INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATO DE EXPERIÊNCIA COMO ENGENHEIRO DE SOFTWARE NA EMPRESA CWI SOFTWARE DURANTE O DESENVOLVIMENTO DE SERVIÇO DE MEIOS DE PAGAMENTO

ALYSSON ALDRIN BARRETO BEZERRA

Cajazeiras
2025

ALYSSON ALDRIN BARRETO BEZERRA

RELATO DE EXPERIÊNCIA COMO ENGENHEIRO DE SOFTWARE NA EMPRESA CWI SOFTWARE DURANTE O DESENVOLVIMENTO DE SERVIÇO DE MEIOS DE PAGAMENTO

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Dr. Fabio Gomes de Andrade

Cajazeiras
2025

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

B574r Bezerra, Alysson Aldrin Barreto.

Relato de experiência como engenheiro de software na empresa CWI software durante o desenvolvimento de serviço de meios de pagamento / Alysson Aldrin Barreto Bezerra. – 2025.

23f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2025.

Orientador: Prof. Dr. Fabio Gomes de Andrade.

1. Engenharia de software. 2. Desenvolvimento de software. 3. Gestão de dados. 4. Rotina de mercado. 5. Relato de experiência. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ

CDU: 004.41:339.14(043.2)

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

ALYSSON ALDRIN BARRETO BEZERRA

RELATO DE EXPERIÊNCIA COMO ENGENHEIRO DE SOFTWARE NA EMPRESA CWI SOFTWARE DURANTE O DESENVOLVIMENTO DE SERVIÇO DE MEIOS DE PAGAMENTO

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Dr. Fabio Gomes de Andrade

Aprovada em: **08 de setembro de 2025.**

Prof. Dr. Fabio Gomes de Andrade - Orientador

Prof. Me. Diogo Dantas Moreira - Avaliador

IFPB - Campus Cajazeiras

Prof. Esp. João Igor Barros Rocha - Avaliador

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Francisco Paulo de Freitas Neto**, **COORDENADOR(A) DE CURSOS - FUC1 - CADS-CZ**, em 10/09/2025 15:59:00.
- **Fabio Gomes de Andrade**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/09/2025 16:57:59.
- **Joao Igor Barros Rocha**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 11/09/2025 08:48:50.
- **Diogo Dantas Moreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/09/2025 10:26:31.

Este documento foi emitido pelo SUAP em 10/09/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 764381
Verificador: 99d0dda5c1
Código de Autenticação:

AGRADECIMENTOS

À minha família, em especial a minha mãe, Veruscka, que, com muita garra, sempre fez o possível e o impossível para que eu pudesse manter meus focos nos estudos.

À minha irmã Maria Alyce, por auxiliar e alegrar o meu dia a dia com seu jeito amável e sua lealdade.

Às minhas cachorras, por trazerem o amor verdadeiro que só elas conseguem demonstrar.

Ao meu orientador, Fabio, pela disponibilidade e paciência para auxiliar no desenvolvimento do trabalho, apesar das noites mal dormidas, e ser um exemplo de profissional que eu espero conseguir ser.

Ao meu grupo de amigos que perpetuam desde o ensino médio em minha vida, que posso contar em todos os momentos, sejam eles bons ou ruins.

Ao Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – *Campus* Cajazeiras, pela hospitalidade e oportunidade de fazer esse trabalho.

RESUMO

Este trabalho apresenta um relato de experiência profissional vivenciado durante a minha atuação como engenheiro de software na empresa *CWI Software*, alocado no time de tecnologia do *Grupo Casas Bahia*, no desenvolvimento de um serviço de meios de pagamento. Esse projeto teve como objetivo centralizar e modernizar a gestão dos meios de pagamento do próprio *Grupo Casas Bahia*, integrando múltiplos canais de venda por meio de uma *Application Programming Interface* robusta e escalável. Para isso, foram aplicadas tecnologias e práticas modernas de desenvolvimento, como C#, ASP.NET Core, *Domain-Driven Design*, Onion Architecture, programação assíncrona, cache distribuído e testes automatizados. O trabalho destaca os principais desafios enfrentados, como a necessidade de alto desempenho e confiabilidade, e evidencia os ganhos obtidos com a adoção de uma arquitetura orientada à qualidade e à escalabilidade. A experiência me proporcionou um aprendizado significativo, unindo teoria e prática em um ambiente corporativo de grande porte, contribuindo para a minha formação técnica e profissional.

