What is Clean Architecture in .NET and How Can I Implement It?

Asked 2 years, 6 months ago Modified 1 month ago Viewed 2k times

-2

What is "Clean Architecture" ? What doesn't qualify as "Clean Architecture" ?

I’m interested in learning about Clean Architecture in .NET and how to implement it effectively. Here are some details about my current setup and what I’m looking to achieve:

Environment: .NET 8 Project Type: ASP.NET Core Web API Current Architecture: Monolithic with some separation of concerns

Questions:

What are the core principles of Clean Architecture, and how do they apply to .NET projects? How should I structure my .NET project to follow Clean Architecture principles? What are the best practices for implementing Clean Architecture in an ASP.NET Core Web API? Can you provide examples or code snippets that illustrate the implementation of Clean Architecture in .NET? What are the common challenges and pitfalls when adopting Clean Architecture, and how can I avoid them? Additional Context:

Database: SQL Server Dependencies: Entity Framework Core, MediatR, AutoMapper Goals: Improve maintainability, testability, and scalability of my application Any insights, resources, or examples would be greatly appreciated!

.net-core clean-architecture

Share Follow

edited Oct 28 at 9:06

asked Jun 14, 2022 at 4:44

amit agarwal

83 ● 5 ● 19

1 This is **much too broad** - entire books have been written about this..... see this post for an intro blog.cleancoder.com/uncle-bob/2012/08/13/… - and gazillions of other blog posts and books about the topic ... – marc_s Jun 14, 2022 at 5:00

I agree , however I don't see clean architecture without mentioning DDD , CQRS and Onion Architecture (sometimes). So the question was does Clean Architecture can co exist without these as well ? – amit agarwal Jun 14, 2022 at 5:25

2 Answers

Sorted by: Highest score (default)

Since the question is too broad, I try to answer it on a high level.

> **What is "Clean Architecture" in .NET ?**

In short: "The same as in every other language". Architecture is mainly about managing dependencies, because dependencies are the main problem when it comes to code smells like *fragility*, *rigidity* and *immobility*. It is also often called the structure of the system and I can structure .NET application the same way as Java, JavaScript or even C++ applications. The concepts of *repositories*, *use cases* (*interactors*), *entities* and so on stay the same, even though their implementation differ based on the language features.

> **What doesn't qualify as "Clean Architecture" ?**

I would say each architecture that breaks the main rule of separating the business value from technical details. That's the core of the clean architecture - to make the business rules technology agnostic with the goal to make them easy to test.

So whenever you have a system structure that needs to boot up a complex framework, a web server or a database (that must be initialized with ddl and dml) just to

test your business rules, you don't have a clean architecture.

> Are CQRS, DDD mandatory for "Clean Architecture" ?

No, these concept usually fit very well with the clean architecture, but they are concepts that the clean architecture does not require. E.g. you can implement your domain logic as an anemic model and still be clean architecture compliant. But I think it would be a better idea to use DDD or at least a kind of rich domain model.

> **What is onion architecture ?**

The onion architecture is an architecture that was introduced by Jeffrey Palermo. He also wants to decouple the business rules from the technology details. Jeffrey Palermo says:

> *Hexagonal architecture and Onion Architecture share the following premise: Externalize infrastructure and write adapter code so that the infrastructure does not become tightly coupled.*

and he says:

> *The database is not the center. It is external.*

Thus the clean archtitecture and the onion architecture have a lot in common. That is not a big suprise, because Robert C. Martin says in his *The Clean Architecture* blog:

> *Over the last several years we've seen a whole range of ideas regarding the architecture of systems. These include:*
>
> - *Hexagonal Architecture (a.k.a. Ports and Adapters) by Alistair Cockburn and adopted by Steve Freeman, and Nat Pryce in their wonderful book Growing Object Oriented Software*
> - *Onion Architecture by Jeffrey Palermo*
> - *Screaming Architecture from a blog of mine last year*
> - *DCI from James Coplien, and Trygve Reenskaug.*
> - *BCE by Ivar Jacobson from his book Object Oriented Software Engineering: A Use-Case Driven Approach*
>
> Though these architectures all vary somewhat in their details, they are very similar.

Thus the clean architecture is a consolidation of other architectures that is enhanced with ideas from Robert C. Martin.

I hope my answer helps you to classify the different terms.

Share Follow

edited Dec 12, 2023 at 7:53

answered Jun 15, 2022 at 6:18

René Link

51.2k • 14 • 117 • 147

3

> What is "Clean Architecture" ?

Let's refer to the primary source, the book "Clean Architecture: A Craftsman's Guide to Software Structure and Design". Here are some quotes from Robert Martin:

> The architect can employ the **Single Responsibility Principle** and the **Common Closure Principle** to *separate those things that change for different reasons*, and to *collect those things that change for the same reasons* — given the context of the intent of the system.

> Thus we find the system divided into decoupled at least four horizontal layers - the **Application-independent business rules**, **Application-specific business rules**, **UI**, and the **Database**.

