Research, Society and Development, v. 11, n. 16, e335111637198, 2022
(CC BY 4.0) | ISSN 2525-3409 | DOI: http://dx.doi.org/10.33448/rsd-v11i16.37198

Estado da arte da pesquisa em: Clean Architecture e princípios de SOLID

State of the art research in: Clean Architecture and SOLID principles

Estado de la técnica: Arquitectura Limpia y principios SOLID

Recebido: 28/10/2022 | Revisado: 15/11/2022 | Aceitado: 01/12/2022 | Publicado: 10/12/2022

Vinicius Barros Silva Ferreira
ORCID: https://orcid.org/0000-0003-1387-2652
Instituto Federal de Educação, Ciência e Tecnologia Goiano, Brasil
E-mail: viniciusbarrosipo@gmail.com
Carlos Antônio Ferreira
ORCID: https://orcid.org/0000-0002-9246-6507
Instituto Federal de Educação, Ciência e Tecnologia Goiano, Brasil
E-mail: carlos.ferreira@ifgoiano.edu.br
Eliana Tiba Gomes Grande
ORCID: https://orcid.org/0000-0003-1155-6357
Instituto Federal de Educação, Ciência e Tecnologia Goiano, Brasil
E-mail: eliana.tiba@ifgoiano.edu.br

Resumo
A alta demanda de serviços *mobile* exige a adoção de uma abordagem que atenda aos requisitos específicos de engenharia de aplicações *mobile*. Dentre esses requisitos estão a criação de uma arquitetura que separe as responsabilidades em camadas e que adapte a mudança e correções. É na fase de arquitetura que o sistema ganha forma pelos desenvolvedores. Esta forma está na divisão desse sistema em componentes, na organização destes e no modo em que comunicam entre si. Neste sentido, o presente trabalho se propôs a elaborar uma revisão bibliométrica a partir da análise da produção científica sobre as abordagens de *Clean Architecture* e *SOLID,* disponível na base de dados Periódico Capes, entre os anos de 2012 e 2022. Os resultados apresentam um total de 27 artigos científicos a partir da aplicação dos termos de busca: *clean architecture, mobile* e *software,* das quais quatro estão alinhados ao escopo do estudo, contemplando os aspectos relacionados ao conceito de arquitetura limpa e o desenvolvimento de aplicações *mobile* utilizando a abordagem de SOLID, embora destes, dois sejam artigos de revisão, não considerando a aplicabilidade destes conceitos na construção de novas aplicações, demonstrando assim a carência de produções científicas.
Palavras-chave: Dispositivos móveis*;* Aplicativo; Arquitetura; SOLID; *Software design.*

Abstract
The high demand for mobile services requires the adoption of an approach that meets the specific engineering requirements of mobile applications. Among these requirements are the creation of an architecture that separates responsibilities into layers and that adapts to change and corrections. It is in the architectural phase that the system takes shape by the developers. This form lies in the division of this system into components, in their organization and in the way they communicate with each other. In this sense, the present work proposed to elaborate a bibliometric review from the analysis of the scientific production on the approaches of Clean Architecture and SOLID, available in the database Periódico Capes, between the years 2012 and 2022. The results present a total of 27 scientific articles from the application of the search terms: clean architecture, mobile and software, of which four are aligned with the scope of the study, covering aspects related to the concept of clean architecture and the development of mobile applications using the SOLID approach , although of these, two are review articles, not considering the applicability of these concepts in the construction of new applications, thus demonstrating the lack of scientific production.
Keywords: Mobile devices; Application; Architecture; SOLID; Software design.

Resumen
La gran demanda de servicios móviles requiere la adopción de un enfoque que cumpla con los requisitos de ingeniería específicos de las aplicaciones móviles. Entre estos requisitos está la creación de una arquitectura que separe las responsabilidades en capas y que se adapte a los cambios y correcciones. Es en la fase arquitectónica que los desarrolladores toman forma el sistema. Esta forma radica en la división de este sistema en componentes, en su organización y en la forma en que se comunican entre sí. En ese sentido, el presente trabajo se propuso elaborar una revisión bibliométrica a partir del análisis de la producción científica sobre los enfoques de Arquitectura Limpia y SOLID, disponible en la base de datos Periódico Capes, entre los años 2012 y 2022. Los resultados presentan un total de 27 artículos científicos a partir de la aplicación de los términos de búsqueda: arquitectura limpia, móvil y software, de los cuales cuatro están alineados con el alcance del estudio, abarcando aspectos relacionados con el concepto de

arquitectura limpia y el desarrollo de aplicaciones móviles bajo el enfoque SOLID, aunque de estos, dos son artículos de revisión, no considerando la aplicabilidad de estos conceptos en la construcción de nuevas aplicaciones, demostrando así la falta de producción científica.
Palabras clave: Dispositivos móviles; Aplicación; Arquitectura; SOLID; Diseño de software.

1. Introdução

Em contextos empresariais e acadêmicos cujas atividades são focadas em tecnologias de programação é comum a aplicação de métodos arquiteturais objetivando tornar o desenvolvimento de software escalável e mais amplo em manutenção, tendo como consequência maior qualidade de código e produtividade em desenvolvimento de sistemas. Estas atividades orientadas em otimização de construção de software, com aplicação de métodos de organização de código, testes e práticas de arquitetura como solução para otimização de tempo e eficiência em times de desenvolvimento, são uma consequência dos desafios levantados pela indústria de sistemas durante a fase conhecido como "crise do software", que se iniciou na década de 1960 quando os programas de computador eram de difícil manutenção, precário em testes, baixa documentação e insuficiente em escalabilidade (Pressman, 2011).

