Módulo de Recursos Humanos**: Representando 5 das 115 funcionalidades do sistema, é essencial para o controle de cargos dos colaboradores e seus respectivos setores, registrando suas funções e qualificações.
- **Módulo de Serviços**: Representando 13 das 115 funcionalidades do sistema, é essencial para empresas de peças e serviços, contendo um histórico de serviços por placa, agendamento, relatórios, comissões para montador ou vendedor, frotas e emissão de nota fiscal de serviço.
- **Módulo de Finanças**: Representando cerca de 18 das 115 funcionalidades do sistema, é essencial para o controle de contas a pagar e receber, faturamentos pendentes, extratos, fluxo de caixa nas empresas, rateio de despesas, transferências e a curva de experiência de clientes e fornecedores.
- **Módulo de Vendas**: Representando 40 das 115 funcionalidades do sistema, é essencial para gerenciar o processo comercial das empresas, desde o atendimento ao cliente até os resultados das vendas, contém a emissão dos pedidos de vendas, comissões de vendas por vendedor, rotas de entrega, relatórios, emissão de notas fiscais e integração com máquinas de cartões.
- **Módulo de Estoque**: Representando 15 das 115 funcionalidades do sistema, é essencial para controlar os estoques das empresas, isto é, controle de produtos para realizar vendas aos clientes e compras de fornecedores.

Os módulos do ERP podem ser visualizados na simulação da interface do sistema legado da Figura 5. A simulação foi necessária devida a recusa da divulgação das interfaces do sistema legado pela empresa referenciada no trabalho.

Seguindo o objetivo do trabalho, foi executada a migração das funcionalidades legadas dos módulos do ERP para aplicações modernas, que neste trabalho será exemplificado em uma aplicação *mobile* na Seção 5.3.3.3. Essa aplicação *mobile* consiste em demonstrar de uma forma resumida e eficaz os dados gerados pelo sistema legado.

Como será apresentado na próxima seção, foi optado por não seguir a abordagem **A1.1** da Seção 4.2, que visa reconstruir inteiramente o sistema legado para uma aplicação *Web*, uma

Figura 5 – Simulação da interface do sistema legado



Fonte: Autoria própria.

vez que se tornou inviável por custos de desenvolvimento, especialmente para empresas que mantêm um sistema legado e que dependem dos seus clientes atuais para obter seus rendimentos.

Seguindo este contexto, o sistema legado continuou seu funcionamento durante o processo de migração, gerando os dados que foram utilizados para consumo da aplicação *mobile*, sem prejudicar assim os usuários atuais.

5.2 Processo de agregação (F2)

Essa fase contempla a combinação das abordagens de migração e a agregação das funcionalidades identificadas pela fase anterior para estruturar como deverá acontecer a migração. Como abordagem principal para este trabalho, foi escolhida a abordagem **A1.2** que é responsável por realizar adaptações para a aplicação *mobile* e utilizar pequenas implantações das principais funcionalidades do ERP.

Após a fase de definição de estado, utilizando a técnica **T1.1** da Seção 4.3, com os conhecimentos de código-fonte do sistema, a primeira agregação foi realizada no módulo de vendas, por ser o maior módulo do sistema em quantidade de funcionalidades. A primeira funcionalidade legada do módulo de vendas que foi transformada em funcionalidade moderna foi a de relatórios, por ser o processo mais demorado devido a grande quantidade de informações no

banco de dados. Dentre tais relatórios, o de resumo de vendas foi escolhido por conter o maior fluxo de dados.

Durante a fase de agregração, houve a indagação por parte do autor, de como os usuários do sistema legado realizariam o acesso às novas aplicações, a exemplo da aplicação *mobile*, pois não foi considerada a transformação dessa funcionalidade. Como descrito na fase F2 da Seção 4.4, ao surgir um requisito não previsto durante a fase de agregação, deverá ser planejada a criação de novas funcionalidades.

Além da agregação das funcionalidades de relatórios, estão vinculadas a essa tarefa, as funcionalidades de *login*, *logout* e configurações dos usuários da aplicação, utilizando os mesmos dados de acesso ao sistema legado (nome do usuário e senha).

5.3 Processo de desenvolvimento (F3)

Procedendo ao desenvolvimento, a fase 3 visa descrever a estratégia de migração, a identificação e remoção de funcionalidades que não agregam valor e a criação das funcionalidades planejadas pela fase anterior.

5.3.1 Estratégia de migração

A técnica **T3** prevê a elaboração de uma estratégia de migração, convertendo os conhecimentos obtidos do código fonte e arquitetura (**T2**) e conhecimento do domínio (**T2**). A estratégia consistiu em desenvolver uma arquitetura que tratasse (ou combatesse) os problemas de custo e disponibilidade, conforme descrito no Capítulo 2. Nesta arquitetura, apresentada na Figura 6, surgem novos conceitos se comparados à arquitetura que depende de um servidor integrado, sendo eles: o conceito de serviços utilizando a SOA e o conceito de Computação em Nuvem (CC) e seus derivados, tal como a Computação em Nuvem Móvel.

Figura 6 – Arquitetura elaborada com o fluxo proposto

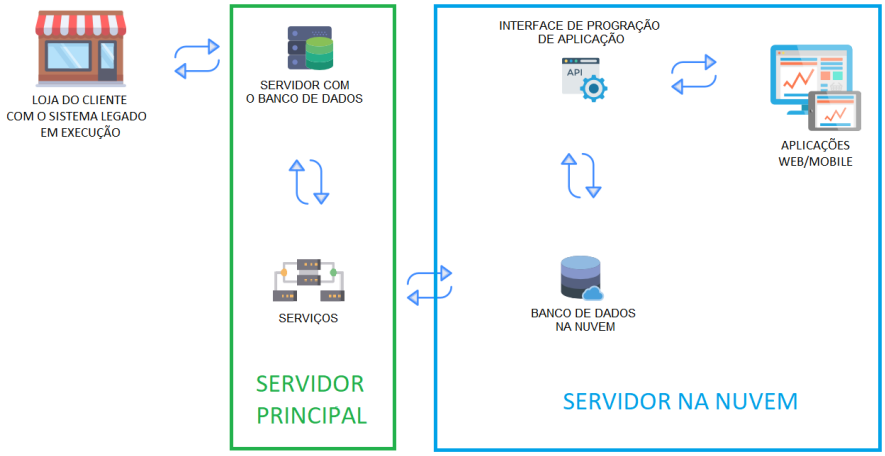
Fonte: Autoria própria.

O conceito de serviços será aplicado para fragmentar o sistema legado em uma arquitetura de microsserviços; esses serviços captam e transmitem em tempo real qualquer alteração nos dados da empresa. Os serviços serão moldados conforme as abordagens do SOA para disponibilizar uma padronização de código às futuras aplicações. O conceito de CC será utilizado para permitir acessar e armazenar dados em servidores, bancos de dados e um conjunto amplo de serviços de aplicações via Internet (AMAZON, 2019).

