CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE SÃO PAULO

BRUNO MUNARETTI FENERICH

Arquitetura Orientada a Serviços e sua integração com Software as a Service

SÃO PAULO

2023

FACULDADE DE TECNOLOGIA DE SÃO PAULO

BRUNO MUNARETTI FENERICH

Arquitetura Orientada a Serviços e sua integração com Software as a Service

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Professor Aristides Novelli Filho

SÃO PAULO
2023

BRUNO MUNARETTI FENERICH

Arquitetura Orientada a
Serviços e sua integração com
Software as a Service

Trabalho submetido como exigência parcial para obtenção do
Grau de Tecnólogo em Análise e Desenvolvimento de
Sistemas.

Parecer do Professor Orientador

_____

_____

_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinada por mim para avaliação do TCC.

Orientador: Professor Aristides Novelli Filho

SÃO PAULO,_____de_____de 2023.

Assinatura do Orientador

Assinatura do aluno

AGRADECIMENTOS

Gostaria de expressar meu profundo agradecimento ao meu professor Aristides por ter me introduzido ao campo dos sistemas distribuídos que por sua vez me trouxe ideias de todo um universo orientado a serviços. Suas aulas, conhecimento e orientação foram essenciais para minha compreensão e ingresso nessa área. Sua dedicação em transmitir o conhecimento e fornecer insights valiosos contribuíram imensamente para o sucesso deste trabalho de conclusão de curso. Além disso, gostaria de expressar minha gratidão à minha família, que sempre esteve ao meu lado, oferecendo apoio incondicional e incentivo constante ao longo dessa jornada acadêmica. Seu apoio, encorajamento e confiança em mim foram a grande força que me impulsionou a superar desafios e alcançar este marco significativo. Sou verdadeiramente grato por todo o apoio e carinho que recebi.

RESUMO

A Arquitetura Orientada a Serviços (SOA) é uma abordagem arquitetônica que tem como objetivo principal facilitar a integração entre diferentes sistemas e aplicações. Ela se baseia na disponibilização de serviços como elementos fundamentais do sistema, que podem ser consumidos por outros serviços ou aplicações para realizar determinadas tarefas. Cada serviço é definido por um conjunto de funcionalidades específicas que podem ser acessadas por meio de contratos de serviços, que estabelecem as regras de acesso, uso e comportamento do serviço.

O Software as a Service (SaaS) é um modelo de distribuição de software em que um provedor de serviços disponibiliza aplicativos pela internet, e os usuários podem acessá-los através de um navegador web ou de um aplicativo específico. Nesse modelo, o provedor de serviços é responsável pela manutenção e atualização do software, enquanto os usuários pagam uma taxa de assinatura para ter acesso ao aplicativo.

O SaaS pode ser visto como uma aplicação prática da SOA, uma vez que se baseia na disponibilização de serviços como elementos fundamentais do sistema. Ambos os conceitos compartilham a ideia de que as empresas não precisam mais desenvolver e manter seus próprios aplicativos e infraestrutura, podendo contar com provedores de serviços para disponibilizar e gerenciar esses recursos. No entanto, enquanto a SOA é uma abordagem mais geral, o SaaS é uma implementação específica que se concentra na distribuição de software como um serviço.

Palavras-chave: Arquitetura Orientada a Serviços, Software como Serviço, Micros serviços.

ABSTRACT

Service Oriented Architecture (SOA) is an architectural approach whose main objective is to facilitate the integration between different systems and applications. It is based on providing services as fundamental elements of the system, which can be consumed by other services or applications to perform certain tasks. Each service is defined by a set of specific functionalities that can be accessed through service contracts, which establish the rules for access, use and behavior of the service.

Software as a Service (SaaS) is a software distribution model in which a service provider makes applications available over the internet, and users can access them through a web browser or a specific application. In this model, the service provider is responsible for maintaining and updating the software, while users pay a subscription fee to access the application.

SaaS can be seen as a practical application of SOA, since it is based on providing services as fundamental elements of the system. Both concepts share the idea that companies no longer need to develop and maintain their own applications and infrastructure, but can rely on service providers to provide and manage these resources. However, while SOA is a more general approach, SaaS is a specific implementation that focuses on delivering software as a service.

Keywords: Service Oriented Architecture, Software as a Service, Microservices.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ERP	*Enterprise Resource Planning*
SAAS	*Software as a Service*
SOA	*Service-Oriented Architecture*
WSDL	*Web Services Description Language*
XML	*Extensible Markup Language*
XSD	*XML Schema Definition*

SUMÁRIO

INTRODUÇÃO

A Arquitetura Orientada a Serviços (SOA) é um paradigma de design que visa criar sistemas de TI modulares, escaláveis e reutilizáveis, através da organização e coordenação de unidades independentes de lógica, conhecidas como serviços. Por outro lado, Software as a Service (SaaS) é um modelo de distribuição de software onde os aplicativos são hospedados e mantidos por um provedor de serviços e disponibilizados aos usuários através da internet.

Atualmente, o SaaS tem adotado amplamente a abordagem SOA em sua arquitetura. A SOA possibilita a construção de soluções SaaS mais flexíveis e adaptáveis às necessidades do cliente, facilitando a integração com outros serviços e sistemas. A utilização de SOA em SaaS promove maior reutilização de componentes, escalabilidade e manutenção simplificada, otimizando o desenvolvimento e a entrega de aplicações em nuvem.

Objetivo Geral

O objetivo geral deste trabalho é que, ao final de sua leitura, seja possível abstrair conceitos gerais tanto sobre Arquitetura Orientada a Serviços quanto conceitos acerca de Software as a Service. Para isso, será utilizada uma metodologia simples que combina a análise de materiais já publicados e a experiência pessoal.

Primeiro, serão revisados conceitos por meio de livros e artigos relacionados aos temas de Arquitetura Orientada a Serviços e Software as a Service para entender as tendências e desafios do campo. Em seguida, será incorporada a experiência pessoal de desenvolvimento de soluções hospedadas em SaaS, incluindo projetos reais e interações com outros profissionais.

Objetivos Específicos

Introduzir temas como SOA e SaaS, apontando conceitos, premissas e características. Para isso, serão abordados os seguintes pontos de ambas as tecnologias:

- Conceitos e terminologias;
- Vantagens e desafios;
- Padrões e implementações;

Justificativa

Nos últimos anos, o mundo dos negócios tem experimentado uma rápida transformação digital, impulsionada pelo aumento da adoção de tecnologias em nuvem, mobilidade e soluções de TI mais eficientes. Nesse cenário, as tecnologias SaaS (Software as a Service) e SOA (Service Oriented Architecture) se destacam como abordagens críticas para a implementação e gerenciamento de soluções de software e serviços. Diante desse contexto, a demanda por profissionais qualificados e conhecedores dessas tecnologias tem crescido exponencialmente.

Nesse contexto, o objetivo desta pesquisa é elucidar esses conceitos cruciais, ao mesmo tempo que fomenta uma compreensão profunda de 'serviços' entre os estudantes de tecnologia. A intenção é prepará-los para navegar com competência neste mundo em constante evolução, onde a estruturação, implementação e manutenção de serviços se tornam habilidades cada vez mais valorizadas.

Benefícios

Os benefícios da adoção da SOA podem ser citados como reutilização de serviços, interoperabilidade, flexibilidade e agilidade, redução de custos e alinhamento entre negócio e TI. Essas vantagens permitem o desenvolvimento eficiente e a integração de sistemas e aplicativos, adaptabilidade a mudanças nos requisitos de negócio e aprimoramento da colaboração entre equipes.

Já os benefícios das soluções no modelo SaaS englobam a redução de custos, acesso fácil e rápido, escalabilidade, atualizações automáticas e implementação mais rápida. Essas vantagens proporcionam economia de infraestrutura e manutenção, maior mobilidade e flexibilidade, crescimento escalonado e acesso às versões mais recentes dos aplicativos sem esforço adicional.

1. Arquitetura Orientada a Serviços

Na sociedade, as pessoas desempenham diferentes funções e atribuições, colaborando e interagindo umas com as outras para alcançar objetivos comuns. Essa colaboração permite que as habilidades e entregas individuais sejam combinadas, gerando resultados mais abrangentes e significativos. Podemos aplicar um raciocínio semelhante ao mundo da Arquitetura Orientada a Serviços (SOA). Neste cenário, os serviços funcionam como os "membros" dessa sociedade digital, cada um com suas próprias responsabilidades e tarefas específicas. Ao trabalharem juntos de maneira coordenada, os serviços podem criar soluções complexas e eficientes que atendem às necessidades em constante evolução das empresas.

Neste capítulo, exploraremos a natureza dos serviços e serviços compostos dentro da Arquitetura Orientada a Serviços, discutindo suas características, benefícios e como eles se relacionam uns com os outros para promover a colaboração e a inovação no desenvolvimento de sistemas de TI.

1.1 Conceitos e princípios

De acordo com Thomas Erl, um serviço é uma unidade de lógica autônoma e independente na Arquitetura Orientada a Serviços (SOA). Ele é projetado para executar uma função específica e bem definida, atuando como um componente reutilizável dentro de um sistema maior. Essa característica de autonomia e independência permite que os serviços sejam desenvolvidos, implantados e gerenciados separadamente, facilitando a modularidade e a flexibilidade dos sistemas aos quais pertencem.

Dentro da SOA, o consumidor é uma entidade que solicita e utiliza a funcionalidade fornecida por um serviço. A comunicação entre o serviço e o consumidor é facilitada por uma Interface de Programação de Aplicações (API), que define a interface e o contrato de serviço, detalhando as operações disponíveis, os formatos de dados esperados e os protocolos de comunicação.

Os serviços seguem um paradigma de design específico que visa promover a interoperabilidade e a facilidade de integração entre componentes e sistemas distintos. Este paradigma estabelece diretrizes e padrões de comunicação, como REST ou SOAP, aos quais os serviços devem aderir para garantir compatibilidade e eficiência na comunicação.

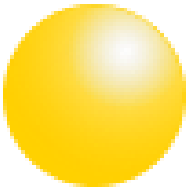

Figura 1: Representação de serviço por Thomas Erl
Fonte: Autor

A entrega do serviço ao consumidor é realizada por meio da API, que atua como um ponto de contato e um mediador entre ambos. Isso permite que os consumidores se integrem facilmente aos serviços e aproveitem suas funcionalidades, promovendo a colaboração e a reutilização no contexto da Arquitetura Orientada a Serviços. Assim, a API garante que os serviços possam ser facilmente descobertos, acessados e utilizados pelos consumidores, contribuindo para a criação de soluções mais flexíveis, escaláveis e adaptáveis no desenvolvimento de sistemas de TI.

1.1.1 Serviços Compostos

Os serviços compostos, conceito central na Arquitetura Orientada a Serviços (SOA), referem-se à combinação e orquestração de diversos serviços individuais para alcançar objetivos de negócio mais abrangentes e complexos. Essa abordagem possibilita o aproveitamento da reutilização e modularidade dos serviços existentes, otimizando a eficiência e adaptabilidade das soluções de TI.

Diferentemente dos serviços simples, que executam funções específicas, os serviços compostos coordenam e integram a funcionalidade de vários serviços menores, criando soluções mais completas e personalizadas. Essa composição ocorre através de técnicas como orquestração, onde um serviço central controla a interação entre os serviços menores, ou coreografia, baseada em regras e protocolos predefinidos.

Para facilitar o entendimento de serviços compostos na Arquitetura Orientada a Serviços (SOA), podemos traçar um paralelo com classes chamando classes em linguagens de programação orientada a objetos. Da mesma forma que classes interagem entre si, compartilhando informações e invocando métodos uns dos outros para criar soluções complexas, os serviços compostos também colaboram de maneira coordenada, combinando e orquestrando suas funcionalidades individuais.

Esse paralelo ajuda a compreender a importância da cooperação entre unidades autônomas de lógica e como elas trabalham em conjunto para atender às necessidades de um sistema maior. Além disso, destaca como a modularidade e a capacidade de reutilização de componentes em ambos os casos resultam em sistemas mais flexíveis e escaláveis, capazes de se adaptar às mudanças nos requisitos de negócio. A comparação entre serviços compostos e classes chamando classes ilustra a relevância da colaboração e coordenação entre diferentes componentes para criar soluções abrangentes e eficientes.

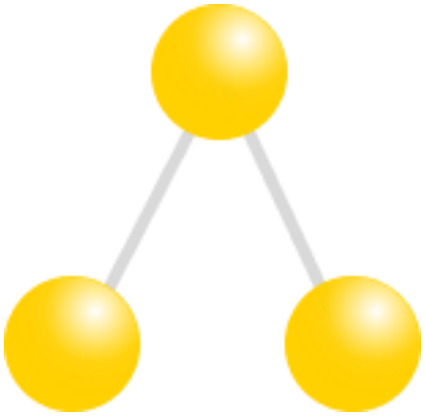

Figura 2: Representação de serviços compostos por Thomas Erl

Fonte: Autor

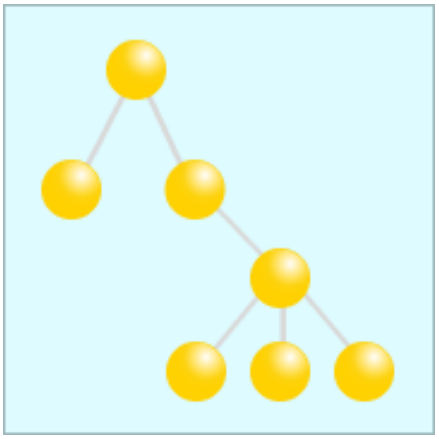

Figura 3: Representação de serviços compostos mais complexos

Fonte: Autor

O conceito de serviços compostos se alinha aos princípios fundamentais da SOA, como a reutilização, a abstração e a interoperabilidade. Ao combinar serviços individuais e reutilizáveis, os serviços compostos facilitam o desenvolvimento de soluções flexíveis e escaláveis, que podem ser facilmente adaptadas para atender às mudanças nas necessidades e requisitos de negócio.

Os serviços compostos também promovem a eficiência no desenvolvimento de sistemas de TI, pois permitem que as organizações aproveitem a lógica e as funcionalidades existentes, evitando a duplicação de esforços e a necessidade de criar componentes semelhantes do zero. Isso resulta em um ambiente de desenvolvimento mais ágil e sustentável, onde os recursos podem ser alocados com mais eficiência e as soluções podem ser entregues mais rapidamente.

1.1.2 Inventário de Serviços & Domínio de Inventário de Serviços

O inventário de serviços é um elemento crucial na Arquitetura Orientada a Serviços (SOA), responsável por gerenciar e organizar todos os serviços disponíveis em uma organização. Funcionando como um catálogo centralizado, o inventário de serviços registra, documenta e mantém serviços atualizados, facilitando a descoberta, reutilização e integração deles em várias soluções de TI.

Thomas Erl destaca a importância do inventário de serviços para a eficiência e eficácia da abordagem SOA, já que permite às organizações identificar facilmente serviços disponíveis e reutilizá-los quando necessário. Isso evita duplicação de esforços e criação de serviços redundantes, otimizando a alocação de recursos e reduzindo tempo e custo de desenvolvimento.

O inventário de serviços também é fundamental na governança SOA, estabelecendo políticas, padrões e diretrizes para orientar o desenvolvimento, implantação e gerenciamento de serviços. Isso assegura que os serviços sejam criados e mantidos seguindo melhores práticas e requisitos de negócio, melhorando a qualidade geral e consistência das soluções desenvolvidas.

Um inventário de serviços bem estruturado e gerenciado facilita a interoperabilidade e integração entre os serviços, promovendo adoção de interfaces e contratos de serviço padronizados. Isso viabiliza a comunicação e interação eficiente e previsível entre serviços, independentemente das tecnologias e plataformas subjacentes, fator crucial para o sucesso de uma arquitetura SOA.

Em paralelo ao inventário de serviços, existe o conceito de Domínio do Inventário de Serviços, que também é importante na Arquitetura Orientada a Serviços (SOA). Um domínio do inventário de serviços representa um subconjunto do inventário geral, agrupando serviços que são relevantes para uma área específica de negócio ou funcionalidade. Essa divisão ajuda a organizar e gerenciar os serviços de maneira mais eficiente, facilitando a descoberta e reutilização deles por equipes e projetos relacionados.

Ao estabelecer domínios do inventário de serviços, é possível definir políticas, padrões e diretrizes específicas para cada área de negócio, garantindo que os serviços desenvolvidos estejam alinhados às necessidades e aos objetivos de cada segmento da organização. Além disso, a divisão em domínios facilita a governança SOA, permitindo o monitoramento e a análise do desempenho e da conformidade dos serviços de acordo com as regras estabelecidas para cada domínio.

O uso de domínios do inventário de serviços também ajuda a promover a colaboração e a comunicação entre diferentes equipes e departamentos, já que cada domínio pode ser gerenciado e mantido por uma equipe responsável. Isso incentiva a troca de informações e conhecimento sobre os serviços disponíveis, otimizando o processo de desenvolvimento e integração de soluções.

Assim o "Inventário de Serviços" é um catálogo centralizado de todos os serviços em uma organização, enquanto o "Domínio de Inventário de Serviços" representa um subconjunto específico relacionado a áreas de negócio ou funcionalidades. Ambos promovem reutilização e interoperabilidade, diferindo apenas na granularidade.

