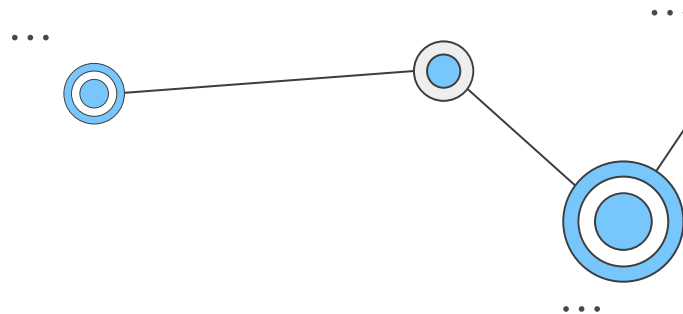


the
developer's
life **WEEKEND**

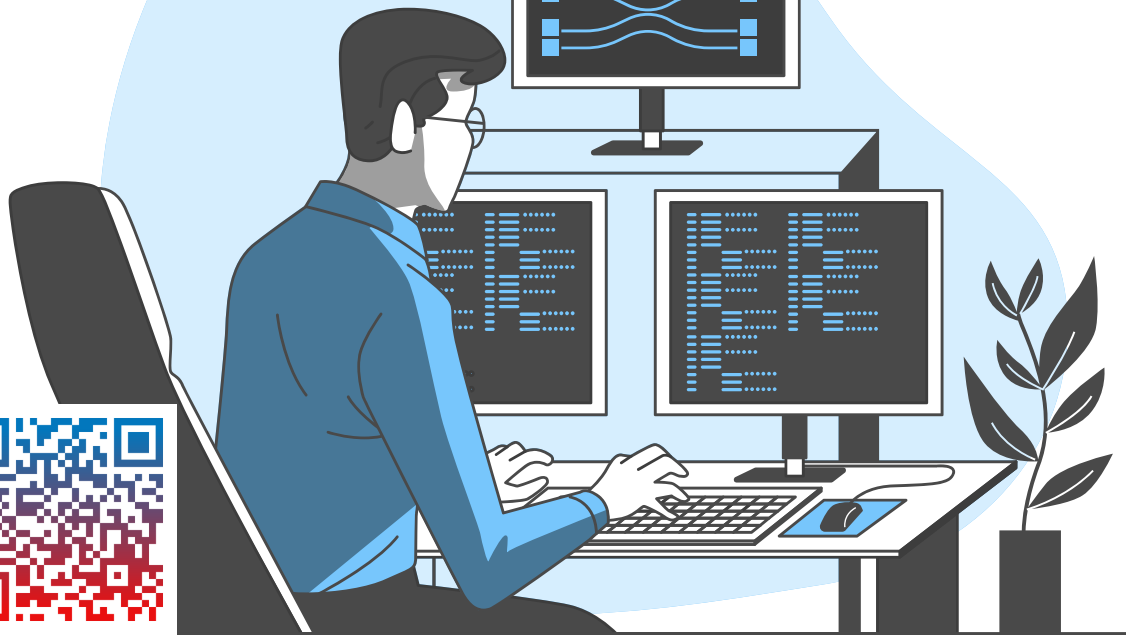


TDD e Clean Architecture no frontend com React

Douglas Nassif Roma Junior



/douglasjunior



Douglas Nassif Roma Junior



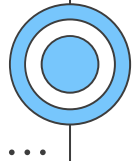
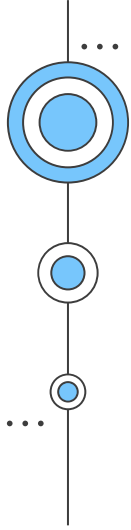
- Desenvolvedor de software desde **2010**
- Formado em Tecnologia em Sistemas para Internet pela UTFPR em **2012**
- Professor universitário desde **2014**
- Fundador dar Smarppy (Lughy) em **2017**
- Staff Software Engineer na DB1 **atualmente**



Se só fizer o que sabe,
nunca será mais do que é agora.

