PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

COORDENAÇÃO ENSINO A DISTÂNCIA – CEAD

ESCOLA POLITÉCNICA E DE ARTES

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PROJETO INTEGRADOR III-A

Docente: José Ricardo Cosme Lérias Ribeiro

Discente: Isaías Correia De Morais

Matrícula: 1132024100768

GOIÂNIA
2025.

Sumário

1. Introdução

A evolução de soluções tecnológicas e o crescimento contínuo das aplicações corporativas tem exigido arquiteturas de softwares mais flexíveis, escaláveis e resilientes. Muitas empresas que iniciaram suas operações digitais com sistemas monolíticos hoje enfrentam dificuldades para acompanhar o aumento de usuários, a complexidade das funcionalidades e a necessidade constante de atualizações rápidas. Esse cenário torna indispensável a adoção de modelos arquiteturais modernos, como a arquitetura distribuída baseada em microsserviços.

Este projeto tem como foco o estudo e a proposta de migração de um sistema monolítico para uma arquitetura estruturada em microsserviços, considerando aspectos de decomposição, integração, comunicação entre serviços, controle de falhas e Segurança da Informação. O cenário utilizado é o da empresa fictícia **TechStore**, que atualmente opera uma plataforma de vendas online construída em um modelo monolítico e que, devido ao crescimento do negócio, passou a enfrentar problemas de desempenho, baixa escalabilidade e dificuldade de manunteção.

Com o objetivo de propor uma solução mais eficiente e moderna, este documento apresenta a análise do sistema atual, a identificação de seus principais módulos, o estudo do cenário simulado e a definição de uma nova arquitetura distribuída composta por serviços independentes. Além disso, são detalhadas a estratégia de migração, os mecanismos de segurança que garantem confidencialidade, integridade e disponibilidade, bem como diagramas representativos da arquitetura atual e da arquitetura proposta.

Assim, está introdução contextualiza a necessidade da transformação arquitetural e fundamenta a importância do trabalho, que busca demostrar como a migração para microsserviços pode melhorar o desempenho, a segurança, a escalabilidade e a manutenção de sistemas corporativos modernos. Tendo resultados positivos para a empresa e atendendo uma demanda significativa fazendo com oque clientes e usuários possam ser conquistados e façam uma avaliação de satisfação elevada.

2. Objetivo

2.1. Objetivo Geral

Propor e documentar uma solução completa para a migração de um sistema monolítico para uma arquitetura distribuída baseada em microsserviços, apresentando suas justificativas técnicas, modelagem, estratégias de integração, decomposição dos serviços, comunicação, controle de falhas e implementação de mecanismos de Segurança da Informação.

2.2. Objetivos Específicos

- Analisar a estrutura atual do sistema monolítico utilizado pela empresa fictícia **TechStore**, identificando suas limitações, riscos e gargalos.

- Identificar e descrever os principais módulos do sistema atual, como autenticação, produtos, pedidos e pagamentos.

- Propor uma arquitetura distribuída composta por microsserviços independentes, definindo claramente as responsabilidades de cada serviço.

- Apresentar estratégias de comunicação entre os microsserviços, incluindo REST, gRPC e mensageria, além de mecanismos de resiliência e tolerância a falhas.

- Definir uma estratégia gradual e segura para a migração do sistema monolítico para a arquitetura distribuída.

- Descrever e aplicar mecanismos de Segurança da Informação que garantam os princípios de Confidencialidade, integridade e Disponibilidade (CID).

- Elaborar diagramas UML representando a arquitetura atual (monolítica) e a arquitetura proposta (microsserviços).

- Produzir um documento técnico estruturado com linguagem clara, detalhada e fundamentada em boas práticas de engenharia de software.

3. Estudo de Caso

Este estudo de caso tem como objetivo analisar o cenário atual da empresa fictícia **TechStore**, compreender como o seu sistema de vendas foi construído, levantar seus principais requisitos e identificar os fatores que justificam a necessidade de uma migração tecnológica. A partir dessa análise inicial, é possível entender de forma mais consistente porque a arquitetura monolítica deixou de ser suficiente para atender â demanda da empresa e quais características devem ser consideradas na construção da nova solução baseada em microsserviços.

3.1. Levantamento Inicial

A **TechStore** iniciou suas operações com um sistema desenvolvido de maneira tradicional, no formato monolítico, atendendo inicialmente um volume pequeno de usuários e com pouca complexidade funcional. Esse sistema centralizava todas as funcionalidades em uma única aplicação – autenticação, produtos, pedidos, pagamentos, notificações e estoque.

Com o crescimento da empresa e o aumento expressivo das vendas online, o sistema começou a apresentar limitações de desempenho, instabilidades em períodos de grande fluxo e dificuldades de evolução. Durante o levantamento inicial, também foram identificados:

- Alto acoplamento entre módulos.
- Dependência de um único banco de dados.
- Lentidão no processamento de pedidos em horários de pico.
- Risco elevado de falhas afetarem todo o sistema.
- Dificuldade em implementar novas funcionalidades sem comprometer outras áreas.

Essas observações mostram que a arquitetura atual não acompanha mais a expansão do negócio.

3.2. Entendimento do Problema

Após o levantamento inicial, ficou evidente que o principal problema da **TechStore** não é apenas tecnológico, mas também estratégico. A empresa depende totalmente da sua plataforma digital, que hoje não consegue:

- Escalar horizontalmente para atender grandes volumes.
- Ser atualizada sem riscos elevados de paradas.
- Suportar novas integrações e funcionalidades.
- Garantir uma boa experiência ao cliente em grandes promoções.
- Permitir que equipes trabalhem de forma independente em partes diferentes do sistema.

