⚡ Clean Architecture in a nutshell

https://github.com/bionic-hillbilly/clean-architecture-lecture

📝 Clean Architecture: where does it come from ?

clean architecture refers to:

- 📝 [an article](#)

- 📖 [a book](#)

written by **Robert C. Martin** aka **Uncle Bob**

🧠 The inspiration

- [Hexagonal Architecture](#)

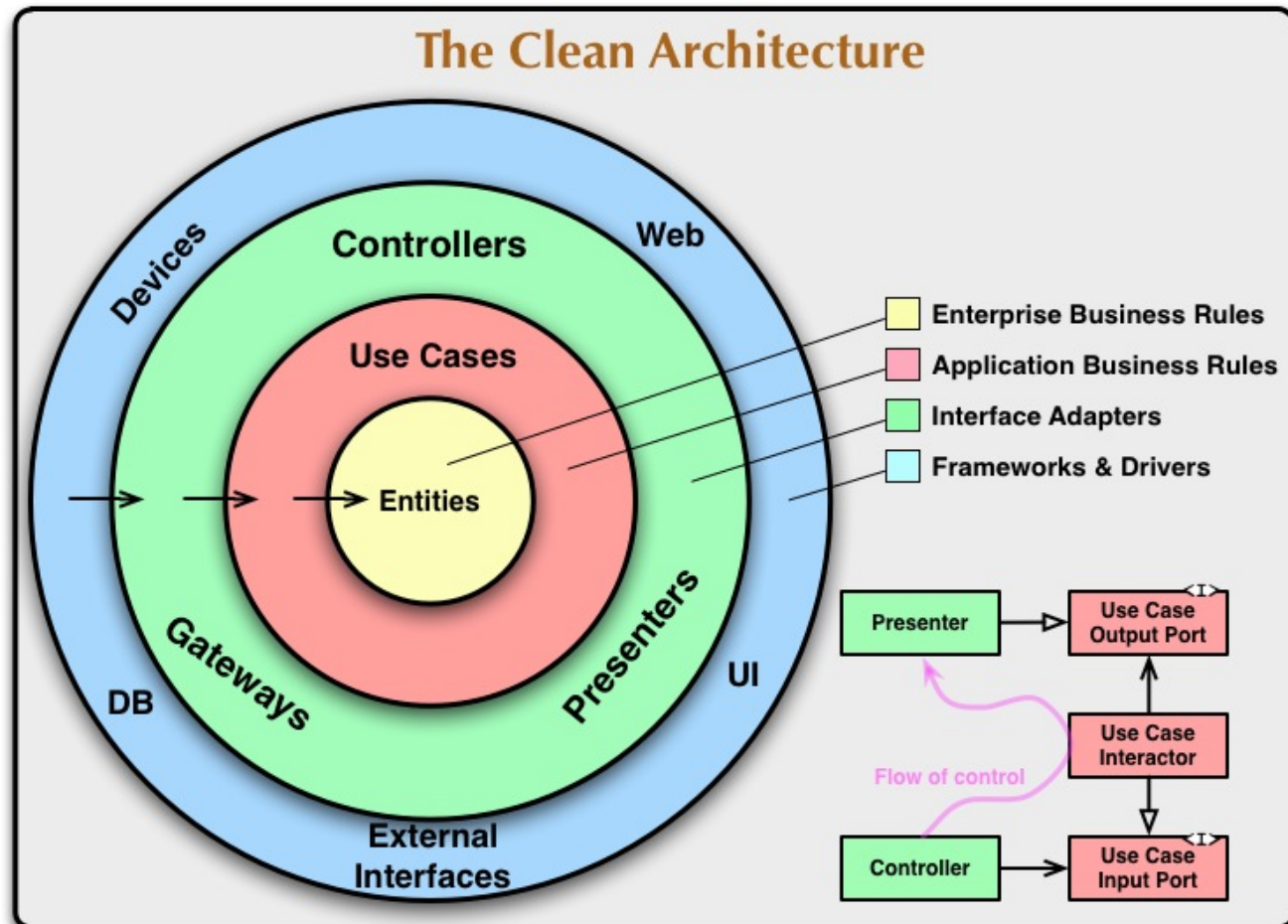- [Onion Architecture](#)

- [Screaming Architecture](#)