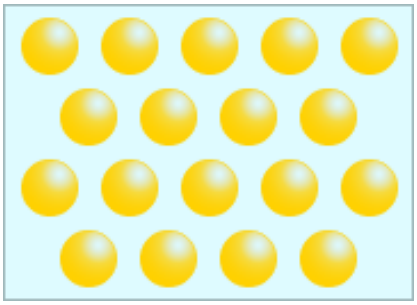

Figura 4: Representação de inventário de serviços.

Fonte: Autor

1.2 Vantagens e Desafios

Nesse subcapítulo, vamos explorar as vantagens da implementação da Arquitetura Orientada a Serviços (SOA) em comparação à arquiteturas mais antigas, no caso, a Arquitetura Baseada em Silos. A ideia central é demonstrar como a adoção do SOA pode ajudar as empresas a alcançar seus objetivos estratégicos, melhorando a eficiência, a escalabilidade e a adaptabilidade dos sistemas de TI.

Discutiremos os objetivos estratégicos das empresas e como a implementação do SOA pode auxiliar no alcance desses objetivos. Abordaremos temas como a reutilização de serviços, a interoperabilidade, a flexibilidade e a capacidade de adaptação a mudanças nos requisitos de negócio. Além disso, destacaremos casos de sucesso e exemplos práticos que ilustram o impacto positivo do SOA no desenvolvimento de sistemas de TI mais robustos e eficientes.

1.2.1 Parelelo com Arquitetura Baseada em Silos

A arquitetura baseada em silos é um modelo de desenvolvimento de sistemas em que os componentes ou módulos são projetados e implementados de forma isolada, sem considerar a integração e a colaboração com outros componentes dentro da organização. Nessa abordagem, cada componente ou aplicativo tende a ter suas próprias funcionalidades, dados e lógica de negócio, muitas vezes redundantes, resultando em um ambiente de TI fragmentado e difícil de gerenciar.

Apesar disso o correto é apontar pontos positivos dessa arquitetura, são elas:

Rapidez na construção: Como as soluções são desenvolvidas para cumprir um conjunto restrito de requisitos relacionados a processos de negócio específicos, elas podem ser construídas de maneira rápida, concentrando-se apenas no que é necessário para atender às necessidades do processo em questão.

Análise de negócios simplificada: Os analistas se concentram em apenas um processo por vez, tornando a definição do processo a ser automatizado mais direta e permitindo que se preocupem apenas com as entidades e domínios de negócios associados a esse processo específico.

Foco tático no design da solução: Com um escopo funcional pré-definido, o design geral da solução e a arquitetura de aplicativos subjacente são simplificados, facilitando a implementação de soluções automatizadas para um conjunto específico de processos de negócio.

Ciclo de vida de entrega previsível: Quando o escopo de entrega é bem definido e não muda, o ciclo de vida de entrega do projeto é simplificado e relativamente previsível, aumentando as chances de sucesso na execução das fases de entrega.

Aproveitamento de avanços tecnológicos: Construir novos sistemas a partir do zero permite que as organizações tirem proveito das últimas inovações tecnológicas, adaptando-se às mudanças no mercado e usando as mais recentes ferramentas e tecnologias disponíveis.

Apesar dos pontos positivos das aplicações baseadas em silo, elas enfrentam desafios significativos em termos de escalabilidade, manutenibilidade, integração com outros sistemas e interoperabilidade. A falta de integração e coordenação entre os componentes pode levar à duplicação de esforços, dificuldades no compartilhamento de informações e limitações na adaptação às mudanças nos requisitos de negócio. Portanto, é crucial avaliar cuidadosamente as necessidades específicas do projeto antes de optar por essa arquitetura, considerando os impactos na flexibilidade e escalabilidade das soluções de TI.

Dessa forma, é evidente que a abordagem da arquitetura orientada a serviços (SOA) pode solucionar os problemas mencionados, relacionados à arquitetura baseada em silos. Ao promover a integração, reutilização e colaboração entre os componentes, SOA proporciona uma solução eficiente e flexível para superar os desafios apresentados pelas arquiteturas tradicionais.

De certa forma, grande parte dos problemas nas arquiteturas mais antigas está relacionada à presença de lógica redundante, uma vez que cada aplicação opera de maneira isolada, levando a repetições nas implementações lógicas. Com isso em mente, podemos dividir os desafios das arquiteturas antigas em cinco categorias principais:

Alto desperdício: A arquitetura baseada em silos tende a gerar desperdício, pois cada silo opera de forma isolada, resultando em redundância de recursos e esforços que poderiam ser compartilhados entre as equipes a longo prazo e com a evolução do sistema o disperdício se torna cada vez mais impactante e relevante.

Não é eficiente do ponto de vista econômico ou financeiro: Como os silos operam separadamente, eles dificultam a otimização de recursos e podem levar a um maior custo operacional, tornando a arquitetura economicamente ineficiente além de gerar para empresa um ROI (Return On Investment) que piora a cada vez que o sistema arcaico cresce tendo em vista que novamente lógica redundante é desenvolvida.

Incha a empresa: Devido à falta de reutilização e compartilhamento de componentes, a arquitetura baseada em silos pode causar um inchaço na estrutura da empresa, com um aumento no número de equipes e recursos necessários para manter e gerenciar os sistemas.

Cria uma arquitetura complexa e eventualmente inviável: A abordagem em silos leva à criação de sistemas complexos e difíceis de gerenciar, que podem se tornar inviáveis com o tempo, devido à crescente dificuldade de manutenção e evolução.

Integrações tornam-se cada vez mais complexas: À medida que a quantidade de silos aumenta, as integrações entre eles se tornam mais complexas e desafiadoras, resultando em maiores custos de manutenção e riscos associados a falhas e incompatibilidades.

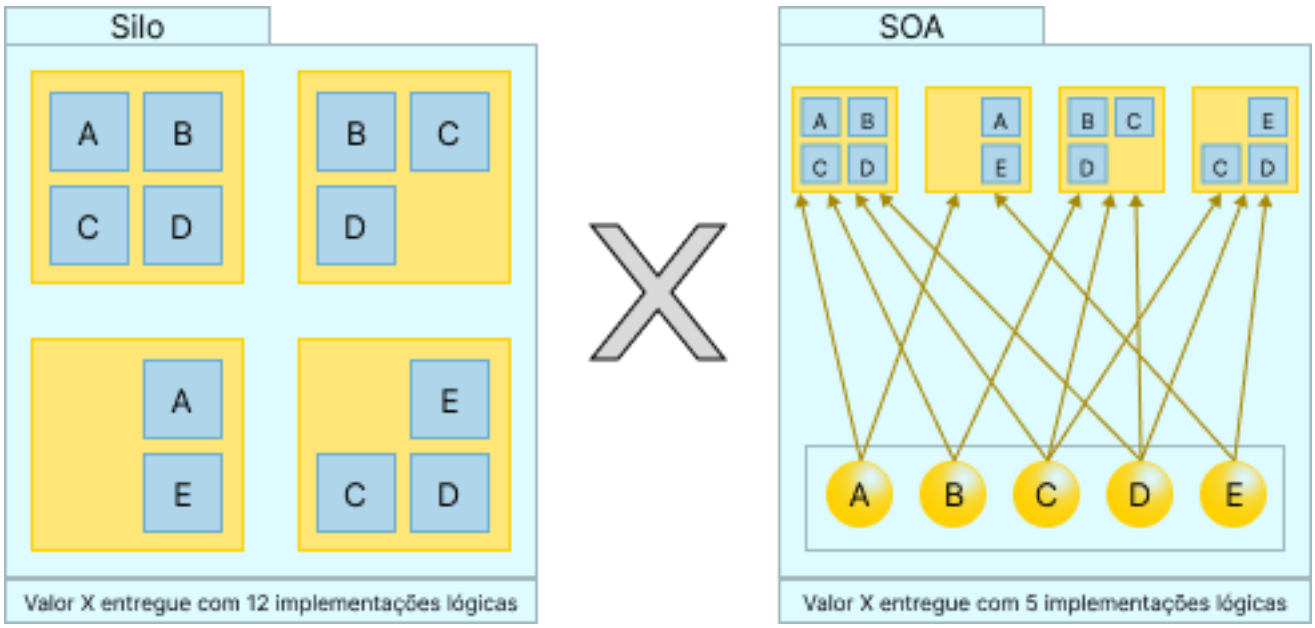

Figura 5: Comparação demonstrando potencial de ganho num cenário em que as lógica para implementação de cada funcionalidade são completamente reutilizáveis.

Fonte: Autor

Em contraste com os problemas apresentados pelas arquiteturas baseadas em silos, a Arquitetura Orientada a Serviços (SOA) traz uma série de benefícios que ajudam a superar essas limitações. Ao adotar a abordagem SOA, as organizações podem experimentar melhorias significativas em termos de eficiência, escalabilidade e adaptabilidade. A seguir, estão três vantagens da SOA que atuam como contrapartidas diretas aos problemas apresentados para as arquiteturas antigas:

Menos redundância de lógica: A abordagem da Arquitetura Orientada a Serviços (SOA) tem como princípio a reutilização de serviços, o que reduz significativamente a redundância de lógica no sistema. Isso ocorre porque os serviços são projetados para serem autônomos, independentes e facilmente integráveis, incentivando a colaboração e a compartilhamento de funcionalidades em vez de duplicá-las.

Menor volume de lógica: A adoção da SOA leva a sistemas projetados para serem modulares e flexíveis, o que resulta em um menor volume de lógica. Ao dividir as funcionalidades em serviços menores e reutilizáveis e eliminar a redundância de lógica, a quantidade de código necessário para implementar e manter o sistema é reduzida, simplificando o processo de desenvolvimento e manutenção.

Interoperabilidade inerente: A SOA garante a interoperabilidade entre os serviços por meio de contratos de serviço padronizados e a utilização de protocolos e formatos de mensagem comuns. Isso permite que os serviços se comuniquem e interajam de maneira eficiente e previsível, independentemente das tecnologias e plataformas subjacentes, facilitando a integração e a cooperação entre diferentes sistemas e aplicações.

Com essa contextualização em mente, podemos prosseguir com a análise dos objetivos estratégicos e dos benefícios proporcionados pela SOA.

1.2.2 Objetivos Estratégicos e Benefícios Estratégicos

Ao implementar a Arquitetura Orientada a Serviços (SOA), as empresas buscam alcançar uma série de objetivos estratégicos e organizacionais. A seguir, apresentamos uma explicação dos sete objetivos que podem ser atingidos por meio de uma implementação correta e adequada da arquitetura SOA:

Aumento da interoperabilidade intrínseca: Thomas Erl destaca a importância da interoperabilidade entre os serviços e sistemas em uma arquitetura SOA. Isso significa que os serviços devem ser capazes de se comunicar e trabalhar em conjunto, independentemente das tecnologias e plataformas subjacentes, facilitando a integração e a colaboração entre diferentes componentes.

Aumento da federação: No contexto de SOA, a federação refere-se à capacidade de unir serviços e sistemas distribuídos em diferentes domínios organizacionais e tecnológicos. Isso permite uma maior cooperação entre as partes e promove a reutilização de serviços, melhorando a eficiência e a escalabilidade das soluções de TI.

Aumento das opções de diversificação de fornecedores: Com a SOA, as organizações têm maior flexibilidade para escolher entre diferentes fornecedores e tecnologias, o que possibilita a diversificação e a otimização das soluções de TI. Isso permite que as empresas selecionem as melhores opções para suas necessidades específicas, sem ficarem presas a um único fornecedor ou plataforma.

Aumento do alinhamento entre domínios de negócios e tecnologia: SOA promove a colaboração entre as áreas de negócio e de TI, garantindo que os serviços sejam desenvolvidos e implementados de acordo com as necessidades e objetivos do negócio. Isso resulta em soluções mais eficientes e eficazes, que apoiam e facilitam as operações de negócios.

Aumento do retorno sobre o investimento (ROI): Ao adotar a SOA, as organizações podem aproveitar a reutilização de serviços, a interoperabilidade e a agilidade, o que leva a uma redução nos custos de desenvolvimento e manutenção. Isso, por sua vez, resulta em um aumento no retorno sobre o investimento em TI.

Aumento da agilidade organizacional: A SOA permite que as empresas respondam rapidamente às mudanças nas necessidades de negócio e no ambiente de mercado, adaptando-se e evoluindo conforme necessário. Isso proporciona maior agilidade organizacional, permitindo que as empresas se mantenham competitivas e inovadoras.

Redução do ônus de TI: A adoção da SOA simplifica o gerenciamento e a manutenção de sistemas de TI, reduzindo a complexidade e a carga de trabalho associadas. Isso permite que as equipes de TI se concentrem em atividades de maior valor, como inovação e desenvolvimento de novas soluções, em vez de gastar tempo e recursos com manutenção e integração de sistemas.

Compreendendo o paralelo entre arquiteturas mais tradicionais e a proposta da arquitetura SOA, podemos perceber que ambas têm seus pontos fortes em determinados contextos. No entanto, a SOA é mais adaptável e oferece melhores resultados em sistemas que evoluem com o tempo, acompanhando de forma mais eficaz o crescimento da empresa, a expansão dos sistemas e atendendo às integrações necessárias que surgem ao longo do tempo. Portanto, devemos buscar entender melhor os princípios e padrões existentes na SOA para aumentar a assertividade em sua implementação, permitindo assim extrair ao máximo as vantagens que essa arquitetura tem a oferecer.

1.3 Padrões e Implementações

Os padrões de implementação de SOA são um conjunto de práticas e diretrizes estabelecidas que auxiliam no projeto, desenvolvimento e gerenciamento de arquiteturas orientadas a serviços de maneira inteligente, fornecendo soluções comprovadas e bem-sucedidas para desafios comuns enfrentados durante a implementação da SOA, como design de serviço, comunicação, segurança, governança e gerenciamento de processos de negócio. Esses padrões são fundamentais para garantir a qualidade, consistência e interoperabilidade dos serviços e sistemas desenvolvidos dentro dessa arquitetura, permitindo que as organizações evitem erros comuns e melhorem a eficiência e a eficácia de suas soluções de TI, além de facilitar a integração e a colaboração entre diferentes serviços e sistemas, promovendo a reutilização de componentes e a modularidade. A seguir vamos trazer alguns conceitos para que possamos entender melhor como essa tecnologia deve ser apropriadamente implementada.

1.3.1 Princípios de design de Serviços

Para começarmos, é importante mencionar os 8 princípios dos serviços apresentados por Thomas Erl. Esses princípios atuam como diretrizes para avaliar se uma implementação pode ser considerada um serviço; quanto mais alinhada aos princípios, melhor será o design dessa implementação sob a perspectiva da orientação a serviços..

A tabela a seguir expõe de maneira resumida quais são os 8 princípios e o que significam:

Tabela 1: Resumo de cada um dos 8 princípios do design de serviços

Princípio	Descrição	Benefícios
Contrato de Serviço Padronizado	Define que os serviços devem seguir um contrato padronizado para garantir a consistência e a interoperabilidade.	Facilita a comunicação e integração, aumentando a interoperabilidade.
Acoplamento Frouxo do Serviço	Minimiza as dependências entre serviços, promovendo maior flexibilidade e facilidade de manutenção.	Promove flexibilidade e facilita a evolução dos serviços.
Abstração do Serviço	Oculta detalhes internos do serviço para simplificar a interação e proteger sua lógica de negócio.	Simplifica a interação e protege a lógica interna dos serviços.
Reutilização do Serviço	Projetar serviços para serem reutilizáveis em diferentes contextos, otimizando recursos e reduzindo custos.	Reduz o tempo de desenvolvimento e otimiza o uso de recursos.
Autonomia do Serviço	Garante que os serviços sejam independentes e autônomos, permitindo o gerenciamento e evolução individual.	Aumenta a confiabilidade, escalabilidade e manutenção dos serviços.
Serviço sem Estado	Serviços devem evitar manter informações de estado para garantir escalabilidade e melhor desempenho.	Melhora o desempenho, a escalabilidade e a simplicidade dos serviços.
Descoberta dos Serviços	Facilita a identificação e a localização de serviços através de metadados e descrições claras.	Aumenta a eficiência na utilização e integração dos serviços.
Composição de Serviços	Permite que serviços sejam combinados e orquestrados para criar soluções mais complexas e personalizadas.	Potencializa a colaboração entre serviços e a criação de soluções.

1.3.2 Pilares Culturais da Orientação a Serviços

Os pilares culturais não estão relacionados diretamente à aplicação técnica ou aos requisitos específicos para que a Orientação a Serviços funcione, mas sim aos conceitos que a empresa deve adotar para aumentar as chances de sucesso na implementação. A sinergia entre equipes, uma cultura voltada para serviços e disciplina na execução de uma implementação coesa e consistente são fundamentais para criar um contexto organizacional onde as pessoas estejam aptas a trabalhar com a SOA.