Além disso, o sistema monolítico dificulta:

- ❖ Manutenção
- ❖ Testes
- ❖ Correção de bugs
- ❖ Implantação de melhorias
- ❖ Evolução do software

O modelo estático e rígido do monolito se tornou um obstáculo para o crescimento da empresa.

3.3. Necessidades Identificadas

Com base na análise e compreensão dos problemas enfrentados pela TechStore, foram levantadas as principais necessidades para a modernização do sistema:

- **Separação dos módulos em componentes independentes**, permitindo que cada parte seja escalada e mantida isoladamente.

- **Adoção de microsserviços**, cada um responsável por uma funcionalidade específica (autenticação, produtos, pedidos, pagamentos etc.).
- **Melhoria na resiliência**, de forma que falhas em um serviço não interrompam todo o sistema.
- **Aumento da performance**, principalmente no fluxo de pedidos e no catálogo de produtos.
- **Suporte a novos recursos**, como integrações externas e aumento de usuários simultâneos.
- **Implantação de mecanismos de segurança mais robustos**, garantindo Confidencialidade, Integridade e Disponibilidade.
- **Escalabilidade Horizontal**, permitindo aumentar apenas os módulos mais demandados.
- **Flexibilidade para evolução futura**, reduzindo riscos e facilitando novos desenvolvimentos.

---

4. Cenário Base Simulado (TechStore)

O cenário utilizado neste projeto tem como base a empresa fictícia **TechStore**, apresentada oficialmente na proposta do Projeto Integrador III-A. A **TechStore** é uma loja virtual especializada na comercialização de produtos eletrônicos, atuando exclusivamente por meio de sua plataforma de vendas online. Como qualquer empresa que depende de operações digitais, suas atividades envolvem o gerenciamento de usuários, o controle de produtos, o registro de pedidos e a finalização de pagamentos.

A empresa enfrenta um crescimento constante no número de clientes e na variedade de produtos oferecidos. Esse aumento natural da demanda tem gerado desafios operacionais para o sistema atual, que foi desenvolvido seguindo uma arquitetura monolítica. Essa arquitetura, embora funcional nos primeiros anos da plataforma, passou a apresentar problemas significativos à medida que o volume de acessos e transação cresceu.

O objetivo desta seção é apresentar o contexto da **TechStore**, descrever seu funcionamento atual e detalhar os problemas que justificam a necessidade de migração para uma nova arquitetura baseada em microsserviços.

4.1. Descrição da Empresa

A **TechStore** é uma empresa de comércio eletrônico voltada para a venda de produtos tecnológicos, tais como smartphones, notebooks, periféricos, acessórios e equipamentos eletrônicos diversos. Sua operação é totalmente digital, oque significa que todo o funcionamento depende diretamente da estabilidade e eficiência do sistema de vendas.

Com uma base de clientes em expansão e uma demanda crescente por produtos, a empresa enfrenta desafios comuns a negócios digitais que trabalham em larga escala. Entre as principais características da **TechStore**, destacam-se:

- Operação 100% online.
- Atendimento a clientes em todas as regiões do país.
- Catálogo com centenas de produtos.
- Processamento diário de diversos pedidos.
- Interação com serviços de pagamento e logística.

Todo esse ecossistema de operações exige um sistema ágil, seguro e capaz de se adaptar âs necessidades do mercado – algo que o modelo monolítico atual não consegue mais oferecer de forma satisfatória.

4.2. Descrição do Sistema de Vendas Atual

O sistema de vendas utilizado pela **TechStore** foi desenvolvido originalmente como uma aplicação monolítica. Isso significa que todas as funcionalidades da plataforma estão agrupadas em um único código e são executadas dentro do mesmo ambiente. Entre essas funcionalidades estão:

- Login e autenticação dos usuários.
- Cadastro e atualização de produtos.
- Registro e acompanhamento de pedidos.
- Processamento de pagamentos.
- Interface de gerenciamento administrativo.

Quando o sistema foi criado, esse modelo atendeu bem às necessidades da empresa. No entanto, com a expansão das operações, ficaram evidentes limitações que dificultam a continuidade do crescimento. O sistema apresenta lentidão em momento de maior movimento, os módulos possuem alta dependência entre si e qualquer atualização afeta toda a aplicação, aumentando o risco de indisponibilidade.

Além disso, eventuais falhas pontuais podem derrubar toda a plataforma por completo, oque representa um grande risco para um negócio que depende 100% da sua presença online.

4.3. Problemas Identificados

A partir da análise do funcionamento atual da plataforma da **TechStore**, é possível identificar uma série de problemas que comprometem o desempenho e a escalabilidade do sistema. Entre os principais pontos observados, destacam-se:

- **Dependência total entre módulos:** como a arquitetura é monolítica, qualquer falha em um módulo afeta toda a aplicação.
- **Dificuldade de manutenção:** mudança simples exigem a atualização e implantação completa do sistema.
- **Escalabilidade limitada:** o sistema só escala de maneira vertical, exigindo servidores mais robustos e aumentando custos.
- **Impacto no desempenho:** alta demanda resulta em lentidão, especialmente em horários de pico.
- **Risco elevado de indisponibilidade:** erros localizados podem causar interrupções em todo o serviço.
- **Baixa flexibilidade para evolução:** a introdução de novas funcionalidades demora mais devido à estrutura rígida.
- **Fragilidade em segurança:** a falta de isolamento torna mais difícil proteger dados e controlar acessos adequadamente.

Esses problemas revelam que a arquitetura atual não atende mais às necessidades da **TechStore**, tornando a migração para microsserviços uma solução apropriada e necessária.

