BÁRBARA MINITTI

Técnica de Aplicação da Arquitetura Hexagonal para Aplicação de Gestão de Pagamentos

São Paulo
2024

BÁRBARA MINITTI

Técnica de Aplicação da Arquitetura Hexagonal para Aplicação de Gestão de Pagamentos

Versão Original

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Engenharia de Software.

Área de Concentração: Engenharia de Software

Orientador: Prof. Alípio Ferro

São Paulo
2024

DEDICATÓRIA

Dedico este trabalho a Deus e aos Orixás, os alicerces que sustentam a minha vida. Em especial ao Caboclo Vigia das Matas, meu mentor espiritual que me orienta, me direciona e me ajuda nessa longa jornada que é a vida.

AGRADECIMENTOS

À Universidade de São Paulo – USP que permitiu a existência desse MBA.

À Escola Politécnica da Universidade de São Paulo – EPUSP que forneceu a oportunidade e grandes professores compartilhando todo seu conhecimento e experiência.

Ao PECE – Programa de Educação Continuada em Engenharia que possibilitou meu aprimoramento na área de Engenharia de Software.

Ao meu orientador Prof. Alípio Ferro por todo o conhecimento passado com grande clareza e objetividade, por toda paciência e parceria, juntamente com o apoio, e direcionamento na organização do trabalho.

Ao meu Prof. Dr. Jorge Luis Risco Becerra pelas aulas de Arquitetura de Software e apoio no desenvolvimento final do trabalho.

Aos meus pais por me incentivar e mostrar o valor dos estudos na vida de uma pessoa, bem como todo suporte em toda em minha jornada acadêmica.

RESUMO

MINITTI, B. **Técnica de Aplicação da Arquitetura Hexagonal para Aplicação de Gestão de Pagamentos**. 2024. 59 p. Monografia (MBA Engenharia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2024.

Esta monografia investiga a aplicação da Arquitetura Hexagonal, ou Arquitetura de Portas e Adaptadores, como solução para mitigar desafios relacionados à manutenibilidade e testabilidade em sistemas de software complexos. Utilizando uma Aplicação de Gestão de Pagamentos como estudo de caso, o trabalho apresenta uma técnica para aplicabilidade da Arquitetura Hexagonal, fundamentada nas diretrizes da norma ISO/IEC/IEEE 42010, em Análise de Correspondências entre duas arquiteturas, está inspirada na Análise de *Gaps* do framework TOGAF e Modelagem C4.

A técnica inclui etapas como: Entendimento da Arquitetura Atual através Diagrama de Contexto do Modelo C4, Diagrama da Arquitetura Atual, identificação de stakeholders e levantamento de preocupações arquiteturais.

A Análise de Correspondências entre Arquitetura Atual e Arquitetura Alvo mostra as necessidades e capacidades entre ambas arquiteturas.

Com os artefatos citados produzidos a Arquitetura Hexagonal é aplicada na Aplicação de Gestão de Pagamentos através de Diagrama de Componentes do Modelo C4, Diagrama de Sequência da UML e Proposta de Implementação da Arquitetura Alvo.

Essas etapas são integradas gerando um processo de Técnica para Aplicabilidade da Arquitetura Hexagonal ilustrado por um diagrama BPMN, garantindo uma abordagem estruturada e replicável.

A aplicação prática da Arquitetura Hexagonal evidenciou vantagens significativas, incluindo maior desacoplamento entre o núcleo da aplicação e suas dependências externas, facilitando alterações tecnológicas e promovendo uma testabilidade robusta por meio de testes isolados no núcleo. Os benefícios também incluíram a redução de débitos

técnicos, maior eficiência no desenvolvimento e manutenção, e uma organização modular que suporta evolução contínua.

Embora a complexidade inicial e a curva de aprendizado sejam desafios a serem considerados, a técnica demonstrou eficácia em fundamentar a aplicabilidade da Arquitetura Hexagonal com base em critérios técnicos e alinhados às necessidades do projeto. Assim, o trabalho conclui que a Arquitetura Hexagonal, quando aplicada com um processo estruturado como o proposto, é uma alternativa viável e robusta para sistemas que exigem alta flexibilidade, escalabilidade e qualidade.

Palavras-chaves: arquitetura hexagonal, porta e adaptadores, testabilidade, manutenibilidade.

ABSTRACT

MINITTI, B. **Technique for Applying the Hexagonal Architecture to Payment Management Applications.** 2024. 59 p. Monograph (MBA in Software Engineering). Continuing Education Program in Engineering at the Polytechnic School of the University of São Paulo. São Paulo. 2024.

This monograph investigates the application of the Hexagonal Architecture, or Port and Adapter Architecture, as a solution to mitigate challenges related to maintainability and testability in complex software systems. Using a Payment Management Application as a case study, the work presents a technique for the applicability of the Hexagonal Architecture, based on the guidelines of the ISO/IEC/IEEE 42010 standard, in Correspondence Analysis between two architectures, inspired by the Gap Analysis of the TOGAF framework and C4 Modeling.

The technique includes steps such as: Understanding the Current Architecture through the Context Diagram of the C4 Model, Current Architecture Diagram, identification of stakeholders and survey of architectural concerns.

The Correspondence Analysis between Current Architecture and Target Architecture shows the needs and capabilities between both architectures.

With the aforementioned artifacts produced, the Hexagonal Architecture is applied to the Payment Management Application through the C4 Model Component Diagram, UML Sequence Diagram and Target Architecture Implementation Proposal.

These steps are integrated, generating a Hexagonal Architecture Applicability Technique process illustrated by a BPMN diagram, ensuring a structured and replicable approach.

The practical application of the Hexagonal Architecture demonstrated significant advantages, including greater decoupling between the core of the application and its external dependencies, facilitating technological changes and promoting robust testability through isolated tests in the core. The benefits also included the reduction of technical debt, greater efficiency in development and maintenance, and a modular organization that supports continuous evolution.

Although the initial complexity and learning curve are challenges to be considered, the technique demonstrated effectiveness in substantiating the applicability of the Hexagonal Architecture based on technical criteria and aligned with the project needs. Thus, the work concludes that the Hexagonal Architecture, when applied with a structured process like the one proposed, is a viable and robust alternative for systems that require high flexibility, scalability and quality.

Keywords: hexagonal architecture, port and adapters, testability, maintainability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API - *Application Programming Interface*

BDD - *Behavior Driven Design*

BPMN - *Business Process Model and Notation*

DDD - *Domain Driven Design*

ECS - *Elastic Container Service*

HYPE - Moda

RFN - Requisitos Não Funcionais

TDD - *Test Driven Design*

SUMÁRIO

1. INTRODUÇÃO

Este capítulo apresenta as motivações, objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

1.1 Motivações

Construir softwares robustos tem sido o principal foco da área de engenharia de software ao longo dos anos.

A robustez de um software está ligada a diversos fatores como saber lidar com os erros e falhas de modo que o software não trave ou tenha algum comportamento indesejado, saber lidar com as entradas para garantir que dados inválidos ou maliciosos comprometam o funcionamento do sistema, a resiliência que está ligada a capacidade do software se recupera rapidamente de falhas e continuar operando num nível mesmo que reduzido de funcionalidades, escalabilidade garantido que mesmo sob carga ou condições variáveis ele se mantenha estável e a manutenibilidade que através de um design e uma arquitetura permita a facilidade de atualização.

Para um desenvolvimento consistente de software levando em consideração todos esses Requisitos Não Funcionais citados devemos nos preocupar com a qualidade do sistema ligado diretamente com testabilidade.

No ponto de vista de testes é preciso garantir a efetividade das manutenções em códigos e componentes.

No ponto de vista de manutenção é preciso uma estratégia para otimizar o tempo de disponibilização de códigos e componentes e evitar a repetição de erros e simplificar as correções.

Essa monografia apresenta a aplicabilidade de uma arquitetura para facilitar o desenvolvimento de software considerando os requisitos não funcionais denominados de, **manutenibilidade e testabilidade de softwares**.

Para solução desses dois RNFs podemos contar com os processos de Arquitetura de Software para definir a estrutura para um sistema em alto nível, incluindo os componentes principais, a relação entre eles e como eles interagem.

A Arquitetura de Software envolve decisões estratégicas que afetam diretamente a organização, escalabilidade, desempenho e a manutenção do sistema. Ela funciona como um “norte” para guiar o desenvolvimento e evolução do sistema ao longo do tempo.

Porém, essa área é ampla e conta com vários designes para desenvolvimento de sistemas, como Arquitetura de Microserviços, Arquitetura em Camadas, Arquitetura Hexagonal e Arquitetura Limpa, tornando a decisão de qual arquitetura utilizar um processo de difícil escolha.

No contexto deste trabalho a Arquitetura Hexagonal é o design escolhido para ser analisado quanto a sua aplicabilidade para desenvolvimento de software e não simplesmente basear a escolha de uma determinada arquitetura por ela estar em alta no mercado ou na *“hype”.*

Em desenvolvimento de software, a habilidade mais importante não é a capacidade de escrever código, mas a capacidade de fazer as escolhas certas. (FOWLER; MARTIN, 2002)

O principal objetivo da Arquitetura Hexagonal é resolver problemas de acoplamento entre código de aplicação e os detalhes de infraestrutura, como banco de dados, interfaces de usuários e serviços externos. A proposta dessa arquitetura é isolar o código de negócio das dependências externas, permitindo maior flexibilidade e facilidade de mudança ao longo do tempo.

1.2 Objetivo

O objetivo desta monografia é analisar a aplicabilidade da Arquitetura Hexagonal para o desenvolvimento de sistemas e propor uma técnica para que arquiteturas escolhidas sejam validadas antes mesmo do início do desenvolvimento.

Para o estudo será usado como referência uma Aplicação de Gestão de Pagamentos, destacando-se as vantagens, desafios e melhores práticas no contexto de desenvolvimento de software.

1.3 Justificativas

O uso indiscriminado de arquiteturas de software no processo de desenvolvimento de sistemas é crescente, muitas vezes a escolha de um design para a construção de um sistema é feito pelo simples motivo da arquitetura estar em alta no mercado ou para atrair talentos, pois os desenvolvedores terão oportunidades de trabalhar com tecnologia de ponta. Diante desses fatores, nem sempre a arquitetura ideal é escolhida para o desenvolvimento de um sistema tornando-o complexo e aumentando os problemas ao longo dos anos.

A escolha por uma arquitetura de software é uma decisão muito crítica para se basear apenas nos fatores justificados no parágrafo anterior.

Majumder e Zoitl (2023), propuseram no artigo *A Domain-Driven Design Oriented OPC UA Server Development Methodology for CPPS* a abordagem de separação de camadas do DDD para o desenvolvimento de um servidor OPC UA para CPPS, nesse contexto a Arquitetura Hexagonal pode ser eficiente considerando que os componentes de software podem ser desenvolvidos independente de qualquer tecnologia específica e abrangem um grande número de casos de uso.

Jackson e Coutinho (2023), realizaram um estudo de caso no artigo *Restructuring the Software Architecture: A Case Study of the CoolBiz Core Banking Platform* e concluíram a importância de adotar abordagens holísticas ao discutir a arquitetura de software, além de questões puramente técnicas e incluindo fatores como atração e retenção de talentos. Alegando que esse aspecto é crucial na academia, devido ao impacto no treinamento profissional e no avanço da disciplina.

O estudo dessa monografia propõe demonstrar através da Arquitetura Hexagonal que a escolha por uma arquitetura pode ser uma utopia para o desenvolvimento de software se escolhida aleatoriamente tornando-o mais complexo e com maiores problemas do que aqueles que a arquitetura propõe-se a resolver.

1.4 Método de Pesquisa

O método de pesquisa adotado para essa monografia é o *Exploratório*.

Inicia-se através de pesquisas em artigos e bibliografias relevantes na literatura sobre a Arquitetura Hexagonal também conhecida por Arquitetura de Portas e Adaptadores idealizada por Alistair Cockburn em meados dos anos 90 sendo postado o primeiro artigo na WikiWikiWeb onde os artigos tinham como principal tema assuntos relacionados a Engenharia de Software.

Serão analisados os *trade-offs* de Arquitetura Hexagonal visando encontrar o cenário adequado para o uso dessa arquitetura em específico.

