SENAC

TechCommerce

Versão 1.0

Carlos Eduardo Martins dos Santos, 1142463887
Murilo Augusto Vieira, 1142521540
Nathan Henrique Vieira Ferreira, 1142502197
Otavio Augusto Reis Almeida, 1142445679
Tiago de Almeida Nobre, 1141983013

Projeto Integrador: Análise de Soluções de Tecnologia da Informação

São Paulo - 2025

Termo de Abertura de Projeto - *Project Charter*

Empresa / Órgão / Setor/ Programa: TechCommerce Ltda			
Nome do projeto:	TechCommerce		
Gerente do projeto:	**Nathan Henrique Vieira Ferreira**		
Elaborado por:	Nathan Henrique Vieira Ferreira	**Versão:**	1.0
Aprovado por:	Carlos Henrique Verissimo Pereira		
Assinatura:		**Data de aprovação:**	27/02/2025

Justificativa do projeto
Este projeto é necessário para atender à crescente demanda por compras online de produtos eletrônicos, expandindo o alcance e oferecendo uma experiência de compra moderna e conveniente. Expandir a presença no mercado digital. Aumentar a receita através de vendas online. Melhorar a experiência do cliente com uma plataforma intuitiva. Otimizar processos internos e reduzir custos. Benefícios: O projeto ampliará o alcance de mercado, reduzirá custos operacionais, aumentará a competitividade e permitirá a coleta de dados para decisões estratégicas, além de fidelizar clientes.

Objetivo(s) do Projeto
O objetivo do projeto é desenvolver um e-commerce voltado para produtos de hardware que alcance o público tech

Descrição do produto do projeto
O projeto **TechCommerce** tem como objetivo desenvolver uma plataforma de e-commerce especializada na venda de produtos de hardware. A plataforma contará com um catálogo abrangente de componentes de computador, periféricos, acessórios, e dispositivos tecnológicos, todos voltados para o público tech.

Premissas (hipóteses) e restrições para o projeto	
Premissas (hipóteses)	**Restrições**
• Os fornecedores de hardware terão disponibilidade regular de produtos para manter o estoque abastecido. • A equipe de desenvolvimento possui as competências técnicas necessárias para entregar o projeto dentro do prazo. • Os servidores da plataforma serão capazes de suportar o tráfego esperado, especialmente durante períodos de alta demanda, como Black Friday. • Os requisitos de segurança para proteção de dados dos clientes serão atendidos, incluindo conformidade com a LGPD (Lei Geral de Proteção de Dados).	• **Prazo**: O projeto deve ser concluído até 30/05/2025. • **Custo**: O orçamento não deve exceder os valores alocados nas macros fases. • **Qualidade**: A plataforma deve ser lançada sem falhas críticas, especialmente nas áreas de pagamento e segurança. • **Tecnologia**: O projeto deve ser desenvolvido usando as tecnologias acordadas, sem mudanças significativas na Stack tecnológica que possam impactar o prazo ou o orçamento.

Macro fase	Data limite	Custo
Abertura do Projeto	27/02/2024	R$ 2.000
Gerenciamento do escopo	25/03/2024	R$ 4.000
Gerenciamento do cronograma	25/03/2024	R$ 5.000
UI	25/03/2024	R$ 3.500
Elaboração da estrutura do Projeto (UML)	30/04/2024	R$ 4.300
Modelo do Banco de Dados	30/04/2024	R$ 5.700
Design Pattern, Ferramentas/Frameworks Adotados e Justificativas	30/04/2024	R$ 3.000
Arquitetura do Projeto	01/05/2024	R$ 3.600
Desenvolvimento do Projeto	05/05/2024	R$ 13.000
Testes	20/05/2024	R$ 2.980
Finalização do Projeto	27/05/2024	R$ 200
	Custo total	R$ 47.280,00

Principais envolvidos

- **Gerente do Projeto**: Nathan Henrique Vieira Ferreira
- **Desenvolvedores**: Carlos, Murilo, Tiago, Nathan, Otavio, Vinícius
- **Cliente/Stakeholder**: Pessoas que tem interesse em comprar Hardwares
- **Equipe de Suporte Técnico**: Carlos, Murilo, Tiago, Nathan, Otavio, Vinícius

Designação de gerente	
Gerente do projeto	Nathan Henrique Vieira Ferreira
Limites de autoridade	- **Orçamentário**: • **Decisão sem aprovação**: O gerente de projeto pode aprovar despesas até um certo valor (por exemplo, R$ 10.000) sem precisar consultar os superiores. • **Acima do limite**: Qualquer despesa acima desse valor precisa de aprovação de um diretor ou do comitê gestor. - **Recursos Humanos**: • **Contratação**: O gerente de projeto pode contratar ou alocar recursos humanos dentro da equipe do projeto, mas qualquer nova contratação ou mudança significativa na equipe pode exigir aprovação. - **Mudanças no Escopo**: • **Pequenas mudanças**: O gerente de projeto pode aprovar pequenas alterações no escopo do projeto. • **Mudanças maiores**: Mudanças significativas que podem afetar o prazo, o orçamento ou os objetivos do projeto precisam de aprovação de partes interessadas superiores. - **Prazos**: • **Ajustes menores**: O gerente pode ajustar prazos internos sem comprometer a data final de entrega. • **Revisão de prazo final**: A necessidade de mudar a data de entrega final requer consulta e aprovação.

