Digitalización y actualidad

Manuela Uribe Zapata
Abril 11 de 2020
Arquitectura de Software

Se puede decir que es de conocimiento público que desde la década del 80 se ha venido dando una revolución a la forma en la que se administra la información, entendiendo información como toda la base de conocimiento de los negocios, se han implementado diferentes dispositivos electrónicos y detrás de estos un mundo enorme de conocimiento en el ámbito de la informática, el desarrollo de software y de hardware.
Los pasos de la esta "revolución" en comparación con otras han sido agigantados y no solo por la velocidad en la que se avanza en los procesos sino por la creación de muchos nuevos, el abarcar esto de forma general toma mucho tiempo y para mi caso bastante complejo por falta de conocimiento en algunas tecnologías e historia por lo cual este texto de enfocará en DDD.

Pero ¿qué es DDD? Se puede preguntar algún lector y la respuesta más simple para esto es que son unas siglas que en inglés significan Domain Driven Design y en español Diseño Guiado por el Dominio, pero la verdad las siglas son lo menos importante, es apenas un abrebocas a un enfoque que nos permitirá visualizar el mundo del desarrollo de software desde otra perspectiva ya que el quedarnos con los enfoques tradicionales en proyectos actuales puede no ser lo más indicado.

Al abandonar el enfoque tradicional de los negocios y querer digitalizarlos se ha dado la posibilidad al gremio de la tecnología de experimentar formas de trabajo, metodologías, lenguajes, enfoque, arquitecturas, etc. que se adapten de una forma idónea a estos, uno de estos enfoques que actualmente es significativo es DDD pues este nos brinda un objetivo claro con un fuerte apoyo en el diseño y este complementa muy bien el trabajo en sistemas distribuidos y arquitecturas limpias, los cuales son fundamentales en muchos de los proyectos actuales.

Creo conveniente dar un muy breve contexto de que son los sistemas distribuidos y de las arquitecturas limpias para así informar al lector en caso de no estar muy contextualizado como el DDD es un enfoque de gran ayuda para estos.

Sistemas distribuidos son aquellos en los que los hay un grupo de computadores los cuales están trabajando al tiempo de forma coordinada con la intención de conseguir un objetivo específico, la información y programas estarán guardados en varios computadores pero para los usuarios finales el producto se percibirá como si hubiese sido desarrollado en

uno solo, estos son muy populares actualmente ya que permite que compartir información entre las computadoras conocidas como nodos pueda ser más sencillo, pueden añadirse permitiendo una escalabilidad de formas más simple y óptima y que a la hora de presentarse un fallo en algún nodo pueda desligarse y resolverse.

Arquitectura limpia, primero la arquitectura de software es la "constitución" del desarrollo de software en esta se informa a los interesados en el proyecto las reglas y formas de trabajar el proyecto permitiendo garantizar un proyecto un proyecto de alta calidad, de buen rendimiento y escalabilidad durante el tiempo que perdure el proyecto y una arquitectura limpia busca que el proyecto a desarrollar tenga independencia de los frameworks que sea testable, que sea independiente de la base de datos y cualquier entidad externa.

Ahora si **DDD** es un enfoque de desarrollo que busca brindar una conexión entre las áreas del negocio, los conceptos y las arquitecturas, como es un enfoque este no brinda al equipo de desarrollo una receta única con la cual se pueda dar solución a las dificultades o cualquier tipo de problema que se presente, pero si tiene una "estructura" que puede dar al lector una idea de como es este:

- Tactical design: Esta sección es la encargada de todo lo que está relacionado o enfocado al código, aquí se manejan diferente conceptos que se describen a continuación.
    - Value object: Es una parte en la cual hay contenido de valor, suele ser una clase.
    - Model-driven design: Es un mapeo desde el contexto del proyecto, ayuda a identificar la forma en la cual se realizará el modelado en término de las clases (value object), entidades y servicios.
    - Aggregates: Conjunto de entidades y clases (value object), en pocas palabras es un modelo conceptual el cual es guardado en la capa de repositorio.
    - Domain event: Como su nombre lo dice es donde se tiene el dominio de los eventos este sirve para expresarlo.

- Strategic design: Es la sección encargada de la parte más humana, la relación entre los equipos de trabajo y con las personas
    - Ubiquitous Language : refleja lo modelado en el negocio en código
    - Bounded context : facilita la implementación de lo agrego
    - Continuous Integration: permite percibir los cambios en el momento, esto en ddd permite que cada área sea independiente y pueda esta subiendo información a producción en cualquier momento , no todo tiene que estar en el mismo lenguaje

- Big ball of mud: Aquí se encuentra el código espagueti, la cual es muy costosa a la hora del soporte
- Anticorruption layer: que las capas no se afecten entre si, que no me afecten el dominio
- Separate ways: permite que todo esté separado y que cada parte del equipo tenga autonomía
- Open host service: genera una api para que las partes se com
- Published language: api
- Shared kernel: lo más general del proyecto

Normalmente cuando se implementa DDD se suele usar una arquitectura hexagonal ya que el dominio no se ve condicionado a la bd, el estudio de la arquitectura hexagonal es de gran relevancia a la hora de hablar de DDD ya que esta es una de las alternativas que ofrece la arquitectura limpia ya que esta facilita el trabajo cuando la complejidad de los proyectos es alta gracias a que esta permite la separación de responsabilidades por capas y aunque esta arquitectura tiene unos lineamientos y reglas específicas es muy flexible lo que permite que esta pueda ser implementada en diferentes proyectos.

Para concluir, el mundo del desarrollo en su parte más fundamental que para mi esta en la arquitectura ha sufrido una revolución llevando a que las personas relacionadas busquen alternativas a las tradicionales permitiendo que los nuevos proyectos puedan ser trabajados de una forma más cómoda y DDD junto con las arquitecturas que se implementan con este y los tipos de sistemas son producto de las necesidades de los business core que han estado avanzar a la digitalización y de los antiguos que han adquirido una complejidad significativa.

Teniendo en cuenta el momento de la historia en el que estamos y que creo que el mundo tendrá un cambio significativo en la forma en la que concibe los negocios generando más cambios y más formas de asumir los retos que se viene.