Trabalho em equipe: Diferentemente de alguns modelos usuais de times e equipes, em que são definidos e separados por funcionalidades ou implementações, em uma empresa que busca migrar para a orientação a serviços, é essencial que os times se integrem cada vez mais. Principalmente em um contexto em que não exista o conceito de aplicação e sim de inventário de serviços, é necessário que o empenho dos times seja integrado para que, por sua vez, os serviços sejam integrados e assim cumpram seus requisitos técnicos.

Educação: Para esse pilar, existem dois pontos em destaque: o conhecimento de serviços e o conhecimento de negócios. Para que a equipe implemente adequadamente serviços e suas composições, é de fato necessário que saibam a técnica e a teoria para uma implementação assertiva. Porém, é também essencial que a equipe tenha domínio do contexto de negócio em que se aplicam. Em última instância, é necessário compreender que os serviços não existem por si só e sim para executar processos de um negócio. E, para assegurar que o serviço que vai atender esse processo tenha um bom design, é necessário que o time responsável pela construção do serviço também tenha domínio do negócio.

Disciplina: Diz respeito à necessidade de consistência. A migração para orientação a serviços pode ser demorada, e em contraste com metodologias tradicionais, seus ganhos não são imediatos. A arquitetura exige um bom design e planejamento antes da implementação, e as equipes eventualmente só conseguirão enxergar os benefícios e ganhos conforme a demanda escalar e o sistema necessitar de melhorias. Até lá, é necessário que o time se mantenha engajado na construção de bons serviços para que todo o esforço faça sentido.

Escopo equilibrado: Por fim, o escopo equilibrado é sobre contexto. De maneira genérica e abstrata, entender que a adoção da SOA deve ocorrer de forma natural e respeitando as peculiaridades e exclusividades da organização em si. Entender que variáveis externas podem e vão afetar a adoção em diferentes regiões do planeta pelos mais diversos motivos. O importante é que tudo isso seja levado em conta e que os serviços acompanhem esses contextos.

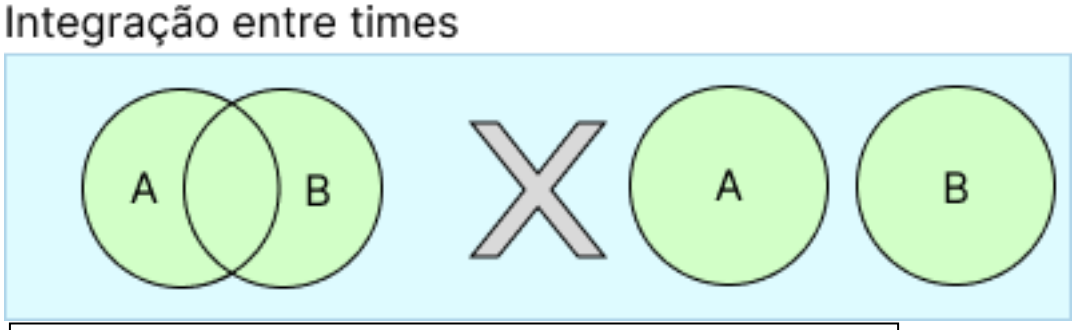

Figura 6: Representação de integração entre times.
Fonte: Autor

1.3.3 Pilares de negócios da Orientação a Serviços

Agora que analisamos os fatores culturais que afetam a implementação da arquitetura orientada a serviços, podemos abordar os fatores de negócio que devem ser levados em conta durante a implementação da arquitetura orientada a serviços. O direcionamento à área de negócios e à empresa, a independência de fornecedores e a centralização na composição são premissas que devem ser seguidas para que a orientação a serviços seja adequadamente implementada. Vejamos a seguir uma breve explicação desses pontos:

Direcionamento aos negócios: É comum que as implementações tecnológicas busquem solucionar demandas em nível operacional (curto prazo) ou tático (médio prazo), mas poucas vezes têm um direcionamento estratégico (longo prazo). É isso que esse pilar busca abordar, principalmente tendo em vista os benefícios a longo prazo da orientação a serviços. O design da arquitetura deve prever e estar alinhado com o nível estratégico, evitando que, com o tempo, a estrutura do inventário de serviços esteja desalinhada com as premissas da empresa, o que resultaria em um esforço na alteração dos serviços.

Direcionamento às empresas: Esse pilar serve para lembrar que o fato de uma empresa adotar a SOA não a impede de criar novos "silos de serviço". Portanto, é essencial que os serviços sejam modelados para a empresa e não para funcionalidades específicas dentro dela. Cabe aos desenvolvedores fazer com que esses serviços criados estejam disponíveis para a maior parte possível da organização (sempre dentro do contexto do domínio).

Independência de fornecedores: A implementação deve ser independente de fornecedores ou terceiros, pelos dois principais motivos: o primeiro é não se limitar a eventuais recursos e dinâmicas desses terceiros, pois o contexto do inventário de serviços deve ser a organização. O segundo motivo é que, caso ocorra uma dependência de um fornecedor, a empresa pode passar a ter que acompanhar a evolução desse fornecedor, assumindo sua identidade e funcionamento.

Centrado na composição: Esse pilar traz a mentalidade de que o serviço é um recurso empresarial e deve estar disponível para o uso da maior parte possível da empresa (sendo idealmente disponível para toda a empresa). Para isso, é necessário que os serviços estejam em dia com sua estrutura focada em composição. É recomendado, inclusive, uma composição forçada entre serviços não relacionados ou que não necessariamente precisem funcionar juntos, isso para assegurar a disponibilidade e a eficiência na utilização dos recursos.

1.3.4 Tipos Arquitetônicos na Orientação a Serviços

O design arquitetônico de um **serviço** na SOA é mais complexo e envolve mais elementos de implementação em comparação a outros tipos de arquitetura, pois sua representação precisa incluir todos os processos, como entradas, saídas e detalhes do contrato de serviço. Este contrato é um componente essencial da SOA, pois define a interface pública de um serviço da web, especificando métodos disponíveis, tipos de dados de entrada e saída e informações sobre o protocolo de comunicação. A seguir vamos entender um pouco mais sobre esse assunto técnicamente.

O WSDL (Web Services Description Language) é uma linguagem baseada em XML que descreve as funcionalidades de um serviço da web, fornecendo uma interface padronizada, incluindo métodos, parâmetros e tipos de dados. Atuando como um contrato entre provedor e consumidores, o WSDL facilita a comunicação e a interoperabilidade entre sistemas e plataformas, permitindo que desenvolvedores gerem automaticamente código cliente e servidor em diversas linguagens de programação para simplificar a implementação e integração de serviços da web.

Os principais elementos do contrato de serviço expressos por meio de um WSDL incluem:

Tipos de dados (wsdl:types): O elemento "types" define os tipos de dados usados nas mensagens de entrada e saída do serviço da web. Ele utiliza o XML Schema Definition (XSD) para descrever a estrutura e a validação dos tipos de dados.

Mensagens (wsdl:message): O elemento "message" descreve as mensagens de entrada e saída para cada operação no serviço da web. Cada mensagem pode conter um ou mais "part"s, que são associações aos elementos definidos no XML Schema.

Operações (wsdl:operation): As operações são definidas dentro do elemento "portType" e representam as ações disponíveis no serviço da web. Cada operação possui elementos de entrada e saída associados a mensagens específicas previamente definidas.

Protocolo e formato de mensagem (wsdl:binding): O elemento "binding" especifica o protocolo de comunicação (como SOAP) e o formato das mensagens (como "document" ou "rpc"). Ele também associa as operações às regras de formatação e transporte, como o atributo "soapAction" no caso do protocolo SOAP.

Endereço do serviço (wsdl:service): O elemento "service" define o endereço do serviço da web, fornecendo informações de localização específicas, como a URL do endpoint. Ele também associa o "port" ao "binding" apropriado.

Em resumo, o WSDL permite expressar os elementos do contrato de serviço de maneira estruturada e padronizada, facilitando a comunicação entre diferentes sistemas e a geração automática de código para clientes e provedores de serviços da web. Ele desempenha um papel fundamental na garantia da interoperabilidade e na promoção da reutilização e modularidade em ambientes SOA.

A seguir você poderá ver um código em XML para montar um WSDL de um serviço genérico para fins didáticos.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:tns="http://example.com/sampleservice"
                  targetNamespace="http://example.com/sampleservice">
  <!-- Define os tipos de dados usados no serviço -->
  <wsdl:types>
    <xsd:schema targetNamespace="http://example.com/sampleservice">
      <xsd:element name="inputParam" type="xsd:string"/>
      <xsd:element name="outputParam" type="xsd:string"/>
    </xsd:schema>
  </wsdl:types>
  <!-- Define as mensagens de entrada e saída -->
  <wsdl:message name="SampleOperationRequest">
    <wsdl:part name="parameters" element="tns:inputParam"/>
  </wsdl:message>
  <wsdl:message name="SampleOperationResponse">
    <wsdl:part name="parameters" element="tns:outputParam"/>
  </wsdl:message>
  <!-- Define as operações disponíveis no serviço -->
  <wsdl:portType name="SampleServicePortType">
    <wsdl:operation name="SampleOperation">
      <wsdl:input message="tns:SampleOperationRequest"/>
      <wsdl:output message="tns:SampleOperationResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- Define o protocolo de comunicação e o formato de mensagem -->
  <wsdl:binding name="SampleServiceBinding" type="tns:SampleServicePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="SampleOperation">
      <soap:operation soapAction="http://example.com/sampleservice/SampleOperation"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <!-- Define o endereço do serviço e associa-o ao binding -->
  <wsdl:service name="SampleService">
    <wsdl:port name="SampleServicePort" binding="tns:SampleServiceBinding">
      <soap:address location="http://example.com/sampleservice/endpoint"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

O arquivo .xsd (XML Schema Definition), outro elemento comum no design de serviços, descreve a estrutura e as regras de validação de um documento XML, definindo elementos, atributos, tipos de dados e restrições. Essa "blueprint" é expressa na linguagem XML Schema Definition Language, um padrão criado pelo World Wide Web Consortium (W3C). O uso do arquivo .xsd garante que os documentos XML sigam um padrão específico e consistente, facilitando a interoperabilidade entre sistemas e aplicações. Além disso, o XML Schema permite gerar automaticamente classes e objetos em várias linguagens de programação, simplificando a manipulação e processamento de documentos XML.

A seguir você poderá ver um código em XML para montar um XSD de um serviço genérico para fins didáticos.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.exemplo.com/servico"
xmlns:tns="http://www.exemplo.com/servico" elementFormDefault="qualified">

  <!-- Definição de tipos complexos -->
  <xs:complexType name="Solicitacao">
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="mensagem" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Resposta">
    <xs:sequence>
      <xs:element name="status" type="xs:string"/>
      <xs:element name="mensagem" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Definição de elementos -->
  <xs:element name="enviarMensagemRequest" type="tns:Solicitacao"/>
  <xs:element name="enviarMensagemResponse" type="tns:Resposta"/>

