INSTITUTO FEDERAL
São Paulo
Câmpus Campinas

TADS - ASW
Setembro 2021

ARQUITETURA LIMPA

Ewerton Soares da Silva
CP3001393

João Lucas Sena Oliveira
CP3001229

Geazi José da Silva
CP3013782

SUMÁRIO

1. CONCEITOS BÁSICOS

Arquitetura Limpa (*Clean Architecture*) é um padrão arquitetural proposto por Robert Martin – mais conhecido como Uncle Bob – com o objetivo de promover a implementação de sistemas que favorecem reusabilidade de código, coesão, independência de tecnologia e testabilidade

A regra principal da arquitetura limpa é que as dependências do código só podem ser movidas dos níveis externos para dentro. O código nas camadas internas pode não ter conhecimento das funções nas camadas externas. As variáveis, funções e classes (quaisquer entidades) que existem nas camadas externas não podem ser mencionadas nos níveis mais internos. Recomenda-se que os formatos de dados também fiquem separados entre os níveis.

A arquitetura limpa foi criada por Robert C. Martin e promovida em seu blog, Uncle Bob. Como outras filosofias de design de software, a arquitetura limpa tenta fornecer uma metodologia econômica que torne mais fácil desenvolver código de qualidade com melhor desempenho, seja mais fácil de alterar e tenha menos dependências.

Visualmente, os níveis de arquitetura limpa são organizados em um número não especificado de anéis. Os níveis externos dos anéis são mecanismos de nível inferior e os níveis internos superiores contêm políticas e entidades.

2. COMPONENTES DA ARQUITETURA

A arquitetura limpa tem quatro principais camadas: entidades, casos de uso, adaptadores de interface e frameworks externos, porém podem ser criadas mais camadas de acordo com a necessidade do software.

2.1. Entidades (Entities)

O conceito principal é de que esta camada deve conter tudo que seja pertinente ao sistema em relação à lógica de negócios, de modo mais genérico possível, ou seja, que tenham menor probabilidade de alterações quando houverem mudanças externas. Nesta camada se encontram os modelos de objetos utilizados para a lógica de negócios, como os POJOs (Plain Old Java Object) e Data Class em Kotlin.

2.2. Casos de Uso (Use Cases)

Se encontram nesta camada, as regras de negócios mais específicas do sistema e ainda, é o lugar onde será verificado como a camada de apresentação receberá os dados.

2.3. Adaptadores de Interface (Interface Adapters)

Esta camada tem como responsabilidade, realizar a conversão dos dados, de modo que seja mais acessível e conveniente possível, para as camadas de Entidades e Casos de Uso.

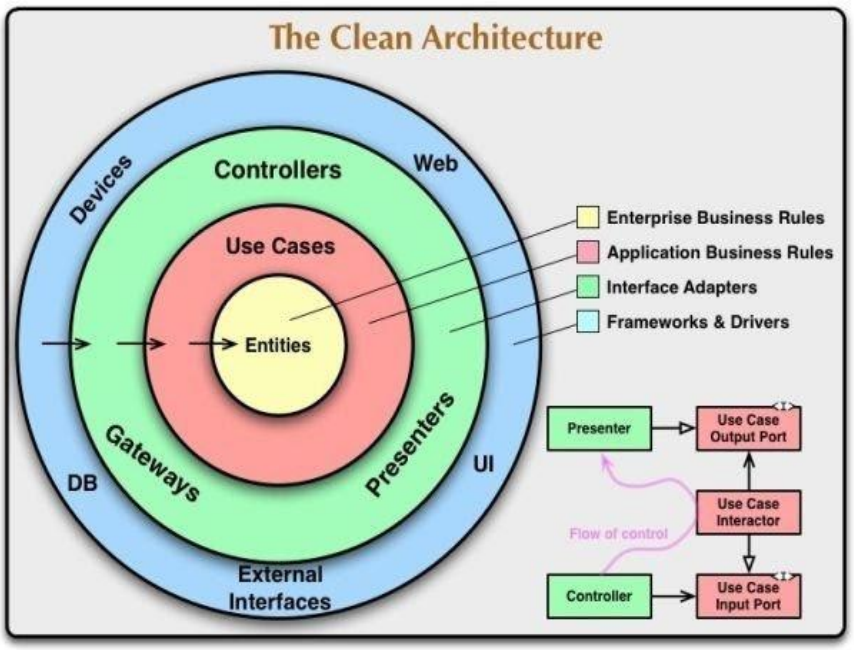
Como exemplo, são capturados dados de entrada em uma interface de usuário e os empacotam de modo conveniente para os casos de uso e entidades. Logo em seguida, recupera os dados de saída dos casos de uso e entidades, e as empacotam em um formato conveniente para que sejam exibidos na interface de usuário ou sejam salvos em um banco de dados.