O desenvolvimento de software é, em sua maior parte, um trabalho grupal e colaborativo (Ivanics, 2016). No contexto moderno, a área de tecnologia destaca-se no cenário empresarial, onde a demanda pela entrega e por novas soluções para atender públicos em larga escala são um desafio. Desta maneira, as empresas buscam se especializar em contratar profissionais com conhecimento em tecnologias de desenvolvimento de software com foco em suas aplicações práticas, além de reter estes profissionais para que estes consigam fazer a entrega adequada dos serviços contratados e conquistar a confiança de clientes (Moreno, et al., 2009). Manter grandes equipes de desenvolvedores de software trabalhando em conjunto em tarefas específicas é um desafio para a área de gestão de tecnologia pelo fato do projeto em software conter demandas de requisitos que devem atender expectativas de entrega. Além disso, é necessário que a equipe tenha acesso a todas as informações necessárias para realizar o trabalho e de que as metas e objetivos prioritários não devam ser alterados sem necessidade relevante (Pressman, 2011).

Dentre os principais atributos para a criação da base de código de software está a criação da arquitetura e dos fluxos de dados dentre as respectivas camadas arquitetônicas. Estas são definidas como o primeiro estágio no processo de construção de projeto de software pois são responsáveis por identificar os principais componentes estruturais de um sistema e os relacionamentos entre eles (Sommerville, 2011, p.103). Além disso, as representações arquiteturais de software, em conjunto com o planejamento de projeto, são um facilitador para a comunicação entre o time de desenvolvimento e as partes interessadas do sistema computacional (Pressman, 2011)

Para condicionar a equipe de acordo com as boas práticas em arquitetura é preciso manter o código do sistema organizado, modularizado e documentado para que seja possível aos desenvolvedores a: i) compreensão dos processos envolvidos nas regras de negócio; ii) alinhamento das soluções em código aos respectivos requisitos levantados; iii) testes automatizados para prevenção de erros e facilidade de promover mudanças. Para conservar estes parâmetros em específico é preciso manter uma arquitetura de sistema que seja pouco acoplada, ou seja, que proporcione suporte aos serviços para que estes possam ser atualizados em ambiente de produção de maneira independente, sem precisar atualizar outros serviços (Kim, et al., 2018, p.90).

Para manter a equipe de desenvolvimento de software alinhada em conjunto com as regras de negócios levantados pelos stakeholders responsáveis pelo projeto é preciso manter uma arquitetura que proponha as partes envolvidas organização, correção constante e facilidade em sua manutenção, proporcionando assim, constante expansão e escalabilidade de funcionalidades em curto prazo de tempo e eficiente em parâmetros de segurança, velocidade e confiabilidade de dados.

Neste sentido, o presente estudo tem por objetivo fazer um levantamento das produções científicas acerca temas relativos à organização de códigos, testabilidade de módulos, práticas de arquitetura e princípios de desenvolvimento de software tais como: SOLID, e Clean Architecture em aplicação mobile no portal de periódicos da Capes, no período de 2012 a 2022, a fim de identificar lacunas de pesquisa, bem como contribuir com reflexões sobre a temática, por meio do método bibliométrico.

2. Design de Sistema

Princípios SOLID

O Clean Architecture tem como base os princípios de design do SOLID, nas quais, orienta a como organizar as funções e estruturas de dados em classes e como estas devem ser interconectadas. Dentre os principais objetivos da abordagem do SOLID estão: tolerância a mudanças; facilidade no entendimento por parte do grupo de desenvolvimento e padronização de componentes para aplicação em múltiplos sistemas (Martin, 2020). SOLID é um acrônimo para cada um dos cinco princípios que fazem parte desse grupo, exposto e descrito no Quadro 1.

Quadro 1 - Definições dos princípios que compõem o SOLID.

Nome	Definição
Single Responsibility Principle (Princípio da Responsabilidade Única)	O princípio de responsabilidade única define que uma classe deve ter apenas uma única responsabilidade perante um ator, ou seja, deve conter funções específicas e limitadas que atendam a demanda gerada pela funcionalidade (Chebanyuk & Markov, 2016).
Open/Closed Principle (Princípio do Aberto/Fechado)	O princípio do aberto/fechado defende a ideia de que as classes devem ser abertas para extensão e fechadas para modificação, ou seja, devem ser extensíveis sem que isso altere a respectiva classe (Chebanyuk & Markov, 2016; Martin, 2020).
Liskov Substitution Principle (Princípio da Substituição de Liskov)	De acordo com Martin (2020) o princípio da substituição de Liskov define que "para cada objeto de tipo o1 de tipo S, houver um objeto o2 do tipo T, de modo que, para todos os programas P definidos em termos de T, o comportamento de P não seja modificado quando o1 for substituído por o2, então S é um subtipo de T". Ou seja, este princípio diz que toda classe que estende uma classe-pai pode ser substituída por outra classe que estende a mesma classe-pai.
Interface Segregation Principle (Princípio da Segregação de Interfaces)	Conforme a definição do princípio de segregação de interfaces, classes que implementam interfaces não devem conter métodos que não serão utilizados. Este princípio auxilia na criação de interfaces específicas para cada contexto, evitando dependências desnecessárias.
Dependency Inversion Principle (Princípio da Inversão de Dependências)	O princípio da inversão de dependências sugere que as dependências de código-fonte se referem apenas a abstrações e não a itens concretos (Martin, 2020). Assim, é possível fazer alterações e expansões de código com mais facilidade.