A técnica utilizada para análise de aplicabilidade será baseada nas diretrizes do Metamodelo de Arquitetura ISO/IEC/IEEE 42010 para levantamento dos elementos arquiteturais da Arquitetura Atual e Arquitetura Hexagonal (Arquitetura Referência) e através desse levantamento e com base na literatura da Arquitetura Hexagonal identificar se de fato ela resolve os problemas de manutenibilidade e testabilidade que essa monografia propõe-se a resolver. Será utilizado uma Análise de Correspondências para comparar e identificar as correspondências entre ambas arquiteturas.

Ao final, esse estudo poderá ser um guia de referência para ajudar na análise de aplicabilidade de outras arquiteturas mais adequadas para o desenvolvimento de software.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA revisa as motivações do conceito da Arquitetura Hexagonal e identifica a aplicabilidade em desenvolvimento de sistemas.

O Capítulo 3 DESENVOLVIMENTO apresenta o contexto e estudo de onde a Arquitetura Hexagonal poderá ser aplicada com eficácia na área de Engenharia de Software através de uma Aplicação de Gestão de Pagamentos com um *case* prático.

O Capítulo 4 ANÁLISE DE RESULTADOS compreende os *trade-offs* no contexto de desenvolvimento para Aplicação de Gestão de Pagamentos.

O Capítulo 5 CONSIDERAÇÕES FINAIS conclui a aplicabilidade da Arquitetura Hexagonal no âmbito do desenvolvimento da Aplicação de Gestão de Pagamentos, contribuindo e definindo uma técnica para a aplicabilidade de uma arquitetura de software na engenharia de sistemas.

2. REVISÃO BIBLIOGRÁFICA

2.1 Fatores Motivacionais do Conceito da Arquitetura de Software Hexagonal

Idealizada por Alistair Cockburn em 1988 teve o primeiro estímulo para a idealização da arquitetura.

Alistair implementou Model-View-Controller no seu protótipo Smalltalk, mas o seu programador C não implementou e quando surgiu a necessidade de mudar as fontes de entradas do programa C o programa precisou ser reescrito. (COCKBURN, ALISTAIR, 2024)

Em 1994 outra “dor”, projetistas de infraestrutura precisavam alterar um projeto de preço e tempo fixo envolvendo um mapeador de objetos relacionais para otimizar o desempenho. A modificação necessária era alterar o design da aplicação para banco de dados SQL.

Para realizar a modificação, os programadores precisaram desligar a aplicação por várias semanas para reescrever o seu mapeador e fazer a substituição de um banco de dados de teste na memória para o banco de dados SQL. (COCKBURN, ALISTAIR, 2024)

Nos anos 2000 o problema de acoplamento excessivo de arquiteturas monolíticas era latente, Alistair Cockburn propôs então o conceito da Arquitetura Hexagonal

conhecida também como Arquitetura de Portas e Adaptadores que tinha como principal objetivo isolar o código de negócio das dependências externas.

A principal ideia era separar o núcleo da aplicação (domínio) através de portas (interfaces) e adaptadores do mundo externo (interfaces de usuários, banco de dados, serviço de terceiros, etc).

Dessa maneira o núcleo do negócio mantinha-se “protegido” das alterações externas que tendem a evoluir mais rapidamente.

Os desafios para decisão de uma arquitetura coesa sempre existiram independente da epoca e tecnologia.

2.2 Definição Arquitetura Hexagonal

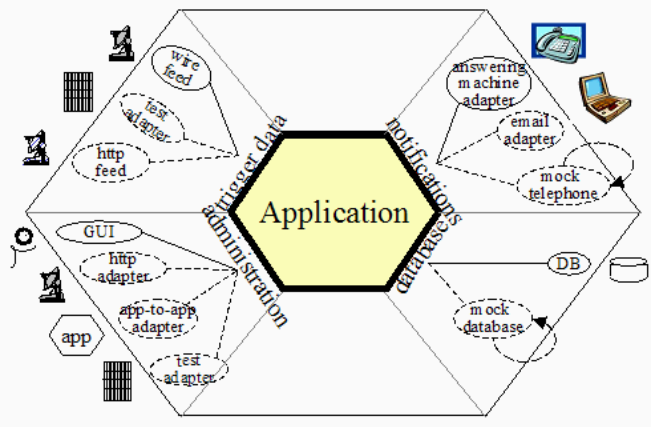
Alistair Cockburn usou a figura geométrica do hexágono para representar que o sistema deveria ser acessível e testável a partir de qualquer lado.

A figura geométrica do hexágono e o número seis não tem nenhum significado particular, é apenas uma figura que permite que as pessoas ao desenhar uma arquitetura tenham mais espaço para incluir portas e adaptadores.

Na prática pode existir três, cinco ou mais portas e não necessariamente seis como o hexágono nos induz a pensar, por esse motivo Alistair em 2005 no artigo *The Pattern: Ports and Adapters (‘’Object Structural”)* explica o motivo pelo qual Portas e Adaptadores é o nome mais adequado para o padrão.

“O termo “porta e adaptadores” pega os "propósitos" das partes do desenho. Uma porta identifica uma conversação proposital. Normalmente, haverá vários adaptadores para qualquer porta, para várias tecnologias que podem se conectar a essa porta. Normalmente, eles podem incluir uma secretária eletrônica, uma voz humana, um telefone touch-tone, uma interface gráfica humana, um test harness, um driver de lote, uma interface http, uma interface direta de programa para programa, um banco de dados simulado (na memória), um banco de dados real (talvez bancos de dados diferentes para desenvolvimento, teste e uso real).” Cockburn (2005).

Figura 1 - Exemplo Complexo de Arquitetura Hexagonal



Fonte: (Cockburn, 2005)

2.3 Elementos do Padrão Arquitetura Hexagonal

O padrão possui quatro elementos básicos e um quinto considerado não oficial, porém necessário.

Os quatro elementos considerados oficiais são:

1. Aplicação ou Sistema;
2. Portas;
3. Atores externos de condução e atores acionados;
4. Adaptadores para Portas.

O quinto elemento não oficial citado é o Configurador que será explicado no tópico 2.3.5.

2.3.1 Aplicação ou Sistema

Aplicação ou Sistema, conhecido também como núcleo ou domínio contém toda a lógica de negócios, independente de qualquer tecnologia externa. Consideramos como tecnologia externa um elemento físico, como um banco de dados, um usuário humano ou um link de rede.

Podemos ainda definir como tecnologia externa situações em que o “mundo exterior” é delimitado por uma fronteira social, ou seja, onde termina o escopo de uma equipe numa determinada parte da aplicação. Nesse caso pode ser criado limites técnicos através de interfaces e testes para proteger esse limite.

Através dessa interface é possível garantir que a aplicação seja chamada de acordo com o “contrato” estabelecido. Essa característica a torna um componente poderoso, possibilitando que ela possa ser conectada a qualquer outra aplicação.

Existe situações em que a aplicação não tem a necessidade de ser usada num contexto diferente, mas Alistair Cockburn afirma no livro *Hexagonal Architecture Explained* que através desse conceito é possível proteger a aplicação contra vazamento de lógica de negócios, alterar tecnologias externas e garantir testes.

2.3.2 Portas

As Portas definem o limite da aplicação. Cada interação entre a aplicação e o mundo exterior acontece através de uma interface de porta estabelecida pela própria aplicação. Portanto a porta demarca o que está dentro da aplicação e o que está externa a ela.

Essa característica de Portas atribui o “poder” ao padrão.

Usualmente existem três tipos de Portas gerais acionadas:

- Portas para obter informações de um repositório;
- Portas para notificar alguém;
- Portas para controlar algum dispositivo.

2.3.3 Atores Externos de Condução e Atores Acionados

Os Atores externos podem ser definidos em atores primários e secundários.

Ator primário é qualquer entidade humana ou eletrônica que coloca a aplicação em ação. Ele solicita um serviço da aplicação iniciando um conjunto complexo de interações de ida e volta.

A aplicação não precisa conhecer o ator primário, o objeto de chamada precisa saber a função na aplicação para acionar o serviço necessário

Ator secundário é qualquer entidade humana ou eletrônica que a aplicação entra em ação, solicitando um serviço.

Esse conceito representa a ideia de Portas, pois os atores primários tornam-se condutores e atores secundários tornam-se atores acionados que interagem com as portas primárias ou secundárias.

Para definir a quantidade de Portas numa aplicação podemos usar a técnica de casos de uso.

Um ator interage com um sistema, isso corresponde a uma porta ou uma interface que perguntamos a qual ator ela serve.

Durante a construção do sistema a arquitetura inicial pode mudar, porém essa técnica funciona como um bom ponto de partida.

2.3.4 Adaptadores para Portas

Existe a possibilidade de um ator usar diretamente a interface fornecida ou necessária, nesses casos não é necessário nenhum adaptador.

Para explicar esse “comportamento”, podemos usar casos de testes como exemplo.

Ao desenvolver uma aplicação casos de testes são escritos para validações. Essas validações parte de uma interação com a aplicação, ou seja, pode-se codificar a interface fornecida pela aplicação diretamente nos casos de testes.

É possível criar duas aplicações para trabalharem em conjunto desde o início. Uma aplicação de portas e adaptadores independentes e uma para interfaces que deverá ser desenvolvida de modo que elas correspondam.

Em casos como esses não é necessário a criação de adaptadores, pois os atores externos já atendem as interfaces fornecidas e necessárias.

Em situações em que o ator externo não corresponde à interface da aplicação, é necessário escrever código para transformar a interface de um ator na do outro. Esse código é um adaptador.

Geralmente adaptadores são necessários para qualquer tecnologia do mundo real, exemplo, internet, banco de dados, feed em tempo real, ou até mesmo um ser humano como já mencionado.

Existem adaptadores fora da aplicação, mas quem decide e organiza os adaptadores que são internos ou externos a aplicação são os engenheiros de sistemas. A Arquitetura de Portas e Adaptadores não estabelece uma regra de como se deve organizar a aplicação, ela indica que há um “mundo” interior e exterior, o limite entre os dois é definido pelas portas fornecidas e necessárias.

Diante desses fatores surgem muitas dúvidas de como organizar os adaptadores de uma aplicação.

Uma estrutura mal organizada pode inviabilizar o uso da Arquitetura de Portas e Adaptadores.

2.3.5 Configurador *(o quinto elemento não oficial)*

É o elemento que o padrão não estabelece como necessário, porém “alguma coisa” precisa conectar todas as peças. Alguém precisa direcionar como a aplicação precisa ser acessada e direcionar a aplicação a quais atores devem ser utilizados. Essa função é de responsabilidade do Configurador.

Para executar essa função o Configurador precisa conhecer todos os elementos. Atores de Condução ou seus Adaptadores e os Atores de Acionamento ou seus Adaptadores.

O padrão não especifica a forma como Configurador deve ser desenvolvido, há várias maneiras de desenvolvê-lo e tudo depende da situação.

Independente da organização, algumas ações devem ocorrer:

- Instanciar cada interagente acionado. (um ator acionado ou um ator que não precisa de adaptador);

- Instanciar a aplicação;
- Enviar para aplicação os interagentes acionados;
- Instanciar cada interagente de condução (ator ou adaptador) e enviar para a aplicação para ser utilizado.

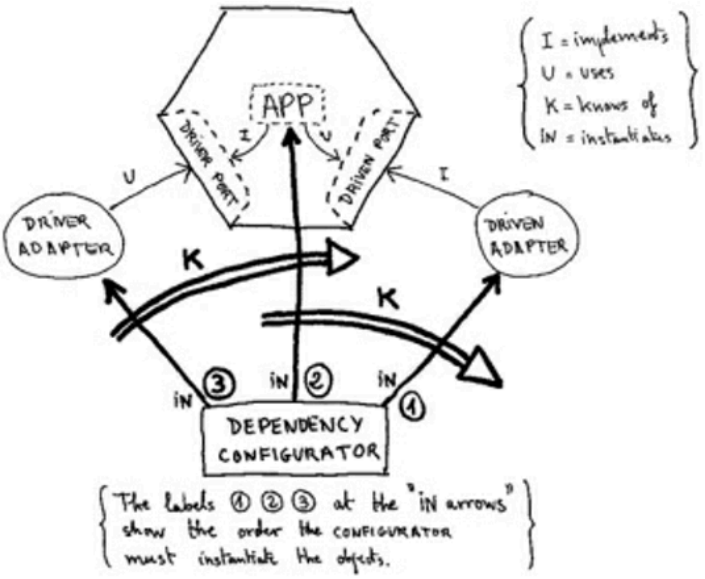
Existem três maneiras para a aplicação “conhecer” o ator acionado.

