INSTITUTO UNIVERSITARIO AERONÁUTICO

FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN

TRABAJO FINAL DE GRADO

"REINGENIERÍA DE PROCESO DE LOGÍSTICA E-COMMERCE"

AUTORES: ***MAGE, MAURICIO JESÚS***

ROMANO, AGUSTÍN

TUTOR: ***ING. CARLOS ALEJANDRO SIMES***

SOLICITUD DE EVALUACIÓN DEL TRABAJO FINAL DE GRADO

INFORME DE EVALUACIÓN DEL TRABAJO FINAL DE GRADO (Apéndice VI)

Agradecimientos

Dedicatoria:

Abstract

ÍNDICE

1. INTRODUCCIÓN

1.1. Antecedentes

Las empresas de logística convencionales se encuentran operando por encima de su capacidad operativa, lo que deviene en deficiencia en la calidad del servicio brindado a todas las partes del proceso, lo cual produce:

- Demoras.
- Disconformidad de los usuarios hacia la empresa.
- En algunos casos daños o pérdidas de los bienes o productos en cuestión.

Se observa deficiencia desde que el producto abandona el depósito de la empresa vendedora hasta que llega al centro de logística del correo. En esta parte del proceso, se debe convenir si el correo debe buscar el producto en el origen, o si la empresa vendedora es quien debe llevarlo hacia el correo (ya sea por sus propios medios o inclusive contratando a un tercero), en cuyo caso deberá ser atendido en el horario establecido y según disponibilidad de personal para su recepción.

Teniendo en cuenta la creciente demanda de compras por internet, hacen que este escenario sea cada vez peor.

Comprar por internet debiera ser el modo más fácil de hacerlo: no se gasta tiempo en ir al lugar, siempre hay disponibilidad de lo que uno busca y es posible comprar las 24 hs. del día. La modalidad de compra online está creciendo en el país y en el resto del mundo, impulsado fundamentalmente por las ventas en cuotas y la financiación con tarjetas de crédito. No obstante, algunas cuestiones logísticas entorpecen dicho crecimiento.

La logística en el e-commerce implica el envío a domicilio de los productos o servicios adquiridos a través de internet por parte de los clientes. Es reconocido por los especialistas como el Talón de Aquiles del e-commerce y se trata de uno de los principales obstáculos para este tipo de plataformas. Aspectos como definir el medio de envío, elegir el tipo de gestión de stock, seleccionar el packaging adecuado y

definir nuevas políticas de cambios y devoluciones pueden hacer que el negocio prospere o no.

En cuestión de números, de acuerdo a la Cámara de Comercio Electrónico (CACE), la mayor barrera para el desarrollo del e-commerce en Argentina y el resto de Latinoamérica es la logística: el 58% de los envíos llegan luego de una semana y el 34% después de 15 días. Seguida de esta barrera está la inseguridad de brindar los datos de la tarjeta de crédito y los altos costos de los envíos. Asimismo, el experto en comercio electrónico y fundador de ShipNow, Tomás Allende, remarca: "es un proceso que el vendedor le echa la culpa a los correos, los correos dicen que el vendedor no lo entregó a tiempo y a ojos del comprador todos se lavan las manos". Consecuentemente, cuando estas complicaciones aparecen en la compra/venta online y los productos tardan en llegar, no se sabe qué pasó y no se puede hablar con el supuesto encargado, la facilidad que provee el e-commerce se transforma en una complicación absoluta.

Paralelo a todo lo recién mencionado, actualmente está tomado notoriedad en todo el mundo una interesante aplicación mobile, que implica una solución por y para la sociedad, teniendo en cuenta que su modelo de negocio está basado en que la comunidad crezca exponencialmente dando soluciones a los usuarios que quieren trasladarse en una ciudad. Quitando problemas de tráfico, sociales y hasta ambientales. Su nombre es Uber, y está cambiando de forma radical la manera en que personas se trasladan en las grandes ciudades.
Esta popular aplicación permite conectar a pasajeros con conductores de vehículos registrados en su servicio. La solución utiliza un software especial para enviar al chofer (registrado y confiable) más cercano a la ubicación del cliente. Cabe aclarar que este no es un servicio de taxi compartido o algo por el estilo. La visión de la empresa para el futuro es contar con una transportación más inteligente, con menos autos, mayor acceso, seguro, económico y confiable

En base al producto a desarrollar, uno de los rasgos sobresalientes del mismo, es que reestructura el tradicional sistema de logística de distribución a través de una completamente renovada y simple, adaptada a los tiempos de hoy. Distinta a las

empresa de logísticas de grandes estructuras y medios de transporte caros, se propone una solución heterogénea y sustentable, mediante una inmensa red de mensajeros que puedan ocuparse tanto de paquetes pesados y livianos como distancias de todo tipo a un bajísimo costo para el cliente y a una mayor eficiencia para la plataforma de e-commerce.

Otro rasgo importante a destacar es que el envío lo puede realizar cualquier individuo que disponga de tiempo libre, posea un smartphone y tengan acceso a cualquier medio de transporte (moto, auto, servicio público, bicicleta, etc.). Cualquiera puede transformarse en socios de la plataforma y poder trabajar. El mensajero es quién va a decidir cuánto trabajar, no depende de un jefe directo para efectuar el trabajo y el cobro por su trabajo es rápido y seguro. De todos modos, para respetar calidad en el envío de las compras y cumpla ciertas exigencias, es imprescindible que cada persona que quiera ser mensajero pase por un estudio previo.

Finalmente, la plataforma va a lograr reducir el tráfico de vehículos motorizados y con esto las emisiones de CO2, como así también generar empleos y mejorar la logística en los envíos del día a día de todas las ciudades que se sumen.

A modo de conclusión, luego de mencionar los problemas del e-commerce sumado a la exitosa experiencia de Uber, más los rasgos importantes y porque del producto a desarrollar, es posible crear una innovadora solución para el mercado. Una solución fácil y segura, que resuelva los inconvenientes en la logística de los e-commerce y otorgue un mejor y mayor trabajo a los mensajeros y a todo aquel que desee trabajar.

1.2. Situación problemática

Como ya se evidencia, el nicho de mercado al cual se apunta con esta solución es a toda empresa de e-commerce o una que disponga de e-commerce como canal de comercialización. Se elige este nicho debido a las enormes deficiencias que presenta la logística de distribución actual, siendo la gestión de los envíos la principal problemática de las tiendas de e-commerce. De todos modos en el mediano plazo es posible ampliar e integrar nuevos mercados.

Los números del e-commerce crecen año tras año, tomando cada vez más espacio en las arenas de los negocios tradicionales y logrando que todo plan comercial o de negocios deba contemplar dentro de sus estrategias el canal online. En el plano internacional para finales de 2015 más del 44% de la población mundial compró online en algún momento en el año. En el 2018 este número saltará a casi el 50%.

A nivel local, el e-commerce en Argentina creció durante el año 2015 un 70.8% respecto al año anterior. Dicha cifra surge del Estudio Anual de Comercio Electrónico en Argentina que realiza TNS para la Cámara Argentina de Comercio Electrónico (CACE). Este fue un crecimiento récord, muy por encima de lo previsto para 2015. El crecimiento interanual fue de 70.8% y la facturación por ventas superaron los $68.486 millones de pesos. Adicionalmente, nuestras proyecciones indican que en 2016 el commerce continuará creciendo en un 64%, por lo que las perspectivas para el sector son más que promisorias.

Las principales características del mercado argentino son:

- La facturación total en 2015 fue de $ 68.486 millones de pesos. Esto implica un crecimiento del 70,8% respecto de 2014.
- El crecimiento sostenido de la proporción de usuarios de internet que realizaron compras en línea: de un 10% aproximado en 2001 al 77% en 2015, año en que los compradores en línea superan los 17 millones de personas.
- El 36% del tráfico total en comercio electrónico en Argentina, provino de dispositivos móviles.
- El 97% de lo compradores manifestaron satisfacción con las compras realizadas.
- El 89% aseguraron que eligen comprar en internet por comodidad, y un 82% por precio.
- En 2015 el 89% de los usuarios de ecommerce utilizaron plataformas para comprar y/o vender. De ellos, un el 64% vendió y el 60% compró al menos un producto en el último año.

Asimismo también se debe hacer mención del mercado de logística de distribución de e-commerce, la e-logística, punto crítico para estas plataformas web. Aspectos como definir el medio de envío, elegir el tipo de gestión de stock, seleccionar el packaging adecuado, y definir buenas políticas de cambios y devoluciones pueden hacer que el negocio prospere. Según el experto en e-commerce y fundador de ShipNow, Tomás

Allende, "las mejoras en logística son el principal factor que puede actuar como acelerador del e-commerce en la Argentina, a la par con las nuevas tecnologías de automatización e integración de procesos". Justamente es lo que se pretende con este proyecto.

1.3. Problema

Imposibilidad o complejidad al solicitar un envío hacia algún punto en particular, sin tener que pretenderlo. Es decir, pocas alternativas de que alguien realice un pedido en tiempo y forma. Como así también no saber quien realiza un pedido y cuando con exactitud cuando el mismo es comprado bajo la normativa de una plataforma e-commerce. Logrando que el Usuario quede pendiente de quien puede retirar su pedido, y además abonar aparte del costo del producto por el mismo.

Poco seguimiento y tracking de la orden, requiriendo el uso de la opción telefónica la cual no siempre es verídica.

1.4. Objeto de estudio

El presente trabajo tiene por objetivo conocer aspectos tecnológicos diversos que puedan integrarse con el fin de desarrollar una solución factible, de fácil implementación y
accesible económicamente para el problema planteado.

Para la consecución de estos objetivos será preciso abordar el estudio de:

- Métodos, tecnologías y herramientas informáticas aplicables, implicadas en la investigación y estudio de los procesos logísticos, seguridad del pedido, condiciones de transporte del delivery y confianza entre usuarios, para lograr el uso correcto del “manual” del proceso logístico e-commerce desde una nueva alternativa y solución.

- Nuevas tendencias en diseño web y en aplicaciones móviles para su interpretación de usuarios al usarla y ser conscientes de las acciones que realicen o soliciten.

- Marco normativo y factibilidad de aplicación de los anteriormente mencionados para efectivización de una solución tecnológica que contemple todos los aspectos involucrados en la problemática de logística, transporte, tránsito y aspectos socio-ambientales.

1.5. Campo de acción

Se pretende enfocar el camino correcto de el completo proceso desde la compra de un producto en una tienda e-commerce o la carga de un pedido por un usuario en la plataforma, hasta la conformidad entre las partes para lograr realizar el delivery.

Actuando de manera didáctica en la postura del cambio de proceso logístico e-commerce. Brindando valores para el fin de alcanzar la solución planteada.

1.6. Objetivos

Los objetivos del proyecto están apuntados a evitar los fallos en la entrega del producto de las plataformas de e-commerce para no se generen clientes insatisfechos, mala imagen de marca o compañía e incluso pérdida del cliente. Además de generar fuente de trabajo para personas que lo dispongan en cuanto la realización del Delivery.

1.6.1. Objetivo general

Como objetivo general se tiene por intención modificar radicalmente el paradigma que engloba el conjunto de medios y métodos necesarios para llevar a cabo la logística de envíos a domicilio de las plataformas e-commerce y desde pedidos que se generen a través de la gente que tiene la necesidad de enviar algo a algún punto. Logrando que Deliveries puedan ver la oportunidad de enviar una oferta al Usuario para llevar ese pedido.

1.6.2. Objetivos específicos

- Incorporar tecnología de punta al servicio de mensajería para resolver el problema de logística de distribución y envío de los pedidos de las plataformas de e-commerce (comercio electrónico).

- Proponer una innovadora forma de logística en el e-commerce en la Argentina y el resto del mundo, buscando una solución fácil y transparente tanto para el vendedor como para el comprador.

- Acelerar el crecimiento del e-commerce fundamentalmente en los países de Latinoamérica, a la par con las nuevas tecnologías de automatización e integración de procesos.

- Disminuir el costo de envío y el tiempo de espera desde el pago del producto hasta la entrega del mismo al domicilio del cliente.

- Potenciar a los servicios de mensajería mediante el desarrollo de una herramienta que optimice el trabajo y amplíe la cartera de clientes de sus trabajadores.

1.7. Idea a defender

Nos basamos en lograr cubrir una necesidad presente la cual pretende cambiar la modalidad de logística e-commerce , teniendo en cuenta que como ya se ha comentado, en ocasiones es un problema contratar a empresas, y a su vez realizar un seguimiento verídico. Por lo tanto, buscamos abrir además una necesidad en la gente de querer realizar estos pedidos (el Delivery) o compras de e-commerce para obtener beneficios y lograr una entrega en la cual cubre la necesidad del Usuario.
Por otro lado, también defendemos la idea de comprobar una mejora en el transporte y entorno social, con los procesos que se darán cuando se transcurre una EnvíoEntrega, permitiendo a usuarios usar sus propios medios de transporte, sin

necesidad de sumar más vehículos empresariales que colapsan el tráfico. De esa forma nosotros ratificamos que es una mejora a nivel social.
En resumen, las principales características de la solución son:

- Envíos más eficientes, rápidos y sustentables.
- Creación de empleo flexible.
- Seguimiento del estado del pedido en tiempo real.
- Amplios horarios de servicio.
- Flexibilidad y personalización del servicio: horarios, niveles de urgencia, costes.
- Menor impacto ambiental.

1.8. Delimitación del proyecto

Se realizará el abordaje de las diferentes tecnologías existentes para el desarrollo e implementación del proyecto .Además lograr y llegar a la solución planteada e identificada.
El proyecto se piensa analíticamente de manera que se presente estable para estimar el desarrollo del mismo y a su vez como luego éste va a impactar en la sociedad y en el uso del mismo.
A su vez, se analiza y planifica los resultados para posibles cambios y/o actualizaciones que estabilizaran y guiarán al prototipo/sistema

1.9. Aporte práctico

El proyecto involucra dos plataformas, una web, que permite que ambos usuarios interactúen cargando envíos y solicitando deliveries, en la cual cada uno de ellos se identifica ante una necesidad de realizar un pedido mediante el envío de una oferta a

un pedido ya cargado. Y también que el usuario que desee enviar algo a algún punto, lo podrá hacer cargando un pedido nuevo.

La otra plataforma es móvil (aplicación) para sistemas operativos tanto Android como iOS. También la aplicación brinda la opción de buscar pedidos para realizar, y cargar nuevos. A su vez, posee la mayor parte del proceso de una orden, ya que al confirmarse una oferta de delivery, ya sea a través de un sitio ecommerce o por un pedido mismo que el Usuario haya solicitado en el sitio o app de EnvioEntregas, actúa la parte en que ambos usuarios (solicitante y delivery) a comunicarse entre ellos cuando el pedido está en camino, ingresando códigos al Inicio del pedido y al finalizar el mismo, para verificar que el pedido ha llegado en condiciones.

Se aclara que al momento de aceptar el pedido, el Usuario que lo solicita, debe abonar el pago antes de aceptar el mismo. Esto se hace a través de un link que dirige a la API de MercadoPago la cual está encargada de operar la gestión y validar la tarjeta de Crédito o Débito, una vez pasada esa condición y haber concluido el Pago, se acepta la orden notificando al Delivery para que éste comience el proceso.

Luego al haberse concluido la orden, ambos Usuarios (Solicitante y Delivery) podrán calificar y ser calificados. Ésto se tiene en cuenta a la hora de seleccionar un delivery de parte del solicitante, con respecto a su reputación y así también del Delivery al elegir qué pedido hacer, con respecto al Solicitante que lo postule.

1.10. Aporte teórico

Partiendo de la idea opción de comprar por internet, sin gastar tiempo y lugar para ir, y que siempre hay disponibilidad de lo que uno busca y es posible comprar las 24 hs. del día; la real problemática de ese proceso es la logística. La modalidad de compra online está creciendo en el país y en el resto del mundo, impulsado fundamentalmente por las ventas en cuotas y la financiación con tarjetas de crédito. No obstante, algunas cuestiones logísticas entorpecen dicho crecimiento.

El envío a domicilio de los productos o servicios adquiridos a través de internet por parte de los clientes, proceso reconocido como Logística e-commerce, el mismo es reconocido por los especialistas como el Talón de Aquiles del e-commerce y se trata de uno de los principales obstáculos para este tipo de plataformas. Aspectos como definir el medio de envío, elegir el tipo de gestión de stock, seleccionar el packaging adecuado y definir nuevas políticas de cambios y devoluciones pueden hacer que el negocio prospere o no.

Por lo tanto, se proponen nuevos modelos y aspectos teóricos de negocio, paralelo a todo lo recién mencionado, actualmente está tomado notoriedad en todo el mundo una interesante aplicación mobile. Su nombre es Uber, y está cambiando de forma radical la manera en que personas se trasladan en las grandes ciudades.
Esta popular aplicación permite conectar a pasajeros con conductores de vehículos registrados en su servicio. La solución utiliza un software especial para enviar al chofer (registrado y confiable) más cercano a la ubicación del cliente. Cabe aclarar que este no es un servicio de taxi compartido o algo por el estilo. La visión de la empresa para el futuro es contar con una transportación más inteligente, con menos autos, mayor acceso, seguro, económico y confiable.

A modo de conclusión, luego de mencionar los problemas del e-commerce sumado a la exitosa experiencia de Uber, es posible crear una innovadora solución para el mercado. Una solución fácil y segura, que resuelva los inconvenientes en la logística de los e-commerce y otorgue un mejor y mayor trabajo a los mensajeros y a todo aquel que desee trabajar.

A su vez, existe otro mercado que es fundamental para cualquier empresa productora de bienes y servicios, y es el de logística de distribución. Su actividad permite el traslado de los productos finales (ya sean bienes o servicios) y los pone a disposición del cliente. El canal de distribución es el que posibilita que el usuario obtenga el producto en el lugar, tiempo y cantidades adecuadas. La calidad de este servicio es fundamental para el éxito de cualquier empresa, y mayor aún si son plataformas de e-commerce.

En el mercado actual donde el cliente es el centro, se debe valorar mucho la unión de culturas que existe entre la logística y el e-commerce, un mercado en crecimiento que requiere un proceso de adaptación. Este crecimiento facilita la elección del consumidor, pero también desafía a las compañías a mantener un buen ritmo de satisfacción de las expectativas de los consumidores en entregas más rápidas y sin errores.

1.11. Factibilidad

1.11.1. Factibilidad técnica

Dada la experiencia de cada uno de los integrantes, sumado a la capacidad de conocimientos técnicos de ambos para el desarrollo e implementación del proyecto, más el proceso logrado en la carrera en la cual estamos, todas las cuestiones son adecuadas para la concreción del mismo.

1.11.2. Factibilidad operativa

Considerando el alcance de la idea y la capacidad para operar y llegar a la solución enfocándose siempre en la mejora inicial del problema, hace que se obtenga una total factibilidad operativa.

1.11.3. Factibilidad económica

La factibilidad económica está impuesta bajo la inversión que ha realizado Fonsoft para el desarrollo y puesta en marcha del proyecto, por lo cual es totalmente factible.

2. PRIMERA PARTE: MARCO CONTEXTUAL

2.1. Entorno del objeto de estudio

Crear una solución tecnológica que permita que cualquier ciudadano pueda, en su recorrido habitual, transformarse en un potencial medio de transportar bienes o productos. De esta manera, se incrementan los agentes de envíos sin sobrecargar los canales de movilidad.

El solicitante tendrá la tranquilidad de saber quién está manipulando su pedido, y además tendrá la posibilidad de comunicarse con dicha persona. Ambas partes se

pondrán de acuerdo en el tiempo y forma de retiro y entrega, lo cual lo hace más flexible y sencillo.

2.2. Relación tesistas y objeto de estudio

El presente trabajo tiene como objetivo conocer alternativas apoyadas por el uso de la tecnología para acompañar al crecimiento de las ventas por internet sin aumentar considerablemente el tránsito.

Algunos aspectos a tener en cuenta son:

- Solución tecnológica, fácil de usar y amigable para el usuario, que bien sea posible acceder desde un ordenador o desde dispositivos móviles.
- Facilitar el proceso entre los agentes involucrados: cliente que desea enviar, y ente que hará el envío.
- Proporcionar seguridad y seguimiento del estado del pedido.
- Posibilidad de reutilizar recorridos habituales de las personas.
- Disminuir la cantidad de intermediarios que participan en el proceso.

2.3. Análisis de los problemas observados

Como se identificó, el e-commerce ha experimentado una notable evolución, pasando de ser un simple catálogo de productos o servicios, construido a partir de una página estática, a convertirse en un medio eficaz para realizar negocios. Es por eso que los números de este mercado crecen año tras año, tomando cada vez más espacio en las arenas de los negocios tradicionales y obligando a que todo el plan comercial o de negocios deba contemplar dentro de sus estrategias el canal online. Para ejemplificar en el 2016, como en años anteriores, el crecimiento del e-commerce en Latinoamérica, según eMarketer, seguirá alcanzando cifras de dos dígitos hasta el 2019. No solo mayores ventas, sino también crecimiento en la comunidad de usuarios y también evolución en la forma en la que usamos la tecnología.

A nivel local, de acuerdo a la Cámara de Comercio Electrónico (CACE), en Argentina de las personas con acceso a internet 8 de cada 10 alguna vez realizaron una compra

online. A su vez el crecimiento que tuvo el e-commerce en el país fue del 61,7% en 2014 y 58% en 2015. La tendencia en preferencias en la región se ha inclinado hacia los envíos a domicilio y el mercado argentino no es la excepción: el 73% de los compradores consultados eligieron esa modalidad, un 59% retiró el producto en el punto de venta, y el 30% lo buscó en una sucursal del correo. Por lo tanto ahi tenemos la noción lo cual nos lleva a la principal problemática.

