UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación de APIs de marketing y migración de componentes a nuevas tecnologías

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Yanni Martínez Martínez

ASESOR DE INFORME

Ing. Alberto Templos Carbajal

Ciudad Universitaria, Cd. Mx., 2024

Agradecimientos:

Mi más sincero agradecimiento a mis padres y hermanos, que durante toda mi trayectoría estuvieron al pendiente de lo que necesitaba para poder desarrollarme de la mejor manera dentro del ámbito académico, personas que se han vuelto fundamentales en este trayecto en las buenas y en las malas, aquellas que nunca dudaron en dar lo máximo de ellos para brindarme lo suficiente para llegar a este punto. El presente agradecimiento queda corto frente a tantos valores que me enseñaron y que con ejemplo siempre los vi ejercer, hoy en día llevo un enorme orgullo de ellos que espero también compartan conmigo este gran logro.

Así mismo agradezco a la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería por permitirme ser parte de esta comunidad la cual me brindó incontables vivencias que sin ellas no sería la persona que soy hoy en día.

Mi más sincero agradecimiento al Ingeniero Alberto Templos Carbajal por ser un excelente guía durante el desarrollo de este escrito, así mismo, agradezco toda la retroalimentación y tiempo dedicado que sin duda enriquecieron el desarrollo de este escrito.

1. Índice

2. Introducción

El crecimiento acelerado de la tecnología es algo que ya es parte de nosotros desde hace unos años a la fecha, cada día es más evidente y afortunadamente también más accesible por lo que contar con dispositivos inteligentes y con conectividad a internet resulta algo común y cotidiano en la vida de la sociedad teniendo una infinidad de aspectos positivos facilitando la vida del ser humano de una manera considerable.

Con el reciente auge de la tecnología en los últimos años y tras el confinamiento en el año 2020, gran parte de la población se vio forzada a adaptarse a nuevas maneras de adquirir suministros de todo tipo, lo cual provocó que áreas mercantiles también se adaptaran a ello propiciando una mayor popularidad a nuevos o ya existentes sitios de comercio electrónico. Si bien eso es algo que hace unos años resultaba casi imposible de imaginar, hoy en día es algo que se ha normalizado por la sociedad permitiendo la expansión del mundo del ecommerce para diferentes áreas y a su vez una gran competencia entre negocios los cuales han buscado ganar terreno invirtiendo en análisis e implementación de nuevas tecnologías buscando destacar frente a sus rivales.

La creciente competencia ha forzado o motivado a diversos sitios electrónicos a invertir en análisis de datos y tecnologías que les permitan conocer a más detalle a la audiencia que los frecuenta, es por ello que áreas de marketing han tenido una gran influencia en el ámbito financiero dentro de las empresas ya que les permiten potenciar las ventas o márgenes de utilidad expandiendo su posible éxito, incluso podríamos llegar a pesar que una área de marketing pudiera tener el mismo peso que aquella que se encarga de la innovación tecnológica ya que ambas permiten la exposición de la información y además pueden ir de la mano implementando tecnología que ayude al negocio a visualizar los datos, a generar métricas o campañas para exponer servicios, productos y demás aspectos relevantes y de ese modo construir ofertas, anuncios, combos o cualquier infinidad de combinaciones posibles.

La tecnología ya forma parte de nosotros en acciones cotidiana es por ello que algunos negocios buscan usar esa información que fluye dentro de su mismo sitio para entender las tendencias, los comportamientos, impulsar a la acción de compra o bien sólo comprender el mercado para poder seguir posicionándose frente a su competencia.

3. Marco teórico:

3.1 Paradigmas de programación

Tomando como referencia la definición de la universidad de valladolid que se expone a continuación:

“Un paradigma de programación indica un método de realizar cómputos y la manera en que se deben estructurar y organizar las tareas que debe llevar a cabo un programa”. [U. Valladolid] [1]

En otras palabras podemos considerados como enfoques o maneras de pensar para diseñar y desarrollar un software, existen diversos tipos de paradigmas de programación como lo pueden ser:

1. **Programación imperativa:**

   “Describe cómo debe realizarse el cálculo, no el porqué” [U. Valladolid]

   En otras palabras se centra en cómo modificar el estado de un programa a través de instrucciones y cambios en variables.

2. **Programación declarativa:**

   “Describe que se debe calcular, sin explicitar el cómo” [U. Valladolid]

   Básicamente dice lo que se quiere lograr sin especificar como hacerlo. Los lenguajes de programación declarativos suelen ser más altos nivel y más fáciles de leer y escribir que los lenguajes imperativos (mencionados previamente). Dentro de este ramo podemos encontrar a SQL y Prolog.

[1] “Paradigmas de programación”. Vaca R. Cesar. (Universidad de Valladolid). 11 de febrero del 2011. Fuente: https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf . Consultado el 11 de marzo del 2024.

3. **Programación orientada a objetos:**

   “La programación orientada a objetos se define como un paradigma que permite realizar una abstracción de la realidad, que se puede implementar en una aplicación de software con el fin de resolver problemas mediante el uso de un lenguaje de programación.” [Flores F. H. Arturo][2]

   Este paradigma se centra en la creación de "objetos" que se componen de atributos y métodos declarados en base a clases que actúan como un molde abstracto que permite representar entidades reales en forma lógica a nivel software, cabe resaltar que los objetos pueden interactuar entre sí enviándose mensajes, realizando acciones o modificando el estado de sus atributos, dentro de este tipo de paradigmas podemos encontrar a Java y Python.

4. **Programación funcional:**

   “La programación funcional es un estilo de programación cuyo método básico de computación es la aplicación de funciones a sus argumentos”. [Alonso Jimenez José A.] [3]

   En resumen podría decirse que este paradigma se enfoca más en un funcionamiento mediante funciones matemáticas. Este tipo de paradigma suele tener buen rendimiento para realizar tareas complejas de entender o manejar. Un ejemplo de lenguaje de programación puede ser Haskell.

Como hemos visto, cada paradigma de programación tiene sus propias ventajas y desventajas, por lo que la elección de uno u otro dependerá totalmente de la aplicación que se quiera implementar. Algunos lenguajes de programación soportan múltiples paradigmas, lo que significa que los programadores pueden utilizar diferentes enfoques según lo que sea más apropiado para el problema en cuestión.

[2] “Programación orientada a objetos usando Java”. Flores F. H. Arturo (s. f.). Fuente: http://librodigital.sangregorio.edu.ec/librosusgp/21047.pdf. Consultado el 16 de marzo del 2024.

[3] “Temas de programación funcional”. Alonso Jimenez José A. (s. f.). Fuente: https://www.cs.us.es/~jalonso/cursos/i1m-19/temas/2019-20-I1M-temas-PF.pdf. Consultado el 19 de marzo del 2024

3.2 Programación orientada a objetos

Previamente se dio una idea muy general acerca del paradigma programación orientada a objetos (POO), pero es importante destacar que este paradigma posee cualidades que le hacen único, a continuación se describirán los pilares de ella.

Pero creo conveniente mencionar la descripción proveniente de la Facultad de Ingeniería de la UNAM[4] la cual menciona que;

"*La programación orientada a objetos se basa en el hecho de que se debe dividir el programa … en modelos de objetos físicos o simulados*". *[FI UNAM].*

En resumen nos da una idea de que la programación orientada a objetos busca plasmar a alto nivel una representación de aquello que nos rodea.

Sin embargo, para terminar de aterrizar el concepto es importante saber la descripción de clase y objeto. Las clases, pueden ser consideradas como plantillas o moldes que se definen para la creación de objetos. Por otro lado, los objetos son instancias o representaciones creadas en base a nuestros moldes llamados clases.

A continuación se mencionan los 4 pilares de la Programación orientada a objetos:

- **Encapsulamiento:**

  "Un objeto es como una caja negra, a la que se le envía un mensaje y éste responde ejecutando el método apropiado, el cual producirá las acciones deseadas. un objeto, una vez programado es solo manipulable a través de mensajes. …A este intrínseco vínculo entre datos y métodos y al modo de acceder y modificar sus datos es a lo que llamamos Encapsulación". [Moreno Francisco][5]

  En resumen, se encarga de proteger todas las cualidades del objeto, es decir, pretende que todo lo que tenga que ver consigo mismo permitiendo

[4] Programación orientada a objetos: Nakayama C, M. (s. f.). Guía práctica de estudio 04: Clases y objetos. Guía Práctica de Estudio, http://profesores.fi-b.unam.mx/annkym/LAB/poo_p4.pdf. pp2. Consultado el 14 de marzo de 2024.

[5] "Introducción a la OOP". Moreno Francisco. (s. f.). Fuente: https://kataix.umag.cl/~ruribe/Utilidades/Introduccion%20a%20la%20Programacion%20Orientada%20a%20Objetos.pdf . Consultado el 19 de marzo de 2024.

modificaciones únicamente mediante métodos definidos.

- **Abstracción:**

  “El ser humano tiende a agrupar seres o cosas -objetos- con características similares en grupos -clases-. Así, aun cuando existen por ejemplo multitud de vasos diferentes, podemos reconocer un vaso en cuanto lo vemos, incluso aun cuando ese modelo concreto de vaso no lo hayamos visto nunca”. [Moreno Francisco]

  Es decir, hace referencia a la manera en la que el objeto permite que sean expuestas sus propiedades o métodos mediante operadores de acceso. Además, es una manera en la que se tiende a representar para ser más entendible para nosotros como humanos.

- **Herencia:**

  Para comprender un poco mejor a que se refiere la herencia en lenguajes de programación podemos referir a la siguiente definición que la describe perfectamente:

  “La herencia es el mecanismo para compartir automáticamente los métodos y datos de la “clase base”, añadiendo otros nuevos a las clases derivadas. Es decir, la herencia impone una organización jerárquica entre clases, permitiendo que los datos y métodos de una clase sean heredados por sus descendientes”. [Mazón Olivo Bertha] [6]

  Resumiéndolo, nos permite propagar atributos o métodos que digan que hacer sin tener que describir cómo hacerlo o reescribiendo código. Algo importante a considerar es que para el caso de java no se permite la multi herencia (lenguaje de estudio en este escrito).

[6] “Fundamentos de programación orientada a objetos Java”. Mazón Olivo Bertha. (2015). fuente: http://repositorio.utmachala.edu.ec/handle/48000/6746 .Consultado el 11 de febrero del 2024

- **Polimorfismo:**

  Este puede definirse como:

  “El polimorfismo se refiere al hecho de que una operación de una clase padre puede sobre-escribirse en diferentes clases derivadas … reaccionan al mismo mensaje de modo diferente”. [Mazón Olivo Bertha]

  En otras palabras, describe que un método puede actuar de diferente manera teniendo el mismo nombre, es decir, puede actuar de diversas maneras dependiendo la situación o contexto bajo el que sea utilizado.

Además como complemento es necesario mencionar algunos aspectos fundamentales en cada objeto en los cuales podemos encontrar:

- **Identidad:** Cada objeto debe ser único, es decir, cada objeto será diferente a otro ya sea en propiedades o acciones que pueda realizar.
- **Estado:** Cada objeto debe ser capaz de almacenar el estado. Para este fin, existen atributos, tales como variables de instancias y campos.
- **Comportamiento:** Cada objeto debe ser capaz de manipular su estado, aquí es en donde entran en juego los métodos los cuales permiten modificar algun atributo del objeto.

3.3 Bases de datos

Una forma de definir una base de datos puede ser la siguiente:

“Una base de datos es una colección de información almacenada y organizada de alguna manera con un fin determinado”. [Méndez Cruz Carlos F. (FCA)][7]

Dicho de otro modo, una base de datos puede definirse como un repositorio de información o datos ya sean estructurados o no, que normalmente es almacenada en un sistema informático.

[7] “Apuntes de la materia de base de datos”. Méndez Cruz Carlos F. (FCA Facultad de contaduría y administración de la UNAM). (s.f). Fuente: http://fcasua.contad.unam.mx/apuntes/interiores/docs/98/6/bases_datos.pdf . Consultado el 19 de marzo del 2024.

Nos permite almacenar, leer y actualizar información que se encuentra concentrada en 1 o más lugares.

3.3.1 Bases de datos relacionales

Tomando como base la definición de la universidad de Granada:

"El modelo de datos relacional organiza y representa los datos en forma de tablas o relaciones:los datos en forma de tablas o relación". [DECSAI][8]

Dicho de otro modo, es un modelo en el que se utilizan como estructuras filas y columnas dentro de tablas las cuales permiten el procesamiento y la consulta de datos. Algunas de las acciones muy generales son: acceder, gestionar, modificar, actualizar, controlar y organizar los datos.

3.4 APIs (Interfaz de programación de aplicaciones)

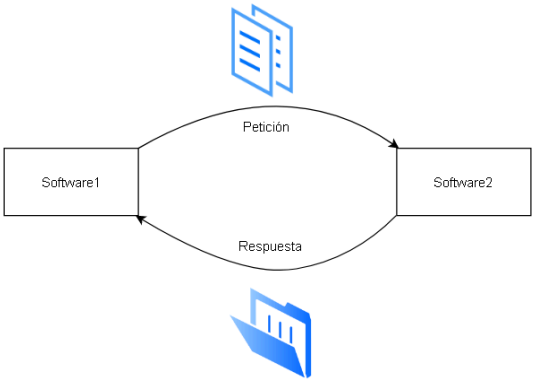
Al hablar de APIs puede ser un poco complejo de entender, sin embargo, una de las mejores definiciones que he encontrado provienen de RedHat la cual las define como:

"*Una API o interfaz de programación de aplicaciones es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones*". [RedHat][9]

Sin embargo, de forma resumida podríamos definir una API como un mecanismo el cual permite la comunicación entre 2 componentes de Software mediante definiciones y protocolos, en este caso la definición de aplicación puede ser considerada como un software, normalmente los 2 servicios que interactúan son software con un enfoque diferente en cuanto a funcionalidad. La interfaz puede verse como un acuerdo entre ambos softwares, dentro de este acuerdo se puede definir como será la comunicación entre peticiones y respuestas, es decir, la definición de cómo hacer y recibir peticiones o respuestas.