Umas delas é enviado ao construtor da aplicação o ator acionado, uma outra maneira é a aplicação disponibilizar uma interface com uma função que define o ator acionado, então um ator condutor chama essa função para definir o ator acionado a qualquer momento.

Por fim, a aplicação pode usar um localizador de serviço e perguntar ao localizador qual ator acionado usar.

Para os dois primeiros casos citados, podemos utilizar uma estrutura de injeção de dependência, como por exemplo, o Spring.

Figura 2 - Configurador introduz atores



Fonte: Cockburn, Alistair; Garrido de Paz, Juan Manuel. Hexagonal Architecture Explained (p. 31). Humans and Technology Inc. Edição do Kindle

2.4 Requisitos e Recomendações do Padrão

Para utilizar um padrão é necessário saber quais são os requisitos para aplicar o padrão desejado.

Seguir o que o padrão recomenda e saber o que está fora do seu escopo pode ser muito útil ao longo do desenvolvimento da aplicação, pode contribuir diretamente na complexidade do desenvolvimento.

Requisitos do padrão:

- Definir uma interface para todas interações externas;
- Definir portas de condução para as interfaces fornecidas e portas de condução para as interfaces acionadas;
- Permitir que atores acionados sejam configurados em tempo de execução;
- Não ter dependência entre o código-fonte e seus atores primários e secundários;
- Atores externos podem interagir somente com portas definidas, eles não tem permissão para interagir com o interior do hexágono;
- As portas e interfaces usam termos para expressar apenas as necessidades de negócios e devem ser neutras a tecnologias;

Recomendações do padrão:

- O padrão não determina como deve ser a nomenclatura das portas, porém sugere que seja “para fazer algo”, pois ajuda a comunicar, porque as interfaces são agrupadas;
- O padrão não define a granularidade das portas ou quantas interfaces de função são agrupadas numa porta. O recomendável é ter menos a mais, iniciando com uma porta por ator primário e uma para o secundário, pois corresponde às intenções de conversas.
- Não diz nada referente como organizar o seu código, porém recomenda boas práticas para facilitar a interpretação das pessoas e a manutenção da arquitetura.
    - Criar dois diretórios de portas, um para as portas de condução e outro para as portas acionadas. Colocar os diretórios no diretório da aplicação, pois eles pertencem a aplicação.
    - Criar dois diretórios de adaptadores, um para adaptadores de condução e outro para adaptadores acionados. Coloque-os em diretórios diferentes da aplicação.

- O padrão não menciona nada como organizar e proteger os adaptadores e nem cita se eles devem interagir ou não. Com exceção dos casos de teste, os adaptadores de condução não interagem diretamente com os adaptadores acionados.

- Não exclui a possibilidade de ter um subsistema de Portas e Adaptadores dentro de algum sistema maior de Portas e Adaptadores. Considerando Portas e Adaptadores como um componente, isso implica que o componente de Portas e Adaptadores será configurável para diferentes atores secundários e será testado isoladamente do restante do sistema maior de Portas e Adaptadores. É improvável, porém não impossível e por esse motivo é dito que o padrão não aninha.

2.5 Fora do Escopo do Padrão

O padrão não cita como deve ser a estrutura da aplicação internamente, por isso pode ser adotado a estrutura que melhor convém, como por exemplo, Domain Driven Design (DDD), Grande Bola de Lama ou até mesmo separar a função do modelo.

O padrão não recomenda e nem restringe a reestruturação interna da aplicação, essa característica o torna diferente das demais arquiteturas.

2.6 Abordagem de Teste do Padrão

Um dos principais benefícios oferecidos pelo padrão é a flexibilidade em relação aos testes.

A característica simétrica do padrão permite que os testes sejam desenvolvidos ao longo do desenvolvimento do software, não sendo necessário aguardar a finalização do software para desenvolvimento dos testes.

A simetria do padrão refere-se ao fato de que a arquitetura não faz distinção entre os lados de entrada (como interfaces de usuário e APIs) e os de saída (como bancos de dados e serviços externos). Ambos os tipos de interações – seja para receber dados ou para enviar e armazenar – são tratados da mesma maneira por meio das portas e adaptadores.

Essa abordagem simétrica permite que qualquer comunicação com o domínio seja realizada através de portas, independentemente da direção, promovendo um desacoplamento total entre a lógica de negócio e as dependências externas. Isso contribui para a flexibilidade e facilidade de teste, pois os adaptadores podem ser substituídos ou modificados sem afetar o núcleo da aplicação.

Diante desse cenário os diferentes componentes do sistema podem ser testados isoladamente durante o desenvolvimento e futuramente o tradicional teste integrado pode e deve ser realizado, porém com vários *bugs* já mitigados pelos testes unitários.

Em suma, a lógica de negócios pode ser testada em memória separadamente dos componentes de banco de dados e APIs.

Todos esses pontos faz-se concluir que o padrão facilita o uso de metodologias direcionadas a testes como TDD (Test Driven Development) e BDD (Behavior Driven Development), entretanto, vale ressaltar que o uso dessas metodologias não são obrigatórias, pode-se concluir o desenvolvimento do software e posteriormente testá-lo como o costume do time de engenheiros.

Do ponto de vista de testes o que fica claro é que o padrão é uma arquitetura bem direcionada a testes unitários e por consequências a viabilidade de testes automatizados.

2.7 Comparação Arquitetura Hexagonal vs Demais Arquiteturas e DDD (Domain Driven Design)

No livro *Hexagonal Architecture Explained* Alistair CockBurn faz uma breve comparação entre Arquitetura em Camadas, Arquitetura Limpa e Arquitetura “Cebola” para elucidar a Portas e Adaptadores.

Portas e Adaptadores tem apenas duas camadas, dentro da aplicação (domínio) e fora da aplicação (qualquer coisa). O requisito principal é que você organize os atores externos para se conectar em portas específicas.

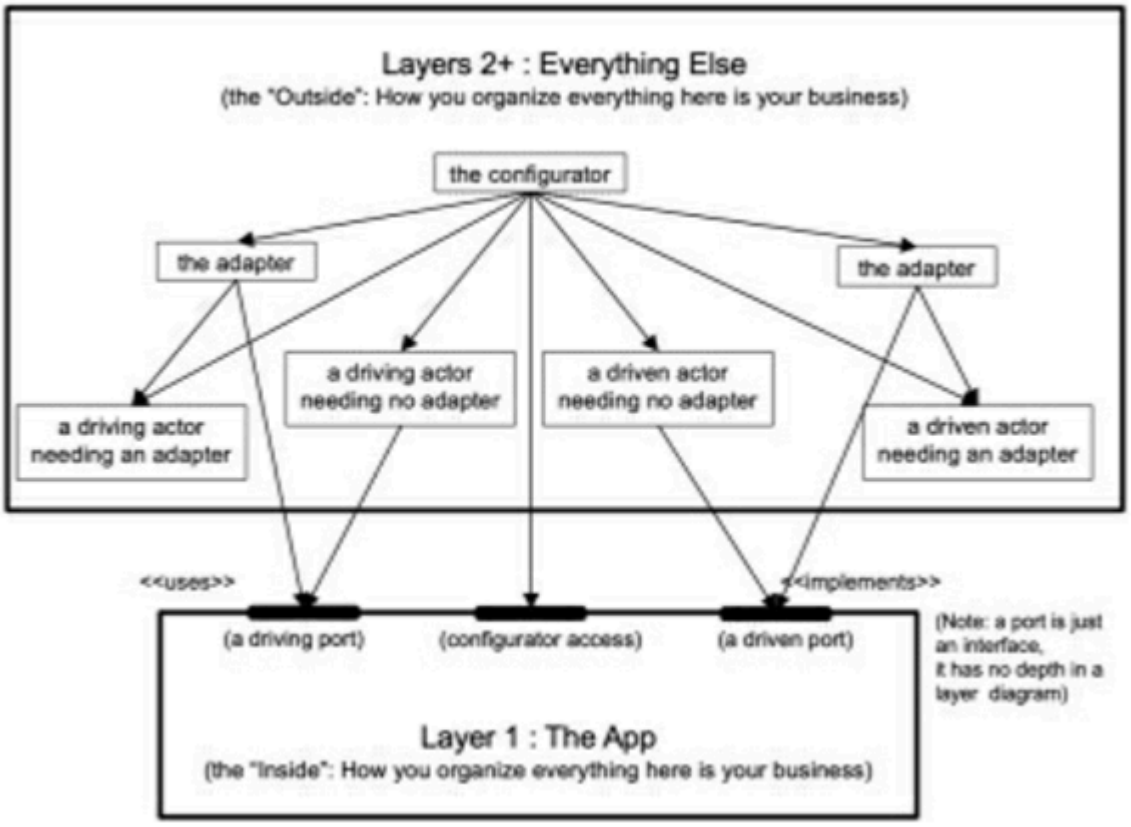
Arquitetura em Camadas requer que o código seja separado por preocupações e os organize em níveis, de modo que o nível superior chame ou dependa dos níveis

inferiores. Preocupações abstratas são colocadas no nível superior da arquitetura e os itens de infraestrutura são colocados na parte inferior da aplicação. Os componentes das camadas superiores dependem das camadas inferiores, porém o inverso não é uma verdade.

A Arquitetura em Camadas coloca a aplicação abaixo da interface do usuário e da infraestrutura.

A Arquitetura Limpa, “Cebola” e de Portas e Adaptadores são semelhantes, o objetivo delas é promover a independência do domínio, inclusive o diagrama dessas arquiteturas aparecem invertidos em comparação com os diagramas tradicionais de arquitetura em camadas.

Figura 3 - Portas e Adaptadores, especificado apenas com duas camadas “dentro” e “fora”

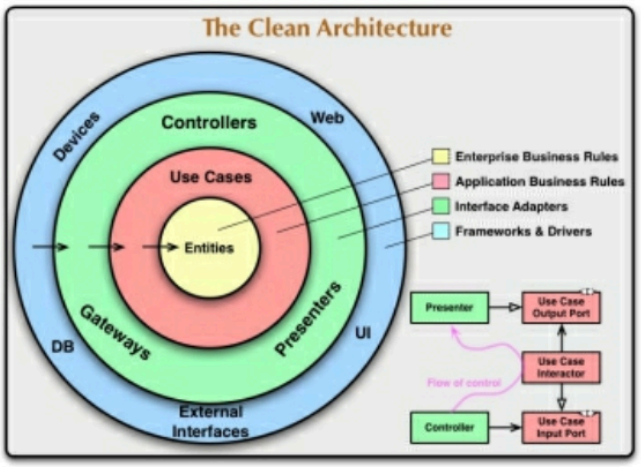


Fonte: Cockburn, Alistair; Garrido de Paz, Juan Manuel. Hexagonal Architecture Explained (p. 97). Humans and Technology Inc. Edição do Kindle.

Em suma, a Arquitetura Hexagonal se diferencia da Arquitetura Limpa e Arquitetura “Cebola” pela forma como se organiza, que são portas e adaptadores.

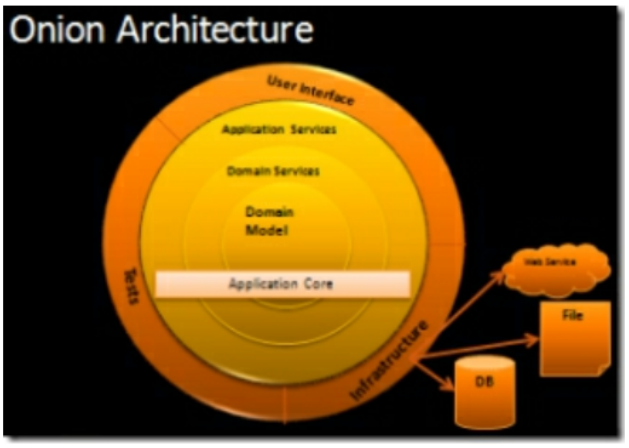
As arquiteturas Limpa e “Cebola” se organizam de forma concêntricas, as camadas externas ao domínio dependem da lógica de negócio, que está no núcleo e não conhecem as camadas externas, ambas são eficientes para desacoplar o domínio de dependências externas, mas a Arquitetura Limpa oferece uma estrutura mais complexa e detalhada, enquanto a Arquitetura em Cebola foca diretamente na proteção do domínio com uma abordagem mais enxuta.