Palavras-chave: Desenvolvimento de Software, ASP.NET Core, Domain-Driven Design, Onion Architecture.

ABSTRACT

This work presents a professional experience report from the perspective of a software engineer at *CWI Software*, assigned to the technology team of *Grupo Casas Bahia*, during the development of a payment service. The project aimed to centralize and modernize payment management for *Grupo Casas Bahia* by integrating multiple sales channels through a robust and scalable Application Programming Interface. To achieve this, modern development technologies and practices were employed, including C#, ASP.NET Core, Domain-Driven Design, Onion Architecture, asynchronous programming, distributed caching, and automated testing. The report highlights the main challenges encountered, such as ensuring high performance and reliability, as well as the benefits obtained from adopting a quality- and scalability-driven architecture. This experience provided substantial learning by combining theory and practice within a large-scale corporate environment, contributing to technical and professional growth.

Keywords: Software Development, ASP.NET Core, Domain-Driven Design, Onion Architecture.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
DDD	*Domain-Driven Design*
GCB	Grupo Casas Bahia
HTTP	*Hypertext Transfer Protocol*
TCC	Trabalho de Conclusão de Curso
TI	Tecnologia da Informação
TPL	*Task Parallel Library*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, a transformação digital tem remodelado profundamente a relação entre empresas e consumidores, ampliando significativamente os canais de interação — como redes sociais, aplicativos móveis e mensageria instantânea — e tornando a jornada de compra mais dispersa e complexa. Apesar de trazer conveniência ao consumidor, essa multiplicidade de canais de comunicação apresenta desafios significativos para as organizações, especialmente no que se refere à gestão integrada de dados, rastreabilidade de interações e padronização de processos.

A literatura aponta que a qualidade da integração entre canais exerce um impacto positivo sobre a satisfação e a lealdade dos clientes, além de facilitar a eficiência operacional das organizações (GASPARIN; SLONGO, 2023). Nesse contexto, a estratégia *omnichannel* (integração de todos os canais de comunicação e venda de uma empresa) emergiu como uma abordagem essencial para integrar esses pontos de contato, proporcionando uma experiência de usuário fluida e coerente (LEMON; VERHOEF, 2016).

Uma prática estratégica adotada por organizações que buscam concentrar esforços em suas atividades principais, transferindo a responsabilidade por serviços de Tecnologia da Informação (TI) a fornecedores especializados é a terceirização de TI. De acordo com Albertin e Sanchez (2008), essa abordagem permite às empresas obterem ganhos de eficiência, flexibilidade e acesso a tecnologias mais avançadas, ao mesmo tempo em que reduz os seus custos operacionais. No entanto, os autores também ressaltam que essa terceirização exige uma gestão cuidadosa para mitigar riscos como a perda de controle sobre processos, dependência do fornecedor e dificuldades na comunicação entre equipes internas e terceirizadas. Uma governança sólida, com contratos bem estruturados e monitoramento contínuo, é essencial para garantir o alinhamento entre os objetivos da organização e os resultados entregues pela empresa terceirizada.

O presente trabalho tem como objetivo relatar a experiência adquirida durante o desenvolvimento de um novo serviço de meios de pagamento para o *Grupo Casas Bahia* (GCB). A atuação ocorreu por meio de terceirização, onde trabalhei como

colaborador da empresa *CWI Software*, alocado no time de tecnologia do GCB. Esse modelo de contratação — bastante comum no setor de tecnologia — permite que empresas ampliem sua capacidade de entrega técnica com agilidade, ao mesmo tempo em que profissionais têm a oportunidade de participar de projetos estratégicos em grandes organizações.

Durante o projeto, foram enfrentados diversos obstáculos relacionados à integração com sistemas legados, à adequação aos múltiplos canais de venda e à adoção de práticas modernas de desenvolvimento de software. O sistema proposto visa centralizar e modernizar a oferta de meios de pagamento, substituindo soluções anteriores baseadas em tecnologias obsoletas.

A adoção dessa nova solução contribui não apenas para a modernização da gestão de pagamentos, mas também para a eficiência operacional da empresa contratante, pois ela foi projetada para atender diferentes canais de venda, incluindo aplicações web e aplicativos móveis, centralizando a disponibilização de meios de pagamento em uma única *Application Programming Interface* (API).

Este trabalho detalha a minha participação no desenvolvimento da solução, destacando as tecnologias utilizadas, os desafios enfrentados e a arquitetura proposta para a API de pagamentos, além de refletir sobre a experiência profissional adquirida no modelo de terceirização em um ambiente corporativo de grande porte.

