Clean Architecture

Moisés García López UO264802

Fabio Cuartas Puente UO257494

Iyán Sanjurjo Luna UO264921

CONTENIDO

INTRODUCCIÓN

En estos últimos años han surgido diferencies ideas en torno a la arquitectura de sistemas (Hexagonal Architecture, Onion Architecture, Screaming Architecture...). Todas ellas con sus particularidades, pero con un concepto común, la división del software en capas.

La Arquitectura Limpia es una de ellas: podríamos describirla como un conjunto de principios centrados en aislar los detalles de implementación de la lógica de dominio, haciendo que sea más mantenible y escalable.

CAPAS

ENTIDADES

En la capa más interna se sitúan las entidades, que encapsulan las reglas de negocio. Reglas que seguirían existiendo, aunque no hubiese aplicación informática (por ejemplo, que un banco cobre a sus clientes un 10% más en una operación).

Las entidades no saben nada del resto de capas ni dependen de ellas, por lo que casi nunca van a cambiar cuando se produzca un cambio externo.

Toman la forma de objetos (con sus métodos) o simplemente un conjunto de estructuras de datos y funciones, lo mismo da, con tal de que el resto de capas puedan usarlas adecuadamente.

CASOS DE USO

El software de esta capa contiene reglas comerciales 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 las externalidades como la base de datos, la interfaz de usuario o cualquiera de los marcos comunes. Esta capa está aislada de tales preocupaciones.

Nosotros sí , sin embargo, esperamos que los cambios en el funcionamiento de la aplicación van a afectar a los casos de uso y por lo tanto el software en esta capa. Si los detalles de un caso de uso cambian, es seguro que algún código de esta capa se verá afectado.

ADAPTADORES

En esta tercera capa, se encuentra todo aquel software que transforma los datos generados a partir de los casos de uso en algo entendible por los elementos externos de la siguiente capa y viceversa.

Por un lado, para presentar visualmente una interfaz gráfica, es necesario convertir objetos de la lógica de negocio en objetos propios del framework que utilicemos para la interfaz, ya sea Swing, JSF...

Por otro lado, para acceder a una base de datos, disponemos de repositorios que realizan esa adaptación según el sistema de persistencia, sea en este caso ORM, JPA, Hibernate...

Además de estas dos opciones, también incluye cualquier tipo de convertidor, por ejemplo, a estructuras JSON o cualquier otro tipo de servicio externo.

FRAMEWORKS Y DRIVERS

Se trata de la capa más externa de las cuatro, en ella se encuentran todas aquellas herramientas que se deben mantener alejadas para evitar cualquier contacto que pueda generar fallos en la aplicación. Como se explicaba anteriormente, entra en contacto con el resto de la aplicación a través de una serie de adaptadores.

Ejemplos de este tipo de herramientas pues pueden ser una base de datos, una interfaz gráfica, o cualquier tipo de dispositivo externo.

REGLA DE DEPENDENCIA

La regla de dependencia se cumple siempre, es la regla que va de fuera a dentro, de infraestructura a aplicación y de aplicación a dominio,

Esta regla dice que las dependencias de código fuente solo pueden apuntar hacia adentro. Nada en un círculo interno puede saber nada sobre algo en un círculo exterior. En particular, el nombre de algo declarado en un círculo exterior no debe ser mencionado por el código en el círculo interno. Eso incluye, funciones, clases. variables, o cualquier otra entidad de software con nombre.

Del mismo modo, los formatos de datos utilizados en un círculo externo no deben ser utilizados por un círculo interno, especialmente si esos formatos son generados por un marco de trabajo en un círculo externo. No queremos que nada en un círculo exterior afecte a los círculos internos.

VENTAJAS

Una de las ventajas principales del uso de arquitectura limpia es la independencia. En primer lugar, la independencia de la interfaz de usuario. Cualquier cambio que se realice en esta no debe influir en el resto de la aplicación. También la independencia de la base de datos, la sustitución de esta no debe influir en el funcionamiento o de hacerlo tener un impacto mínimo. Finalmente, la independencia de frameworks; estos actúan como lo que deberían ser, herramientas, y no nos obligan a actuar de una forma u otra.

Otra gran ventaja es la posibilidad de testeo de la lógica de negocio sin necesitar ni interfaz gráfica ni ningún otro componente. Es decir, se trata de crear una aplicación completamente desacoplada, esto de por sí implica una mayor facilidad de testeo.

La última de las ventajas principales es disponer de un dominio en el centro de toda la aplicación. Un dominio que no depende de ninguna de las demás capas, pero del cual dependen todas las demás.

Además de todas estas ventajas, la arquitectura limpia te proporciona muchas otras, que, dependiendo de hacia donde esté enfocado tu proyecto puede servirte de ayuda. Como puede ser una evolución más rápida del producto o la eliminación y desactivación de cierta funcionalidad. También ganarás confianza como programador, sabiendo que un cambio no debería producir errores en funcionalidades ya existentes. Y, por último, a largo plazo se va a conseguir un código muchísimo más manejable sobre todo para temas de mantenimiento.

DESVENTAJAS

Metodología: Todos los desarrolladores deben conocer la metodología de la arquitectura limpia lo que conlleva que todos deben ser responsables de respetar las capas y garantizar la regla de dependencia.

Complejidad: La curva de desarrollo es lenta al principio ya que hay que establecer los pilares de la aplicación y hacerse a la nueva forma de trabajar. La idea es que poco a poco y a la larga el desarrollo y mantenimiento de la aplicación sea más sencillo.

CONCLUSIONES

Debemos tener en cuenta que la Arquitectura Limpia no establece guías ni manuales. Simplemente nos aporta una serie de principios a los que ajustarnos en mayor o menor medida, todo dependiendo del proyecto que tengamos entre manos.

Nosotros hemos hablado de cuatro capas, pero en ocasiones pueden darse arquitecturas con tres o cinco.

Al fin y al cabo, la separación en capas, la comunicación entre estas mediante Inversión de Dependencias... Son medios para ahorrarnos futuros dolores de cabeza, aumentando productividad y comodidad.