---

5. Descrição do Sistema Atual (Monolítico)

O sistema utilizado atualmente pela **TechStore** foi desenvolvido seguindo uma arquitetura monolítica tradicional. Nesse modelo, todas as funcionalidades da aplicação – desde a autenticação até o processamento de pagamento – estão em um único bloco de código, executado no mesmo ambiente e compartilhando os mesmos recursos de infraestrutura.

Embora essa abordagem seja comum em projetos iniciais, ela tende a apresentar limitações conforme o sistema cresce em escala, número de usuários e complexidade de operações. Na **TechStore**, o monólito central concentra todos os módulos essenciais para o funcionamento da loja virtual, oque torna a aplicação altamente dependente de sua estrutura interna e suscetível a falhas generalizadas.

Por se tratar de um único sistema, qualquer alteração, atualização ou correção realizada em uma parte específica exige a implantação de toda a aplicação novamente. Isso gera riscos elevados de indisponibilidade e compromete a agilidade da equipe de desenvolvimento. Além disso, o aumento de acessos e operações simultâneas tem impactado diretamente o desempenho, tornando-se um fator crítico para a continuidade dos serviços.

A partir desta análise inicial, torna-se evidente que o modelo monolítico atual não oferece mais condições de acompanhar o crescimento da empresa, justificando a necessidade de migração para uma arquitetura mais flexível e escalável.

5.1. Funcionamento Atual

No modelo monolítico da **TechStore**, todos os módulos e funcionalidades estão integrados e executam de forma centralizada. O fluxo de funcionamento segue uma estrutura linear e independente. Entre suas principais características, destacam-se:

- **Código único:** todas as funcionalidades, interfaces e regras de negócio estão dentro de um mesmo projeto.
- **Banco de dados centralizado:** um único banco atende todas as tabelas e operações, o que aumenta o acoplamento entre as funcionalidades.

- **Escalabilidade vertical:** para lidar com picos de acesso, a empresa precisa aumentar a capacidade do servidor, tornando a operação mais cara e limitada.
- **Processamento encadeado:** cada módulo depende diretamente do outro, dificultando a paralelização e aumentando o tempo de resposta em operações de maior volume.
- **Implantação unificada:** qualquer mudança no sistema – mesmo mínima – exige uma nova implantação completa, elevando o risco de falhas generalizadas.

Esse modelo de funcionamento era adequado quando o sistema atendia a um público menor, mas com o crescimento das operações, tornou-se insuficiente para garantir performance, segurança e disponibilidade.

5.2. Limitações e Riscos

Durante a análise do sistema atual, foi possível identificar diversos pontos críticos que afetam diretamente o desempenho e a confiabilidade da aplicação. A seguir, são apresentadas as principais limitações e riscos associados à arquitetura monolítica.

1. Alto acoplamento entre módulos

Como todas as funcionalidades estão integradas, uma falha em qualquer módulo – como autenticação ou cadastro de produtos – pode comprometer todo o sistema.

2. Dificuldade para evoluir

A introdução de novas funcionalidades demanda alterações em um código extenso, aumentando o tempo de desenvolvimento e a ocorrência de erros.

3. Risco elevado de indisponibilidade

Uma simples atualização ou erro de configuração pode tornar o ambiente inteiro indisponível, impactando diretamente as vendas e a experiência do usuário.

4. Baixa escalabilidade

A escalabilidade vertical tem limites físicos e financeiros, dificultando acompanhar períodos de alta demanda, como promoções ou datas sazonais.

5. Vulnerabilidade de segurança

O compartilhamento do mesmo ambiente e banco de dados aumenta a exposição a falhas de integridade, vazamento de dados e ataques mais amplos.

6. Manutenção complexa

A grande quantidade de códigos concentrada em um único local torna mais difícil identificar e corrigir problemas específicos.

7. Dependência de implantação total

Cada atualização exige reimplantar todo o sistema, o que eleva o risco de interrupções e erros imprevistos.

Essas limitações deixam claro que, embora funcional, o sistema monolítico não atende mais às necessidades da **TechStore** em termos de desempenho, segurança e escalabilidade.

---

6. Análise dos Módulos do Sistema (Arquitetura Atual – Monolítica)

Nesta seção são analisadas, de forma detalhada, os principais módulos que compõem o sistema monolítico da **TechStore**. O objetivo é identificar responsabilidades, pontos de acoplamento, gargalos de desempenho e riscos operacionais que justificam a migração para uma arquitetura distribuída. As observações abaixo foram extraídas e adaptadas do levantamento inicial presente na documentação do projeto.

6.1. Módulo de Autenticação

Descrição: controla o cadastro de usuários, login, recuperação de senha, gerenciamento de sessões e autorização (perfis/roles).

Comportamento no monolito: funções de autenticação acessam diretamente o banco central e expõem APIs internas consumidas por outras funcionalidades (p. ex. painel administrativo, checkout).

Problemas e impactos:

- **Ponto único de falha:** indisponibilidade ou lentidão no módulo de autenticação afeta toda a plataforma (usuários não conseguem logar; sessões expiram).
- **Acoplamento com outros módulos:** atualizações nas regras de autenticação implicam implantação do monolito inteiro.
- **Risco de segurança ampliado:** credenciais e tokens manipulados no mesmo escopo do restante do código aumentam superfície de ataque.
- **Dificuldade de escalonamento seletivo:** não é possível escalar apenas a autenticação sem replicar toda a aplicação.

Recomendação (para migração): isolar autenticação como serviço independente, usar tokens (JWT com refresh controlado), armazenar credenciais em cofre/secret manager e implementar limites de taxa (rate limiting).

6.2. Módulo de Gestão de Produtos

Descrição: gerencia catálogo (cadastro, edição, categorias, imagens), consultas de listagem e controle de estoque.