Al tener un publico y numero de personas importantes (cantidad) que soliciten productos por internet, o requieran de que alguien les lleve un pedido que ellos quieran enviar desde su casa, ya hace a que empresas con las que ya existen de correo, no den a basto, o requieran de mas transporte del habitual para lograr hacerlos, lo que conlleva a problemas socio-ambientales y imposibilidades de tener muchas mas alternativas del lado del usuario.
Es por eso que la idea es solucionar lo que se ha nombrado anteriormente, y además lograr una gran oportunidad a la persona común que en sus caminos y rutas habituales hacia el trabajo, o donde quiera que se dirija, pueda hacerse con un beneficio económico, y además brindando la garantía de que va a poder ser observado y consultado en todo su proceso, logrando la confianza y agilidad de la orden.

Otro dato que aporta a la fuerte abundancia de solicitudes en internet, es que la facturación fue de $40.100 millones en el 2014, a diferencia del 2015, que trepó a los $68.486 millones. Por último, y siguiendo al CACE, el crecimiento del 70,8% anual, permite estimar que en 2016 crecerá un 64% (el estimado para 2015 era de un 58%).

Por otro lado, existe otro mercado que es fundamental para cualquier empresa productora de bienes y servicios, y es el de logística de distribución. Su actividad permite el traslado de los productos finales (ya sean bienes o servicios) y los pone a disposición del cliente. El canal de distribución es el que posibilita que el usuario obtenga el producto en el lugar, tiempo y cantidades adecuadas. La calidad de este servicio es fundamental para el éxito de cualquier empresa, y mayor aún si son plataformas de e-commerce, de ésta manera se identifica otro de los problemas principales que corrompen en la mala planificación o escasez de tiempo para el envío

y/o retiro de un bien para llevarlo a otro punto. Lo cual hace que se desprenda la confianza en el que solicita con el Delivery, obteniendo malos resultados y problemáticas en la empresa.

En el mercado actual donde el cliente es el centro, se debe valorar mucho la unión de culturas que existe entre la logística y el e-commerce, un mercado en crecimiento que requiere un proceso de adaptación. Este crecimiento facilita la elección del consumidor, pero también desafía a las compañías a mantener un buen ritmo de satisfacción de las expectativas de los consumidores en entregas más rápidas y sin errores. Problema por el cual hoy en día la gente desiste o ya está entregada a lo que puede suceder.

2.4. Antecedentes de proyectos similares

A la vista está el crecimiento y desarrollo de Uber como se ha nombrado anteriormente, plataforma tecnológica que pone en contacto a socios conductores con pasajeros gracias a una aplicación móvil.
Funciona de la siguiente manera. Primero, se solicita un viaje desde la aplicación en cualquier ciudad donde Uber esté activo. La solicitud se enviará a los socios conductores más cercanos. Cuando un conductor acepte la solicitud de viaje, el usuario podrá consultar en la aplicación el tiempo estimado de llegada del conductor a la ubicación de recogida. También le llegará una notificación al usuario cuando el conductor esté a punto de llegar.

Al solicitar un viaje, la aplicación te proporciona el nombre del conductor, el tipo de vehículo y su número de matrícula para que puedas reconocerlo sin problemas. Cuando el usuario está dentro del vehículo, comprueba que el destino que has introducido en la aplicación sea el correcto. Los conductores utilizan su aplicación Uber para confirmar los detalles del viaje. Si el usuario prefiere alguna ruta en concreto, se lo informa al conductor.

Cuando se llega a destino, el conductor finalizará el viaje. El precio del viaje se calcula automáticamente y se cobra a través del método de pago vinculado a la cuenta de Uber. En determinadas ciudades Uber permitirá pagar con dinero en efectivo.

Inmediatamente después, la aplicación pedirá que valores la experiencia del viaje. A los conductores también se les pide que valoren el viaje. Este sistema de comentarios bidireccional de Uber promueve una comunidad de respeto y confianza en la plataforma, y garantiza un servicio de alta calidad para todos los usuarios.

Por lo tanto el modelo de negocios que posee Uber, es muy similar al del proyecto actual, ya que el Usuario y el Conductor, serìan el Usuario que compra un bien en una tienda web, o que carga un nuevo pedido que quiere que le retiren y lleven a un punto con el Delivery que es quien solicita al Usuario para poder enviar su pedido.
De esa forma, tanto el Conductor (Uber) como el Delivery (EnvioEntregas) obtienen beneficio económico, brindándoles una solución al Usuario que quiere llevar un pedido a algún destino, o que espera que le lleven el producto que compro en la tienda web a su casa. Al igual que el tipo de beneficio/servicio que es concedido y obtenido por el Usuario de Uber.

Al momento de la finalizaciòn del proceso, en ambos modelos de negocio tienen en cuenta la valoración entre ambos actores, es decir, calificacion de manera bidireccional, ya que permite que la idea sea conclusa y de confianza, para que el Delivery confíe en los usuarios como los choferes confían en sus tripulantes, y que los Usuarios confíen en el Delivery y/o chofer, guiándose ambos, a travès de su reputación propiamente dicha y generada por la comunidad.

3. SEGUNDA PARTE: MARCO TEÓRICO

3.1. Introducción

Al hablar de los marcos conceptuales del problema logístico y comunicacional que existe hoy en día en el e-commerce al a hora de enviar/entregar un bien o producto, obliga a que se analice cada punto con el sentido de encontrar la solución visible al determinado asunto.

La idea radica en centrar los principales nexos que unen al problema y hacer de éste aplicable al desenlace.

Se toma como primer instancia abordar los temas conceptuales en cuanto al significado de logística e e-commerce, luego siguiendo con los actores quienes interactúan en el proceso, identificar las acciones que estos puedan realizar y participar como así también la relación entre éstos en relación bidireccional.

De ésta forma, a medida que se van desglosando estos ítems, se comienza a planificar y diagramar las entes y estrategias del desarrollo de la solución. Para luego, después de un proceso llevado a cabo con metodologías ágiles, con lo que eso involucra, más una conclusión en base a lo planeado, llegar a la solución conceptual, para dar inicio al trabajo.

3.2. Marco teórico del objeto de estudio

3.2.1. E-commerce Origen y Evolución Histórica

En los últimos decenios del siglo XIX empresas comerciales como Montgomery Ward y luego Sears iniciaron la venta por catálogo en los Estados Unidos. Este sistema de venta, revolucionario para la época, consiste en un catálogo con fotos ilustrativas de los productos a vender. Este permitió a las empresas captar nuevos segmentos de mercado que no estaban siendo atendidos. Además, otro punto importante a tener en cuenta es que los potenciales compradores pueden escoger los productos en la tranquilidad de sus hogares, sin la asistencia o presión, según sea el caso, de un vendedor. La venta por catálogo tomó mayor impulso con la aparición de las tarjetas de crédito; además de determinar un tipo de relación de mayor anonimato entre el cliente y el vendedor.

La práctica del comercio electrónico comenzó a principios de 1970, con novedosas aplicaciones como la transferencia de fondos monetarios. Después apareció el intercambio de datos vía electrónica, que produjo una expiación en el comercio electrónico, dando lugar a otros tipos de procesos comerciales. Todos estos procesos permitieron que pequeñas empresas pudieran aumentar su nivel de competitividad

implementando el comercio electrónico en sus actividades diarias. Debido a esto el comercio en línea se ha expandido muy rápidamente gracias a los millones de consumidores potenciales a los que se puede llegar a través de este medio.

A principio de los años 1970, aparecieron las primeras relaciones comerciales que utilizaban una computadora para transmitir datos, tales como órdenes de compra y facturas. Este tipo de intercambio de información, si bien no estandarizado, trajo aparejadas mejoras de los procesos de fabricación en el ámbito privado, entre empresas de un mismo sector.

A mediados de 1980, con la ayuda de la televisión, surgió una nueva forma de venta por catálogo, también llamada venta directa. De esta manera, los productos son mostrados con mayor realismo, y con la dinámica de que pueden ser exhibidos resaltando sus características. La venta directa se concreta mediante un teléfono y usualmente con pagos de tarjetas de crédito.

En 1995 los países integrantes del G7/G8 crearon la iniciativa Un Mercado Global para PYMES,1 con el propósito de acelerar el uso del comercio electrónico entre las empresas de todo el mundo. Y fué así como dio inicio al mundo insuperable del e-commerce

3.2.2. Cambiando las formas tradicionales de hacer negocios

Las redes sociales se han convertido en una fuente informativa y de negocios de alto impacto, por ejemplo Facebook, en donde las personas pueden conectarse directamente con otros miembros de la red social para establecer las actividades comerciales de cliente a cliente (C2C), ofreciendo muchas funcionalidades para el contenido generado por el usuario. Los usuarios (en este caso vendedores) pueden publicar anuncios ilimitados de sus productos en diferentes secciones de Facebook como muros, páginas y grupos, donde pueden ver miles de personas de las cuales resulta un número de usuarios interesados (en este caso compradores), quienes pueden ponerse en contacto con los vendedores directamente para iniciar el proceso de compra, estableciendo un método de pago y entrega.

Ya que Facebook es una red social gratuita y de fácil acceso, se ha convertido en un fundador de realizaciones comerciales popular para las personas que desean intercambiar bienes o servicios, es decir, es un tipo especial de comercio electrónico C2C, sin embargo, es de alto riesgo ya que nadie le garantiza (por el hecho de ser usuarios desconocidos) a un comprador inconforme que le devuelvan su dinero, en caso por ejemplo, de que una transacción salga mal, porque los compradores no pueden evaluar con precisión la credibilidad y reputación de un vendedor como lo hacen con el comercio electrónico B2C, por lo que sus procesos de compra son inciertos.

Prueba irrefutable de que la información manejada en este tipo de comercio electrónico es peligrosa, resulta el 18 de Marzo del 2018, con el escándalo de Cambridge Analytica, compañía que obtuvo la información de usuarios de la red social y luego construyó un programa para manipular la votación de millones de usuarios en las elecciones presidenciales de los Estados Unidos en 2016 dando como resultado de la investigación, una estruendosa caída de las acciones de Facebook en la bolsa de valores.

Es por esto que la información personal de las personas debe manejarse con mucho cuidado, brindando la mayor seguridad posible para que no corran riesgos ni se sientan inseguros en cuanto a su información que en este caso, va más allá de unos cuantos likes, como lo son las cuentas bancarias, direcciones residenciales, etc

3.2.3. La logística presente en Restaurantes

Como ya se ve en toda Latinoamérica, PedidosYa se ha instaurado en todas las casas, ofreciendo un sitio y aplicación que dispone de cientos de restaurantes en la ciudad que uno está y facilitando la elección, pedido y entrega de la misma.

Justamente en la entrega es que queremos enfocarnos, debido a que la logística aplicada en éste caso es perfecta. Ya que personas que no están en el medio, es decir, simples usuarios que disponen de tiempo y necesidad de transportar comidas

del restaurante a la persona que la compra, se encargan de realizar el proceso completo de la logística de los pedidos.

De ésta forma, interviniendo de lleno en los procesos logísticos de los locales de comida. Logrando a su vez parte de lo que el proyecto EnvioEntregas se evoca, en lograr una mejora sustentable social y ambiental en cada ciudad. Logrando que el transporte disminuya su carga.

Es así como cada vez más crece este servicio y la logística del mismo va creciendo a pasos agigantados, de tal manera es como se relaciona con la logística del e-commerce en el proyecto; invocando a usuarios que posean tiempo y necesidad de transportar y llevar destino bienes y productos que gente en éste caso, desee transportar a cierto punto de la ciudad.

3.2.4. La e-logística

Tratándose de un proceso tan novedoso. Todavía no se encuentra una definición “oficial” para el término e-logística o e-logistics. Pero podríamos decir que se trata de todos los procesos logísticos relacionados con una ecommerce. El objetivo principal que persigue la e-logística es que la relación comercial entre comprador y e-shop sea la mejor posible.

La logística en un comercio electrónico adquiere tanto peso y tanta importancia que tiene su propio término para designarla. Y es que cualquier empresa que vaya a empezar a funcionar online debe tener muy en cuenta aspectos directamente relacionados con la logística tales como la experiencia de compra del usuario, el transporte, el funcionamiento de los procesos de devolución, etc.

Además se hace también importante contar si es posible con profesionales que sepan controlar la situación logística de cada empresa, dominar la cadena de suministro y poder implementar soluciones y estrategias en entornos de comercio electrónico.

Sólo cuidando estos aspectos al máximo conseguiremos que el cliente repita experiencia con nosotros e incluso nos recomiende a otros de sus amigos y compañeros.

Son muchas las ventajas que da la e-logística al consumidor final como la comodidad de comprar desde su casa, no tener horarios ni de envío ni de compra, mucha más variedad en los productos y precios más competitivos.

Hay 3 pilares que fundamentan la e-logística:

Los sistemas de información logísticos: Para que la logística dentro de una ecommerce funcione correctamente es imprescindible contar con una o varias plataformas tecnológicas capaces de gestionar todo aquello referente a stocks, pedidos, devoluciones, etc. Serán el “puente” que una todos los procesos logísticos con los de la tienda en sí. Estos sistemas tecnológicos de los que hablamos son fundamentales para garantizar una experiencia de compra online satisfactoria y fidelidad para el cliente. En otras palabras es el puente que conecta el entorno online con la gestión offline. Un funcionamiento correcto que optimice al máximo los tiempos de entrega, tenga una trazabilidad de los pedidos, etc.

- El almacenamiento: Varios temas a tener en cuenta aquí;

La preparación de los pedidos más pequeños orientados al picking.
Que el packing sea de calidad. Hablamos del embalaje, etiquetado de productos y servicio de paquetería. Una de las partes más importantes en este punto ya que el embalaje debe ser el adecuado para que cada producto llegue sano y salvo a su destino. No es lo mismo si envías una caja entera de vinos que un paquete con ropa.
Lo ágil con lo que se trate el stock en tránsito y los sistemas de despacho, sistemas de despacho, preparación de pedidos. Todo bajo un sistema de seguimiento muy estricto y en tiempo real.

La distribución: Es el tercer pilar fundamental dentro de la e-logística, la logística aplicada al ecommerce y se trata de conocer de primera mano los puntos de

distribución en las ciudades donde se vaya a funcionar. Darle trazabilidad al proceso y tener siempre en cuenta el futuro para asegurar un crecimiento sostenido en el tiempo. Otro punto importante sería la flexibilidad en los horarios de entrega, servicio urgente, avisos de entrega al móvil, facturación y pago. El alcance de distribución que quieras dar es también importante antes de definir los procesos de logística de tu tienda online, ya que puede ser local, regional, nacional o internacional. Aquí tu partner en logística podrá dar solución a todos los requerimientos de tu negocio y tus clientes online.

Es por eso que en el proyecto la e-logística es el rol y concepto principal que se tiene en cuenta, ya que para cualquier empresa productora de bienes y servicios, le va a ser útil EnvioEntregas debido a que permite el traslado de los productos finales (ya sean bienes o servicios) y los pone a disposición del cliente. El canal de distribución es el que posibilita que el usuario obtenga el producto en el lugar, tiempo y cantidades adecuadas. La calidad de este servicio es fundamental para el éxito de cualquier empresa, y mayor aún si son plataformas de e-commerce.

En el mercado actual donde el cliente es el centro, se debe valorar mucho la unión de culturas que existe entre la logística y el e-commerce, un mercado en crecimiento que requiere un proceso de adaptación. Este crecimiento facilita la elección del consumidor, pero también desafía a las compañías a mantener un buen ritmo de satisfacción de las expectativas de los consumidores en entregas más rápidas y sin errores.

3.2.5. Impacto del Software y Procesos Logísticos en el Mercado Extranjero

A nivel coyuntural, la industria del software en la Argentina avanzó en 2015 a buen ritmo, con 81.800 profesionales y un crecimiento tanto en las ventas como en las exportaciones respecto al anterior período, ascendiendo a 3.479 millones de dólares y más de 1.000 millones respectivamente, de acuerdo a las cifras publicadas por Observatorio Permanente Software y Servicios Informáticos (OPSSI) de la Cámara de Empresas de Software y Servicios Informáticos de Argentina (CESSI). Estados Unidos

es el principal destino de las exportaciones de software argentino, y representa el 50,5 por ciento de los ingresos, seguido por Uruguay y México. En definitiva, Argentina tiene un buen posicionamiento en materia de software en diversos países de América. Esto le da una inmensa ventaja frente a otro tipo de soluciones que existen en el mundo.

Entorno a lo que es el desarrollo propio del software, el mismo se va a situar en la nube y podrá utilizarse mediante una PC, Smartphones, Tablet y cualquier otro tipo de dispositivo móvil con conexión a internet. Su costo de implementación es cero, por lo que llevar la solución a otros países no presentará grandes dificultades (si se logra primero el éxito en Argentina).
Por ende, el proyecto posee un gran potencial para comercializarse por fuera del país, trayendo importantes beneficios en materia de recaudación impositiva para el Estado.

Durante un panel titulado Logística: un puente para la prosperidad global, en el Global Forum de Wharton celebrado en Estambul el pasado mes de junio, el moderador George Day describió la logística como “el nexo de unión que hace funcionar a la economía”. La logística, dijo, puede ser “una enorme fuente de ventaja competitiva, además de ayudar a expandir y lanzar nuevos modelos de negocios”. Combinado con la tecnología de la información, añadió, la logística puede “extender dramáticamente el alcance geográfico de las organizaciones pequeñas y grandes”. Para explicar el creciente papel de la logística, Day, profesor de Marketing de Wharton, contó con la colaboración de Michel Akavi, consejero delegado de DHL Worldwide Express, Mirzan bin Mahathir, presidente del consejo y director general de la empresa con sede en Malasia, Konsortium Logistiks Berhad, y Yavuz Cizmeci, presidente de las líneas aéreas turcas ACT.

“La logística consiste en trasladar un producto concreto, en cantidades exactas, a un lugar concreto en un tiempo concreto”, declaró Day, profesor de Marketing de Wharton que ha estudiado la logística basada en el desempeño de empresas como Cisco Systems y General Electric. “Las cadenas de abastecimiento realmente buenas han conseguido reducir los costes de manera significativa, menor volumen de inventario y mejor prestación de servicios al consumidor. En el caso de Cisco, su

grupo de servicio post-venta es un negocio de 4.000 millones de dólares que distribuye 720.000 piezas de reposición a las distintas fábricas de la empresa. La logística de esa operación incluye clientes, ingenieros de campo y centros de pedido, distribución y reparación de materiales. “Cuanto más eficientemente se administre la logística, más efectivamente se disipa la incertidumbre del sistema”, dijo.

Ocho tendencias en logística

Michel Akavi, otro de los invitados, dijo a la audiencia que cuando él preguntó al organizador de la conferencia donde tendría lugar el panel, ella contestó que la gente “se atrasaría un poco, pero que poco a poco irían llegando. Yo dije: Genial. Necesitan una sesión de logística”.

Akavi proporcionó respuestas a las necesidades de logística con un discurso sobre lo que, en su opinión, son las ocho tendencias que actualmente influyen en la logística. La primera es la “explosión” de comercio y producción global debido al “cambio en el viejo orden político, especialmente la caída del comunismo. Además, las barreras comerciales han caído, especialmente en Europa, y hay un mayor comercio entre el Este y el Oeste del continente. Akavi además citó el NAFTA (Acuerdo Comercial de América del Norte), el MERCOSUR (Acuerdo comercial de Sudamérica), la Organización Mundial del Comercio (OMC), y el GATT (Acuerdo General de Tarifas y Comercio) como los responsables de la “ola de comercio internacional. Cuánto más acuerdos como estos haya, mayor será la necesidad de logística”.

Fíjense en Internet, señaló. Siendo una empresa de envío de documento puerta a puerta, “teníamos miedo de Internet. Pero afortunadamente, los documentos todavía necesitan ser firmados, cerrados y sellados… Esperemos que Turquía no adopte el mal hábito de las firmas electrónicas cuando se una a la UE”, dijo Akavi con una sonrisa, añadiendo que “los productos no viajan electrónicamente, gracias a Dios”. Cuánta más gente use Internet, mayor será el volumen de negocio, más pesados serán los paquetes y mayor será la necesidad de que las cartas viajen por el mundo.

La segunda tendencia es la transición a una sociedad post-industrial, dijo Akavi. “Tenemos una población estancada en los países occidentales; el promedio de edad está aumentando, se gasta más dinero en comunicación y sanidad, y menos en productos producidos en masa. La tendencia es más nichos de productos más transitorios e individuales combinados con servicios”. Esto significa que una mayor variedad de productos necesitan ser transportados, de manera más especializada, directamente a consumidores y usuarios”. Por tanto, la industria de logística debe especializarse en nichos, así como la industria textil requiere empresas que sean sensibles a las tendencias de moda. No puedes producir un millón de productos en un sitio en un momento determinado. Tienes que producirlos rápidamente”, a menudo en diferentes partes del mundo.

La tercera tendencia es que ahora vivimos en “un mundo on-demand (en que el consumidor dice lo que quiere, cuándo y cómo lo quiere), dijo Akavi. Estamos en una sociedad en la que el tiempo es dinero. Estamos moviéndonos a una competición basada en el tiempo. La velocidad es casi más importante que un precio barato. Lo vemos en la microelectrónica, con sus chips y consolas de juegos. En el segmento de los PC y los teléfonos, el término utilizado es agilidad, es decir, la capacidad para llegar el primero al mercado. La demanda está cambiando el mundo de la logística”.