Figura 4 - Arquitetura Limpa



Fonte: Cockburn, Alistair; Garrido de Paz, Juan Manuel. Hexagonal Architecture Explained (p. 98). Humans and Technology Inc. Edição do Kindle.

Figura 5 - Arquitetura “Cebola”



Fonte: Cockburn, Alistair; Garrido de Paz, Juan Manuel. Hexagonal Architecture Explained (p. 98). Humans and Technology Inc. Edição do Kindle.

Analisando e comparando todos esses pontos é fácil ser induzido ao pensamento de que Portas e Adaptadores e DDD (Domain Driven Design) são padrões arquitetônicos relacionados, no entanto não são.

Alistair Cockburn discute no livro *Hexagonal Architecture Explained* como os dois padrões arquitetônicos se complementam ao buscar a construção de sistemas centrados no domínio desacoplados e flexíveis.

No DDD (Domain Driven Design), o objetivo é refletir no código a linguagem e os processos da área de domínio, facilitando a colaboração com especialistas do negócio e criando um sistema que evolua com as necessidades de negócio.

À semelhança do foco do DDD (Domain Driven Design) com o objetivo da Arquitetura Hexagonal é proeminente, ambos têm como preocupação principal a proteção do domínio da aplicação, onde a lógica de negócio é completamente isolada das dependências externas. Isso permite que o sistema evolua de forma alinhada com as necessidades do negócio, mantendo alta manutenibilidade e testabilidade.

2.8 *Trade-Offs* Arquitetura Hexagonal

Como benefícios da arquitetura podemos dizer que:

1.) A arquitetura aumenta significativamente a testabilidade da aplicação pelos seguintes motivos:
    a.) A lógica de negócios pode ser testada rapidamente em memória;
    b.) Os diferentes componentes do sistema podem ser testados separadamente e posteriormente integrados;
    c.) Ao alterar um código da aplicação por qualquer motivo, seja uma nova feature, correção de bugs, etc. Os testes já existentes podem ser usados como teste de regressão. Isso permite verificar se a alteração feita introduz um comportamento não desejado na funcionalidade já existente, bem como verificar qualquer vazamento de lógica entre a aplicação e o mundo exterior;
2.) Metodologias como TDD (Test Driven Development) e BDD (Behavior Driven Development) são fáceis de serem aplicadas nessa arquitetura;

3.) Portas e Adaptadores aumenta a manutenibilidade da aplicação, fornecendo uma separação de preocupações e desacoplamento da lógica de negócios, o que facilita a localização do código que se deseja modificar;
4.) A manutenibilidade de aplicação é um RNF que está ligado diretamente com a redução de débitos técnicos, pois quanto maior a capacidade de manutenção de um sistema, os débitos técnicos tendem a ser menores pela facilidade de alteração, logo a Arquitetura Hexagonal é um grande aliado em redução de débitos técnicos;
5.) Portas e Adaptadores facilita a evolução e adição de novas tecnologias;
6.) A lógica de negócios pode ser desenvolvida sem necessidade de saber quais tecnologias serão utilizadas, evitando o atraso de decisões tecnologias.

Como custos temos:

1.) A estrutura, e processo de desenvolvimento da aplicação são mais complexos do que em outras arquiteturas, como por exemplo a arquitetura clássica de três camadas;
2.) Portas e Adaptadores introduz mais um nível de indireção do lado acionado, uma vez que os adaptadores devem traduzir as interfaces necessárias em interfaces específicas das diferentes tecnologias.
3.) Pode haver a necessidade de adição de mapeadores que mapeiam entidades do modelo do domínio em entidades do modelo de persistência.
4.) Aprender essa arquitetura não é fácil, requer experiência e pode ser difícil para programadores iniciantes;
5.) O início de um novo projeto leva mais tempo em relação às demais arquiteturas, portanto Portas e Adaptadores é indicado para projetos de médio a grande porte.

2.9 Considerações do Capítulo

Este capítulo teve como objetivo estudar os conceitos da Arquitetura Hexagonal e identificar a aplicabilidade em desenvolvimento de sistemas.

A introdução a essa arquitetura se faz necessária como parte do processo de estudo para análise de aplicabilidade ou não desta arquitetura em desenvolvimento de sistemas.

Este capítulo servirá como insumos para o próximo capítulo que focará no procedimento da técnica para aplicabilidade propriamente dito.

3. DESENVOLVIMENTO

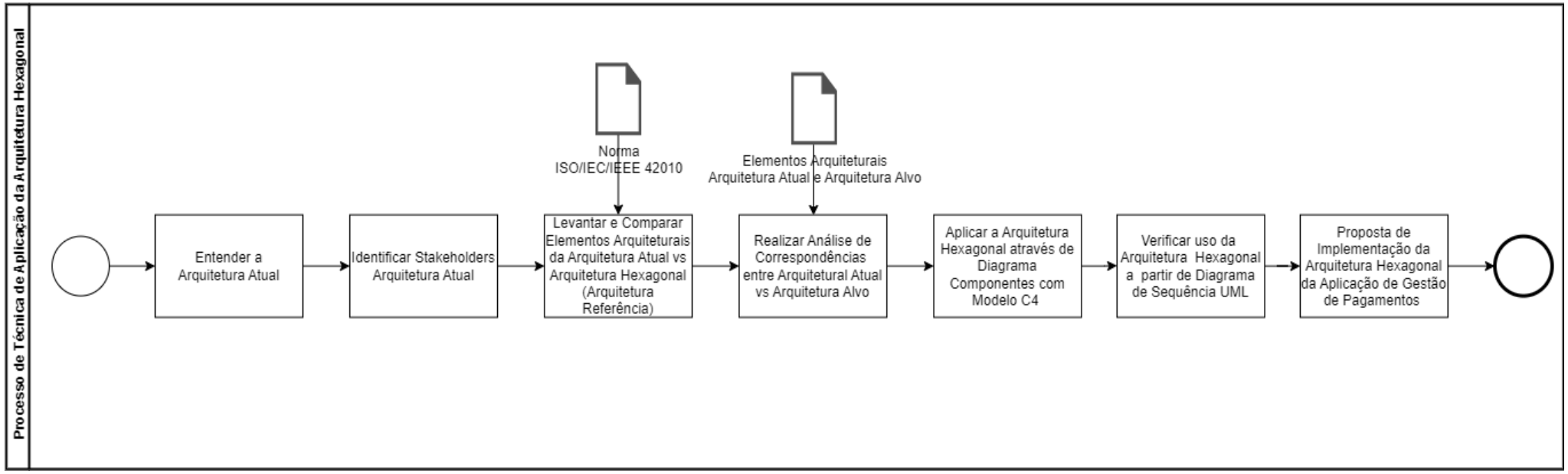
3.1 Procedimento da Técnica para Aplicabilidade da Arquitetura Hexagonal

Neste capítulo será demonstrado a técnica utilizada para aplicação da Arquitetura Hexagonal.

Para elaboração da técnica será apresentado um diagrama de processos do BPMN ilustrando cada etapa de estudo para aplicabilidade da Arquitetura Hexagonal.

Figura 6 - Processo de Técnica de Aplicação da Arquitetura Hexagonal

Processo de Técnica de Aplicação da Arquitetura Hexagonal

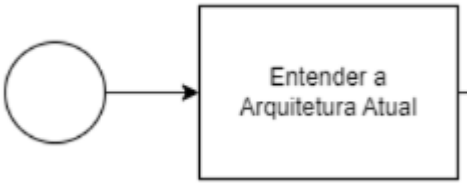


Fonte: Autora

As etapas do processo serão explicadas nos próximos tópicos.

3.2 Entender a Arquitetura Atual

Tabela 1 - Atividade Entender a Arquitetura Atual

<table>
<tr><th colspan="2">Entender a Arquitetura Atual</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Entender a Arquitetura Atual através de técnicas que identifiquem o contexto, estrutura e interação dos componentes da aplicação em estudo.</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Aplicação/Sistema</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Identificar o contexto da Aplicação/Sistema.
2. Identificar Arquitetura da Aplicação/Sistema
3. Identificar Interação dos Componentes da Aplicação/Sistema</td><td>● Diagrama de Contexto Modelo C4
● Diagrama Arquitetura Atual
● Diagrama de Componentes Modelo C4</td></tr>
</table>

Como *case* prático para o estudo será utilizado uma Aplicação de Gestão de Pagamentos de uma instituição financeira.

3.2.1 Identificação do Contexto da Aplicação Gestão de Pagamentos.

A Aplicação de Gestão de Pagamentos consiste em gerir os compromissos através do aplicativo bancário do cliente exibindo os pagamentos a vencer, vencidos, débitos automáticos e agendamentos.

O sistema tem como característica a integração com sistemas do mainframe para identificar os boletos a vencer e vencidos, os boletos em débito automático e os agendados para posterior pagamento, além de consumir APIs corporativas da instituição para identificação do cliente e conta corrente.

O sistema do mainframe por sua vez integra com a Nuclea (Sistema Nacional de Boletos) para identificar boletos emitidos por outras instituições no CPF do cliente.

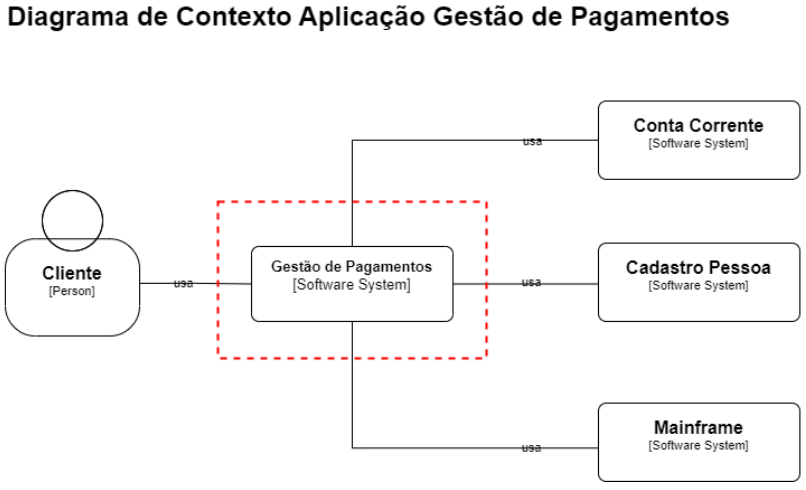
A core do sistema se divide em três fluxos, Boletos, Débitos Automático e Agendamentos sendo o de boletos um dos mais complexos por conta de algumas regras de status de boletos que são aplicadas nesse fluxo após a consulta no mainframe.

O fluxo escolhido para utilizar a técnica de aplicação de Arquitetura Hexagonal, será o de Boletos.

Para representar o contexto da aplicação o Diagrama de Contexto do Modelo C4 será aplicado.

