

Clean Architecture



- ¿Que es la arquitectura de software?
- ¿Que es Clean Architecture?
- ¿Cuál es el objetivo de Clean Architecture?
- Diagrama explicado
 - Enterprise Business Rules
 - Application Business Rules
 - Interface Adapters
 - Frameworks and Drivers
- Ejemplo en código

¿Que es la arquitectura de software?



¿Que es la arquitectura de software?

¿Qué es?

La arquitectura de software son patrones o lineamientos que ayudan a la construcción de un programa (aplicación). Estos patrones permiten tener una guía para los desarrolladores, analistas y todos los cargos relacionados para lograr cumplir con los requerimientos de la aplicación.

*En otras palabras se define por las decisiones tomadas por sus constructores. Esta estructura se materializa en la organización de los componentes del sistema y la manera en que estos interactúan entre sí

Importancia de la arquitectura de software

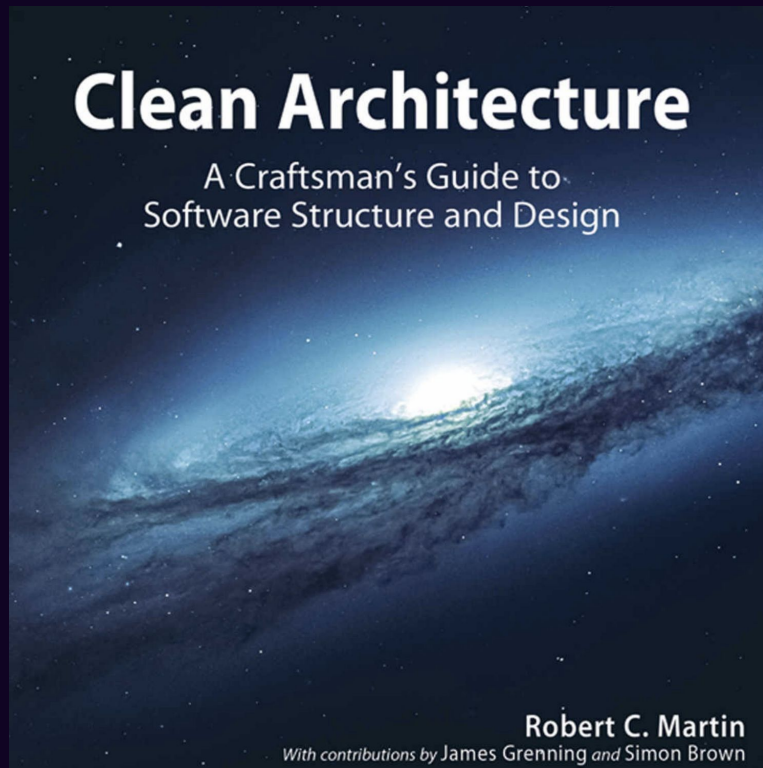
- Coste: ¿Cuánto estamos dispuestos a invertir en el desarrollo y mantenimiento de nuestro sistema?
- Tiempo de desarrollo: Muy relacionado con lo anterior, debemos de preguntarnos cuánto tiempo disponemos para desarrollar el producto
- Número de usuarios: Sin duda uno de los ítems críticos a la hora de desarrollar el producto es preguntarnos qué tipo de producto es y cuantos usuarios soporta ¿Funciona a través de web?, ¿Debe de soportar cargas elevadas por diseño?, etc
- Nivel de aislamiento: Otro factor importante a tener en cuenta es si nuestro producto funciona de forma aislada al resto de productos del usuario o si debe de integrarse o permitir integraciones de terceros



¿Que es Clean Architecture?



¿Que es Clean Architecture?



¿Que es?

Es una perspectiva de arquitectura enfocada en la lógica de negocio, que emplea capas para separar componentes que pueden cambiar por razones diversas, y establece reglas de dependencia entre ellos. Esta metodología se basa en la premisa de estructurar el código en capas contiguas, es decir, que solo tienen comunicación con las capas que están inmediatamente a sus lados.