Conforme a Figura 6, os serviços serão os responsáveis pela migração dos dados entre o banco de dados principal e um ou mais bancos de dados armazenados em um servidor na nuvem. Com isso, é possível migrar somente as informações pertinentes a cada aplicação. No gerenciamento e autenticação dos acessos as informações do banco de dados na nuvem, será necessária a construção de uma API para realizar a integração com as aplicações de maneira ágil e segura, conforme as regras de negócio de cada empresa.

Para melhor entendimento, podemos separar a arquitetura elaborada e o fluxo proposto em duas partes, apresentadas na Figura 7, onde são separados os componentes que estão armazenados no servidor principal destacado em verde e os que estão armazenados no servidor na nuvem destacado em azul.

Figura 7 – Arquitetura elaborada com o fluxo proposto separado por servidores



Fonte: Autoria própria.

Caso ocorra um fluxo não esperado no servidor principal, a interrupção dos componentes não afetará o funcionamento das demais aplicações, pois o servidor na nuvem não depende do servidor principal e os dados que foram gerados pelas aplicações serão armazenadas em contingência no banco na nuvem. Assim que a interrupção for resolvida, o fluxo proposto volta ao funcionamento esperado, os serviços são reativados e os dados gerados pelas aplicações são migrados normalmente entre os servidores e os bancos de dados.

Seguindo a estratégia de migração definida, esta arquitetura visa diminuir os custos de produtividade em decorrência de inatividade do sistema, tanto pelo lado da empresa fornecedora quanto pelo lado da empresa cliente. Neste fluxo existe a correção do problema do funcionamento do servidor na escala 24x7, que gerava um alto custo de energia para manter as aplicações

em execução. A implementação da nova arquitetura ocorrerá com a construção da infraestrutura necessária para construção do servidor na nuvem e será apresentada na Seção 5.3.3.

Como descrito na abordagem **A4**, as novas funcionalidades identificadas devem ser desenvolvidas seguindo um conceito de alto nível e com foco no desenvolvimento dos serviços por meio de regras de negócio. Conforme Wiggins (2017) e Fowler (2012), para seguir essa convicção, devem ser implementadas as seguintes boas práticas:

- **Base de Código**: O código deve ter controle de versão e deve ser capaz de gerar diversas implantações.
- **Processos**: Executar a aplicação como um ou mais processos sem depender do estado de outros processos.
- ***Port binding***: Exportar serviços por meio de vinculação de porta.
- **Concorrência**: Isolamento de cada processo.

Juntamente com a abordagem citada acima, a técnica **T2**, citada na Seção 4.3, foi utilizada para centralizar as regras de negócios do sistema legado que irão persistir nas aplicações modernas se tornando assim requisitos durante o desenvolvimento, tais regras são:

- **Segurança de acesso**: Somente usuários do sistema legado terão acesso à aplicação *mobile*.
- **Segurança de dados**: As senhas dos usuários devem ser criptografadas no banco de dados da nuvem.
- **Performance de dados**: O serviço deve apenas enviar informações resumidas do banco de dados legado para o banco de dados na nuvem.

5.3.2 Identificação e remoção de funcionalidades

Essa atividade foi responsável por identificar e remover funcionalidades que não agregam valor. Neste estudo de caso, para exemplificação do cenário de remoção que pode acontecer durante este processo, foi optado por remover a funcionalidade de configurações do usuário definida durante o processo de agregação (**F2**) descrito na Seção 5.2, Essa remoção ocorreu pelo motivo da aplicação *mobile* ser somente a visualização de dados resumidos de relatórios em forma de gráficos, não requerendo a utilização de nenhuma das configurações de usuário.

5.3.3 Criação das funcionalidades

Utilizando a ténica **T2**, relacionando o custo-benefício e entrando no contexto da criação das funcionalidades, foram definidas as tecnologias, ferramentas e conceitos que podem

ser utilizados de forma gratuita. Tais escolhas para a seleção das bibliotecas para o desenvolvimento da aplicação *mobile* e para a utilização da Computação em Nuvem no processo, foram também baseadas em estudos atuais sobre as linguagens de programação mais utilizadas e com maior visão de crescimento no mercado (ALURA, 2021). Outro motivo para as devidas escolhas, é o conhecimento de desenvolvimento do autor na ferramenta e na linguagem de programação. Todo o código foi disponibilizado na internet e pode ser acessado a partir do link https://github.com/PauloMachine/tcc-dataneed. As tecnologias utilizadas estão descritas a seguir:

- **Visual Studio Code**: Dentre as ferramentas, foi escolhido o Visual Studio Code pela sua simplicidade e diversas integrações que facilitam o desenvolvimento da aplicação, no mesmo programa é possível codificar tanto o *Back-end* como o *Front-end*.

  Criado em 2015 pela Microsoft, o Visual Studio Code é um editor de código-fonte leve e poderoso, está disponível para Windows, macOS e Linux. Ele tem suporte interno para JavaScript, TypeScript e Node.JS e possui um rico ecossistema de extensões para outras linguagens (como C++, Java, Python, PHP, Go) e tempos de execução (como .NET e Unity). O Visual Studio Code dispõe para uso o JavaScript IntelliSense que fornece a conclusão inteligente de código, informações sobre parâmetros, pesquisa de referências e muitos outros recursos avançados de idioma (MICROSOFT, 2015).

- **JavaScript**: Dentre as linguagens de programação, foi escolhido o JavaScript pelo motivo de se utilizar a mesma linguagem para programar o Back-end e o Front-end, simplificando assim, a manutenção e o conhecimento de demais linguagens.

  De acordo com Flanagan (2006), o JavaScript é uma linguagem de programação e foi criada na empresa Netscape, durante a fase inicial da Web. A ampla maioria dos sites modernos usa JavaScript e todos os navegadores modernos, desde computadores de mesa, tablets e smartphones incluem interpretadores JavaScript. É considerada uma linguagem de alto nível, dinâmica, interpretada e com foco a programação orientada a objetos.

Para construção da arquitetura na nuvem serão utilizados os seguintes conceitos, componentes e serviços:

- **Node.JS**: O Node.JS será utilizado para desenvolver nosso Back-end, o mesmo será o servidor da arquitetura e implementará toda a regra de negócio, isto é, sendo a API que o Front-end vai se conectar e se comunicar.

  O Node.JS foi projetado para criar aplicativos de rede escaláveis. Os usuários do Node.JS estão livres de preocupações de bloquear processos, pois não há bloqueios. Quase nenhuma função no Node.JS executa entradas e saídas diretamente, portanto,

o processo nunca bloqueia. Como nada bloqueia, Node.JS é muito indicado para o desenvolvimento de sistemas escaláveis. (DAHL, 2009).