A escolha pelo Modelo C4 para representação do Diagrama de Contexto e Diagrama de Componentes da Aplicação se dá pela facilidade de comunicação que essa técnica de modelagem de arquiteturas oferece, elucidando claramente a comunicação entre os diversos stakeholders envolvidos, desde executivos até desenvolvedores

Figura 7 - Diagrama de Contexto Aplicação Gestão de Pagamentos

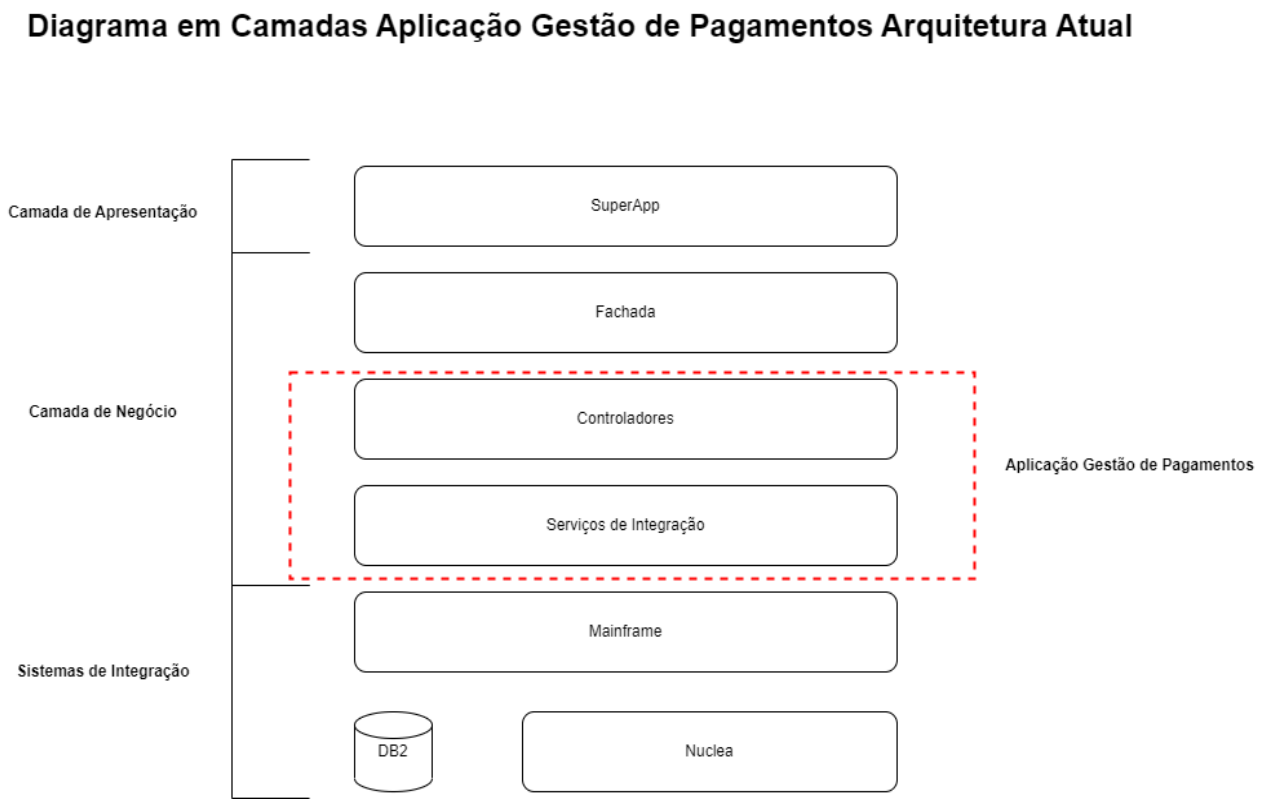


Fonte: Autora

3.2.2 Identificação da Arquitetura da Aplicação Gestão de Pagamentos.

Essa aplicação foi desenvolvida em Arquitetura de Camadas, uma característica da Arquitetura em Camadas é a forte dependência da camada superior com a camada inferior do sistema causando complexidade de manutenção e teste a cada alteração, conforme citado no tópico *2.7 Comparação Arquitetura Hexagonal vs Demais Arquiteturas e DDD (Domain Driven Design)* deste trabalho.

Figura 8 - Diagrama em Camadas Aplicação Gestão de Pagamentos Arquitetura Atual

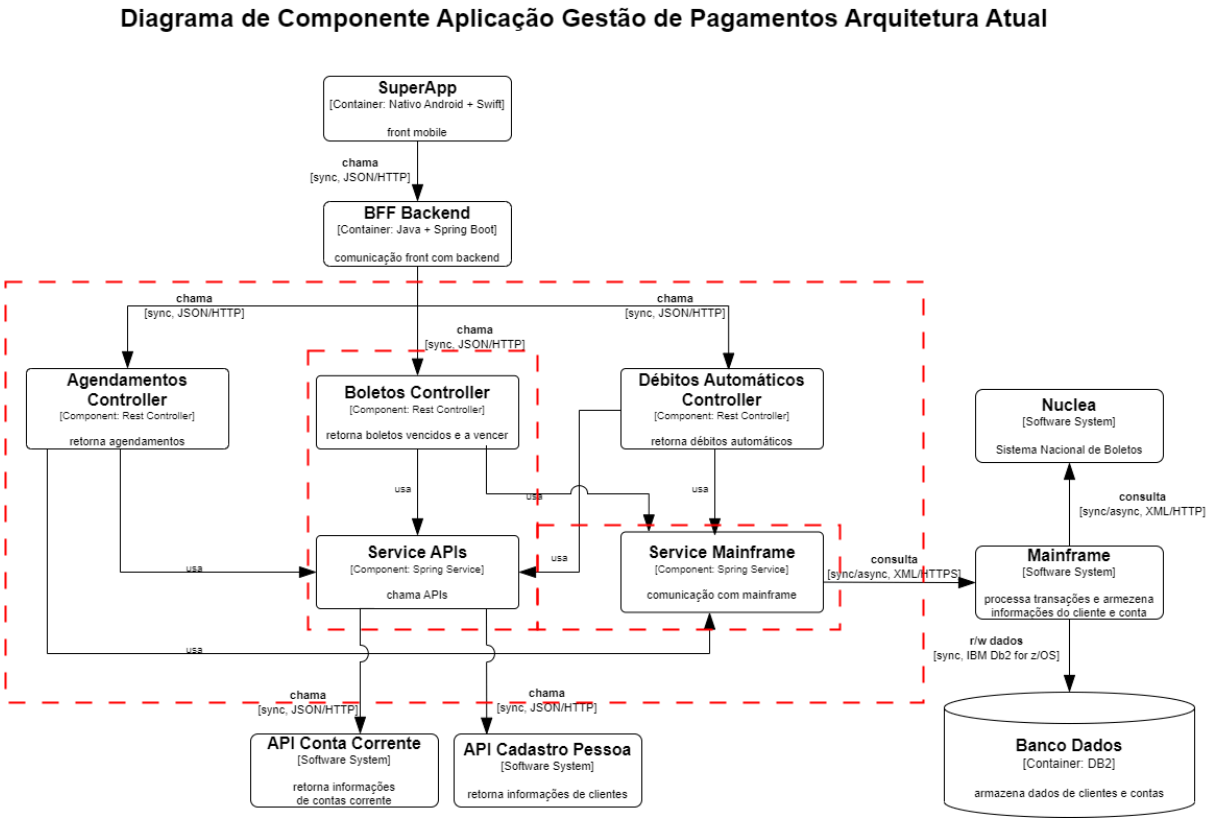


Fonte: Autora

3.2.3 Identificação Interação dos Componentes da Aplicação Gestão de Pagamentos

A partir da identificação da Arquitetura da Aplicação é possível obter uma visão mais detalhada da interação dos componentes através do Diagrama de Componentes do Modelo C4.

Figura 9 - Diagrama de Componentes Aplicação Gestão de Pagamentos Arquitetura Atual



Fonte: Autora

Observando o diagrama em questão é perceptível a forte dependência dos componentes de entrada *(Controllers)* da aplicação com os *Services.* Todas as entradas *Controllers* interagem com esses componentes ocasionando uma **difícil manutenção** pela complexidade e quantidade de interações com aplicações externas e regras de negócio.

O **teste** se torna **custoso** nesse cenário, pois qualquer alteração feita em algum dos *“Services”* todas as entradas *“Controllers”* (agendamentos, boletos e débito automáticos) precisam ser testadas para garantir que nenhuma funcionalidade da aplicação foi afetada por conta de alguma alteração.

3.3 Identificar Stakeholders da Arquitetura Atual

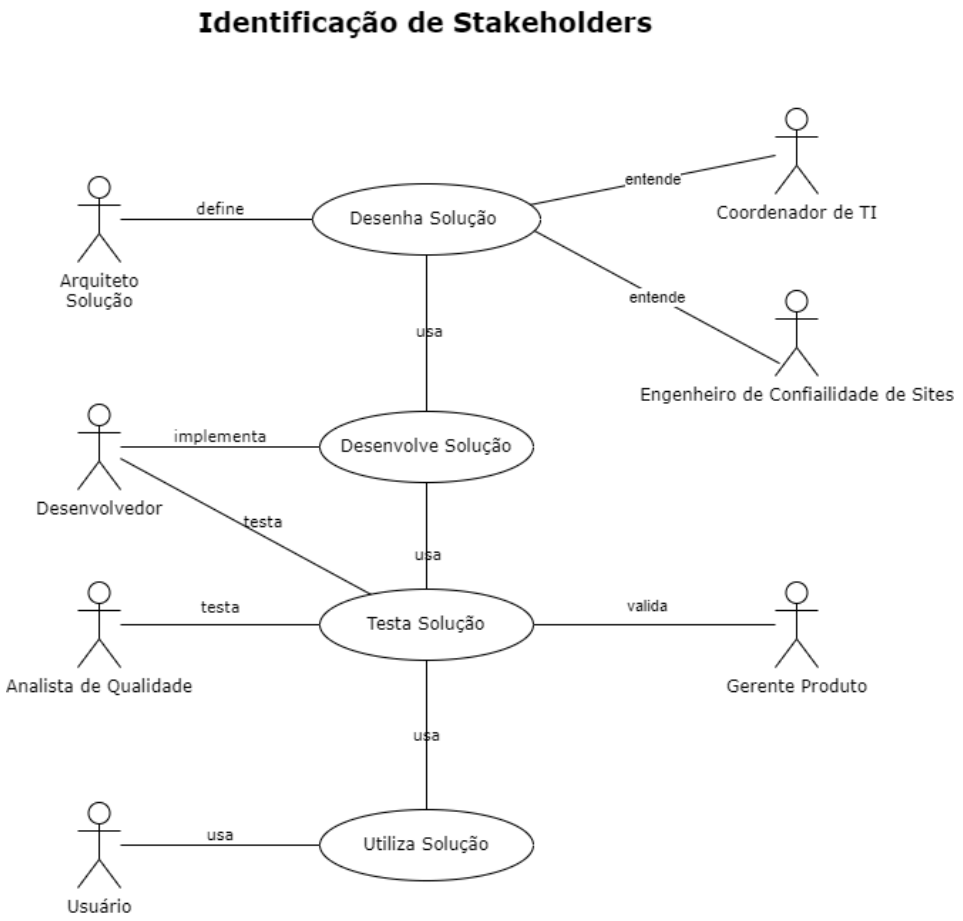
Tabela 2 - Atividade Identificar Stakeholders da Arquitetura Atual

<table>
<tr><th colspan="2">Identificar Stakeholders da Arquitetura Atual</th></tr>
<tr><td>Descrição</td><td rowspan="6"> 
</td></tr>
<tr><td>Identificar os stakeholders da aplicação/sistema para apoio no levantamento de requisitos</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Aplicação/Sistema</td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Identificar stakeholders da aplicação/sistema.</td><td>● Diagrama de Caso de Uso UML</td></tr>
</table>

Para identificação dos stakeholders será utilizada a técnica de Caso de Uso da UML que permite identificar os atores que interagem com o sistema. Os atores representam usuários, sistemas externos ou qualquer entidade que execute ações ou receba respostas. Cada ator será representado por uma figura de *"stickman*" e conectado aos casos de uso que ele pode executar, desenhados como elipses.

A clareza visual desse diagrama comunica rapidamente as principais interações dos atores, ajudando na compreensão das funcionalidades e das expectativas dos stakeholders.

Figura 10 - Caso de Uso Identificação de Stakeholders



Fonte: Autora

A identificação dos stakeholders é importante pelo fato deles serem impactados diretamente com os problemas da aplicação, sendo no contexto da Aplicação de Gestão de Pagamentos a manutenibilidade e testabilidade.

Qualquer alteração ou mudança de arquitetura é necessário estar alinhado com os stakeholders, inclusive para levantamento dos requisitos, eles reportam os problemas referente a Arquitetura Atual.

No contexto da Aplicação de Gestão de Pagamentos os stakeholders identificados são: Arquiteto de Solução, Coordenador de TI, Desenvolvedores, Analista de Qualidade, Engenheiro de Confiabilidade de Sites, Gerente de Produtos e Usuário. Todos eles desempenham um papel importante na Aplicação de Gestão de Pagamentos.

O Arquiteto de Solução é responsável pela solução da aplicação, ele desenha a arquitetura que melhor convém às necessidades dos demais stakeholders envolvidos.

O Coordenador de TI, participa do desenho de arquitetura juntamente com o Arquiteto de Solução contribuindo e sugerindo possíveis soluções de acordo com a necessidade do time de produtos e do negócio, ele é responsável por traduzir a “linguagem” de negócios para a “linguagem” de tecnologia e por orquestrar o time de desenvolvedores para garantir que o desenho proposto pelo arquiteto seja aplicado na aplicação/sistema.

Os Desenvolvedores participam do desenho de solução opinando e sugerindo, por serem responsáveis pela construção e tornarem real o desenho de solução do Arquiteto de Softwares.

O Analista de Qualidade verifica a funcionalidade da aplicação/sistema, encontram o maior número de bugs possíveis antes mesmo que a aplicação chegue nas mãos do usuário. Eles são responsáveis por garantir a qualidade da aplicação/sistema.