[8] "El modelo relacional. Fundamentos del diseño de base de datos". Universidad de Granada DECSAI. (s.f). Fuente: . Consultado el 19 de marzo del 2024

[9] API: ¿Qué es una API y cómo funciona? (s. f.). https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. Consultado el 18 de octubre del 2023



[Figura 1. Ejemplificación API]

A continuación se mencionan tipos de APIs:

- **API SOAP:** Su intercambio de comunicación es mediante mensajes de formato XML, su información viaja de forma estructurada y mediante atributos dentro de sus etiquetas.

- **API REST:** Actualmente este tipo de APIs son las más comunes y consideradas más flexibles respecto a las SOAP, en este escenario el cliente envía solicitudes al servidor como datos, el servidor lee esa petición y acciona funciones internas las cuales permiten devolver información.

3.5 Arquitectura cliente servidor

Para poder explicar a detalle este concepto me gustaría exponer la definición que propone la UNAM:

"*Los usuarios trabajan en computadoras denominadas "sistemas frontales" (front-end) e interaccionan con sistemas servidores denominados "posteriores" (back-end) que proporcionan servicios, tales como el acceso a una base de datos, la gestión de red y el almacenamiento centralizado de archivos. Una red de computadoras ofrece la plataforma de comunicación en la que numerosos clientes pueden actuar con uno o más servidores. La interacción entre la aplicación que ejecutan los usuarios en el front-end y el programa (generalmente una base de datos o un sistema operativo de red) en el back-end se denomina relación cliente/servidor.*" [UNAM][10]

[10] Conceptos de informática: UNAM. (s. f.). Conceptos de informática. Recuperado 10 de febrero de 2024, de http://profesores.fi-b.unam.mx/heriolg/Infdist9.pdf pp.173

A rasgos generales se podría mencionar que esta arquitectura se compone de dos partes, un servidor y múltiples clientes los cuales se conectan al servidor para leer o enviar información necesaria para funcionar, dicho de otro modo, el cliente representa datos, envía y lee información necesaria para detonar acciones del lado del servidor para que este retorne información necesaria.Por lo que el servidor será el encargado de interpretar las solicitudes, realizar un procesamiento o almacenamiento y retornar alguna respuesta al cliente en caso de ser requerido.

En esta arquitectura el servidor deberá exponer un mecanismo que permite a los clientes conectarse a él, por lo general es mediante los protocolos TCP/IP, los cuales permiten una comunicación continua y bidireccional, lo cual permitirá que el cliente pueda enviar y recibir datos del servidor o del servidor al cliente.

3.6 Repositorios y sistema control de versiones

A rasgos generales podemos definir un repositorio como un lugar físico o lógico que contiene todos los archivos necesarios de un proyecto, normalmente está incluido el código, configuraciones y demás.

La siguiente definición puede describir de forma perfecta que es un sistema control de versiones:

“Un sistema de control de versiones es una herramienta capaz de registrar todos los cambios que se realizan en uno o más proyectos, guardando a su vez versiones anteriores del proyecto…”. [Acens][11]

En otras palabras, un sistema control de versiones es una herramienta de software que facilita gestionar los cambios en el código fuente o cambios de un archivo a lo largo del tiempo. A medida que crecen los desarrollos, los sistemas de control de versiones ayudan a los equipos de software a trabajar de forma más ordenada, segura y eficiente. En caso de existir errores o se quiera regresar a un punto en el que un desarrollo funcionaba estos son ideales ya que permiten volver entre versiones y seguir modificando o recuperar información en caso de ser necesaria.

---

[11] “Control de versiones Git y GitHub”. Acen. (s.f). Fuente: https://www.acens.com/comunicacion/wp-content/images/2015/07/git-github-wp-acens.pdf .Consultado el 17 de marzo del 2024

3.7 Patrones de diseño

Los patrones de diseño en el ámbito de programación podrían considerarse como soluciones que son recomendadas ya que han sido probadas y evaluadas para problemas comunes en el diseño y desarrollo de software.

Otra definición muy acertada para ello es la que exponen Gamma Erich y Elm Richard a continuación:

"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces sin hacer lo mismo dos veces". [Gamma Erich, Elm Richard][12]

Algunos ejemplos de patrones de diseño son:

- **Patrón de diseño Singleton:** Garantiza que exista una sóla instancia de una clase durante toda la ejecución del programa por lo que es una instancia de acceso global la cual puede ser consultada en cualquier parte de la aplicación.

  Tal como describe la siguiente definición de la Universidad Autónoma de Madrid:

  "Debe haber exactamente una instancia de una clase, que debe ser accesible a los clientes a través de un punto de acceso conocido". [Guerra Sanchez Esther][13]

- **Patrón de diseño Observer**:

  Este patrón puede ser definido en base a la siguiente definición:

  "Define una dependencia de uno-a-muchos entre objetos de forma que, cuando un objeto cambia de estado, se notifica a los objetos dependientes para que se actualicen automáticamente." [Guerra Sanchez Esther][14]

  Dicho de otro modo, permite que los objetos estén pendientes de los cambios en otro objeto, es decir, en todo momento reciben alertas acerca de lo que sucede en la aplicación y de este modo poder detonar acciones específicas.

---

[12] "Patrones de diseño. Elementos de software orientado a objetos reutilizable". Gamma Erich, Elm Richard. (2003). Fuente: https://profeuttec.yolasite.com/resources/Patrones%20de%20dise%C3%B1o%20-%20Erich%20Gamma.pdf . Consultado el 16 de marzo del 2024.

[13] "Patrones de diseño. Patrones de creación". Guerra Sanchez Esther. (s.f). Fuente: http://arantxa.ii.uam.es/~eguerra/docencia/0809/06%20Creacion.pdf Consultado el 16 de marzo del 2024.

[14] "Patrones de diseño. Patron de comportamiento observer". Guerra Sanchez Esther. (s.f). Fuente: http://arantxa.ii.uam.es/~eguerra/docencia/0809/09%20Observer.pdf Consultado el 16 de marzo del 2024.

- **Patrón de diseño Decorator**:

  Según la definición de Postman Ezequiel:

  “Asigna responsabilidades a un objeto dinámicamente, proporcionando una alternativa flexible a la herencia para extender la funcionalidad”. [Postman Ezequiel]

  Nos dice que este patrón nos permite añadir cualidades adicionales a un objeto durante el tiempo de ejecución, siendo una alternativa a la herencia, además permite combinar varios objetos para crear uno más completo según sea necesario.

3.8 Encriptación de datos

A modo de describir a rasgos generales a que se refiere la encriptación de datos, podemos referirnos a la definición de google, la cual menciona que:

“El encriptado es uno de los pilares fundamentales de la ciberseguridad y sirve para proteger los datos y evitar que se roben, cambien o se vulneren. Para ello, descifra datos en un código secreto que solo se puede desbloquear con una clave digital única”. [Google][15]

Es por ello que es de suma importancia hablar de encriptación de datos ya que es la conversión de datos de un formato legible a un formato codificado, es decir, que los datos cifrados solo se pueden leer o procesar sí y solo sí son descifrados. Esto nos permite enviar información de un extremo a otro con un certeza de que la información no será interpretado por un tercero o bien le será muy complicado hacerlo, hoy en día hacer uso de un cifrado resulta muy importante para ser usado entre usuarios individuales y grandes organizaciones para garantizar la protección de información de usuarios, es por ello que hoy en día uno de los cifrados más recomendados es hacer uso de algún algoritmo de SHA, entre más grande sea su complejidad mejor.

[15] ¿Qué es el encriptado?. Google. (s.f). fuente: https://cloud.google.com/learn/what-is-encryption?hl=es .Consultado el 19 de marzo del 2024.

Para darle una mejor definición a que se refiere un algoritmo hash podemos referir a la definición de IBM la cual menciona:

“El algoritmo hash se designa para minimizar la posibilidad de que dos entradas tengan el mismo valor de hash, que se denomina colisión.

Puede utilizar funciones hash para agilizar la recuperación de los registros de datos (búsquedas simples en una sola dirección) para la validación de los datos ("sumas de comprobación") y para cifrado”. [IBM][16]

A rasgos generales con un algoritmo hash podríamos ser capaces almacenar información y generar su valor hash, sin embargo, de un hash no hay forma de obtener toda su información, a menos que se haga una prueba de fuerza bruta.

Existen varias formas de crear hashes, sin embargo, el algoritmo más usado es SHA-256 ya que se ha identificado una buena relación entre seguridad y tiempo de procesamiento.

Para comprenderlo un poco mejor podemos referir a la siguiente definición:

“SHA-256 es un algoritmo criptográfico de hash en el que los datos de entrada se procesan a través de una sofisticada función matemática, dando como resultado un hash de salida distinto. Este hash actúa como una huella digital, representando de forma única los datos originales”. [SSL Dragon][17]

Además una de las bondades de SHA-256 es que la longitud del hash resultante es siempre igual, no importa lo extenso que sea el contenido por lo que en cuestión de eficiencia es excelente.

3.9 Estimación de tiempo por 3 valores

Es una técnica para la estimación de costos sugerida por el PMBOK[18], la cual consiste en estimar la duración de una actividad utilizando las estimaciones pesimista, más probable y optimista. De hecho también es conocida como PERT: Program Evaluation and Review Technique.

[16] “Funciones Hash”. IBM. (s.f). Fuente: https://www.ibm.com/docs/es/psfa/7.1.0?topic=toolkit-hashing-functions Consultado el 13 de marzo del 2024.
[17] “Qué es el algoritmo SHA-256 y cómo funciona”. SSL Dragon. (s.f). Fuente: https://www.ssldragon.com/es/blog/sha-256-algoritmo/ .Consultado el 12 de marzo del 2024.
[18] PMBOK: Documento que contiene procesos, prácticas recomendadas, terminologías y directrices para una gestión de proyectos eficiente tanto en la rama del software como muchas otras

Hace uso de una distribución beta para obtener el valor más probable en el que se realizará la actividad a analizar. Para hacer uso de ella hay que considerar los siguientes supuestos en cuanto a los tiempos propuestos:

- **Pesimista (P).** La estimación que se le da a la actividad en el peor escenario, normalmente está basado en experiencia propia o sugerida.

- **Más Probable (MI).** Caso ideal en el que se puede desarollar la actividad, normalmente influenciada por una persona con experiencia en el ramo.

- **Optimista (O).** Es la estimación con menor tiempo para finalizarla

Con base a lo anterior podemos definir el mejor tiempo dado por la siguiente expresión:

$$E = \frac{(P + 4MI + O)}{6}$$

4. Datos generales de la empresa

Se trata de una empresa dedicada al rol de retail, es decir, se enfoca en ventas al por menor, normalmente es manejado por venta en unidades al cliente final, si bien tiene sucursales en físico es una verdad que cada vez cobra mayor relevancia el comercio electrónico permitiendo abrir más puertas al público. Ahora bien dentro del modelo de negocio se tienen como objetivo dos perspectivas que serán mencionadas a continuación de forma breve:

- **B2B:** Conocido como Business to Business, está principalmente enfocado a destinar ventas de negocio a negocio, permitiendo asociaciones y/o reventas de los artículos entre negocios donde cada asociación tiene un nivel de contrato el cual le permite acceder a diversos beneficios como descuento en precios, mayoreo, etc.

- **B2C**: Es el modelo más tradicional ya que se basa en Business to Customer, es decir, las ventas se concentran en el negocio a los clientes, de hecho es el modelo más común que llegamos a ver dentro de las empresas de ventas del negocio a

cualquier usuario que entre al sitio con el solo hecho de tener cuenta o incluso participar como invitado.

La bondad de ofrecer ambos enfoques de negocio es ampliar el mercado, generar más ganancias e incluso asociaciones con otras empresas dentro del mismo ramo.

5. Contexto profesional y proyecto

La importancia de tener un rendimiento atractivo en el valor de la empresa es algo que resulta fundamental para cada organización ya que puede marcar una clara diferencia entre seguir en pie con el mercado y actualizar la sana competencia, es por ello que la tendencia actual para los negocios es implementar estrategías de marketing que puedan llegar a una mayor audiencia con base a los sistemas que emplea la organización, tal como puede ser un ecommerce. La capacidad de entender al público puede ser primordial para realizar lanzamiento de estrategias, ofertas ó entender preferencias de los clientes. Hoy en día los datos resultan ser el activo fundamental, por lo que comprender la huella que deja un cliente sobre el negocio puede ser de gran utilidad si se trata de potenciar ventas, utilidades y demás aspectos financieros.

En este caso la empresa a la que le ofrecí mis servicios es un ecommerce enfocado al retail, el cual, ofrece una amplia variedad de productos de diversas categorías desde cosas del hogar, materiales de construcción, artículos tecnológicos, entre otros. Su foco principal se divide en 2 sectores, las tiendas físicas y el ecommerce (Donde yo me desempeñé durante aproximadamente 2 años hasta la fecha). El servicio ofrecido es de Ingeniero en sistemas enfocado en el desarrollo de software para los servidores de dicha empresa la cual mantendremos como anónima.