- **MongoDB**: Dentre os bancos de dados, foi escolhido o MongoDB por permitir um integração simplificada com o Node.JS. Outro motivo foi por ser um banco não relacional e que trabalha com estrutura JSON.

  O MongoDB foi criado em fevereiro de 2009 pela empresa MongoDB Inc. É um banco de dados de código aberto, gratuito e que contém uma alta performance por trabalhar sem esquemas e ser orientado a documentos. MongoDB foi criado com o intuito de trabalhar com grandes volumes de dados, pois suporta tanto escalonamento horizontal quanto vertical usando replica *sets* (instâncias espelhadas) e *sharding* (dados distribuídos), aguentando assim volumes de dados (DUARTE, 2017).

Para a construção da aplicação *mobile* consumidora da nova arquitetura serão utilizadas as seguintes tecnologias e conceitos:

- **React Native**: Criado pelo Facebook em 26 de março de 2015, o React Native combina as melhores partes do desenvolvimento nativo *mobile* com o React e utiliza uma biblioteca superior de JavaScript para criar interfaces para usuário que permitem compilar projetos nos sitemas operacionais Android e iOS (NATIVE, 2015).

5.3.3.1 Back-end Serviço (SERVIDOR PRINCIPAL)

Contemplando o processo de migração e atendendo o fluxo proposto foram desenvolvidos três serviços com execução automática a cada minuto. Estes serviços realizam consultas no banco de dados (Firebird) do sistema ERP. Após o retorno destas consultas, é realizado o envio dos dados obtidos para o banco de dados na nuvem (MongoDB), tendo como regra somente enviar as informações pertinentes a serem consumidas pela aplicação *mobile*. Este processo pode ser visualizado nos códigos a seguir.

No Código 1, utilizando a técnica **T1.1**, com os conhecimentos de código-fonte do sistema legado, é apresentada a inicialização do microsserviço. Na linha 1 é instanciado o *CronJob* que realiza a execução do microsserviço a cada minuto. Dentro do controlador *ERPController* estão agrupados todos os serviços e as chamadas para a execução ocorrem como descrito a seguir.

O serviço *findUsers* na linha 3 é responsável por migrar os usuários, o serviço *findSales* na linha 4 é responsável por migrar as vendas mensais e o serviço *findProducts* na linha 5 é responsável por migrar os produtos mais vendidos.

Como todos os serviços seguem a mesma estrutura, para exemplificação, será demonstrado o código do serviço de migração de usuários. Dentro do serviço *findUsers*, no Código 5.2,

Listagem 1 – Exemplo da chamada dos serviços em JavaScript

```
1  new CronJob('* * * * *', () => {
2    try {
3      ERPController.findUsers(); \\ Service 1
4      ERPController.findSales(); \\ Service 2
5      ERPController.findProducts(); \\ Service 3
6    } catch (err) {
7      console.log('MIGRATION BETWEEN FIREBIRD AND MONGO: ', err);
8    }
9  }, null, true);
```

Fonte: Autoria própria.

na linha 4, é realizada a consulta no banco de dados legado (Firebird) para retornar os dados do usuário, isto é, nomes e senhas que servirão para acessar a aplicação *mobile*.

O Código 2 também contém a aplicação da abordagem **A4**, que visa a utilização da regra de negócio de segurança de acesso. Na linha 8, é chamada a função *updateOrCreate* que é responsável por enviar o acesso de cada usuário para o banco de dados na nuvem (MongoDB) que será processado no Código 3.

No Código 3, dentro da função *updateOrCreate*, é realizado o envio dos dados de acesso de cada usuário, na linha 3 e 4 é realizada uma verificação e uma validação, onde é verificado se existe um usuário com o mesmo nome do que está sendo enviado na requisição. Neste momento entra a validação, se não houver nenhum usuário é realizada a criação do mesmo no banco de dados da nuvem, conforme pode ser visualizado na linha 5, 6 e 7.

Listagem 2 – Exemplo de consulta no banco de dados Firebird do sistema legado em JavaScript

```
1  async findUsers() {
2    await Firebird.attach(Conexao, async (err, connection) => {
3      try {
4        connection.query("SELECT NOME, SENHA FROM USERS",
5          async (err, result) => {
6            try {
7              return await result.map(async (user) => {
8                UserController.updateOrCreate(user)
9              });
10           } catch (err) {
11             console.log('FIREBIRD QUERY USERS REQUEST: ', err);
12             return { error: err };
13           }
14         }
15       )
16     } catch (err) {
17       console.log('FIREBIRD CONNECTION REQUEST: ', err);
18       return { error: err };
19     }
20   });
21 }
```

Fonte: Autoria própria.

5.3.3.2 Back-end API (SERVIDOR NA NUVEM)

Adentrando no código-fonte da API, no Código 4, é possivel observar as rotas de acesso da aplicação *mobile*, isto é, na linha 9, está a rota de *login*, onde o usuário irá passar as informações de acesso (nome e senha) para se conectar na aplicação. Na linha 10, está a rota de vendas, que irá buscar as informações de vendas por mês e por fim, na linha 11, está a rota de produtos, que irá buscar as informações dos produtos mais vendidos, ressaltando que todo esse processo acontece em um servidor na nuvem.

No Código 5, é possível observar a rota de login e suas respectivas verificações e validações. Na linha 3, são obtidas as informações repassadas pelo usuário (nome e senha), após isso, na linha 5, é verificado se existe algum usuário com aquele nome. Se existir, na linha 8,

Listagem 3 – Exemplo de envio de dados para o banco de dados na nuvem MongoDB em JavaScript

```
1  async updateOrCreate(user) {
2    try {
3      const findUser = await UserSchema.findOne({ nome: user.NOME });
4      if (!findUser)
5        return await UserSchema.create({
6          nome: user.NOME, senha: user.SENHA
7        });
8    } catch (err) {
9      console.log('USER UPDATE OR CREATE REQUEST: ', err);
10     return { error: err };
11   }
12 }
```

Fonte: Autoria própria.

é verificado se a senha repassada é igual à senha criptografada no banco de dados da nuvem; se a comparação for correta, o usuário recebe um *token* de autenticação e consegue acessar a aplicação *mobile*.

Nos Códigos 6 e 7, é possível observar as rotas de produtos e vendas. As linhas 3 e 4 de cada código realizam a consulta no banco de dados e retornam as informações resumidas para a aplicação *mobile*.