</xs:schema>
```

Em resumo, tanto o XSD quanto o WSDL são arquivos que, por meio da linguagem de marcação XML, estabelecem padrões e regras dentro do contexto em que se aplicam, ou seja, os serviços. Além de todos os benefícios, como escalabilidade, disponibilidade e integração, esses documentos bem desenvolvidos podem proporcionar automação para a criação de outros serviços que podem e devem enxergar os campos nesses documentos como parâmetros a serem replicados.

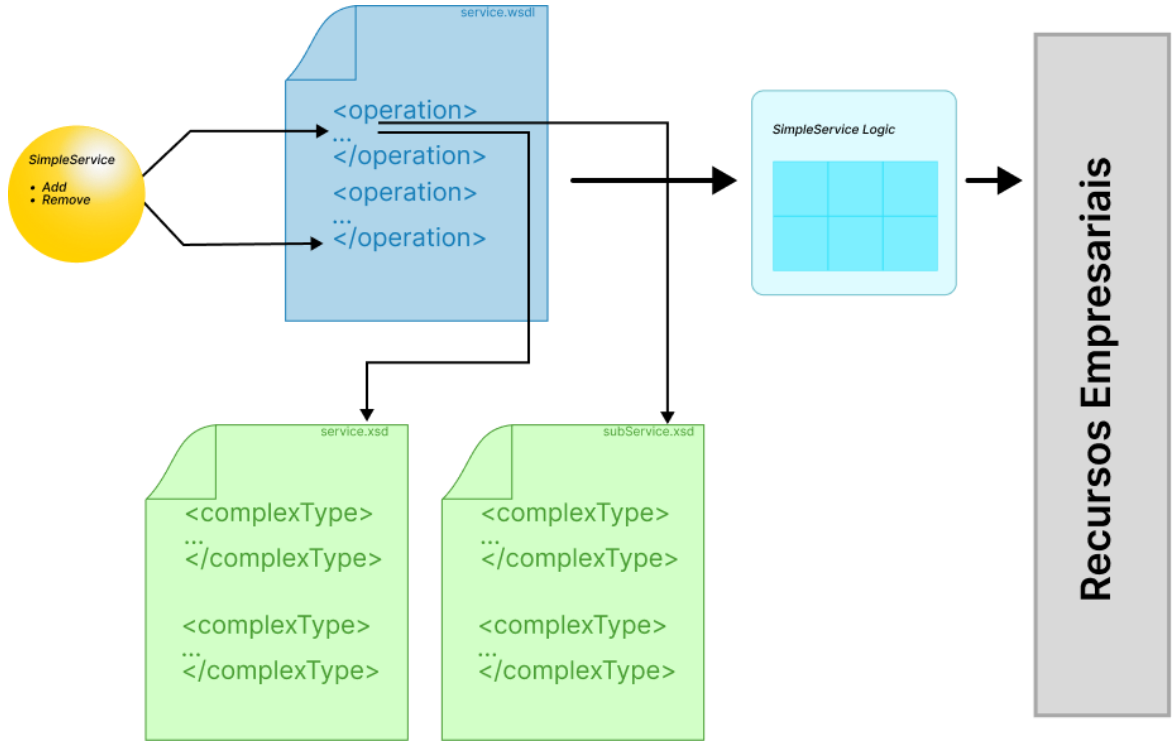

Figura 7: Relação entre o serviço, seu contrato presente no arquivo WSDL, com as operações devidamente definidas e com regras parametrizadas por meio dos arquivos XSD. Em cima disso, é construída uma lógica que acessa os recursos empresariais para ser executada.
Fonte: Autor.

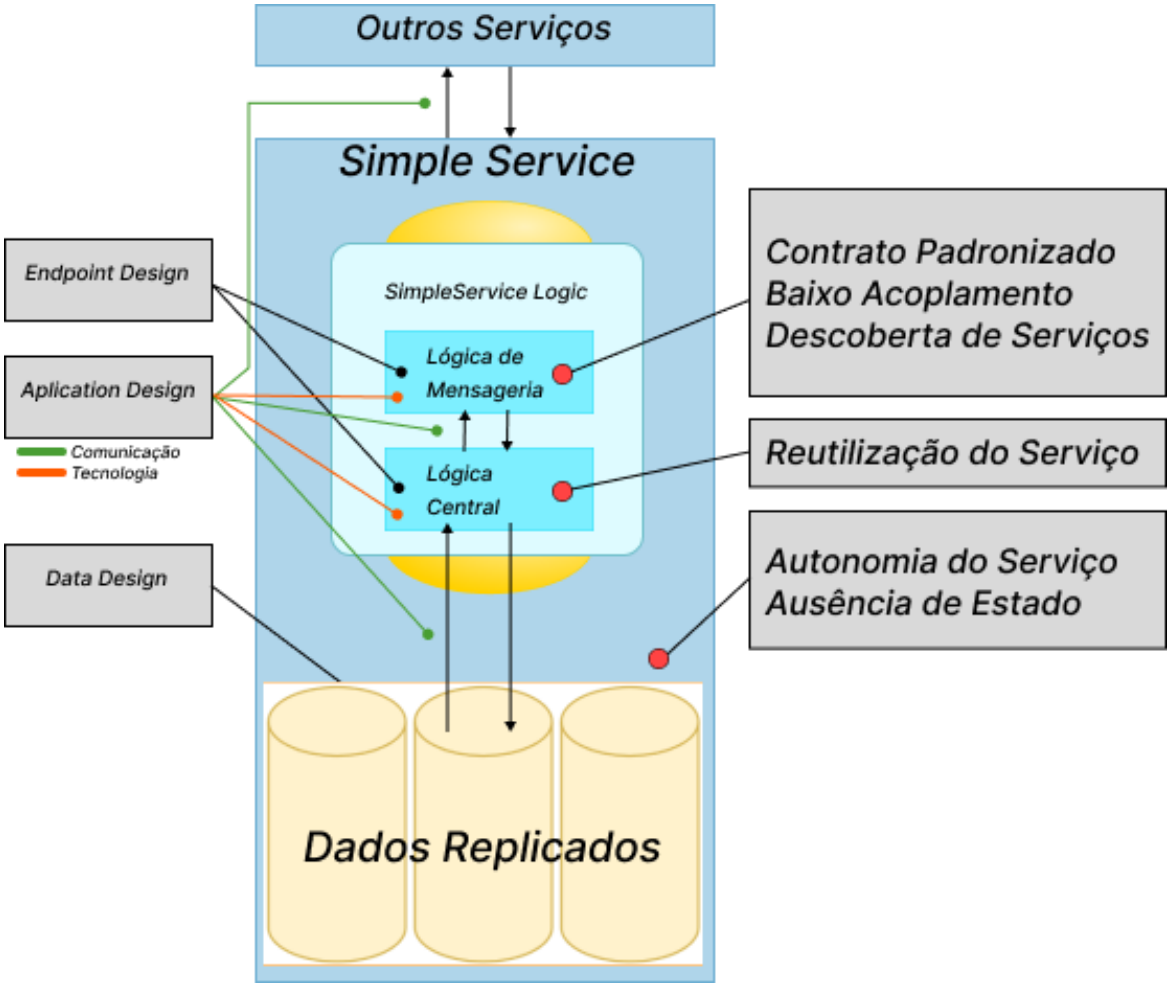

Figura 8: Demonstração de elementos dos design de componentes, quais princípios são afetados por cada componentes e divisão do componente a nível de dados, aplicação e mensageria.
Fonte: Autor.

O design da composição de serviços envolve muito mais do que apenas conectar serviços individuais. Ele abrange a orquestração, o gerenciamento e a coordenação das interações entre os serviços, garantindo que eles trabalhem em conjunto para alcançar os objetivos de negócio desejados. Além disso, o design de composições de serviços também deve abordar o tratamento de caminhos de exceção, garantindo que o sistema seja capaz de lidar com falhas e erros de maneira adequada e eficiente.

A orquestração desempenha um papel fundamental no design da composição de serviços, pois é responsável pela definição do fluxo de controle e das dependências entre os serviços envolvidos. Isso pode incluir a execução sequencial ou paralela de serviços, a implementação de padrões de interação, como publicar/assinar ou solicitar/responder, e a coordenação das transações e dos processos de negócio. A orquestração pode ser realizada por meio de tecnologias e linguagens específicas ou utilizando soluções baseadas em microsserviços e arquiteturas orientadas a eventos.

O tratamento de caminhos de exceção é igualmente crucial no design da composição de serviços, pois garante a resiliência e a confiabilidade do sistema. Um bom design deve levar em consideração possíveis falhas e erros, implementando mecanismos de tratamento de exceções e estratégias de recuperação para lidar com situações adversas. Isso pode incluir a captura e o tratamento de exceções específicas, a propagação de exceções para um nível superior, a implementação de políticas de retentativa, o uso de roteamento alternativo e o monitoramento e registro de eventos de exceção.

Em suma, o design da composição de serviços é um processo complexo e multidimensional que requer uma compreensão profunda das interações entre os serviços e das possíveis falhas e erros que podem ocorrer durante a execução. Um design eficiente de composições de serviços deve equilibrar a orquestração e o tratamento de caminhos de exceção para criar soluções robustas, escaláveis e resilientes, capazes de atender às necessidades de negócio em constante evolução.

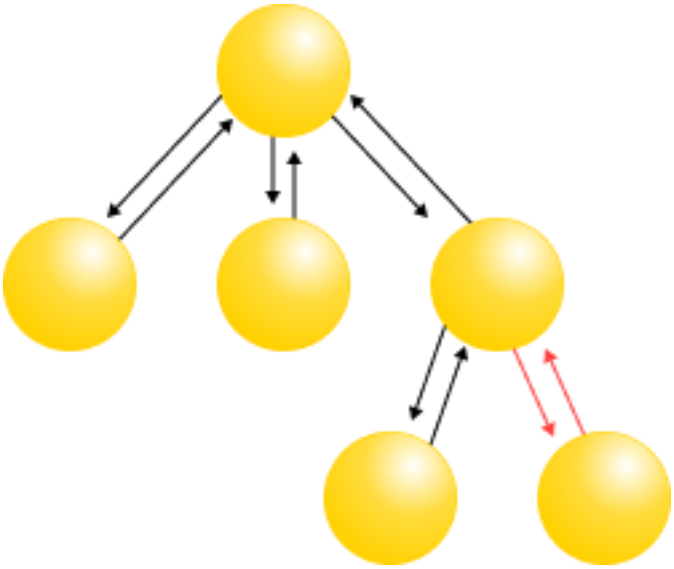

Figura 9: Serviços compostos com definição de fluxos, assim como fluxos de exceção.
Fonte: Autor

O design de inventário de serviços, parte fundamental da arquitetura orientada a serviços (SOA), visa padronizar e garantir a consistência dos serviços e suas composições para maximizar o reuso e facilitar a integração. Os arquitetos estabelecem diretrizes e padrões claros abordando aspectos como nomenclatura, estrutura das interfaces, protocolos de comunicação e formatos de dados, promovendo a reutilização e interoperabilidade. A padronização simplifica a composição de serviços e cria soluções flexíveis e ágeis, adaptáveis às mudanças nos requisitos de negócio e ambiente de TI. Além disso, promove a governança e o controle de qualidade, melhorando o desempenho e a confiabilidade dos serviços.

Para isso, podemos propor quatro etapas, que, por meio da iteração entre elas, é possível gerar de maneira incremental um projeto de inventário de serviços, sempre atualizado com o contexto da organização. As quatro etapas são:

Definir modelo de negócio empresarial: Nesta etapa é necessário entender o escopo da organização, assim como os objetivos de automação a serem atingidos por meio dos serviços.

Definir arquitetura tecnológica: Nesta etapa deve ser analisado os fatores tecnológicos que afetam os serviços, assim como sua composição. Deve ser sempre buscado tecnologias e métodos que possibilitem que os princípios dos serviços sejam atendidos.

Definir "blueprint" do inventário de serviços: Deve ser decidida uma população que deve ser integrada no inventário. A cada iteração, esse inventário pode crescer ou, dependendo do contexto, ser dividido em novos domínios. O essencial é não permitir que o inventário se torne um silo, com baixa integração e pouco reuso.

Realizar a análise de orientação a serviço: A etapa final da iteração, depois que todo o macro foi definido, envolve, de modo simplificado, a modelagem dos serviços, assim como o máximo de composições possíveis dentro daquilo que foi estabelecido dentro do inventário.

Por fim, a arquitetura corporativa orientada a serviços é o nível mais macro de abstração de arquitetura sendo de modo genérico o modelo de gerenciamento da SOA na organização, assim como os padrões que devem ser adotados pelos inventários, composições e pelos serviços. Essa arquitetura pode definir os limites da orientação a serviços na organização, assim como documentar quais faces da empresa se encontram no modelo orientado a serviços.

1.3.5 Conclusão da Orientação a Serviços

Com isso, já é possível ter uma compreensão abrangente da orientação a serviços, tanto nos níveis mais teóricos, com os princípios e modelos que devem ser seguidos, quanto nos níveis mais técnicos, com a compreensão do design de serviços e dos elementos técnicos relacionados ao seu desenvolvimento, como a criação de um contrato de serviço por meio do WSDL e descrições de regras e validações por meio dos arquivos XSD.

A seguir vamos abordar outra tecnologia que vem se desenvolvendo e ganhando cada vez mais relevancia e mais espaço no mundo de soluções em nuvem, com esse conhecimento poderemos então integrar as duas tecnologias em um conceito final.

2. Software as a Service

Neste capítulo, vamos nos aprofundar no tema do Software as a Service (SaaS). Entretanto, para que possamos compreender melhor este modelo, é necessário, primeiramente, estabelecer um entendimento mínimo sobre a base fundamental na qual ele se apoia, a computação em nuvem. É com base nesse conceito, que a SaaS se estrutura e toda sua fundamentação é capacitada por essa tecnologia.

Primeiramente, vamos explorar a definição e o propósito da computação em nuvem, traçando seu contraste com os tradicionais sistemas de TI "on-premise" e destacando as razões que levam as organizações a optarem por soluções baseadas em nuvem. Em seguida, nos aprofundaremos nos três paradigmas da computação em nuvem, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) e, finalmente, Software as a Service (SaaS). Esta tríade forma uma estrutura hierárquica, na qual cada camada se constrói sobre a anterior, adicionando mais funcionalidades e abstrações.

Ao entendermos esses conceitos-chave e a relação entre eles, estaremos preparados para analisar o SaaS em profundidade, explorando seus aspectos técnicos e de negócios e o seu papel dentro da arquitetura orientada a serviços (SOA). Assim, nos equiparemos com o conhecimento necessário para aproveitar ao máximo as possibilidades oferecidas por este modelo de prestação de serviços por meio de páginas web.

2.1 O Básico de Cloud Computing

A computação em nuvem, ou "Cloud Computing", é um modelo que visa redefinir a maneira como acessamos e utilizamos recursos computacionais. Basicamente, a computação em nuvem permite o acesso generalizado, conveniente e sob demanda a uma variedade de recursos, como redes, servidores, armazenamento, aplicações e serviços, que podem ser rapidamente provisionados e liberados com mínimo esforço de gestão. A computação em nuvem é caracterizada por cinco características essenciais, são elas, **autoatendimento sob demanda, amplo acesso à rede, agrupamento de recursos, elasticidade rápida e serviço mensurável**. Além disso a Cloud Computing conta com quatro modelos de implantação, **nuvem privada, nuvem comunitária, nuvem pública e nuvem híbrida** comumente enquadrados em algum dos três modelos de serviço (**SaaS, PaaS e IaaS**).

2.1.1 Cinco características de Cloud Computing

Autoatendimento sob demanda: Este é um dos principais atrativos da computação em nuvem. Os usuários podem provisionar recursos de computação, como tempo de processador, armazenamento de rede ou capacidades de software, conforme necessário e sem a necessidade de interação humana com o provedor de serviços. Isso permite que as empresas escalonem seus recursos de TI para cima ou para baixo rapidamente, conforme as necessidades mudam.

Amplo acesso à rede: Os serviços em nuvem estão disponíveis através da internet, o que significa que eles podem ser acessados de qualquer lugar e em qualquer dispositivo que tenha uma conexão à internet. Isso é essencial para suportar a mobilidade e o trabalho remoto, além de facilitar a colaboração entre equipes geograficamente dispersas.

Agrupamento de recursos: Na computação em nuvem, os recursos de processamento, armazenamento e rede são agrupados e servidos a múltiplos usuários. Este modelo de "multitenancy" permite que os recursos sejam compartilhados eficientemente, com cada usuário tendo acesso a uma "fatia" dos recursos totais. Isso permite economias de escala e reduz o custo por usuário.

Elasticidade rápida: A capacidade de escalar recursos de forma rápida e eficaz é outra característica-chave da computação em nuvem. As empresas podem expandir ou reduzir os recursos quase instantaneamente, de acordo com as demandas do negócio. Isso é particularmente útil para lidar com picos de demanda ou para suportar o crescimento do negócio.

Serviço Mensurável: Na computação em nuvem, o uso dos recursos é monitorado, controlado e relatado, proporcionando transparência tanto para o provedor quanto para o usuário do serviço. Isso permite que os usuários paguem apenas pelos recursos que realmente usam e ajuda os provedores a alocar recursos de forma mais eficiente. Além disso, a mensuração permite que as empresas monitorizem e ajustem seu uso de recursos de acordo com suas necessidades.

2.1.2 Modelos de Implantação de Nuvem

Nuvem Privada: A nuvem privada é uma infraestrutura de computação em nuvem dedicada a uma única organização. Os recursos de computação não são compartilhados com outras organizações. As nuvens privadas podem ser hospedadas no local (no data center da própria empresa) ou podem ser hospedadas por um provedor de serviços terceirizado. O principal benefício é a segurança melhorada e o controle sobre os dados e aplicações, pois o acesso e o uso dos recursos podem ser gerenciados e restritos.

Nuvem Comunitária: A nuvem comunitária é uma infraestrutura de computação em nuvem compartilhada por várias organizações que têm interesses comuns ou requisitos regulatórios semelhantes. Por exemplo, bancos ou agências governamentais podem escolher esse modelo para beneficiar-se da computação em nuvem, mantendo a conformidade regulatória. Ela permite o compartilhamento de custos entre os membros, enquanto ainda oferece um nível maior de privacidade, segurança e/ou conformidade com políticas que um modelo de nuvem pública.

Nuvem Pública: A nuvem pública é uma infraestrutura de computação em nuvem onde os serviços são entregues através da internet e compartilhados entre os usuários. Os provedores de nuvem pública (como Amazon Web Services, Google Cloud, e Microsoft Azure) possuem e operam os recursos de hardware e eles são responsáveis por todo o gerenciamento e manutenção do sistema. O principal benefício deste modelo é a facilidade de escalabilidade e o pagamento pelo uso, que reduz a necessidade de investimento inicial em infraestrutura de TI.

Nuvem Híbrida: A nuvem híbrida combina elementos dos modelos de nuvem pública, privada e comunitária. Permite que uma organização utilize a nuvem pública para recursos não sensíveis (como o desenvolvimento de aplicações, por exemplo) e mantenha os dados sensíveis ou críticos para os negócios em uma nuvem privada ou comunitária. Isso permite flexibilidade e eficiência, permitindo que a organização aproveite os benefícios da nuvem pública, mantendo o controle sobre os dados e aplicações críticas.

2.1.3 Camadas de Serviço de Cloud Computing

Entendendo um pouco de computação em nuvem, podemos entrar em nossa análise aprofundada do Software as a Service (SaaS), mas antes é essencial reconhecer a estrutura fundamental que sustenta este modelo. As camadas de serviço da computação em nuvem é comumente estruturada em três níves hierárquicos, na qual cada camada, ilustrada em forma de pirâmide, representa um tipo de serviço. A hierarquia, constituída por Infrastructure as a Service (IaaS), Platform as a Service (PaaS) e SaaS, configura o ecossistema da computação em nuvem. Cada camada se estabelece sobre a anterior, adicionando abstrações e funcionalidades incrementais. Tal estrutura é crucial para a compreensão do SaaS, realçando como este modelo depende e aproveita das camadas subjacentes para que consiga prover seus serviços e de maneira eficiente e eficaz.

Infrastructure as a Service (IaaS):

Infraestrutura como Serviço é um modelo de serviços de computação em nuvem que oferece infraestrutura de TI virtualizada, acessível e escalável por meio da internet. Neste modelo, servidores, armazenamento e outros dispositivos periféricos são disponibilizados ao consumidor como serviços sob demanda.

O principal objetivo do IaaS é fornecer uma infraestrutura de TI flexível e escalonável que pode ser adaptada rapidamente para atender às demandas em constante mudança do negócio. Isto elimina a necessidade de investimento em hardware caro e a manutenção associada a ele, permitindo às empresas focar em seus principais negócios em vez de se preocuparem com questões de infraestrutura de TI.

Comparado ao modelo on-premise, onde a infraestrutura de TI é de propriedade da empresa e localizada fisicamente em suas instalações, o IaaS oferece várias vantagens. Entre elas, destaca-se a redução de custos, uma vez que o IaaS elimina a necessidade de comprar, gerenciar e manter servidores físicos e outros equipamentos de data center. A infraestrutura é fornecida como um serviço pago conforme o uso, o que pode resultar em economia significativa.

Outra vantagem do IaaS é a mitigação de riscos associados à infraestrutura física. Como a infraestrutura é gerenciada pelo provedor de serviços, a responsabilidade pela manutenção, segurança e atualizações recai sobre o provedor, não sobre a empresa que contrata o serviço. Isso também inclui a responsabilidade por quaisquer interrupções de serviço ou falhas de hardware, reduzindo a exposição da empresa a tais riscos.