O Engenheiro de Confiabilidade de Sites monitora o desempenho da aplicação no ambiente produtivo, se antecipando a possíveis falhas que podem acontecer ou resolvendo falhas e problemas que acontecem com ambiente e aplicação/sistema durante o seu funcionamento.

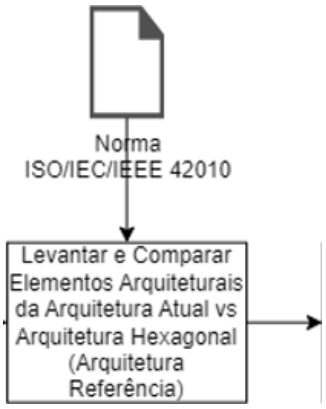
O Gerente de Produtos define os requisitos funcionais da aplicação/sistema e garante que todos os requisitos funcionais foram atendidos na aplicação/sistema de acordo com as necessidades do negócio.

O usuário no contexto da Aplicação de Gestão de Pagamentos é o cliente ele se beneficia da solução ele usufrui dos requisitos funcionais para resolver problemas ou facilitar uma ação cotidiana.

A Aplicação de Gestão de Pagamentos no âmbito de requisitos funcionais possui o controle de pagamentos, como boletos a vencer, boletos vencidos, boletos agendados, tributos agendados e débitos automáticos.

3.4 Levantar e Comparar Elementos Arquiteturais da Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)

Tabela 3 - Atividade Levantar e Comparar Elementos Arquiteturais da Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)

<table>
<tr><th colspan="2">Levantar e Comparar Elementos Arquiteturais da Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Baseado na Norma ISO/IEC/IEEE 42010 (Descrição de Padrão de Arquitetura) listar e comparar os elementos da Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Aplicação/Sistema
● Norma ISO/IEC/IEEE 42010</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Listar e Comparar Elementos Arquiteturais da Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)</td><td>● Tabela listando Elementos Arquiteturais da Arquitetura Atual, Arquitetura Hexagonal (Referência) e Arquitetura Alvo</td></tr>
</table>

Focando nos problemas já identificados da Aplicação de Gestão de Pagamentos é com base na Descrição de Padrão de Arquitetura da Norma ISO/IEC/IEEE 42010, será levantando os Elementos Arquiteturais da Arquitetura Atual e da Arquitetura Hexagonal (Arquitetura de Referência) segundo a literatura de Alistair Cockburn citada no capítulo 2 desta monografia, um comparativo será realizado gerando insumos para Arquitetura Alvo.

Tabela 4 - Tabela Comparativa Elementos Arquiteturais Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)

Tabela Comparativa Elementos Arquiteturais

Arquitetura Atual vs Arquitetura Hexagonal (Arquitetura Referência)

Elementos Arquiteturais ISO/IEC/IEEE 42010	**Aplicação Gestão de Pagamentos (Arquitetura Atual)**	**Arquitetura Hexagonal por Alistair Cockburn (Arquitetura Referência)**	**Aplicação Gestão de Pagamentos (Arquitetura Alvo)**
Sistema	API Boletos	- APP (domínio/núcleo/lógica de negócio)	API Boletos
Stakeholders	- Arquiteto de Solução - Desenvolvedor - QA - Product Owner - Coordenador de TI - SRE - Usuário	-	- Arquiteto de Solução - Desenvolvedor - QA - Product Owner - Coordenador de TI - SRE - Usuário
Preocupações *(concerns)*	Requisitos Não Funcionais - Manutenabilidade - Testabilidade	Esta arquitetura é especialmente útil para construir sistemas modulares e flexíveis, facilitando testes e manutenção.	Requisitos Não Funcionais - Manutenibilidade: sistema deve ser desenvolvido de forma modular, com cada componente isolado para facilitar futuras alterações e atualizações. - Testabilidade: grau em que um sistema ou componente facilita a criação, execução e avaliação de testes, permitindo a verificação de seus comportamentos e a detecção de falhas, de forma precisa e eficiente.

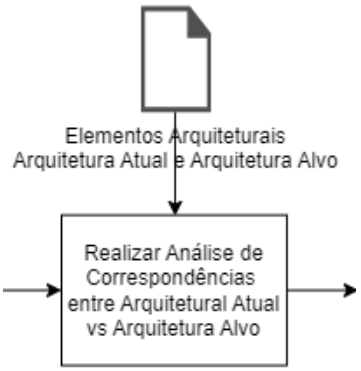
Ponto de Vista	Ponto de Vista de Organização em Camadas: Cada camada tem responsabilidades específicas e interage com outras camadas de maneira hierárquica e estruturada.	Ponto de Vista de Conectividade: aborda como os adaptadores se conectam e interagem com o núcleo da aplicação e as dependências externas	Abordar modularidade e a capacidade de troca de adaptadores sem impactar núcleo
Visão	-Visão Lógica: mostra a organização em diferentes níveis de abstração, detalhando a estrutura e as interações entre as camadas, como apresentação, lógica de negócio e dados. Cada camada possui uma função distinta, interagindo de forma definida e controlada com as camadas adjacentes	- Visão Lógica: Mostra a organização do núcleo, portas, e adaptadores em um diagrama de camadas para comunicar a estrutura modular do sistema.	- Visão Lógica: será representada por Diagrama de Componentes Modelo C4
Arquitetura	- Camada de Apresentação - Camada de Lógica de Negócios - Sistemas de Integração	- Entidades e Casos de Uso - Portas - Adaptadores - Interfaces Externas e Dependências	Hexagonal

Modelo de Arquitetura	Representado visualmente em camadas é uma representação detalhada de como o sistema é estruturado e organizado. Este modelo concretiza a visão de camadas ao detalhar os componentes, módulos e interfaces de camada, como apresentação, lógica de negócios e dados.	Representado visualmente com um hexágono central para o núcleo, cercado por adaptadores e interfaces que conectam o sistema às dependências externas. Este modelo ajuda a visualizar a independência do núcleo, realçando a modularidade e a capacidade de trocar adaptadores sem impactar a lógica de negócio.	Diagrama de Componentes Modelo C4
Relações e Correspondências Regras de Consistência	Na arquitetura em camadas, as relações especificam as interações e dependências entre as camadas, enquanto as correspondências verificam que essas relações são consistentes e atendem aos requisitos e preocupações dos stakeholders.	Separação clara entre o núcleo e os adaptadores, evitando que o núcleo dependa de detalhes de implementação externa.	Garantir que os adaptadores estejam isolados e sigam contratos (interfaces) para interagir com o núcleo
Biblioteca de Padrões	- Camada de Apresentação - Camada de Lógica de Negócios - Sistemas de Integração	- Entidades e Casos de Uso - Portas - Adaptadores - Interfaces Externas e Dependências	Hexagonal

Fonte: Autora

3.5 Realizar Análise de Correspondências entre Arquitetura Atual vs Arquitetura Alvo

Tabela 5 - Realizar Análise de Correspondências entre Arquitetura Atual vs Arquitetura Alvo

<table>
<tr><th colspan="2">Realizar Análise de Correspondências entre Arquitetura Atual vs Arquitetura Alvo</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Com os Elementos Arquiteturais da Arquitetura Alvo listados efetuar uma análise de correspondência entre Arquitetura Atual e Arquitetura Alvo</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Elementos Arquiteturais Arquitetura Atual
● Elementos Arquiteturais Arquitetura Alvo</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Realizar Análise de Correspondências entre Arquitetura Atual e Arquitetura Alvo</td><td>● Matriz de Correspondências da Arquitetura Atual vs Arquitetura Alvo</td></tr>
</table>

Inspirado na Matriz de Análise de *Gaps* do TOGAF será criado uma Matriz de Correspondências entre a Arquitetura Atual e Arquitetura Alvo visando encontrar os componentes correspondentes, componentes não correspondentes e que deverão ser removidos e componentes que necessitam ser desenvolvidos para a Arquitetura Alvo.

Tabela 6 - Matriz de Correspondências entre Arquitetura Atual vs Arquitetura Alvo

Matriz de Correspondências entre

Arquitetura Atual vs Arquitetura Alvo

arquitetura alvo ➡️ **arquitetura atual ⬇️**	**controller**	**núcleo**	**portas**	**adaptadores**	**objetos eliminados**
controller	corresponde				
camada de negócios		corresponde			
camada de integração com serviços					não corresponde
novo ➡️			não corresponde: as portas deverão ser desenvolvidas	não corresponde: os adaptadores deverão ser desenvolvidas	

Fonte: Autora

Com a Matriz de Correspondências é possível identificar que os componentes Portas e Adaptadores deverão ser desenvolvidos para Arquitetura Alvo.

3.6 Aplicar a Arquitetura Hexagonal através de Diagrama de Componentes com Modelo C4

Tabela 7 - Aplicar a Arquitetura Hexagonal através de Diagrama de Componentes com Modelo C4

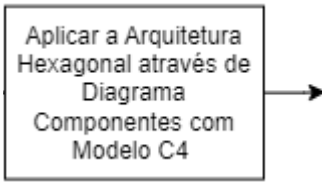
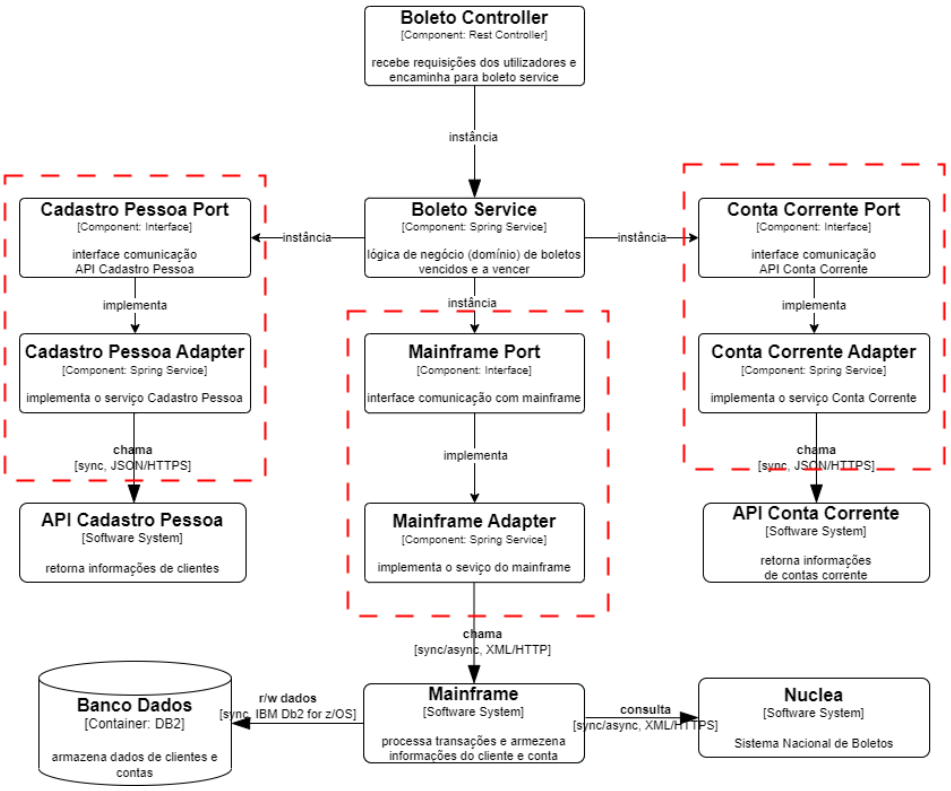
<table>
<tr><th colspan="2">Aplicar a Arquitetura Hexagonal através de Diagrama de Componentes com Modelo C4</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Aplicar a Arquitetura Hexagonal através de Diagrama de Componentes do Modelo C4 na Aplicação Gestão de Pagamentos com os componentes gerados a partir da Análise de Correspondências.</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Matriz de Correspondências entre Arquitetura Atual vs Arquitetura Alvo</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Aplicar a Arquitetura Hexagonal através de Diagrama de Componentes com Modelo C4</td><td>● Diagrama Componentes Modelo C4 com a Arquitetura Hexagonal aplicada na Aplicação de Gestão de Pagamentos</td></tr>
</table>

Figura 11 - Diagrama de Componente Aplicação Gestão de Pagamentos com Arquitetura Hexagonal Aplicada

Diagrama de Componente Aplicação Gestão de Pagamentos - [API Boletos Arquitetura Alvo]



Fonte: Autora

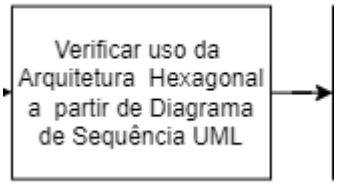
Através do Diagrama de Componentes do Modelo C4 é possível visualizar as portas e adaptadores da Arquitetura Hexagonal sendo aplicada na Aplicação de Gestão de Pagamentos trazendo interdependência dos componentes isolando o domínio da aplicação.