Comportamento no monolito: operações de leitura e escrita compartilham recursos com pedidos e demais módulos.

Problemas e impactos:

- **Alto volume de leitura:** consultas ao catálogo (páginas, buscas) geram grande carga que compromete outras funcionalidades.
- **Conflitos de concorrência:** alterações de estoque durante picos (vendas simultâneas) podem gerar inconsistências quando não há isolamento transacional adequado.
- **Impacto em deploys:** mudanças de estrutura de produto demandam deploys completos, arriscando indisponibilidade.

Recomendação: transformar gestão de produtos em serviço com banco desacoplado, implementar cache estratégico (CDN / Redis) para leituras pesadas e endpoints de consulta otimizados (paginação, filtros).

6.3. Módulo de Processamento de Pedidos

Descrição: realiza fluxo de checkout, valida itens, reserva estoque, cria pedidos e coordena confirmação com pagamentos e logísticas.

Comportamento no monolito: processamento síncrono com dependências diretas ao módulo de pagamentos e ao banco central.

Problemas e impactos:

- **Latência em picos:** processamento encadeado aumenta tempo de resposta em horários de alta demanda.
- **Falhas em cascata:** problemas em pagamentos ou estoque podem travar o processamento de pedidos e impactar conversões.
- **Dificuldade para testes e rollback:** testes em ambiente de produção são arriscados porque uma falha pode afetar fluxo completo.

Recomendação: torná-lo um serviço independente que utilize mensageria para operações longas (ex.: reserva de estoque, confirmação de pagamento), permitir processamento assíncrono e implementar sagas ou orquestração para manter consistência distribuída.

6.4. Módulo de Pagamentos

Descrição: interface com gateways e provedores financeiros, valida transações, atualiza status de pagamento e disponibiliza respostas ao módulo de pedidos.

Comportamento no monolito: integrações externas são feitas diretamente do código principal, sem isolamento.

Problemas e impactos:

- **Alta criticidade:** falhas ou lentidão impactam diretamente receitas (impossibilidade de concluir vendas).
- **Exposição de dados sensíveis:** manuseio de dados de pagamento sem isolamento aumenta requisitos de conformidade (PCI-DSS).
- **Dificuldade de alternância de provedores:** trocar/avaliar um gateway exige mudanças em toda aplicação.

Recomendação: isolar pagamentos como microserviço separado com canal seguro para transações, uso de tokenização para dados sensíveis, filas para processar callbacks e reconciliação financeira, além de testes de integração isolados.

---

7. Proposta de Arquitetura Distribuída em Microsserviços (Arquitetura Nova)

A nova arquitetura proposta para a **TechStore** tem como base o modelo de microsserviços, no qual cada funcionalidade do sistema é separada em serviços independentes, autônomos e escaláveis. Essa abordagem tem como objetivo resolver os problemas identificados no sistema monolítico, oferecendo maior flexibilidade, facilidade de manutenção, resiliência e melhor desempenho em ambientes de alto volume de requisições.

Na arquitetura distribuída, cada microsserviço possui seu próprio ambiente de execução, banco de dados, escalabilidade específica e mecanismo de comunicação. Isso reduz o acoplamento entre módulos e permite que cada parte

evolua sem comprometer o restante da aplicação. A seguir, são descritos os microsserviços propostos, os tipos de comunicação utilizados e os mecanismos de resiliência adotados.

7.1. Definição e Responsabilidades dos Microsserviços

A decomposição do sistema monolítico da **TechStore** resultou na identificação de microsserviços essenciais para a operação da plataforma. Cada serviço foi definido com uma responsabilidade clara, seguindo o princípio do Single Responsibility (SRP), garantindo uma arquitetura organizada e eficiente.

1. **Microsserviço de Autenticação (Auth Service)**

Responsabilidades:

- Gerenciar cadastro e autenticação de usuários.
- Gerar e validar tokens de acesso (JWT).
- Controlar permissões e perfis.
- Executar recuperação e redefinição de senha.

Justificativa:

A separação da autenticação traz maior segurança, escalabilidade e controle de acessos.

---

2. **Microsserviço de Produtos (Product Service)**

Responsabilidades:

- Cadastro, listagem, atualização e remoção de produtos.
- Controle de estoque.
- Gerenciamento de categorias e descrições.

Justificativa:

Serviços de catálogo costumam ter grande volume de consultas.

---

3. Microsserviço de Pedidos (Order Service)

Responsabilidades:

- Criação e gerenciamento de pedidos.
- Validação dos itens do carrinho.
- Integração com o serviço de pagamentos e estoque.
- Atualização de status (pendente, pago, enviado).

Justificativa:

O fluxo de pedidos é uma das áreas mais críticas. Isolá-lo reduz falhas e permite processamentos paralelos.

---

4. Microsserviço de Pagamentos (Payment Service)

Responsabilidades:

- Processar transações financeiras.
- Integrar com provedores externos (gateways).
- Validar pagamentos e informar respostas ao serviço de pedidos.
- Gerenciar callbacks e conciliação financeira.

Justificativa:

Por lidar com dados sensíveis, exige segurança reforçada e isolamento total.

---

5. Microsserviço de Notificações (Notification Service)

Responsabilidades:

- Enviar e-mails, SMS e notificações push.
- Informar usuários sobre status de pedidos e confirmação de pagamento.
- Receber eventos do Order Service e Payment Service.

Justificativa:

Desacopla comunicações assíncronas e evita sobrecarregar serviços principais.

---

6. **API Gateway**

Responsabilidades:

- Centralizar o acesso dos clientes aos microsserviços.
- Autenticar e encaminhar requisições.
- Aplicar regras de segurança como rate limiting.