Ecossistema da Solução

O ecossistema da solução é composto por um conjunto de microserviços independentes e especializados, desenvolvidos com tecnologias modernas como **Golang**, **Quarkus (Java)** e **TypeScript (Hono.js)**, organizados em um **monorepo**. Cada serviço é responsável por uma parte específica da aplicação, como gerenciamento de identidade, catálogo de produtos, pedidos e envio de e-mails.
Toda a comunicação entre os usuários e os serviços é centralizada através de um **API Gateway Kong**, que atua como camada de entrada, roteando requisições HTTP para os

serviços apropriados. O frontend, que será desenvolvido em **React**, irá consumir essas APIs para fornecer uma interface responsiva e dinâmica aos usuários.
A autenticação é realizada por meio de **JWT (JSON Web Token)**, com suporte a múltiplas **roles e permissões**, garantindo um controle de acesso seguro e flexível. Além disso, a aplicação utiliza o **Amazon SQS** como mecanismo de mensageria assíncrona, permitindo uma comunicação desacoplada entre serviços. O armazenamento de dados é feito em **bancos de dados distintos por serviço**, garantindo isolamento e escalabilidade. Imagens de produtos, por exemplo, são armazenadas em um **bucket S3**.
A arquitetura também prevê fácil integração com sistemas externos e suporte a logs, monitoramento e escalabilidade horizontal, visando sempre desempenho, segurança e facilidade de manutenção.

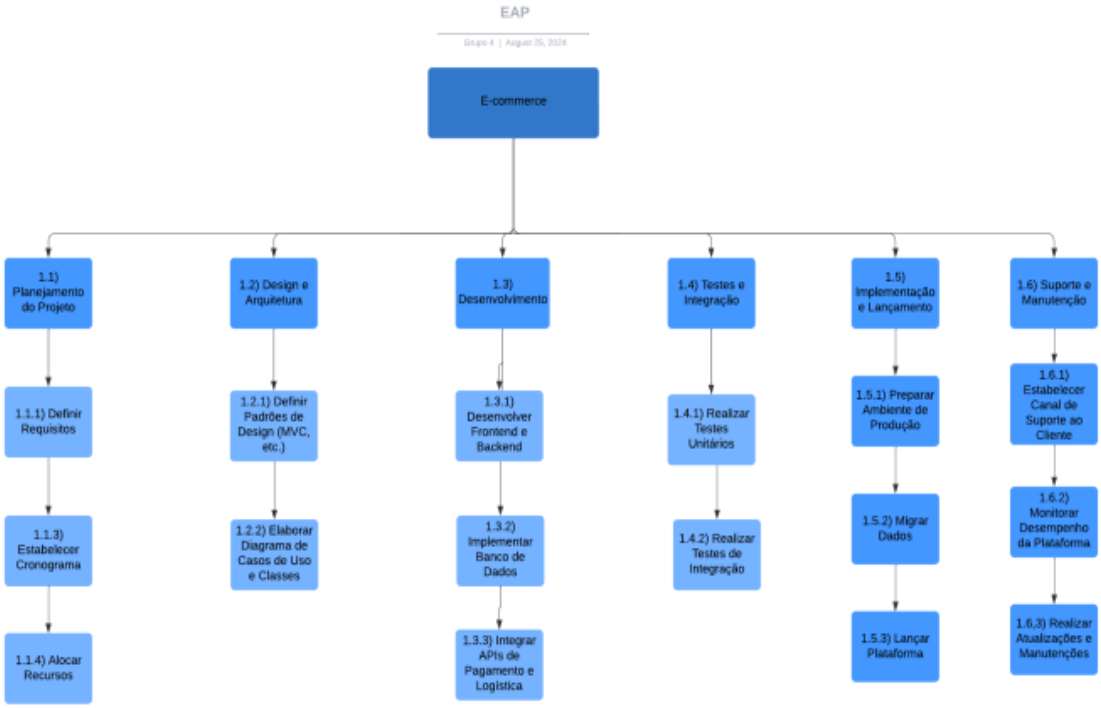
Arquitetura de Software
A arquitetura da solução adota uma abordagem **baseada em microsserviços desacoplados**, cada um com sua responsabilidade bem definida. Os serviços são escritos utilizando linguagens como **Go**, **Java com Quarkus** e **TypeScript com Hono.js**, escolhidas conforme as características e necessidades de cada domínio. As APIs seguem o modelo **RESTful**, expostas através do **API Gateway Kong**, que centraliza o roteamento, autenticação via JWT e demais funcionalidades de segurança e monitoramento. Cada microserviço possui seu próprio **banco de dados isolado**, garantindo autonomia e facilidade de escalabilidade. A persistência de dados é feita de forma independente, utilizando **PostgreSQL** como padrão principal. Para comunicação assíncrona entre os serviços, é utilizado o **Amazon SQS**, permitindo maior resiliência e desacoplamento entre produtores e consumidores de eventos. O **bucket S3** é utilizado para armazenamento de imagens e arquivos estáticos do sistema. A infraestrutura é totalmente **containerizada com Docker** e gerenciada via **Terraform**, o que garante portabilidade, reprodutibilidade e automação no provisionamento dos recursos. A autenticação e autorização são implementadas com base em **JWTs** e um modelo robusto de **roles e permissions**, permitindo uma segurança granular e eficiente. Essa arquitetura modular e extensível permite que a solução cresça de forma organizada, mantenha a segurança dos dados e facilite a manutenção e a evolução contínua do sistema.