1.1 OBJETIVOS

Esta seção descreve os propósitos que orientaram a elaboração deste trabalho, os quais estão organizados em dois níveis: objetivo geral e objetivos específicos. O objetivo geral define a proposta principal deste trabalho, ressaltando sua contribuição para a transformação digital de serviços no contexto empresarial e tecnológico. Já os objetivos específicos detalham as etapas, estratégias e tecnologias envolvidas no desenvolvimento do serviço de meios de pagamento, desde seu desenvolvimento até sua validação prática, servindo como base para o relato da experiência vivenciada pelo autor durante sua atuação no projeto.

1.1.1 OBJETIVO GERAL

O objetivo deste Trabalho de Conclusão de Curso (TCC) é relatar a experiência adquirida durante o desenvolvimento de um serviço de meios de pagamento na empresa *CWI Software*, atuando como Desenvolvedor de Software para o GCB.

1.1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

- Contextualizar a necessidade de uma solução para meios de pagamento, destacando os desafios enfrentados pela comunicação descentralizada e os impactos da falta de integração entre os diferentes canais de venda (web e aplicativo);
- Apresentar as tecnologias utilizadas no desenvolvimento da solução, incluindo linguagens de programação, frameworks, padrões arquiteturais, bancos de dados e mecanismos de integração com APIs externas e sistemas legados;
- Documentar as decisões técnicas e estratégicas adotadas durante o desenvolvimento, evidenciando os principais desafios enfrentados e as soluções aplicadas para garantir a centralização das regras de negócio e a interoperabilidade entre sistemas diversos.

1.2 ORGANIZAÇÃO DO DOCUMENTO

O restante deste documento está estruturado em quatro capítulos, cada um destinado a abordar uma etapa fundamental para o desenvolvimento do tema.

O Capítulo 2 apresenta a empresa na qual a experiência foi realizada, contemplando sua trajetória, áreas de atuação e os principais aspectos que a caracterizam no mercado.

O Capítulo 3 reúne a fundamentação teórica, trazendo conceitos, tecnologias e metodologias que servem de base para compreender o contexto da experiência relatada.

Já o Capítulo 4 dedica-se à descrição prática da vivência profissional, evidenciando as atividades realizadas, os desafios enfrentados e as soluções implementadas ao longo do processo.

Por fim, o Capítulo 5 traz as considerações finais, discutindo os resultados alcançados, os aprendizados obtidos e o impacto da experiência no meu desenvolvimento pessoal e profissional.

2 APRESENTAÇÃO DA EMPRESA

A *CWI Software*[1] é uma empresa brasileira de tecnologia que atua há mais de trinta anos no desenvolvimento de soluções de software sob medida, consultoria especializada e alocação de profissionais altamente capacitados. Com sede no Rio Grande do Sul e presença nacional, a *CWI* atende empresas de grande porte dos setores de varejo, finanças, telecomunicações, entre outros, entregando produtos digitais escaláveis, seguros e alinhados com as boas práticas de engenharia de software. Por meio de times multidisciplinares e metodologias ágeis, a empresa participa ativamente da transformação digital de seus clientes, promovendo inovação contínua e resultados sustentáveis.

Já o GCB tem papel fundamental no varejo brasileiro, com histórico marcado pela democratização do consumo, especialmente entre as classes C, D e E. Atualmente, o grupo também administra marcas como *Ponto* (antigo *Pontofrio*) e a plataforma de marketplace *Extra*, figurando entre as maiores empresas do setor na América Latina. Nos últimos anos, o grupo passou por um intenso processo de transformação digital, com foco na modernização da sua infraestrutura tecnológica e na integração entre os seus canais físicos e digitais, adotando uma abordagem *omnichannel*. A empresa investe fortemente em inovação, com iniciativas como o banco digital *banQi*, o uso de inteligência artificial, a automação de tarefas e a utilização da arquitetura de microsserviços.

[1] https://cwi.com.br

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as principais tecnologias e conceitos aplicados no desenvolvimento do projeto, detalhando suas origens, características técnicas e aplicações práticas no contexto da arquitetura de software e desenvolvimento backend com a stack *.NET*.

3.1 LINGUAGEM C#

O *C#* (HEJLSBERG; WILTAMUTH; GOLDE, 2003) é a principal linguagem da plataforma *.NET*, um ambiente de desenvolvimento gratuito, multiplataforma e de código aberto. Trata-se de uma linguagem de propósito geral que possibilita a criação de aplicações para diversos dispositivos e contextos, abrangendo sistemas desktop, aplicativos para dispositivos móveis e servidores.