5.3.3.3 React Native

Devido a utilização a abordagem **A1.2**, onde a migração é realizada em pequenas implantações, todas as informações migradas ainda são geradas pelas funcionalidades do módulo de vendas do sistema legado, requisitadas por meio da API e visualizadas na aplicação *mobile*. Na Figura 8, é possível visualizar as telas da aplicação *mobile*, contendo respectivamente:

- **Tela de *login*:** Utiliza as informações migradas dos usuários do banco de dados legado para o banco de dados na nuvem;
- **Tela de gráficos de vendas por mês e Tela de *ranking* de produtos mais vendidos:** Utiliza as informações migradas do relatório de resumo de vendas do sistema legado;

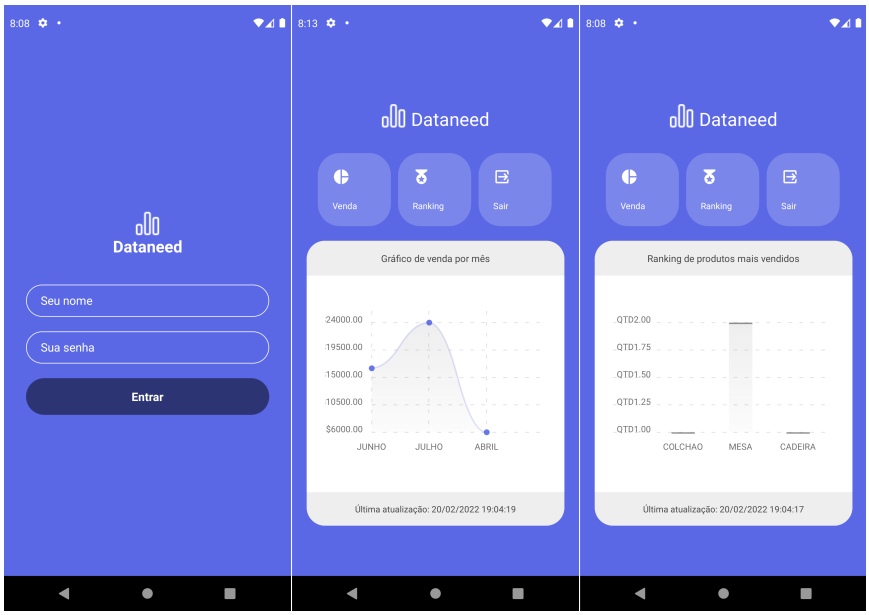
Os dados dos gráficos da aplicação *mobile* são atualizados a cada requisição e a cada minuto, conforme os dados são gerados pelos usuários no sistema legado. Para obter controle

Listagem 4 – Rotas de acesso da API em JavaScript

```
1  const express = require("express");
2
3  const AuthMiddleware = require('./middlewares/AuthMiddleware');
4  const AuthController = require('./controllers/AuthController');
5  const SaleController = require("./controllers/SaleController");
6  const ProductController = require("./controllers/ProductController");
7
8  const routes = new express.Router();
9  routes.post("/auth", AuthController.find);
10 routes.get("/sale", AuthMiddleware, SaleController.find);
11 routes.get("/product", AuthMiddleware, ProductController.find);
12
13 module.exports = routes;
```

Fonte: Autoria própria.

Figura 8 – Telas de *login* e dos gráficos



Fonte: Autoria própria.

dessa situação em casos de erros no servidor principal, foi desenvolvida a funcionalidade "Última atualização" que demonstra a data e a hora da última atualização de dados enviada do servidor principal para o servidor na nuvem.

Listagem 5 – Exemplo de autenticação de acesso do usuário em JavaScript

```
module.exports = {
  async find(req, res) {
    const { nome, senha } = req.body;

    let user = await User.findOne({ nome });

    if (!user) return res.status(400).send({ error: 'USER NOT FOUND!' });
    if (!await bcrypt.compare(senha, user.senha))
      return res.status(400).send({ error: 'INVALID PASSWORD!' });

    user.password = undefined;

    const token = jwt.sign({ id: user.id },
                  authConfig.secret,
                  { expiresIn: 86400 })
    return res.json({ user, token });
  }
}
```

Fonte: Autoria própria.

5.4 Processo de documentação (F4)

Essa fase tem como foco construir toda a documentação dos processos realizados durante a migração, sendo responsável por ordenar e categorizar as funcionalidades. Conforme a Figura 9, é possivel visualizar o planejamento geral com os processos realizados por mês, havendo inicio em Junho e finalizando em Dezembro de 2021.

Listagem 6 – Exemplo da rota de produto em JavaScript

```
module.exports = {
  async find(req, res) {
    const Product = await ProductSchema.find();
    return res.json(Product);
  }
};
```

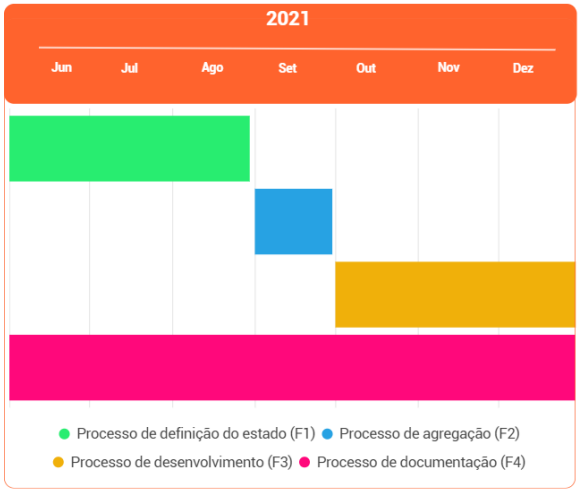
Fonte: Autoria própria.

Listagem 7 – Exemplo da rota de vendas em JavaScript

```
module.exports = {
  async find(req, res) {
    const Sale = await SaleSchema.find();
    return res.json(Sale);
  }
};
```

Fonte: Autoria própria.

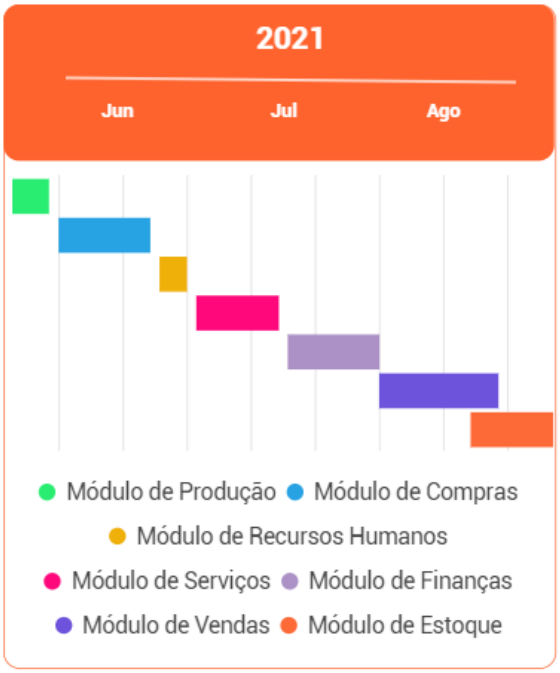
Figura 9 – Planejamento geral



Fonte: Autoria própria.