Além disso, o IaaS oferece alta escalabilidade, permitindo que as empresas aumentem ou diminuam sua infraestrutura de TI conforme necessário, de acordo com as demandas do negócio. Isso oferece uma grande flexibilidade, permitindo que as empresas se adaptem rapidamente a mudanças no ambiente de negócios.

Platform as a Service (PaaS):

A Plataforma como Serviço é um modelo de computação em nuvem que oferece aos desenvolvedores um ambiente na nuvem para criar, testar e implantar aplicações. Essa plataforma é utilizada tanto por desenvolvedores quanto por implementadores e possui uma arquitetura altamente escalável de várias camadas, como exemplificado pelos serviços da Azure e Salesforce.com.

Diferentemente do Software como Serviço (SaaS), que hospeda apenas aplicações completas na nuvem, o PaaS fornece uma plataforma de desenvolvimento tanto para aplicações finalizadas quanto para aquelas ainda em progresso. Ele oferece um ambiente onde os desenvolvedores podem criar e implementar aplicações sem necessariamente precisar saber a quantidade de memória e processadores que a aplicação irá usar.

O modelo PaaS oferece benefícios aos desenvolvedores ao entregar uma plataforma para desenvolver desenvolver o ciclo de vida completo do software, desde o planejamento até o design, construção da aplicação, implantação e manutenção. Ele oferece um nível de abstração mais alto, permitindo que o consumidor crie o software usando as ferramentas e/ou bibliotecas fornecidas pelo provedor.

O consumidor também controla a implantação do software e as configurações, enquanto o provedor fornece as redes, servidores, armazenamento e outros serviços necessários para hospedar a aplicação do consumidor. As ofertas PaaS facilitam a implantação de aplicações sem o custo e a complexidade de comprar e gerenciar o hardware e o software subjacentes e provisionar as capacidades de hospedagem.

O modelo Plataforma como Serviço (PaaS) traz benefícios significativos para as empresas. Ele permite a redução de custos, já que os investimentos em hardware, instalações de data center e manutenção são gerenciados pelo provedor de serviços. Além disso, proporciona escalabilidade sem a complexidade de ajustar a infraestrutura física, aumentando a velocidade e a agilidade nas operações, já que as equipes de desenvolvimento se concentram em suas atividades, e não na infraestrutura. O PaaS dá acesso a tecnologias avançadas sem custos adicionais, permitindo que as empresas se concentrem em suas competências principais. A responsabilidade pela infraestrutura é do provedor, o que inclui a segurança, manutenção e atualizações. Por fim, soluções robustas de continuidade de negócios e recuperação de desastres estão inclusas, garantindo a operação mesmo em situações adversas.

Simplificando, o PaaS é um modelo de serviço de computação em nuvem que facilita e otimiza o desenvolvimento de aplicações, removendo a necessidade de gerenciamento de infraestrutura e oferecendo uma série de benefícios em relação ao modelo on-premise.

Software as a Service (SaaS):

Tendo em vista que o foco deste capítulo é o Software as a Service (SaaS), vamos expandir, a seguir, a abordagem desse tema em relação ao IaaS e PaaS. Para começar, vamos estabelecer uma compreensão clara do conceito de SaaS. Em seguida, analisaremos os diferentes modelos de entrega de SaaS, observando suas particularidades. Avançaremos para explorar as camadas do SaaS, para entender como esses serviços são estruturados. Depois, iremos examinar os níveis de maturidade da arquitetura SaaS, que refletem o seu progresso e evolução. E para concluir, vamos trazer alguns exemplos práticos de SaaS para ilustrar nossa explicação.

Figura 10: Conforme aumentamos o nível do serviço em cloud computing reduzimos a responsabilidade da empresa contratante do serviço e aumentamos a responsabilidade dos fornecedores da solução em cloud.
Fonte: Autor

2.2 A arquitetura Multi Inquilino

A arquitetura multitenante pode ser vista como um prédio de apartamentos no qual cada inquilino (ou usuário) ocupa sua própria unidade (ou espaço de trabalho digital), mas todos compartilham a mesma infraestrutura subjacente (o prédio e suas instalações). Cada inquilino tem seu próprio espaço privado e seguro que é isolado e invisível para os outros inquilinos, assim como um apartamento é separado e privado em relação aos outros no mesmo prédio.

Neste cenário, o prédio de apartamentos representa o aplicativo de software, e cada unidade de apartamento individual representa uma "instância" do software que é exclusiva para cada inquilino. Assim como no prédio, todos os inquilinos compartilham as infraestruturas comuns, como o sistema elétrico, a água e os serviços de manutenção, na arquitetura multitenante, todos os inquilinos compartilham os mesmos recursos de hardware, banco de dados e aplicativo.

Continuando com a analogia do prédio, cada inquilino paga aluguel (ou uma taxa de assinatura) com base no espaço que ocupa e nos recursos que usa. Da mesma forma, em um ambiente de software multilocatário, os usuários normalmente pagam uma taxa de assinatura com base em seu nível de uso, número de usuários ou volumes de dados gerenciados dentro do aplicativo.

A grande vantagem da arquitetura multilocatária para os provedores de software é a eficiência e a economia de escala. Em vez de ter que manter e atualizar várias instâncias separadas do software para cada cliente (como em uma arquitetura de inquilino único), eles podem manter e atualizar uma única instância que serve a todos os inquilinos. Isso geralmente resulta em custos operacionais mais baixos, que podem ser repassados aos clientes na forma de preços mais baixos.

Além disso, assim como um inquilino de um prédio não precisa se preocupar com a manutenção do prédio (que é responsabilidade do proprietário ou da administração do prédio), os inquilinos em uma arquitetura de software multilocatária não precisam se preocupar com a manutenção do software. As atualizações, patches e novos recursos são adicionados pelo provedor do software, liberando o inquilino para se concentrar em suas próprias tarefas e operações comerciais.

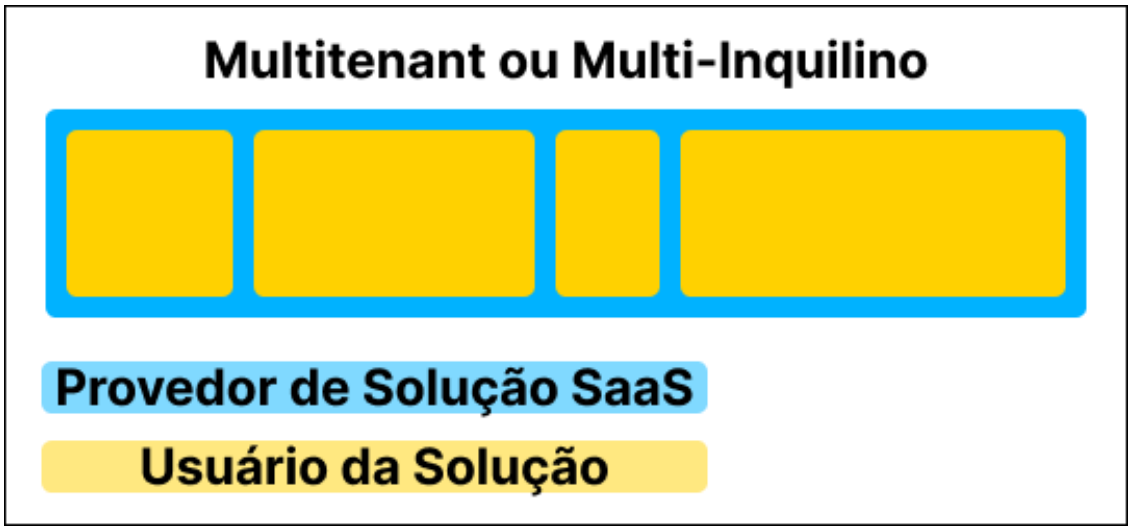

Figura 11: A estrutura em azul representa uma única instância da solução do provedor de software. Dentro dessa estrutura, diversos usuários podem adquirir partes dessa instância ao mesmo tempo. Todo esse processo é invisível, o que significa que nenhum usuário é capaz de acessar a aplicação ou os dados de outro usuário.
Fonte: Autor

2.3 Conceitos de Software as a Service

Software as a Service (SaaS) é um modelo de distribuição de software em que os aplicativos são hospedados por um provedor de serviços e disponibilizados aos clientes através de uma rede, geralmente a Internet. Isso significa que, em vez de os usuários terem que instalar e manter o software em seus próprios dispositivos ou servidores, eles podem acessá-lo de qualquer lugar com uma conexão à Internet.

A popularidade do SaaS vem crescendo à medida que as tecnologias subjacentes, como os serviços web e a Arquitetura Orientada a Serviços (SOA), amadurecem e novas abordagens de desenvolvimento ganham popularidade. Isso está alinhado com uma tendência mais ampla para a computação em nuvem, onde os recursos são acessados através da Internet, em vez de serem mantidos localmente.

O SaaS é frequentemente associado a um modelo de licenciamento de assinatura "pay-as-you-go", o que significa que os usuários pagam pelo uso do software em um período de tempo específico, em vez de ter que fazer um grande investimento inicial para comprar licenças de software. Isso pode tornar o SaaS mais acessível, especialmente para pequenas e médias empresas.

Além disso, os aplicativos SaaS são projetados para serem interoperáveis, o que significa que eles devem ser capazes de interagir com outros dados e aplicativos em uma ampla variedade de ambientes e plataformas. Isso pode ser útil para as empresas que usam uma variedade de diferentes sistemas de software e querem que eles trabalhem juntos de forma eficiente. //Isso é comumente visto com Salesforce que, praticamente sempre tem uma integração com um sistemas de ERP como o SAP.

Em termos de uso, o SaaS é comumente implementado para fornecer funcionalidades de software de negócios a clientes empresariais. Isso ocorre porque o SaaS pode oferecer os mesmos benefícios que o software licenciado comercialmente, operado internamente, mas sem a complexidade e o alto custo inicial associados à instalação, gestão, suporte e licenciamento desses sistemas.

2.3.1 Características de SaaS e seus impactos na estrutura organizacional

As características de Software as a Service (SaaS) ajudam a identificar esse tipo de software, embora seja importante ressaltar que essas características estão em constante evolução e podem variar. A seguir estão algumas das principais características de SaaS:

Acesso via Web: O Software as a Service (SaaS), acessível via web, revolucionou o uso de software por sua conveniência e eficiência. Com um navegador padrão e uma conexão à internet, usuários podem acessar as aplicações SaaS de qualquer lugar, a qualquer momento. Isso elimina as restrições associadas a softwares tradicionais que requerem instalação em um dispositivo específico e só podem ser acessados a partir dele.

Além disso, o modelo SaaS dispensa o usuário da manutenção do software. Enquanto os softwares tradicionais podem exigir que o usuário instale atualizações e correções de segurança, no SaaS, essas responsabilidades são assumidas pelo provedor do serviço. Isso libera o usuário para se concentrar em suas tarefas principais, ao invés de gerenciar a manutenção do software.

O acesso via web ao SaaS também reduz custos com hardware e software, uma vez que o software é hospedado e executado nos servidores do provedor. Isso minimiza a necessidade de computadores de alto desempenho ou de grande espaço de armazenamento. Além disso, a colaboração torna-se mais simples e eficaz, já que todos os usuários estão trabalhando na mesma versão do software, em tempo real, independentemente de onde estejam.

Suporte do fornecedor SaaS: O suporte do fornecedor é uma característica essencial do modelo SaaS que distingue drasticamente do software tradicional. No modelo SaaS, o software é hospedado e gerenciado pelo próprio desenvolvedor, eliminando a necessidade do usuário de instalar, atualizar ou corrigir problemas de software. Toda a infraestrutura técnica, desde servidores até o gerenciamento de banco de dados, é cuidada pelo fornecedor. Essa responsabilidade transferida simplifica a experiência do usuário e libera recursos internos que, de outra forma, poderiam ser dedicados à manutenção e gestão do software.

Além disso, como o fornecedor SaaS tem controle total sobre a aplicação, ele é capaz de resolver problemas técnicos de forma mais eficiente e eficaz. Isso significa que o usuário pode contar com um suporte técnico especializado, disponível para tratar qualquer problema que possa surgir durante o uso do software.

Por fim, essa abordagem permite ao usuário final se concentrar em suas principais atividades de negócio, sem a preocupação de lidar com questões técnicas ou de segurança da informação relacionadas ao software. No modelo SaaS, o fornecedor não apenas gerencia o funcionamento do software, mas também assume a responsabilidade pela segurança da informação. Isto inclui a implementação de medidas para proteger os dados do usuário, garantir a privacidade e cumprir com os regulamentos de proteção de dados. Portanto, com o SaaS, os usuários podem aproveitar ao máximo as funcionalidades do software sem se preocupar com os desafios técnicos e de segurança que normalmente acompanham o gerenciamento de software. Essa combinação de eficiência, simplicidade e segurança é uma das razões pelas quais o SaaS está se tornando cada vez mais popular entre as empresas que buscam otimizar suas operações tecnológicas.

Preços de assinatura SaaS: O modelo de preços de assinatura do SaaS traz uma abordagem financeiramente amigável para o uso de software. Em vez de exigir que os usuários façam um investimento inicial substancial para adquirir uma licença de software, como é comum nos modelos tradicionais de licenciamento de software, o SaaS opera em um modelo de assinatura. Isso significa que os usuários pagam uma taxa mensal ou anual recorrente para usar o software.

Essa estrutura de preços tem várias vantagens. Primeiro, elimina a barreira financeira que pode impedir muitos usuários, especialmente pequenas e médias empresas, de acessar o software de alta qualidade. Em vez de ter que desembolsar uma grande soma de dinheiro à vista, os usuários podem acessar o software pagando uma quantia menor ao longo do tempo.

Além disso, a natureza recorrente do modelo de assinatura também permite um fluxo de receita mais previsível e estável para os fornecedores de SaaS, permitindo-lhes investir mais em desenvolvimento e suporte de produtos. Para os usuários, isso pode se traduzir em atualizações de produtos mais frequentes e melhor suporte ao cliente. Em suma, o modelo de preços de assinatura SaaS beneficia tanto os usuários quanto os fornecedores de software.

Baixa personalização SaaS: A característica de baixa personalização é uma faceta importante do modelo SaaS. As aplicações SaaS são muitas vezes altamente padronizadas, ou seja, criadas com uma configuração padrão que é aplicada a todos os usuários. A padronização é uma vantagem para muitos usuários, pois simplifica o uso e a manutenção do software. Com menos opções de personalização, os usuários podem começar a usar o software mais rapidamente, sem a necessidade de configurações complexas ou personalizações. Além disso, com todas as atualizações e manutenção realizadas pelo fornecedor, os usuários não precisam se preocupar com a manutenção contínua do software.

No entanto, a padronização também tem suas limitações. Com menos oportunidades para personalização, os usuários podem achar que o software SaaS não se adapta perfeitamente às suas necessidades específicas. Por exemplo, pode haver características específicas que um usuário gostaria de ter, mas que não estão disponíveis na versão padrão do software. Ou talvez o usuário queira integrar o software SaaS com outros sistemas ou aplicações que a empresa já utiliza, o que pode ser mais desafiador se o software SaaS não for projetado para ser altamente personalizável.

É importante notar, no entanto, que muitos fornecedores de SaaS estão reconhecendo essa limitação e estão começando a oferecer mais opções de personalização para seus usuários. Isso está sendo alcançado através de várias estratégias, como a criação de APIs para facilitar a integração com outros sistemas ou a inclusão de opções de configuração que permitem aos usuários adaptar certas características do software às suas necessidades específicas. Portanto, enquanto a personalização ainda pode ser limitada em algumas aplicações SaaS, a tendência é que isso mude à medida que a tecnologia e o mercado continuam a evoluir.

Atualizações gerenciadas SaaS: A gestão de atualizações é uma característica crucial do modelo SaaS. Diferentemente dos softwares tradicionais, onde os usuários precisam se encarregar de baixar e instalar atualizações, no SaaS, essas responsabilidades ficam a cargo do provedor do serviço. Isso significa que os usuários sempre têm acesso à versão mais recente do software, sem a necessidade de dedicar tempo e recursos para gerenciar as atualizações.

