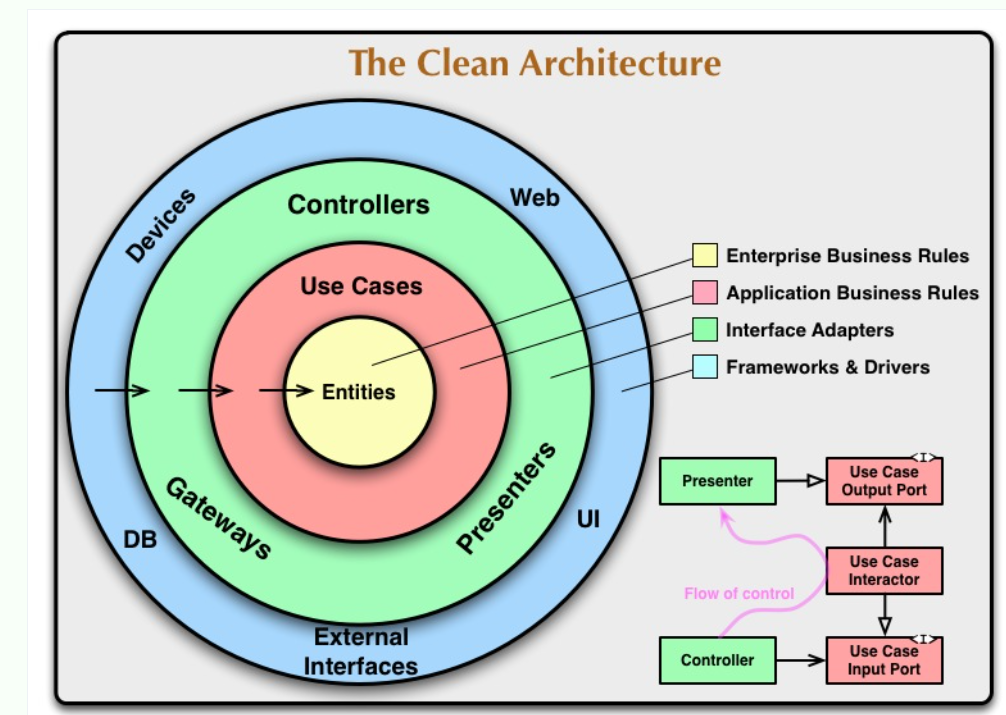
**Clean Architecture**

**Clean Architecture** is a software design philosophy that emphasizes separating business logic from the technical details of a system. The goal is to create systems that are:

* **Independent of frameworks:** You're not tied to a specific library or tool.
* **Testable:** Business rules can be tested without the UI, database, or other external elements.
* **Independent of the UI:** You can easily change the user interface without affecting the core system.
* **Independent of the database:** You can swap out the database without changing the business logic.
* **Independent of external agencies:** Your business logic doesn't depend on anything outside the core system.



**The Dependency Rule**

The key principle of Clean Architecture is the **Dependency Rule**. It states that dependencies should only point inwards. Inner layers should not depend on outer layers. This creates a stable core of business logic that is isolated from the ever-changing external world.

**Layers of Clean Architecture**

Clean Architecture typically involves the following layers:

* **Entities:** The core business objects and rules. They are the most independent part of the system.
* **Use Cases (Interactors):** Application-specific business rules. They control the flow of data to and from the entities.
* **Interface Adapters:** Converts data from the format most convenient for the use cases and entities to the format most convenient for some external agency such as the database or the web.
* **Frameworks and Drivers:** The outermost layer, containing frameworks, databases, UI, and other external tools.