No período de Junho a Agosto, destacado na cor verde da Figura 9, ocorreu a primeira fase da migração. Neste tempo, houve a identificação de todos os módulos do sistema legado que podem ser visualizados na Figura 10.

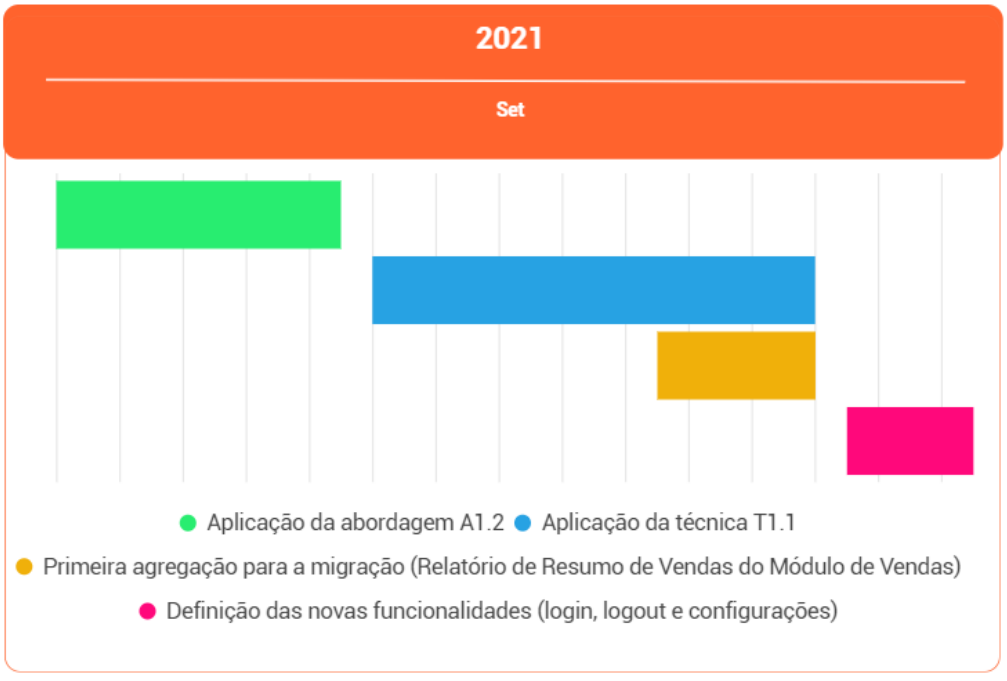
Figura 10 – Período para identificação dos módulos e suas funcionalidades



Fonte: Autoria própria.

No período de Setembro, destacado na cor azul da Figura 9, ocorreu a segunda fase da migração. Neste tempo, houve o envolvimento das abordagens de migração, agregação das funcionalidades identificadas pela primeira fase e o planejamento de novas funcionalidades, conforme pode ser visualizado na Figura 11 e conforme descrito na Seção 5.2.

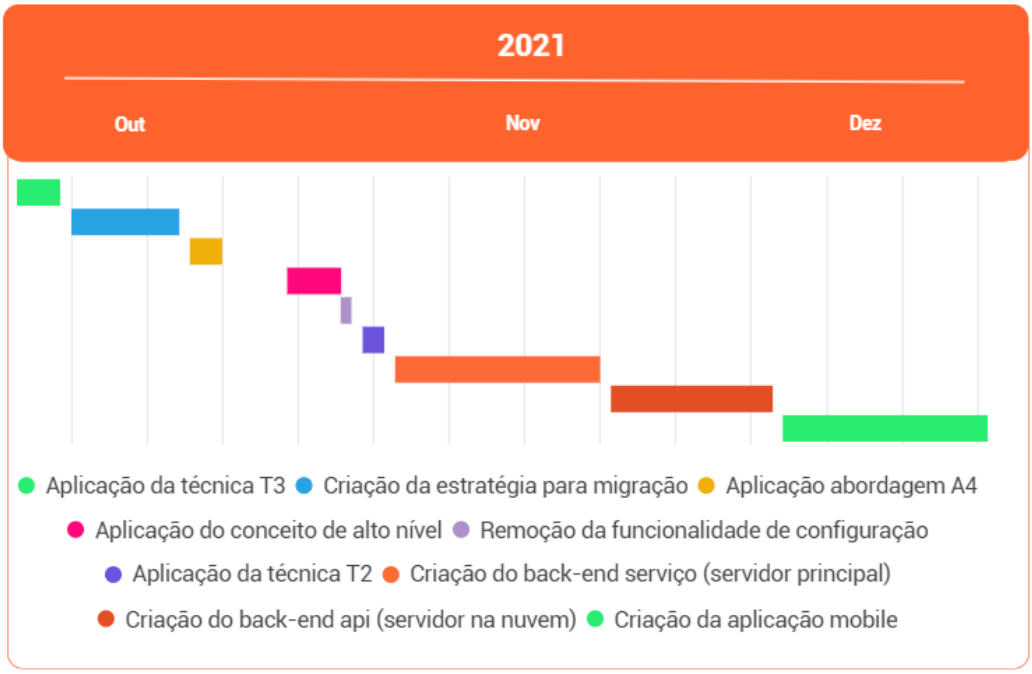
Figura 11 – Período de definição das funcionalidades a serem migradas e desenvolvidas



Fonte: Autoria própria.

No período de Outubro a Dezembro, destacado na cor amarela da Figura 9, ocorreu a terceira fase da migração. Neste tempo, houve a identificação e remoção de funcionalidades que não agregam valor, a criação das funcionalidades planejadas pela fase anterior e recuperação do conhecimento da estratégia de migração, conforme pode ser visualizado na Figura 12 e conforme descrito na Seção 5.3.

Figura 12 – Período de desenvolvimento das funcionalidades



Fonte: Autoria própria.

Por fim, durante todo o período do migração, isto é, de Junho a Dezembro, houve a documentação de todos os processos das fases, contendo seus processos, suas técnicas e abordagens individuais como relatado neste Capítulo. Neste estudo de caso, foi a realizada a migração de somente uma das funcionalidades de um dos módulos do sistema legado. Sendo assim, a primeira funcionalidade migrada foi a de Resumo de Vendas que se encaixa na categoria de Vendas e na subcategoria de Relatórios do sistema legado.

Após a finalização dos processos das fases do plano de migração, obtemos os seguintes resultados: uma redução de custos para o usuário final e uma otimização de tempo na geração dos relatórios. As conclusões destes resultados foram apresentadas no Capítulo 6.