O *C#* é uma linguagem de programação orientada a objetos que, além de seguir os princípios desse paradigma, também incorpora recursos de outros estilos, como a programação funcional. Ele oferece funcionalidades que permitem trabalhar próximo ao hardware de forma controlada, garantindo desempenho sem abrir mão da segurança (MICROSOFT, 2025).

Por pertencer à família da linguagem de programação *C* (KERNIGHAN; RITCHIE, 1988), a sua sintaxe é parecida com as das linguagens *C* e *Java* (GOSLING, 2000), o que facilita a curva de aprendizado para quem já possui experiência com essas tecnologias. Integrado ao ecossistema *.NET* (WATKINS; HAMMOND; ABRAMS, 2003), o *C#* é amplamente utilizado para criar aplicações que vão desde sistemas corporativos a soluções em nuvem, sendo atualmente a principal linguagem da plataforma e uma das mais adotadas no mercado.

3.2 ASP.NET CORE

O *ASP.NET Core* (LOCK, 2023) é um framework de desenvolvimento web de alto desempenho, gratuito, de código aberto e multiplataforma, criado sobre a plataforma *.NET* para possibilitar a construção de aplicações modernas e escaláveis. Projetado para lidar com cargas de trabalho de qualquer porte, ele oferece robustez e flexibilidade, sendo especialmente indicado para aplicações empresariais.

Entre os seus principais recursos estão o pipeline *Hypertext Transfer Protocol* (HTTP) personalizável, o suporte nativo à injeção de dependência, a configuração baseada em ambientes e uma série de ferramentas avançadas para monitoramento da saúde da aplicação, como logging, rastreamento e métricas de execução. Além disso, o *ASP.NET Core* se integra de forma fluida com as principais IDEs atuais, como Visual Studio e Visual Studio Code, otimizando o desenvolvimento e a manutenção de aplicações (MICROSOFT, 2025).

3.3 DOMAIN-DRIVEN DESIGN

Domain-Driven Design (DDD) (EVANS, 2004) é uma abordagem de desenvolvimento que foca na modelagem do domínio do negócio para criar software alinhado com a realidade da organização. O DDD incentiva a colaboração entre especialistas do domínio e desenvolvedores, usando uma linguagem ubíqua para descrever conceitos, regras e processos. Essa metodologia promove a criação de modelos ricos, encapsulando regras de negócio complexas e facilitando a evolução do sistema.

Além disso, o DDD organiza o sistema em módulos claros e independentes, facilitando a manutenção, os testes e a evolução das funcionalidades. Essa abordagem contribui para soluções mais robustas e alinhadas aos objetivos estratégicos da organização, além de integrar de forma natural com arquiteturas modernas, garantindo que o núcleo do domínio permaneça isolado de detalhes tecnológicos externos.

3.4 ONION ARCHITECTURE

O padrão arquitetural conhecido como *Onion Architecture* propõe uma abordagem que prioriza a separação de responsabilidades em aplicações duradouras e com comportamento complexo. Diferentemente da arquitetura tradicional em camadas, na qual cada camada depende diretamente da anterior e, frequentemente, todas acessam os serviços comuns de infraestrutura — o que gera acoplamento indesejado. O padrão em “cebola” inverte essa lógica de dependências. Isso preserva o núcleo da aplicação, tornando-o independente de mudanças externas (PALERMO, 2008).

De acordo com esse padrão, o núcleo da aplicação contém exclusivamente o modelo de domínio, representando os conceitos fundamentais do negócio, enquanto as camadas externas definem as interfaces que ele utiliza. A implementação dessas interfaces ocorre nas camadas periféricas, como interface, infraestrutura e testes, garantindo que qualquer acoplamento ocorra apenas em direção ao centro. Esse projeto segue o *Princípio da Inversão de Dependência* (MARTIN, 2002), facilitando testes e manutenção, e proporcionando flexibilidade para adaptar tecnologias externas sem impactar o núcleo do sistema.

3.5 API

Uma *Application Programming Interface* (API) consiste em qualquer conjunto de operações e dados semanticamente relacionados, geralmente vinculados a um domínio específico. Componentes de software, módulos, bibliotecas e frameworks disponibilizam APIs que expõem suas funcionalidades a outros elementos do sistema. No cenário atual de desenvolvimento de software, as APIs assumem um papel central, visto que mesmo os programas mais simples dependem das interfaces fornecidas por bibliotecas e frameworks (AFONSO; CERQUEIRA; DE SOUZA, 2012).