La cuarta tendencia es un crecimiento de la sensibilidad en relación al medioambiente. La gente ahora se pregunta: “Cómo podemos transportar menos, de una manera más eficiente, y cómo podemos conseguir reciclar más”, dijo Akavi. “En Europa, vemos que el tráfico de camiones en las autopistas está sujeto a mayores restricciones. Austria está prohibiendo parte del tráfico de camiones durante el fin de semana. Los trenes se están utilizando con mayor frecuencia para transportar bienes porque se gasta menos energía. Existe, además, una mayor preocupación por los aviones ruidosos. Tenemos que cambiar nuestra flota en Bruselas por aviones menos ruidosos, estamos moviendo nuestra central de vuelos de Bruselas a Leipzig, en Alemania, una zona menos poblada. El cuidado del medioambiente está modelando la industria”.

La quinta tendencia es el “redescubrimiento de procesos de organización de los procesos estructurales”, basados en la mayor eficiencia y una mejor organización, y la sexta tendencia consiste en la “desregulación y privatización de los servicios públicos en comunicación y transporte. Somos un buen ejemplo de eso”, dice Akavi. El correo alemán (Deutsche Post), dueño de DHL, “era un sistema postal ineficiente y apático. Después de haber sido privatizado y modernizado, empezó a ser rentable. Entonces se preguntó cuáles deberían ser su misión; no podía limitarse a vender sellos el resto de su vida. Así que pasó de ser un servicio postal a convertirse en una empresa de logística y transporte integrados”.

La séptima tendencia enfatiza la generación de valor para el accionista. “La logística procura resaltar ahora las competencias básicas. Hemos visto cómo algunas empresas se desprenden de negocios para poder concentrarse en sus competencias básicas. Existe hoy en día una mayor subcontratación de la función del transporte”, lo que ayuda a terceros proveedores como DHL a expandirse y además contribuye al crecimiento de empresas de logística de transporte especializado.

La octava y última tendencia, según Akavi, son las tecnologías de comunicación más nuevas. “Con Internet, puedes saber donde está tu envío y contactar al centro de llamadas si el paquete está retenido en algún lugar. Pero ahora, también se pueden usar los teléfonos móviles para hacerlo. El rastreo y la localización son cada vez más comunes. Nuestra empresa puede seguir automáticamente todos los envíos y solventar cualquier problema antes de que el cliente se de cuenta de que su envío no ha llegado”. La tecnología RFID –identificación por radio frecuencia- tiene “una importancia enorme para nuestro sector. Sin RFID, sería muy difícil encontrar nuestro envío en nuestros gigantescos almacenes. Esta tecnología tendrá un impacto importante en los años venideros”.

El ejemplo de Singapur

Mirzan bin Mahathir, que describió su empresa con sede en Kuala Lumpur, Konsortium Loistiks Berhad, como una entidad dividida entre componentes y soluciones de logística estaba de acuerdo con el comentario de Akavi acerca de que las empresas están subcontratando la parte logística con mayor frecuencia para

dedicarse a sus competencias básicas. Continuó diciendo que “el puente a la prosperidad global tiene dos niveles”, el nivel país y el nivel empresa.

A nivel país, “los estados deben desarrollar su infraestructura logística para poder competir. No es suficiente atraer al sector de fabricación si no se puede comercializar los productos manufacturados de una manera eficiente”, dijo. La infraestructura incluye la construcción de puertos, aeropuertos, carreteras y puentes para transportar no sólo bienes, también gente. Los aeropuertos son especialmente críticos: “En Oriente Medio todo el mundo se está dando cuenta de que los aeropuertos son una pieza vital en la infraestructura”, por eso se están construyendo muchos, algunos inclusive bien cerca unos de otros.

Así todo, con la parte física no es suficiente. La infraestructura física no sirve si no se usa efectivamente”, señaló Mirzan. Se necesitan tres cosas: tecnología de la información, movimiento físico eficiente y un sistema financiero fiable. “En los países en desarrollo esas tres cosas necesitan ir de la mano. Esto ocurre en pocos lugares, como Singapur. Y está mejorando en algunos otros países también”.

En el negocio logístico, añadió Mirzan, los bienes necesitan ser mandados lo más directamente posible al mercado, y necesitan alcanzar su destino a tiempo. Así todo, varios obstáculos gubernamentales pueden torcer este proceso, incluyendo demasiada burocracia, inspecciones y corrupción. “Si el liderazgo de cada país valorara la logística para determinar su papel en la prosperidad de sus habitantes, vería que es necesario prestar atención a estas tres áreas”. Algunos países se dan cuenta de ello, dijo Mirzan; otros, no.

Desde el punto de vista de la empresa, añadió, algunos ya se están empezando a ver cómo empresas logísticas, incluso cuando están produciendo un producto cualquiera; es una ventaja competitiva. Dell, sugirió Mirzan, “de hecho es una empresa logística que actúa en el negocio de los ordenadores. Son muy eficientes en lo que hacen, pero actúan fundamentalmente en el área de la logística”.

Es por éstas cuestiones las que implica un cambio y/o innovación mundial en los procesos de logística e-commerce, llevando a cabo la idea que venimos comentando e interpretando.

Además hoy en día en el mundo entero, se tiene muy presente al medio ambiente. La humanidad está cada vez más concienciada por el medio ambiente y los temas sociales. Las empresas se verán obligadas en cierto modo a cambiar sus políticas de responsabilidad corporativa. Por ejemplo, en el mundo del transporte, los operadores logísticos tendrán muy en cuenta los transportes que respeten el medio ambiente.

Las políticas medio ambientales y aquellos procesos logísticos con un impacto medio ambiental menor serán valorados cada vez más tanto por los clientes particulares como por otras grandes empresas con las que se trabaje o colabores.

3.2.6. Procesos de Nuevo Pedido, Negociación y Envío.

Comenzando con la postura de generar un nuevo pedido para que sea enviado a algún punto, el usuario está obligado a indicar las características del producto a enviar y a su vez lugar de recogida, lugar de entrega, y cuándo lo desea. Siendo éste el caso donde un Usuario ya sea particular o de una empresa que produce bienes, desea que el producto sea enviado a algún sitio.

Una vez de completar el proceso de carga de datos, la plataforma retorna un costo estimado de envío, donde se le indica al usuario, y éste puede decidir cuándo está disponible a pagar (modificándolo si desea), después de ese paso, logra la publicación del Nuevo Pedido.

Por otro lado en el caso de ser un modelo de negocios también en e-commerce, los productos ya están cargados en el sitio, como así también el detalle y características de éstos, salvo la dirección de entrega, que la debe indicar el usuario que compra dicho producto.

Otro proceso importante es el que sigue denominado Negociación. Donde usuarios que quieren actuar como “Deliveries” tienen el tiempo y la necesidad de recoger un bien o producto en el lugar de origen y llevarlo a destino para lograr obtener dinero a cambio de ello. Es así cuando éste paso, se inicia por el usuario interesado en transformarse en un Delivery.

El mismo, puede ser notificado de nuevas órdenes por hacer, mediante un e-mail o una notificación en su aplicación, ya sea porque ha realizado alguna vez rutas similares o cercanas a la nueva orden que se carga, o porque el Usuario realiza búsquedas a través de la aplicación o sitio web, donde indica origen y destino y se le ofrecen diversos pedidos para que éste elija el más conveniente y comience a negociar para realizarlo.

Para dar inicio el Delivery enviar una oferta al Usuario, indicando cuando cobra por ese envío, en que momento puede hacerlo y el medio de transporte que posee. Al momento de enviar la oferta, se notifica al Usuario (vía mail y notificación push) que cargó el Pedido, que alguién está interesado.

Si el Usuario lo ve conveniente con respecto a lo que tiene que trasladar y en que, además del precio, puede Aceptar o seguir con la Negociación (Ambas acciones, se notifica al Delivery en éste caso, también vía mail y notificación push).

A su vez, existe una política de cancelación la cual está ligada a dependiendo el momento en que el Usuario decida cancelar el Pedido. Siempre tiene que ser un día previo al menos al dia de recogida, si no, no es posible cancelar el pedido.
En éstos también ambos Usuarios reciben la noticia que el Pedido ha sido cancelado, vía push y email.

Si el Pedido fue aceptado, el Delivery es notificado que aceptaron su oferta, vía mail y notificación push, y al Usuario le llegan dos mails con el Código de Envío y Código de Entrega. El Delivery al pasar a retirar el Paquete por el lugar de origen, solicita el Código de Envío al Usuario para validar que todo está en orden para iniciar el proceso de Envío. Una vez validado, el Pedido pasa a un estado: En Proceso.
Aquí el Delivery en el momento de hacer el transporte puede ir enviando ubicación actual, y notificarle al Usuario que va en camino por “tal lugar”. A su vez, el Usuario puede ir consultando vía Mensajes al Delivery por cualquier duda o consulta, y obviamente el Delivery responder a ello. Éste intercambio de mensajes son notificados via email y push.

Al llegar a Destino, el Destinatario corrobora el paquete, si está todo en condiciones, le indica el Código de Entrega al Delivery para que éste lo marque como Completado al Pedido, y se finaliza la “EnvíoEntrega”.

De ésta manera, se deja en claro que la comunicación en términos de Logística es lo más importante a cubrir y a lograr el buen funcionamiento en ella. Ya que en un mal intercambio de información entre: Usuario - Delivery, Delivery - Usuario y Usuario - Destinatario, puede provocar un mal menor o mayor lo que genera en muchas ocasiones, pérdida de tiempo, por lo tanto pérdida de dinero, malestar entre las partes, pedidos incompletos, inseguridades, incertidumbre y por sobre todo pérdida de confianza en el Servicio Logístico.

Es por eso que se busca afianzar y se afirma que los procesos de logística e-commerce de ésta manera se revolucionan y logran énfasis en el logro de los objetivos los cuales contienen siempre, el tiempo, orden y confianza en los procesos.

3.2.7. Rol Delivery en la logística (Reputación)

Para enfocarnos en el tema Entrega en la logística, observamos obviamente que es una parte fundamental sobre todo en la confiabilidad del lado del Usuario que espera recibir ese paquete, y a su vez la necesidad en la persona o empresa al hacer de eso un beneficio moral y económico.

Por lo tanto, es importante destacar que todo el proceso es basado en la confianza y rigidez en los pasos, ya que en el fallo de alguno puede dañar los parámetros nombrados. Es por eso que al provocar que cualquier persona o empresa en la ciudad que se transporte rutinariamente o no en la misma, con cualquier tipo de transporte, hasta sea caminando, pueda contar con la necesidad de querer realizar una envío de algún bien que a algún le urge. Lo que conlleva a se promueva mucho más el servicio y la costumbre de publicar para que alguien lo transporte, a menor costo, mejor tiempo (porque es el tiempo del Delivery el que se dispone, ya que posiblemente el mismo esté yendo a su trabajo o lugar de rutina, y lo tenga que hacer rápido) y mejor confiabilidad, debido a que no es relativamente una empresa de correo con la que en

repetidas ocasiones es confuso o complicado contactarse, sino que son personas comunes que reciben la consulta y contacto a través de la aplicación móvil y responden fácilmente, generando una conexión completamente bidireccional.

Al hablar de responsabilidad, también se habla de reputación. Es por eso que en cada pedido que se completa, ambas partes (Usuario y Delivery) pueden calificar al otro con respecto a la calidad del servicio que ha brindado o como se ha comportado el Usuario con el proceso. Ayudando de ésta forma a la comunidad del sitio, logrando interpretar la calidad de servicio que brindan los Deliveries y a éstos permitiéndoles saber con que Usuario están tratando, demás está decir, que ambos roles pueden intercambiarse debido a que una persona puede solicitar que envíen un pedido y en otras ocasiones realizar un Delivery por ejemplo.

Todas éstas cuestiones planteadas, hacen del Delivery un ente totalmente abierto y sentido necesario fundamental para la acción que cumple, logrando satisfacer la necesidad del Usuario, que siempre es lo primordial en la Logística.

3.2.8. Seguridad y protección del Pedido

Al hablar de la seguridad y protección del Pedido, nos referimos a que las personas que soliciten los envíos, puedan Asegurar dicho Pedido. Para de ésta forma evitar cualquier evento de robo, extravío, daños y demás que puedan ocurrir en o durante el proceso de envío, afectando al Usuario en primer lugar y a la empresa como consecuencia.

Con ese razonamiento, es que se estudia el tema de la seguridad y protección de un bien siendo dependiente de un servicio brindado.

Las compras por Internet se han expandido en todo el mundo. Un ejemplo como el que se cita ahora es el de España. Datos como que el 85% de los usuarios realizarán sus compras de Navidad por la red lo confirman. La cifra se extrae del último estudio Webloyalty, compañía de desarrollo de estrategias online, donde se indica que tanto

las mujeres como los hombres españoles son activos en las compras vía Internet con un 83% y un 82%, respectivamente.

Por su parte, el portal de moda Dawanda.es indica que los compradores acuden a Internet porque les aporta "comodidad y sencillez" al evitar acudir a tiendas físicas y grandes almacenes. También explica que en Internet pueden encontrar una mayor oferta de productos, la posibilidad de realizar sus compras en cualquier momento y la oportunidad de encontrar grandes descuentos y ofertas, además de adquirir artículos en otros países.

No obstante, a la hora de hacer un pedido por Internet, puede surgir cierta desconfianza. Que el artículo llegue en mal estado, que sea diferente del que se pidió o que nunca llegue a su destino son varios ejemplo de las preocupaciones que pueden surgir a la hora de efectuar el pedido.

- Compradores seguros

Por ello, existen varias plataformas online que aseguran al comprador el envío en buen estado del producto que han pedido. Por ejemplo, la plataforma lightinthebox.com permite al usuario asegurar los envíos y poder saber dónde se encuentran hasta que llegan a su destino. Al hacer el pedido, el cliente puede optar por contratar un seguro de envío por 1,99 dólares, que se encarga de cubrir cualquier pérdida o daño durante el envío del pedido.

La plataforma china de compras online Aliexpress garantiza a los usuarios que el pago del artículo al vendedor se realiza únicamente después de que se confirme la recepción de la compra. Además, explican que si no se recibe el artículo en un plazo de entrega estimado de 60 días máximo, se reembolsará el coste completo del producto. No obstante, los reembolsos no están disponibles si el envio no llega debido a factores que sean responsabilidad del comprador o a las circunstancias excepcionales ajenas al comprador y vendedor como el rechazo en aduanas o los desastres naturales.

Por otro lado, eBay no tiene ningún seguro para contratar. Sin embargo, tanto la plataforma como PayPal ofrecen a los usuarios las coberturas del programa Protección del comprador de PayPal. Este servicio te cubre si no recibes el pedido, si ha llegado roto o no era lo que se pidió. Las transacciones que la plataforma considera aptas reciben una cobertura que cubre el precio total de la compra y los gastos de envío originales.

No obstante, antes de efectuar la compra se debe comprobar si un artículo puede recibir esta cobertura. Esto se puede verificar al encontrar el icono de PayPal en la sección Conoce al vendedor del anuncio. Por otro lado, si compras un artículo y lo pagas a través de PayPal, el programa de protección cubrirá el valor total del artículo, incluidos los gastos de envío y embalaje, en caso de incidencia.

- ¿Qué seguros de envíos pueden contratar los vendedores?

Por otro lado, si te encuentras en el otro lado y eres un vendedor en una plataforma o tienes una tienda online, existen pólizas para asegurar tus envíos. Por ejemplo, compañías de envíos como Seur o TNT permiten contratar pólizas a Todo Riesgo para cubrir pérdidas o averías en los envíos nacionales e internacionales. Además, Correos utiliza varias modalidades de seguros para los envíos de paquetería estándares o a todo riesgo que cubren extravíos, robos en reparto, mercancía dañada o pérdidas del contenido.

- ¿Cómo se puede reclamar a la compañía?

Si el producto que envías se extravía o llega en mal estado, te tendrás que dirigir directamente a la compañía con la que contrataste el seguro. Probablemente, la empresa pida pruebas del estado de la mercancía, por lo que puede pedirte fotografías del producto antes del envío y después de recibirlo, además de documentos que acrediten su precio, como una factura.

- ¿Cómo comprar seguro en la red?

A la hora de comprar por Internet es conveniente asegurarse que el proceso de pago es seguro. La plataforma de pago PayPal indica que es importante que la tienda online utiliza Secure Sockets Layer (SSL), una herramienta que se dedica a codificar

información delicada. Es fácilmente identificable porque en el proceso de compra seguro aparece como un icono de un candado cerrado en la parte inferior de la ventana del navegador.

Otro punto que destaca la plataforma, es asegurarse de la acreditación del vendedor. Por ejemplo, si se compra en portales como eBay es importante comprobar los votos, comentarios y clasificaciones de compradores anteriores. También es recomendable confirmar que la tienda online tiene una política de reembolso y devoluciones y cerciorarse de algunos aspectos como el tiempo límite de devolución de un artículo.

Por otro lado, Paypal hace especial hincapié en tener cuidado con los artículos de gran demanda o de un valor muy alto. Según explica, los productos que son difíciles de encontrar o muy caros como los de informática, la joyería o la electrónica, requieren especial precaución. Es importante "realizar comprobaciones y dobles comprobaciones antes de realizar la compra", indican. Además, es esencial confirmar la autenticidad de los productos.

Paypal establece que los indicadores que pueden hacer saltar la alarma de posible fraude en un envío es el retraso del paquete, recibir un correo electrónico no solicitado de un vendedor con un producto similar al que se ha comprado o los típicos chollos que ofrecen por la web.

Comparar seguros antes de contratarlos
Antes de contratar un seguro para tu envío o producto, es importante que te cerciores de qué coberturas ofrece cada compañía. Para ello, puedes utilizar el Comparador de Seguros.es, con el que podrás ahorrar tiempo y dinero, además de contratar la póliza que más se adapta a tu situación. De esta forma, evitarás adquirir seguros que incluyan servicios que no necesitas.

Entre éstas cuestiones, se plantea al Usuario de la plataforma, que la única posibilidad de contratar un asegurar el pedido, es una. Debido a que la mayoría de los pedidos, van a ser bienes menores, según tenemos analizado el mercado y situación, y no va a requerir de grandes aseguradores ni costosos contratos. Es por eso que se

le ofrece una aseguradora que asegura el 100% del pedido, contra robo, daños y extravío. Brindando ésta vez al usuario una cuenta más de confianza en el servicio y en los procesos de logística e-commerce.

3.3. Marco teórico del campo de acción

3.3.1. Diseño Responsivo

3.3.1.1. Introducción

El diseño web adaptable (también diseño web adaptativo o responsivo; este último calco del inglés responsive web design), es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas. Hoy día las páginas web se ven en multitud de dispositivos como tabletas, teléfonos inteligentes, libros electrónicos, portátiles, PC, etcétera. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, sistema operativo o capacidad de memoria entre otras. Esta tecnología pretende que con un único diseño web, todo se vea correctamente en cualquier dispositivo.

El diseñador y autor norteamericano Ethan Marcotte creó y difundió esta técnica a partir de una serie de artículos en A List Apart,1 una publicación en línea especializada en diseño y desarrollo web, idea que luego extendería en su libro Responsive Web Design.
Tanto la idea como el propósito del diseño web adaptable fueron previamente discutidos y descritos por el World Wide Web Consortium (W3C) en julio de 2008 en su recomendación Mobile Web Best Practices bajo el subtítulo «One Web».2

Dicha recomendación, aunque específica para dispositivos móviles, puntualiza que está hecha en el contexto de One Web, y que por lo tanto engloba no solo la experiencia de navegación en dispositivos móviles sino también en dispositivos de mayor resolución de pantalla como dispositivos de sobremesa.3

El concepto de One Web hace referencia a la idea de construir una ‘Web para Todos’ (Web for All) y accesible desde cualquier tipo de dispositivo (Web on Everything).4

Hoy en día, la variedad de dispositivos existentes en el mercado ha provocado que la información disponible no sea accesible desde todos los dispositivos, o bien sea accesible pero con una experiencia de navegación muy pobre.
Como se sabe el uso de dispositivos móviles ha aumentado notablemente en los años 2010, en particular, el uso de tabletas y teléfonos inteligentes. La evolución de la navegación en Internet ha ido a la par, y ello ha popularizado la navegación en Internet mediante una creciente variedad de dispositivos y resoluciones de pantalla, lo que a su vez ha creado unas necesidades de adaptar la experiencia de la navegación web a ellos.

Con una sola versión en HTML y CSS se pueden cubrir todas las resoluciones de pantalla, con lo que el sitio web estará optimizado para distintos dispositivos y resoluciones de pantalla. Esto mejora la experiencia de usuario a diferencia de lo que ocurre, por ejemplo, con sitios web de ancho fijo cuando se acceden desde dispositivos móviles. De esta forma se reducen los costos de creación y mantenimiento cuando el diseño de las pantallas es similar entre dispositivos de distintos tamaños. También evita tener que desarrollar aplicaciones ad-hoc para cada sistema operativo móvil: iOS, Android, Windows Phone, BlackBerry OS, etcétera,5 aunque hoy en día las webs para móviles todavía no pueden realizar las mismas funciones que las aplicaciones nativas.

Desde el punto de vista del posicionamiento en buscadores, aparecería una única URL en los resultados de búsqueda, con lo cual se ahorrarían múltiples redirecciones y los fallos que se derivan de estas. También se evitarían errores al acceder al sitio web en concreto desde los llamados social links, es decir, desde enlaces que los usuarios comparten en medios sociales tales como Facebook o Twitter, y que pueden acabar en error dependiendo desde qué dispositivo se copió y desde qué dispositivo se intenta acceder a ese enlace.

3.3.1.2. Media Queries

Son un módulo de CSS que nos permite identificar el medio en el cual se está mostrando nuestro sitio y bajo qué condiciones para, en base a esa comprobación, aplicar estilos específicos.
Usando media queries, podremos saber, por ejemplo, cuándo el sitio se está presentado en una ventana que mide 768 pixeles de ancho o menos, o cuando se está enviando a imprimir.