7.2. Comunicação entre Serviços (REST gRPC, Mensageria)

A comunicação em uma arquitetura distribuída deve ser cuidadosamente projetada para garantir eficiência, flexibilidade e tolerância a falhas. Na proposta da TechStore, recomenda-se uma combinação de três mecanismos principais:

1. **Comunicação via REST (HTTP/JSON) – Síncrona**

Usada nos casos em que é necessário obter respostas imediatas:

- Autenticação
- Consultas ao catálogo de produtos
- Atualização de dados administrativos

Vantagens:

- Simples implementação
- Compatível com diferentes linguagens
- Fácil integração externa

---

2. **Comunicação via gRPC – Síncrona, alto desempenho**

Indicada para comunicações internas entre microsserviços que exigem baixa latência, como:

- Comunicação entre Pedido ↔ Pagamento
- Atualizações de estoque

Benefícios:

- Respostas rápidas
- Suporte a streaming
- Menos sobrecarga que REST

---

3. **Comunicação via Mensageria (RabbitMQ ou Kafka) – Assíncrona**

Utilizada para operações que não exigem resposta imediata:

- Envio de notificações
- Atualização de status do pedido
- Comunicação entre Pedido → Pagamento
- Eventos de sistema (ex.: “pedido criado”)

Benefícios:

- Alta resiliência
- Evita travamentos se um serviço estiver offline
- Garantia de entrega (mesmo em falhas)
- Filas independentes que permitem escalar módulos isoladamente

Essa abordagem híbrida garante equilíbrio entre simplicidade, desempenho e robustez.

7.3. Controle de Falhas e Resiliência

Para manter a plataforma disponível mesmo em situações de erro, são adotados mecanismos de resiliência e tolerância a falhas:

1. **Circuit Breaker**
   - Impede tentativas repetidas de chamar um serviço instável.
   - Evita falhas em cascata na arquitetura.

---

2. **Retry Automático**
    - Reenvio automático de requisições quando ocorrem falhas temporárias.

---

3. **Timeouts Configurados**
    - Cada serviço possui limite de tempo para aguardar resposta.
    - Impede travamentos prolongados no sistema.

---

4. **Balanceamento de Carga**
    - Distribui requisições entre múltiplas instâncias do mesmo serviço.

---

5. **Health Checks e Monitoramento**

Ferramentas como Prometheus, Grafana e ELK são usadas para:

- Monitorar performance
- Detectar falhas
- Gerar alertas
- Registrar logs centralizados

---

6. **Escalabilidade Horizontal**
    - Permite aumentar apenas a quantidade de instâncias dos serviços mais exigidos, como Produtos e Pedidos.

---

7. **Bancos de Dados Independentes**
    - Evita bloqueios e concorrência desnecessária.
    - Reduz o risco de falha total da aplicação.

