**Marco teorico**

El desarrollo de software en la actualidad se divide en dos metodologías principales, las metodologías tradicionales y las ágiles. En primer lugar, el proceso de desarrollo en una metodología tradicional se basa en técnicas predictivas, las cuales permiten estimar el alcance de un producto según sus requerimientos y el presupuesto asignado al mismo, por lo tanto todas las etapas del ciclo de vida del desarrollo deben ajustarse en gran medida al cronograma pactado al inicio del proyecto, estas etapas se describen a continuación [1]:

* Análisis de requisitos: En este punto se analiza la necesidad del cliente con el fin de identificar la factibilidad económica, técnica y operacional del proyecto.
* Definición de requisitos: Luego de ser analizados, los requisitos deben ser debidamente documentados y aprobados por el cliente para ser diseñados y desarrollados en posteriores etapas del ciclo de vida del proyecto.
* Diseño de la arquitectura del producto: Se desarrolla una propuesta arquitectónica para el desarrollo la cual debe ser aprobada por todas las partes interesadas, en esta propuesta se deben tener en cuenta aspectos como el riesgo, robustez del producto, presupuesto y restricciones de tiempo.
* Desarrollo del producto: En esta etapa se crea el código fuente del producto basado en los requisitos y estructurado según la arquitectura definida, ademas se debe cumplir con especificaciones de lenguaje y framework de programación a utilizar.
* Pruebas del producto: La funcionalidad del producto debe ser probada con el fin de identificar defectos para implementar su debida solución.
* Operación y mantenimiento: Una vez el producto es probado se procede a incluirlo en ambiente de producción y liberarlo al mercado.

En metodologías tradicionales este ciclo de vida se desarrolla una única vez en todo el proyecto, entregando un producto terminado al cliente con todos los requisitos solicitados al inicio.

Por otro lado las metodologías tradicionales basan su filosofía en entregas continuas que generen valor al negocio del cliente por lo tanto todo el ciclo de vida del producto se realiza múltiples veces durante su desarrollo en los llamados Sprints con la finalidad de realizar los cambios necesarios en el momento oportuno, estos Sprints tienen una duración promedio de 2 a 4 semanas, al ser un periodo de tiempo tan corto en cada uno de ellos se desarrollan pocos requisitos del producto y se repite el ciclo de manera incremental hasta alcanzar un producto viable que pueda ser llevado a producción (Release), con el cual el cliente pueda generar retorno de inversión.

A continuación se presenta una tabla comparativa entre metodologías de desarrollo tradicional y ágil:

Ahora, independiente de la metodogia con la que se aborde el desarrollo del producto, es necesario establecer herramientas que faciliten el ciclo de vida del desarrollo de software, para el caso de las dos primeras etapas del ciclo (Análisis de requisitos y Definición de requisitos) existen múltiples maneras de ser abordadas, en la actualidad suele utilizarse una metodología llamada Design Thinking ya que se incluye activamente al cliente dentro del proceso.

Design Thinkin es reconocido como un nuevo paradigma desde el año 2009 ya que se adaptó muy bien a la solución de problemas en diversos tipos de industrias, su éxito lo debe a la manera de abordar la concepción tanto del problema como su solución en donde modifica la formula que utiliza normalmente el raciocinio humano para encontrar una alternativa de solución. Normalmente el ser humano conoce muy bien el QUE se debería realizar y el COMO debe hacerse, pero tiene total desconocimiento del valor o el resultado que le otorgará este proceso, por otro lado design thinking se centra en el valor que espera tener, luego de tener muy claro este valor se procede a definir la estrategia (o el COMO) se pretende obtener el resultado esperado y por ultimo se identifica el QUE o la solución especifica para cumplir con la meta [2].

Si bien design thinking permite una definición clara y precisa del problema y la estrategia a abordar, es necesario validarla, para esto es conveniente utilizar un enfoque de doble diamante, el cual consta de cuatro etapas que pretenden dar una idea prematura de la viabilidad de la solución tanto de manera técnica como de mercado [3].