A adoção de APIs traz diversas vantagens que impulsionam a sua ampla utilização. Elas promovem a reutilização de código, reduzindo esforço e tempo de desenvolvimento, e facilitam a integração entre sistemas heterogêneos, permitindo que diferentes aplicações se comuniquem de forma padronizada. Ademais, elas contribuem para a escalabilidade e modularidade das soluções, uma vez que as funcionalidades podem ser atualizadas ou substituídas sem impactar o restante do sistema. Por fim, o uso de APIs favorece a inovação, ao possibilitar que desenvolvedores criem novos serviços e funcionalidades a partir de recursos já existentes.

3.6 PROGRAMAÇÃO ASSÍNCRONA E PARALELISMO NO .NET

Para garantir robustez e performance na comunicação com serviços externos via endpoints HTTP, implementamos técnicas de paralelismo utilizando os recursos async/await e Task.WhenAll da biblioteca *Task Parallel Library* (TPL) (LEIJEN; SCHULTE; BURCKHARDT, 2009) do *.NET*. Essas abordagens são fundamentais

para otimizar operações de entrada e saída, como chamadas simultâneas a diferentes APIs, sem bloquear a execução do programa.

O modelo assíncrono baseado em tarefas (GORANOVA; KALCHEVA-YOVKOVA; PENKOV, 2015) possibilita uma aplicação iniciar diversas operações em paralelo e aguardar sua finalização de forma coletiva. Com o *Task.WhenAll* é possível executar várias tarefas ao mesmo tempo, eliminando a necessidade de aguardar cada uma individualmente. Essa estratégia resulta em maior desempenho e responsividade, sendo fundamental para aplicações que exigem alta disponibilidade e baixo tempo de resposta (MICROSOFT, 2025).

3.7 CACHE REDIS DISTRIBUÍDO

O *Redis* (CARLSON, 2013) é um banco de dados em memória de código aberto, amplamente utilizado para melhorar a performance de aplicações por meio de técnicas de cache. Ele armazena dados como strings, hashes, listas, conjuntos e conjuntos ordenados, permitindo o acesso a esses dados de uma forma extremamente rápida e eficiente. A sua arquitetura é simples e flexível, o que facilita a sua integração com diversas linguagens de programação, incluindo *C#, Python, JavaScript* e *Java*. Além disso, o Redis oferece suporte a operações atômicas, persistência opcional e replicação, o que o torna adequado para uma variedade de cenários, desde cache de sessões até filas de mensagens e contadores em tempo real (REDIS, 2025).

3.8 TESTES AUTOMATIZADOS COM XUNIT

O *xUnit.net* (MESZAROS, 2007) é um framework de código aberto para testes unitários em *C#* amplamente utilizado no ecossistema *.NET*. Ele permite criar, organizar e executar testes automatizados de forma eficiente, promovendo maior qualidade e confiabilidade do software. Inspirado em frameworks clássicos como *JUnit* (TAHCHIEV, 2010) e *NUnit* (HAMILTON, 2004), o *xUnit.net* apresenta uma arquitetura moderna, com suporte a extensões e integração com ferramentas como *Visual Studio* e a linha de comando do *.NET SDK.* Esse framework é essencial em projetos *ASP.NET Core* para validar o comportamento das unidades de código, permitindo refatorações seguras e acelerando o ciclo de entrega contínua (XUNIT.NET, 2025).

4 RELATO DE EXPERIÊNCIAS

O ingresso na empresa ocorreu por meio do programa *CWI Crescer*, iniciativa voltada à capacitação e inserção de novos talentos na área de tecnologia. O processo teve início com um desafio técnico focado em lógica de programação. Após essa etapa, fui selecionado para ingressar no Level 1, composto por aulas online e atividades práticas que abordaram fundamentos de programação, banco de dados e desenvolvimento web.

Concluído o Level 1, participei de uma entrevista com a equipe de recrutamento e instrutores, que teve como objetivo avaliar tanto o desempenho técnico quanto o alinhamento com os valores da empresa. Em seguida, com a aprovação, avancei para o Level 2, no qual tive contato direto com as tecnologias utilizadas no dia a dia dos projetos da empresa, como *React, .NET* e metodologias ágeis, além da realização de um trabalho final em equipe.