Boleto Service é o núcleo da aplicação e para cada sistema externo teremos uma Porta e um Adaptador para conexão.

A Porta expõem as funções dos Adaptadores de forma abstrata e nas funções dos Adaptadores teremos a lógica de conexão com os sistemas externos.

3.7 Verificar o Uso da Arquitetura Hexagonal a partir de Diagrama de Sequência da UML

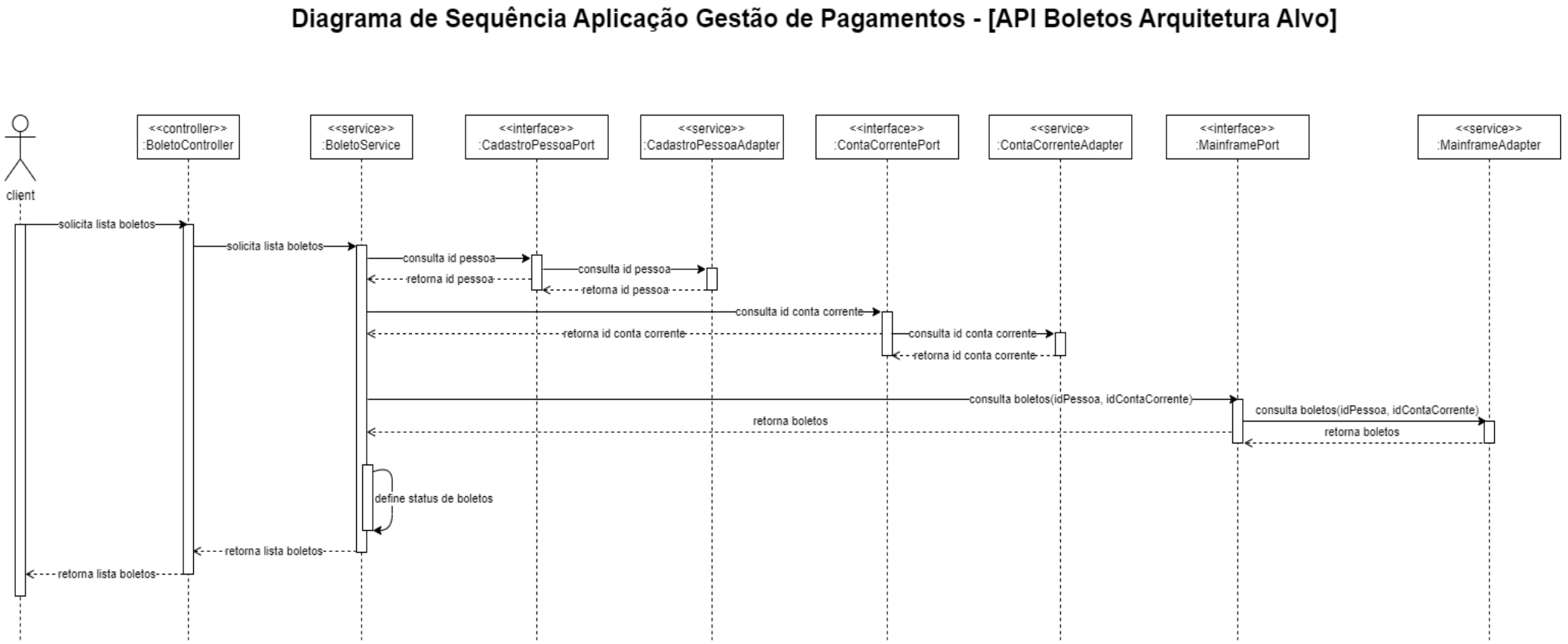
Tabela 8 - Verificar o Uso da Arquitetura Hexagonal a partir de Diagrama de Sequência da UML

<table>
<tr><th colspan="2">Verificar o Uso da Arquitetura Hexagonal a partir de Diagrama de Sequência da UML</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Através do Diagrama de Sequência da UML verificar a eficácia da Arquitetura Hexagonal para mitigar o problema de manutenibilidade e testabilidade na Aplicação de Gestão de Pagamentos</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Diagrama de Componentes Modelo C4 com Arquitetura Hexagonal aplicada na Aplicação Gestão de Pagamentos</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Verificar o Uso da Arquitetura Hexagonal a partir de Diagrama de Sequência da UML</td><td>● Diagrama de Sequência da UML da Aplicação Gestão de Pagamentos com a Arquitetura Hexagonal Aplicada</td></tr>
</table>

Usando o Diagrama de Sequência da UML é possível verificar a coesão e a interdependência dos componentes sugeridos pela Arquitetura Hexagonal na Aplicação de Gestão de Pagamentos.

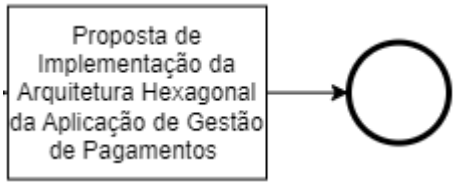
Dessa forma é possível ter uma visão se de fato a Arquitetura Hexagonal resolve o problema de manutenibilidade e testabilidade da aplicação.

Figura 12 - Diagrama de Sequência UML da Aplicação Gestão de Pagamentos com Arquitetura Hexagonal Aplicada



3.8 Proposta de Implantação da Arquitetura Hexagonal da Aplicação Gestão de Pagamentos

Tabela 9 - Proposta de Implementação da Arquitetura Hexagonal da Aplicação Gestão de Pagamentos

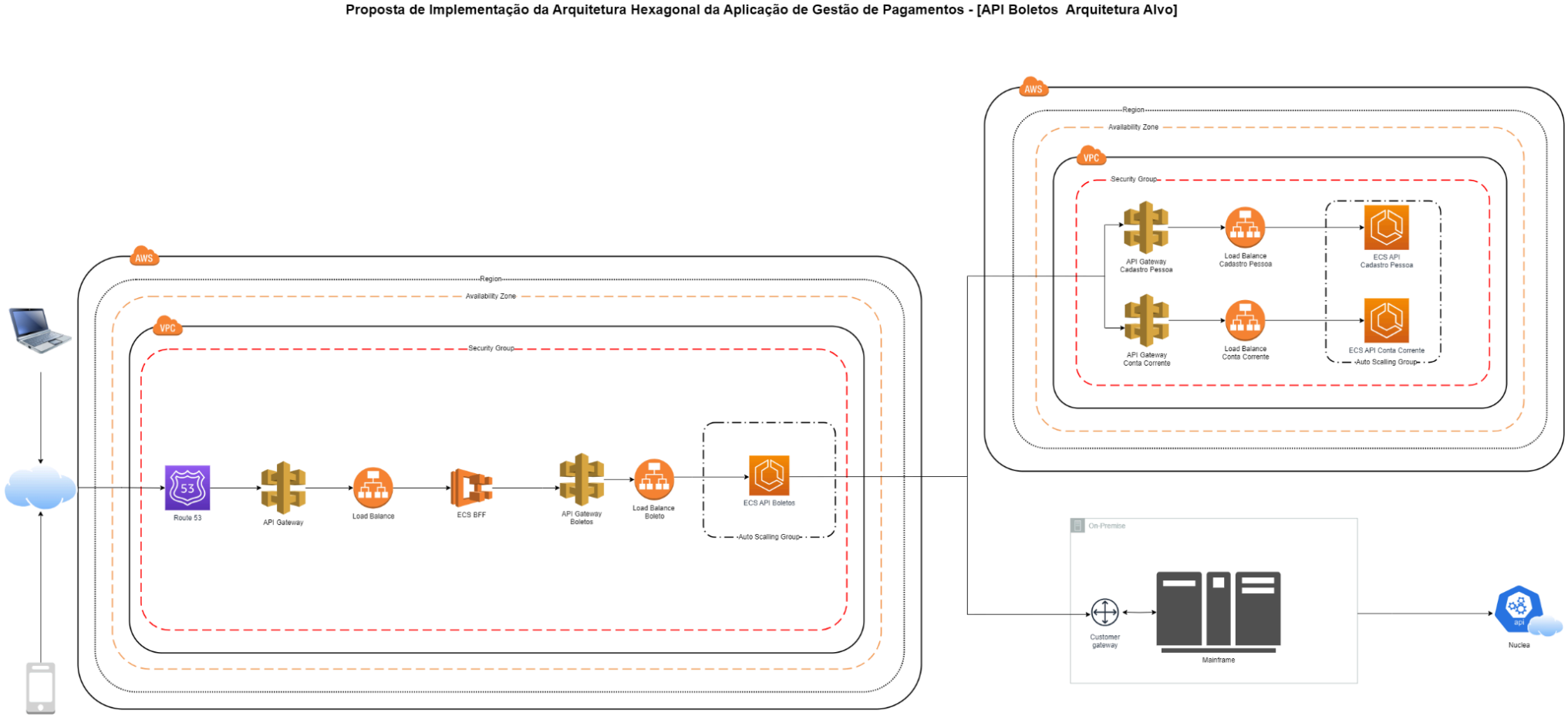
<table>
<tr><th colspan="2">Proposta de Implantação da Arquitetura Hexagonal da Aplicação Gestão de Pagamentos</th></tr>
<tr><td>Descrição</td><td rowspan="5"> 
</td></tr>
<tr><td>Através de Diagrama de Visão Tecnológica propor um modelo de implementação da Aplicação Gestão de Pagamentos com Arquitetura Hexagonal aplicada.</td></tr>
<tr><td>Executores</td></tr>
<tr><td>Arquiteto de Software</td></tr>
<tr><td>Entradas</td></tr>
<tr><td>● Diagrama de Componentes Modelo C4 com Arquitetura Hexagonal aplicada na Aplicação Gestão de Pagamentos</td><td></td></tr>
<tr><td>Tarefas</td><td>Saídas</td></tr>
<tr><td>1. Proposta de Implantação da Arquitetura Hexagonal da Aplicação Gestão de Pagamentos.</td><td>● Diagrama de Visão Tecnológica da Aplicação Gestão de Pagamentos - [API Boletos]</td></tr>
</table>

Como proposta para instalação da Aplicação Gestão de Pagamentos especificamente para API Boletos, conforme abordado no tópico *3.2.1 Identificação do Contexto da Aplicação Gestão de Pagamentos,* apenas esse fluxo seria utilizado como *case* prático para aplicação da Arquitetura Hexagonal, será utilizado infraestrutura AWS Cloud.

A sugestão é a API Boleto ser instalada em AWS ECS (Elastic Container Service).

A indicação do ECS simplifica a implantação de aplicações conteinerizadas, reduzindo a complexidade operacional.

Figura 12 - Diagrama de Visão Tecnológica da Aplicação Gestão de Pagamentos - [API Boletos]



3.9 Considerações do Capítulo

Este capítulo teve como objetivo principal avaliar a aplicabilidade da Arquitetura Hexagonal na mitigação de problemas relacionados à manutenibilidade e testabilidade na Aplicação de Gestão de Pagamentos.

A manutenibilidade, entendida como a capacidade de modificar e aprimorar sistemas ao longo de seu ciclo de vida, é um dos desafios recorrentes em sistemas complexos. Igualmente importante, a testabilidade, que se refere à facilidade de verificar e validar funcionalidades de um sistema, representa um aspecto crítico para garantir qualidade e confiabilidade.

A Arquitetura Hexagonal, também conhecida como Arquitetura de Portas e Adaptadores, propõe um modelo que visa desacoplar o núcleo da aplicação de suas dependências externas, como bancos de dados, frameworks e APIs. Essa separação promove uma organização modular, permitindo que alterações em componentes externos não afetem diretamente a lógica central da aplicação.

No contexto da Aplicação de Gestão de Pagamentos a Arquitetura Hexagonal oferece mecanismos para encapsular essas integrações, reduzindo impactos em caso de mudanças e aumentando a flexibilidade do sistema.