Por um lado, isso traz benefícios significativos em termos de conveniência e segurança. Os usuários não precisam se preocupar com a manutenção do software, o que pode liberar tempo e recursos para se concentrar em outras tarefas. Além disso, ter sempre a versão mais recente do software pode aumentar a segurança, pois as atualizações frequentemente incluem patches para vulnerabilidades de segurança que foram descobertas desde o lançamento da versão anterior.

No entanto, a gestão de atualizações pelo fornecedor também significa que os usuários têm menos controle sobre o processo de atualização. Eles não podem escolher quando as atualizações são implementadas, o que pode ser um problema se uma atualização for lançada em um momento inconveniente. Além disso, se uma atualização alterar uma característica do software de que o usuário gosta ou depende, ele pode não ter opção a não ser aceitar a mudança. Ainda assim, a maioria dos fornecedores de SaaS está ciente desses problemas e trabalha para minimizá-los, por exemplo, comunicando-se antecipadamente sobre as atualizações e fornecendo suporte para ajudar os usuários a se ajustarem às novas versões.

Mudança para mentalidade baseada em serviço SaaS: A transição para uma mentalidade baseada em serviço é um elemento fundamental da mudança para o modelo SaaS. Tradicionalmente, os fornecedores de software desenvolviam produtos, os vendiam aos clientes e, em seguida, a responsabilidade pelo gerenciamento, atualização e solução de problemas do software era transferida para o usuário ou para o departamento de TI do usuário. No entanto, o modelo SaaS muda essa dinâmica.

Com o SaaS, a responsabilidade pela manutenção e melhoria contínua do software permanece com o fornecedor. Isso significa que o fornecedor não apenas desenvolve o software, mas também fornece uma série de serviços de suporte que acompanham o software. Esses serviços podem incluir a implementação e configuração do software, teste e garantia da qualidade, treinamento para os usuários, solução de problemas e assistência técnica, manutenção regular e atualizações, hospedagem do software e garantia da segurança dos dados.

Essa mudança de uma abordagem centrada no produto para uma abordagem centrada no serviço tem várias implicações. Para os usuários, pode significar um nível mais alto de suporte e um menor ônus em termos de manutenção e gerenciamento do software. Para os fornecedores, implica um compromisso contínuo com o sucesso do cliente e uma necessidade de investir em infraestrutura, pessoal e processos para fornecer um alto nível de serviço. No entanto, essa mudança também pode proporcionar uma oportunidade para os fornecedores construírem relações mais fortes e duradouras com os clientes, ao oferecer um serviço que vai além do simples fornecimento de um produto.

de receita baseado em sucesso SaaS: No modelo SaaS, o sucesso financeiro do fornecedor está diretamente ligado ao sucesso do cliente em usar o software. Isso incentiva os fornecedores a fornecer um alto nível de serviço ao cliente para garantir a satisfação do cliente e retenção de assinatura.

Modelo de receita baseado no sucesso do SaaS: Este modelo de receita baseado no sucesso SaaS evidencia uma das principais características do SaaS: a dependência do sucesso do cliente para a prosperidade do fornecedor. Em ambientes de software tradicionais, o investimento inicial é frequentemente substancial, cobrindo custos de licença de software, hardware e recursos de implementação. No entanto, no modelo SaaS, esse investimento inicial é eliminado. O cliente paga uma taxa de assinatura, geralmente mensal ou anual, e o fornecedor é remunerado apenas se o cliente encontrar valor e satisfação na solução de software e optar por continuar a assinatura.

Isso cria um forte incentivo para os fornecedores de SaaS investirem em qualidade, usabilidade, suporte ao cliente e inovação contínua, pois a retenção do cliente é vital para a sustentabilidade do negócio. Se os usuários não estiverem satisfeitos com o serviço, eles têm a liberdade de cancelar a assinatura a qualquer momento e migrar para um concorrente. Isso é particularmente possível no ambiente SaaS devido à falta de compromissos de longo prazo e a menor dependência de infraestruturas físicas em comparação com soluções de software tradicionais.

Portanto, o modelo de receita baseado no sucesso do SaaS enfatiza a importância de fornecer um serviço de alto valor, centrado no cliente e focado na satisfação do usuário. Ele dá mais poder aos usuários finais em sua relação com os provedores de software e promove um ambiente em que o sucesso do fornecedor de SaaS está inextricavelmente ligado ao sucesso e satisfação do cliente.

2.3.2 Camadas da Arquitetura SaaS

As camadas em SaaS referem-se à estrutura arquitetural existente nessa tecnologia. Assim, cada camada tem suas responsabilidades e características que devem ser pensadas e implementadas para atingir os objetivos da camada, tendo em mente que todas elas devem estar integradas e ser capazes de se comunicar entre si.

Camada de Usuário: Em SaaS é a interface na qual os usuários interagem com o sistema. É a entrada onde os usuários fazem login no sistema e as telas que possibilitam o usuário utilizar o sistema. Deve ter alta estabilidade e uma tecnologia de segurança de rede fácil de gerenciar para proteger as informações dos usuários. Além disso, essa camada é projetada para lidar com diferentes tipos de usuários, que podem ter diferentes permissões de acesso ou conjuntos de recursos, dependendo da aplicação específica.

Esta camada é a mais próxima do usuário final na arquitetura de SaaS e é a interface através da qual os usuários acessam e interagem com a aplicação. Os usuários não precisam se preocupar com a infraestrutura subjacente, o gerenciamento de plataforma ou o desenvolvimento de aplicações; eles simplesmente utilizam o software fornecido.

Camada de Protocolo SaaS: Refere-se à camada na qual os serviços são transportados para os usuários através da Internet. Essa camada é responsável por garantir a precisão e a segurança na entrega das informações. No contexto de SaaS, a Camada de Protocolo é crucial para a interoperabilidade, segurança e eficiência na comunicação entre o cliente e o servidor.
Os protocolos comuns utilizados nesta camada incluem SOAP (Simple Object Access Protocol) e HTTPS (Hypertext Transfer Protocol Secure), que fornecem um meio de comunicação segura entre o cliente e o servidor. Além disso, essa camada também usa XML (eXtensible Markup Language) para estruturar os dados enviados através desses protocolos, e WSDL (Web Services Description Language) e UDDI (Universal Description, Discovery and Integration) para descrever, descobrir e obter metadados de serviço.

Camada de Componente SaaS: Essa é a camada na qual os serviços são encapsulados e agendados. Ela fornece um ambiente para o desenvolvimento e a biblioteca de componentes para os usuários, bem como programas de treinamento auto-orientados. Em outras palavras, essa camada é responsável por empacotar os recursos e fazer a ponte entre a funcionalidade do serviço e a maneira como essa funcionalidade é apresentada e disponibilizada ao usuário.
Os componentes nesta camada são essencialmente os blocos de construção que formam a aplicação SaaS. Eles podem incluir vários módulos ou partes de um software que são desenvolvidos para realizar funções específicas. Esses componentes são encapsulados como serviços e podem ser configurados com parâmetros pelos usuários.

Camada de Expansão de Função SaaS: Descreve a camada onde a funcionalidade dos serviços pode ser expandida ou estendida. Esta camada permite que o sistema, com base nos cursos originais ou componentes de software, seja expandido ou desenvolvido ainda mais através da API fornecida pelo sistema de treinamento.
O objetivo principal desta camada é permitir a personalização e a extensão das funcionalidades do serviço SaaS. Isso pode envolver o uso de APIs para integrar novas funcionalidades, modificar funcionalidades existentes ou conectar o serviço a outros sistemas ou serviços.
Essa camada é fortemente relacionada aos conceitos de PaaS uma vez que envolve geralmente a customização, criação de códigos, integrações e novos componentes para complementar ou extender as funcionalidades já existentes na aplicação SaaS.

Camada de Plataforma SaaS e PaaS: Essa camada é uma parte importante da arquitetura de uma aplicação baseada em serviços, e serve como um ponto de intersecção entre os recursos de Software como Serviço (SaaS) e Plataforma como Serviço (PaaS).
Esta camada é responsável por integrar um ambiente de engenharia de software e uma biblioteca de componentes públicos. Isso significa que ela fornece as ferramentas e recursos necessários para desenvolver, gerenciar e implantar componentes de software.
Além disso, a camada de plataforma SaaS e PaaS também fornece APIs (Interfaces de Programação de Aplicativos) e serviços web que servem à camada de tecnologia de serviço. Essas APIs e serviços web permitem que outros sistemas ou aplicações se conectem e interajam com a plataforma, estendendo suas funcionalidades e permitindo uma maior interoperabilidade.
O objetivo dessa camada é fornecer um ambiente unificado onde os desenvolvedores possam construir, testar e implantar suas aplicações. A disponibilidade de uma biblioteca de componentes públicos pode acelerar o processo de desenvolvimento, pois os desenvolvedores podem reutilizar os componentes existentes em vez de ter que construir tudo do zero.
Além disso, ao fornecer APIs e serviços web, esta camada também visa facilitar a integração e a comunicação entre diferentes sistemas e aplicações. Isso é particularmente importante em um ambiente de computação em nuvem, onde a interoperabilidade e a capacidade de trabalhar com várias plataformas e serviços diferentes são cruciais.

Podemos afirmar que a "Camada de Plataforma SaaS e PaaS" fornece a implementação necessária para que, na "Camada de Expansão de Função SaaS", sejam possíveis as personalizações para o Software as a Service.

Camada de Dados SaaS: Essa camada é essencialmente responsável por todas as questões relacionadas ao gerenciamento de dados dentro de uma plataforma SaaS. Isso inclui armazenar uma grande quantidade de dados de maneira eficiente e segura, o que geralmente é realizado por meio do uso de bancos de dados que podem ser implementados usando uma variedade de tecnologias e estruturas.
Além disso, esta camada é responsável por garantir que os dados sejam gerenciados de maneira eficaz. Isso pode envolver ações como garantir a consistência dos dados, realizar backup dos dados e recuperar dados em caso de falha do sistema.
Na arquitetura SaaS, é comum ter vários clientes, ou "inquilinos", compartilhando a mesma instância de uma aplicação. Portanto, a camada de dados deve garantir que os dados de cada inquilino sejam mantidos isolados uns dos outros. Isso é geralmente feito através do uso de um campo de identificação do inquilino, conhecido como Tenant ID.
A camada de dados também deve prover mecanismos para que os usuários e outras partes do sistema acessem e manipulem os dados. Isso pode ser feito através da criação de APIs, serviços de dados ou outras interfaces que permitem o acesso aos dados.
Finalmente, a segurança dos dados é uma responsabilidade crítica da camada de dados. Isso pode envolver a implementação de medidas de segurança, como criptografia e controle de acesso, para proteger os dados contra acesso não autorizado.

Apesar de termos aqui uma divisão da infraestrutura da SaaS em camadas é sempre importante reforçar que novas propostas e maneiras de se desenhar SaaS tem surgido, algumas mais detalhadas e outras mais simplificadas. Algumas análises buscam isolar mais o SaaS assim como alguns outros tentam expandir as relações com PaaS e IaaS contectando e relacionando todas essas estruturas.

2.3.3 Níveis de Maturidade em SaaS

Em termos de arquitetura de Software as a Service , o conceito de "maturidade" aponta o quão avançada e eficiente é uma determinada implementação de SaaS. Esse conceito de maturidade foi descrito pela Microsoft e é dividido em quatro níveis, cada um representando um passo adiante na evolução de uma aplicação SaaS.
Esses níveis de maturidade são definidos por três atributos principais: facilidade de configuração, eficiência de multi-inquilino (ou seja, a capacidade de servir múltiplos clientes de forma eficiente a partir de uma única instância da aplicação) e escalabilidade (a capacidade de a aplicação crescer e lidar com o aumento da demanda).
Esses níveis de maturidade de SaaS oferecem uma estrutura para entender o progresso e o desenvolvimento de uma aplicação SaaS, permitindo que as empresas avaliem a sofisticação de suas soluções SaaS assim possam melhorar.

Nível de Maturidade Arquitetônica SaaS 1 - Ad-Hoc: Neste nível, cada cliente tem uma instância única e personalizada do aplicativo hospedado. A maturidade neste nível é quase inexistente, pois não há compartilhamento de recursos entre os clientes. Cada cliente tem seu próprio conjunto de recursos dedicados. Isto é equivalente a executar uma aplicação local, mas hospedada em um servidor remoto.

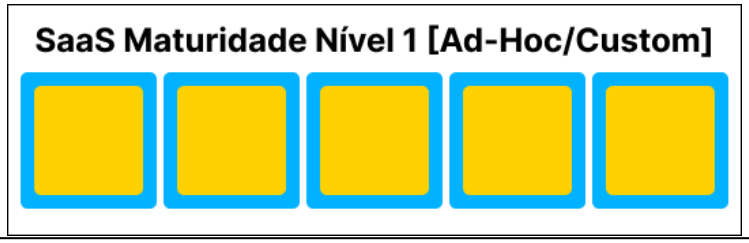

Figura 12: Representação gráfica de estrutura SaaS no nível 1 de maturidade.
Fonte: Autor

Nível de Maturidade Arquitetônica SaaS 2 - Configurabilidade: Neste nível, a maturidade é aumentada através da introdução de metadados de configuração. Muitos clientes podem usar instâncias separadas do mesmo aplicativo, mas estas instâncias são configuradas de acordo com as necessidades específicas de cada cliente. Isto permite que o fornecedor mantenha uma única base de código para o aplicativo, mas forneça experiências de usuário diferenciadas através da configuração.

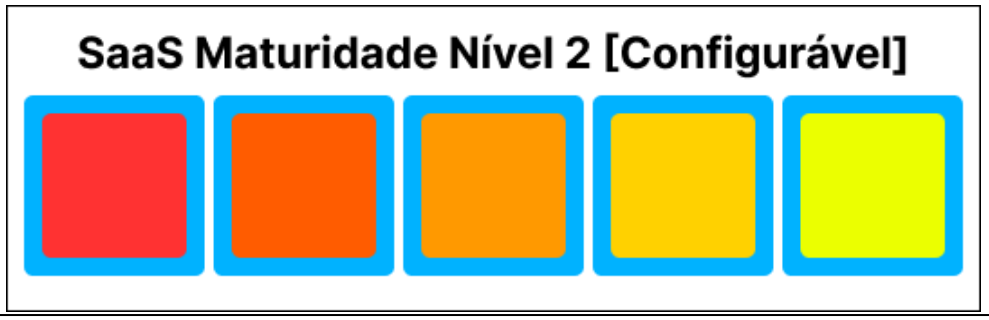

Figura 13: Representação gráfica de estrutura SaaS no nível 2 de maturidade.
Fonte: Autor

Nível de Maturidade Arquitetônica SaaS 3 - Eficiência Multi-inquilino: Neste nível, a arquitetura passa a suportar a multi-inquilinidade. O mesmo aplicativo é usado por todos os clientes, mas o aplicativo é projetado de tal forma que cada cliente se sente como se estivesse usando sua própria instância do aplicativo. Isto permite um uso mais eficiente dos recursos do servidor, pois uma única instância do aplicativo pode atender a todos os clientes.

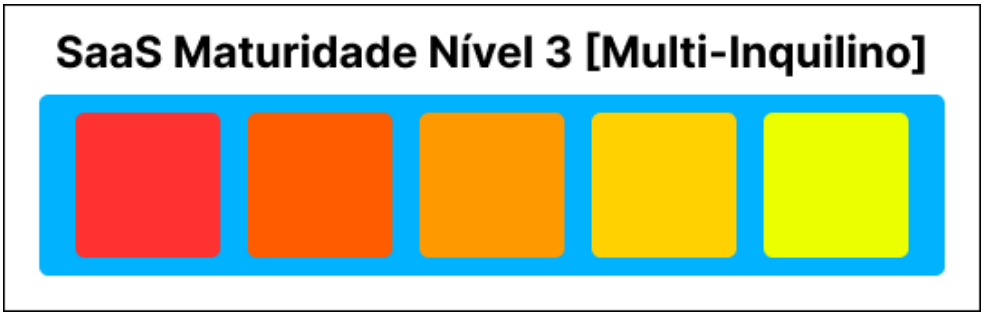

Figura 14: Representação gráfica de estrutura SaaS no nível 3 de maturidade.
Fonte: Autor

Nível de Maturidade Arquitetônica SaaS 4 - Escalável: Neste nível, a arquitetura é projetada para ser escalável. Isso é alcançado através da introdução de uma arquitetura multi-camadas, onde várias instâncias do mesmo aplicativo podem ser executadas em paralelo em servidores diferentes. Isso permite que o fornecedor aumente ou diminua a capacidade do sistema de acordo com a demanda, simplesmente adicionando ou removendo servidores. Este nível de maturidade é necessário para aplicações que precisam de usuários simultaneamente.

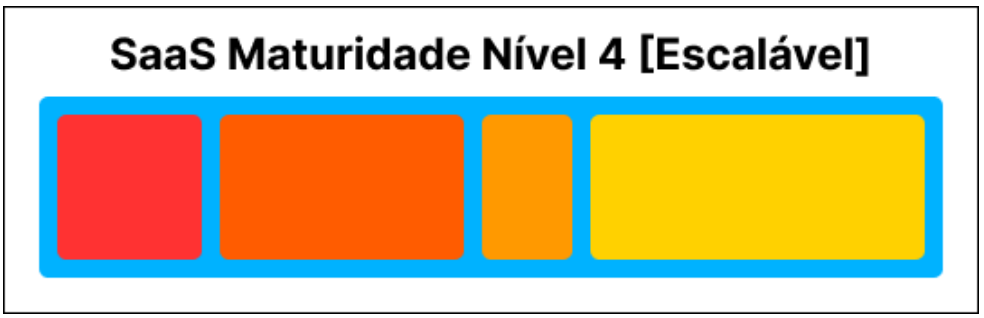