Ao término do programa, participei da formatura do Crescer e recebi o certificado de conclusão. Nesse momento, aconteceu a avaliação quanto ao meu desempenho, evolução técnica e postura profissional. Após essa avaliação, fui convidado a integrar oficialmente a empresa como estagiário, sendo alocado no grupo que desenvolvia um sistema para o GCB. Após cerca de quatro meses, fui promovido para desenvolvedor.

Apesar de atuar como profissional terceirizado, eu tive oportunidade de contribuir ativamente nas decisões do processo de desenvolvimento, oferecendo sugestões técnicas e opiniões estratégicas. Minha atuação não se limitou à execução de tarefas, sendo valorizada a participação no planejamento e na arquitetura do sistema, reforçando a importância de minha contribuição para o sucesso do projeto.

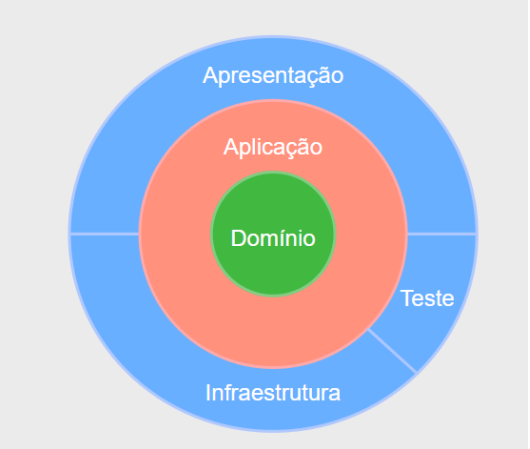
Nesse cliente desenvolvi projetos com a stack *.NET* marcados por desafios técnicos relevantes e um sólido aprendizado prático em arquitetura de software, testes automatizados e escalabilidade de aplicações distribuídas. Atuei como desenvolvedor backend utilizando principalmente *C#* com *ASP.NET*, aplicando princípios de DDD organizados através da *Onion Architecture*.

Um dos principais desafios enfrentados durante um dos projetos foi a substituição de um sistema legado que apresentava diversas limitações. Esse sistema possuía regras de negócio defasadas, rígidas e de difícil manutenção, além de operar quase exclusivamente de forma síncrona, o que comprometia significativamente a performance e a escalabilidade do serviço. Era necessário redesenhar a arquitetura de forma a centralizar as regras de negócio, permitir operações assíncronas e garantir interoperabilidade com múltiplos canais de venda e serviços externos facilitando futuras evoluções do sistema.

Para garantir uma arquitetura organizada e de fácil manutenção, adotou-se o ASP.NET Core, estruturado em camadas segundo os princípios da Onion Architecture. Com isso, foi usada a arquitetura mostrada na Figura 1. Essa arquitetura é composta pelas seguintes camadas:

- **domínio:** núcleo que encapsula as regras de negócio e os modelos da aplicação. Essa camada é totalmente independente de bibliotecas externas;
- **aplicação**: responsável por coordenar as interações entre o domínio e os recursos externos;
- **infraestrutura**: camada que implementa a comunicação com serviços externos e persistência de dados;
- **apresentação**: interface HTTP exposta aos consumidores. Essa camada permite que os consumidores utilizem as funcionalidades da aplicação por meio de uma API;
- **teste**: camada que valida as funcionalidades de todas as demais camadas, abrangendo testes unitários e de integração.