6 RESULTADOS

Neste Capítulo são apresentados os resultados obtidos neste estudo de caso. Dentre os resultados, a utilização da nova arquitetura com foco na diminuição do custo de produtividade para o usuário final, é apresentada na Seção 6.1. A performance de dados com a utilização do novo fluxo de transmissão de dados e com o servidor na nuvem são demonstrados na Seção 6.2. Por fim, na Seção 6.3 são discutidas as ameaças à validade do presente trabalho.

6.1 Redução de custos para o usuário final

De acordo com a estrutura de custos definida no Capitulo 3, os problemas enfrentados de erros inesperados ou desligamento do servidor juntamente com a arquitetura de fluxo legada, resultavam em custos financeiros tanto para a empresa quanto para seu usuário final. Estes custos e seus respectivos processos durante o problema são:

- **Detecção**: Para a empresa tem como custo de atividade o tempo até a descoberta inicial e investigação sobre o erro ocorrido no servidor. Para o usuário final tem como custo o tempo de inatividade até a solução do problema.
- **Contenção**: Para a empresa tem como custo de atividade o tempo para realizar uma estratégia de contenção para controle do erro, impedindo que ele se espalhe, agrave ou cause maiores pertubações. Para o usuário final tem como custo o tempo de inatividade até a solução do problema.
- **Recuperação**: Para a empresa tem como custo de atividade o tempo para reorganização das redes e sistemas principais para o estado estável após o erro ter ocorrido. Para o usuário final tem como custo o tempo de inatividade até a solução do problema.
- **Resposta**: Para a empresa tem como custo de atividade o tempo de resposta das redes e sistemas principais para o estado estável após o erro ter ocorrido. Para o usuário final tem como custo o tempo de inatividade até a solução do problema.
- **Equipamento**: Para a empresa tem como custo de atividade o gasto de compras e reparos de novos equipamentos, incluindo reformas.
- **Produtividade (TI)**: Para a empresa tem como custo de atividade o tempo de inatividade e as despesas relacionadas aos erros e manutenções no servidor.
- **Produtividade (Usuário final)**: Tem como custo de atividade o tempo de inatividade do usuário final perante ao não funcionamento do servidor.

- **Terceiros**: Para a empresa tem como custo de atividade o gasto com contratados, consultores, auditores e outros especialistas engajados para ajudar a resolver os problemas não planejados.

A Tabela 1 demonstra um resumo cada custo da estrutura e respectivamente os efeitos financeiros pelo processo de tentativa de correção do problema.

Tabela 1 – Efeitos financeiros causados por erros inesperados ou desligamento do servidor juntamente com a arquitetura legada.

Afetado pelo custo		
Custos	Empresa	Usuário final
Detecção	Sim	Sim
Contenção	Sim	Sim
Recuperação	Sim	Sim
Resposta	Sim	Sim
Equipamento	Sim	Não
Produtividade (TI)	Sim	Não
Produtividade (Usuário final)	Não	Sim
Terceiros	Sim	Não

Fonte: Autoria própria.

O estudo de Ponemon (2016) foi realizado com empresas dos ramos de comércio eletrônico, serviços financeiros, cuidados de saúde, serviços gerais, setor industrial, setor público, produtos de consumo, educação, meios de comunicação e transporte. Quarenta e nove empresas disponibilizaram dentre um a dez servidores para o estudo. Para uma empresa com um único servidor, os custos para o usuário final, com base no tempo de inatividade durante os doze meses de estudo, somaram no ano de 2016 o valor aproximado de **R$ 13.923,00**. Esses custos se dividem como mostrado abaixo:

- **Detecção:** R$ 1.800,00.
- **Contenção:** Não informado no estudo.
- **Recuperação:** R$ 1.382,00.
- **Resposta:** R$ 910,00.
- **Produtividade:** R$ 9.831,00.

Outro agravante ao custo de produtividade é sua relevância na chance de desistência de uso do sistema por clientes existentes da empresa, pois as falhas de servidor causam a inativação do usuário final. Por não conseguir utilizar as aplicações e consequentemente não conseguir finalizar seus serviços, gera-se assim problemas financeiros ao cliente.

Para sanar esses problemas, o novo fluxo de transmissão de dados entre o sistema legado e as aplicações modernas faz com que, ao ocorrer alguma falha no servidor principal, o usuário

final consiga trabalhar em contingência até que ocorra a normalização do servidor. Com esse tratamento, a inatividade do usuário final não acontece. Dessa forma, a estrutura de custos se modifica, conforme pode ser visualizada na Tabela 2.

Tabela 2 – Efeitos financeiros após a implementação do processo MIGPLAN

	Afetado pelo custo	
Custos	Empresa	Usuário final
Detecção	Sim	Parcialmente
Contenção	Sim	Parcialmente
Recuperação	Sim	Parcialmente
Resposta	Sim	Parcialmente
Equipamento	Sim	Não
Produtividade (TI)	Sim	Não
Produtividade (Usuário final)	Não	Não
Terceiros	Sim	Não

Fonte: Autoria própria.

A descrição **parcialmente** continua se referindo aos custos de problemas com o servidor, já que não estão funcionando como deveriam, porém, por meio da nova arquitetura, tais problemas não afetam a produtividade do usuário final. Como os custos do usuário final são relacionados ao tempo de inatividade ao não conseguir efetuar seus serviços, por meio do uso da aplicação em contingência, tais custos acabam camuflados até ocorrer a normalização do servidor. Ou seja, a aplicação continua operacional, mesmo com dados antigos. Resolve-se assim dois problemas: os custos para o usuário final e a chance de desistência e insatisfação do mesmo.

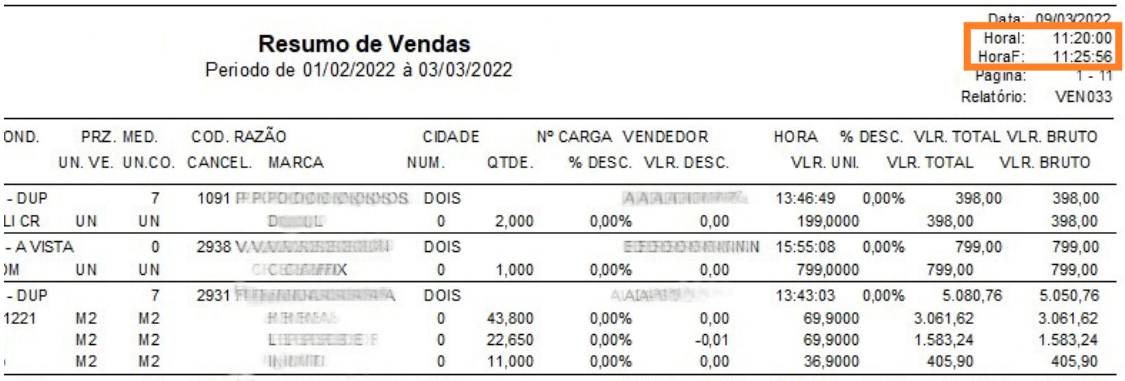
Juntamente a essas informações, o mesmo estudo de Ponemon (2016) relatou que o tempo médio para recuperação parcial a recuperação total do servidor é de aproximadamente noventa e cinco minutos, englobando todos os custos. Com a nova arquitetura, o usuário final não será afetado por este tempo. Como citado acima, o sistema continua operacional mesmo em contigência.