Fonte: Elaborado pelos autores (2022).

Arquitetura de *Software*

O projeto de arquitetura de sistema é um processo criativo, no qual, o time de desenvolvimento de software projeta uma estrutura de sistema para satisfazer requisitos funcionais e não funcionais de um sistema (Sommerville, 2011). É na fase de arquitetura que o sistema ganha forma pelos desenvolvedores. Esta forma está na divisão desse sistema em componentes, na organização destes e no modo em que comunicam entre si (Martin, 2020).

Dentre os aspectos essenciais levantados na fase de projeto arquitetônico estão: i) o desempenho do sistema: parâmetro necessário para definir a velocidade das operações e a eficiência de entrega dentre os subsistemas presentes no sistema principal; ii) proteção de dados: para uma melhor proteção de dados em sistemas cujo a segurança seja prioridade, há a necessidade de adotar uma arquitetura baseada em camadas para proteger os ativos mais críticos que serão protegidos por camadas mais internas, com alto nível de validação (Sommerville, 2011); iii) manutenção: caso a manutenção de um sistema seja prioritário a longo prazo, é necessário adotar uma arquitetura que seja de fácil modificação, ou seja, que sejam independentes de outros componentes e que sejam responsáveis por tarefas limitadas (Pressman, 2011).

Dentre os parâmetros mencionados, o padrão de arquitetura em camadas é o mais compatível com as características necessárias para desempenho, proteção e manutenção a longo prazo. Esta abordagem apoia o desenvolvimento incremental de sistemas, ou seja, quando uma camada é desenvolvida esta já pode ser disponibilizada aos usuários. Além disso, este modelo é mutável e portável assim quando a camada de interface se altera ou novos recursos são adicionados, apenas a camada adjacente é afetada (Sommerville, 2011). Este conceito de arquitetura em camadas é compatível com os princípios de design do aberto/fechado e de responsabilidade única expostos pela abordagem SOLID e abre possibilidades para sua implementação em aplicações que crescem constantemente, como o ambiente de desenvolvimento mobile, que possui um ciclo de vida amplo e alto custo em sua produção. Este modelo arquitetural pode diminuir o custo da vida útil do sistema e não maximizar a produtividade do programador (Martin, 2020). O Quadro 2 mostra as vantagens e as desvantagens da utilização de uma arquitetura em camadas.

Quadro 2 - Vantagens e desvantagens da utilização do padrão de arquitetura em camadas.

Vantagens	Desvantagens
Possibilita trabalhar em níveis crescentes de abstração.	Alguns sistemas e/ou plataformas não se adequam a organização em camadas.
Suporta facilmente a aplicação de melhorias, correções e expansões.	Requisitos de desempenho podem levar a quebra das regras de organização em camadas.
Suportam o reuso de componentes de *software*.	Pode haver duplicação de funcionalidade e impedimentos em estruturar um sistema através de camadas.
Facilidade de compreensão, leitura, manutenção e desenvolvimento independente de funcionalidades.	*Overhead* de implementação e desempenho.

Fonte: Dantas et al. (2021, p.4).

Clean Architecture

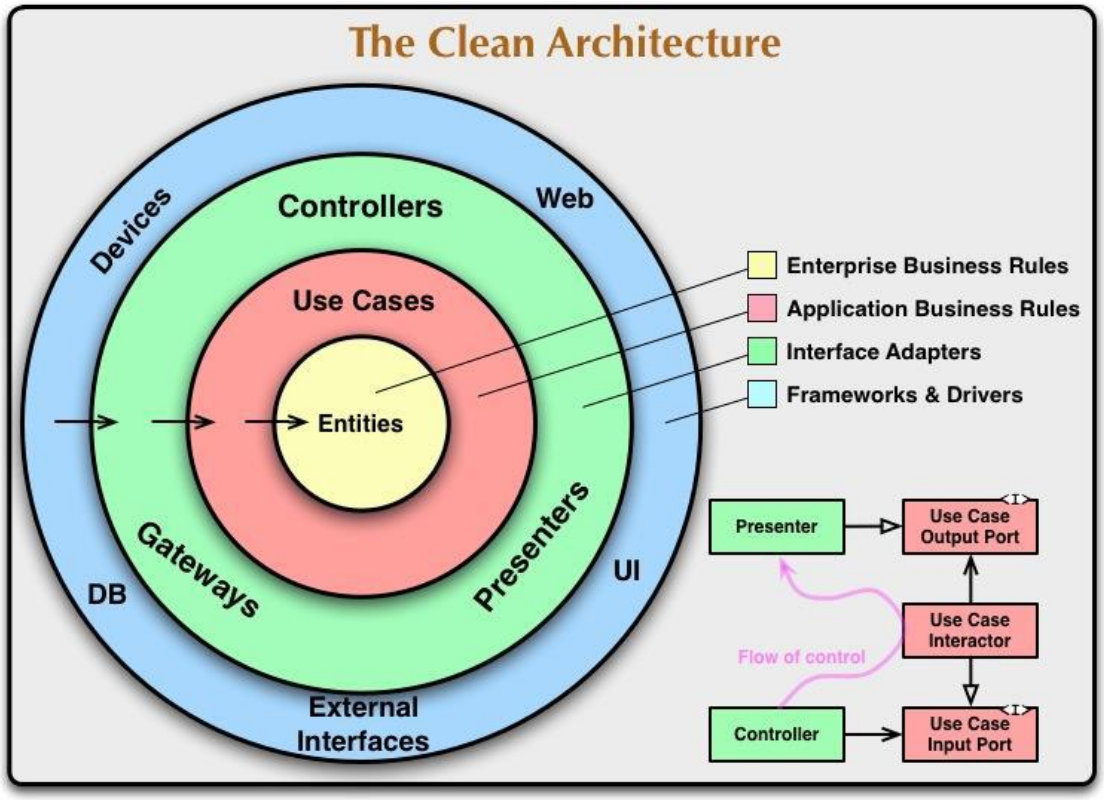
O Clean Architecture ou Arquitetura Limpa foi um modelo de organização e construção de sistemas proposta pelo blog The Clean Code Blog (Uncle Bob, 2012) escrita pelo engenheiro de software Robert C. Martin - também um dos autores do Manifesto Ágil - no ano de 2012. O objetivo da Clean Architecture, como abordagem de composição de sistemas, é a de organizar e estruturar códigos de programação para facilitar o desenvolvimento, implantação, manutenção e integração de regras de negócios empresariais (Martin, 2020). Assim sendo, a aplicação da Clean Architecture possibilita a separação das camadas do software onde cada camada fica independente de outras partes, sendo possível assim ter uma arquitetura de software limpa, testável, escalável e manutenível.