Nuestro dominio no necesita saber qué framework web o qué sistema de base de datos estamos utilizando



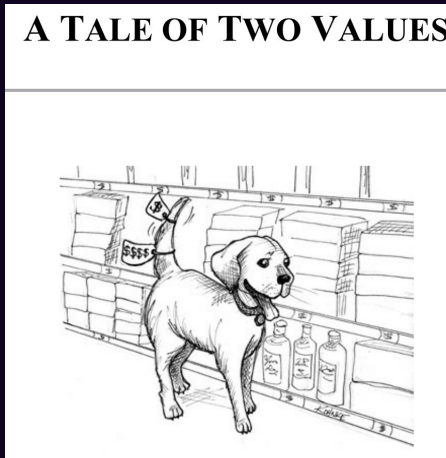
¿Que es Clean Architecture? – Valores

Comportamiento

Se contratan programadores para fabricar sistemas que se comporten de tal manera que genere o ahorre dinero para las partes interesadas.

Escribimos el código que hace que las máquinas satisfagan esos requisitos

A TALE OF TWO VALUES



Arquitectura

El software fue inventado para ser "fácil". Su objetivo es poder cambiar fácilmente el comportamiento de las máquinas.

Cuando las partes interesadas cambian de opinión sobre una característica, ese cambio debería ser simple y fácil de realizar.

La dificultad para realizar tal cambio debería ser proporcional sólo al alcance del cambio y no a la forma del cambio

¿Cuál es el objetivo de Clean Architecture?



¿Cuál es el objetivo de Clean Architecture?

Independencia de cualquier agente externo

Las reglas de su negocio simplemente no saben nada sobre el mundo exterior

Independencia de DB

Puede cambiar a Oracle o SQL Server por Mongo, BigTable u otra cosa. Sus reglas de negocio no están vinculadas a la base de datos

Testeable

Las reglas de negocio se pueden probar sin la interfaz de usuario, la base de datos, el servidor web o cualquier otro elemento externo



Independencia del Frameworks

La arquitectura no depende de la existencia de alguna biblioteca de software cargado de funciones. Esto le permite utilizar dichos marcos como herramientas, en lugar de tener que meter su sistema en sus limitaciones limitadas

Independencia de la UI

La interfaz de usuario puede cambiar fácilmente, sin cambiar el resto del sistema. Una interfaz de usuario web podría reemplazarse por una interfaz de usuario de consola, etc



Diagrama explicado



Diagrama

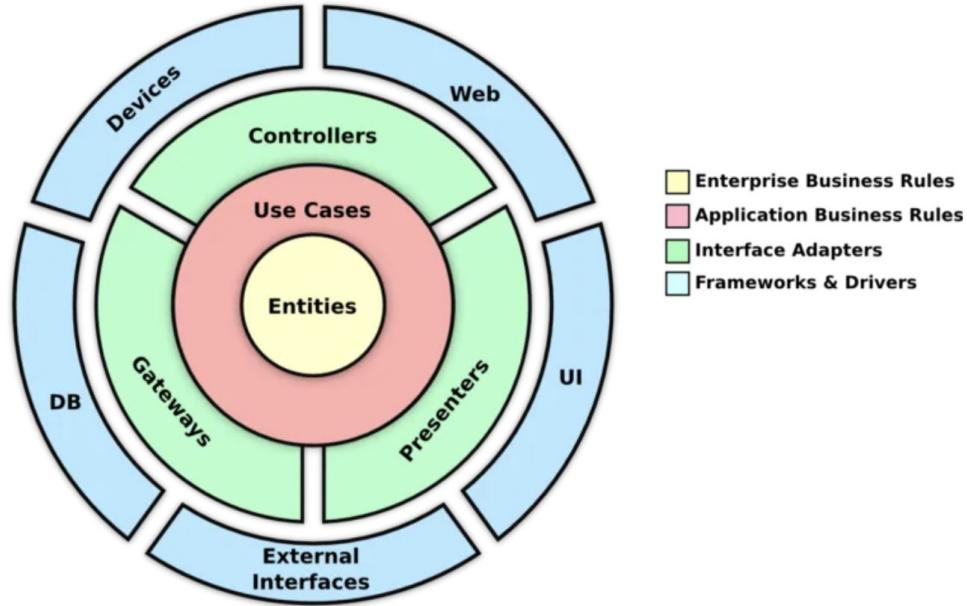


Diagrama – Enterprise Business Rules

Entities

Las entidades encapsulan reglas de negocio de toda la empresa. Una entidad puede ser un objeto con métodos o puede ser un conjunto de estructuras de datos y funciones. Siempre y cuando las entidades puedan ser utilizadas por muchas aplicaciones diferentes en la empresa.