Hay 2 formas de usar los media queries:

En la llamada al archivo CSS:
Usando este método le indicamos al navegador que utilice el archivo CSS que estamos cargando solo si las condiciones se cumplen:

<link rel="stylesheet" media="only screen and (max-width: 768px)" href="ejemplo.css" >
Con esta línea, que debiera ir dentro del head de nuestro HTML, estamos cargando la hoja de estilos ejemplo.css, pero sus propiedades solo aplicarán si el sitio se está viendo en una pantalla, dentro de una ventana de máximo 768 pixeles de ancho.
El problema con este enfoque es que cada vez que queramos cargar estilos de acuerdo a ciertas condiciones, tendríamos que cargar un archivo nuevo, y eso, por supuesto, hace nuestro sitio más lento al añadir solicitudes HTTP adicionales.

Dentro del archivo CSS:
Esta es la forma más recomendada, ya que nos permite tener múltiples comprobaciones, sin la necesidad de cargar diferentes archivos por cada una de las media queries que hagamos.
La forma de escribir un media query en CSS, para obtener el mismo resultado que en el ejemplo anterior, es la siguiente:

@media only screen and (max-width: 768px) {

```
  /* Aquí van todos los estilos CSS */
}
```

En los inicios de la era responsive, lo que se recomendaba era crear los estilos estándar para tu sitio y luego, al final de tu CSS, añadir ciertos media queries basados en dispositivos específicos y hacer que el sitio se adaptara a ellos.
Si bien este método sirvió durante un tiempo, la proliferación de dispositivos móviles con diferentes tamaños y resoluciones hicieron que este enfoque quedara obsoleto.
Lo que se recomienda hoy, es tener media queries destinados a cada área de la página y que la medida en la cual se haga el quiebre (o breakpoint por su nombre en inglés) dependa exclusivamente en el requerimiento de cada elemento. Esto significa en la práctica que ya no vamos a tener sólo 4 media queries, sino varios de ellos por cada sección.

Así entonces, si tenemos una cabecera que contiene un menú y un logo y notamos que estos elementos comienzan a tocarse cuando la ventana se reduce hasta 814 píxeles, entonces ese es el ancho que usaremos para crear el quiebre.

Un ejemplo del uso de media queries con este enfoque sería el siguiente:

```
/*
* Cabecera
*/

.logo {
        width: 300px;
        float: left;
}

.menu {
        width: 500px;
        float: right:
}

@media only screen and ( max-width: 814px ) {

        .logo {
                width: 200px;
        }
```

```
        .menu {
                width: 450px;
        }

}

@media only screen and ( max-width: 660px ) {

        .logo,
        .menu {
                width: 100%;
                float: none;
        }

}
```

Y esto mismo se puede repetir tantas veces como sea necesario, teniendo media queries propios para el pie de página, la barra lateral o cualquier otro elemento dentro de la página.

3.3.1.3. Tipografía

Las fuentes (tipografías) también tienen cabida dentro del mundo del CSS. De hecho, son muy importantes. La elección de una tipografía, su tamaño, color, interlineado y otras características pueden cambiar, de forma consciente o inconsciente, la forma en la que una persona interpreta o capta los contenidos de una página web.
Existen muchos estilos de fuentes, pero antes de profundizar en ello, veamos algunas de las características principales de una tipografía:

Serifa: Las fuentes o tipografías que utilizan serifa o gracia, son aquellas que incorporan unos pequeños adornos o remates en los extremos de los bordes de las letras.

Paloseco: Las fuentes o tipografías de paloseco o sin serifa (sans serif) son unas tipografías lisas, sin adornos o remates en los extremos de los bordes de las letras. Tradicionalmente, se han utilizado tipografías con serifa en medios impresos argumentando que dichos bordes ofrecen una mayor legibilidad que las tipografías de paloseco. Sin embargo, en medios digitales, estas últimas suelen ser más utilizadas puesto que dan un aspecto más limpio y claro que las tipografías con serifa.

Por otro lado, existe un tipo de fuente denominada fuente monoespaciada, que se basa en que cada una de sus letras tienen exactamente la misma longitud horizontal. Son muy útiles para tareas de programación o emuladores de terminal, donde se leen mejor líneas con estas características.

Existe un amplio abanico de propiedades CSS para seleccionar las fuentes o tipografías a utilizar. Empezaremos por la más lógica, la de seleccionar una familia tipográfica, que no es más que indicar mediante CSS el nombre de la tipografía que queremos utilizar

- font-family: Indica el nombre de la tipografia a utilizar.
- font-size: Indica el tamaño de la tipografia.

Con la primera propiedad, denominada font-family, podemos seleccionar se puede seleccionar cualquier tipografía simplemente escribiendo su nombre. Si dicho nombre está compuesto por varias palabras separadas por un espacio, se aconseja utilizar comillas simples:

```
div {
   font-family: Verdana;
   font-family: 'PT Sans';     /* Otro ejemplo */
}
```

Esta es la forma más básica de indicar una tipografía. Sin embargo, hay que tener en cuenta que estas fuentes sólo serán visibles si el usuario las tiene instaladas en su sistema.

Esto hace que esta tarea se convierta en algo muy complejo, puesto que los diferentes sistemas operativos (Windows, Mac, GNU/Linux) tienen diferentes tipografías instaladas (y carecen de tantas otras) así como que también entramos en temas de licencias y tipografías propietarias, que no se permiten utilizar en según que casos.

Otra de las propiedades más utilizadas con las tipografías es font-size, que permite especificar el tamaño que tendrá la tipografía que utilicemos.

Se pueden indicar tres tipos de valores:

- Medidas absolutas: Palabras clave como medium que representan un tamaño medio (por defecto), small: tamaño pequeño, x-small: tamaño muy pequeño, etc...

- Medidas relativas: Palabras clave como smaller que representan un tamaño un poco más pequeño que el actual, o larger que representa un tamaño un poco más grande que el actual.

- Medida específica: Simplemente, indicar píxeles, porcentajes u otra medida para especificar el tamaño concreto.

A las tipografías elegidas se les puede aplicar ciertos estilos, muy útil para maquetar los textos, como por ejemplo negrita o cursiva (italic).

La propiedad que utilizamos es font-style y puede tomar los siguientes valores:

- normal: Estilo normal, por defecto. Sin cambios aparentes.
- italic: Cursiva. Estilo caracterizado por una ligera inclinación de las letras hacia la derecha.
- oblique: Oblícua. Idem al anterior, salvo que esta inclinación se realiza de forma artificial.
-

Con la propiedad font-style podemos aplicar estos estilos. En la mayoría de los casos, se aprecia el mismo efecto con los valores italic y oblique, no obstante, italic muestra la versión cursiva de la fuente, específicamente creada por el diseñador, mientras que oblique es una representación forzosa artificial de una tipografía cursiva

Por otro lado, tenemos el peso de la fuente, que no es más que el grosor de la misma. También depende de la fuente elegida, ya que no todas soportan todos los tipos de grosor. De forma similar a como hemos visto hasta ahora, se puede especificar el peso de una fuente mediante tres formas diferentes:

- font-weight: normal | bold Medidas absolutas (predefinidas)

- font-weight: bolder | lighter Medidas relativas (dependen de la actual)
- font-weight: peso Medida específica (número del peso concreto)

Las versiones previas de CSS tenían una gran limitación: la imposibilidad de utilizar fuentes personalizadas. Es cierto que bastaba con utilizar la propiedad font-family y especificar el nombre de la fuente a utilizar. Pero resulta que no era tan bonito como podía parecer en un principio. Por varias razones:

Las fuentes especificadas mediante la propiedad font-family debían estar instaladas en el sistema donde se visualizaba la página web. En el ejemplo superior y al contrario que Times New Roman o Georgia (las cuales venían de serie en un sistema Windows), la tipografía Vegur no viene instalada de serie en Windows. Así pues, nosotros (y los usuarios con esa fuente instalada) veríamos la página web con dicha tipografía, pero en el navegador de otro usuario que no la tenga, no se encontraría la fuente y se pasaría a la siguiente alternativa (Times New Roman), y así sucesivamente.

Otro de los problemas que existían se basa en el mismo problema, pero con matices diferentes. Mientras que las tipografías que vienen en sistemas como Microsoft Windows de serie (Times New Roman, Verdana, Tahoma, Trebuchet MS...) se verían correctamente en navegadores con dicho sistema operativo, no ocurriría lo mismo en dispositivos con GNU/Linux o Mac. Y lo mismo con tablets o dispositivos móviles. Esto ocurre porque frecuentemente dichas tipografías son propietarias y tienen licencias que permiten usarse sólo en dispositivos de dicha compañía.
En definitiva, aunque teníamos los mecanismos, era complicado conseguir cambiar la fuente y obtener el mismo resultado de diseño en todos los navegadores y sistemas disponibles.

Actualmente, todo esto ha cambiado debido a CSS3 y la regla @font-face, la cual permite descargar una fuente o tipografía, cargarla en el navegador y utilizarla en nuestras páginas. Todo ello de forma transparente al usuario.

La regla @font-face permite crear uno (o varios) bloques donde definir las tipografías a cargar en el documento. En el ejemplo superior vamos a hacerlo con la fuente Open Sans, una tipografía libre creada por Steve Matteson para Google.

Básicamente, abrimos un bloque @font-face, establecemos su nombre mediante font-family y definimos sus características mediante propiedades como font-style o font-weight. El factor clave viene a la hora de «adjuntar» la tipografía, cosa que se hace mediante la propiedad src con los siguientes valores:

- local('Open Sans'): Si hay fuente 'Open Sans' instalada en el sistema, se ahorra la descarga.
- url(opensans.eot): Se enlaza tipografía en formato EOT Embedded OpenType
- url(opensans.ttf): Se enlaza tipografía en formato TTF True Type
- url(opensans.otf): Se enlaza tipografía en formato OTF Open Type
- url(opensans.svg): Se enlaza tipografía en formato SVG Scalable Vector Graphics
- url(opensans.woff): Se enlaza tipografía en formato WOFF Web Open Font Format

En la actualidad, es muy común utilizar Google Fonts como repositorio proveedor de tipografías para utilizar en nuestros sitios web por varias razones:

Disponen de un amplio catálogo de fuentes y tipografías libres y/o gratuitas.
Resulta muy cómodo su uso y simplifica mucho la utilización de tipografías externas.
El servicio está muy extendido y utiliza un sistema llamado CDN, que brinda ventajas de velocidad.

En la propia página de Google Fonts podemos seleccionar las fuentes con las características deseadas y generar un código HTML con la tipografía (o colección de tipografías) que vamos a utilizar. No perder de vista el medidor que Google pone a nuestra disposición, porque dependiendo del tipo de fuente o cantidad de características puede resultar una colección de tipografías muy pesadas que ralentizan nuestra página de forma notable.

Todo esto nos generará el siguiente código que será el que tendremos que insertar en nuestro documento HTML, concretamente, antes de finalizar la sección <head>:

<link href='http://fonts.googleapis.com/css?family=Open+Sans:300,400,700' rel='stylesheet' type='text/css'>

Como se puede ver, al añadir este código, estamos enlazando en el documento HTML un enlace al repositorio de Google, que nos genera automáticamente todo el código CSS necesario de @font-face especialmente diseñado para las características y tipografías escogidas.

Si además, por ejemplo, añadimos también la familia de tipografías Roboto (con grosor 400) y Lato (con grosor 300 y 400), el código necesario sería el siguiente:

<link href='http://fonts.googleapis.com/css?family=Open+Sans:300,400,700|Roboto:400|Lato:300,400' rel='stylesheet' type='text/css'>

De esta forma conseguimos cargar varias tipografías desde el repositorio de Google de una sola vez, sin la necesidad de varias líneas de código diferentes, que a su vez conectarían varias veces con Google, pudiendo hacerlo de una sola vez.

Para terminar, sólo necesitaremos añadir la propiedad font-family:'Open Sans', font-family:'Lato' o font-family:'Roboto' a los textos que queramos dar formato con dichas tipografías.

3.3.2. Geolocalización

La API de geolocalización es una de las nuevas tecnologías que nos trae HTML5, esta API se utiliza para obtener la posición geográfica de un usuario cuando accede a un sitio web siempre y cuando el usuario lo apruebe y esté navegando con un navegador que soporte esta tecnología.

Esta API utiliza varios medios para calcular la posición geográfica del usuario como el GPS, la dirección IP y otras señales de red, con las que obtenemos entre otros datos la latitud y la longitud del lugar desde el que se está navegando.

La mayoría de navegadores modernos soportan esta tecnología, Internet Explorer le da soporte a partir de la versión 9, Safari y Chrome a partir de la versión 5, Firefox a partir de la 3.5 y la mayoria de navegadores para Android o iPhone la soportan, por lo que a fecha de hoy, no hay ningún motivo para no usar esta API.

El uso de la API es muy sencillo, y se realiza de la siguiente manera. A través de JavaScript:

```
if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position) {
        var lat = position.coords.latitude;
        var lng = position.coords.longitude;
        var options = { position: new google.maps.LatLng(lat, lng) }
        var marker = new google.maps.Marker(options);
        marker.setMap(map);
        });
}
```

De ésta forma, la propiedad de sólo lectura *Navigator.geolocation* devuelve un objeto Geolocation que proporciona acceso web a la ubicación de un dispositivo. Esto permite ofrecer al sitio web o aplicación resultados personalizados basados en la ubicación del usuario.

Luego el método *Geolocation.getCurrentPosition()* se utiliza para obtener la posición de un dispositivo, donde se pasa un parámetro *position* que el mismo es un objeto el cual se puede acceder a las coordenadas "*coords"* , *latitude* y *longitude.*
Las últimas 3 líneas del código son las que dibujan el mapa y colocan el pin en las coordenadas obtenidas.

- La precisión de los resultados depende de muchos factores tales como:
- La ubicación del usuario
- La disponibilidad de fuentes de datos (como puntos de acceso inalámbricos y dirección IP)
- El dispositivo en sí

Aunque la primera impresión sea que sólo será útil para usuarios de navegadores móviles, la realidad es que éste utiliza otros medios además del GPS para calcular la ubicación del usuario, como por ejemplo a través de su dirección IP.
Para dar a entender mejor el tema, se explican los conceptos de ***georreferenciación****,* como la técnica de posicionamiento espacial de una entidad en una localización geográfica única y bien definida en un sistema de coordenadas y datum específicos. Es una operación habitual dentro de los sistema de información geográfica (SIG) tanto para objetos ráster (imágenes de mapa de píxeles) como para objetos vectoriales (puntos, líneas, polilíneas y polígonos que representan objetos físicos).
Por otro lado el ***geoetiquetado*** como el proceso de agregar información geográfica en los metadatos de archivos de imágenes, vídeos, sonido, sitios web, etc. que sirva para su georreferenciación. Por lo general estos datos suelen ser coordenadas que definen la longitud y latitud donde el archivo multimedia ha sido creado, aunque también puede incluir la altitud, nombre del lugar, calle y número de policía, código postal, etc. para posteriormente hallar sus coordenadas geográficas (véase geocodificación).

Entonces se llega a la conclusión que la geolocalización es representar de manera gráfica y entendible la posición geográfica de un "algo", donde ese algo puede ser cualquier cosa, persona, vehículo, imagen, video, etc.

Las maneras de obtener la Geolocalización es a través de dispositivos diversos, como smartphones, GPS, sistemas de navegación, cámaras con GPS integrado, algunos,

celulares, computadoras (conectadas a alguna red), entre muchos otros, pero estos son los más conocidos y quizá los más usados.

Así para obtener la **georreferenciación** se puede acudir a tres métodos:

- Por medio de GPS (Global Positioning System: sistema de posicionamiento mundial), estos obtienen su localización vía satélites, es la más usada y confiable de todas, por la cobertura, movilidad y sobre todo la precisión que ofrece. Ejemplo: GPS Dedicados, Sistemas de Navegación, algunos SmartPhones.

- Triangulación de antenas de redes celulares, es la más distribuida y la menos usada, esto debido a que no es muy difundido por las compañías, el costo no es muy accesible, en muchos casos las mismas compañías bloquean el acceso. Aunque la precisión no es tan buena, se aproxima bastantea lo real. Ejemplo: Celulares de las 2 últimas generaciones, algunos smartphones lo usan como alternativa, conexiones 3G.

- Por dirección IP, ya muchos scripts y nuevos estándares son capaces de ofrecer información sobre localización por medio de IP, sin embargo, es aún de las más inexactas en la mayoría de las ciudades, debido a que localiza el último nodo que distribuye señal al hogar u oficina, a veces puede ser muy cercano, aunque a veces puede no serlo, dependerá del proveedor de Internet, que tan bien mapeados y distribuidos estén sus nodos de conexión. De hecho si se está conectado mediante un Proxy o un Túnel a otra IP dará la localización de esa IP y no la del usuario realmente.

Los móviles más nuevos (desde hace un par de años para acá) incorporan receptores de GPS, o es decir, del Sistema de Posicionamiento Global. Estos terminales pueden dar la ubicación exacta. Pero el móvil también puede saber dónde está a través de otras señales, bien sea porque el GPS está apagado o porque el equipo no cuenta con esa opción.

Un teléfono móvil, es un sistema sofisticado de radiocomunicaciones en el que intervienen torres con antenas (estaciones base) y la red se dispone en múltiples celdas o células, que se encargan de enviar y recibir esas señales de radio. Los móviles tienen transmisores de baja potencia que le permiten comunicarse con la antena de la estación base más cercana.

Mientras una persona de desplaza de un lado a otro con su móvil, el terminal va saltando de una celda a otra en busca de “señal”. Las estaciones base se encargan de monitorizar la fuerza de señal del móvil. En sitios más rurales o remotos, las torres de comunicación suelen estar muy separadas y por eso a veces la señal es irregular. La señal también se puede interrumpir en lugares con muchas montañas o edificios altos.

Un movil sin GPS puede proporcionar información de su ubicación, esto gracias a la forma en que se comunica con la red de telefonía en general. Una computadora o una aplicación en el móvil puede determinar la localización gracias a tres cosas: la aproximación a la torres de telefonía; por el tiempo que tarda la señal en ir de torre a torre y por último, por la fuerza de la señal recibida. La localización ocurre gracias a la triangulación, combinación de las señales de radio entre (varias) torres de radio de la red y el teléfono.

Este método sin GPS es menos preciso. Esto debido a los múltiples obstáculos que se pueden atravesar en el camino de la señal, bien sea árboles, edificios, montañas. Es por eso, que programas como Google Maps en el teléfono, generalmente recomiendan encender el GPS para obtener información más precisa.

En caso de no tener esa opción, la aplicación también advierte del margen de error (en metros) que puede haber en la localización. Esa inexactitud también se plasma en otras aplicaciones, como Facebook (móvil), que permite chequear en el lugar desde dónde se escribe el post, o Facebook Messenger, que informa sobre el sitio desde dónde llega el mensaje.

Cuando no se tiene activo el GPS muchas veces la localización toma como referencia la torre más cercana de telefonía y por eso aparece una ciudad, barrio o calle distint a

la que está la persona, aunque, generalmente está relativamente cerca.
En el caso del GPS, el Sistema de Posicionamiento Global es una red compuesta por cerca de 30 satélites que orbitan la Tierra a una altitud de 20.000 kilómetros. Es un sistema diseñado originalmente por el gobierno de los Estados Unidos para la navegación militar. Hoy en día, es una tecnología accesible para muchos, bien sea desde un móvil, o un dispositivo portátil de GPS, como los que se utilizan en los autos. Cualquiera de ellos puede recibir las señales que los satélites envían.
Hay por lo menos cuatro satélites de GPS que están visibles en cualquier momento. Cada uno de ellos transmite una señal sobre su ubicación en intervalos de tiempo regular.

Dichas señales viajan a la velocidad de la luz y son interceptadas por el “receptor de GPS”, llámese Sony, Nokia, iPhone 5 o Garmin. El receptor calcula a qué distancia está de cada satélite según el tiempo que le tomó recibir el mensaje.

Debe hacer esto no sólo con la señal de un satélite, sino con la de al menos tres de ellos. El receptor debe “triangular” las señales y gracias a ellas puede determinar su ubicación. Este proceso es denominado trilateración. Cuantos más satélites “vea” el dispositivo, mayor será la precisión con la que la unidad GPS puede determinar dónde se encuentra.
Por cierto, los satélites de GPS cuentan a bordo con un “reloj atómico”. Estos relojes son mucho más precisos que los habituales y dan la hora exacta de la señal.
Hay otro sistema de posicionamiento basado en WiFi (WPS). Éste se utiliza cuando el GPS no es apropiado, como en el interior de un edificio que bloquea la señal. El posicionamiento WiFi aprovecha los puntos de acceso inalámbricos en las áreas urbanas. En estos casos se utiliza la intensidad de la señal recibida. Este sistema se utiliza muchas veces en combinación con el GPS, por eso, es común que al abrir Google Maps en el móvil, la aplicación también recomienda encender el WiFi del teléfono.

3.3.3. Notificaciones Push/Email

Las Notificaciones Push son mensajes que se envían de forma directa a dispositivos móviles (Smartphones y/o tablets) con sistema operativo iOS, Android, Blackberry y/o Windows Phone.

Las notificaciones Push ayudan a los desarrolladores independientes y dueños de aplicaciones a mantener informados a sus usuarios. Mediante la implementación de las mismas se puede interactuar con las personas que descargaron una aplicación y enviarles mensajes de forma directa a sus dispositivos móviles.

El mejor ejemplo para tener como referencia es WhatsApp: cuando nos envían un mensaje por este medio el "aviso" que nos aparece es una notificación push.