A arquitetura limpa propõe uma camada a nível mais externo que se separa das camadas mais internas do software, assim, os frameworks podem ser utilizados, embora sejam facilmente substituídos. Através deste conceito é possível realizar a separação da camada de negócios de dependências externas, seguindo assim o princípio de responsabilidade única do SOLID. Nesse contexto pode-se usar abstrações, onde através da utilização do design pattern de injeção de dependência, é possível a

implementação de interfaces sem a necessidade de um código concreto. Dessa maneira os frameworks podem ser utilizados como ferramenta de construção, ao invés de se tornar regra de negócio principal do projeto (Boukhary & Colmenares, 2019).

No modelo da Clean Architecture as respectivas camadas de regras de negócios são independentes da interface de usuário, bancos de dados e quaisquer ferramentas externas, seguindo a independência e proteção das regras de domínio. Através deste é possível testar cada camada sem mesmo existir implementação em códigos das camadas mais externas. Sendo, portanto, possível garantir que as regras de domínio, ou seja, as regras de negócio do sistema não sejam modificadas diretamente. Ademais aplica-se a abordagem do Test Driven Development (TDD) na qual a codificação das funcionalidades do sistema começa a partir da escrita de testes unitários. Este será responsável por manter as responsabilidades de código coeso, funcional e solucionar problemas de dependência e acoplamento entre as camadas da Clean Architecture (Dantas, et al., 2021). Essa técnica foi criada por Kent Beck, sendo um dos pilares do Extreme Programming (XP). Com essa técnica é possível que antes mesmo da construção de um banco de dados seja realizado uma implementação total de uma funcionalidade, garantindo assim os testes das camadas internas de domínio e a manutenção a longo prazo. A Figura 1, apresenta as camadas da arquitetura limpa, na qual as entidades não possuem relação de dependência com as camadas externas a ela.

Figura 1 - As camadas da Clean Architecture.



Fonte: Uncle Bob (2019, s.p.)

Regra da dependência

Os círculos representam diferentes áreas de software: UI, Web, bancos de dados etc. A regra de substituição que faz essa arquitetura funcionar é a Regra de Dependência. Essa regra diz que as dependências do código-fonte só podem apontar para dentro. Nada declarado em uma camada externa pode ser observado pelas camadas internas. Isso inclui: funções; classes; variáveis ou qualquer outra entidade de software nomeada. Isso garante um desacoplamento entre as camadas e independência

de cada uma delas. Da mesma forma, os formatos de dados usados em um círculo externo não devem ser usados por um círculo interno, especialmente se esses formatos forem gerados por uma estrutura em um círculo externo.

Entidades

Os círculos mais internos representam as entidades comumente chamadas por camada de domínio, é a camada mais interna que não conhece as camadas mais externas e é conhecida e usada por todas as camadas. As entidades encapsulam regras de negócio de toda empresa. Por sua vez, elas são menos propensas a mudar quando alguma regra externa da aplicação se altera.

Casos de Uso

A camada de Casos de Uso possui as regras existentes na aplicação e os casos de uso que existem na camada de Presenter. Nela são encapsulados e implementados todos os casos de uso do sistema. Os casos de uso são os que regem o fluxo dos dados das entidades para atingir as metas dos casos de uso. Mudanças nessa camada não devem afetar as entidades. E mudanças externas também não podem acabar afetando esta camada, ela deve ser uma camada totalmente independente que é usada pelas camadas externas.

Arquitetura *Mobile*

Os smartphones e dispositivos móveis modernos possuem uma grande capacidade computacional, nos quais, desenvolvedores possuem a liberdade para executar tarefas complexas no dispositivo de maneira local ou utilizando a computação em nuvem. Porém, mesmo aplicativos que utilizem um ambiente de processamento em nuvem, ainda é preciso uma arquitetura que permita a flexibilidade na mudança de serviços e a possibilidade de testar dependências externas (Nunkesser, 2021). Isso mostra que a arquitetura possui grande relevância para o desenvolvimento de software mobile moderno e muitas ideias desenvolvidas pioneiramente nestas áreas, criadas durante a virada do século, encontraram destaque no contexto do mobile software (Bagheri, et al., 2016).