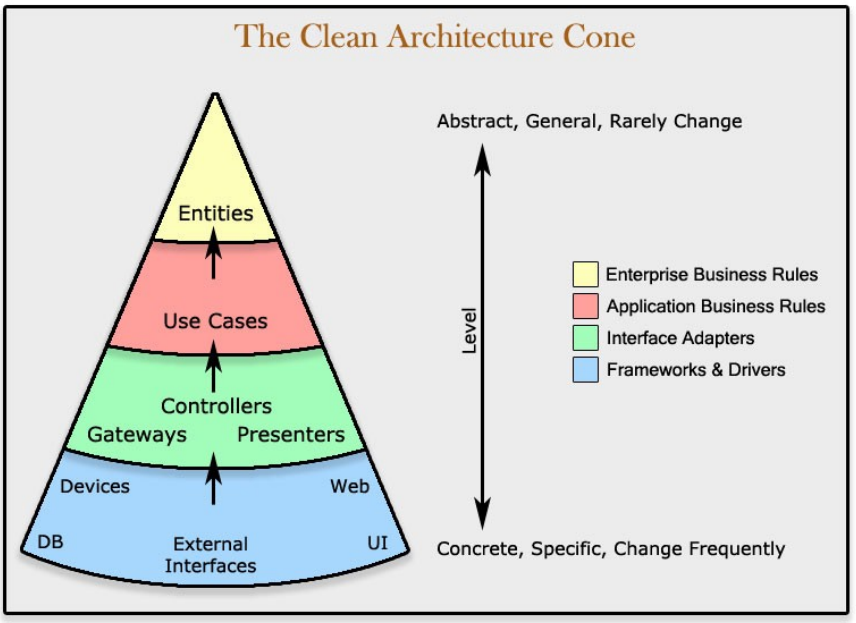
2.4. Frameworks

A camada mais externa geralmente é composta de estruturas e ferramentas de banco de dados, web, interface de usuário, etc. Nesta camada, deve possuir o mínimo de código possível e necessário de modo a interligar as camadas e interferir o mínimo possível.

3. DIAGRAMA DA ARQUITETURA

O padrão arquitetural é melhor entendido pelos diagramas abaixo e detalhes





Regra de Dependência

Em uma Arquitetura Limpa, as classes de uma camada X não devem conhecer nenhuma classe de uma camada Y mais externa. No seu livro, Uncle Bob afirma categoricamente:

> "O nome de um elemento declarado em uma camada externa não deve ser mencionado pelo código de uma camada interna.

> Isso inclui funções, classes, variáveis e qualquer outro elemento de código.”

Assim, em uma Arquitetura Limpa, as camadas centrais são mais estáveis – menos sujeitas a mudanças – do que as camadas mais externas. Por exemplo, as entidades de um sistema raramente precisam ser modificadas. Sobre os casos de uso, é verdade que eles, às vezes, precisam ser mantidos. Porém, queremos evitar que essas mudanças sejam motivadas por mudanças nas tecnologias adotadas na aplicação, como bancos de dados, frameworks e bibliotecas.

Resumindo, a Regra de Dependência garante que **entidades e casos de uso são classes “limpas” de qualquer tecnologia** ou serviço externo ao sistema.

4. VANTAGENS ARQUITETURA

4.1. Testabilidade

Por ser dividida em camadas, as regras de negócio podem ser testadas sem interferir na interface do usuário, banco de dados, servidor ou qualquer outro elemento externo.