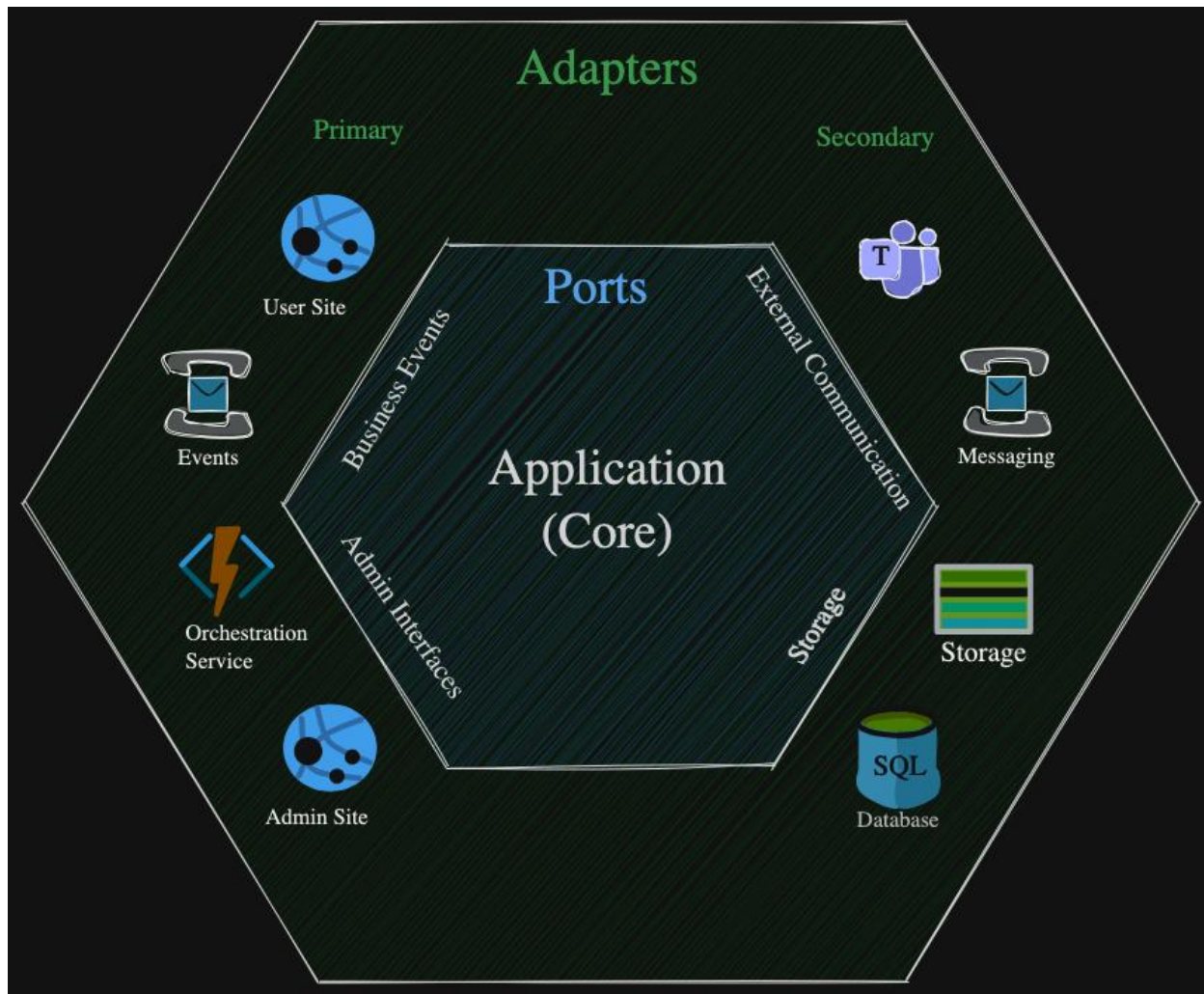
Mestre Shifu – Kung Fu Panda 3

Hexagonal Architecture

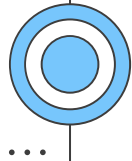
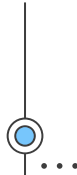
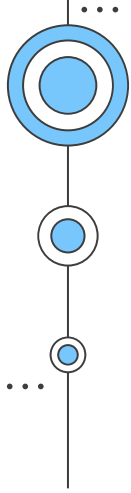


Hexagonal Architecture

- Não tem como falar de Arquitetura Limpa (Clean Architecture) sem falar de Arquitetura Hexagonal (Hexagonal Architecture).
- Segundo Alistair Cockburn (autor), Arquitetura Hexagonal permite que sua aplicação:
 - **Resources**: seja desenvolvida em isolamento de APIs externas, banco de dados, file system etc.
 - **Drivers**: seja guiada em isolamento de telas, chamadas de API, teste etc.