---

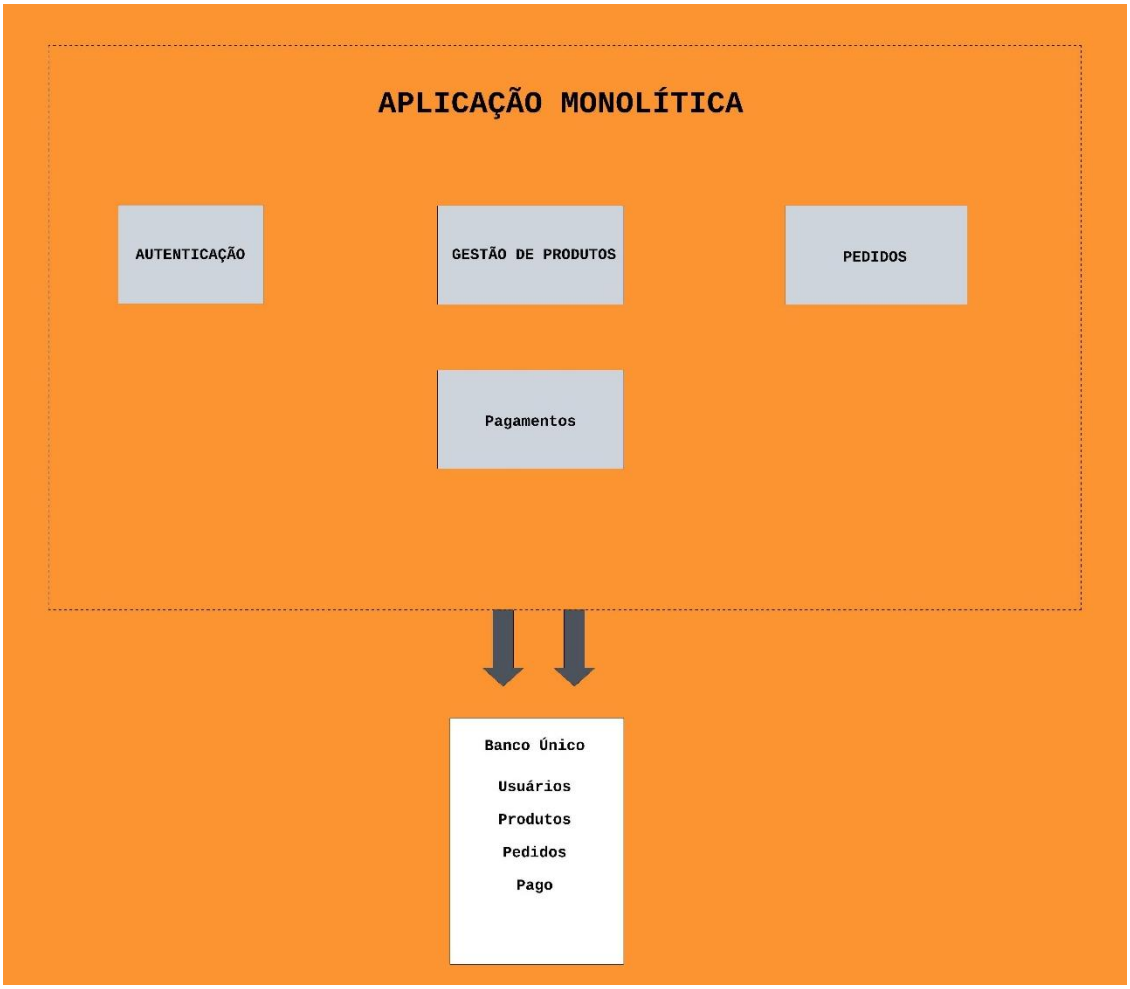
8. Diagramas da Arquitetura

Esta seção apresenta a representação visual das duas arquiteturas analisadas na **TechStore**: a arquitetura monolítica atual e a arquitetura distribuída proposta baseada em microsserviços. Os diagramas têm como objetivo ilustrar os componentes principais, suas interações e as diferenças estruturais entre os dois modelos, facilitando o entendimento da evolução arquitetural sugerida no projeto.

8.1. Arquitetura Antiga – Sistema Monolítico

A seguir, apresenta-se o diagrama que representa a arquitetura monolítica utilizada pela **TechStore**. Nele é possível observar que todas as funcionalidades estão concentradas em uma única aplicação, com um banco de dados central e forte acoplamento entre os módulos.

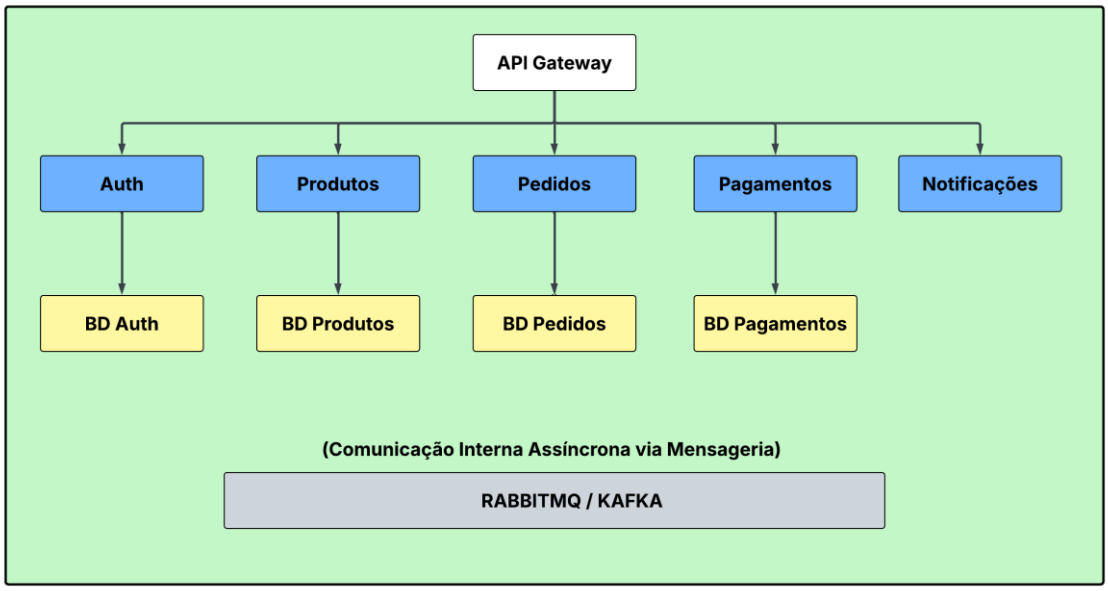
Diagrama (Monolito)



8.2. Arquitetura Proposta – Microsserviços

O diagrama abaixo representa a arquitetura distribuída recomendada. Cada microsserviço possui responsabilidades próprias, banco isolado e comunicação controlada via API Gateway ou mensageria.

Diagrama (Microsserviços)



---

9. Estratégia de Migração do Monolito para Microsserviços

A migração do sistema monolítico da **TechStore** para uma arquitetura distribuída baseada em microsserviços deve ser planejada com cuidado, seguindo etapas progressivas que evitem interrupções no ambiente produtivo. A estratégia adotada prioriza a continuidade do serviço, a segurança das informações e a consistência dos dados durante todo o processo. Além disso, a migração gradual reduz riscos e permite que os serviços sejam validados conforme são liberados.

A seguir, apresenta-se a proposta completa de migração, estruturada em fases sequenciais com foco em integração, escalabilidade e redução do acoplamento entre os módulos do sistema atual.

9.1. Fases da Migração

A migração será dividida em oito fases, garantindo um processo seguro, previsível e tecnicamente consistente.

Fase 1 – Levantamento e análise do monolito

- ✓ Mapeamento completo dos módulos existentes.
- ✓ Identificação de dependências entre componentes.
- ✓ Análise de pontos críticos que exigem maior cuidado.

Fase 2 – Definição dos microsserviços

- ✓ Criação do modelo de decomposição do monolito.
- ✓ Definição dos serviços: Autenticação, Produtos, Pedidos, Pagamentos, Notificações.
- ✓ Escolha das tecnologias, banco de dados e padrões de comunicação.

Fase 3 – Implementação do API Gateway

- ✓ O Gateway passa a ser o único ponto de entrada para o sistema.
- ✓ Roteamento centralizado de requisições.
- ✓ Autenticação inicial via tokens.
- ✓ Isso permite que microsserviços sejam incluídos sem impactar o cliente.

Fase 4 – Extração do Serviço de Autenticação

- ✓ Criação do Auth Service com banco independente.
- ✓ Substituição gradual das chamadas internas do monolito para o novo serviço.
- ✓ Testes de login, perfis e tokens JWT.