Por meio de técnicas consagradas da Arquitetura de Software, como identificação de padrões, aplicação de princípios de design e análise de requisitos, é possível fundamentar a escolha de uma arquitetura que atenda às demandas específicas do domínio. Nesse processo, o uso de argumentos sólidos, baseados em boas práticas e estudos de caso, contribui para decisões técnicas mais assertivas.

Em suma, a aplicação dos conceitos da Arquitetura Hexagonal pode não apenas solucionar problemas específicos de manutenibilidade e testabilidade, mas também estabelecer uma base robusta para o desenvolvimento contínuo, favorecendo a evolução do sistema e a adaptação às mudanças de mercado.

4. ANÁLISE DE RESULTADOS

A análise de resultados deste trabalho tem como foco avaliar a aplicabilidade da Arquitetura Hexagonal na Aplicação de Gestão de Pagamentos, conforme proposto nos objetivos da monografia. Este capítulo examina os resultados obtidos e objetivos apresentados, relacionando-os com a literatura revisada e discutindo os desafios e benefícios identificados ao longo do estudo.

4.1 Avaliação dos Objetivos Propostos

O principal objetivo deste trabalho foi propor e avaliar um método para a aplicabilidade de uma arquitetura de software que mitigue problemas relacionados à manutenibilidade e estabilidade. Especificamente, o estudo focou na aplicação da Arquitetura Hexagonal em uma aplicação real de Gestão de Pagamentos.

Os objetivos foram amplamente atingidos, com a validação dos benefícios do modelo proposto no âmbito de desacoplamento, organização modular e flexibilidade do sistema. Além disso, uma técnica de aplicabilidade de arquiteturas baseado em diretrizes da norma ISO/IEC/IEEE 42010 e em Análise de Correspondências inspirado na análise de *Gaps* TOGAF mostrou-se eficaz, permitindo uma transição estruturada entre a Arquitetura Atual e a Arquitetura Alvo.

4.2 Melhorias Identificadas

Os principais benefícios observados com a aplicação da Arquitetura Hexagonal foram a flexibilidade na testabilidade, organização do sistema em portas e adaptadores que permite que testes unitários sejam realizados no núcleo da aplicação, sem dependências externas, reduzindo o tempo de execução de testes e aumentando a confiabilidade dos resultados.

Melhora na manutenibilidade, a separação de responsabilidades e o desacoplamento entre o núcleo da aplicação e os componentes externos facilitam as atualizações e correções, reduzindo o risco de introdução de novos erros.

4.3 Comparativo Arquitetura Atual vs Arquitetura Alvo

A Matriz de Correspondências inspirada na Análise de *Gaps* TOGAF revelou que objetos não correspondentes como componentes com funções duplicadas ou excessivamente acoplados foram substituídos por adaptadores específicos.

Objetos foram criados para as portas e adaptadores para mediar a interação entre o núcleo da aplicação e as dependências externas.

Elementos como controladores foram preservados, mas com ajustes para se alinhar à nova arquitetura.

A transição para a Arquitetura Alvo demonstrou ser viável e vantajosa, com uma organização mais robusta para suportar evoluções futuras.

4.4 Comparação com a Literatura

A revisão bibliográfica indicou que a Arquitetura Hexagonal se diferencia das Arquiteturas em Camadas e Limpa pelo uso de Portas e Adaptadores, que promovem um desacoplamento total. Os resultados obtidos corroboram os estudos de Alistair Cockburn, que destacam a flexibilidade e a testabilidade como os principais benefícios da abordagem.

Por outro lado, os *trade-offs* identificados na literatura, como maior complexidade inicial e curva de aprendizado, também foram observados durante o estudo. Estes desafios podem ser mitigados por meio de treinamentos e documentação adequada.

4.5 Benefícios e Limitações

Como benefícios destaca-se a possível redução de débitos técnicos, a manutenibilidade aumentada pode resultar em menos custos associados a correções e melhorias futuras.

A Testabilidade aprimorada pode facilitar significativamente a utilização de TDD (Test Driven Development) e BDD (Behavior Driven Development).

A arquitetura resultante promoveu maior clareza na separação de responsabilidades com a organização modular.

Como limitações deve ser levado em consideração a complexidade inicial, o tempo e esforço necessários para implementar a Arquitetura Hexagonal que podem ser superiores aos de outras arquiteturas.

Desenvolvedores menos experientes podem enfrentar dificuldades em compreender e aplicar o modelo pelo tempo da curva de aprendizado.

4.6 Considerações do Capítulo

Os resultados obtidos reforçam a validade da Arquitetura Hexagonal para sistemas que exigem alta manutenibilidade e testabilidade. No caso da Aplicação de Gestão de Pagamentos, a aplicação dos conceitos de portas e adaptadores não apenas mitiga problemas existentes, mas também criou uma base sólida para evolução futura.

Espera-se que a técnica desenvolvida neste trabalho sirva como referência para outros sistemas que necessitam de um processo estruturado para aplicar uma arquitetura, contribuindo para a expansão e melhoria das práticas na área de engenharia de software.

5. CONSIDERAÇÕES FINAIS

Este capítulo apresenta as conclusões, contribuições do trabalho e algumas possibilidades de trabalhos futuros

5.1 Conclusões

A presente monografia demonstrou a relevância da aplicação da Arquitetura Hexagonal no desenvolvimento de sistemas complexos, com foco específico na Aplicação de Gestão de Pagamentos. Os resultados apresentados evidenciam que, ao isolar o núcleo da aplicação das dependências externas por meio de portas e

adaptadores, a arquitetura promove maior flexibilidade, escalabilidade e testabilidade. Esses fatores são determinantes para reduzir custos com manutenção, minimizar débitos técnicos e garantir que o sistema se mantenha adaptável às mudanças ao longo do tempo.

Além da técnica aplicada ao caso prático, o trabalho propôs um modelo sistemático para a possível aplicação de arquiteturas de software. Esse modelo combina diretrizes estabelecidas na norma ISO/IEC/IEEE 42010, Análise de Correspondências inspirado na Análise de *Gaps* do framework TOGAF e o uso de diagramas do Modelo C4. Tal abordagem oferece um roteiro claro e replicável que pode ser utilizado em outros contextos da engenharia de sistemas. Ele não apenas facilita a escolha de uma arquitetura adequada às necessidades específicas do projeto, mas também fundamenta essa decisão com base em critérios objetivos e alinhados às melhores práticas da área.

5.2 Contribuição Acadêmica

Como contribuição acadêmica e prática, este trabalho apresenta a Arquitetura Hexagonal como uma alternativa eficiente às arquiteturas tradicionais, destacando-se especialmente em sistemas que demandam alta manutenibilidade e testabilidade. Por meio de sua abordagem baseada no desacoplamento entre o núcleo da aplicação e as dependências externas, a arquitetura proporciona uma organização modular e flexível, facilitando a evolução contínua dos sistemas, mesmo em ambientes dinâmicos e de alta complexidade.

Os benefícios observados incluem a possibilidade de reduzir débitos técnicos, simplificar o processo de manutenção e melhorar a qualidade geral do sistema por meio da separação clara de responsabilidades. Além disso, a estrutura de portas e adaptadores torna a testabilidade um elemento central, permitindo que metodologias como TDD (Test Driven Development) e BDD (Behavior Driven Development) sejam aplicadas com maior eficiência. Essas vantagens são particularmente relevantes em sistemas críticos, como a Aplicação de Gestão de Pagamentos explorada neste estudo.

Entretanto, apesar dos benefícios significativos, a adoção da Arquitetura Hexagonal apresenta desafios que precisam ser considerados. A implementação inicial pode demandar mais tempo e recursos, em comparação com arquiteturas tradicionais, devido à necessidade de estruturar portas, adaptadores e configurar um núcleo desacoplado. Além disso, a curva de aprendizado associada ao modelo é um fator relevante, principalmente para desenvolvedores menos experientes ou equipes acostumadas a paradigmas mais simples, como arquiteturas em camadas.

Esses desafios, no entanto, podem ser mitigados com a oferta de treinamentos específicos e o uso de boas práticas de engenharia, como documentação clara e frameworks que suportem o padrão arquitetural. Com isso, espera-se que a Arquitetura Hexagonal não apenas resolva problemas técnicos e organizacionais, mas também contribua para o avanço do desenvolvimento de sistemas robustos e preparados para mudanças futuras.

5.2 Trabalhos Futuros

Para trabalhos futuros, recomenda-se a aplicação da técnica desenvolvida em outros domínios e contextos de sistemas, a fim de validar sua generalida0de e eficiência. Adicionalmente, seria interessante explorar a integração da Arquitetura Hexagonal com práticas emergentes, como arquiteturas orientadas a eventos e sistemas baseados em inteligência artificial, analisando os benefícios e limitações em cenários contemporâneos. Por fim, estudos complementares poderiam investigar estratégias para minimizar os custos iniciais de adoção, tornando essa arquitetura ainda mais acessível e viável para projetos de pequeno e médio porte.

Assim, espera-se que as contribuições deste trabalho sirvam como base sólida para decisões arquitetônicas mais assertivas e embasem novas investigações no campo da engenharia de software, ampliando o entendimento sobre as potencialidades e aplicações da Arquitetura Hexagonal.

REFERÊNCIAS

COCKBURN, Alistair; GARRIDO DE PAZ, Juan Manuel. **Hexagonal Architecture Explained.** Humans and Technology Incorporated, 2024. Edição Kindle

FOWLER, Martin; RICE David; FOEMMEL, Matthew; HIEATT, Edward; MEE, Robert; STAFFORD, Randy. **Patterns of Enterprise Application Architecture.** Addison-Wesley Professional, 2002.

MAJUMDER Mainak; ZOITL Alois **A Domain-Driven Design Oriented OPC UA Server Development Methodology for CPPS**. In 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), April, 2023. **Proceedings [...].** DOI: 10.1109/ETFA54631.2023.10275496. Disponível em: https://ieeexplore.ieee.org/abstract/document/10275496?casa_token=6EbLE4d4NCQAAAAA:DBkKeavG0_CYeI_T2eSBDsr3IhcYL2LL9QAhTL2YhxDEFg8W34dD6OBAxhopns-EDFOiiZPj. Acesso em: 15 set. 2024.

JÚNIOR Jackson; COUTINHO Pedro **Restructuring the Software Architecture: A Case Study of the CoolBiz Core Banking Platform.** In 2023, CAPSI 2023. **Proceedings [...].** Disponível em: https://aisel.aisnet.org/capsi2023/3/. Acesso em: 15 set. 2024.

FILHO, Nagib S. **Comparação entre Portas na Arquitetura Hexagonal e Interfaces na Arquitetura Limpa: Uma Análise Conceitual e Prática.** LEADERS.TEC.BR, vol 1, 26 aug 2024. Disponível em: https://leaders.tec.br/artigo/comparacao-entre-portas-na-arquitetura-hexagonal-e-interfaces-na-arquitetura-limpa-uma-analise-conceitual-e-pratica. Acesso em: 14 set. 2024.

JEMUOVIC, Valentina **Hexagonal Architecture - Ports and Adapters.** Optivem Journal, 30 mar 2023. Disponível em: https://journal.optivem.com/p/hexagonal-architecture-ports-and-adapters. Acesso 14 set. 2024.

IEEE/ISO/IEC **International Standard for Software, systems and enterprise--Architecture description**, In ISO/IEC/IEEE 42010:2022(E) , vol., no., pp.1-74, 7 Nov. 2022, **Proceedings [...].** DOI: 10.1109/IEEESTD.2022.9938446.

Disponível em: https://ieeexplore.ieee.org/document/9938446. Acesso em 29 out. 2024.

Padrão de Arquitetura Hexagonal, Disponível em: https://docs.aws.amazon.com/pt_br/prescriptive-guidance/latest/cloud-design-patterns/hexagonal-architecture.html Acesso em 15 set. 2024

Model and Meta Model Matters, Disponível em: http://www.iso-architecture.org/ieee-1471/meta/. Acesso em 29 out. 2024

The TOGAF® Standard, Version 9.2 > Part III: ADM Guidelines & Techniques > Gap Analysis, Disponível em: https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap23.html. Acesso em 29 out. 2024.

C4 Model, Disponível em: https://c4model.com/. Acesso em 24 out. 2024

AWS Fargate para Amazon ECS, Disponível em https://docs.aws.amazon.com/pt_br/AmazonECS/latest/developerguide/AWS_Fargate.html. Acesso em 15 nov. 2024