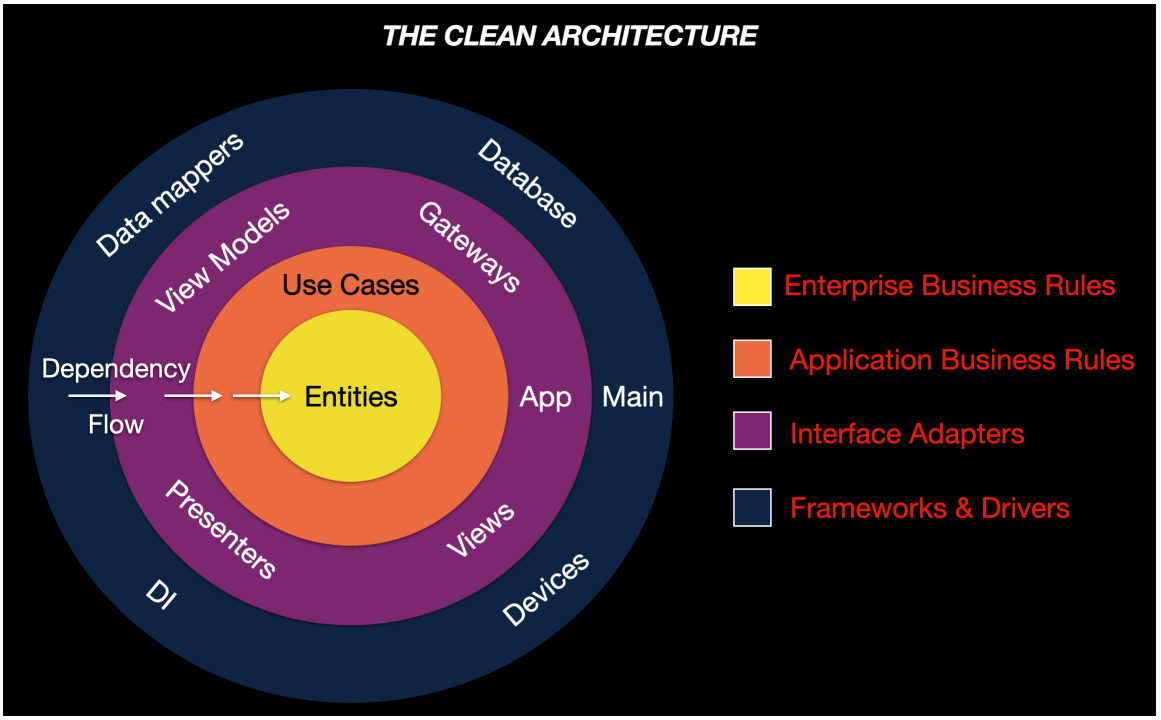
Later Uncle Bob describes the implementation of The Clean Architecture with the corresponding four layers, giving them explicit names:

- Enterprise Business Rules (Entities);
- Application Business Rules (Use Cases);
- Interface Adapters;
- Frameworks & Drivers;

> ...arrange the code in those components such that the arrows between them point in one direction—toward the core business.
>
> You should recognize this as an application of the *Dependency Inversion Principle* and the *Stable Abstractions Principle*. Dependency arrows are arranged to point from lower-level details to higher-level abstractions.

For communication between layers, we use interfaces and abstract methods and their implementation in outer

layers.



Ok it was the original, true Clean Architecture, if we talk about .NET, then we can see, that microsoft documentation has a different point of view on it, even though they use the same reference. Here it is https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures#clean-architecture. But that documentation use that term "Clean Architecture" as the generic description of number of patterns, such as "Onion Architecture", "Hexagonal Architecture", "Ports-and-Adapters", saying that it is the same thing, completely devaluing Robert Martins's work.

So if you follow instructions provided by Uncle Bob, as the primary source and use Microsoft instructions for parts, when something is not clear, I believe that could be considered as the Clean Architecture (*in .NET).

> What doesn't qualify as "Clean Architecture" ?

There is no clear-cut answer on what doesn't qualify as Clean Architecture since different developers may have varying interpretations and opinions on how to apply its principles and guidelines.

However, some examples of what may not qualify as Clean Architecture include:

- A **monolithic application that lacks separation between business logic, presentation, data access, and external dependencies**. This type of application would be tightly linked to particular technologies or frameworks, making it difficult to test, change, or reuse.
- Another example is a layered **application that violates the Dependency Rule, which states that source code dependencies should only point inward toward higher-level policies**. In such an application, circular dependencies between layers or dependencies on concrete implementations rather than abstractions would be present.
- Finally, an **application that doesn't follow the Single Responsibility Principle, which states that a class or module should only have one reason to change**, would be an example of what doesn't qualify as Clean Architecture. In such an application, classes or modules would have multiple

responsibilities or concerns, making them challenging to understand, modify, or test.

Share Follow

answered Jun 11, 2023 at 17:14

Dmytro T

336 ● 4 ● 12