Figura 1 - Diagrama arquitetura do projeto

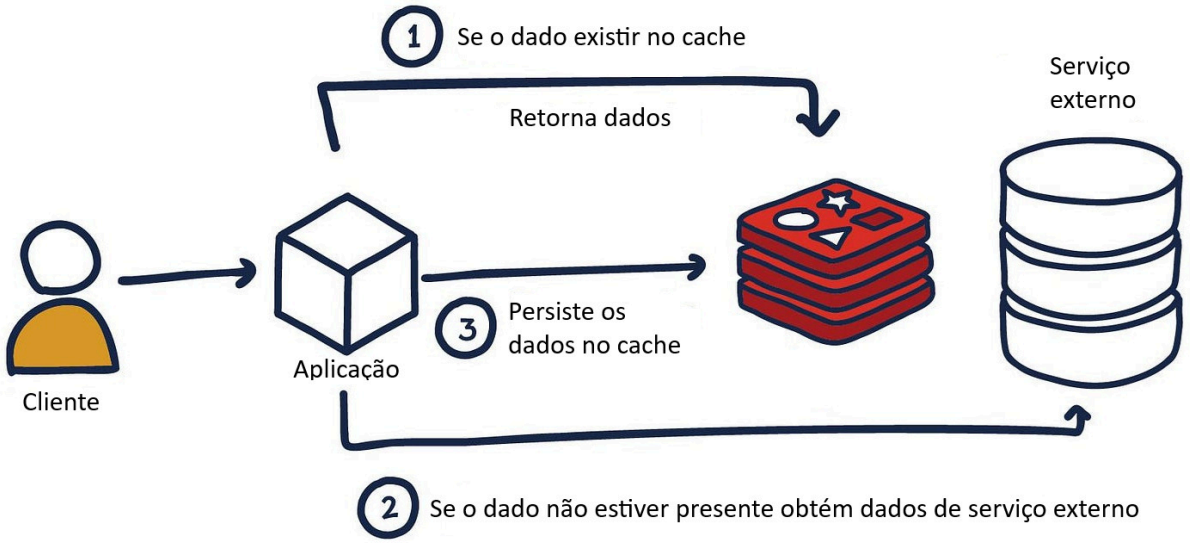


Fonte: Adaptado de Palermo.

A comunicação com plataformas externas continuou sendo realizada por meio de endpoints HTTP, porém, diferentemente do sistema legado, em que maior parte das operações eram síncronas e bloqueavam os recursos do servidor, comprometendo a performance, a solução implementada adotou estratégias assíncronas com async/await e Task.WhenAll. Essa abordagem permitiu executar múltiplas chamadas paralelas a diferentes serviços sem bloquear a execução, garantindo melhor utilização dos recursos do servidor, menor tempo de resposta e maior escalabilidade do sistema. Assim, o uso eficiente de endpoints HTTP passou a suportar um alto volume de requisições simultâneas, superando os gargalos presentes na arquitetura anterior.

Outro desafio foi o controle de dados temporários e de alta demanda, como informações relacionadas aos meios de pagamento. Para mitigar problemas de latência e sobrecarga, implementou-se um cache distribuído com Redis, armazenando dados críticos e reutilizáveis, reduzindo drasticamente o número de chamadas redundantes a serviços externos e otimizando a performance geral.

Figura 2 - Funcionamento de cache com Redis



Fonte: Adaptado de Nath, 2024.

Essa abordagem resultou em uma solução modular, escalável e resiliente, capaz de suportar cenários de alta disponibilidade e facilitar futuras evoluções do sistema.

Do ponto de vista de qualidade de código, a aplicação foi intensivamente coberta por testes automatizados com xUnit. Criamos testes unitários para as camadas de domínio e aplicação, com uso de mocks para simular dependências externas. A estrutura de testes foi fundamental para garantir a refatoração segura e acelerar as entregas em um ambiente de desenvolvimento contínuo.

5 CONCLUSÃO

A experiência adquirida durante o desenvolvimento do serviço de meios de pagamento durante a atuação na *CWI Software* representou um marco significativo para minha formação profissional. A participação em um projeto de grande porte, voltado para a orquestração de mensagens transacionais e integração de sistemas internos e de terceiros, permitiu consolidar conhecimentos teóricos e aplicar práticas modernas de desenvolvimento de software em um ambiente real.

O contato direto com tecnologias da stack *.NET*, incluindo *C#, ASP.NET Core*, DDD e *Onion Architecture*, proporcionou uma compreensão aprofundada sobre arquitetura de software escalável, separação de responsabilidades e construção de sistemas testáveis e de fácil manutenção. A aplicação prática de paralelismo assíncrono e do cache distribuído *Redis* evidenciou a importância de otimizar o desempenho e garantir a disponibilidade de informações críticas, aspectos essenciais para sistemas corporativos de alta complexidade.

Além dos conhecimentos técnicos, a experiência contribuiu significativamente para o desenvolvimento de habilidades estratégicas e comportamentais, como análise e resolução de problemas, trabalho em equipe, comunicação eficaz e adaptação a novas tecnologias. A execução de testes automatizados com *xUnit*, aliada ao uso de mocks, reforçou a prática de desenvolvimento orientado à qualidade, permitindo refatorações seguras e acelerando o ciclo de entrega contínua.

Em síntese, a vivência prática proporcionou um aprendizado abrangente, integrando teoria e prática, e destacou a relevância de processos estruturados, boas práticas de engenharia de software e metodologias ágeis para a entrega de soluções robustas e escaláveis. Este conhecimento adquirido constitui uma base sólida para futuras atuações profissionais, capacitando-me a enfrentar desafios complexos e a contribuir de forma efetiva em projetos tecnológicos estratégicos.