Clean Architecture



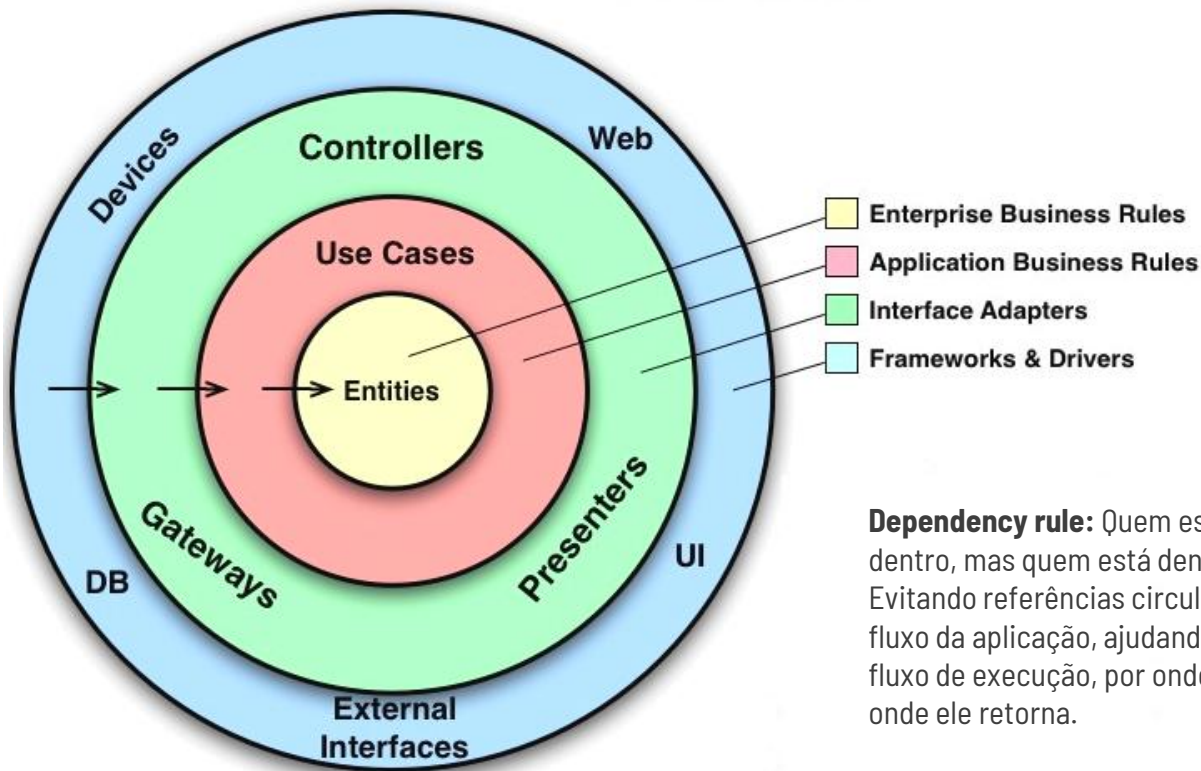
Clean Architecture

- A Clean Arch é um modelo que tem como objetivo o **desacoplamento entre as regras de negócio, ou domínio, da aplicação e os recursos externos como frameworks e banco de dados.**
- De certa forma, a implementação de **Clean Architecture traz também as ideias da Arquitetura Hexagonal**, porém com uma pitada a mais.

Clean Architecture

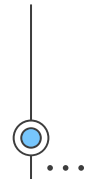
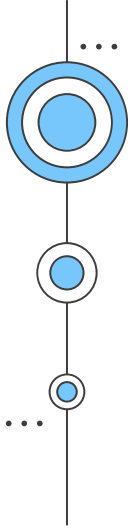
- Segundo Robert Martin (autor), o centro de sua aplicação não é o banco de dados ou framework que você utiliza. **O centro da sua aplicação são os casos de uso (use cases) da sua aplicação.**
- Frameworks tendem a serem opinativos e trazem um tipo de design de software muito específico, então **o que a gente normalmente enxerga primeiro são os frameworks e não o comportamento real da aplicação que estamos desenvolvendo.**

Clean Architecture



Dependency rule: Quem está fora conhece quem está dentro, mas quem está dentro não conhece quem está fora. Evitando referências circulares, facilitando a entender o fluxo da aplicação, ajudando a entender por onde passa esse fluxo de execução, por onde ele vem, até onde ele vai, por onde ele retorna.

Clean Code VS Clean Architecture





Clean Code vs Clean Architecture



- **Clean Code** lida com código:
 - resolve problemas dentro dos métodos da aplicação, ajuda a dar nomes melhores para as coisas, lidar melhor com comentários e linhas em branco, tratar erros, simplificar condições, saber quando e como refatorar.
- **Clean Architecture** lida com design:
 - diz como distribuir o comportamento entre as camadas da aplicação, qual a responsabilidade entre cada uma delas e onde elas deveriam estar.
- Clean Code é dentro e Clean Architecture é fora.

Desafio



<https://github.com/douglasjunior/react-tdd-clean-arch>

Referências

- Hexagonal Architecture Explained: <https://a.co/d/1XA0xZ4>
- Padrões de Arquitetura de Aplicações Corporativas: <https://a.co/d/3LVhKq5>
- Arquitetura Limpa: o Guia do Artesão Para Estrutura e Design de Software: <https://a.co/d/cnaqMbu>



/douglasjunior