Tecnologias	
Linguagem	O backend da aplicação é desenvolvido em **Java**, utilizando o framework **Spring Boot** para criação da **API REST**. No frontend, é utilizado **React**, proporcionando uma interface dinâmica e responsiva para os usuários.
Banco de dados	O banco de dados utilizado é o **PostgreSQL**, garantindo robustez, escalabilidade e suporte a transações complexas. Ele é gerenciado dentro de um ambiente **Docker**, proporcionando maior flexibilidade no desenvolvimento e na implantação.
Repositórios	A autenticação é realizada exclusivamente por meio de **JWT (JSON Web Token)**, garantindo um controle seguro e sem necessidade de sessões no servidor. Além disso, são adotadas práticas como criptografia de senhas e configurações de segurança para proteger os endpoints da

	API.
Segurança	O código-fonte do projeto é armazenado em **repositórios Git**, possibilitando controle de versão eficiente, colaboração entre desenvolvedores e integração contínua.
Provedores em Nuvem	A aplicação é implantada na **Azure**, garantindo alta disponibilidade, escalabilidade e suporte a diversos serviços complementares, como banco de dados gerenciado, monitoramento e balanceamento de carga.

Requisitos Funcionais	
Cadastro e Autenticação	• O sistema permite o cadastro de usuários com nome, e-mail e senha. • O sistema permite o login via e-mail e senha. • O sistema permite a recuperação de senha via e-mail.
Gestão de Produtos	• O sistema permite a busca de produtos por nome e categoria. • O sistema exibe detalhes dos produtos, incluindo imagens e especificações. • O sistema permite a adição e remoção de produtos ao carrinho de compras.
Área do Cliente	• O sistema permite a edição dos dados cadastrais. • O sistema exibe o histórico de produtos visualizados pelo usuário.

Requisitos Não Funcionais	
Desempenho e Escalabilidade	• O sistema suporta um volume inicial de acessos moderado. • O tempo de resposta das páginas não deve ultrapassar 3 segundos em condições normais de tráfego.
Segurança	• O sistema utiliza criptografia para armazenamento de senhas • O sistema segue a LGPD para proteção dos dados dos clientes.
Disponibilidade	• O sistema tem que ter um uptime mínimo de 99%. • O sistema conta com redundância básica para evitar falhas críticas.
Compatibilidade e Acessibilidade	• O sistema tem que ser responsivo e funcionar em dispositivos móveis e desktops. • O sistema tem que ser compatível com os navegadores mais populares (Chrome, Firefox, Edge, Safari).


EAP
Grupo 4 | August 20, 2024
E-commerce
1.1) Planejamento do Projeto
1.1.1) Definir Requisitos
1.1.3) Estabelecer Cronograma
1.1.4) Alocar Recursos
1.2) Design e Arquitetura
1.2.1) Definir Padrões de Design (MVC, etc.)
1.2.2) Elaborar Diagrama de Casos de Uso e Classes
1.3) Desenvolvimento
1.3.1) Desenvolver Frontend e Backend
1.3.2) Implementar Banco de Dados
1.3.3) Integrar APIs de Pagamento e Logística
1.4) Testes e Integração
1.4.1) Realizar Testes Unitários
1.4.2) Realizar Testes de Integração
1.5) Implementação e Lançamento
1.5.1) Preparar Ambiente de Produção
1.5.2) Migrar Dados
1.5.3) Lançar Plataforma
1.6) Suporte e Manutenção
1.6.1) Estabelecer Canal de Suporte ao Cliente
1.6.2) Monitorar Desempenho da Plataforma
1.6.3) Realizar Atualizações e Manutenções