REFERÊNCIAS

AFONSO, Luiz; CERQUEIRA, Renato; DE SOUZA, Clarisse (2012). **Evaluating application programming interfaces as communication artefacts**. System, 100, 8-31.

ALBERTIN, Alberto; SANCHEZ, Otávio. (2008). **Outsourcing de TI**. FGV Editora.

CARLSON, Josiah. (2013). **Redis in Action.** Simon and Schuster.

EVANS, Eric. (2004). **Domain-driven design: tackling complexity in the heart of software**. Addison-Wesley Professional.

GASPARIN, Isadora; SLONGO, Luiz. (2023). **Omnichannel as a Consumer-Based Marketing Strategy**. Revista de Administração Contemporânea, v. 27, n. 4, e220327.

GORANOVA, Mariana; KALCHEVA-YOVKOVA, Elena; PENKOV, Stanimir. (2015). **Task-based Asynchronous Pattern with async and await.** International Scientific Conference Computer Science.

GOSLING, James. (2000). **The Java Language Specification.** Addison-Wesley Professional.

HAMILTON, Bill. (2004). **NUnit Pocket Reference: Up and Running with NUnit**. O'Reilly Media, Inc.

HEJLSBERG, Anders; WILTAMUTH, Scott; GOLDE, Peter. (2003). **C# Language Specification**. Addison-Wesley Longman Publishing Co., Inc.

KERNIGHAN, Brian W.; RITCHIE, Dennis M. (1988). **The C Programming Language.** Pearson Educación.

LEMON, Katherine; VERHOEF, Peter. (2016). **Understanding customer experience throughout the customer journey**. Journal of Marketing, v. 80, n. 6, p. 69–96.

LOCK, Andrew. (2023). **ASP.NET Core in Action**. Simon and Schuster.

MARTIN, Robert C. (2002). **Agile Software Development: Principles, Patterns, and Practices**. Upper Saddle River: Prentice Hall.

MESZAROS, Gerard. (2007). **xUnit Test Patterns: Refactoring Test Code**. Pearson Education.

MICROSOFT. **Programação assíncrona baseada em tarefas**. In: Microsoft Learn – Task-based Asynchronous Programming. 2025. Disponível em: https://learn.microsoft.com/pt-br/dotnet/standard/parallel-programming/task-based-asynchronous-programming. Acesso em: 20 ago. 2025.

MICROSOFT. **Um tour pela linguagem C#**. In: Microsoft Learn – Tour of C#. 2025. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/overview. Acesso em: 20 ago. 2025.

MICROSOFT. **Visão geral do ASP.NET Core**. In: Microsoft Learn. 2025. Disponível em: https://learn.microsoft.com/pt-br/aspnet/core/overview?view=aspnetcore-9.0. Acesso em: 20 ago. 2025.

NATH, Tushar. **Unlocking Performance: Leveraging Redis Caching in Node.js TypeScript Backend.** Medium, 8 fev. 2024. Disponível em: Medium – Unlocking Performance: Leveraging Redis Caching in Node.js TypeScript Backend. Acesso em: 27 ago. 2025.

PALERMO, Jeffrey. **The Onion Architecture: part 1**. Programming with Palermo, 29 jul. 2008. Disponível em: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1. Acesso em: 20 ago. 2025.

REDIS. **O que é o Redis?** In: Redis.io. 2025. Disponível em: https://redis.io/pt/learn/develop/node/nodecrashcourse/whatisredis. Acesso em: 20 ago. 2025.

TAHCHIEV, Petar; et al. (2010). **JUnit in Action.** Manning Publications Co.

WATKINS, Damien; HAMMOND, Mark J.; ABRAMS, Brad. (2003). **Programming in the .NET Environment.** Addison-Wesley Professional.

XUNIT.NET. **xUnit.net**. 2025. Disponível em: https://xunit.net/?tabs=cs. Acesso em: 20 ago. 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras - Código INEP: 25008978
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

Trabalho de conclusão de curso

Assunto:	Trabalho de conclusão de curso
Assinado por:	Alysson Aldrin
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Alysson Aldrin Barreto Bezerra**, **DISCENTE (202112010019) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 12/09/2025 02:35:09.

Este documento foi armazenado no SUAP em 12/09/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1606478
Código de Autenticação: 37007bf115