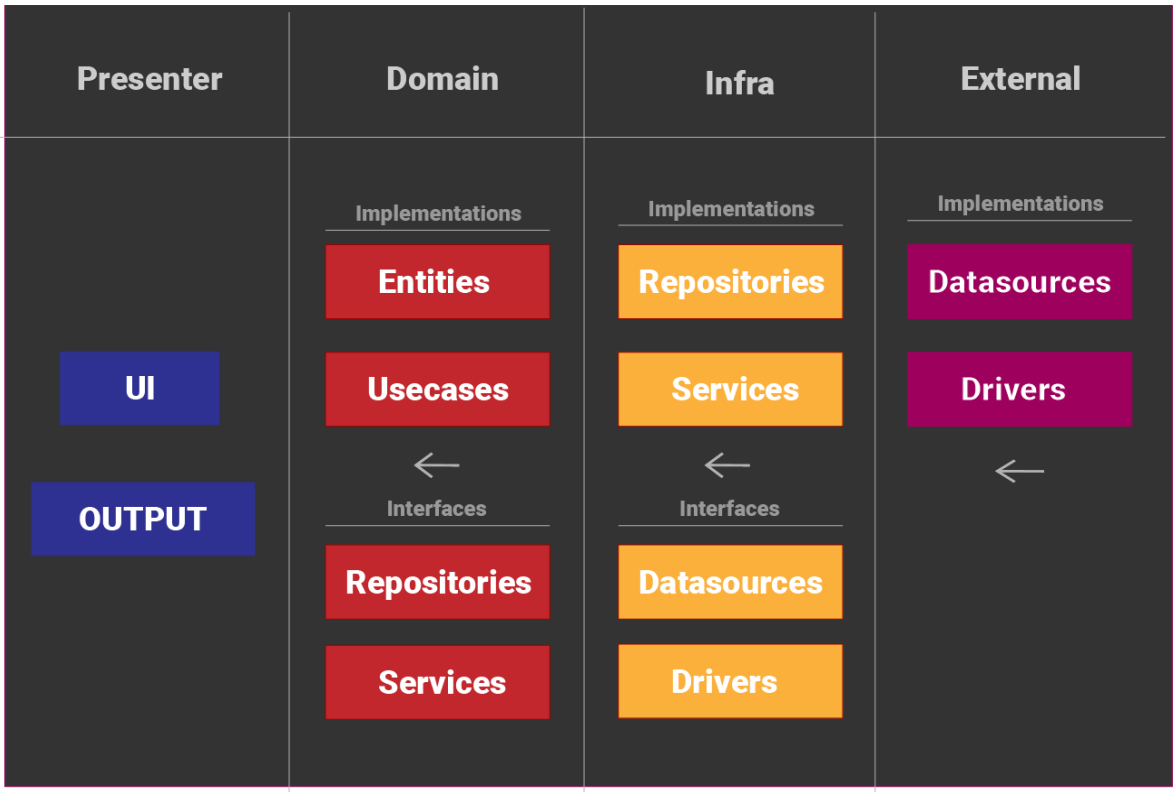
Devido à alta demanda de serviços que utilizam plataformas mobile e o alto número de usuários que interagem com dispositivos móveis, é necessário que seja desenvolvido uma abordagem que atenda requisitos específicos de engenharia de aplicações mobile, tais como: i) criação de modelo de interface de usuário que se adapte a uma variedade de tamanhos de tela de aparelhos móveis presentes no mercado; ii) Adaptação de serviços aos sensores, componentes e ao público-alvo das aplicações; iii) Gestão de armazenamento de dados locais, sincronia com banco de dados em nuvem e disponibilidade de dados em ambientes sem conexão à internet; iv) Necessidade de código eficiente para gestão de memória com o objetivo de economia de bateria.

Para cobrir estes aspectos para criação e manutenção do ciclo de vida a longo prazo, o padrão arquitetural em camadas fornece as abordagens corretas para expansão de funcionalidades e correções de erros. Dentre os aspectos deste padrão, que se adaptam ao ambiente de aplicações mobile, destacam-se as:

- Independências de frameworks: Este aspecto se define como a separação de responsabilidades em contexto de regras de negócio para facilitar a flexibilidade a mudanças e a realização de testes (Ivanics, 2016). Além disso, através deste conceito, é possível com apenas uma base de código a execução em múltiplos ambientes de plataforma e dispositivos de hardware. Dentre as plataformas disponíveis no mercado, destaca-se o framework Flutter no qual é uma ferramenta open-source para desenvolvimento de software criado pela Google (Wikipédia, 2022). Baseado na proposta do Clean Architecture, a comunidade de desenvolvedores brasileira Flutterando criou uma proposta de arquitetura limpa chamada Clean Dart, presente na figura 2, que ressalta características principais deste modelo de arquitetura em camadas. Dentre as camadas estão caracterizadas os módulos de: i) Presenter: nos quais são responsáveis pela interface de usuário do sistema e de seus respectivos controladores para conexão com a próxima camada; ii) Domain: Responsável

pelas regras de domínio do projeto, onde estão ressaltados as entidades de classes e as usecases, nos quais, estão expostos os atores e as funcionalidades que estes interagem; iii) Infra: A camada de Infra é responsável pela adaptação dos dados externos para o aplicativo, sendo responsável por implementar as interfaces levantadas na camadas de Domain. Toda a comunicação obedece aos princípios do Aberto/Fechado promovido pela abordagem SOLID; iv) External: Esta camada fica responsável pela conexão a base de dados, dispositivos e comunicações externas ao projeto, tais como hardwares, packages ou acessos específicos tais como banco de dados e servidores.

Figura 2 - As camadas da Clean Dart, proposta baseado na *Clean Architecture*.