Figura 15: Representação gráfica de estrutura SaaS no nível 4 de maturidade.
Fonte: Autor

2.4 Conclusão de Software as a Service

Este capítulo foi dedicado a esclarecer os princípios de SaaS e sua arquitetura, fornecendo uma base sólida para o entendimento dessa importante área da computação em nuvem. No entanto, ainda seria possível expandir e aprofundar mais em diversos temas e subtemas aqui trazidos. A arquitetura multi-inquilino, por exemplo, é um campo que merece um estudo mais aprofundado devido à sua relevância para a eficiência e escalabilidade das soluções SaaS. No próximo capítulo, daremos uma olhada em um estudo de caso de um produto SaaS que exemplifica as características da Arquitetura Orientada a Serviços (SOA). Este estudo de caso irá nos ajudar a entender melhor a aplicação prática dos conceitos que discutimos até agora.

3. Estudo de Caso Salesforce.com

Neste momento, o estudo avança para seu último capítulo. Este segmento tem como foco a tecnologia Salesforce, que incorpora em sua estrutura conceitos fundamentais de SOA e SaaS. A exploração começará com uma descrição concisa de Salesforce, para então fazer conexões entre essa tecnologia e os princípios de SaaS, e em seguida, SOA. Este percurso tem o objetivo de elucidar como esses conceitos se entrelaçam na prática dentro desta plataforma específica.

3.1 Introdução ao Salesforce

Salesforce é uma empresa multinacional americana que oferece um conjunto de soluções de negócios baseadas em nuvem, focadas em Customer Relationship Management (CRM). A empresa se destacou como uma força inovadora, revolucionando o espaço do CRM graças à sua abordagem pioneira baseada em nuvem. Foi fundada em 1999 por Marc Benioff, Parker Harris, Dave Moellenhoff e Frank Dominguez, com a visão de reinventar o CRM e transformar o modo como as empresas interagem com seus clientes.

Antes da Salesforce, o software de CRM era geralmente alojado localmente nos servidores das empresas, tornando-o caro e difícil de implementar e atualizar. A Salesforce, no entanto, inaugurou uma nova era ao oferecer seu software como um serviço através da internet, seguindo o modelo de Software as a Service (SaaS). Isso permitiu que as empresas acessassem e usassem o software de qualquer lugar, sem a necessidade de instalação ou manutenção de hardware e software.

O CRM da Salesforce foi um dos primeiros a adotar o modelo SaaS e a ideia de entregar software empresarial pela internet. Essa abordagem revolucionária contribuiu para a popularização do modelo SaaS em toda a indústria de software e estabeleceu a Salesforce como uma inovadora no setor.

Desde a sua fundação, a Salesforce expandiu significativamente seu portfólio de produtos para além do CRM, incluindo ferramentas de vendas, serviço, marketing, comércio, plataforma e muitos outros. A empresa se tornou uma líder de mercado no espaço de CRM e é reconhecida por seu compromisso contínuo com inovação e melhoria de produtos.

Um testemunho do caráter inovador da Salesforce é o seu reconhecimento no Quadro Mágico da Gartner de 2023, onde foi nomeada como líder em Plataformas de Aplicação de Baixo Código Corporativo. Este reconhecimento ressalta a posição da Salesforce na vanguarda da tecnologia de baixo código, um setor de rápido crescimento que está remodelando o desenvolvimento de software empresarial.

Hoje, a Salesforce é uma das maiores e mais reconhecidas empresas de software do mundo, servindo a milhares de clientes em vários setores e regiões. Eles continuam a liderar a indústria com sua abordagem baseada em nuvem para CRM e SaaS, enquanto também exploram novas tecnologias, como inteligência artificial e aprendizado de máquina, para melhorar ainda mais seus produtos e serviços e manter sua posição como uma das empresas mais inovadoras na indústria de software.

A Salesforce oferece uma variedade de produtos e serviços desenhados para atender a necessidades de negócios em diversas áreas. Vamos dar uma olhada mais de perto em alguns desses produtos.

Sales Cloud: Esta é a solução de CRM (Customer Relationship Management) da Salesforce, projetada para ajudar as empresas a gerenciar seus ciclos de vendas de maneira eficiente. Com Sales Cloud, as empresas podem acompanhar o desempenho das vendas, identificar tendências e padrões e manter relacionamentos com os clientes. Ele oferece recursos como gestão de leads, previsões de vendas, automação de vendas e muito mais, permitindo que as empresas fechem negócios mais rapidamente e aumentem a receita.

Service Cloud: Esta é uma plataforma dedicada ao suporte ao cliente e serviços. Service Cloud permite às empresas gerenciar eficientemente as interações com os clientes e fornecer suporte através de vários canais, incluindo telefone, email, chat ao vivo e redes sociais. Ele oferece recursos como gestão de casos, conhecimento do cliente, roteamento de serviço inteligente e muito mais, ajudando as empresas a fornecer um atendimento ao cliente excepcional e a aumentar a satisfação do cliente.

Marketing Cloud: Esta é uma plataforma de marketing digital abrangente que fornece ferramentas para email marketing, automação de marketing, gerenciamento de campanhas em mídias sociais, publicidade e personalização. Marketing Cloud permite que as empresas criem e gerenciem campanhas de marketing eficazes, segmentem clientes com precisão e entreguem mensagens personalizadas em escala.

Commerce Cloud: Esta solução permite às empresas gerenciar suas operações de comércio eletrônico em várias plataformas e canais. Com o Commerce Cloud, as empresas podem criar experiências de compra personalizadas, gerenciar inventário e pedidos, e integrar-se facilmente com outras soluções da Salesforce para proporcionar uma experiência de cliente unificada.

Platform: Esta é a plataforma de desenvolvimento da Salesforce que permite aos desenvolvedores criar e personalizar aplicativos na nuvem. Com a Platform, os desenvolvedores podem aproveitar os recursos de baixo código da Salesforce para criar aplicativos personalizados de forma rápida e eficiente. Além disso, os desenvolvedores podem acessar uma ampla gama de APIs para integrar seus aplicativos com outras soluções e serviços da Salesforce.

Cada um desses produtos da Salesforce desempenha um papel fundamental na estratégia de negócios de uma empresa, permitindo-lhes gerir eficazmente suas operações de vendas, serviços, marketing e comércio eletrônico. Combinados, eles fornecem uma solução completa para empresas que desejam melhorar suas relações com os clientes e impulsionar o crescimento dos negócios.

3.2 Salesforce como SaaS

Nessa seção o foco será em aplicar o conhecimento adquirido sobre SaaS ao ecossistema Salesforce. Serão discutidos a arquitetura, características e nível de maturidade dessa plataforma, observando sua implementação como uma solução SaaS. Além disso, será avaliada a arquitetura multi-tenant da Salesforce, uma característica fundamental de seu modelo. Por fim, serão destacados os benefícios e desafios resultantes da utilização da Salesforce como uma ferramenta baseada no modelo SaaS.

3.2.1 Como o Salesforce se encaixa no modelo SaaS

O Salesforce é um exemplo de aplicação que se enquadra no modelo SaaS, atendendo às características desse tipo de software. A seguir, exploraremos cada um dos pontos mencionados, com foco na aplicação Salesforce.

Acesso via Web: Como plataforma baseada na nuvem, o Salesforce é acessível pela web. Isso permite que os usuários acessem suas funcionalidades de qualquer lugar e a qualquer momento, desde que possuam uma conexão com a internet, inclusive em plataformas móveis.

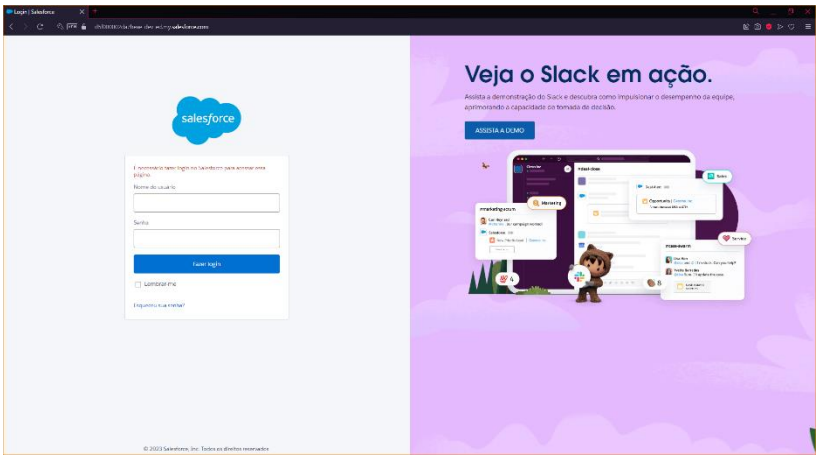

Figura 16: Essa captura de tela mostra a interface em que o usuário insere seu "username" e senha. O endereço da página web inserido no navegador direciona o usuário para o seu ambiente pessoal, onde ele é capaz de ver apenas seus dados e configurações.

Fonte: Autor

Suporte do fornecedor SaaS: O Salesforce oferece suporte abrangente ao cliente, incluindo manutenção, gerenciamento e segurança do software. Isso libera os usuários dessas tarefas, permitindo que se concentrem em suas operações principais.

Preços de assinatura SaaS: O Salesforce adota um modelo de preços por assinatura e licenciamento, permitindo aos usuários acessar sua plataforma sem um investimento inicial substancial. Isso torna a solução mais acessível para empresas de todos os tamanhos e permite que a solução escale de acordo com a demanda, seja de recursos ou de usuários.

Atualizações gerenciadas SaaS: O Salesforce gerencia todas as atualizações de software, garantindo que os usuários sempre tenham acesso à versão mais recente da plataforma. Isso elimina a necessidade dos usuários gerenciarem as atualizações por conta própria.

Baixa personalização SaaS: Embora muitas soluções SaaS sejam altamente padronizadas, o Salesforce se destaca por oferecer amplos recursos de personalização. Isso permite que as empresas adaptem a plataforma às suas necessidades específicas. Isso é feito por meio de ferramentas nativas de configuração dentro da plataforma e por meio da extensão dessas configurações via código, permitindo que os desenvolvedores expandam as capacidades da ferramenta.

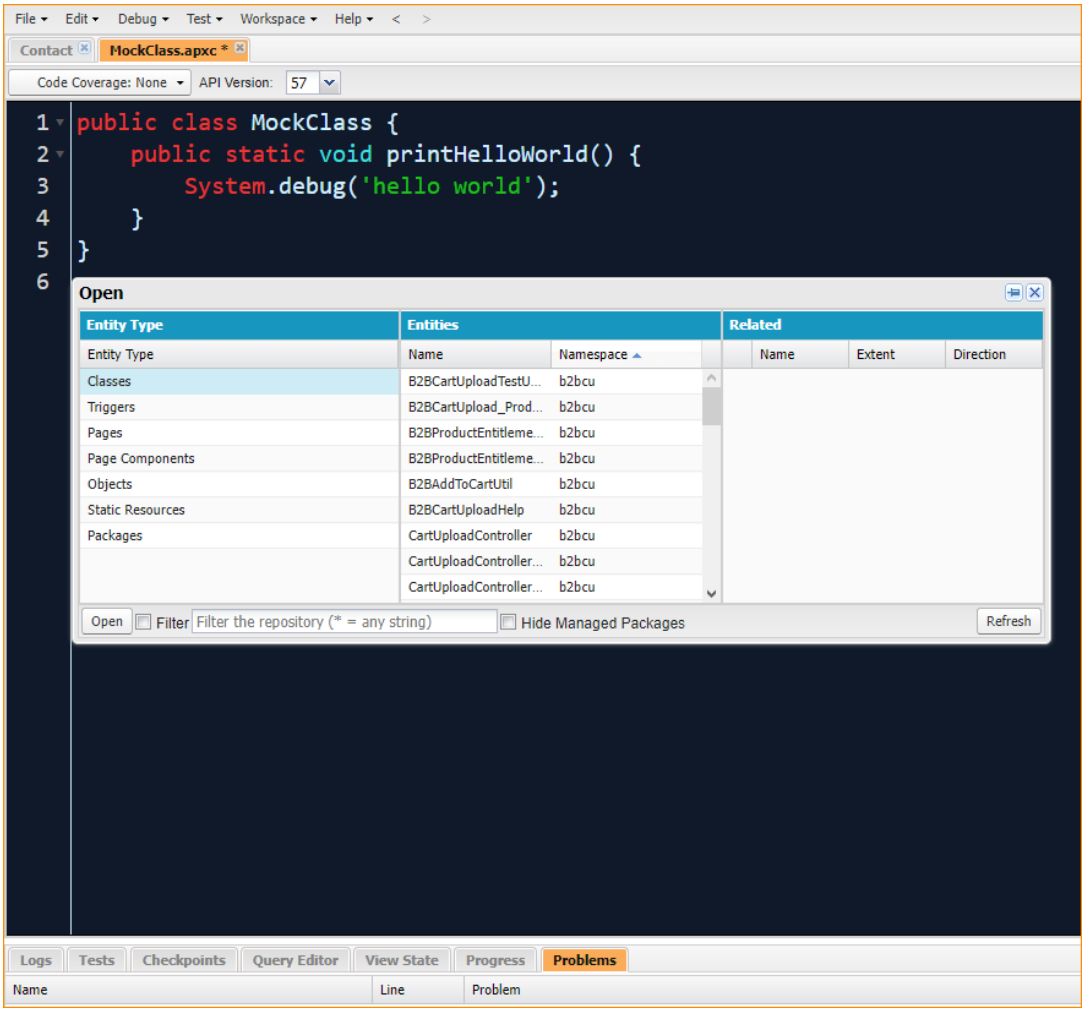

Figura 17: Essa captura de tela mostra o "Developer Console", uma área dentro da organização que permite aos desenvolvedores criar classes, visualizar objetos e acessar muitos outros recursos para configurar e estender as funcionalidades da plataforma. Além disso, o Salesforce possui integração com a popular IDE Visual Studio Code, onde os desenvolvedores podem ampliar ainda mais essas capacidades..

Fonte: Autor

Mudança para mentalidade baseada em serviço SaaS: O Salesforce destaca-se por ofertar um serviço integral. Além de fornecer uma robusta plataforma de software, também disponibiliza serviços de configuração, assistência técnica e manutenção.

Modelo de receita baseado no sucesso do SaaS: Como outros provedores de SaaS, a receita do Salesforce está diretamente ligada ao sucesso de seus clientes. Assim, conforme o negócio do cliente cresce por meio das ferramentas fornecidas pela Salesforce, este cliente passa a demandar mais recursos do fornecedor, gerando um ciclo produtivo para ambas as partes. Isso incentiva o Salesforce a fornecer um alto nível de serviço e a se esforçar para garantir a satisfação do cliente, a fim de garantir a retenção de assinaturas e a expansão de seus clientes.

Agora, podemos observar como cada camada da arquitetura SaaS pode é implementada dentro do contexto Salesforce:

Camada de Usuário: Nessa camada, os usuários interagem diretamente com a interface do Salesforce. A interface é intuitiva, facilitando o acesso aos dados e às funcionalidades da plataforma.

Camada de Protocolo SaaS: O Salesforce é responsável por garantir a segurança na transmissão de serviços pela Internet. Ele utiliza protocolos como HTTPS para assegurar a proteção dos dados durante a transmissão via web.

Camada de Componente SaaS: O Salesforce possui um ambiente de desenvolvimento conhecido como Salesforce Platform. Este ambiente disponibiliza uma biblioteca de componentes para os usuários, além de oferecer recursos de personalização com baixo e alto código.

Camada de Expansão de Função SaaS: O Salesforce permite que a funcionalidade do serviço seja personalizada e expandida. Isso possibilita que as empresas moldem a plataforma de acordo com suas necessidades específicas.

Camada de Plataforma SaaS e PaaS: O Salesforce Platform integra características tanto de SaaS (Software as a Service) quanto de PaaS (Platform as a Service), fornecendo um ambiente de desenvolvimento de software e uma biblioteca de componentes para o público.

Camada de Dados SaaS: O Salesforce é responsável pela gestão eficiente e segura dos dados, incluindo tarefas de armazenamento, gerenciamento, backup e segurança.

Porfim, sobre os níveis de maturidade, a seguir como a Salesforce se porta perante cada nível, assim como podemos confirmar que já se encontra no quarto nível uma vez que já é capaz de atender as demandas de cada um dos graus de maturidade:

Nível 1 - Ad-Hoc/Custom: O Salesforce permite extensas personalizações, possibilitando que cada cliente tenha uma instância única do aplicativo.

Nível 2 - Configurabilidade: A plataforma Salesforce permite que os clientes usem a mesma instância do aplicativo, mas com configurações específicas para atender às suas necessidades particulares.

Nível 3 - Eficiência Multi-inquilino: O Salesforce é um exemplo de aplicação multi-inquilino. Nesse modelo, o mesmo aplicativo é usado por todos os clientes, mas cada cliente tem a sensação de estar usando sua própria instância.

Nível 4 - Escalável: O Salesforce foi projetado para ser escalável, com a capacidade de aumentar ou diminuir a capacidade conforme a demanda dos usuários.

3.2.2 Como a arquitetura multilocatária é aplicada na Salesforce

No Salesforce, a arquitetura multilocatária é implementada de maneira eficiente e segura, permitindo que múltiplos usuários e organizações compartilhem o mesmo ambiente de aplicação, ao mesmo tempo em que mantêm seus dados e configurações isolados.

Segurança e Isolamento de Dados: No Salesforce, o isolamento de dados é garantido por meio de um modelo chamado "organizações". Cada cliente tem sua própria organização, que é uma instância virtual do Salesforce, fornecendo um ambiente separado para seus dados e configurações. Embora essas organizações estejam todas hospedadas na mesma infraestrutura física, o Salesforce utiliza várias camadas de segurança para garantir que os dados de uma organização nunca sejam acessíveis por outra.

Por exemplo, os dados de cada organização são armazenados em tabelas de banco de dados compartilhadas, mas cada linha de dados é marcada com um identificador de organização único. Quando um usuário faz uma solicitação de dados, o Salesforce verifica o identificador da organização do usuário e só retorna as linhas de dados que correspondem a esse identificador.

Eficiência e Atualizações: A arquitetura multilocatária do Salesforce também facilita a eficiência e a manutenção do sistema. Todos os clientes compartilham a mesma base de código do aplicativo, o que significa que quando o Salesforce faz uma atualização, todos os clientes se beneficiam dela automaticamente, sem tempo de inatividade ou interrupção.
Por exemplo, quando o Salesforce lança uma nova funcionalidade, como a Lightning Experience, ela é disponibilizada para todas as organizações ao mesmo tempo. Os clientes podem então escolher quando e como habilitar a nova funcionalidade para seus usuários.

Personalização: A arquitetura multilocatária também permite um alto grau de personalização. No Salesforce, cada organização pode personalizar a aplicação para atender às suas necessidades específicas, sem afetar outras organizações.
Por exemplo, uma organização pode adicionar campos personalizados a um objeto padrão, como uma oportunidade de vendas, para rastrear informações específicas para o seu negócio. Ou uma organização pode criar seus próprios objetos personalizados para rastrear tipos de dados que são únicos para o seu negócio.