Fase 5 – Migração do módulo de Produtos

- ✓ Criação do Product Service.
- ✓ Sincronização inicial dos dados de catálogo.
- ✓ Roteamento de consultas via API Gateway.
- ✓ Desativação progressiva do módulo monolítico.

Fase 6 – Migração do módulo de Pedidos

- ✓ Separação do Order Service.
- ✓ Adoção de mensageria para integração com o Pagamentos e Estoque.
- ✓ Ajuste no fluxo do checkout para o novo serviço.

Fase 7 – Integração com o Serviço de Pagamentos

- ✓ Criação do Payment Service.
- ✓ Integração segura com gateways financeiros.
- ✓ Configuração de callbacks e confirmação assíncrona.
- ✓ Retirada gradual do módulo de pagamentos do monolito.

Fase 8 – Desativação total do monolito

- ✓ Remoção completa do código antigo.
- ✓ Cada microsserviço passa a operar de forma totalmente independente.
- ✓ Monitoramento contínuo pós-migração.

9.2. Integração e Escalabilidade

A integração entre os serviços será realizada de forma híbrida, combinando comunicação síncrona e assíncrona:

✓ Comunicação síncrona (REST / gRPC)

Usada quando o retorno do serviço é imediato:

- ✓ Autenticação
- ✓ Consultas a produtos
- ✓ Verificação de estoque

✓ Comunicação assíncrona (Mensageria – RabbitMQ ou Kafka)

Usada nos fluxos:

- ✓ Atualização de status de pedidos
- ✓ Notificações ao cliente
- ✓ Processamento de pagamentos

Essa abordagem aumenta a resiliência, evita gargalos e reduz o impacto de falhas pontuais.

✓ Escalabilidade Horizontal

Após a migração, cada microsserviço poderá escalar individualmente:

- ✓ **Pedidos** escala em datas com alto volume (Black Friday)
- ✓ **Produtos** escala em buscas e listagens
- ✓ **Pagamentos** escala durante picos de checkout

Isso elimina a limitação da escalabilidade vertical do monolito e reduz custos de infraestrutura.

9.3. Riscos da Migração e Mitigações

A migração de um monolito para microsserviços envolve riscos técnicos e operacionais. A seguir, destacam-se os principais riscos e suas respectivas estratégias de mitigação.

Risco 1 – Inconsistência de dados

Durante a migração, dados podem divergir entre sistemas.

Mitigação:

- ❖ Utilização de sincronização incremental
- ❖ Uso de mensageria para manter estados alinhados
- ❖ Backups antes de cada fase crítica

Risco 2 – Indisponibilidade temporária

Modificações estruturais podem causar interrupções.

Mitigação:

- ❖ Migração faseada
- ❖ Deploy com zero downtime
- ❖ Testes automatizados e rollback rápido

Risco 3 – Falhas de comunicação entre serviços

Microsserviços dependem de rede. Falhas podem se propagar.

Mitigação:

- ❖ Implementação de circuit breaker
- ❖ Timeouts configurados
- ❖ Retries automáticos

Risco 4 – Aumento da complexidade operacional

Mais serviços = mais ambientes para monitorar.

Mitigação:

- ❖ Ferramentas centralizadas de observabilidade
- ❖ Dashboards de logs e métricas (Grafana, Kibana, Prometheus)

Risco 5 – Equipe não familiarizada com arquitetura distribuída

Maior curva de aprendizado.

Mitigação:

- ❖ Treinamentos técnicos
- ❖ Documentação clara
- ❖ Padrões internos de desenvolvimento e APIs

---

10. Segurança da Informação na Arquitetura Proposta (CID)

A migração para uma arquitetura distribuída baseada em microsserviços exige a adoção de controles robustos de Segurança da Informação. Para garantir a proteção adequada dos dados e a continuidade dos serviços da TechStore, foram implementadas práticas alinhadas aos princípios fundamentais da segurança: Confidencialidade, Integridade e Disponibilidade (CID).

A seguir, são descritos os mecanismos adotados em cada pilar, considerando o funcionamento distribuído da nova arquitetura e as recomendações de normas como ISO/IEC 27002 e práticas modernas de engenharia de software segura.

10.1. Confidencialidade

A Confidencialidade visa proteger os dados contra acessos não autorizados.

Na arquitetura de microsserviços, esse pilar foi reforçado com as seguintes medidas:

1. Autenticação via tokens JWT

O microsserviço de Autenticação gera tokens criptografados, utilizados para validar o acesso de cada usuário por meio do API Gateway.

Isso impede que serviços internos sejam acessados sem autorização.

2. Criptografia de dados em trânsito (HTTPS/TLS)

Toda comunicação entre:

- clientes,
- API Gateway
- e microsserviços

é realizada por meio de HTTPS com TLS 1.2 ou superior, prevenindo interceptação de dados sensíveis.

3. Criptografia de dados em repouso

Informações confidenciais (usuários, pagamentos, pedidos) são armazenadas com criptografia no banco de dados de cada microsserviço.

4. Bancos de dados separados

Cada microsserviço possui seu próprio banco, reduzindo riscos de acesso indevido e limitando a exposição de dados em caso de incidente de segurança.

5. Controle de acesso baseado em papéis (RBAC)

A plataforma define perfis de acesso (usuário, administrador, operador), garantindo que somente pessoas autorizadas executem ações críticas.

6. Proteção de credenciais e chaves

Chaves de API, tokens e senhas de banco são armazenados em cofre seguro (Secret Manager), evitando exposição no código.

Essas medidas garantem que apenas usuários e serviços devidamente autenticados acessem informações sensíveis da TechStore.