Fonte: Flutterando (2021, s.p.)

- Testabilidade: Com o aspecto de independência de frameworks implementado no projeto é possível realizar os testes em cada camada separadamente. Quando as camadas internas são testadas, as dependências podem ser testadas com dados de testes, ou seja, são dados não reais, porém, compatíveis com os tipos de variáveis. Assim é possível fazer testes confiáveis independente de implementação (Boukhary & Colmenares, 2019). Um dos testes que beneficiam a confiabilidade do projeto, de acordo com Beck (2002, como citado por Nunkesser, 2021) é o teste unitário, no qual possibilita rápido feedback de funcionamento no dispositivo em que o sistema está sendo executado.

3. Metodologia

O presente estudo possui caráter exploratório-descritivo, e abordagem quantitativa e qualitativa (método misto), face a compreensão do fenômeno analisado em termos qualitativos, combinado com a mensuração dos resultados, decorrentes da identificação e avaliação as produções científicas (artigos publicados nos idiomas: inglês e português) sobre: SOLID e Clean Architecture. Para tanto, realizou-se uma análise bibliométrica conforme adotado por Azevedo (2018), Sousa Júnior et al., (2020), Pereira Júnior *et al.* (2021). Optou-se pela base de dados multidisciplinar Periódicos Capes, com corte temporal definido

entre 2012 e 2022, ao considerar ser uma abordagem recente, cujos trabalhos voltados para os conceitos de arquitetura limpa iniciaram em 2011, conforme postulado pelo idealizador e criador desta abordagem Robert C. Martin. Adotou-se o banco de dados Periódicos Capes por contemplar diversas áreas de conhecimento e indexação de produção científica com reconhecimento internacional e nacional, além de disponibilizar um conjunto de metadados necessários para a análise bibliométrica.

No modelo de pesquisa bibliométrica é necessário que o pesquisador elabore um roteiro e siga cada passo, pois o método de estudo bibliométrico quando seguido em forma de etapas passa a garantir um rigor metodológico, além de entregar credibilidade ao trabalho elaborado. Contudo, é necessário que anterior à escolha do material a ser analisado, os pesquisadores definam uma ou mais bases de pesquisa, que podem ser livros, periódicos, jornais, dentre outras formas de literatura escrita. Após a escolha da base, é necessário fazer buscas prévias para garantir a confibiliadade dos assuntos, pois pode ocorrer de a base inicialmente apresentada como proposta, não apresente eficácia no processo de busca e análise do material encontrado, desta forma será necessário repensar descritores, bases ou até novas fontes de análise, o que pode atrasar o processo de pesquisa (Azevedo, 2018). A sequência das etapas metodológicas é descrita no Quadro 3.

Quadro 3 - Etapas metodológicas do processo de pesquisa bibliométrica.

1º	Definição do escopo da pesquisa
2º	Definição da Base de Dados a ser consultada
3º	Aplicação dos Termos de Busca
4º	Análise Prévia do Material Encontrado: Leitura do Título e Resumo
5º	Exclusão do Material Não Enquadrado
6º	Análise do Material Alinhado ao Objetivo da Pesquisa
7º	Discussões e Resultados

Fonte: Elaborado pelos autores (2022).

Para elaboração da revisão bibliográfica da literatura, utilizou-se uma sequência de fases, estruturadas a saber: i) aplicação de termos de busca relacionados a *Clean Architecture* e SOLID em contexto de construção de *software;* ii) artigos; iii) revisado por pares.

4. Resultados

A análise bibliométrica realizada, compreendida pelo recorte temporal dos anos de 2012 a 2022, apresentou um total de 27 artigos científicos a partir da aplicação dos termos de busca: clean architecture, mobile e software. Após a leitura inicial do resumo e título, constatou-se que destes, quatro estão alinhados ao escopo do estudo, contemplando os aspectos relacionados ao conceito de arquitetura limpa e o desenvolvimento de aplicações mobile utilizando a abordagem de SOLID, face a relação de complementaridade existentes entre os princípios de design de software e arquitetura. Dentre os trabalhos selecionados, o idioma de publicação foi em língua inglesa, tendo sido divulgado em revistas com Qualis Capes A2 a B2 na área de pesquisa em Ciências da Computação nos seguintes periódicos: i) International Journal of Computer Science; ii) Future Internet Journal; iii) International Journal of Computer Science; iv) Journal of Software: Practice and Experience.

Em termos cronológicos as publicações iniciam em 2016, com o estudo de Chauhan, Babar e Benatallah, intitulado: "Architecting cloud-enabled systems: a systematic survey of challenges and solutions", que consistem em uma revisão sistemática de literatura que versa sobre a criação de arquiteturas de software voltado para o armazenamento e processamento

de dados em ambiente de cloud, onde as camadas de desenvolvimento seguem os princípios de SOLID para manutenção e expansão, além de processamentos de dados em grande escala e sob demanda. Os autores identificaram e revisaram um total de 133 artigos cujos conteúdos são voltados para soluções de arquitetura baseadas em cloud. Através da análise do levantamento dos trabalhos, definiram 44 categorias de desafios e suas respectivas soluções relacionadas a sistemas de software baseado em nuvens.