El soporte brindado se sustentó en 2 áreas, desarrollo Front-End (Parte visual y que percibe el usuario final) y desarrollo Back-End (Lógica que no ve el usuario final pero que define la lógica de negocio). Mi aportación en el proyecto fue para ambas áreas dado que el equipo de trabajo con el que se contaba era pequeño constando del líder técnico, 1 tester y 2 desarrolladores incluyendome, es por ello que, podría decirse que fue un proyecto en el que integré varias áreas de conocimiento por ejemplo: El manejo de base de datos relacionales, el manejo de servidores linux, testing de desarrollos, mantenimiento en ambientes bajos DEV, QA y ocasionalmente STG (Paso previo al productivo), entre otras.

Cabe mencionar que para este punto la empresa ya cuenta con un ecommerce productivo, sobre el cual se planteó la implementación de APIs, por lo que el servicio fue sobre servidores ya existentes de la empresa. Es por eso que este proyecto define un complemento al sitio existente con el fin de recopilar información para el equipo de marketing y así generar nuevas estrategías de venta.

El proyecto que se redacta a continuación fue la base de referencia sobre la cual se aplicarían otras implementaciones de marketing, que más adelante fueron desarrolladas para el análisis de ventas, búsquedas y todo lo relacionado a promocionar el negocio mediante anuncios en otras plataformas tal como Meta (Proyecto abordado en este documento), Pinterest y Tiktok. Si bien estos no son los únicos desarrollos en los que me involucré, considero que son los más completos donde puse en práctica los conocimientos adquiridos durante mi formación académica. La implementación de Pinterest y Tiktok no se abordan en este documento debido a que son implementaciones similares por lo que sería redundante.

Con base a todo lo anteriormente mencionado, el desarrollo del proyecto se basó principalmente en implementar lógica de negocio enfocada al marketing en servidores activos de la empresa a fin de recopilar métricas que pudiesen ser de utilidad en un futuro para idear, diseñar, lanzar y evaluar campañas de marketing, comprensión de sus usuarios incluidas las preferencias, las tendencias, los artículos más visitados, solicitados o demandados dentro del sitio ecommerce.

Cabe resaltar que el análisis de esta información ocurre en tiempo real cuando el usuario está navegando dentro del ecommerce de la organización a modo de poder recopilar datos sobre la huella digital que deja su usuario dentro del servidor. Ahora bien, es importante destacar que el usuario que navega en el sitio puede ser invitado o en su defecto tener su sesión activa (tener una cuenta registrada), en ambos escenarios la interacción del servidor será la misma, capturar la huella y mandar información mediante una solicitud REST mediante el protocolo HTTP.

Dicho de otro modo, el principal objetivo de la organización es implementar lógica en el servidor a modo de recopilar información sobre las acciones de sus usuarios sin la necesidad de interferir en la privacidad de los mismos ni en la experiencia de usuario que obtienen dentro del ecommerce tanto para su versión abierta al público y su otra versión dedicada a otras compañías, es aquí donde entra a juego las siglas B2B y B2C haciendo alusión a Business to Business (Negocio a negocio) y Business to Customer (Negocio a cliente). La principal motivación es generar estrategias de marketing con base a los

datos obtenidos para obtener un mejor beneficio sobre el modelo de negocio, permitiéndoles ser competitivos frente a sus contrapartes en el rol del retail.

6. Requerimientos

A continuación se listan los requerimientos recabados por el líder y que fueron acordados con el equipo de negocio para potenciar la visibilidad al área de marketing.

Requerimientos Funcionales

1. Invocación de API de Terceros para Generar Métricas de Marketing:
    a. RF01: El sistema debe invocar la API de conversiones de Facebook[19] para recopilar información para métricas de marketing cada vez que ocurra uno de los eventos definidos.
    b. RF02: La API debe ser invocada cuando el usuario es invitado o registrado.
    c. RF03: La respuesta de la API debe ser procesada y enviada al sitio de facebook business.

2. Cifrado de Datos:
    a. RF04: Los datos enviados a la API deben ser cifrados utilizando algún algoritmo para garantizar la privacidad del usuario.

3. Implementación en Todas las Versiones del Negocio:
    a. RF07: El sistema debe ser compatible con todas las versiones de las plataformas de negocio existentes (versión legacy V8 y versión v9 próxima a liberar en todos sus ambientes partiendo desde desarrollo hasta productivo).
    b. RF08: La implementación debe mandar la misma información en cada ambiente (Respetar estructura y valores).

4. Determinados Flujos del Usuario:
    a. RF09: El sistema debe invocar la API de métricas de marketing durante los siguientes eventos:
        i. RF09.1: Cuando un usuario visita una página específica (sin importar cual).
        ii. RF09.2: Cuando el usuario hace alguna búsqueda dentro del sitio.
        iii. RF09.3: Cuando el usuario visita la página que describe un artículo

[19] Facebook API Convertions: Menciona que está al día con las regulaciones de CCPA y GDPR, las cuales, en resumen son normas legales las cuales indican cómo debe ser tratada la información en base a leyes de estados unidos. Información recopilada de: https://www.cookieyes.com/blog/ccpa-vs-gdpr/ el 18 de mayo del 2024.
Además en su sitio oficial menciona que están al día con la GDPR:
https://www.facebook.com/business/news/facebooks-commitment-to-data-protection-and-privacy-in-compliance-with-the-gdpr consultado el 24 de mayo del 2024.

iv. RF09.4: Cuando el usuario agrega un artículo al carrito.
v. RF09.5: Cuando el usuario inicia el proceso de compra.
vi. RF09.6: Cuando el usuario selecciona un método de pago.
vii. RF09.7: Cuando el usuario finaliza la compra.
viii. RF09.8: Cuando el usuario se registra al boletín de “Newsletter”.

b. RF10: El sistema debe ser capaz de habilitarse o deshabilitarse mediante un valor en base de datos.

Requerimientos No Funcionales

1. Seguridad:
    a. RNF01: El sistema debe asegurar que todas las comunicaciones con la API de conversiones de facebook se realicen utilizando el protocolo HTTPS.
    b. RNF02: Los datos deben ser cifrados con SHA256 según establece la documentación de la API para garantizar la privacidad del usuario.
    c. RNF03: Sólo los encargados del proyecto pueden hacer modificaciones en la lógica interna.

2. Desempeño:
    a. RNF04: La invocación a la API de terceros debe ocurrir siempre de forma asíncrona para no entorpecer otros procesos.
    b. RNF05: El sistema debe enviar siempre los eventos previstos, en caso de errores seguir las recomendaciones que establece la API.

3. Compatibilidad:
    a. RNF06: El sistema debe ser compatible con navegadores web, tanto dispositivos de escritorio como dispositivos móviles.
    b. RNF07: La implementación debe ser compatible con las distintas arquitecturas y tecnologías de backend existentes y siguientes tomando como referencia el stack de tecnologías actual.

4. Mantenibilidad:
    a. RNF08: El código del sistema debe seguir las mejores prácticas de desarrollo, ser modular y documentado para facilitar el mantenimiento y la evolución futura.
    b. RNF09: El sistema debe incluir pruebas que garanticen que el funcionamiento es como se espera.

7. Tecnologías empleadas

En esta sección se dará un panorama general acerca de las tecnologías empleadas en el desarrollo del proyecto para cada una de las versiones.

Versión 8 B2C y B2B:
Se trata de una versión en la cual tanto el frontend como el backend viven en el mismo servidor comunicados de una manera más monolítica. Las tecnologías usadas se listan a continuación:

BackEnd:

- Java EE basado en Java 8
- Basado en WCS[20] versión 8.1
- Base de datos DB2
- Servlets.

FrontEnd:

- JavaScript
- JSP (Java Servlet pages)
- JSF (Java Servlet Faces)

Básicamente en este stack de tecnologías el backend basado en java tiene una comunicación directa con el frontend mediante los archivos JSP y JSF que permiten obtener valores directos de las peticiones HTTP y variables expuestas desde java las cuales permiten la convivencia entre código HTML, JavaScript, Java y etiquetas definidas en JSF, es por ello que se ve como un sistema monolítico ya que viven en el mismo servidor y existe una relación muy estrecha que no sería posible si el back y front vivieran en servidores separados.

[20] WCS: En resumen se puede definir como una tecnología la cual provee una base para la construcción de eCommerce personalizables para cada empresa facilitando la construcción de lógica y comunicación entre servicios especializados en el comercio electrónico. Consultado en: https://help.hcltechsw.com/commerce/8.0.0/index.html el 24 de mayo del 2024.

Versión 9 B2C:

Se trata de una versión basada en la nube, contenedores permitiendo romper el esquema monolítico y dando una mejor escalabilidad con respecto a su versión inferior. Las tecnologías usadas se listan a continuación:

BackEnd:

- Java EE basado en Java 8 con posibilidad de usar nuevas versiones según sea necesario.
- Basado en WCS versión 9.1
- Base de datos DB2
- Docker
- Kubernetes
- Google Cloud Platform

FrontEnd:

- TypeScript
- REACT

A diferencia de la versión 8, la versión 9 propone una arquitectura donde el back y el front vivan de forma independiente permitiendo que la asignación de recursos sea a demanda según sea necesario, además se tiene la facilidad de tener un orquestador como kubernetes el cual garantiza que los servicios estén funcionando tomando en cuenta instrucciones configuradas en el sistema.

8. Análisis, metodologías empleadas.

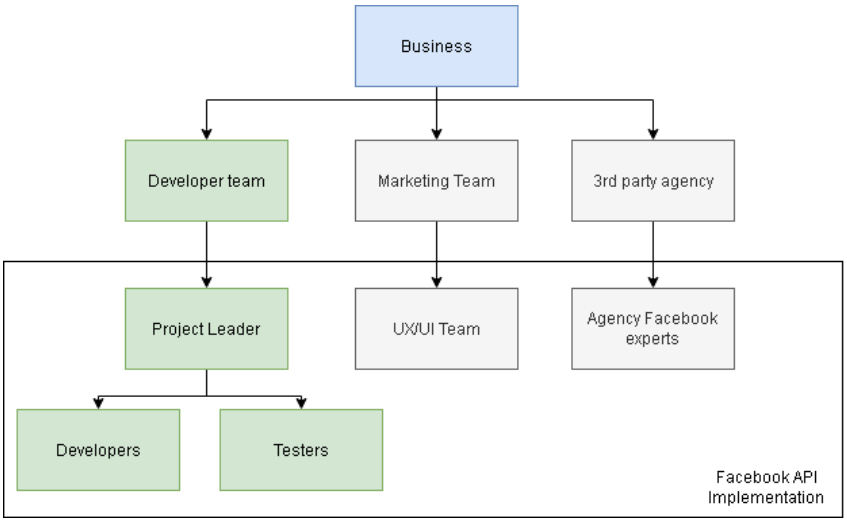
Al llevar a cabo el proyecto el equipo adoptó una metodología similar a SCRUM sin serlo en su totalidad, se tomó como base para llevar a cabo la ceremonia de reuniones cortas diarias (Máximo 15 minutos) en las cuales cada integrante del equipo expone el avance o complicaciones con base al planteamiento, desarrollo o pruebas. Al ser un equipo multicultural esto resultó muy enriquecedor ya que las reuniones se llevaron a cabo en el idioma inglés y siendo 2 desarrolladores se podía estar al tanto de las actividades que se realizaban (en caso de requerirse en un futuro), así mismo era un lugar donde se exponían y aclaraban dudas.

Al fin de cada desarrollo parcial se exponían avances de forma rápida durante la reunión a fin de mostrar la funcionalidad, posteriormente se hacían llegar los comentarios al equipo de QA el cual se encargaba de revisar a detalle diferentes escenarios a modo de garantizar que el desarollo cumplia con todos los puntos previstos en la etapa de requerimientos. Al obtenerse el visto bueno de parte del líder y del equipo de QA el desarrollo se liberaba a un ambiente posterior para repetir el ciclo de pruebas hasta llegar al ambiente productivo. Si en el intermedio se detectaba un fallo o bug en la funcionalidad era reportado al desarrollador para que lo solucionara lo antes posible, una vez corregido el defecto se repetía el ciclo de pruebas pero ahora únicamente sobre los defectos de manera puntual.

Dentro de las ceremonias diarias nos encontrábamos 2 desarrolladores, 1 tester y 1 líder que fungía como SCRUM Master. La comunicación fue sencilla, la líder se encargaba de tocar base con negocio respecto a posibles nuevas implementaciones y tocar base con proveedores en caso de ser necesario para que finalmente compartiera a los desarrolladores los detalles, en caso de existir dudas en el requerimiento se hacían saber y en conjunto se llegaba a un acuerdo o propuesta de estimación de tiempo y esfuerzo en caso de ser necesario. Finalmente todo desarrollo pasaba al tester que se encargaba de probar una buena cantidad de escenarios a modo de garantizar que cada desarrollo funcionara como describía el requerimiento, a continuación muestro una estructura más detallada de las diversas etapas en el desarrollo involucrando otros componentes y equipos que fungían como complementarios pero que enriquecieron el proyecto con su feedback.