6.2 Performance de dados

Conforme demonstra a Figura 13, o relatório de Resumo de Vendas demora aproximadamente seis minutos para ser apresentado ao usuário final, considerando o fluxo de execução normal do sistema legado, ou seja, sem erros. Esse cálculo considerou o horário inicial da geração do relatório e o horário em que o relatório foi gerado, ambos destacados em vermelho no canto superior direito da figura, respectivamente pela váriavel **HoraI** (hora inicial) e **HoraF** (hora final).

Para uma comparação enviesada, foi utilizado um processo simulado da nova arquitetura, sem a utilização dos dados reais da empresa, pois não houve a autorização da mesma para obter essa métrica. No banco de dados em nuvem, são enviados apenas os dados persistentes que

Figura 13 – Tempo para geração do relatório legado Resumo de Vendas.



Resumo de Vendas
Periodo de 01/02/2022 à 03/03/2022

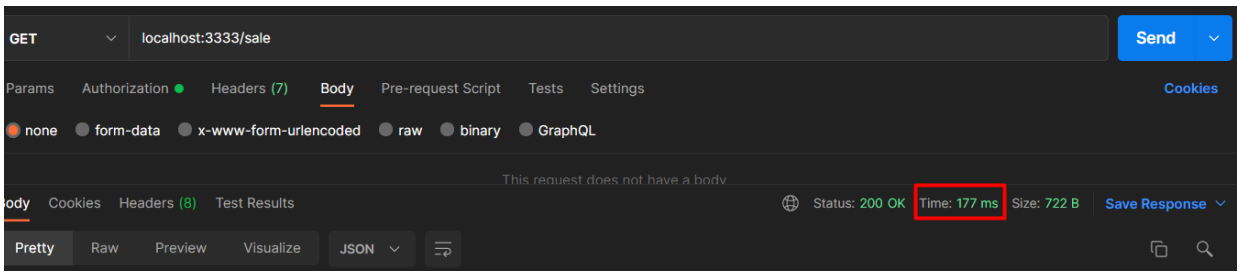
Data: 09/03/2022
Horal: 11:20:00
HoraF: 11:25:56
Pagina: 1 - 11
Relatório: VEN033

OND.	PRZ. MED.	COD. RAZÃO	CIDADE	Nº CARGA	VENDEDOR	HORA	% DESC.	VLR. TOTAL	VLR. BRUTO
	UN. VE. UN.CO.	CANCEL. MARCA	NUM.	QTDE.	% DESC. VLR. DESC.	VLR. UNI.		VLR. TOTAL	VLR. BRUTO
- DUP	7	1091	DOIS			13:46:49	0,00%	398,00	398,00
LI CR UN	UN		0	2,000	0,00% 0,00	199,0000		398,00	398,00
- A VISTA	0	2938	DOIS			15:55:08	0,00%	799,00	799,00
)M UN	UN		0	1,000	0,00% 0,00	799,0000		799,00	799,00
- DUP	7	2931	DOIS			13:43:03	0,00%	5.080,76	5.050,76
1221 M2	M2		0	43,800	0,00% 0,00	69,9000		3.061,62	3.061,62
M2	M2		0	22,650	0,00% -0,01	69,9000		1.583,24	1.583,24
M2	M2		0	11,000	0,00% 0,00	36,9000		405,90	405,90

Fonte: Autoria própria.

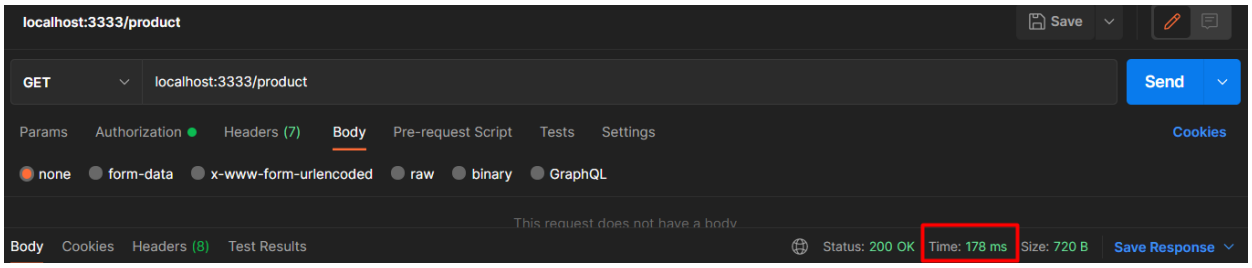
os gráficos necessitam para serem apresentados. Consequentemente, as suas requisições ficaram menores do que duzentos milissegundos, conforme demonstrado na Figura 14 e na Figura 15. O componente do gráfico demora entre três a quatro segundos para renderizar no *mobile* e, para exemplificação, o tempo total para a apresentação de cada gráfico foi arredondado para cinco segundos.

Figura 14 – Tempo da requisição para gerar o gráfico de vendas por mês



Fonte: Autoria própria.

Figura 15 – Tempo da requisição para gerar o gráfico de *ranking* de produtos mais vendidos



Fonte: Autoria própria.

Como mencionado, a comparação é enviesada pois não foi possível comparar o funcionamento das duas arquiteturas com o mesmo volume de dados. Além disso, a funcionalidade de fornecer dados resumidos para geração de gráficos não era presente na arquitetura legada,

fazendo com que não houvesse um balanceamento entre o funcionamento do sistema legado e o sistema com nova arquitetura.

Ainda com essas dificuldades, existe o tratamento de contingência da aplicação em caso de queda, o que diminui os custos associados ao usuário final, e comentado na Tabela 2. Sugere-se que novas aplicações do MIGPLAN considerem análises de performance para mensurar a melhoria de novas arquiteturas propostas

Outro ganho através da nova arquitetura consiste nos serviços que enviam os dados a cada minuto. Com isso, para obter os novos dados, basta realizar uma nova requisição e esperar o tempo de geração dos gráficos em cinco segundos. Se comparado à arquitetura legada, para obter os novos dados, seria necessário gerar novamento o relatório de Resumo de Vendas, ou seja, aguardar o tempo de geração de aproximadamente seis minutos.