Dentre os atributos qualitativos levantados e especificados nos desafios para a criação de um modelo arquitetônico para ambientes cloud estão as formas de processamento em tempo de design (interoperabilidade, privacidade, portabilidade, segurança e consistência) e em tempo de execução (disponibilidade, escalabilidade, elasticidade, performance e adaptabilidade). Cada um destes atributos pertence a uma parte de um sistema, nos quais, devem ser conduzidos individualmente por uma arquitetura que comporte características relativas à sua função. Os autores levantam desafios relacionados à arquitetura em ambientes cloud voltados para o desenvolvimento mobile, no quais, possibilitam serviços de computação em larga escala para usuários que usam dispositivos com capacidade de processamento e armazenamento limitada como smartphones. As soluções para estes desafios, segundo o resultado que os autores descobriram, era a aplicação de uma arquitetura em camadas cujas responsabilidades sejam divididas para cada função específica e que cada camada possa ser expandida e modificada, obedecendo os princípios do Aberto/Fechado da abordagem do SOLID.

Por sua vez, o artigo de Moghaddam, et al., (2017) intitulado "Empirical validation of cyber-foraging architectural tactics for surrogate provisioning", apresentam uma abordagem arquitetônica baseada no cyber-foraging, que segundo os autores, é um mecanismo que aproveita servidores cloud ou locais para aumentar o processamento e armazenamento de dispositivos móveis com recursos limitados com o objetivo de estender a vida útil da bateria. Este modelo específico possui duas principais formas: computation offload e data staging. A primeira refere-se a transferência de processamento para uma plataforma externa, que neste caso, seria um ambiente de processamento em nuvem. A segunda refere-se a preparação e adaptação de dados para transferência entre dispositivos móveis e um ambiente externo. A Clean Architecture oferece em sua camada de Frameworks e Drivers uma adaptação de dados externos para transferência a suas camadas interiores e vice-versa, ficando responsável pela adaptação de dados e flexibilidade de testes devido a seu modelo de comunicação através de abstrações. No trabalho os autores referem a eficácia do método chamado de Static Surrogate Provisioning, no qual por meio da arquitetura, o aplicativo mobile decide se escolhe uma execução remota ou local de processamento de dados utilizando algoritmos e camadas de execução para verificar a disponibilidade, a relevância e a demanda da informação a ser processada, simplificando assim o processo de deploy. Essa abordagem é vantajosa para aplicações de baixa demanda operacionais de processamento computacional.

Dentre a estrutura de arquitetura cloud voltado para mobile, o trabalho de Alreshidi, et al., (2019) intitulado "Software Architecture for Mobile Cloud Computing Systems", consiste em uma revisão sistemática de literatura identificando abordagens com foco em construção de ambiente de computação e processamento mobile com o intuito de portabilizar a computação cloud através de arquiteturas que auxiliam na abstração e desenvolvimento do design do sistema. O estudo foi baseado em 121 artigos selecionados em revistas de pesquisas científicas publicadas entre o intervalo dos anos de 2006 e 2019. O foco da pesquisa foi na identificação de arquiteturas em cloud que suportem: i) software como serviço para dispositivos móveis; ii) transferência de dados disponíveis localmente para servidores em nuvem; iii) internet das coisas; iv) computação de aspectos relacionados à privacidade e segurança de dados de dispositivos móveis. Os pesquisadores tomaram como base três termos para o estudo: computação móvel, computação em nuvem e computação cloud focado em mobile. Durante o estudo foram encontrados desafios relacionados à arquitetura em modelos de aplicação para o desenvolvimento para plataformas móveis, cuja pauta principal é como alcançar a eficiência computacional mantendo a portabilidade e o desempenho de um dispositivo móvel com recursos limitados. Dentre as soluções encontradas estão a implementação de arquitetura baseadas em camadas cuja responsabilidades sejam separadas para manter a organização e flexibilidade de mudanças, seguindo assim, o princípio da responsabilidade única

da abordagem SOLID. Os dispositivos móveis representam, como camada front-end, representam uma interface de usuário com as devidas views e componentes. Por outro lado, servidores baseados em nuvem, como back-end, disponibilizam softwares de processamento e armazenamento para os respectivos dispositivos clientes.

Em relação ao trabalho de Sanchez, et al., (2022) intitulado "Towards a Clean Architecture for Android Apps using Model Transformations" foi pesquisado a relação entre a demanda de expansão, escalabilidade e testabilidade no mercado de aplicativos e o conceito de arquitetura limpa proposto por Robert C. Martin na plataforma móvel Android. Os autores propuseram a criação de um padrão para geração de código automática que será executado no sistema Android com a linguagem de programação Kotlin. Este padrão foi chamado de "Model Transformation", no qual, consiste na criação de um metamodelo contendo atributos necessários para a criação de aplicativos tais como regras de negócios e requisitos funcionais. Então através desta criação é possível gerar código de execução específico para o sistema operacional. Como resultado o código gerado vai de acordo com as regras de implementação e camadas da Clean Architecture, dessa maneira, o processo de separação de componentes e implementação das abstrações para comunicação torna-se automatizado.

5. Discussão

Dentre os trabalhos selecionados, o foco está no desenvolvimento de aplicações mobile direcionado para o processamento e armazenamento em cloud, nos quais há desafios relacionados a arquiteturas que entregam mais eficácia na transferência e comunicação de dados. Ao mesmo tempo, que se adaptem aos princípios de SOLID e que estabeleçam vantagens para dispositivos que consomem o processamento, tais como: velocidade de transferência, gestão de memória e economia de energia.

Apesar de seguir os princípios de SOLID para a flexibilidade de criação de testes e separação de responsabilidades utilizando os princípios de Responsabilidade Única e do Aberto/Fechado, inerentes à organização e comunicação entre as camadas, não sendo identificado nos resultados da pesquisa uma abordagem integrada de *Clean Architecture* e SOLID, visto a interdependência existente entre os conceitos de arquitetura e *design*.

