**ARQUITETURA LIMPA**

1. O que é Arquitetura Limpa?

A ideia da Arquitetura Limpa, proposta por Robert C. Martin, é uma perspectiva de uma arquitetura modular sistema que sustentará a flexibilidade e a independência do sistema a longo prazo da teoria do design. Isso é a premissa fundamental é a separação das camadas, o que significa que o design permite uma dependência lógica mínima dos componentes internos e externos. Existem alguns princípios básicos para implementar a arquitetura:

Independência de Frameworks: o sistema não deve ser constrangido ao framework em que o programa é implementado. As estruturas devem ser usadas como ferramentas, não como peças- chave.

- Testabilidade: é necessário que o sistema na sua totalidade seja testável e a sua lógica de negócios seja isolada e separada.

- Independência da Interface do Usuário: é necessário garantir que qualquer alteração na Ul não afete a lógica dos negócios.

- Independência do Banco de Dados: é fundamental para uma arquitetura limpa que se evite tornar o banco de dados parte da lógica ou a base da arquitetura.

- Regra de Dependência: o núcleo lógico está separado de frameworks e implementações de fontes de dados; portanto, a dependência é da lógica e nunca o contrário. A imagem mais comum da arquitetura limpa é estrutura de círculos. Círculos concêntricos com o núcleo que contém todas as regras de negócios e interfaces externas como frameworks, UI e banco de dados que se movem para longe do centro.

2. Vantagem

- Facilidade de Manutenção: O desacoplamento das camadas facilita a atualização, ajuste e escalonamento do sistema.

- Reutilização: Componentes centrais podem ser reutilizados em diferentes aplicações.

- Teste Independente: Testar a lógica de negócios é direta, uma vez que ela é isolada.

- Resiliência à Mudança: Mudanças feitas em tecnologias externas, como a troca do banco de dados ou do framework front-end, têm pouco impacto.

Desvantagem

- Curva de Aprendizado: conhecer a Arquitetura Limpa e compreender profundamente seus conceitos é necessário para sua implementação.

- Complexidade Inicial: O custo adicional de design e desenvolvimento pode ser inicialmente alto. - Engenharia Demais: Para sistemas simples, a implementação de um sistema totalmente baseado na Arquitetura Limpa pode ser demais.

3. Principais distinções nas arquiteturas comuns

- Comparado à Arquitetura em Camadas: embora pareça que a Arquitetura Limpa é semelhante à Arquitetura em Camadas, essa última tem dependência unidirecional, da interface do usuário

para o banco de dados. No entanto, a Arquitetura Limpa coloca uma lógica de negócios no centro e permite que informações externas orbitais sejam armazenadas do lado de fora.

- Comparado à Arquitetura de Hexágono: embora tenhamos o foco no desacoplamento, a última não enfatiza claramente a hierarquização de dependências.

- Comparado à Arquitetura do Microkernel: enquanto a Arquitetura do Microkernel é projetada para extensibilidade usando plug-ins, a Arquitetura Limpa é construída em torno de como todos os componentes internos se relacionam.

Considerações Finais e Trabalhos Futuros

Acreditamos que a melhor forma de avaliar a proposta do Archunit neste trabalho é aplicá-la diretamente em algum projeto de implantação Arquitetura Limpa. Dessa forma, podemos elaborar ou revisar um sistema simples e avaliar quanto ao uso da ferramenta. Inspire-se: software.ohnestoproject.com.

4. Aplicação a um serviço para uma livraria

4.1 Descrição do serviço

Os seguintes recursos disponíveis estão:

- Criação, exclusão, atualização e busca por livros.

- Consulta ao serviço ms-book-precification: o preço é retirado do livro baseado no ISSN.

- Banco de dados: PostgreSQL ou MongoDB.

4.2 Razões para escolha do Banco de Dados

\*PostgreSQL\*

- Justificativa: o PostgreSQL será usado.

\* Motivos para escolha:\*

- Modelo Relacional: estrutura bem definida para tabelas e relações - ideal para livros e seus

atributos fixos: ISSN, nome e preço.

- Robustez: tI suporta transações ACID, garantindo consistência nas operações de criação e exclusão.

- Compatibilidade: Boa integração com ferramentas modernas e APls REST.

- Comunidade e Suporte: Um dos bancos de dados relacionais mais populares com ampla documentação.

- Custo: Código aberto, não é preciso pagar por licenciamento.

\*Comparação com MongoDB:\*

- MongoDB é melhor para dados não estruturados e flexíveis, enquanto o serviço de livros ainda tem uma estrutura sólida. Portanto, o PostgreSQL é uma solução melhor.

- Operações relacionais funcionarão de forma mais eficiente no PostgreSQL.

4.3 Modelagem de Arquitetura Limpa \*Componentes do sistema:\*

- Entidades: Representam as principais regras de negócios (por exemplo, Livro com atributos de ID, Nome, ISSN, Data de Lançamento e Preço).

- Casos de uso: Contêm lógica de aplicação (por exemplo, registrar um livro, consultar o preço de ms-book-precification).

- Adaptadores: Responsáveis ​​pela tradução entre externo e interno camadas de comunicação (por exemplo, REST API, acesso ao banco de dados).

- Infraestrutura: comunicação com o banco de dados PostgreSQL e serviço ms-book-precification.