10.2. Integridade

A Integridade assegura que os dados sejam exatos, completos e não adulterados.
Na arquitetura proposta, esse pilar é garantido por meio de:

1. Hash seguro para senhas

Senhas são armazenadas utilizando algoritmos como bcrypt ou Argon2, evitando que possam ser reconstruídas mesmo em caso de vazamento.

2. Logs e trilhas de auditoria

Cada microsserviço mantém logs detalhados das operações realizadas, permitindo:

- rastrear falhas
- monitorar ações suspeitas
- validar transações

3. Assinatura digital em mensagens

Em comunicação assíncrona via mensageria (RabbitMQ/Kafka), mensagens podem ser assinadas digitalmente para evitar adulteração no trânsito.

4. Mecanismos de consistência distribuída

Operações críticas entre microsserviços, como pedido + pagamento + estoque, utilizam padrões como SAGA, garantindo consistência entre transações distribuídas.

5. Validação de entrada e saída (sanitização)

Todas as requisições passam por validação para impedir:

- dados inválidos
- comandos maliciosos
- manipulação intencional de informações

Com isso, a integridade dos dados é preservada em todas as etapas do fluxo da TechStore.

10.3. Disponibilidade

A Disponibilidade assegura que o sistema permaneça acessível mesmo em cenários de falhas ou picos de uso.

Para garantir esse pilar, foram adotadas as seguintes medidas:

1. Escalabilidade horizontal

Cada microsserviço pode ser escalado individualmente, aumentando o número de instâncias conforme a demanda, especialmente nos serviços de:

- Produtos
- Pedidos
- Pagamentos

2. Balanceamento de carga

O API Gateway distribui requisições entre diferentes instâncias dos serviços, evitando sobrecarga e aumentando a capacidade de atendimento simultâneo.

3. Replicação de bancos de dados

Bancos dos serviços podem operar com réplicas para evitar indisponibilidade em caso de falha de um nó.

4. Circuit Breaker

Evita falhas em cascata, isolando serviços instáveis e impedindo que um erro derrube toda a aplicação.

5. Timeouts e tentativas automáticas (Retry)

Garantem que falhas temporárias na rede ou nos serviços sejam tratadas de forma automática.

6. Mensageria para comunicação assíncrona

Se um serviço estiver temporariamente fora do ar, mensagens ficam na fila e são processadas assim que o serviço voltar, evitando perda de informação.

7. Monitoramento contínuo

Ferramentas como Prometheus, Grafana e ELK Stack permitem:

- detectar falhas em tempo real
- acompanhar métricas de desempenho
- monitorar segurança e disponibilidade

Esses mecanismos tornam o sistema mais resiliente e preparado para operar em ambiente distribuído.

---

11. Conclusão

A realização deste projeto permitiu uma compreensão aprofundada dos desafios enfrentados pela **TechStore** em sua arquitetura monolítica atual, evidenciando limitações que comprometem desempenho, segurança, escalabilidade e continuidade operacional. Por meio da análise estruturada dos módulos do sistema, foi possível identificar gargalos críticos, como alto acoplamento, dificuldade de manutenção, riscos de falhas generalizadas e baixa capacidade de evolução tecnológica.

A partir desse diagnóstico, foi proposta uma arquitetura distribuída baseada em microsserviços, estruturada de forma a isolar responsabilidades, reduzir dependências internas e permitir que cada serviço opere de maneira autônoma. A decomposição em serviços como Autenticação, Produtos, Pedidos, Pagamentos e Notificações possibilita maior flexibilidade, escalabilidade horizontal e evolução independente de cada componente da plataforma.

Além disso, a definição de mecanismos de comunicação síncrona e assíncrona, a utilização de mensageria, o emprego de API Gateway e a aplicação de padrões de resiliência garantem maior robustez à solução proposta. A estratégia de migração apresentada contempla um processo gradual, seguro e cuidadosamente planejado, visando minimizar riscos e assegurar a transição com o menor impacto possível.

Outro ponto fundamental foi a incorporação dos pilares de Segurança da Informação — Confidencialidade, Integridade e Disponibilidade — que fortalecem a proteção dos dados e asseguram um funcionamento confiável da plataforma. Medidas como criptografia, controle de acesso, logs de auditoria, replicação, monitoramento e tolerância a falhas contribuem diretamente para a construção de um ambiente moderno e seguro.

Dessa forma, a migração para uma arquitetura de microsserviços não apenas resolve os problemas enfrentados pelo modelo monolítico, mas também prepara a **TechStore** para o crescimento sustentável, a ampliação de funcionalidades e a adaptação contínua às demandas do mercado digital. O projeto demonstra

claramente como uma abordagem arquitetural bem planejada pode elevar o nível de eficiência, segurança e escalabilidade de sistemas corporativos modernos.

---

12. Referências Bibliográficas

- **PRESSMAN, Roger S.; MAXIM, Bruce R.** Engenharia de Software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.
- **SOMMERVILLE, Ian.** Engenharia de Software. 10. ed. São Paulo: Pearson, 2020.
- **TANENBAUM, Andrew S.; VAN STEEN, Maarten.** Sistemas Distribuídos: princípios e paradigmas. 3. ed. São Paulo: Pearson, 2017.
- **BASS, Len; CLEMENTS, Paul; KAZMAN, Rick.** Software Architecture in Practice. 4. ed. Boston: Addison-Wesley, 2021.
- **SCHNEIER, Bruce.** Applied Cryptography. 20th Anniversary Edition. Wiley, 2015.
- **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION.** ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection – Information security controls. Geneva: ISO, 2022.
- **COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim.** Sistemas distribuídos: conceitos e projeto. 4. ed. Porto Alegre: Bookman, 2007.