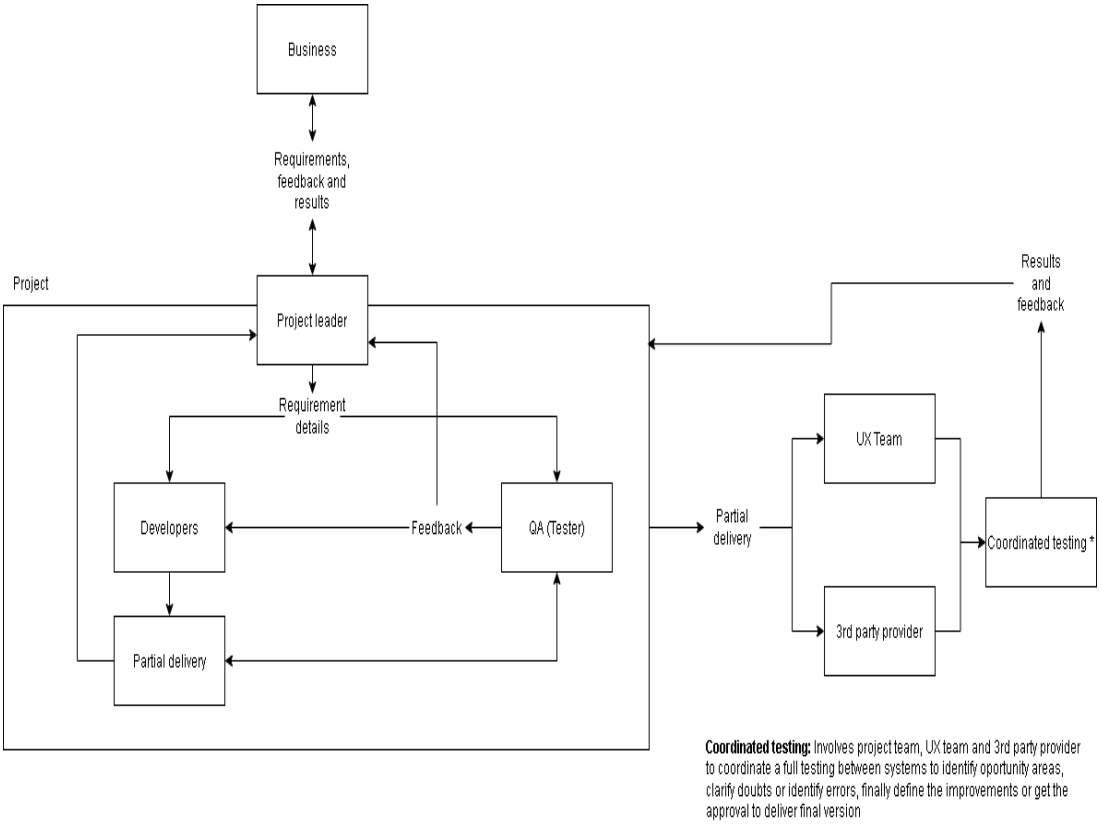
En este proyecto lamentablemente no se tuvo apoyo de arquitectos de software o analistas ya que se trató de un requerimiento directo de negocio el cual no modificaba la estructura entre los componentes fundamentales en el servidor. Además no se tenía en el radar que estas APIs modificaran algo en la arquitectura ni aumento de presupuesto,

únicamente fue monitoreado por el equipo pertinente de operaciones para validar el performance en el sitio y no tuviera algún impacto en el rendimiento del servidor.



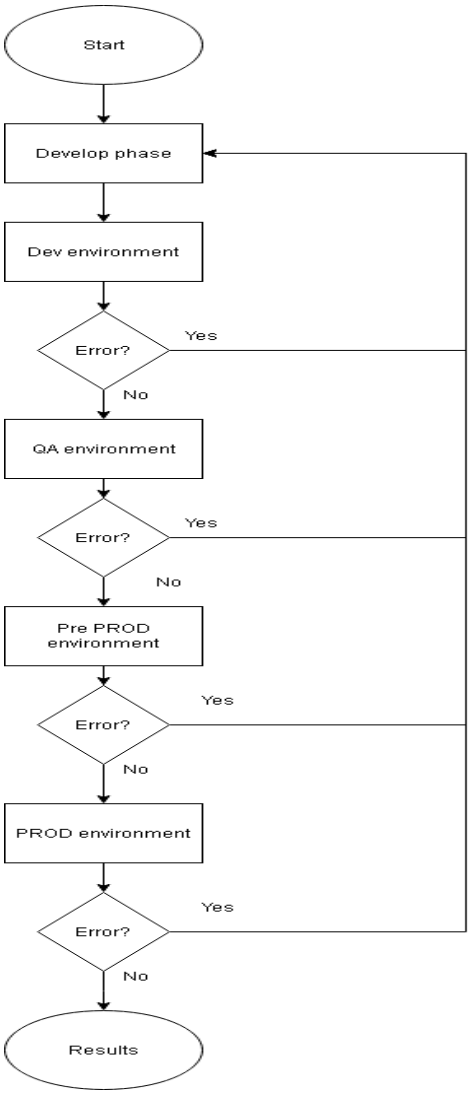
[Figura 2. Organigrama del proyecto a alto nivel]

Así mismo, más adelante se muestra un diagrama a grandes rasgos de la interacción entre equipos y como se llevó a cabo el proyecto considerando diversas etapas.



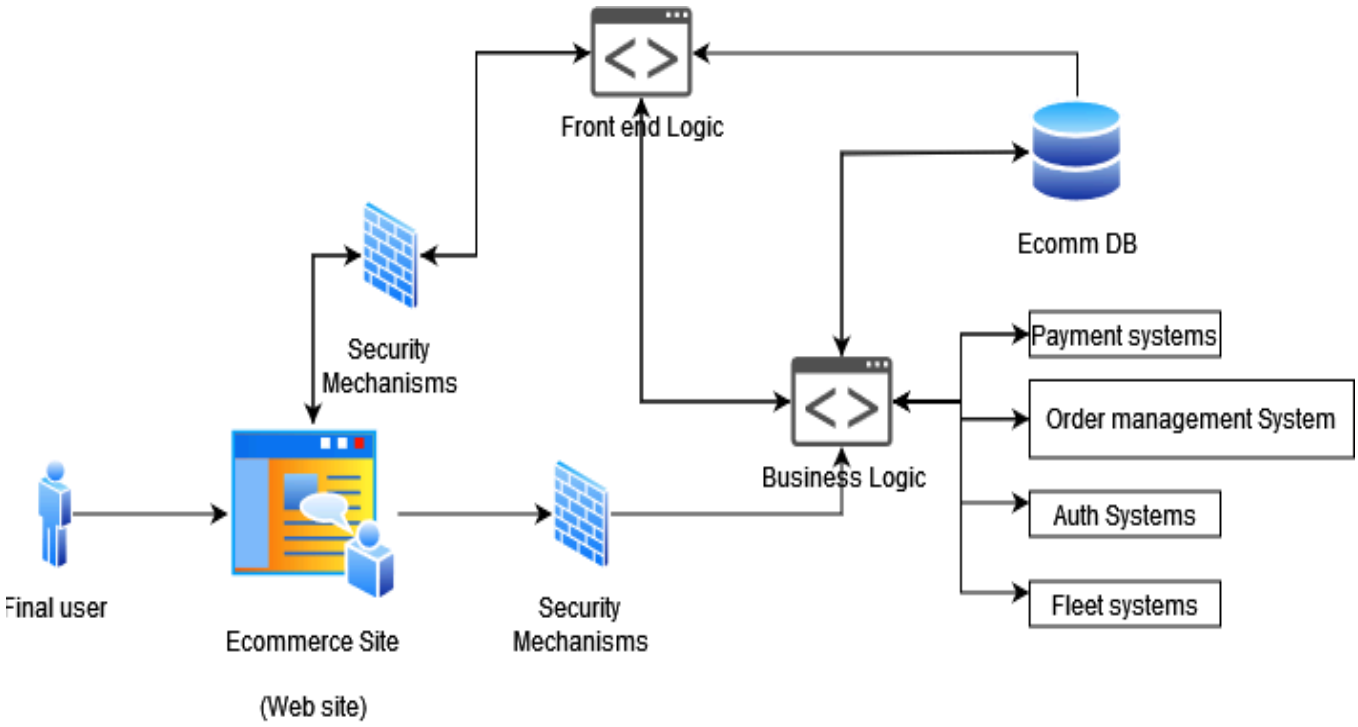
[Figura 3. Diagrama ilustrativo fases del proyecto]

El apartado de entrega parcial se basa en los desarrollos finalizados y versionado, es decir, cada desarrollo va ligado a su respectiva prueba en la cual se define si el funcionamiento es el esperado, en caso de que sí, la funcionalidad pasa de un ambiente a otro para ser validado de nueva cuenta, en caso de existir un error, bug o comportamiento inesperado en algún escenario, entonces el feedback del tester pasa al lider y al mismo tiempo al desarrollador para que comience a replicar el comportamiento y trabaje en una solución. Una vez trabajada la solución se libera una nueva versión del desarrollo y el ciclo se repite hasta que todo funcione como se espera en los escenarios definidos. Principalmente los desarrollos pasaban por ambientes de desarrollo, calidad, pre productivo y finalmente productivo (Cabe mencionar que las liberaciones productivas ocurrían cada mes).



[Figura 4. Diagrama ilustrativo liberación entre ambientes]

Para comenzar con el desarrollo de este proyecto fue indispensable entender un poco más acerca de la arquitectura que maneja el negocio en cuanto a sus sistemas informáticos para de ese modo definir el modo de trabajar y saber el método exacto de extraer la información requerida, a continuación plasmaré un diagrama a muy alto nivel del modo en el que se comunican los sistemas y el que se tomó como base para determinar el punto de partida.



[Figura 5. Diagrama ilustrativo lógica de negocio]

Con base al panorama general de la arquitectura y del funcionamiento a muy grandes rasgos también es indispensable hablar un poco de la arquitectura sobre la cual funciona tanto en la versión legacy (v8) como la nueva versión a ser liberada (v9).

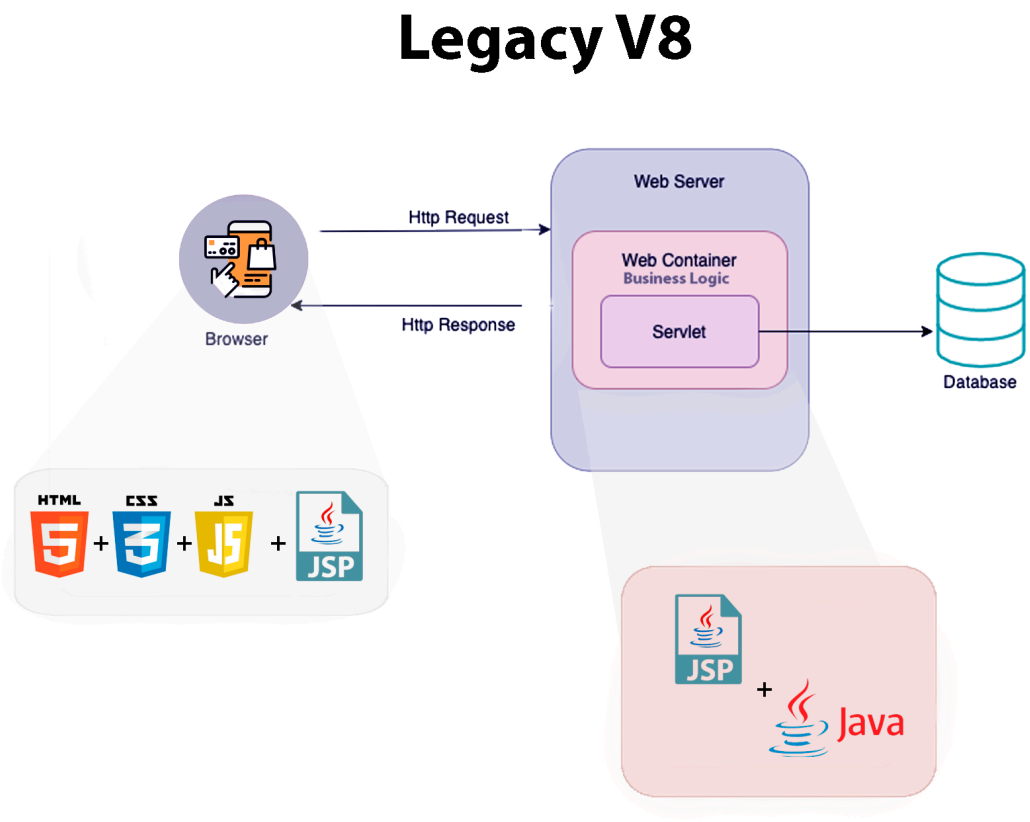
Versión 8:

Al hacer referencia a V8 o versión Legacy se encontraba un stack de tecnologías meramente basada en JAVA en su gran mayoría, en la que tanto front como back son manejados mediante lógica de Servlets desarrollados en mismo lenguaje de programación, una de las grandes ventajas al usar este enfoque, fue que todo podía ser unificado desde un mismo servidor el cual es capaz de comunicar la interfaz del usuario con el procesamiento robusto que ocurre de lado del servidor. En esta versión se tienen 3 áreas principales:

La interfaz web con la que interactúa el usuario basada en HTML, JavaScript y JSPs los cuales permiten la comunicación directa con código JAVA ya sea mediante su sintaxis o mediante una biblioteca llamada JSTL que a grandes rasgos es un conjunto de etiquetas las cuales encapsulan la lógica para generar una conexión entre el frontend y el backend.