Lo que más destaca de las notificaciones push es su inmediatez, ya que no hace falta estar ejecutando la aplicación para que nos llegue. Aunque la tengamos apagada o en segundo plano, cada vez que el servidor reciba una información nueva nos avisará de su existencia, es decir, las notificaciones push despiertan al móvil esté o no ejecutando la aplicación.

Por definición, para que el servidor envíe el mensaje al usuario, éste se habrá tenido que suscribir previamente a sus canales de información, para que el servidor conozca a donde hay que enviar esa push, es decir, en el registro de un usuario en algún momento, hay que capturar el Registration ID en Android y el Devide Token en iOS y almacenarlo para poder enviar la push cuando el contenido esté disponible en alguno de estos canales, con ello lo enviarán al usuario según llegue.

Es importante que distingamos la tecnología Push de la Pull. La diferencia entre ambas reside en quién inicia la comunicación. En la tecnología Pull es el cliente el que la inicia. Esto es, cuando configuramos manualmente la frecuencia con la que queremos que el servidor nos avise cuando hay contenido nuevo. Por ejemplo, si configuramos la frecuencia de actualización de contenido a 5 o 10 minutos, este será el tiempo que tardemos en enterarnos si tenemos información nueva. El servidor no

nos alertará inmediatamente sino en función de esta frecuencia. Como ejemplo está el Pull down to refresh que consiste en arrastrar un elemento , generalmente hacia abajo para obtener actualización del contenido, es el usuario quien inicia esta acción.

¿Cómo funcionan las Notificaciones Push?

Básicamente desde el servidor se abren conexiones con Apple y Google y son estos quienes se comunican directamente con el móvil (Apple por APNS y Google por GCM)

En este sentido, fue Blackberry la primera plataforma que implementó la tecnología Push para comunicar a sus clientes la recepción de correos electrónicos de manera instantánea, lo que marcó una revolución en el sector de los dispositivos móviles. Esto fue posible gracias a que RIM (Research in Motion) firmó un convenio con las compañías telefónicas mediante el cual establecía una conexión abierta permanentemente con los servidores operados por RIM. A partir de aquí, han sido muchas las compañías las que empezaron a hacer uso de esta innovadora tecnología.

Actualmente, las aplicaciones que más se aprovechan de esta tecnología son las aplicaciones nativas y los desarrollos de aplicaciones híbridas en PhoneGap desarrollando un complemento nativo para ello. Muchos desarrolladores optan por su utilización para aumentar la interacción del usuario con la app.

Para incorporar notificaciones push en un proyecto se puede hacer de 2 maneras:

1) Desarrollando propiamente las notificaciones en el servidor y conectando con los servidores de Apple y Google e implementando en los móviles. Este desarrollo es complejo y dependiendo de la cantidad de push que se vayan a enviar consume datos de un servidor y sobre todo puede enviar muchas peticiones. También hay que tener en cuenta que tipos de push vamos a enviar y dónde se van a almacenar (si es un chat o si son mensajes) etc.

2) Incorporando alguna solución existente tipo Urban Airship: Consiste en incorporar un SDK a las aplicaciones móviles que conectan con un tercero que se encarga de enviar la información a Apple y Google. Estas plataformas generalmente cobran por cada notificación que se envía, si bien es cierto que el coste de cada push no es muy elevado. Generalmente implementar esta opción es más rápido y por tanto económico en cuanto al desarrollo, si bien dependes de otra plataforma para enviar notificaciones, los servicios que ofrecen pueden compensar el precio por cada notificación.

Con respecto a las notificaciones push en aplicaciones móviles, una muy usada es FCM Firebase Cloud Messages, dicha herramienta provee notificaciones a tu aplicación a ciertos dispositivos que desees.
FCM es una solución de mensajería multiplataforma que te permite enviar mensajes de forma segura y gratuita.

Con FCM, puedes notificar a una app cliente que un correo electrónico nuevo o que otros datos están disponibles para la sincronización. Puedes enviar mensajes de notificación para volver a atraer a más usuarios y aumentar su retención. Para los casos prácticos de mensajería instantánea, un mensaje puede transferir una carga de hasta 4 KB a una app cliente.

Acerca de los mensajes de FCM
Firebase Cloud Messaging (FCM) ofrece una amplia variedad de funciones y opciones de mensajería. La información de esta página tiene por objetivo ayudarte a comprender los diferentes tipos de mensajes de FCM y lo que puedes hacer con ellos.

Tipos de mensajes
Con FCM, puedes enviar dos tipos de mensajes a los clientes:

Mensajes de notificación, que a veces se consideran "mensajes de pantalla". El SDK de FCM maneja automáticamente estos mensajes

Mensajes de datos, que maneja la app cliente

Los mensajes de notificación contienen un conjunto predefinido de claves visibles para el usuario. Los mensajes de datos, por el contrario, sólo contienen pares clave-valor personalizados definidos por el usuario. Los mensajes de notificación pueden contener una carga de datos opcional. La carga de datos máxima para ambos tipos de mensajes es de 4 KB, excepto cuando se envían mensajes desde Firebase console. Esta consola impone un límite de 1024 caracteres.

	Caso de uso	Modo de envío
Mensaje de notificación	FCM muestra automáticamente el mensaje en los dispositivos del usuario final en nombre de la app cliente. Los mensajes de notificación tienen un conjunto predefinido de claves visibles para el usuario y una carga de datos opcional de pares clave-valor personalizados.	1. En un entorno de confianza como Cloud Functions o el servidor de apps, utiliza el SDK de Admin o los Protocolos del servidor de FCM: Configura la clave `notification`. Puede tener una carga de datos opcional. Siempre se puede contraer. 2. Usa el compositor de Notifications: ingresa el mensaje, el texto, el título, etc. y envíala. Ingresa datos personalizados para agregar una carga de datos opcional.
Mensaje de datos	La app cliente es responsable de procesar los mensajes de datos. Los mensajes de datos solo tienen pares clave-valor personalizados.	En un entorno de confianza como Cloud Functions o el servidor de apps, utiliza el SDK de Admin o los Protocolos del servidor de FCM: Configurar solo la clave `data`.

Si deseas procesar los mensajes en tu app cliente, usa mensajes de datos.

FCM puede enviar un mensaje de notificación que incluya una carga de datos opcional. En estos casos, FCM se encarga de mostrar la carga de notificaciones y la app cliente controla la carga de datos.

Mensajes de notificación

Para realizar pruebas, campañas de marketing y volver a generar la participación de los usuarios, puedes enviar mensajes de notificación con Firebase console.

Para enviar mensajes de notificación de forma programática con el SDK de Admin o los protocolos de FCM, configura la clave notification con el conjunto predefinido de opciones clave-valor necesarias para la parte del mensaje de notificación que es visible para el usuario. Por ejemplo, el siguiente es un mensaje de notificación con

formato JSON en una app de IM. El usuario debería ver un mensaje con el título "Portugal vs. Denmark" y el texto "great match!" en el dispositivo:

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "notification":{
      "title":"Portugal vs. Denmark",
      "body":"great match!"
    }
  }
}
```

Los mensajes de notificación se envían a la bandeja de notificación cuando la app se ejecuta en segundo plano. En el caso de las apps en primer plano, las devoluciones de llamada que controlan los mensajes son las siguientes:

- didReceiveRemoteNotification: en iOS
- onMessageReceived() en Android. La clave de notification del paquete de datos contiene la notificación
- onMessage() en Web/JavaScript

Consulta la documentación de referencia para ver la lista completa de claves predefinidas disponibles para crear mensajes de notificación:

- Objeto de notificación del protocolo HTTP v1
- Carga de notificaciones del protocolo HTTP heredado
- Carga de notificaciones del protocolo XMPP

Mensajes de datos

Configura la clave apropiada con tus pares clave-valor personalizados para enviar una carga de datos a la app cliente. Por ejemplo, el siguiente es un mensaje con formato JSON en la misma app de IM anterior, cuya información está encapsulada en la clave data común y la app cliente debe interpretar el contenido:

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "data":{
```

```
      "Nick" : "Mario",
      "body" : "great match!",
      "Room" : "PortugalVSDenmark"
    }
  }
}
```

El ejemplo anterior muestra el uso del campo común data de nivel superior, que los clientes de todas las plataformas interpretan como receptor del mensaje. En cada plataforma, la app cliente recibe la carga de datos en la devolución de llamada correspondiente:

En Android, una app cliente recibe un mensaje de datos onMessageReceived() y puede manejar los pares clave-valor según corresponda. La carga de datos se puede recuperar en el intent que se usó para iniciar la actividad. Consulta Cómo recibir mensajes en una app para Android.

En iOS, la carga de datos se encuentra en didReceiveRemoteNotification: . Consulta Cómo recibir mensajes en un cliente de iOS.

Para los clientes Web/JavaScript, la carga de datos se recibe en onMessage() cuando la página se encuentra en primer plano. Cuando la aplicación web se encuentra en segundo plano, la carga se entrega al controlador de mensajes en segundo plano del service worker. Consulta Cómo recibir mensajes en un cliente de JavaScript.
Mensajes de notificación con carga de datos opcional
Puedes enviar mensajes de notificación que contengan una carga opcional de pares clave-valor personalizados, ya sea de forma programática o a través de Firebase console. En el Compositor de Notifications, usa los campos Datos personalizados en Opciones avanzadas.

El comportamiento de la app cuando recibe mensajes que incluyen cargas de notificaciones y de datos depende de si la app se encuentra en segundo plano o en primer plano; en otras palabras, si está activa o no cuando recibe los mensajes.

- Cuando está en segundo plano, la app recibe la carga de notificaciones en la bandeja de notificaciones y solo maneja la carga de datos cuando el usuario presiona la notificación.
- Cuando está en primer plano, la app recibe un objeto de mensaje con ambas cargas disponibles.

El siguiente mensaje con formato JSON contiene la clave notification y la clave data

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "notification":{
      "title":"Portugal vs. Denmark",
      "body":"great match!"
    },
    "data" : {
      "Nick" : "Mario",
      "Room" : "PortugalVSDenmark"
    }
  }
}
```

- FCM a Multiplataformas

Los mensajes enviados mediante el protocolo FCM HTTP v1 pueden contener dos tipos de pares de claves JSON:

Un conjunto común de claves que pueden interpretar todas las instancias de la app que reciben el mensaje
Bloques de claves específicos de la plataforma que solo pueden interpretar las instancias de la app que se ejecutan en la plataforma especificada

Los bloques específicos de la plataforma brindan la flexibilidad de personalizar mensajes para diferentes plataformas a fin de garantizar que se manejen correctamente cuando se reciben. En muchos escenarios, es lógico usar tanto claves comunes como claves para plataformas específicas en un mensaje determinado.

➔ Cuándo usar claves comunes, siempre que apuntes a instancias de la app en todas las plataformas (iOS, Android y Web)

➔ Cuando envíes mensajes a temas

Las claves comunes que todas las instancias de la app pueden interpretar, independientemente de la plataforma, son message.notification.title, message.notification.body y message.data.

- Cuándo usar claves específicas de la plataforma
- Cuando desees enviar campos únicamente a plataformas específicas
- Cuando envíes campos específicos de ciertas plataformas, además de claves comunes
- Cuando desees enviar valores únicamente a plataformas específicas, no utilices claves comunes; utiliza bloques de claves específicos de la plataforma. Por ejemplo, para enviar una notificación solo a iOS y Web, pero no a Android, debes usar dos bloques de claves separados, uno para iOS y otro para Web.

Al enviar mensajes con opciones de entrega específicas, utiliza claves específicas de la plataforma para configurarlos. Puedes especificar diferentes valores por plataforma si lo deseas; pero incluso cuando desees especificar esencialmente el mismo valor en todas las plataformas, debes usar claves específicas para cada una. Esto se debe a que cada plataforma puede interpretar el valor de forma levemente diferente; por ejemplo, el tiempo de vida en Android es un tiempo de vencimiento que se configura en segundos, mientras que en iOS es una fecha de vencimiento.

Ejemplo: mensaje de notificación con opciones de entrega específicas de la plataforma
La siguiente solicitud de envío v1 envía el mismo título de la notificación y el mismo contenido a todas las plataformas, pero también envía algunas anulaciones a plataformas específicas. En detalle, la solicitud realiza lo siguiente:

- Configura un tiempo de vida prolongado para las plataformas Android y Web, al tiempo que establece la prioridad de los mensajes APN (iOS) en una opción baja.

- Configura las claves apropiadas para determinar la acción que se realizará cuando un usuario presione la notificación en Android y en iOS (click_action y category, respectivamente).

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "notification":{
      "title":"Match update",
      "body":"Arsenal goal in added time, score is now 3-0"
    },
    "android":{
      "ttl":"86400s",
      "notification"{
        "click_action":"OPEN_ACTIVITY_1"
      }
    },
    "apns": {
      "headers": {
        "apns-priority": "5",
      },
      "payload": {
        "aps": {
          "category": "NEW_MESSAGE_CATEGORY"
        }
      }
    },
    "webpush":{
      "headers":{
        "TTL":"86400"
      }
    }
  }
}
```

Android y habilita opciones similares en iOS y Web. Por ejemplo, el comportamiento de los mensajes "contraíbles" es compatible con Android a través de collapse_key de FCM, con iOS a través de apns-collapse-id y con JavaScript/Web a través de Topic. Para obtener detalles, consulta las descripciones de esta sección y la documentación de referencia relacionada.

- Mensajes contraíbles y no contraíbles

Un mensaje no contraible indica que cada mensaje individual se entrega al dispositivo. Un mensaje no contraible entrega cierto contenido útil, a diferencia de un mensaje contraible como un "ping" que no tiene contenido, destinado a comunicarse con el servidor y recuperar datos.

Algunos casos de uso típicos de mensajes no contraíbles son los mensajes de chat o los mensajes críticos. Por ejemplo, en una app de IM, querrás enviar cada mensaje, ya que cada mensaje tiene un contenido diferente.

En Android, existe un límite de 100 mensajes que se pueden almacenar sin colapsar. Si se alcanza el límite, todos los mensajes almacenados se descartan. Cuando el dispositivo vuelve a estar en línea, recibe un mensaje especial que indica que se alcanzó el límite. Entonces, la app puede manejar la situación de manera adecuada, por lo general a través de una solicitud de sincronización completa desde el servidor de apps.

Un mensaje contraible es un mensaje que se puede reemplazar por un mensaje nuevo si todavía no se ha enviado al dispositivo.

Un caso de uso común de los mensajes contraíbles son los mensajes que se utilizan para indicar a una app para dispositivos móviles que sincronice los datos del servidor. Un ejemplo sería una app de deportes que muestra marcadores actualizados a los usuarios. Solo el mensaje más reciente es relevante.

Para marcar un mensaje como contraible en Android, incluye el parámetro collapse_key en la carga útil del mensaje. FCM permite que el servidor de apps utilice un máximo de cuatro claves de contracción diferentes por dispositivo Android en cualquier momento. En otras palabras, el servidor de FCM puede almacenar simultáneamente cuatro mensajes contraíbles diferentes por dispositivo, cada uno con una clave de contracción diferente. Si excedes esta cantidad, FCM únicamente conserva cuatro claves de contracción, sin garantizar cuáles.

Por lo tanto los mensajes contraíbles son una mejor opción desde el punto de vista del rendimiento, siempre y cuando tu app no necesite usar mensajes no contraíbles. Sin embargo, si usas mensajes contraíbles, recuerda que FCM solo permite que el servidor de conexiones de FCM use un máximo de cuatro claves de contracción diferentes por token de registro en cualquier momento. No debes exceder esta cantidad, o podría causar consecuencias impredecibles.

3.3.4. Servidor Express aplicando NodeJS

En sitios web o aplicaciones web dinámicas, que accedan a bases de datos, el servidor espera a recibir peticiones HTTP del navegador (o cliente). Cuando se recibe una petición, la aplicación determina cuál es la acción adecuada correspondiente, de acuerdo a la estructura de la URL y a la información (opcional) indicada en la petición con los métodos POST o GET. Dependiendo de la acción a realizar, puede que se necesite leer o escribir en la base de datos, o realizar otras acciones necesarias para atender la petición correctamente. La aplicación ha de responder al navegador, normalmente, creando una página HTML dinámicamente para él, en la que se muestre la información pedida, usualmente dentro de un elemento específico para este fin, en una plantilla HTML.

Express posee métodos para especificar que función ha de ser llamada dependiendo del verbo HTTP usado en la petición (GET, POST, SET, etc.) y la estructura de la URL ("ruta"). También tiene los métodos para especificar que plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y como generar la visualización adecuada para cada caso. El middleware de Express, puede usarse también para añadir funcionalidades para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET. Puede utilizarse además cualquier sistema de trabajo con bases de datos, que sea soportado por Node (Express no especifica ningún método preferido para trabajar con bases de datos).

En las siguientes secciones, se explican algunos puntos comunes que se pueden encontrar cuando se trabaja con código de Node y Express.

- Hola Mundo en Express:

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hola Mundo!');
});

app.listen(3000, function() {
  console.log('Aplicación ejemplo, escuchando el puerto 3000!');
});
```

Las primeras dos líneas incluyen (mediante la orden require()) el módulo de Express y crean una aplicación de Express. Este elemento se denomina comúnmente app, y posee métodos para el enrutamiento de las peticiones HTTP, configuración del 'middleware', y visualización de las vistas de HTM, uso del motores de 'templates', y gestión de las configuraciones de las aplicaciones que controlan la aplicación (por ejemplo el entorno, las definiciones para enrutado ... et cetera.)

Las líneas que siguen en el código (las tres líneas que comienzan con app.get) muestran una definición de ruta que se llamará cuando se reciba una petición HTTP GET con una dirección ('/') relativa al directorio raíz. La función 'callback' coge una petición y una respuesta como argumentos, y ejecuta un send() en la respuesta, para enviar la cadena de caracteres: "Hola Mundo!".

El bloque final de código, define y crea el servidor, escuchando el puerto 3000 e imprime un comentario en la consola. Cuando se está ejecutando el servidor, es posible ir hasta la dirección localhost:3000 en un navegador, y ver como el servidor de este ejemplo devuelve el mensaje de respuesta.

- Importando y creando módulos:

Un modulo es una librería o archivo JavaScript que puede ser importado dentro de otro código utilizando la función require() de Node. Por sí mismo, Express es un modulo, como lo son el middleware y las librerías de bases de datos que se utilizan en las aplicaciones Express.

El código mostrado abajo, muestra como puede importarse un modulo con base a su nombre, como ejemplo se utiliza el framework Express . Primero se invoca la función require(), indicando como parámetro el nombre del módulo o librería como una cadena ('express'), posteriormente se invoca el objeto obtenido para crear una aplicación Express.
Posteriormente, se puede acceder a las propiedades y funciones del objeto Aplicación.

```
var express = require('express');
var app = express();
```

- ❏ Uso de API's Asíncronas:

El código JavaScript usa frecuentemente APIs asíncronas antes que sincrónicas para operaciones que tomen algún tiempo en completarse. En una API sincrónica cada operación debe completarse antes de que la siguiente pueda comenzar. Por ejemplo, la siguiente función de registro es síncrona, y escribirá en orden el texto en la consola (Primero, Segundo).

```
console.log('Primero');
console.log('Segundo');
```

En contraste, en una API asincrónica, la API comenzara una operación e inmediatamente retornará (antes de que la operación se complete). Una vez que la operación finalice, la API usara algún mecanismo para realizar operaciones adicionales. Por ejemplo, el código de abajo imprimirá "Segundo, Primero" porque aunque el método setTimeout() es llamado primero y retorna inmediatamente, la operación no se completa por varios segundos.

```
setTimeout(function() {
  console.log('Primero');
  }, 3000);
console.log('Segundo');
```

Usar APIs asíncronas sin bloques es aun mas importante en Node que en el navegador, porque Node es un entorno de ejecución controlado por eventos de un

solo hilo. "Un solo hilo" quiere decir que todas las peticiones al servidor son ejecutadas en el mismo hilo (en vez de dividirse en procesos separados). Este modelo es extremadamente eficiente en términos de velocidad y recursos del servidor, pero eso significa que si alguna de sus funciones llama a métodos sincrónicos que tomen demasiado tiempo en completarse, bloquearan no solo la solicitud actual, sino también cualquier otra petición que este siendo manejada por tu aplicación web.

Hay muchas maneras para una API sincrónica de notificar a su aplicación que se ha completado. La manera mas común es registrar una función callback cuando usted invoca a una API asincrónica, la misma sera llamada de vuelta cuando la operación se complete. Este es el enfoque utilizado anteriormente.

- ❏ Manejadores de Rutas:

En nuestro ejemplo anterior de "Hola Mundo!" en Express (véase mas arriba), definimos una función (callback) manejadora de ruta para peticiones HTTP GET a la raíz del sitio ('/').

```
app.get('/', function(req, res) {
  res.send('Hello World!');
});
```

La función callback toma una petición y una respuesta como argumentos. En este caso el método simplemente llama a send() en la respuesta para retornar la cadena "Hello World!". Hay un numero de otros métodos de respuesta para finalizar el ciclo de solicitud/respuesta, por ejemplo podrá llamar a res.json() para enviar una respuesta JSON o res.sendFile() para enviar un archivo.
El objeto que representa una aplicación de Express, también posee métodos para definir los manejadores de rutas para el resto de los verbos HTTP: post(), put(), delete(), options(), trace(), copy(), lock(), mkcol(), move(), purge(), propfind(), proppatch(), unlock(), report(), mkactivity(), checkout(), merge(), m-search(), notify(), subscribe(), unsubscribe(), patch(), search(), y connect().

Hay un método general para definir las rutas: app.all(), el cual será llamado en respuesta a cualquier método HTTP. Se usa para cargar funciones del middleware en una dirección particular para todos los métodos de peticiones. El siguiente ejemplo (de

la documentación de Express) muestra el uso de los manejadores a /secret sin tener en cuenta el verbo HTTP utilizado (siempre que esté definido por el módulo http).