4.2. Independência da interface do usuário

Assim como as regras de negócio podem ser testadas sem interferir em nenhum outro componente do sistema, a interface também pode ser alterada com facilidade.

Desta forma, por esse padrão, uma UI da Web pode ser substituída por uma UI do console, por exemplo, sem alterar as regras de negócio.

4.3. Independência do banco de dados

Como as camadas do projeto operam de forma independente, as regras de negócio também podem não ter vínculo com o banco de dados do sistema. Com isso, é possível trocar o Oracle ou SQL Server, por Mongo, BigTable, CouchDB ou qualquer outro.

4.4. Independência de qualquer agente externo

Na verdade, suas regras de negócios simplesmente não sabem nada sobre o mundo exterior, não estão ligadas a nenhum Framework.

5. LIMITAÇÕES DA ARQUITETURA

A dificuldade em absorver e aplicar os conceitos da Arquitetura Limpa é uma das maiores limitações do padrão arquitetural, sendo necessário considerável dedicação do desenvolvedor para superar a íngreme curva do aprendizado, especialmente quando o objetivo é o funcionamento conjunto de todas as camadas.

Outra limitação é a necessidade de o padrão exigir o acréscimo de muitas classes adicionais, circunstância que impede a utilização do modelo para projetos de baixa complexidade.

6. APLICAÇÕES DA ARQUITETURA

6.1. Utilizar nomenclatura clara e intuitiva

A nomenclatura é utilizada em diferentes partes do código da aplicação, como para atribuir nomes a variáveis, funções, classes, parâmetros e, até mesmo, os nomes dos arquivos. É importante utilizar nomes que tenham relação com a finalidade do código pois torna intuitivo e melhora a compreensão.

Além disso, o ideal é que, ao atribuir nomes às variáveis, eles sejam substantivos, enquanto que as funções devem ser nomeadas com verbos.

6.2. Realizar tratamento de erros

É importante realizar tratamentos de erros do código para garantir o bom funcionamento da aplicação e exibir mensagens esclarecedoras sobre o problema encontrado.

Negligenciar a trativa dos erros pode causar a interrupção da aplicação além de prejudicar a usabilidade de diferentes formas.

6.3. Manter os dados de configuração separado do código fonte

Os dados de configuração, como strings de conexão com o banco de dados, devem ser adicionados em um arquivo separado do código fonte. Isso permite, por exemplo, alterar a configuração do banco com facilidade sem a necessidade de modificar o código da aplicação.

7. REFERÊNCIAS

MICHIURA, Fabio. OBJECTIVE. **Clean Architecture com MVVM: o que é, vantagens e como utilizar em aplicações Android.** Disponível em <https://www.objective.com.br/insights/clean-architecture-com-mvvm/> Acesso em 11/10/2021.

MEDEIROS, Felipe. **Clean Architecture: o que é arquitetura limpa e como aplicar no desenvolvimento de softwares?** Disponível em <https://startecjobs.com/artigos/clean-architecture/> Acesso em 08/10/2021.

SILVA, James G. **Arquitetura limpa**. Disponível em <https://www.slideshare.net/jamersonweb/arquitetura-limpa-131970246> Acesso em 10/10/2021.

TECH TARGET. **Definição de "Clean Architecture".** Disponível em <*https://whatis.techtarget.com/definition/clean-architecture*> Acesso em 09/10/2021.

VALENTE, Marco Tulio. **Construindo Sistemas com uma Arquitetura Limpa**. Disponível em <https://engsoftmoderna.info/artigos/arquitetura-limpa.html> Acesso em 08/10/2021.

CARVALHO, Mayelle. Medium. **Arquitetura Limpa: O Melhor Da Arquitetura Em Camadas.** Disponível em <https://mayellecarvalho.medium.com/arquitetura-limpa-o-melhor-da-arquitetura-em-camadas-7c945715ca64> Acesso em 08/10/2021.

YOUTUBE. CÓDIGO FONTE TV. **Clean Architecture (Arquitetura Limpa)**. Disponível em <https://www.youtube.com/watch?v=ow8UUjS5vzU> Acesso em 08/10/2021.