La siguiente capa son los servlets los cuales actúan como un intermediario entre los JSP (Plantillas que muestran contenido en el front y pueden obtener información de lógica JAVA o JSTL) estos elementos son bastante prácticos ya que interpretan información proveniente del cliente y las transforma a peticiones hacia el servidor haciendo uso del protocolo HTTP, además permiten tanto peticiones como carga de información en en términos prácticos son el intermediario a la tercera etapa.

Finalmente en la tercera etapa se encuentra la lógica del servidor y lógica de negocio la cual interpreta las peticiones provenientes del servlet para retribuir información requerida desde un simple texto hasta información aún más detallada del ecommerce, normalmente esta etapa es la que se encarga de la persistencia de datos y la consulta de los mismos. En este caso la persistencia se encuentra en bases de datos estructuradas de DB2 donde se alojan todos los productos, configuraciones e información extra dentro del sitio. A continuación se muestra un diagrama que representa de forma gráfica el funcionamiento de la versión legacy y el stack tecnológico.



[Figura 6. Diagrama ilustrativo versión 8]

Versión 9 orientada a la nube y mayor escalabilidad:

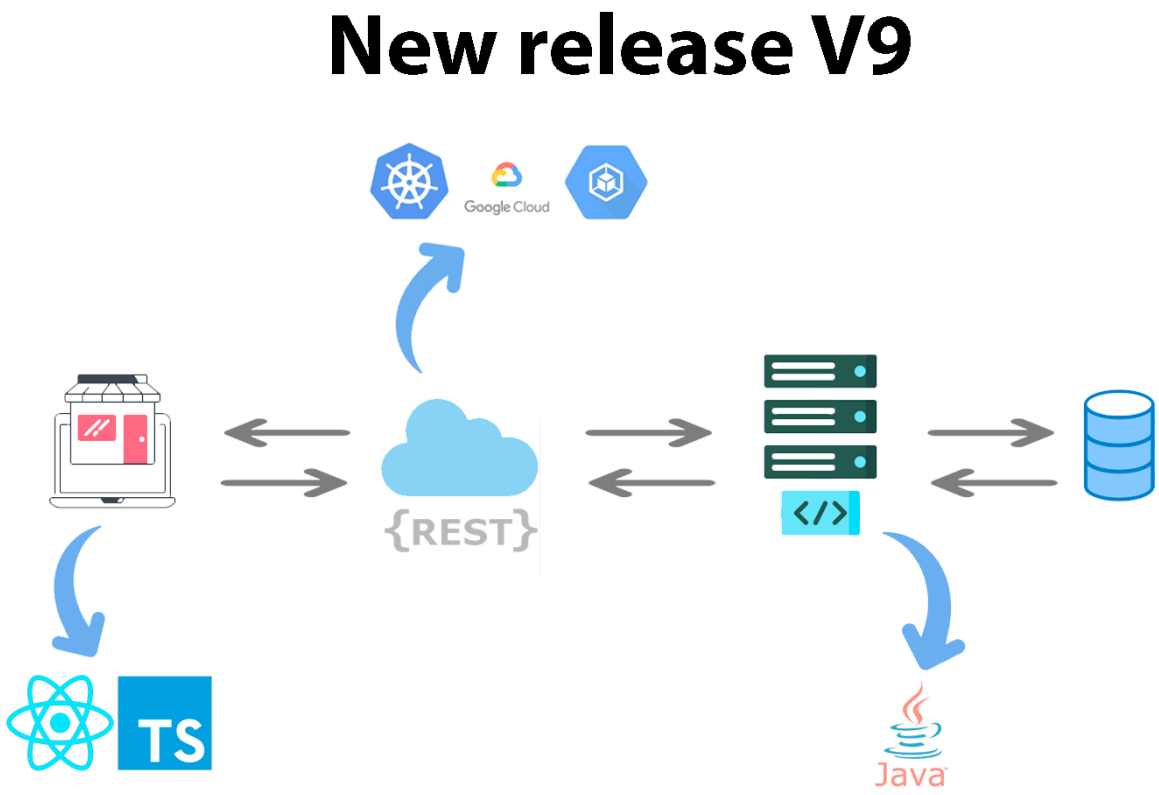
Al hacer referencia a V9 se hace una referencia a un stack de tecnologías basadas en la escalabilidad donde se emplea el uso de la nube para obtener mejores resultados tanto en desempeñó como escalabilidad y disponibilidad, dentro de esta se encuentran conceptos como contenedores y kubernetes en la nube, GCP (Google Cloud Platform) para ser un poco más preciso. Lo que diferencia esta versión con la legacy no solo es el uso de la nube sino también la separación del frontend con el backend. Dentro de esta arquitectura también encontramos 3 fases:

La primera fase es el frontend, aquella interfaz con la que interactúa el usuario final desarrollada con el framework de REACT y diferentes conceptos del mismo que permiten la comunicación vía REST con el backend mediante el protocolo HTTPS.

La segunda etapa es la nube por la cual se maneja toda la seguridad mediante firewalls los cuales permiten dar una capa de seguridad de las peticiones entre el front y el

backed, además es la que provee la infraestructura donde están alojandose los servidores.

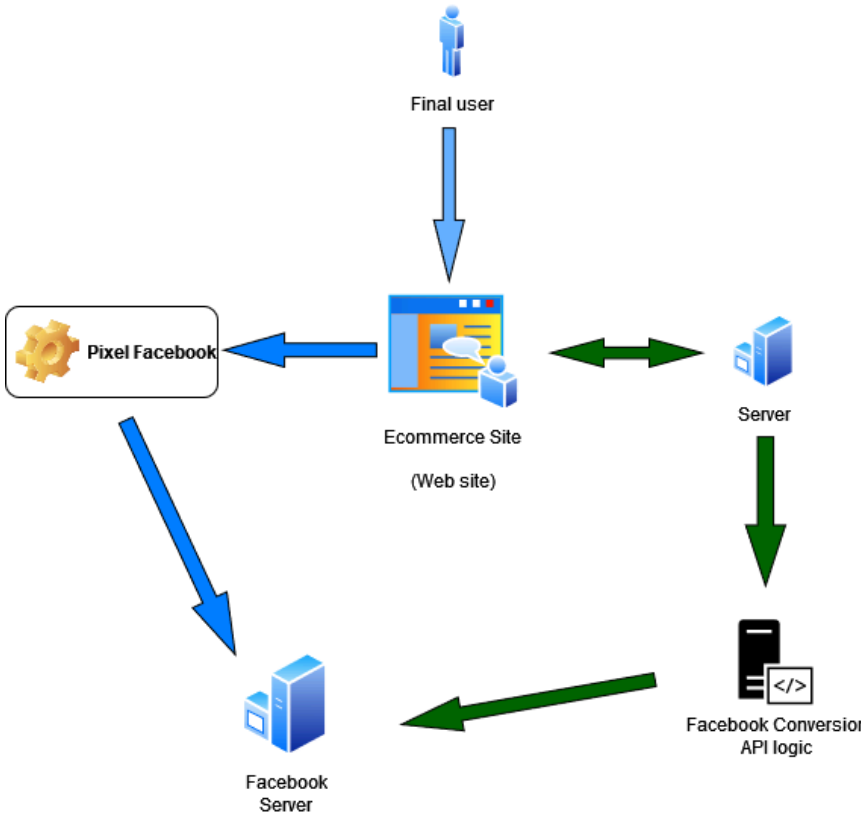
Finalmente en la tercera etapa se encuentra la lógica del servidor y lógica de negocio la cual interpreta las peticiones provenientes del frontend con las peticiones REST ya que ahora todos los servicios necesarios son expuestos para ser consumidos mediante APIS de caja o personalizadas por el negocio. De igual manera que en su contraparte legacy, en esta capa encontramos el manejo de persistencia con bases de datos estructuradas DB2 donde se alojan todos los productos, configuraciones e información extra dentro del sitio. A continuación se muestra un diagrama que representa de forma gráfica el funcionamiento de la versión legacy y el stack tecnológico. Por último dentro de este servidor web se implementó el consumo a otras APIS a servidores de búsqueda los cuales agilizan el procesamiento y tiempo de carga.



[Figura 7. Diagrama ilustrativo versión 9]

Una vez recabados los requerimientos del proyecto y teniendo contexto acerca de la visión, el primer paso fue comenzar la revisión de la documentación existente sobre el servicio que el negocio quería implementar, en este caso se trata de un servicio de terceros el cual ofrece una gestión más sencilla de la información que a largo plazo puede representar un beneficio al momento de querer visualizar información, la creación de campañas de marketing e incluso comprender cómo enganchar mejor la atención de los usuarios a su favor.

El principal funcionamiento de este servicio de terceros fue la implementación de una API (API de conversiones de Facebook) basada en la arquitectura de eventos, la cual se enfoca en el cambio de estado o actualización de un elemento. Por ejemplo, cuando el usuario agrega un artículo al carrito dentro del ecommerce o al visitar cualquier página. En otras palabras, se detona una notificación cada que ocurre una actualización sobre un elemento dado. A continuación se muestra un diagrama a alto nivel, que tiene como objetivo representar el flujo de la API.



[Figura 8. Diagrama ilustrativo API Facebook]

Para el contexto del proyecto se plantearon diferentes eventos los cuales se mencionan brevemente a continuación:

- Vista de pantallas: El cual se encarga de identificar qué secciones o páginas dentro del sitio web han sido visitadas por el usuario en la sesión.

- Búsqueda de artículos: Hace referencia cuando el usuario tiene interés en un artículo en especial y lo busca dentro del ecommerce.

- Detalle del artículo: Ocurre cuando el usuario entra a ver los detalles del artículo.

- Agregar al carrito: Ocurre cuando el usuario agrega un artículo al carrito porque realmente le interesa.

- Inicio de compra: Es cuando el usuario inicia el proceso de compra y llena sus datos personales y de envío.

- Confirmación de compra: Permite saber si el usuario concretó la compra, el monto y los artículos involucrados.

- Suscripción a noticias: Permite identificar si un usuario se registró al “Newsletter”

Tomando como base los eventos anteriores es importante mencionar que cada uno implicó su propia lógica en el backend la cual se encargará de recopilar información detallada con ayuda de otros métodos, comandos, componentes o incluso base de datos. Para poder llevar a cabo esa interacción entre el ecommerce y el backend se requirió hacer uso del protocolo HTTP para consumir servicios REST en los cuales se alojó toda la lógica que desarrollé en ell lenguaje de programación JAVA haciendo uso de la programación orientada a objetos, en dicha lógica se pueden encontrar métodos cuyo enfoqué fue el de obtener a detalle la información de la página en curso, detalles de un artículo, los detalles de la tienda seleccionada así como la geografía, entre otros. De este modo toda la lógica corre de forma asincrónica permitiendo que el procesamiento siga sin tener que impactar la experiencia del usuario y sin representar un mayor tiempo de carga en los flujos habituales de búsqueda, visualización o proceso de pago.

Como complemento cabe mencionar que al hacer uso de servicios REST el formato JSON fue fundamental a la hora de implementar la lógica, dicho formato se dividió principalmente en 3 secciones:

- Primera sección: Correspondiente a descripción del evento en la cual podemos encontrar el nombre del mismo, la hora en la que se detonó, e identificador único.

- Segunda sección: Corresponde a información cifrada en formato SHA256 la cual garantiza un cifrado completo de extremo a extremo para evitar intrusion por un tercero, además garantiza la privacidad de datos que pudiesen llegar a ser sensibles.

- Tercera sección correspondiente a detalles específicos sobre el sitio visitado o detalles de un artículo del ecommerce.