🎯 The objectives

Separation of concerns:

- Architecture should be as independent of frameworks as possible
- Business rules should be as testable as possible
- Business rules should be independent of user interface
- Architecture should be independent of the database
- Business rules should be independent of outside world

🎉 The best of both worlds

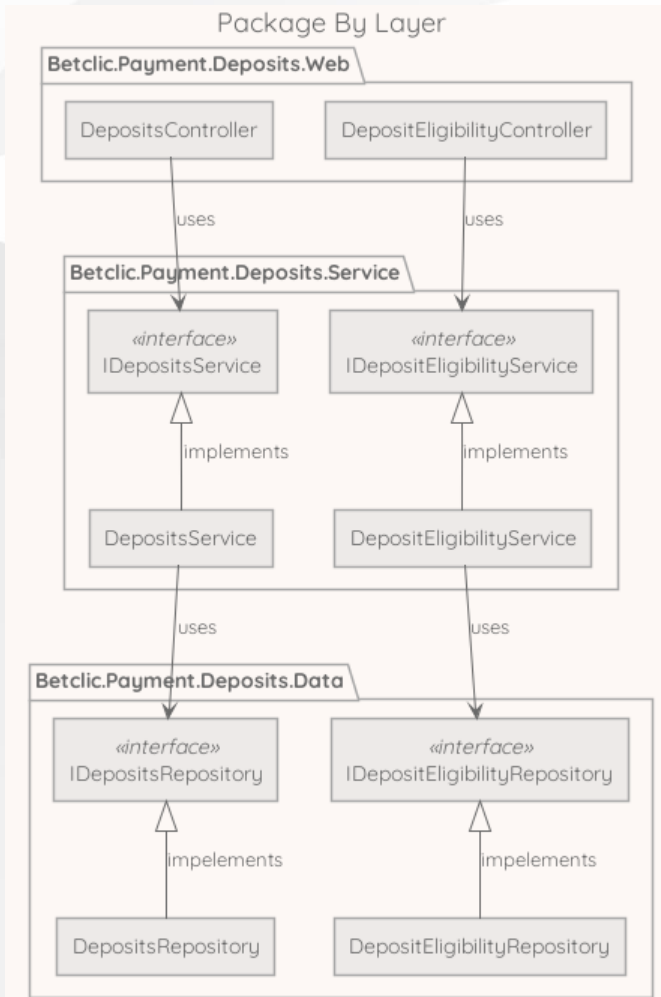The Clean Architecture

❤️ **Benefits**

- built-in separation of concerns

- testability

- screaming use cases

- straightforward dependency rules

- future-proof

⚠️ Constraints

- same discipline across the team
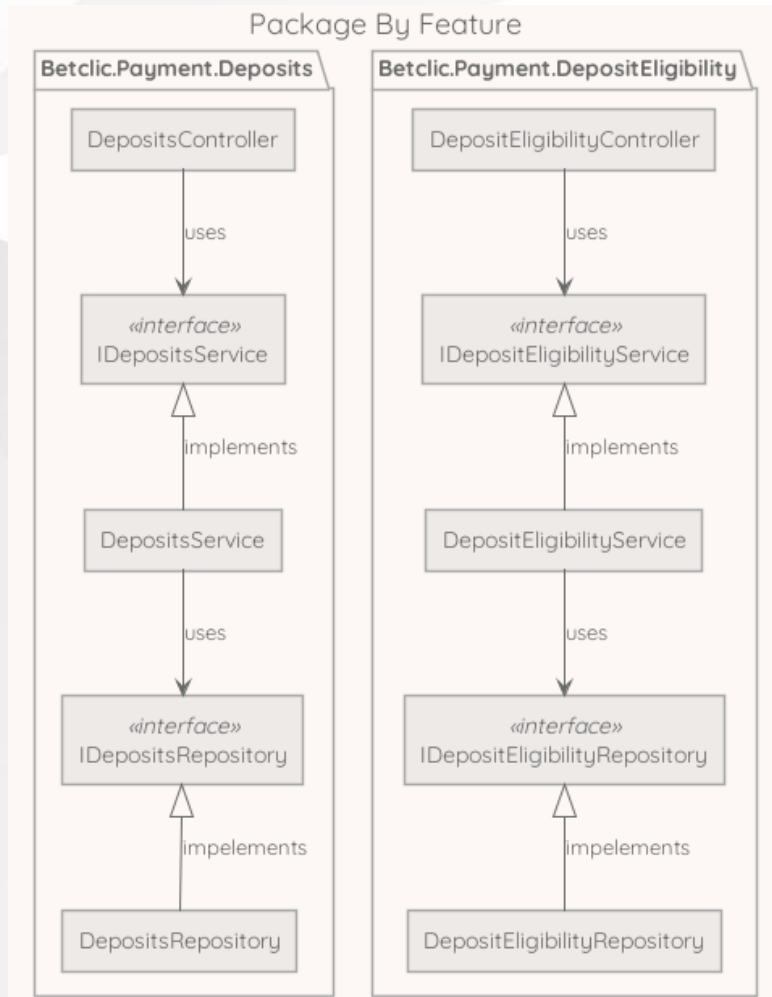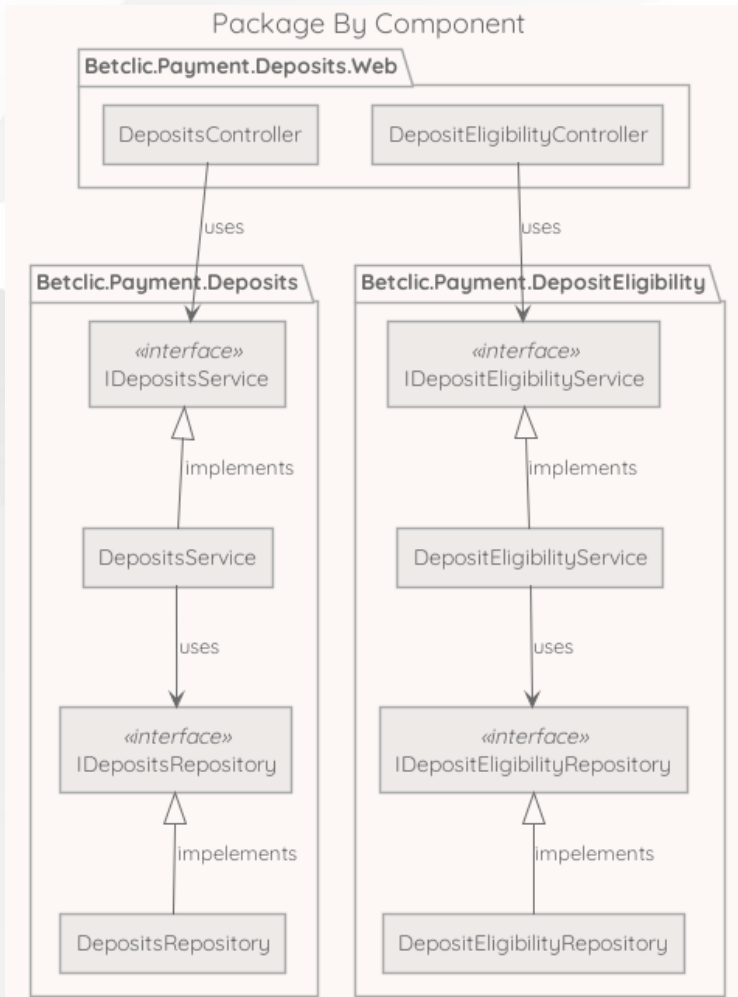- dto & verbosity
- overwhelming structure

📦 Package by layer

📦 Package by feature

Package By Feature

Betclic.Payment.Deposits

DepositsController

⮟ uses

«interface»
IDepositsService

△ implements

DepositsService

⮟ uses

«interface»
IDepositsRepository

△ impelements

DepositsRepository

Betclic.Payment.DepositEligibility

DepositEligibilityController

⮟ uses

«interface»
IDepositEligibilityService

△ implements

DepositEligibilityService

⮟ uses

«interface»
IDepositEligibilityRepository

△ impelements

DepositEligibilityRepository

📦 Package by component

🚀 Make a choice

To conclude, moving to something like Clean Architecture is a pretty tough subject and should be driven in a similar way to:

1. evaluate interest across the team

2. if the team is into it, make sure everyone can learn from the source (if not, I strongly suggest we can go drink some beers)

3. when everyone is interested and have read the source book, we can recruit volunteers in order to make a choice of implementation