Essas personalizações são armazenadas separadamente da base de código principal do Salesforce, em um metadado que descreve a configuração da organização. Isso permite que as personalizações sejam preservadas mesmo quando a base de código principal é atualizada.

Portanto, a arquitetura multilocatária é uma parte essencial do Salesforce, permitindo que ele ofereça um serviço eficiente, escalável e personalizável para seus clientes, mantendo ao mesmo tempo um alto nível de segurança e isolamento de dados.

3.2.3 Benefícios e desafios da Salesforce como SaaS

Os benefícios da Salesforce como uma plataforma SaaS são consideráveis e têm impacto direto na eficácia dos negócios. A acessibilidade é um dos atributos mais destacados da plataforma. Como solução baseada em nuvem, permite que seus usuários acessem o sistema de qualquer lugar com uma conexão à internet. Isso facilita, por exemplo, que um representante de vendas possa atualizar as informações de um cliente em tempo real durante uma reunião externa, utilizando apenas um dispositivo móvel. Esta capacidade aumenta a eficiência e a precisão das operações de negócios.

A escalabilidade é outra vantagem significativa da Salesforce. Graças ao alto nível de serviço prestado pela Salesforce, as empresas podem ajustar a quantidade de recursos que utilizam, conforme suas necessidades se alteram. Isso pode ser feito por e-mail ou por painéis em que usuários administradores fazem a solicitação. Assim uma pequena empresa pode começar com um número limitado de licenças de usuário e, à medida que expande, pode facilmente adicionar mais licenças. Esta facilidade de escalabilidade permite que as empresas se adaptem e cresçam sem preocupações logísticas significativas.

A manutenção e as atualizações são outras áreas em que a Salesforce se destaca. A empresa se encarrega de todos os aspectos de manutenção e atualizações, liberando os usuários de tais responsabilidades. As atualizações são implementadas automaticamente, garantindo que os usuários estejam sempre utilizando a versão mais atual da plataforma. Além disso, a Salesforce fornece uma documentação densa e detalhada sobre todas as atualizações, facilitando o entendimento das novas funcionalidades e mudanças.

A personalização é mais um benefício de usar a Salesforce. A plataforma permite que as empresas adaptem suas funcionalidades às suas necessidades específicas. Além disso, a Salesforce oferece a possibilidade de extensão de funcionalidades via código, permitindo uma personalização ainda maior. Uma empresa pode, por exemplo, personalizar seu CRM para rastrear métricas específicas vitais para o seu negócio.

No entanto, a utilização da Salesforce como SaaS também apresenta desafios. A dependência do fornecedor é um deles. Se a Salesforce enfrentar uma interrupção de serviço por exemplo, isso pode afetar o acesso à plataforma e gerar impactos negativos em todos processos que dependam do Salesforce para serem executados.

Além disso, a segurança dos dados pode ser um desafio. Apesar das robustas medidas de segurança de dados implementadas pela Salesforce, a natureza baseada em nuvem da plataforma implica que os dados são armazenados fora das instalações da empresa. Embora isso traga benefícios em termos de acessibilidade e redução de custos de infraestrutura, também pode apresentar riscos associados à segurança e ao controle dos dados em uma eventual invasão ao fornecedor.

3.3 Salesforce sua relação com SOA

A aplicação da Arquitetura Orientada a Serviços na Salesforce é facilitada pelo uso extensivo de APIs e WebServices, elementos cruciais na estrutura da plataforma. Esses recursos permitem a expansão das funcionalidades da Salesforce para além de suas capacidades nativas, especialmente quando os desenvolvedores necessitam criar soluções personalizadas ou integrar a plataforma com outros sistemas empresariais, demonstrando assim a flexibilidade e adaptabilidade da Salesforce como um ambiente de desenvolvimento orientado a serviços.

3.3.1 Implementação de princípios da SOA em Salesforce

A Salesforce incorpora os princípios fundamentais da Arquitetura Orientada a Serviços (SOA) de forma eficaz através de uma variedade de APIs e serviços da Web .A SOA é um estilo arquitetônico que define o uso de serviços para suportar os requisitos de negócios dos usuários. A principal característica da SOA é a sua capacidade de permitir a integração de diferentes aplicações e serviços de maneira mais flexível e escalável, o que é de vital importância para organizações modernas que buscam digitalizar suas operações de negócios e é isso que a Salesforce faz ao trazer uma premissa de ser totalmente integrável com qualquer outro sistema externo como ERPs, sistemas de controle de analíticos ou qualquer outro que o cliente veja como necessário.
Na Salesforce, a implementação da SOA é visível na forma como suas APIs permitem aos desenvolvedores interagir com a plataforma. A empresa fornece uma variedade de APIs, cada uma projetada para atender a diferentes necessidades de negócios e cenários técnicos. Estas incluem a API REST, a API SOAP, a API Bulk, entre outras, bem como APIs específicas disponíveis através da Salesforce Commerce Cloud, como os serviços de Shipment, Tax Calculation e Payment.

API REST: é uma interface de programação de aplicativos baseada em HTTP que permite a interação fácil com a plataforma Salesforce usando métodos HTTP padrão. Ela é comumente usada para integrar a Salesforce com outras aplicações externas e permite aos desenvolvedores trabalhar com registros, pesquisar e recuperar metadados, entre outras funções, de forma programática.

API SOAP: da Salesforce, por outro lado, é uma API baseada em padrões da web que permite que os desenvolvedores criem, recuperem, atualizem ou excluam registros. Ela é frequentemente usada quando há uma necessidade de interação intensiva com dados, como quando se trabalha com um grande volume de registros.

API Bulk: é especificamente projetada para fornecer uma maneira programática de carregar e consultar um grande volume de dados na Salesforce. Ela permite aos desenvolvedores realizar operações em massa, como inserir, atualizar, excluir e exportar dados em lote, com eficiência e eficácia.

Além disso, a Salesforce fornece uma ampla gama de funcionalidades para diversas necessidades de negócios, existem também APIs dedicadas especificamente a certos produtos. Um exemplo notável disso é encontrado na Commerce Cloud da Salesforce, que possui várias APIs específicas para apoiar as operações de e-commerce. Essas APIs incluem:

APIs de Pagamento: Permitem a integração de várias soluções de pagamento na plataforma de e-commerce, facilitando transações seguras e eficientes.

APIs de Cálculo de Impostos: Usadas para calcular automaticamente os impostos aplicáveis com base na localização do cliente e no tipo de produto vendido.

APIs de Envio: São usadas para integrar soluções de logística e entrega, permitindo um processo de envio eficiente e rastreamento em tempo real das remessas.

Essas APIs dedicadas adicionam uma camada extra de funcionalidade e personalização, permitindo que as empresas moldem suas experiências de e-commerce de acordo com suas necessidades e preferências específicas.

Essas APIs e serviços web, que permitem a interoperabilidade entre a Salesforce e outros sistemas, são a manifestação dos princípios da SOA na plataforma Salesforce. Eles permitem que os desenvolvedores e as empresas criem soluções complexas e integradas que podem evoluir e escalar de acordo com as necessidades do negócio.

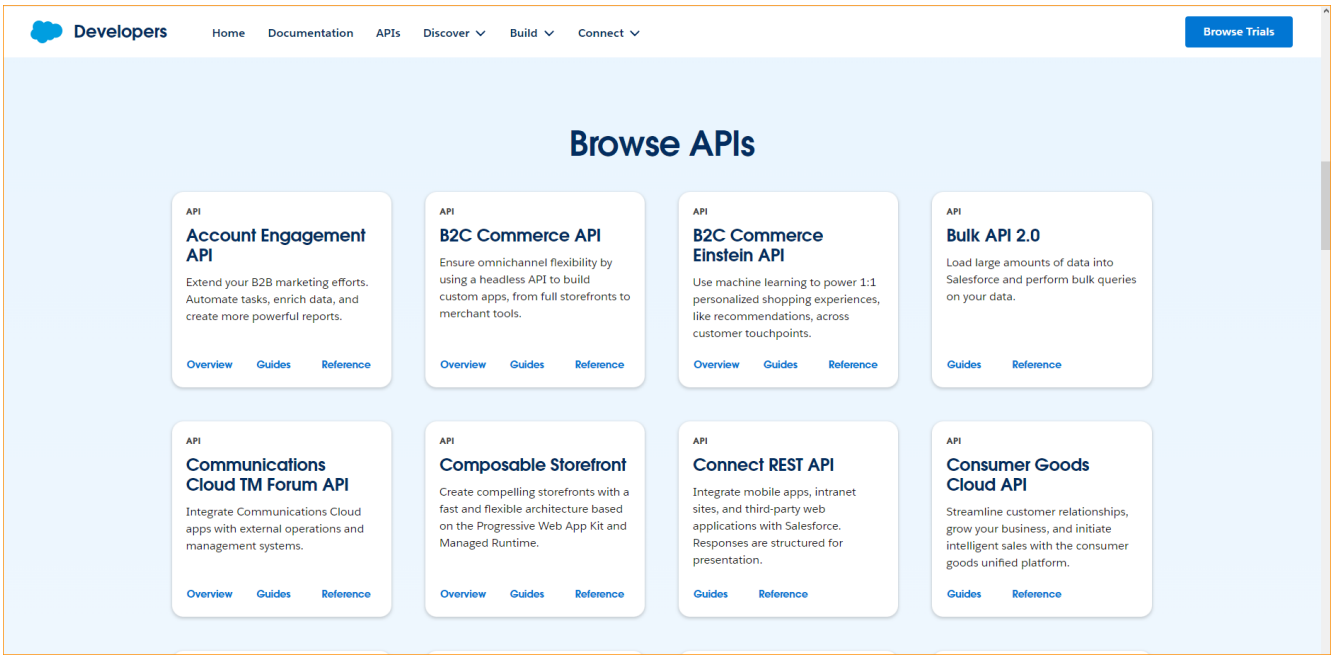

Figura 18: Essa captura de tela mostra a documentação do Salesforce para desenvolvedores, nessa em específico o desenvolvedor pode pesquisar por uma série de APIs para as mais diversas demandas.

Fonte: Autor

3.3.2 Composição de serviços em Salesforce

A Salesforce oferece uma ampla gama de serviços que podem ser compostos para criar soluções de negócios personalizadas. Cada serviço é projetado para atender a uma necessidade específica de negócios, mas quando combinados, eles proporcionam uma solução mais completa e robusta.

Vamos considerar um exemplo que envolve a combinação de Sales Cloud, Service Cloud, Marketing Cloud e Commerce Cloud para criar uma solução de negócios abrangente:

Sales Cloud é o serviço de CRM de vendas da Salesforce, proporcionando uma variedade de recursos que ajudam as empresas a gerenciar seus leads, oportunidades, contas e contatos, facilitando o fechamento de negócios de forma mais rápida e eficiente.

Service Cloud é uma plataforma de atendimento ao cliente que permite às empresas fornecer um suporte pós-venda eficaz. Inclui recursos como gerenciamento de casos, canais de serviço, roteamento de trabalho e outros.

Marketing Cloud é uma plataforma de marketing digital que ajuda as empresas a criar e gerenciar campanhas de marketing eficazes. Oferece ferramentas para email marketing, publicidade digital, personalização, segmentação e análise da jornada do cliente.

Commerce Cloud é a plataforma de comércio eletrônico da Salesforce, que permite às empresas criar experiências de compra personalizadas e unificadas em todos os canais. Inclui serviços como gerenciamento de produtos, carrinho de compras, finalização de compra, gerenciamento de pedidos, bem como APIs específicas para tarefas como cálculo de impostos, envio e pagamento.

Quando esses quatro serviços são combinados, eles criam uma solução completa de CRM que cobre todos os aspectos do ciclo de vida do cliente, desde a aquisição até a retenção passando pelo suporte pós-venda e até mesmo o comércio eletrônico.

Além disso, a integração e a composição desses serviços podem ser ainda mais personalizadas e ampliadas por meio do uso de APIs e serviços web. Por exemplo, uma empresa pode usar a API REST da Salesforce para integrar a plataforma Salesforce com outros sistemas empresariais como SAP, ou pode usar a API Bulk para realizar operações em massa.

Desta forma, a Salesforce permite que as empresas combinem e configurem seus serviços de acordo com suas necessidades específicas, criando soluções de negócios robustas, personalizadas e escaláveis.

3.4 Conclusão do estudo de caso Salesforce e sua relação com SaaS e SOA

Como profissional atuante na área de tecnologia, minha experiência com Salesforce, uma tem sido profundamente enriquecedora. Trabalhei em diversas implementações, presenciando os paradigmas da plataforma e enxergando cada vez mais os princípios da Orientação a Serviços, não somente sob a ótica técnica como também sob uma ótica de negócios A facilidade em criação de funcionalidades dado o caráter de SaaS me permitiu desviar da necessidade de implementar CRUDs ou outros conceitos "comuns" uma vez que tudo que é "comum" já é previsto pela SaaS e então pré-implementado.

Explorar a customização do Salesforce para atender às necessidades específicas de cada organização, criando telas e componentes para serem utilizados em diversos contextos. Essa flexibilidade, aliada à segurança e integridade dos dados nativa da plataforma, é uma forte evidência do potencial do Salesforce na era digital. Esse potencial permite que os desenvolvedores foquem naquilo que é específico do cliente e deixem as funcionalidade repetitivas como uma responsabilidade da Salesforce.

CONSIDERAÇÕES FINAIS

Com a ascensão incomparável da computação em nuvem e uma crescente demanda por reutilização e integração, torna-se evidente que a orientação a serviços e os softwares como serviço assumirão um papel cada vez mais proeminente nos próximos anos. Diante desse cenário, é crucial que os desenvolvedores ampliem suas habilidades e se preparem para atender essas demandas emergentes.

Neste trabalho, exploramos inicialmente os conceitos fundamentais da Arquitetura Orientada a Serviços, compreendendo a natureza dos serviços tanto em termos técnicos quanto teóricos. Esse conhecimento nos habilitou a desenvolver serviços, começando por estruturas simples e gradualmente avançando para composições mais complexas, capazes de atender a uma ampla variedade de necessidades. Em seguida, mergulhamos nos conceitos de Software como Serviço, começando com uma breve introdução à computação em nuvem e, posteriormente, detalhando as camadas de SaaS e compreendendo as funcionalidades que devem ser presentes em cada uma.

Por fim, aplicamos e conectamos esses conhecimentos por meio de um estudo de caso Salesforce, uma tecnologia que emprego em minha rotina diária e que incorpora os conceitos de SaaS e SOA.

Assim, posso afirmar que as lições aprendidas neste trabalho representam um passo significativo na jornada em direção à implementação de serviços em nuvem. A partir destes conceitos fundamentais, uma infinidade de possibilidades se abre, permitindo a exploração de temas mais complexos derivados dos tópicos aqui abordados.

REFERÊNCIAS

Service Oriented Architecture:

ERL, T.; PAULO MERSON; STOFFERS, R. **Service-oriented architecture : analysis and design for services and microservices**. Boston: Prentice Hall, 2017.

AMAZON.**O que é SOA? – Explicação sobre arquitetura orientada a serviços – AWS**. Disponível em: <https://aws.amazon.com/pt/what-is/service-oriented-architecture/>.

ATLASSIAN. **O que é um sistema distribuído?** Disponível em: <https://www.atlassian.com/br/microservices/microservices-architecture/distributed-architecture>.

ARCITURA PATTERNS .**The Annotated SOA Manifesto | Arcitura Patterns**. Disponível em: <https://patterns.arcitura.com/soa-patterns/basics/soamanifesto/annotated>.

ARCITURA PATTERNS .**Service Inventory Analysis | Arcitura Patterns**. Disponível em: <https://patterns.arcitura.com/soa-patterns/basics/soaproject/serviceinventory>.

IBM.**O que é WSDL?** Disponível em: <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=services-what-is-wsdl>.

AMAZON.**O que é uma API? – Explicação sobre interfaces de programação de aplicações – AWS**. Disponível em: <https://aws.amazon.com/pt/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming%20Interface.>.

W3SCHOOLS.**XML Schema Tutorial**. Disponível em: <https://www.w3schools.com/xml/schema_intro.asp>.

IBM. **XML schema definition (XSD) assets**. Disponível em: <https://www.ibm.com/docs/en/iis/11.5?topic=types-xml-schema-definition-xsd-assets>.

IBM. **O que é WSDL?** Disponível em: <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=services-what-is-wsdl>.

Software as a Service:

Salesforce.**What is Software as a Service (SaaS) - Salesforce.com**. Disponível em: <https://www.salesforce.com/saas/>.

RANI, D.; RANJAN, R. K. A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering, v. 4, n. 6, jun. 2014. ISSN: 2277-128X.

SATYANARAYANA, S. Cloud Computing: SaaS. GESJ: Computer Science and Telecommunications, n. 4(36), 2012. ISSN 1512-1232.

LIAO, H. Design of SaaS-based Software Architecture. In: INTERNATIONAL CONFERENCE ON NEW TRENDS IN INFORMATION AND SERVICE SCIENCE, 2009. Anais... Nanchang, JiangXi Province, China: School of Software, JiangXi University of Finance & Economics, 2009.

Salesforce:

SALESFORCE. **Salesforce Recognized as a Leader in 2023 Gartner® Magic Quadrant™ for Enterprise Low-Code Application Platforms**. Disponível em: <https://www.salesforce.com/news/stories/gartner-low-code-magic-quadrant-2023/>.

SALESFORCE. **API Library | Salesforce Developers**. Disponível em: <https://developer.salesforce.com/docs/apis>.