```
app.all('/secret', function(req, res, next) {
  console.log('Accediendo a la seccion secreta ...');
  next(); // pasa el control al siguiente manejador
});
```

Las rutas le permiten igualar patrones particulares de caracteres en la URL, y extraer algunos valores de ella y pasarlos como parámetros al manejador de rutas (como atributo del objeto petición pasado como parámetro).

- ❏ Middlewares:

El "middleware" es ampliamente utilizado en las aplicaciones de Express: desde tareas para servir archivos estáticos, a la gestión de errores o la compresión de las respuestas HTTP. Mientras las funciones de enrutamiento, con el objeto express.Router, se encargan del ciclo petición-respuesta, al gestionar la respuesta adecuada al cliente, las funciones de middleware normalmente realizan alguna operación al gestionar una petición o respuesta y a continuación llaman a la siguiente función en la "pila", que puede ser otra función de middleware u otra función de enrutamiento. El orden en el que las funciones de middleware son llamadas depende del desarrollador de la aplicación.

La mayoría de las aplicaciones usan middleware desarrollado por terceras personas, para simplificar funciones habituales en el desarrollo web, como puede ser: gestión de cookies, sesiones, autentificado de usuarios, peticiones POST y datos en JSON, registros de eventos, etc. Puede encontrar en el siguiente enlace una lista de middleware mantenido por el equipo de Express (que también incluye otros paquetes populares de terceras partes). Las librerías de Express están disponibles con la aplicación NPM (Node Package Manager).

3.3.5. Aplicaciones Móviles en base a Ionic

3.3.5.1. Introducción

El mundo de la programación y el desarrollo es un campo en constante evolución al que se van incorporando nuevas tecnologías o metodologías a fin de simplificar la tarea de creación de software de cualquier tipo.

Hoy, al igual que hicimos con Meteor, descubrimos Ionic Framework, un marco de desarrollo que junto con AngularJS nos facilitará sobremanera la tarea de creación de aplicaciones , que parece ser el campo en el que se especializa esta herramienta.

Desarrollado sobre AngularJS y lanzada su primera beta en el 2013, Ionic utiliza su base para proporcionarnos la estructura de aplicación mínima sobra la que poder comenzar a trabajar , mientras que Ionic en sí nos ofrecerá facilidades en el desarrollo de la interfaz de usuario. Con esta dupla, AngularJS con su versatilidad y potencia para la creación de aplicaciones e Ionic Framework para el desarrollo de la interfaz , obtenemos una herramienta de creación de aplicaciones completísima, con la que ahorraremos tiempo y trabajo en el desarrollo de cada proyecto.

Ionic proporciona un conjunto de directivas desde Angular (como elementos HTML personalizados) que podrán ser usados por sus propios módulos, por lo que construir una app será prácticamente como incorporar un “widget” al código , como si de líneas de código se tratase.

Otras características que incorpora Ionic Framework para el desarrollo de aplicaciones son el reconocimiento táctil, lógica de animación de interfaces, verificador HTML o comunicación asíncrona .

Además y para facilitar la usabilidad de la herramienta, aunque Ionic puede comenzar a ser usado justo tras descomprimir sus librerías en nuestro sistema, podremos añadirle una interfaz o CLI basada en Node.JS

De momento AngularJS es la base de Ionic Framework, pero según sus desarrolladores, no descartan incorporar o versionar este framework sobre otros cimientos para abarcar más público y sectores a alcanzar . En algunas de sus entrevistas han mencionado algunos como KnockOut o EmberJS, pero de momento mantienen AngularJS como base estable, como decía antes, por su estabilidad, rapidez y potencia.

3.3.5.2. Patrones MVC y MVVM

En sus inicios AngularJS basaba su estructura en MVC (Model View Controller), pero con el paso de los años se ha orientado más hacia MVVM (Model View ViewModel) obteniendo mejores resultados en la previsualización de un proyecto sin necesidad de recargar el proceso una y otra vez, ya que está directamente vinculado a la capa de visualización en lugar de referenciar los componentes.

Imperando hoy día entonces, el modelo MVVM, se separará eficientemente la presentación de la información (lógica de interfaz) de la efectividad del programa en sí mismo. Asimismo, esta flexibilidad otorga a los desarrolladores tiempos de producción más cortos y eficientes , aunque por supuesto, se podrá usar el modelo que más convenga en cada proyecto y según cada programador

Su significado se basa en patrones de diseño o modelos de abstracción utilizados para definir y estructurar los componentes necesarios en el de desarrollo de software.

Una manera muy simple de describirlos sería:
MVC significa Modelo Vista Controlador, porque en este patrón de diseño se separan los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. Cuando la lógica de negocio realiza un cambio, es necesario que ella sea la que actualiza la vista.

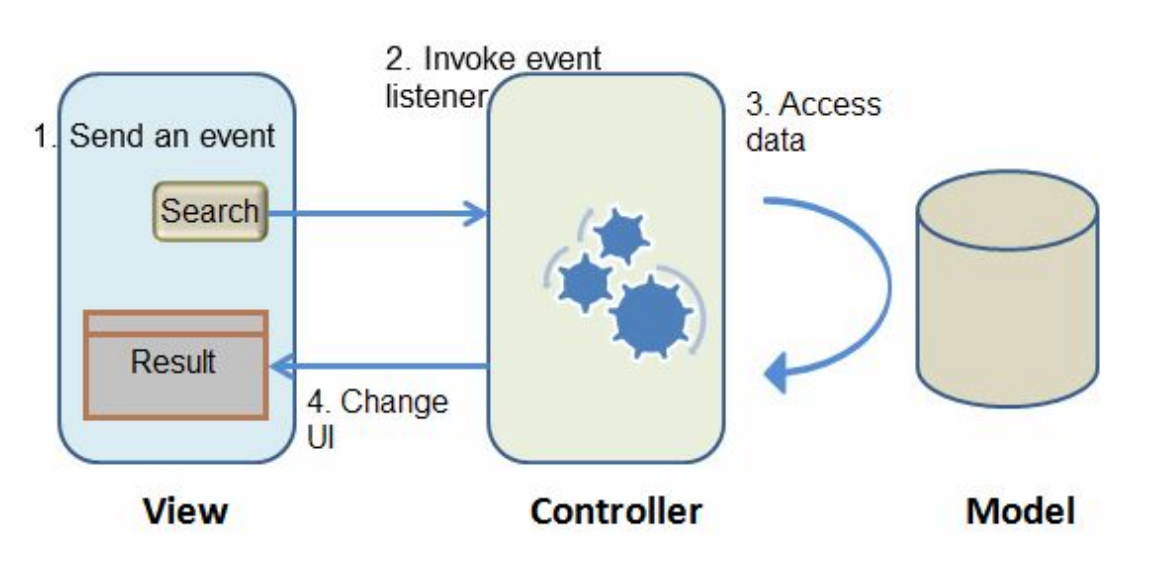
MVVM significa Modelo Vista VistaModelo, porque en este patrón de diseño se separan los datos de la aplicación, la interfaz de usuario pero en vez de controlar manualmente los cambios en la vista o en los datos, estos se actualizan directamente

cuando sucede un cambio en ellos, por ejemplo si la vista actualiza un dato que está presentando se actualiza el modelo automáticamente y viceversa.

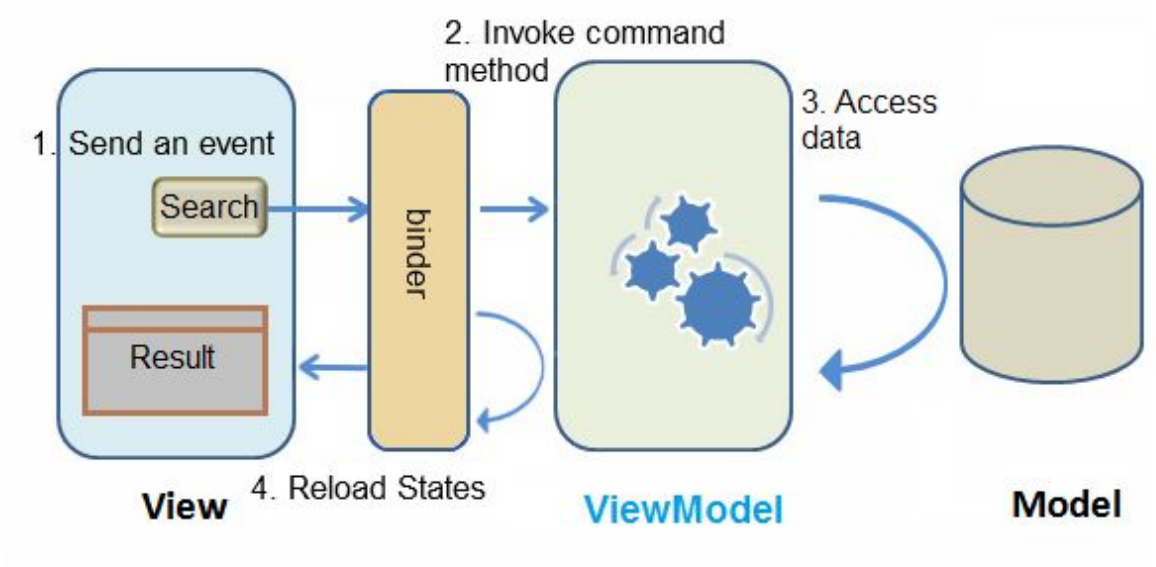
La primera imagen describe la interacción que se produce con MVC en ZK, y la siguiente es la correspondiente al enfoque MVVM que hace ZK.

Las principales diferencias entre MVC y MVVM son que en MVVM el “Controller” cambia a “ViewModel” y hay un “binder” que sincroniza la información en vez de hacerlo un controlador “Controller” como sucede en MVC.

MVC



MVVM



Construir una aplicación mediante MVC puede resultar más intuitivo, porque directamente controlas lo que ves en la vista y su comportamiento, es decir manualmente. MVC se caracteriza porque tienes control total de los componentes, por lo tanto puedes crear componentes hijo dinámicamente (“child”), controlar componentes propios personalizados, o realizar cualquier cosa sobre el componente que este pueda hacer por sí mismo.

En el patrón MVVM, puesto que la capa “ViewModel” esta débilmente acoplada con la vista (no está referenciada a los componentes de la vista), podemos usarla con múltiples vistas sin tener que modificarla. Por lo tanto los diseñadores y programadores pueden trabajar en paralelo. Si la información y comportamiento no cambian, un cambio en la vista no provoca que se tenga que modificar la capa “ViewModel”. Además, como la capa “ViewModel” es un POJO, es fácil realizar tests unitarios sobre ella sin ninguna configuración ni entorno especial. Lo que significa que la capa “ViewModel” tiene una mejor reusabilidad, testabilidad, y los cambios en la vista le afectan menos.

Comparación específica:

	MVC	MVVM
Acoplamiento con la vista	Muy poco con plantilla	Muy poco
Acoplamiento con el componente	Un poco	Muy poco
Codificar en la vista	Mediante el ID del componente	A través de una expresión Data binding
Implementación de un controlador	Extendemos ZK's Composer	Es un POJO
Acceso a la información de la UI	Acceso directo	Automático
Acceso a la información desde el backend	Acceso directo	Acceso directo
Actualización de la interfaz de usuario	Manipulamos directamente los componentes	Automático (@NotifyChange)
Nivel de control del componente	Elevado, control total	Normal
Rendimiento	Alto	Normal

3.3.5.3. Sistema de Clases

Ni AngularJS ni Ionic son marcos de desarrollo orientado a objetos JavaScript , por lo tanto no harán uso de un sistema de clases. Esta característica puede estar tanto en el lado de los pros como en el de los contras para la valoración de su implementación en un proyecto.

3.3.5.4. Control DOM

AngularJS incluye j qLite, una pequeña versión ultraligera de jQuery que permite la gestión de DOM (Document Object Model, Modelo de Objetos del Documento) de forma compatible en la inmensa mayoría de navegadores, dejando una pequeñísima huella en los mismos haciendo la app mucho más ligera y eficiente. Por supuesto, si con jqLite no tuviésemos suficiente, podremos implementar una funcionalidad extendida cargando jQuery con el documento, con la consecuente pérdida de velocidad.

3.3.5.5. Interfaz y Temas

De una forma similar (que no igual) a como hace Google con Android, Ionic procura hacer uso de las capacidades que ofrecen HTML5 y CSS3 para ofrecer experiencias de usuario sobretodo rápidas.

Como la interfaz de usuario que por defecto facilita Ionic es prácticamente HTML5 en su estado más puro , la personalización de ésta se realiza mediante SASS junto con las librerías propias que incorpora Ionic, con más de 450 iconos y símbolos de código abierto para su libre uso, además de otras utilidades con las que extender esta capacidad de personalización.

3.3.5.6. Widgets

Los componentes de Ionic son simples. Se trata de elementos HTML personalizados que incorporan controladores para complementar configuración e interacción . A diferencia de otros framework donde podemos encontrar widgets más complejos, Ionic ofrece bloques de código más simples y orientados a que sean combinados para ofrecer una interfaz de usuario más rica, intuitiva y atractiva para el público receptor.

3.3.5.7. Diseño Responsivo

Ionic, es un framewoek que a su Diseño lo implementa a través de SASS (Pedazos de Estilos Sintácticos Increibles, por sus siglas en Inglés).

La principal ventaja de SASS es la posibilidad de convertir los CSS en algo dinámico. Permite trabajar mucho más rápido en la creación de código con la posibilidad de crear funciones que realicen ciertas operaciones matemáticas y reutilizar código gracias a los mixins, variables que nos permiten guardar valores. SASS, en definitiva, se convierte en tu mejor ayudante.

Y algo siempre importante cuando te decantas por una herramienta con alternativas, dispone de una gran comunidad que la hace progresar, por lo que se le augura un gran futuro por delante.
ASS dispone de dos formatos diferentes para la sintaxis, lo que hace se traduce en dos extensiones de fichero diferentes: .sass y .scss
El primero en salir fue .sass y se caracteriza, al igual que Stylus y coffeescript para el lenguaje de programación Javascript, en no hacer uso de llaves ni punto y coma final, en busca de la simplicidad y eliminación de ruido
Los .scss salieron con la versión 3 de preprocesador y permiten utilizar llaves e incorporar código de CSS clásico.

Tanto la sintaxis de .sass como la de .scss no puede ser interpretada directamente por los navegadores, por lo se debe compilar para traducir nuestro archivo SASS en un clásico fichero CSS.
Algunas funcionalidades básicas de SASS utilizando la sintaxis .scss

3.3.5.8. Soporte para PC's

Ionic está destinado principalmente a crear aplicaciones para dispositivos móviles, como última instancia, dispositivos híbridos. En su desarrollo no se pensaba en cómo desarrollar o disponer información para equipos de sobremesa o portátiles. No obstante la información se mostrará igualmente, pero el rendimiento óptimo se obtiene cuando se ejecutan los proyectos creados en los dispositivos para los que está diseñado este Framework.

3.3.5.9. Comunidad

Al tratarse de un framework de desarrollo muy reciente, su comunidad aún no es equiparable a la de otros sistemas de desarrollo del mercado, no obstante, aquellos usuarios que la componen compensan su escaso número con un ímpetu de colaboración y resolución de incidencias sorprendente.
Aunque debido a la popularidad que está alcanzando este framework durante este 2016, el número de usuarios que aportan a la comunidad se ha incrementado más allá de lo que los propios desarrolladores esperaban para tan corto plazo de tiempo en el mercado.

3.3.6. Importancia de los Frameworks - PHP Laravel y Blade

3.3.6.1. Introducción

Laravel es un Framework (metodología) de trabajo para PHP, también se puede decir que es una herramienta que cuenta con la metodología para realizar cualquier desarrollo de cualquier sistema que se requiera, ya que incorpora módulos que nos ayudarán a desarrollar de una mejor forma, y sobre todo de una manera estándar en donde cualquier otro desarrollador que conozca Laravel pueda colaborar en nuestros proyectos.

La principal metodología que debemos manejar para utilizar Laravel viene siendo la de MVC (Modelo Vista Controlador) ya que es la base del desarrollo en este framework y en muchos más. En donde también necesitamos comprender que todo inicia cuando definimos nuestras primeras rutas, que es en donde parte todo el desarrollo de Laravel.

Las rutas vienen siendo las URL de nuestro sistema, luego estas rutas se enlazan con nuestro Controlador. Y en nuestros controladores podemos enviar variables a nuestras Vistas y también podemos consultar nuestros Modelos que son los objetos que estan enlazados a nuestras tablas de nuestra base de datos.

3.3.6.2. Herramientas

Laravel esta compuesto de herramientas externas para poder trabajar de una mejor manera y de forma integrada con este framework.

- Composer: es una herramienta para la gestión de las dependencias en PHP. Nos permite declarar las bibliotecas que necesitaremos en nuestros proyectos y de cuales libreras depende y Composer nos ayudara a (instalar / actualizar) de una manera sencilla y practica.

- Blade: es un motor de plantillas, Blade es simple pero potente y esta integrado ya en Laravel. A diferencia de otros motores de plantillas de PHP, Blade no restringe el uso de código PHP normal en sus vistas. De hecho, todas las vistas se compilan en código PHP simple y son guardados hasta que sean modificadas con esto no generar lentitud en la carga de tu aplicación utilizando Blade. Los archivos de vista de Blade utilizan la extensión de archivo .blade.php y normalmente se almacenan en el directorio Resources / views.

- Artisan: es la interfaz en línea de comandos (shell) que se incluye con Laravel. Proporciona una serie de comandos útiles que nos ayudaran mientras se construye su aplicación. Por ejemplo cuando queremos crear una tabla en nuestra base de datos se realizar con migraciones de Laravel (algo que veremos más adelante) y esto funciona por la vía de Artisan desde consola.

- Configuración de variables de entorno: algo muy bueno de Laravel son sus variables de entorno ya que es útil tener diferentes valores de configuración basados en el entorno de la aplicación o sea en donde ejecuta la misma. Por ejemplo, es posible que desee utilizar un controlador de caché diferente en ambiente de desarrollo que en el de producción. Para que esto simple laravel utiliza la biblioteca DotEnv PHP. En una instalación nueva laravel, el directorio raíz de la aplicación contendrá un archivo .env.example. Si instala a través de

laravel composer, este archivo automáticamente se cambiará el nombre a .env. De lo contrario, se debe cambiar el nombre del archivo de forma manual.

- PHP namespaces: aunque cualquier código válido de PHP puede estar contenido dentro de un namespace, sólo los siguientes tipos de códigos se ven afectados por los espacios de nombres: clases, interfaces, funciones y constantes. Los namespaces se declaran usando la palabra clave namespace. Un archivo que contiene un namespace debe declarar el namespace en la parte superior del archivo antes de cualquier otro código – con una excepción: la palabra clave declare.

- Json: JavaScript Object Notation, es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

- Git: Software de control de versiones, pensado en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

3.3.6.3. Componentes

A su vez, laravel está compuesto por diferentes componentes que hacen de su funcionamiento potencial:

❏ Migraciones

Cuando creamos nuestras bases de datos solemos crear diagramas que nos facilitan la abstracción de como se va a almacenar nuestra información, pero la forma de llevarlo a la realidad en algun gestor de bases de datos, como por ejemplo: MySQL, SQLite, PostgreSQL, SQL Server, etc., lo más comun es meternos al lenguaje de script encargado de implementar nuestra idea de la BD y ejecutar dicho script, o incluso ocupar programas más avanzados que nos sirven como interfaz para crearlas de una forma más gráfica y sin la necesidad de profundizar demasiado en el lenguaje, como Workbench o Navicat.

En Laravel se lleva a otro contexto esta situación, puesto que visto de la forma tradicional si se requieren cambios en la base de datos tenemos que meternos ya sea a otro programa para cambiar el diagrama de la base o a un archivo SQL con una sintaxis usualmente complicada o difícil de leer y ejecutar los cambios para reflejarlos en el proyecto, sin embargo, con esto no contamos con un control de los cambios (control de versiones) sobre la base de datos, si necesitamos consultar un cambio anterior o de repente la solución previa o inicial era la que se necesita al momento

debemos re-escribir todo otra vez, cosa que con la migraciones se soluciona instantáneamente.

Las migraciones son archivos que se encuentran el la ruta database/migrations/ de nuestro proyecto Laravel, por defecto en la instalación de Laravel 5 se encuentran dos migraciones ya creadas, create_users_table y create_password_resets_table

Para crear nuestras migraciones en Laravel se usa el siguiente comando:

php artisan make:migration nombre_migracion

que nos crea el archivo limpio para escribir nuestra migración, o bien el comando:

php artisan make:migration nombre_migracion --create=nombre_tabla

que nos agrega una plantilla de trabajo básica para empezar a trabajar.

Como ejemplo del curso se tomará este comando:

php artisan make:migration crear_tabla_pasteles --create=pasteles

el cual nos dará este resultado:

Created Migration: 2015_06_23_054801_crear_tabla_pasteles

Y nos creará además el siguiente archivo:

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CrearTablaPasteles extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('pasteles', function (Blueprint $table) {
            $table->increments('id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('pasteles');
    }
}

```

Ahora bien se puede observar que el archivo como tal no se llama simplemente crear_tabla_pasteles sino 2015_06_23_054801_crear_tabla_pasteles, esto pasa porque Laravel al crear una migración agrega como préfijo la fecha y hora en la que fué creada la migración para poder ordenar qué migración va antes que otra, por lo cual si tu ejecutas este comando, obviamente el nombre de tu archivo será diferente pues la fecha y hora no pueden ser las mismas que la mia al crear este ejemplo. Además las migraciones que vienen por defecto en Laravel también se encuentran con este formato por lo cual podemos observar que estos dos archivos si tienen el mismo nombre.