El proceso de doble diamante comienza con una etapa de descubrimiento, en ésta se realiza una investigación tanto de mercado como de posibles usuarios con el objetivo de comprender si la posible solución encaja o no en algún sector del mercado haciendo el proyecto viable, en este punto es conveniente realizar encuestas o estudios de aceptación con posibles usuarios del producto. Posteriormente con la finalidad de cubrir las demás etapas del ciclo de vida del desarrollo se pasa a una etapa de definición donde se alinean las necesidades encontradas en el sector de mercado con objetivos de negocio, luego se inicia la tapa de desarrollo en la cual se crean los elementos de software necesarios, siempre teniendo presentes las pruebas de software que aseguren la calidad del producto construido y por ultimo se lleva a cabo la etapa de liberación en la cual se realiza el lanzamiento del producto en el mercado objetivo identificado en las etapas iniciales [4].

Ahora que se puede definir de manera concisa el problema y a su vez el valor que se generará, es necesario centrarse en el modo en que será desarrollada la solución. En el desarrollo de software se identifican dos principales arquitecturas, la monolítica y la orientada a microservicios, esta primera consiste en un desarrollo centralizado donde todos los módulos y funcionalidades se encuentran en un mismo proyecto lo que a gran escala representa una aplicación robusta y segura, pero presenta un inconveniente al momento de abordar necesidades de escalabilidad, un ejemplo de esto es cuando se necesita escalar un modulo muy popular para los usuarios del sistema, esto implica a su vez escalar los módulos que no son tan populares lo que representa mayor consumo de recursos de los módulos que no los necesitan y, como los recursos suelen ser limitados, se limita también la capacidad de escalar. Otra característica de los sistemas monolíticos es que suelen ser desarrollados en una misma base de código con un mismo lenguaje lo que, aunque lo hace más comprensible por parte de los desarrolladores, obstruye el potencial que puede sacarse de otras herramientas, ademas, aumenta la complejidad en el caso de una posible migración.

Precisamente para solventar los inconvenientes de la arquitectura monolítica se plantea la arquitectura de microservicios, en la cual cada modulo del sistema es un proyecto a parte y por lo tanto posee sus propios datos, tecnologías, lenguajes o herramientas otorgando la posibilidad de que cada microservicio sea desplegado en un servidor a parte con propiedades específicas, así, si un modulo es mas popular para el usuario puede desplegarse en un servidor con más recursos que un servidor en el que se aloje un módulo menos popular. También, llegado el caso en el que se necesite agregar un nuevo modulo o funcionalidad solo se debe crear un nuevo microservicio y el desarrollo de este no afectara el funcionamiento de los demás. Todos estos beneficios están sujetos a un proceso de despliegue con una complejidad mas alta, el cual en la mayoría de los casos, debido a la cantidad de microservicios creados, debe ser automatizado.

**Metodologia**

Con el objetivo de desarrollar una plataforma que se ajuste al modelo de negocio planteado por Ruta N, se asistió a unas sesiones de acompañamiento con el laboratorio de creación que dispone la corporación; En estas sesiones se buscaba plantear de manera clara la necesidad que busca solventar la plataforma, identificar el mercado objetivo y sus competencias, prototipado de la solución y por ultimo aterrizar un modelo de negocio viable.

Las sesiones del laboratorio se distribuyeron en una sesión de 4 horas

semanal durante cuatro semanas donde se utilizó una metodología compuesta por conceptos de design thinking y metodología doble diamante

Referencias

[1] Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software development: Agile vs. traditional. Informatica Economica, 17(4), 64.

[2] Dorst, K. (2011). The core of ‘design thinking’and its application. Design studies, 32(6), 521-532.

[3] Clune, S. J., & Lockrey, S. (2014). Developing environmental sustainability strategies, the Double Diamond method of LCA and design thinking: a case study from aged care. *Journal of cleaner production*, *85*, 67-82.

[4] [https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons\_Design\_Council%20(2).pdf](https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council (2).pdf)+

[5] Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *Computing Colombian Conference (10CCC), 2015 10th* (pp. 583-590). IEEE.

[6] <http://www.adeveloperdiary.com/java/spring-boot/develop-microservices-using-netflix-oss-spring-boot/>

[7] López, D., & Maya, E. (2017). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.