6. Considerações Finais

O presente trabalho se propôs a elaborar uma análise da produção científica sobre Clean Architecture e SOLID, disponível na base de dados Periódico Capes, entre os anos de 2012 e 2022. Para tanto, adotou-se a ferramenta de análise bibliométrica como método de pesquisa e identificação do estado da arte acerca do tema. Os resultados sinalizam uma carência de produções científicas que contemplem o uso de abordagens arquitetônicas baseadas em Clean Architecture em plataformas mobile, em um contexto prático, posto que dos trabalhos selecionados dois são artigos de revisão.

Neste sentido, considerando o método de desenvolvimento do Clean Architecture postulado por Robert C. Martin, ainda ser recente e a comunidade acadêmica e empresarial de desenvolvedores de software ainda estar em fase inicial de adoção destes respectivos métodos e suas aplicações, faz-se necessário mais estudos acerca da aplicação da arquitetura em ambientes mobile que envolvam grande demanda e consumo de dados em uma base de usuários e dados extensas que sejam passíveis de expansão. Além disso, o resultado do estudo apresenta limitações por se tratar da pesquisa em apenas uma base de dados em específico. Sugerindo que haja a exploração em base de dados e periódicos ligadas a área de Sistemas de Informação, Engenharia de Software e Ciências da Computação.

Referências

Alreshidi, A., Ahmad, A., Altamimi, A. B., Sultan, K., & Mehmood, R. (2019). Software Architecture for Mobile Cloud Computing Systems. *Future Internet.* 11(238). 1-35.

Azevedo, I. M. (2018). Análise e Aplicação Bibliométrica na Administração e Áreas Afins: Um Levantamento Nacional. *Interscientia*. 6 (2), 146-164.

Bagheri, H., Garcia, J., Sadeghi, A., Malek, S., & Medvidović, N. (2016). Software architectural principles in contemporary mobile software: from conception to practice. *Journal Systems Software*, 119, 31-44.

Beck, K. (2002*). Test Driven Development: By Example.* Boston, MA: Addison - Wesley Professional.

Boukhary, S., & Colmenares, E. (2019). A Clean Approach to Flutter Development through the Flutter Clean Architecture Package. *International Conference on Computational Science and Computational Intelligence*. 1115-1120.

Chauhan, M. A., Babar, M. A., & Benatallah, B. (2016). Architecting cloud-enabled systems: a systematic survey of challenges and solutions. *Software Practice and Experience*. 47. 599– 644.

Chebanyuk, E., & Markov, K. (2016). An approach to class diagrams verification according to SOLID design principles. 4th *International Conference on Model-Driven Engineering and Software Development*. 435-441.

Dantas, C., Casillo, L. A., & Neto, F. M. M. (2021). Uma proposta de arquitetura de software limpa baseada em microsserviços.

Ivanics, P. (2016). An Introduction to Clean Software Architecture. *Department of Computer Science*, University of Helsinki.

Flutterando. (2021*). Clean Dart*. https://github.com/Flutterando/Clean-Dart.

Flutter (software). (2022). Wikipédia. https://en.wikipedia.org/wiki/Flutter_(software)

Kim, G., Humble, J., Debois, P., & Willis, J. (2018). *Manual de DevOps: Como Obter Agilidade, Confiabilidade e Segurança em Organizações Tecnológicas.* (1a ed) Rio de Janeiro, RJ: Alta Books Editora.

Martin, R. C. (2020*). Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software.* Rio de Janeiro, RJ: Alta Books.

Moghaddam, F. A., Procaccianti, G., Lewis, G. A., & Lago, P. (2017). Empirical validation of cyber-foraging architectural tactics for surrogate provisioning. *The Journal of Systems and Software*. 138. 37-51.

Moreno, V., Cavazotte, F., & De Farias, E. (2009). Novos Desafios para o Profissional de TI: Estudo de Caso de uma Empresa de Prestação de Serviços de Tecnologia da Informação**.** *Revista de Gestão da Tecnologia e Sistemas de Informação*. 6. 437-462.

Nunkesser, R. (2021). Choosing a Global Architecture for Mobile Applications. 10.36227/techrxiv.14212571.v1

Pereira Junior, E., D'Avila, L., da Cruz, A., & Amaro, R. (2021). Do Que Se Constitui Um Empreendedor? Panorama Da Produção Científica Mundial Sobre O Background Do Empreendedor. *Revista Da Micro E Pequena Empresa, 15*(1), 3-23. https://doi.org/10.6034/rmpe.v15i1.1576

Pressman, R. S. (2011). *Engenharia de Software: Uma Abordagem Profissional.* (7a ed.) Porto Alegre, RS: AMGH.

Sanchez, D., Rojas, A. E., & Florez, H. (2022). Towards a Clean Architecture for Android App using Model Transformations. *IAENG International Journal of Computer Science*. 49(1). 270-278.

Sommerville, I. *Engenharia de Software*. (9a ed). São Paulo, SP: Pearson Prentice Hall.

Sousa Júnior, J. C., Rocha, F. R. T., & Coelho, K. O. (2020). Análise bibliométrica em frango e frango caipira/colonial. *Pesquisa, Sociedade e Desenvolvimento, 9* (8), e773986354. https://doi.org/10.33448/rsd-v9i8.6354.

Uncle Bob. (2019). The Clean Coder Blog - The Clean Architecture. https://blog.cleancoder.com/uncle-bob/2012/08/13/ the-clean-architecture.html