A continuación se muestra una estructura general de cómo luciría un objeto JSON usado en el petición a la API.

```
{
  "nombre_evento": "VisitaPagina",
  "hora": "hora_formato_unix",
  "info_usuario": {
       //Aquí se detalla información cifrada en SHA256
  },
  "informacion_adicional": {
       //Detalle del sitio o articulos vistos
  },
  "identificador": "Id unico"
}
```

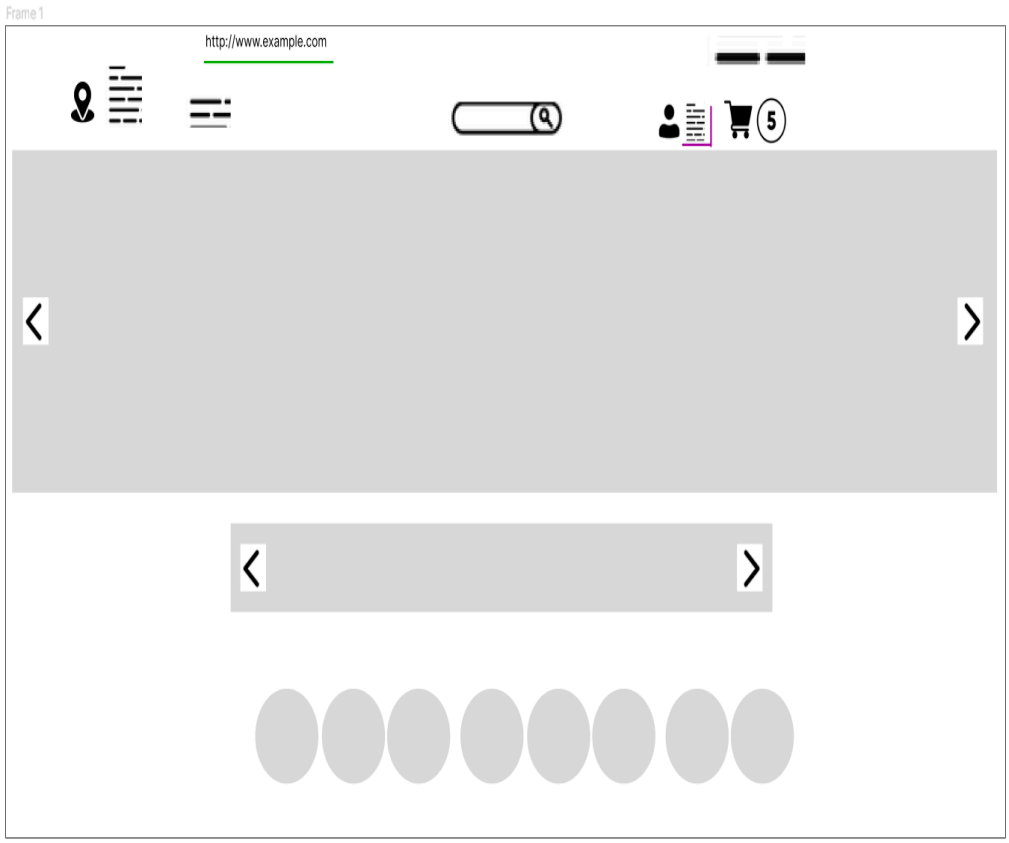
[Figura 9. Estructura general de un evento]

Es importante resaltar el hecho de que cada evento tiene su propio identificador único con el fin de diferenciarlo y ser usado por la otra herramienta que trabaja en paralelo llamado pixel que a rasgos generales es un Script que vive en el front end el cual sensa con puro JavaScript las vistas con las que el usuario interactúa y es el complemento perfecto a la lógica trabajada ya que actúa como un verificador de que un evento está ocurriendo, la idea es usar el pixel como un copiloto y comprobar la implementación en el backend, además sirve como un vigilante para identificar si existió un evento duplicado, de ser así el servidor de facebook lo desecha para que influya en los últimos análisis que serán usados por el equipo de marketing.

A continuación se muestran algunas imágenes y descripciones ilustrativas a modo de representar un poco de algunos eventos empleados.

PageView:

Es el evento con mayor presencia en toda la implementación ya que se detona de inmediato cuando el usuario visita un página dentro del sitio, este evento lo que hace principalmente es recopilar información de que página ve el usuario en ese momento, datos de su cuenta en caso de existir, información de cookies creadas durante la implementación como el event_id y otras cookies (Valores leídos desde la lógica del backend) que pueden existir si el usuario interactúa en el sitio en base a un anuncio que vio en redes sociales u otro sitio web, la hora en la que se detonó el evento y más información necesaria para la llamada a la API como el custom data el cual normalmente incluye información de productos.



Usuario Invitado:

```
Event={
  name_event = "PageView",
  event_id = "170768207612300213",  //Unixdate + Random
  fbc= "", //Cookie value
  fbt= "", //Cookie value
  user_info = {
    name=""
    last_name="",
    phone="",
    email="",
    agent_name="Google Chrome",
    country="MX",
    custom_agent="111.178.29.184"
  }
  resource_name="www.example.com",
  Custom_data=[]  //Información de artículos
}
```

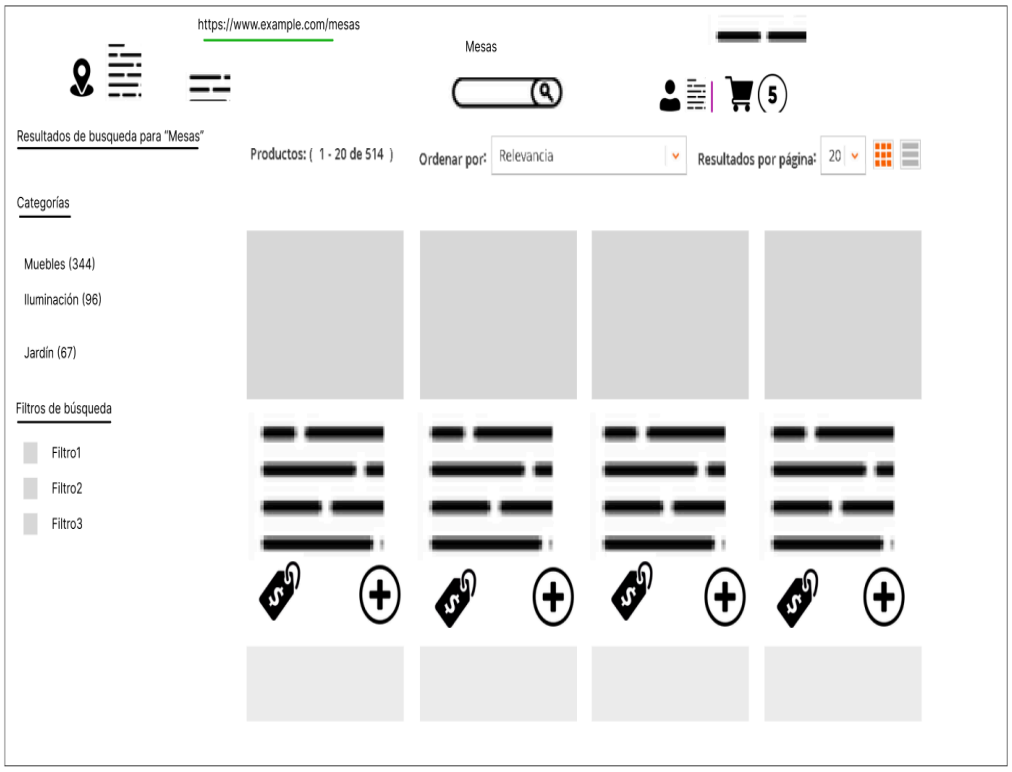
Usuario registrado:

```
Event={
  name_event = "PageView",
  event_id = "170768207612300213",  //Unixdate + Random
  fbc= "", //Cookie value
  fbt= "", //Cookie value
  user_info = {  //SHA256
    name="A10415383402E005336F225FD72E5CFA43E1"
    last_name="8FA1DDDD53606CEB933C5C6A12E714ED4",
    phone="",
    email=C3641F8544D7C02F3580B07C0F9887F0C6A2",
    agent_name="Google Chrome",
    country="MX",
    custom_agent="111.178.29.184"
  }
  resource_name="www.example.com",
  Custom_data=[] //Información de artículos
}
```

[Figura 10. Vista y representación evento PageView]

Search:

Es uno de los eventos más importantes ya que permite recopilar información de artículos o términos más buscados en el sitio web, posee un estructura similar al evento pasado con diferencia de que este hace diversas llamadas en el servidor para poder identificar el valor de búsqueda que efectuó el usuario en el ecommerce. Para poder obtener este valor se desestructuran varios objetos que forman parte del request en el sitio.



Usuario Invitado:

```
Event={
  name_event = "Search",
  event_id = "170768207612330213",  //Unixdate + Random
  fbc= "", //Cookie value
  fbt= "", //Cookie value
  user_info = {
    name=""
    last_name="",
    phone="",
    email="",
    agent_name="Google Chrome",
    country="MX",
    custom_agent="111.178.29.184"
  }
  resource_name="www.example.com/mesas",
  Custom_data=[
    search_term = "Mesas",
    Filters= "NA", //Includes de filters applied
    conditions= "NA"
    sort="Default"
  ]  //Información de artículos
}
```

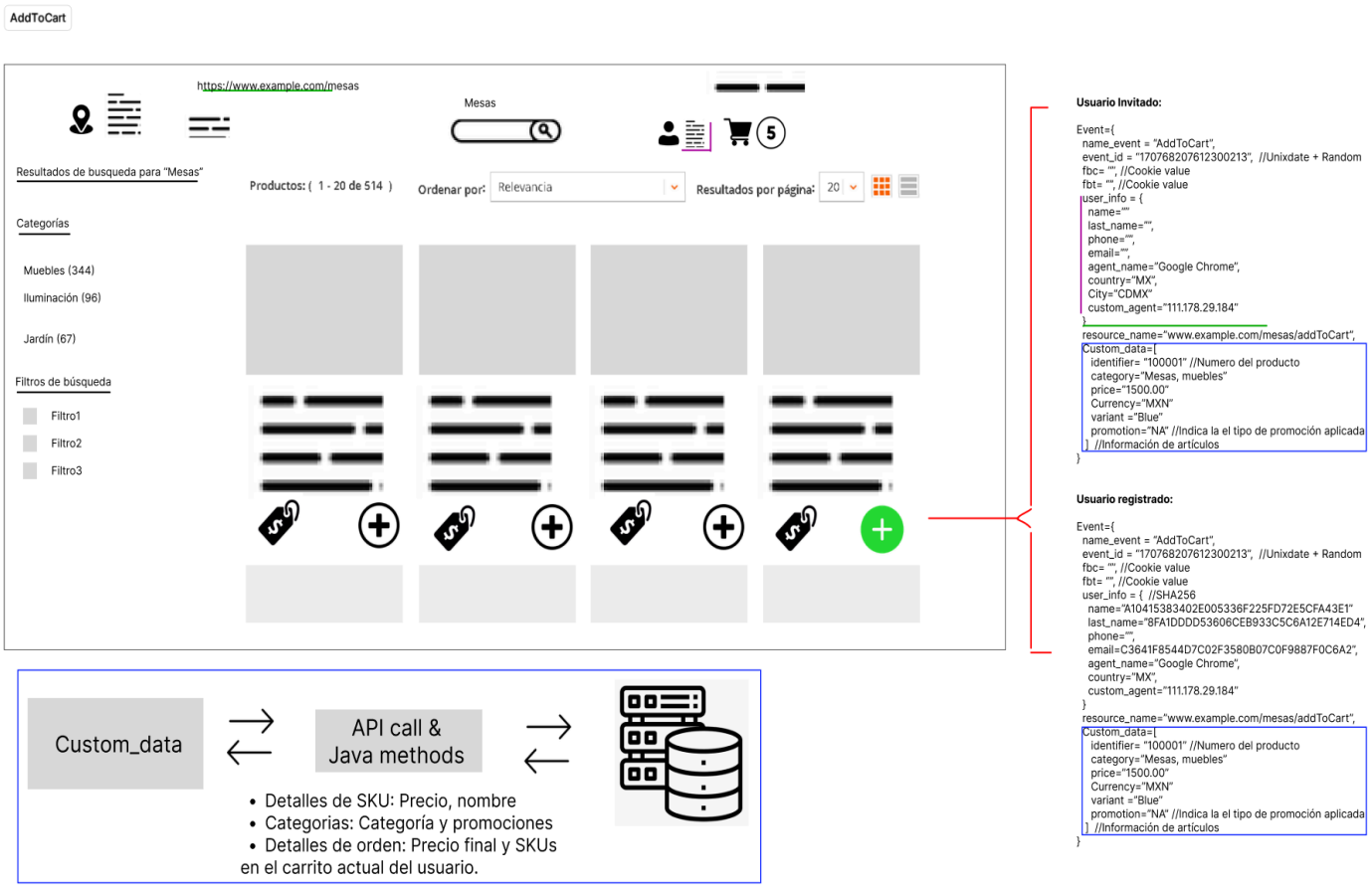
Usuario registrado:

```
Event={
  name_event = "PageView",
  event_id = "170768207612300213",  //Unixdate + Random
  fbc= "", //Cookie value
  fbt= "", //Cookie value
  user_info = {  //SHA256
    name="A10415383402E005336F225FD72E5CFA43E1"
    last_name="8FA1DDDD53606CEB933C5C6A12E714ED4",
    phone="",
    email=C3641F8544D7C02F3580B07C0F9887F0C6A2",
    agent_name="Google Chrome",
    country="MX",
    custom_agent="111.178.29.184"
  }
  resource_name="www.example.com",
  Custom_data=[
    search_term = "Mesas",
    Filters= "NA", //Includes de filters applied
    conditions= "NA"
    sort="Default"
  ]  //Información de artículos
```

[Figura 11. Vista y representación evento Search]

AddToCart:

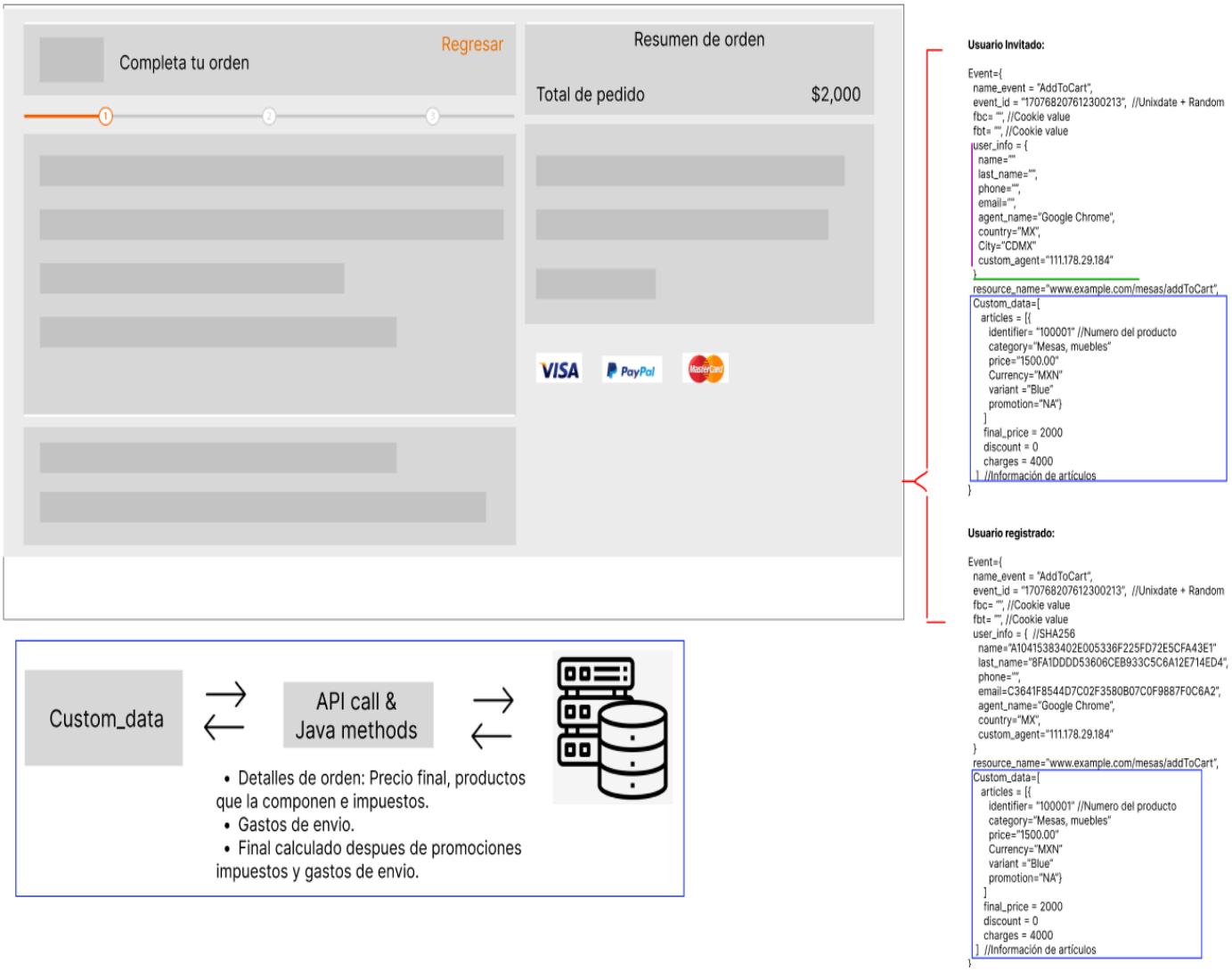
Permite obtener todos los detalles del artículo agregado al carrito entre sus detalles se puede encontrar el nombre, las categorías, el precio, la moneda y más detalles de la orden, para obtener toda esa información se consultó una serie de APIs y métodos JAVA en el backend para consultar de la base de datos y llamar lógica que calcula precios después de impuestos y/o promociones, de ese modo se construyó el JSON para ser enviado en la API de facebook, este evento permite dar una idea a negocio de qué productos son agregados juntos o que es lo que los usuarios llegan a armar como combo y en base a ello analizar posibles ofertas en un futuro. Además como complemento, esta información permite dar una idea de cuál podría ser el monto de compra según regiones u otros factores.



[Figura 12. Vista y representación evento AddToCart]

OrderShipping - Inicio de compra:

En esencia este evento se encarga de obtener todo el detalle de la orden hasta el punto en donde el usuario está llenando su información restante para identificar si es para envío o recolección en tienda, el principal funcionamiento radica en obtener toda la información desde artículos, precios, impuestos, ofertas y cargos extra de envio en la orden tomando de referencia información localizada en la base de datos que se encuentra repartida en diversas tablas.



[Figura 13. Vista y representación evento PageView]

Seguridad e implementación:

Como se vió previamente en algunos de los eventos se hace manejo de información la cual puede considerarse como sensible, es por ello que “meta” solicita hacer el envío de esa información cifrandola en formato SHA256 para evitar que un intermediario pueda leer la información, de esta forma y mediante la comunicación HTTPS que existe entre servidores se puede garantizar una mayor seguridad en la información. Además, en su sitio oficial se menciona que hoy en día “meta” está al margen cumpliendo las normas de la GDPR (Reglamento general de protección de datos) el cual, en resumen propone normas para el tratamiento de datos personales.

En caso de que no se envíe la información en el formato esperado, meta hace varios avisos mediante la aplicación de meta business center la cual sugiere corregir el formato lo antes posible, definiendo un tiempo para que la empresa corrija los datos y no sea suspendida al no tratar la información como se espera, este tipo de suspensiones pueden repercutir las métricas o incluso una suspensión de cuenta.

A continuación se muestra un ejemplo donde se reportan los problemas y más información detallada por ejemplo una posible suspensión de eventos en caso de infringir las normas de cifrado o envío inapropiado de información:

[Figura 14. Avisos de seguridad y problemas detectados]

Planificación:

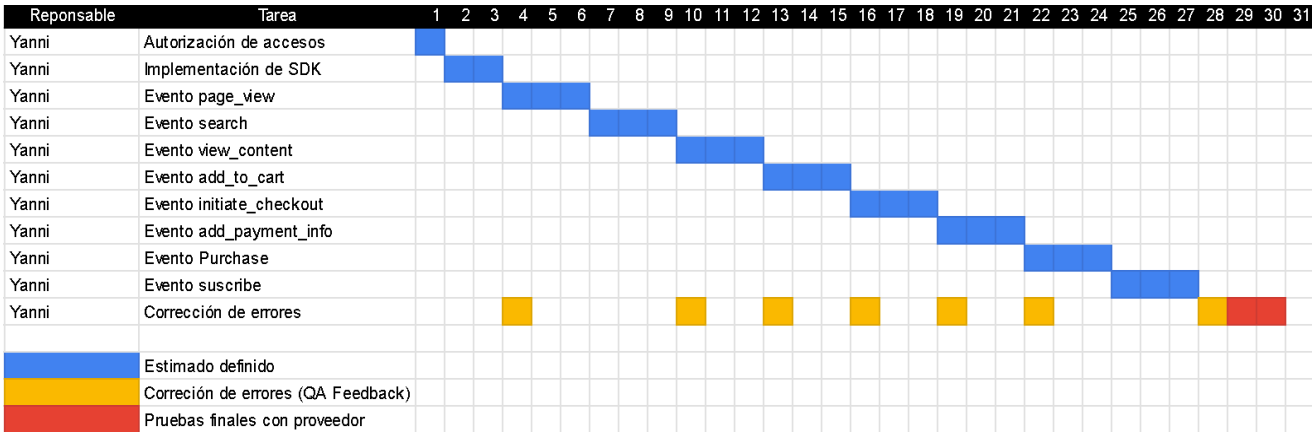
Si bien es muy importante el detalle del desarrollo, es cierto que la estimación de tiempo y esfuerzo fue un punto primordial para poder hacer una buena entrega de la

implementación, es por ello que basado en experiencias previas dentro de la compañia, algunos desarrollos y la amplia colaboración del equipo multicultural proveniente de México, Estados unidos e India pude llegar a una estimación realista que permitió entregar a tiempo el desarrollo. Para realizar la planificación se utilizó la técnica llamada “Estimación por 3 valores” descrita en una famosa guia llamada “PMBOK Guide” dedicada a planificación y estimación de proyectos, dicha técnica se basa en proponer 2 posibles tiempos y obtener un esfuerzo probable. Para ello se definieron tiempos optimistas tomando en cuenta comentarios de compañeros más experimentados y experiencias propias por lo que el resultado optimista y pesimista estaba basado en otros desarrollos, que si bien no eran iguales, tenían al menos 1 cosa en común, dicho eso, a continuación se expone la estimación realizada para 1 ambiente de forma preliminar:

Actividad	Predecesora	Duración Días			Tiempo esperado (te)
		Optimista (a)	Más probable(m)	Pesimista(b)	
Autorización de accesos	-	3	4	5	4
Vista de pantallas	Autorización de accesos	2	3	4	3
Busqueda de artículo	Vista de pantallas	2	3	4	3
Detalle de artículo	Vista de pantallas	2	3	4	3
Agregar al carrito	Detalle de artículo	3	4	5	4
Inicio de compra	Agregar al carrito	3	4	5	4
Confirmación de compra	Inicio de compra	2	3	5	3
Integracion Event_id	Autorización de accesos	4	5	7	5
Pruebas generales	Finalización de cada evento	1	1	1	1

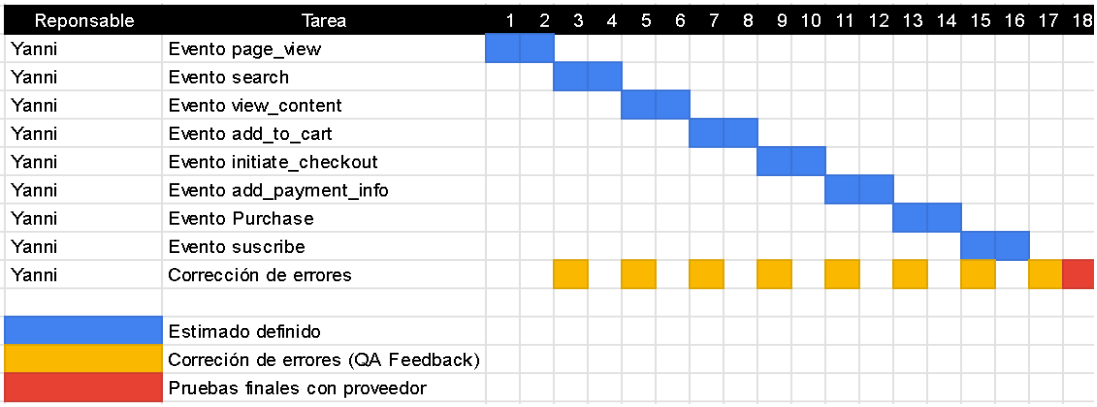
[Figura 15. Estimación de costos usando técnica de estimación por 3 valores]

Llevándolo a la práctica mediante un diagrama de Gantt teniendo en cuenta tiempos y pruebas se llegó a una planificación en conjunto con el líder el cual sugirió priorizar tiempos para cada evento, evitando que se trabajaran en paralelo y así tener una mejor organización y tener espacio para las pruebas pertinentes:



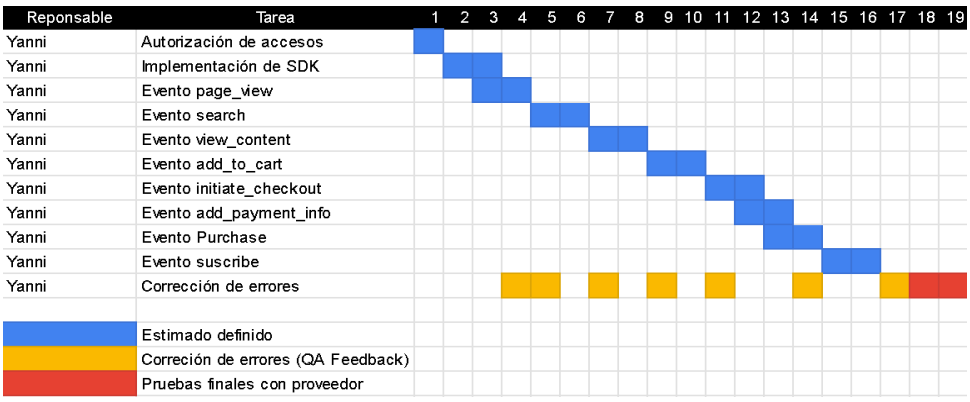
[Figura 16. Diagrama de Gantt implementación API v8]

Dicha estimación dio como resultado un total de aproximadamente 30 días hábiles para finalizar el desarrollo en el ambiente. Cabe resaltar que en un inicio la implementación sólo fue planeada para 1 ambiente (la versión V8-B2C es decir, usuarios comunes), posteriormente se solicitaría la implementación en la versión profesional (V8-B2B , es decir, venta a otros negocios por mayoreo y contrato). Debido a que el desarrollo de la versión profesional compartía lógica en común con la versión V8-B2C, se llegó a un acuerdo de validar los eventos y aplicar únicamente los cambios necesarios:



[Figura 17. Diagrama de Gantt implementación API V8 versión PRO]

Con el paso del tiempo y tras más desarrollos, la compañía decidió tomar la decisión de hacer una migración de la versión 8 a la versión 9 con el uso de nuevas tecnologías, en el frontend basándose en REACT y con similitudes en el backend haciendo uso de JAVA. Es por ello que este proyecto se volvió a trabajar pero realizando la migración a las nuevas tecnologías sin alterar los requerimientos previos. Para llevar a cabo la implementación se basó en lo que se desarrolló en la versión v8 y se adaptó a las nuevas formas de trabajo por lo que se volvió a hacer una estimación, la cual ahora tomaba en cuenta la experiencia, lo cual permitió reducir el tiempo, además que pudo reutilizarse parte de la lógica del backend dando como resultado la siguiente estimación:



[Figura 18. Diagrama de Gantt implementación API]

Como se aprecia en los diagramas, se definieron 3 etapas para garantizar el correcto funcionamiento del desarrollo, el apartado azul hace referencia al tiempo que se invirtió por cada desarrollo haciendo pruebas en el dispositivo local y ambiente de desarrollo. El apartado amarillo describe el tiempo estimado que tomó el equipo de QA para probar ese evento y en caso necesario levantaba bug o incidentes para que el desarrollador los trabajara en paralelo con el desarrollo del nuevo evento, de este modo se garantizó una comunicación eficiente entre calidad y codigo permitiendo mejoras mientras se trabajaba en nuevos eventos. Finalmente la sección de color rojo hace referencia a pruebas en conjunto entre el equipo de desarrollo y el proveedor (mediante llamada) para garantizar que todos los parametros y funcionalidades trabajaran como se esperaba, en caso de detectar incidencias se trabajaba antes de agendar una nueva reunión.