6.3 Ameaças à validade

Como ameaça à validade interna, existe a possibilidade de queda dos serviços na nuvem por quem realiza a hospedagem, pois o novo fluxo é dependente de tais serviços para manter a contingência do cliente. Consequentemente, sem a disponibilidade desses serviços, configuram-se os mesmos custos para o usuário final na arquitetura legada. Este é um fator que não pode ser controlado neste estudo de caso e pode afetar os seus resultados.

Como ameaça à validade externa, existe a dificuldade de generalizar o processo do MIGPLAN para todos os tipos de sistema. Para estender o uso do mesmo para outras empresas, o plano de migração deve ser modulado, a partir de um objetivo específico (a exemplo da rentabilidade para o atual trabalho), com novas escolhas de técnicas e abordagens conforme a necessidade, para assim, definir o melhor processo para a realização da migração.

Como ameaça à validade de construção, o novo processo da arquitetura é simulado, não sendo possível identificar o seu comportamento em um volume real de dados, acarretando para que as comparações fossem enviesadas. Isto se deve ao autor não pertencer mais ao quadro de funcionários da empresa citada nesse estudo de caso, o que ocorreu durante o processo de desenvolvimento deste trabalho.

7 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um processo de reengenharia de software a partir da proposta de uma nova arquitetura orientada a serviços e com conceitos de computação em nuvem, com o objetivo principal de melhorar a rentabilidade nas empresas, ou seja, reduzir os custos relacionados à utilização de servidores.

Este trabalho abrangeu uma revisão narrativa que obteve um levantamento e análise de abordagens e técnicas para auxiliar na migração de funcionalidades. Como resultado, o trabalho também apresentou a aplicação dessas técnicas e abordagens na realização das migrações de funcionalidades legadas para funcionalidades modernas.

O MIGPLAN foi demonstrado em uma funcionalidade chave para a empresa citada neste trabalho, ocasionando uma lição aprendida, a ameaça à validade externa. Apesar da dificuldade em exemplificar o processo e mensurar a melhoria de uma nova arquitetura proposta, espera-se que o trabalho possa contribuir com trabalhos futuros de reengenharia de software focados em melhoria de performance e tratamento de custos.

Pretende-se contribuir com trabalhos relacionados à reengenharia de sistemas legados para um novo paradigma de arquitetura utilizando Computação em Nuvem e orientada a serviços. Outras contribuições deste trabalho são destinadas às empresas que não conseguem evoluir e acompanhar o crescimento do mercado, e que possam ter a chance de obter avanços tecnológicos sem grande impacto na sua arquitetura de software existente.

O MIGPLAN deve ser aplicado para solucionar problemas internos dentro das empresas, tais como, a não utilização de um processo sólido para executar as migrações de funcionalidades, o custo de energia para se manter servidores em execução na escala 24x7 e as aplicações que necessitam da execução desses servidores para funcionar.

REFERÊNCIAS

ALURA. **As 10 linguagens de programação mais usadas segundo o GitHub**. 2021. Disponível em: https://www.alura.com.br/empresas/artigos/linguagens-de-programacao-mais-usadas. Acesso em: 27 mar. 2022.

AMAZON, A. **O que é a computação em nuvem?** 2019. Disponível em: https://aws.amazon.com/pt/what-is-cloud-computing/. Acesso em: 14 mai. 2022.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. [*S.l.*]: Addison-Wesley Professional, 2003.

BIELSKI, L. Breakout systems and applications give bankers new options. **American Bankers Association. ABA Banking Journal**, Naylor Communications Ltd., 2005.

CIGANEK, A. P.; HAINES, M. N.; HASEMAN, W. Challenges of adopting web services: experiences from the financial industry. *In*: **38th International Conference on System Sciences**. [*S.l.*: *s.n.*], 2005. p. 168b–168b.

DAHL, R. **NodeJS**. 2009. Disponível em: https://nodejs.org/en/about/. Acesso em: 20 fev. 2022.

DUARTE, L. **Boas práticas com MongoDB**. 2017. Disponível em: https://blog.umbler.com/br/boas-praticas-com-mongodb/. Acesso em: 15 jan. 2022.

FERNANDES, F. C. F.; BERTOLLO, R. M. Avaliação do impacto da reengenharia nas grandes empresas do brasil. **Gestão & Produção**, FCF Fernandes, 1999.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. **Future generation computer systems**, Elsevier, 2013.

FLANAGAN, D. **JavaScript: the definitive guide**. [*S.l.*]: "O'Reilly Media, Inc.", 2006.

FOWLER, M. **Patterns of Enterprise Application Architecture**. [*S.l.*]: Addison-Wesley, 2012.

FOWLER, M. **Microservices and the first law of distributed objects**. 2014. Disponível em: https://martinfowler.com/articles/distributed-objects-microservices.html. Acesso em: 10 jan. 2022.

FOWLER, M.; LEWIS, J. **Microsserviços em poucas palavras**. 2015. Disponível em: https://www.thoughtworks.com/pt/insights/blog/microservices-nutshell. Acesso em: 12 dez. 2021.

KUMAR, S.; DAKSHINAMOORTHY, V.; KRISHNAN, M. S. Does soa improve the supply chain? an empirical analysis of the impact of soa adoption on electronic supply chain performance. *In*: **40th International Conference on System Sciences**. [*S.l.*: *s.n.*], 2007. p. 171b–171b.

MICROSOFT. **Code editing. Redefined.** 2015. Disponível em: https://pt-br.reactjs.org/. Acesso em: 26 dez. 2021.

NATIVE, R. **React Native: Learn once, write anywhere.** 2015. Disponível em: https://facebook.github.io/react-native/. Acesso em: 06 abr. 2022.

PONEMON, I. Data center performance benchmark series. **Cost of Data Center Outages**, Independently conducted by Ponemon Institute, 2016.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [*S.l.*]: McGraw Hill Brasil, 2016.

RAZAVIAN, M.; LAGO, P. A systematic literature review on soa migration. **Journal of Software: Evolution and Process**, Wiley Online Library, 2015.

ROTHER, E. T. Revisão sistemática x revisão narrativa. **Acta paulista de enfermagem**, Universidade Federal de São Paulo, 2007.

SORDI, J. O. D.; MARINHO, B. de L.; NAGY, M. Benefícios da arquitetura de software orientada a serviços para as empresas: análise da experiência do abn amro brasil. **JISTEM: Journal of Information Systems and Technology Management**, Universidade de São Paulo, 2006.

THÖNES, J. Microservices. **IEEE software**, IEEE, 2015.

UNISC, E. **Rentabilidade: o que é? Como calcular? [Guia Completo]**. 2020. Disponível em: https://ead.unisc.br/blog/rentabilidade. Acesso em: 29 jun. 2022.

WIGGINS, A. **12-Factor**. 2017. Disponível em: https://12factor.net/. Acesso em: 20 fev. 2022.