Dentro de la estructura del archivo podemos ver dos funciones, una llamada up() y otra llamada down(), la primer función es en donde vamos a especificar la estructura de nuestra tabla, inicialmente y gracias al comando se encuentran ya algunas cosas escritas como lo son la clase Schema en la cual se llama al método create, el cual nos permite crear la tabla en nuestra base de datos, esta recibe dos parámetros, el primero es el nombre que va a recibir la tabla que si no mal recuerdan es el que se le dio en el comando y por lo cual ya se encuentra en su lugar, y el segundo parámetro es una función closure o función anónima que lo que hace es definir las columnas de

nuestra tabla, a su vez esta función anónima recibe como parámetro un objeto de tipo Blueprint que se agregó dentro del namespace con la palabra use en la cabecera del archivo, el objeto $table es con el que vamos a trabajar para definir los campos, como se ve en la imagen anterior esto se logra escribiendo $table->tipo_dato('nombre');, y esto puede variar dependiendo el tipo de dato que se use y para ello podemos revisar la documentación oficial de Laravel aquí para poder ver todos los tipos de campos con los que contamos.

- ❏ Modelos y uso de Eloquent

En Laravel podemos hacer uso de un ORM llamado Eloquent, un ORM es un Mapeo Objeto-Relacional por sus siglas en ingles (Object-Relational mapping), que es una forma de mapear los datos que se encuentran en la base de datos almacenados en un lenguaje de script SQL a objetos de PHP y viceversa, esto surge con la idea de tener un codigo portable con el que no tengamos la necesidad de usar lenguaje SQL dentro de nuetras clases de PHP.

Eloquent hace uso de los Modelos para recibir o enviar la información a la base de datos, para esto analizaremos el modelo que viene por defecto en Laravel, este es el modelo User que se ubica en la carpeta app/, los modelos hacen uso de PSR-4 y namespaces, un modelo nos ayuda a definir que tabla, atributos se pueden llenar y que otros se deben mantener ocultos.
Los modelos usan convenciones para que a Laravel se le facilite el trabajo y nos ahorre tanto líneas de código como tiempo para relacionar más modelos, las cuales son:

- El nombre de los modelos se escribe en singular, en contraste con las tablas de la BD que se escriben en plural.

- Usan notacion UpperCamelCase para sus nombres.

Estas convenciones nos ayudan a detectar automaticamente las tablas, por ejemplo: el modelo User se encuentra en singular y con notacion UpperCamelCase y para Laravel poder definir que tabla es la que esta ligada a este modelo le es suficiente con realizar la conversion a notacion underscore y plural, dando como resultado la tabla: users.

Y esto aplica para cuando queremos crear nuestros modelos, si tenemos una tabla en la base de datos con la que queremos trabajar que se llama user_profiles, vemos que se encuentra con las convenciones para tablas de bases de datos (plural y underscore), entonces el modelo para esta tabla cambiando las convenciones seria: UserProfile (singular y UpperCamelCase).

Retomando el ejemplo que vimos en el Capítulo 6 sobre la migracion de pasteles, crearemos ahora un modelo para poder trabajar con esa tabla, el cual recibira el nombre de Pastel y el comando para poder crear nuestro modelos es:

php artisan make:model Pastel

Con esto se generara el archivo en donde ya se encuentra el modelo User en la carpeta app/ y dentro de el vamos a definir la tabla que se va a usar con esta linea:

protected $table = 'pasteles';

Bien una vez creado nuestro modelo pasaremos a crear una ruta de tipo get en nuestro archivo routes.php, posteriormente estudiaremos el enrutamiento básico en Laravel en el Capítulo 9, por el momento solo seguiremos el ejemplo, que quedaria de la siguiente forma:

Route::get('pruebasPastel', function(){

});

Dentro de esta ruta de prueba vamos a usar nuestro modelo, pero como estamos usando la especificacion PSR-4 debemos incluir el namespace del modelo al inicio del archivo, que seria igual a esto

use Curso\Pastel;

Con esto estamos diciendo que incluya la clase Pastel que es nuestro modelo, y con esto podemos ya hacer consultas a nuestra BD y mapear a objetos PHP. En la documentacion oficial de Laravel podemos ver todas las opciones que nos permite Eloquent, unas de las instrucciones basicas de este son get() que nos regresa todos los registros de la BD y first() que nos regresa el primer registro de una seleccion.

Además con Eloquent tambien podemos insertar, actualizar o eliminar registros, por ejemplo:

Para insertar la sintaxis seria la siguiente:
$pastel = new Pastel;
$pastel->nombre = 'Pastel Richos Style';
$pastel->sabor = 'chessecake';
$pastel->save();

Para actualizar seria la siguiente:
$pastel = Pastel::find(51);
$pastel->sabor = 'chocolate';

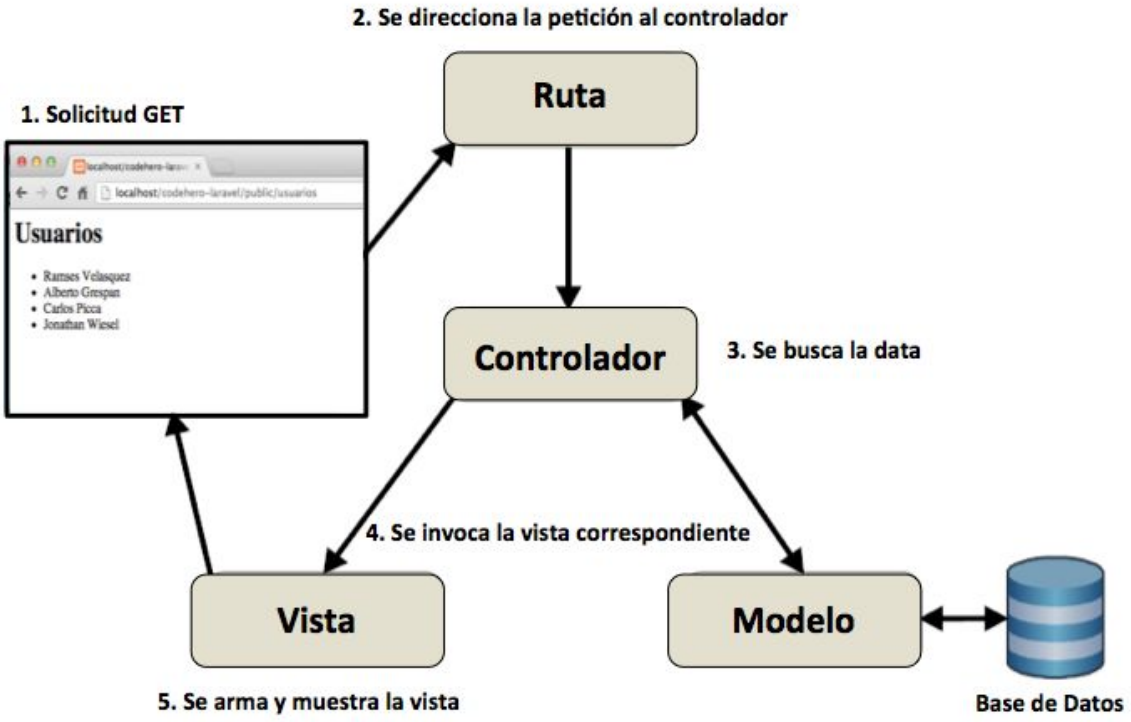
$pastel->save();

Para eliminar seria la siguiente:
$pastel = Pastel::find(51);
$pastel->delete();

o bien podriamos destruir el registro directamente con el modelo si tenemos su ID:
Pastel::destroy(51);

- ❏ Enrutamiento básico:

La siguiente imágen muestra el proceso que se realiza cuando ingresamos a una URL. Además muestra la arquitectura del patrón MVC que utiliza laravel para el desarrollo de proyectos.



Cuando ingresamos a una url directamente desde el navegador lo hacemos mediante una petición http de tipo GET, esta solicitud se envía al archivo routes.php ubicado dentro de app/Http/routes.php, en caso de no existir nos dará un error, si la ruta existe, nos llevará a un controlador en el cuál se encuentra la lógica , el controlador interaccionará con un modelo (opcionalmente) para recuperar información de una base de datos. Esta información llega al controlador y desde el controlador invocamos una vista, las vistas se encuentran en el directorio resources/views, finalmente la vista se carga y se muestra en el navegador.

Así es como funciona el modelo MVC (Model-View-Controller).

Supongamos que queremos ingresar a la siguiente URL http:/dominio.com/saludo y desplegar una página con el mensaje “Bienvenido :)”. En laravel la porción /saludo pertenecería a una ruta que regresa una respuesta o una vista dependiendo lo complejo que llegue a ser lo que queramos mostrar. La parte de dominio.com pertenecería a localhost si lo andamos probando de manera local.

Las rutas están siempre declaradas usando la clase Route . Eso es lo que tenemos al principio, antes de :: . La parte get es el método que usamos para ‘capturar’ las peticiones que son realizadas usando el verbo ‘GET’ de HTTP hacia una URL concreta.

Todas las peticiones realizadas por un navegador web contienen un verbo. La mayoría de las veces, el verbo será GET , que es usado para solicitar una página web. Se envía una petición GET cada vez que escribes una nueva dirección web en tu navegador.

Aunque no es la única petición. También está POST , que es usada para hacer una petición y ofrecer algunos datos. Normalmente se usa para enviar un formulario en la que se necesita enviar los datos sin mostrarlo en la URL.

Cualquier método de la clase Route recibe siempre dos argumentos, el primero es la URI con la que queremos hacer coincidir la URL y el segundo es la función a realizar que en este caso es un Clousure que no es otra cosa que una función anonima, es decir, que no tiene un nombre.

- ❏ Vistas

Las vistas en Laravel son la parte pública que el usuario de nuestro sistema va a poder ver, se escriben en HTML junto con un motor de plantillas llamado Blade. Las vistas se encuentran ubicadas en la carpeta resources/views/ y Laravel por defecto trabaja con la idea de que tenemos que escribir la menor cantidad de código repetido, modularizar nuestro código en donde mas se pueda, y si esto lo aplicamos en nuestros modelos, controladores y demás partes de nuestro proyecto, entonces, ¿Por que no hacerlo tambien en las vistas?.
Laravel usa unos archivos que se llaman plantillas o templates que suelen ser nuestros archivos principales, que tienen los segmentos de código que se repiten en más de una vista, como por ejemplo la barra de navegación, un menú de opciones, la estructura del acomodo de nuestro proyecto, etc. y como deben de estar prácticamente presentes en todos lados, no tiene sentido estar repitiendo los en todas las vistas. Por defecto Laravel contiene un template llamado app.blade.php, usualmente los templates contienen el head del HTML, las ligas del CSS del sistema y una seccion exclusiva para los archivos Javascript.

Además de los templates, se cuentan con archivos que se llaman partials, estos archivos son pequeños segmentos de código que suelen ser usados comúnmente en

partes del sistema en específico, como los formularios o secciones de mensajes, estos archivos surgen por el código que es más pequeño que repetimos mucho pero no es lo suficientemente grande como para considerarlo un template.

Esto hace que las vistas de cada parte del proyecto, que suelen ser llamadas por una ruta o controlador sean mucho más pequeñas que usando otro tipo de frameworks para desarrollo Web, y para poder unir todos estos archivos o piezas del rompecabezas usamos el motor de plantillas de Laravel llamado BLADE.

- ❏ Controladores:

En lugar de definir en su totalidad la lógica de las peticiones en el archivo routes.php, es posible que desee organizar este comportamiento usando clases tipo Controller. Los Controladores puede agrupar las peticiones HTTP relacionada con la manipulación lógica en una clase. Los Controladores normalmente se almacenan en el directorio de aplicación app/Http/Controllers/.
Un controller usualmente trabaja con las peticiones:

GET.
POST.
PUT.
DELETE.
PATCH.

Asociando los métodos de la siguiente forma:

GET: index, create, show, edit.
POST: store.
PUT: update.
DELETE: destroy.
PATCH: update.

Los controladores nos ayudan a agrupar estas peticiones en una clase que se liga a las rutas, en el archivo app/Http/routes.php

Route::resource('pasteles', 'PastelesController');

Esta ruta nos creara un grupo de rutas de recursos con las peticiones que estas mencionadas arriba: index, create, show, edit, store, update, destroy. Estas son las operaciones mas usadas en una clase y para no tener que crear una ruta para cada método es que Laravel agrupa todo esto con una ruta de tipo resource que se liga a un controlador.

Estos métodos significan:

- index: Es el método inicial de las rutas resource, usualmente lo usamos para mostrar una vista como página principal que puede contener un catalogo o resumen de la información del modelo al cual pertenece o bien no mostrar información y solo tener la función de página de inicio.

- create: Este método lo podemos usar para direccionar el sistema a la vista donde se van a recolectar los datos(probablemente con un formulario) para después almacenarlos en un registro nuevo, usualmente redirige al index.

- show: Aquí podemos hacer una consulta de un elemento de la base de datos o de todos los elementos o registros por medio del modelo para realizar una descripción.

- edit: Este método es similar al de create porque lo podemos usar para mostrar una vista que recolecta los datos pero a diferencia de create es con el fin de actualizar un registro.

- store: Aqui es donde se actualiza un registro en especifico que proviene del método create y normalmente redirige al index.

- update: Al igual que el store, solo que en vez de provenir de create proviene de edit y en vez de crear un nuevo registro, busca un existente y lo modifica, tambien suele redirigir al index.

- destroy: En este método usualmente se destruye o elimina un registro y la petición puede provenir de donde sea siempre y cuando sea llamado con el método DELETE, después puede redirigir al index o a otro sitio dependiendo si logro eliminar o no.

❏ Validaciones en Laravel:

Existen varias formas de validar nuestra aplicación para cubrir aspectos de seguridad como SQL Injection, ataques XSS o CSRF, algunas de ellas son:

- Validación de lado del cliente (Javascript y etiquetas HTML).
- Validación a nivel de base de datos (Migraciones y modelos).
- Validación de formularios (Request).

❏ Middlewares:

Nos permiten proteger las rutas de accesos no autorizados, como su nombre lo indica (middle) se ubica en el medio de la petición (Request), entonces si deseamos agregar un nuevo nivel de seguridad a nuestro sistema los Middleware son la respuesta.

Primero vamos a analizar un Middleware para la autenticación o logueo de los usuarios en nuestras rutas. Por defecto en nuestro proyecto de Laravel debemos de contar con un middleware llamado auth, este middleware de lo que se encarga es de

ver que el usuario se encuentre con una sesion activa, recuerden que en Laravel ya tenemos por defecto el manejo de sesiones junto con las tablas de la base de datos. Para decirle a nuestro proyecto que las rutas de nuestro controlador de pasteles van a estar protegidas por el middleware auth usamos el método middleware('name'); dentro del constructor de nuestra clase.

3.3.7. SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

Ver en detalle cuales son los beneficios de Scrum, sus fundamentos y sus requisitos.

El proceso
En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste (que el equipo estima considerando la Definición de Hecho) y quedan repartidos en iteraciones y entregas.
Las actividades que se llevan a cabo en Scrum son las siguientes:

Planificación de la iteración
El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

- Selección de requisitos (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

- Planificación de la iteración (4 horas máximo). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas.

Ejecución de la iteración
Cada día el equipo realiza una reunión de sincronización (15 minutos máximo), normalmente delante de un tablero físico o pizarra (Scrum Taskboard). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:

- ¿Qué he hecho desde la última reunión de sincronización?
- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimentos tengo o voy a tener?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad.

- Elimina los obstáculos que el equipo no puede resolver por sí mismo.
- Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Durante la iteración, el cliente junto con el equipo refinan la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian o replanifican los objetivos del proyecto para maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

Inspección y adaptación
El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

- Demostración (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.

- Retrospectiva (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados.

4. TERCERA PARTE: MARCO TEÓRICO

4.1. Introducción

De acuerdo a lo establecido en la teoría del punto anterior se llegó a comprender la problemática que hoy en día se genera en los envíos de paquetería realizado por la empresa que lo provee, o algún correo en particular. De ésta manera, fue determinada la logística, un punto clave en el comercio electrónico, y en la necesidad de las personas que adquieren productos.
A su vez, se ha abierto y detectado una necesidad la cual incluye a los usuarios que desean llevar por sus medios de transporte propio, diferentes paquetes que la gente necesite enviar, ya sea que lo ha comprado en una tienda e-commerce o desea enviar algo que ya tiene un su casa y precisa que llegue a otro punto de la ciudad o de otra. Dicho ésto, el problema de logística sigue retumbando, a lo que se suma otro, que es el entorno social y medio ambiente, y a su vez el aumento de desconfianza en los procesos logísticos.

De ésta manera, uniendo las herramientas ya planteadas en la teoría de los puntos anteriores, se constituye la posibilidad de contar con una plataforma y aplicación, que resuelva estos problemas nombrados. En ésta sección, se buscará cumplir con los siguientes objetivos específicos:

- Definir todos los requerimientos que hacen surgir a EnvioEntregas, los cuales van a ser de gran utilidad para demostrar en detalle el lo que va a cubrir la plataforma y la App en base a la solución de la problemática y reingeniería en los procesos logísticos.

- Análisis y Diseño del sistema completo, considerando las interfaces del mismo, las tecnologías aplicadas, los casos de uso que se plantean, logrando el modelado del completo sistema.

Entonces, se desarrolla una teoría la cual brinda y explica la solución planteada tanto en aspecto lógico como desde aspecto físico, haciendo énfasis en el marco teórico aportado obtenido de la formación académica; con el fin de obtener un producto final que contenga las características que se han planteado y requerido, integrando nuevos conceptos y mostrando la responsabilidad y potencialidad que tiene la capacidad de interrelación de la visión sistémica.

4.2. Planificación

Se realiza la planificación del proyecto en base a todos los recursos que son necesarios para el cumplimiento del mismo. Por más que somos dos autores en el mismo, las recursos necesarios son más que dos, por lo que nuestros nombres se van a repetir.
Se realizará una división de tareas, las cuales van a ser asignadas a los recursos disponibles en el proyecto. A su vez, tendrán un tiempo determinado para cumplirse en cierta etapa en el transcurso del proyecto
Para las actividades que se detallan posteriormente, como se indicó más arriba, van a ser asignadas a los recursos del proyecto que son: Jefe de Proyecto, Analista, Diseñador, Desarrolladores, Ingenieros QA.

Etapa	Nombre	Fecha de inicio	Fecha de fin	Duración
Introducción	Declaración de objetos de estudio y campos de acción	03/01/18	03/01/18	1
	Definición de la Situación Problemática	04/01/18	04/01/18	1
	Estudio e investigación de Antecedentes	04/01/18	04/01/18	1
	Definición de objetivos	05/01/18	05/01/18	1
	Definición de Alcances y Límites	08/01/18	09/01/18	2
	Definición de Idea a defender	10/01/18	11/01/18	2
	Desarrollo de aporte Teórico y Práctico	12/01/18	15/01/18	2
	Análisis de factibilidad	16/01/18	16/01/18	1
Marco contextual	Descripción del entorno de objeto de estudio	17/01/18	17/01/18	1
	Análisis de Problemas observados	18/01/18	19/01/18	2
	Análisis de antecedentes de proyectos similares	22/01/18	24/01/18	3
Marco Teórico	Investigación de Marco Teórico de Objeto de estudio	25/01/18	21/03/18	40
	Investigacion de Marco Teórico en campos de acción	23/03/18	07/06/18	55
	Análisis y Desarrollo de Diagnóstico	08/06/18	28/06/18	15
Modelo Teórico	Planificación	29/06/18	05/07/18	5
	Definición de Actores de Negocio	06/07/18	06/07/18	1
	Definición de Actores de Sistema	09/07/18	09/07/18	1
	Aplicación de Tecnologías de Campo de Acción	10/07/18	18/07/18	7
	Identificación de Requerimientos	18/07/18	26/07/18	7
	Listado de Casos de Uso del Sistema	27/07/18	30/07/18	2
	Realización de Diagramas de Casos de Uso	31/07/18	06/08/18	5
	Desarrollo de Casos de uso de sistema	07/08/18	10/08/18	4
	Arquitectura del Software	13/08/18	15/08/18	3
	Desarrollo del Modelo de Datos	16/08/18	22/08/18	5
	Diseños de Interfaz	23/08/18	11/09/18	14
	Diagramas de Secuencia y Clases de Diseño	11/09/18	24/09/18	10
Concreción del Modelo	Generacion de Interfaces Web	25/09/18	05/10/18	9
	Codificación de Casos de Uso	08/10/18	12/10/18	5
Revisión	Revisión del Sistema	15/10/18	23/10/18	7
	Implementación de Puesta en Marcha	24/10/18	26/10/18	3
	Documentaciones	29/10/18	31/10/18	3

Tabla 1: Listado de Etapas, Actividades, y Duración (en días) del proyecto.

4.2.1. Diagrama de Recursos

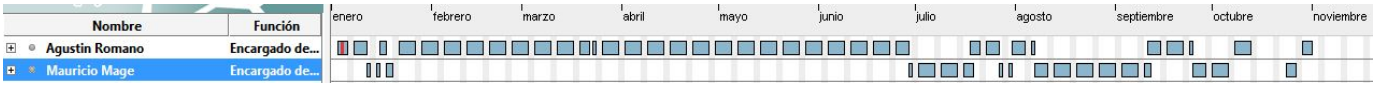


Figura 1.1 Diagrama de Recursos con sus tareas asignadas.