Si solo está escribiendo una sencilla aplicación, entonces estas entidades son los objetos comerciales de la aplicación. Encapsulan las reglas más generales y de alto nivel. Son los que tienen menos probabilidades de cambiar cuando algo externo cambia. Por ejemplo, no esperaríamos que estos objetos se vieran afectados por un cambio en la navegación de la página o en la seguridad



**Enterprise Business
Rules**
(Entities)

Características

- Lógica de negocio crítica, de alto nivel
- Objetos con métodos o funciones
- No cambian, a menos que lo haga el negocio
- Existen sin que exista la aplicación



Diagrama – Application Business Rules

Use Cases

El software de esta capa contiene reglas de negocio específicas de la aplicación. **Encapsula e implementa todos los casos de uso del sistema.** Estos casos de uso orquestan el flujo de datos hacia y desde las entidades, y dirigen a esas entidades a utilizar sus reglas comerciales de toda la empresa para lograr los objetivos del caso de uso.

No esperamos que los cambios en esta capa afecten a las entidades. Tampoco esperamos que esta capa se vea afectada por cambios en externalidades como la base de datos, la interfaz de usuario o cualquiera de los marcos comunes. Esta capa está aislada de tales preocupaciones. Sin embargo, esperamos que los cambios en el funcionamiento de la aplicación afecten los casos de uso. **Si los detalles de un caso de uso cambian, entonces parte del código de esta capa seguramente se verá afectado**



**Application Business
Rules**
(Use Cases)

Características

- Lógica de negocio de la aplicación
- Produce la información a ser presentada
- Orquestan la lógica de negocio de las entidades



Diagrama – Interface Adapters

Controllers, Gateways, Presenters

El software en esta capa es un conjunto de adaptadores que **convierten datos del formato más conveniente para los casos de uso y entidades, al formato más conveniente para alguna agencia externa como la Base de Datos o la Web**. Ningún código dentro de este círculo debería saber nada sobre la base de datos. Si la base de datos es una base de datos SQL, entonces todo el SQL debe restringirse a esta capa y, en particular, a las partes de esta capa que tienen que ver con la base de datos.

También en esta capa se necesita cualquier otro adaptador para convertir datos de algún formato externo, como un servicio externo, al formato interno utilizado por los casos de uso y las entidades.



Interface Adapters
(Controllers, Gateways,
Presenters)

Características

- Adaptan información de los casos de uso a agentes externos
- Adaptan información de agentes externos al formato requerido por casos de uso

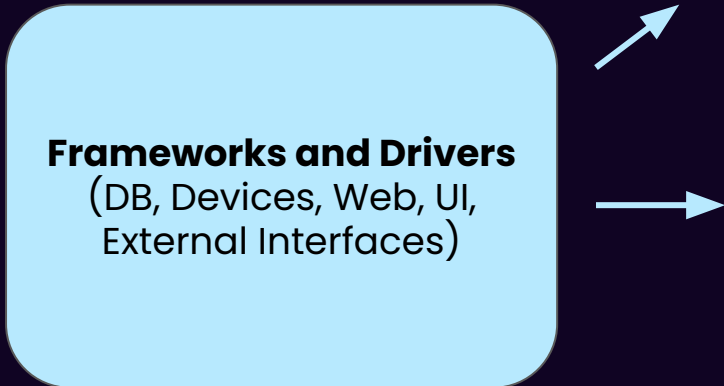


Diagrama – Frameworks and Drivers

DB, Devices, Web, UI, External Interfaces

La capa más externa generalmente se **compone de marcos y herramientas como la base de datos, el marco web, etc.** Generalmente no se escribe mucho código en esta capa aparte del código adhesivo que se comunica con el siguiente círculo hacia adentro.

Esta capa es donde van todos los detalles. La Web es un detalle. La base de datos es un detalle. Mantenemos estas cosas en el exterior, donde pueden causar poco daño



Frameworks and Drivers
(DB, Devices, Web, UI,
External Interfaces)

Características

- Frameworks web y drivers de base de datos
- Acá se encuentran todos los detalles de la aplicación
- Se ubican en la capa más alejada, donde pueden hacer poco daño



Ejemplo de código



Github

En el siguiente link encontraran un ejemplo creado en typescript con la implementación de clean-architecture

<https://github.com/julianmerlo95/clean-architecture-example>

Referencias

- **Medium:**
<https://medium.com/swlh/clean-architecture-a-little-introduction-be3eac94c5d1>
- **Cleancoder:**
<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>
- **Openwebinars**
<https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/>