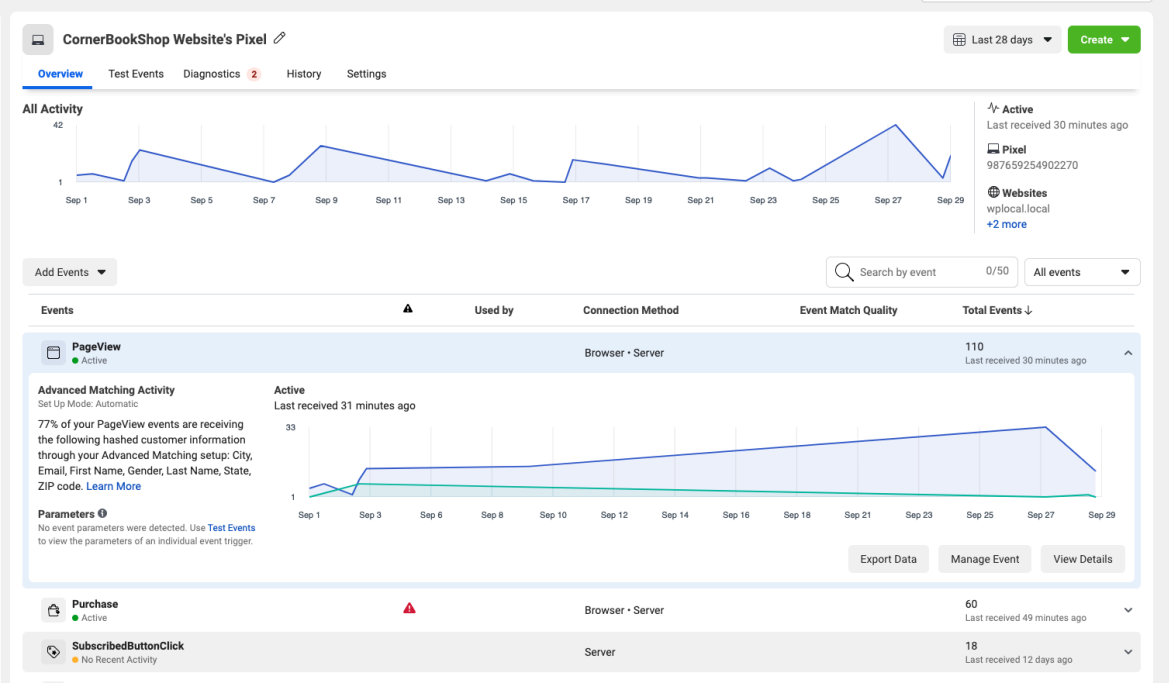
La manera en la que se llevó a cabo el seguimiento de este desarrollo fue basándose en una especie de metodología muy parecida a SCRUM ya que diariamente se tuvieron sesiones rápidas donde se expuso el avance, las dudas y/o problemas que se fueran presentando a modo que el líder técnico estuviera al tanto de las necesidades en el desarrollo, de este modo la confianza del equipo se fortaleció y se benefició. Es importante mencionar que se realizaron entregas parciales cada 2 semanas a modo que el equipo QA pudiera validar eventos y si hubieran áreas de mejora se trabajaran y corrigieran de ser necesario, de este modo la implementación fluyó sin retraso.

9. Resultados y aportaciones

Como se mencionó previamente, la mayor parte de la implementación de la API de facebook fue en el backend del servidor el cual está principalmente construido y compilado con JAVA EE basado en WebSphere tomando como base los desarrollos de la tecnología WCS (La cual proporciona una base para desarrollar nuevos eCommerce), la cual provee una solución especializada en ecommerce, de la cual cada compañía es libre de realizar modificaciones acorde a su lógica de negocio. Cabe mencionar que para llegar a una solución final se siguió un robusto proceso de calidad pasando primero por ambientes de prueba los cuales fueron; Desarrollo (Dev), Calidad (QA), Stagging (Pre-productivo) y finalmente PROD (Sitio productivo). Para cada uno de los ambientes se armaron y ejecutaron matrices de prueba a modo de garantizar que a implementación era correcta en diferente aspectos desde inspeccionar a detalle cada atributo o variable, detonarlos en diferentes páginas del sitio o diferentes casos de uso, este proceso fue ejecutado en cada ambiente y con ello se obtuvo una mejora continua que permitió hacer una entrega la cual cumpliera todos los requerimientos establecidos al inicio del proyecto.

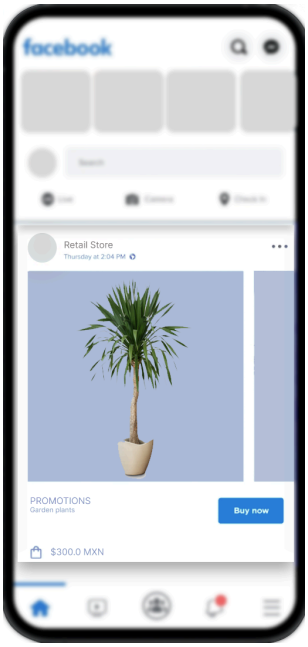
Como resultado de esas buenas prácticas se tuvo un lanzamiento a producción con éxito para la versión v8 basada en JSPs, Servlets y JAVA sin ningún error o impacto al performance del sitio, esto permitió implementar una solución para el equipo de marketing y negocio sin mermar la experiencia de usuario. Es por ello que, también se desarrolló su ajuste para la versión 9 próxima a salir en los próximos meses cuando otros procesos se encuentren listos, en dicha versión se trabajó sobre nuevas tecnologías destacando REACT como frontend y nuevas versiones JAVA en el backend.

Así mismo se proporcionó una implementación la cual al recabar información de forma periódica y constante en el Ecommerce permitió visualizar gráficos y métricas para el equipo de marketing los cuales tuvieron más visibilidad acerca del número de usuarios en ciertas ciudades, los productos más vistos, los productos más vendidos e incluso un promedio de gasto en cada compra en un intervalo de tiempo, con esa información se crearon campañas de marketing por diversos medios digitales y mejorar una estructura en el sitio.



[Figura 19. Ejemplo de resultados finales Conversions API - Facebook]

Este fenómeno se vio reflejado en temporadas altas como el Buen Fin donde se promocionaron artículos destacados y promociones para aquellos que no habían tenido tanta exposición incluso formando diferentes combinaciones de combos. Finalmente algunas campañas tuvieron una estructura similar a la que se muestra a continuación:



[Figura 20. Ejemplo de estrategia marketing]

10. Conclusiones

Considero que mi experiencia durante este proyecto fue muy nutritiva respecto a conocimiento adquirido, además que me dio una amplio panorama de cómo funcionan los sistemas en el mundo empresarial desde ambientes de desarrollo hasta ambientes productivos, considero que realmente me sacó de mi zona de confort varias veces dado que debía ser una implementación muy completa la cual implicó manejo de diferentes conocimientos, desde el uso de servidores, aplicar conceptos de programación, estimación de tiempo y esfuerzo. además demandó bastante razonamiento y análisis ya que durante el proceso de implementación surgieron varias complicaciones o errores que estaban fuera de lo común, por lo que la habilidad de investigar correctamente tomó un papel determinante para poder entregar a tiempo la implementación o bien, recurrir a la negociación de la fecha de entrega basándose en los desafíos que se iban presentando.

Considero que la formación adquirida durante la carrera me permitió tener ese enfoque analítico necesario para darle la vuelta a los problemas buscando enfoques creativos o soluciones basadas en conocimientos previos. Por otro lado, este proyecto en verdad explotó habilidades blandas para entablar comunicación con otras áreas y/o equipos lo cual me permitió comprender cómo funcionan diversos procesos y de ese modo tomar en cuenta varios procesos internos y así evitar un impacto en la operación, o simplemente pedir apoyo en caso de ser necesario.

Me atrevo a decir que el salto a un proyecto profesional es verdaderamente interesante ya que ayuda a conocer áreas de oportunidad que previamente podríamos desconocer, ayuda a comprender la importancia de la tecnología y el impacto de la misma sobre procesos que no son tan sencillos de identificar en el mundo académico, sin embargo, al empaparse del modelo de negocio, de las necesidades y visiones de la organización, poco a poco se empieza formular un camino en el que los conocimientos adquiridos durante el proceso académico cobran sentido y adquieren relación para poder aplicar la ingeniería para el beneficio de la sociedad con información ya existente o nueva y esto facilite actividades, ideas, procesos o negocios sacando el mayor provecho posible a la tecnología existente.

11. Referencias:

- *Project Management Institute, Inc. (2021). A Guide to the project Management Body of Knowledge PMBOK Guide Seventh edition (Seventh edition). pp. 178, https://ibimone.com/PMBOK%207th%20Edition%20(iBIMOne.com).pdf. Consultado el 21 de mayo del 2023.*
- *Gamma, E., & Helm, R. (2009). Design Patterns Elements of Reusable Object-Oriented Software. https://www.javier8a.com/itc/bd1/articulo.pdf. Consultado el 25 de mayo del 2023.*
- *Gabbrielli, M., & Martini, S. (2010). Programming Languages: Principles and Paradigms. En Undergraduate topics in computer science. http://160592857366.free.fr/joe/ebooks/ShareData/Programming%20Languages%20-%20Principles%20and%20Paradigms.pdf. Consultado el 11 de mayo del 2023.*
- *Apigee, & Nijim, S. (2014). APIs for Dummies. https://cloud.google.com/files/apigee/apigee-apis-for-dummies-ebook.pdf Consultado el 19 de marzo del 2024.*
- *Lane, K. (2022). The API-First transformation. https://voyager.postman.com/book/api-first-transformation-book-postman-9798986951812.pdf Consultado el 18 de octubre del 2023.*
- *¿Qué es el sector retail? Descubre cómo iniciarte en él con tu ecommerce*. (2022, 7 febrero). Shopify. https://www.shopify.com/mx/blog/que-es-retail. *Consultado el 22 de mayo del 2023.*
- EALDE [EALDE Business School]. (2020, 5 mayo). *4 métodos de estimación de duración de las actividades de un Proyecto*. EALDE Business School.

https://www.ealde.es/duracion-actividades-proyectos/ .*Consultado el 28 de mayo del 2023.*

- *Arquitectura basada en eventos*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/event-driven-architecture/. *Consultado el 28 de mayo del 2023.*
- *Facebook - Meld je aan of registreer je*. (s. f.). Facebook. https://www.facebook.com/unsupportedbrowser *Consultado el 2 de junio del 2023.*
- *metodologías para obtener un estimado de tiempo en un proyecto - Google Zoeken*. (s. f.). https://www.google.com/search?q=metodologias+para+obtener+un+estimado+de+tiempo+en+un+proyecto .*Consultado el 2 de junio del 2023..*
- *¿Qué es la arquitectura basada en eventos?* (s. f.). https://www.redhat.com/es/topics/integration/what-is-event-driven-architecture *Consultado el 4 de junio del 2023.*
- Busio, O. J. G., & Busio, O. J. G. (2018, 26 enero). *Estimación por tres valores*. TodoPMP. https://todopmp.com/herramientas/estimacion-tres-valores/ .*Consultado el 10 de junio del 2023.*
- Ingenieria, F. de. (s. f.). *Facultad de Ingeniería / Historia*. https://www.ingenieria.unam.mx/nuestra_facultad/historia.php *Consultado el 10 de junio del 2023.*
- *Programación orientada a objetos*. (s. f.). https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming .*Consultado el 15 de junio del 2023.*
- Alarcon, J. (2021, 26 julio). *Los conceptos fundamentales sobre Programación Orientada Objetos explicados de manera simple*. campusMVP.es.

https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx .*Consultado el 16 de junio del 2023.*

- Oracle. (s. f.). *¿Qué es una base de datos?* OCI. https://www.oracle.com/mx/database/what-is-database/ .*Consultado el 16 de junio del 2023.*
- *Arquitectura Cliente-Servidor*. (s. f.). https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor .*Consultado el 15 de junio del 2023.*
- *Acerca de los repositorios*. (s. f.). GitHub Docs. https://docs.github.com/es/repositories/creating-and-managing-repositories/about-repositories. *Consultado el 17 de junio del 2023.*
- *Acerca de los repositorios*. (s. f.). GitHub Docs. https://docs.github.com/es/repositories/creating-and-managing-repositories/about-repositories .*Consultado el 18 de junio del 2023.*
- Soto, N. (2022, 17 abril). *¿Qué son los patrones de diseño en programación? 2023*. Craft - Code | La Academia de las Buenas Prácticas. https://craft-code.com/que-son-los-patrones-de-diseno/ .*Consultado el 20 de septiembre del 2023.*
- Kaspersky. (2022, 11 febrero). *¿Qué es el cifrado de datos? Definición y explicación*. latam.kaspersky.com. https://latam.kaspersky.com/resource-center/definitions/encryption .*Consultado el 5 de noviembre del 2023.*
- *¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/. *Consultado el 12 de enero del 2023.*

Recopilación de iconos para realizar diagramas:

- Flaticon.com
- draw.io