Con respecto a la asignación de recursos, se tuvo en cuenta que los únicos recursos del proyecto se deben encargar de completar todas las actividades que se programaron en determinadas etapas y con cierta duración.
Durante la planificación de las tareas en cuanto a los días y duración se tuvo en cuenta que el proyecto se comienza a principios de año, como fecha deseo de terminar fines de octubre, un total aproximado a 10 meses de duración.

En todas las etapas, se tienen en cuenta como días laborables sólo los días de semana (lunes a viernes) sin tener en cuenta feriados y tomándose días libres sólo los fines de semana.

4.2.2. Diagrama de Gantt

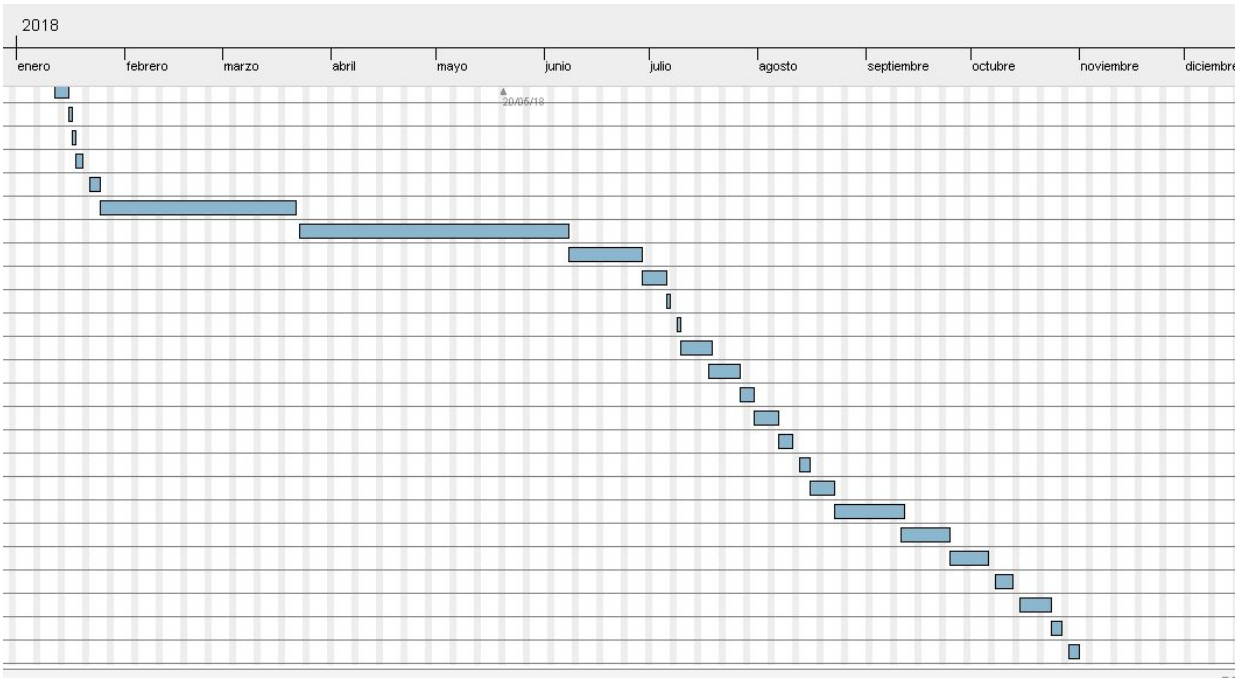


Figura 1.2 Diagrama de Gantt con todas las actividades programadas.

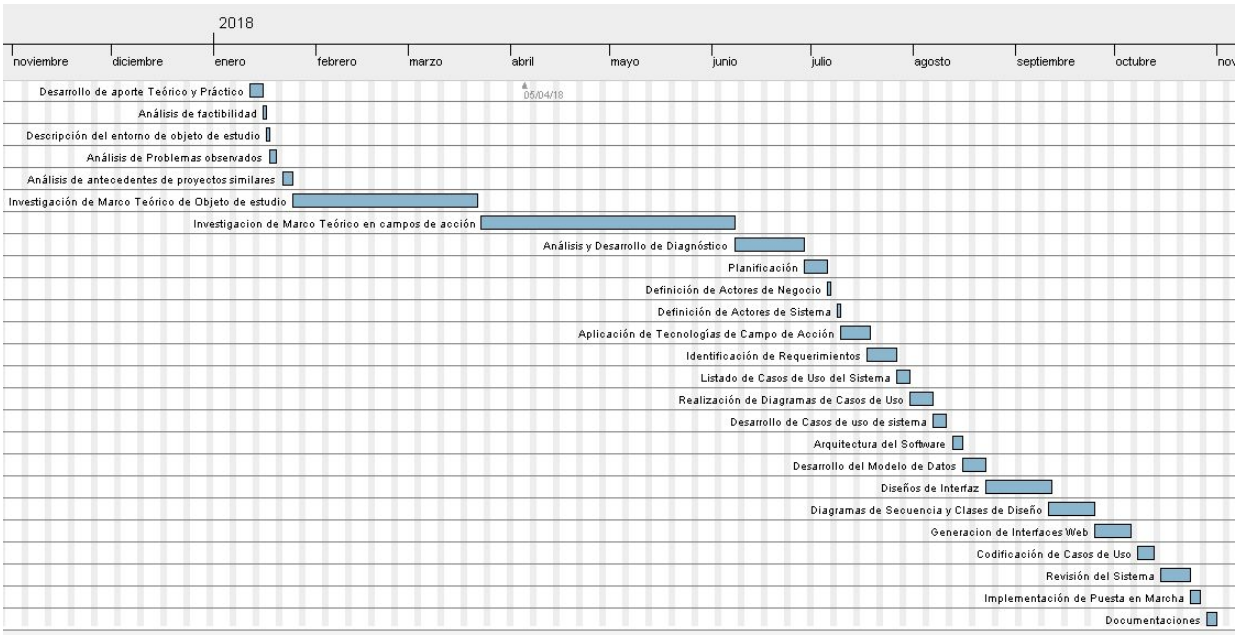


Figura 1.3 Diagrama de Gantt con todas las actividades programadas y etiquetadas.

4.3. Requerimientos

4.3.1. Rangos de Calidad

Entre los criterios de calidad a considerar se tienen en cuenta los siguientes:

- Accesibilidad para lo usuarios, deben tener la facilidad de acceder al sistema e información con suma claridad.
- Interfaz amigable: con facilidad de operación para los usuarios. Agradable y minimalista en su diseño.
- Seguridad y confidencialidad en los datos de los usuarios. Buena gestión de los mismos.
- Evolutivo: flexibilidad con respecto a las diferentes actualizaciones que pueden ocurrir en la plataforma.
- Seguridad: posibilidad de asegurar los datos de los usuarios y sus movimientos, poder reestablecer contraseñas importantes.
- Còdigo fuente reutilizable, tener la capacidad de desarrollar algoritmos utilizados globalmente por diferentes entes en el sistema.
- Disponibilidad de información en cualquier momento para usuarios que lo soliciten.

4.3.2. Estructura Organizacional

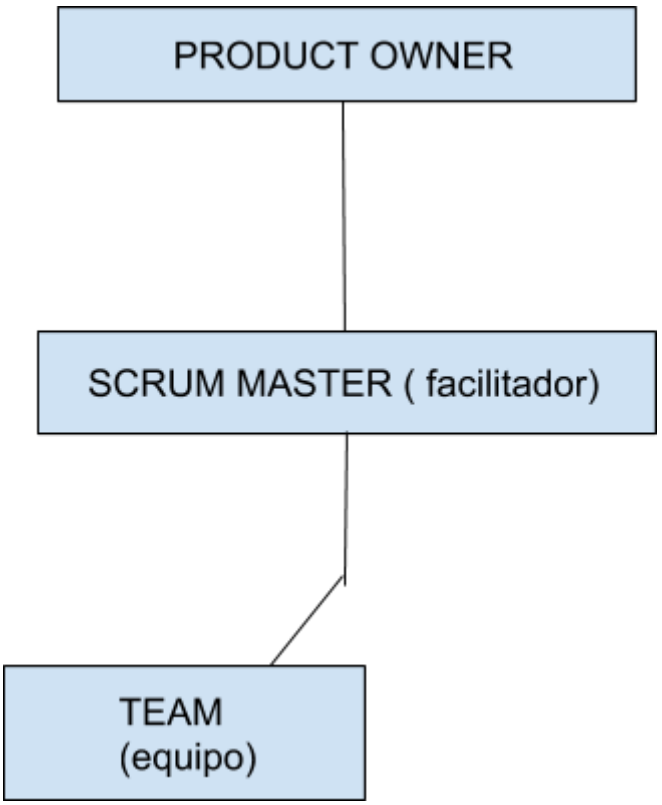
Como se indicó anteriormente, la metodología ágil de desarrollo seleccionada, es SCRUM. Por lo tanto, las personas que conforman el equipo de trabajo del proyecto, se clasifican en:

- Product Owner: Se asegura que el equipo de Scrum trabaje coordinadamente y de manera adecuada, bajo las condiciones y perspectiva del negocio. El Product Owner ayuda al usuario a escribir las historias de usuario, las prioriza, y las coloca en el Product Backlog.

- Scrum Master (o el facilitador): su trabajo es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo, sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido.Él es quien hace que las reglas se cumplan.

- Equipo de Desarrollo: El equipo tiene la responsabilidad de entregar el producto. Es recomendable un pequeño equipo de 5 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

En éste proyecto, teniendo en cuenta que se trata de un proyecto específico, con un objetivo concreto como la culminación del Proyecto Final, los diferentes roles serán asumidos por los dos integrantes, quienes se encuentran comprometidos al cien por ciento con el proyecto. De ésta forma, los *roles* se estructuran de la siguiente manera:



4.3.3. Actores de Negocio

Luego del despliegue de los puntos anteriores, se considera a trabajar y desarrollar el Negocio, que implica todo el proceso que conlleva a la solución de la problemática planteada en el tema.

Considerando todo ésto, más los conceptos y aprendizajes adquiridos durante la formación profesional de ambos, se prosigue a indicar cuales son esos procesos de negocios para luego identificar los agentes involucrados a éstos. Cada uno de estos agentes, tienen asignados a su vez, diferentes roles o actividades que llevan a cabo a partir del caso de uso de negocio, y también lo relacionan con otros procesos o actividades de otros paralelos.

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

Para cada actor del negocio que se identifica se debe escribir una breve descripción que incluya sus responsabilidades y por qué interactúa con el negocio.
Los actores del negocio interactúan con el negocio enviando y recibiendo mensajes, y para conocer el papel del actor se debe precisar en qué procesos se involucra el actor. Esto se muestra por la llamada asociación de comunicación entre el actor del negocio y el caso de uso del negocio que representa al proceso

Para identificarlos correctamente, se deben tener en cuenta cuales son esos procesos de negocios o actividades y como ellos se relacionan.

- Usuarios (tanto el Usuario como el Delivery)
- Sistema
- Base de Datos

En primer lugar, el **Usuario**, es quien cumple el rol principal en el Negocio, ya que es quien demanda los Pedidos para que sean procesados como un envío, o bien cumple el rol de Delivery y busca pedidos para enviar. A su vez, desde el punto de vista del Negocio, el Usuario abarca la mayoría de los procesos definidos y planificados de la plataforma, ya que interactúa directamente con la plataforma y también influye en los demás actores.
Por lo tanto, conserva la mayoría de los Casos de Uso.

Por otro lado, se tiene en cuenta al **sistema** como otro actor principal, no es el sistema en general como si fuese la plataforma quien actúa, nada de eso, sino vendría a ser el eje o actor abstracto el cual se encarga de roles como los de ejecutar ciertos Jobs que forman parte de procesos de Negocio como notificar vía mails, o notificaciones push, además contiene toda la lógica de Negocios propiamente dicha para el completo funcionamiento de la plataforma.

Por último tenemos a la **Base de Datos**, que es encargada de administrar toda la información que se maneja en el sistema, ya sea de Usuarios, Pedidos u otras entidades, es por eso que forma parte del Negocio, ya que contiene todas las

entidades definidas por sus modelos del proyecto. Contempla también en parte la mayoría de los Casos de Uso, ya que se relaciona con casi todos los métodos de los controladores, por el hecho de que son acciones de *requests* o *responses* que llegan a la Base.

4.3.4. Actores de Sistema

Como sabemos, un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. A estos actores se los denomina *actores de sistema.*

Considerando la lógica y el funcionamiento del so procesos de sistema de la plataforma, se debe tener en cuenta quienes son los actores que determinan y realizan las funciones o actividades, es decir, quienes actúan de manera directa e indirecta sobre los Casos de Usos planteados anteriormente.

Es por eso, que bajo la contemplación de los mismos, y lo adquirido profesionalmente en el proyecto y materia, se definen los siguientes actores de sistema:

- Usuario (o Delivery)
- Administrador
- Notificación
- MercadoPago

Comenzando con el **Usuario** o **Delivery**, ya que éste actor tiene dos Roles diferenciados, con respecto a su función principal en la plataforma, por un lado, el Usuario es quien interactúa en procesos y Casos de Uso como Nuevo Pedido, es decir, es quien solicita el envìo de un Pedido, mientras que el Delivery, que no deja de ser un Usuario, es decir, la entidad y Modelo es el mismo, es quien lleva ese Pedido,

y tiene roles correspondientes por ejemplo al Caso de Uso Nueva Oferta , es decir, quien envìa una oferta al Usuario para poder llevar ese Pedido.

Es por eso, que éste actor, tiene los roles fundamentales, con respecto al proceso del Sistema, es por eso, que se indica primero y se lo desarrolla como tal.

El Usuario es el encargado de realizar todo tipo de acciones en el sistema, desde su perfil, hasta interactuar con otro usuario, por ejemplo a partir de mensajes.

En segundo lugar, ocupa su lugar el **Administrador**, es quien gestiona todas las actividades con respecto a control y auditoría del sistema. Además, posee el control sobre la base de datos, y a su vez, respecto a parametrizaciones que tienen que ver con porcentaje de ganancias, o impuestos o también temas acerca del seguro de los Envíos.

Cumple un rol fundamental, ya que gracias a él, se desarrollan con seguridad todos los procesos del proyecto.

Continuamos con el tercer Actor seleccionado, se trata de la **Notificación**, actuando de manera pasiva en el sistema. El mismo, cumple la función al momento de recibir peticiones acerca de un nuevo pedido, nueva oferta, mensajes, cancelaciones, y demás con respecto a las intervenciones en un Pedido a través de otros actores ya nombrados que son los Usuarios y Deliveries.

Es por eso, que éste actor, tiene un punto fundamental que es el de notificar a todos los agentes sobre la petición propuesta, se maneja en un entorno abstracto del sistema, complementando cada acción y caso de uso resuelto por otros actores.

El último actor que se identifica, es **MercadoPago**, es otro actor pasivo, que recibe peticiones o *request* desde la plataforma, para devolver la validación y formas de pago de un Pedido que se solicitó.

Por lo tanto también es un actor abstracto, formando parte de casos de uso de sistema como el de Validar Pago, contemplando todas los requisitos del mismo.

4.3.5. Aplicación de las tecnologías del Campo de Acción

4.3.5.1. Diseño Responsivo

4.3.5.2. Geolocalización

4.3.5.3. Notificaciones Push/Email

4.3.5.4. Servidor Express aplicando NodeJS

4.3.5.5. Aplicaciones Móviles en base a Ionic

4.3.5.6. Importancia de los Frameworks - PHP Laravel y Blade

4.3.5.7. SCRUM

4.3.6. Requerimientos del Sistema

4.4. Análisis y Diseño

4.4.1. Análisis del Sistema

4.4.1.1. Listado de Casos de Uso del Sistema

En base al análisis de las necesidades y requerimientos del proyecto, se listan los siguientes Casos de Uso:

1. Registro de Usuario.
2. Ingresar al sistema.
3. Editar datos de Usuario.
4. Recuperar y renovar contraseña.
5. Validar email.
6. Validar celular.
7. Guardar token del dispositivo.
8. Generar configuración de Notificaciones.
9. Ingresar con Facebook
10. Validar cuenta Facebook

11. Generar sesión de Usuario
12. Ingresar a cargar un Pedido
13. Sacar foto del Pedido
14. Subir foto del Pedido
15. Ingresar datos del Pedido
16. Visualizar direcciones en el Mapa
17. Validar datos del Pedido
18. Calcular Precio Estimado del Pedido
19. Guardar Pedido.
20. Notificar Pedido Nuevo
21. Cancelar Pedido
22. Notificar Cancelar Pedido
23. Enviar Oferta.
24. Responder Oferta con un Mensaje
25. Rechazar Oferta.
26. Aceptar Oferta.
27. Generar Pago de Envío.
28. Pagar Envío.
29. Validar Datos de Pago.
30. Enviar Mensajes de Consulta.
31. Notificar Mensaje Nuevo
32. Generar Código Inicial de Envío
33. Generar Código Final de Envío.
34. Notificar Códigos.
35. Notificar Oferta Aceptada.
36. Ingresar Código Inicial de Envío.
37. Validar Código Inicial.
38. Indicar Pedido En Proceso
39. Notificar Pedido En Proceso
40. Enviar Ubicación en camino.
41. Notificar Ubicación de Envío
42. Ingresar Código Final de Envío.
43. Indicar Pedido Completado.

44. Notificar Pedido Completado
45. Calificar al Usuario.
46. Notificar Calificación.
47. Registro de Actividades de Usuario.
48. Registro de Calificaciones.
49. Registro de Pagos de Usuario.
50. Calcular Reputación de Usuario.
51. Mostrar Pedidos en Mapa.
52. Mostrar Pedidos
53. Mostrar Mensajes.
54. Editar Datos de Usuario.
55. Mostrar Datos de Usuario.
56. Editar Configuración de Notificaciones.
57. Cuantificar Pedidos en sus Estados.
58. Obtener Ubicación Actual de Usuario.
59. Filtrar Pedidos por Ubicación Actual.
60. Visualizar Ruta de Origen a Destino del Pedido.

4.4.1.2. Descripción de Casos de Uso del Sistema

Caso de Uso Nº 1	**Registro de Usuario**
Actores	Usuario
Tipo	Primario
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 2	**Ingresar al Sistema**
Actores	Usuario
Tipo	Primario
Descripción	Permite el acceso al Sistema mediante el ingreso del Mail y Contraseña

Caso de Uso Nº 3	**Editar Datos de Usuario**
Actores	Usuario
Tipo	Opcional
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 4	**Recuperar y renovar contraseña**
Actores	Usuario
Tipo	Primario
Descripción	Permite a Usuario renovar su contraseña a través de un link que se le envía a su email, ingresado por él mismo.

Caso de Uso Nº 5	**Validar Email**
Actores	Usuario
Tipo	Primario
Descripción	Una vez que el Usuario se Registra, el mismo debe validar su mail de usuario, desde un link que le llega a su email.

Caso de Uso Nº 6	**Validar Celular**
Actores	Usuario
Tipo	Primario
Descripción	El Usuario recibe un Código de Validación a su celular, y debe ingresar el mismo en el sistema para validarlo.

Caso de Uso Nº 7	**Guardar token del dispositivo**
Actores	Sistema

Tipo	Primario
Descripción	Al registrarse, la plataforma guarda el token del dispositivo para luego poder enviar notificaciones al mismo.

Caso de Uso Nº 8	**Generar configuración de Notificaciones**
Actores	Sistema
Tipo	Primario
Descripción	Una vez que se registra le Usuario, el sistema genera la configuración por defecto de Notificaciones para el Usuario.

Caso de Uso Nº 9	**Ingresar con Facebook**
Actores	Usuario
Tipo	Secundario
Descripción	El Usuario puede Iniciar en la plataforma con Facebook, conectándose con su cuenta personal a la plataforma.

Caso de Uso Nº 10	**Validar cuenta Facebook**
Actores	Sistema, API Facebook
Tipo	Primario
Descripción	Se valida la cuenta del usuario de Facebook con la API Facebook, y se guardan los datos de la cuenta en la base de datos del Sistema.

Caso de Uso Nº 11	**Generar sesión de Usuario**
Actores	Sistema
Tipo	Primario
Descripción	El sistema genera una sesión una vez que el Usuario ingresa, ésto permite que el mismo

	pueda interactuar con la plataforma vinculando sus datos anteriores o guardandolos para un posible ingreso nuevamente.

Caso de Uso Nº 12	**Ingresar a cargar un Pedido**
Actores	Usuario
Tipo	Opcional
Descripción	Permite ingresar al Usuario a la sección de carga de un Pedido, indicando los datos a completar para ello.

Caso de Uso Nº 13	**Sacar foto del Pedido**
Actores	Usuario
Tipo	Opcional
Descripción	Puede optar el Usuario por sacar una foto del Pedido, permite el uso de la cámara del celular para ello.

Caso de Uso Nº 14	**Subir Foto del Pedido**
Actores	Usuario
Tipo	Opcional
Descripción	Puede optar el Usuario por subir una foto del Pedido, permite el ingreso a sus documentos en la PC..

Caso de Uso Nº 15	**Ingresar datos del Pedido**
Actores	Usuario
Tipo	Primario
Descripción	Permite el ingreso de todos los datos necesarios del Pedido

Caso de Uso Nº 16	**Visualizar direcciones en el Mapa**
Actores	Sistema, API Google Maps.
Tipo	Primario
Descripción	Permite al Usuario indicar y/o visualizar las ubicaciones de Origen y Destino en el Mapa

Caso de Uso Nº 17	**Validar datos del Pedido**
Actores	Sistema
Tipo	Primario
Descripción	El Sistema valida los datos cargados del Pedido para que se cargue correctamente el mismo.

Caso de Uso Nº 18	**Calcular Precio Estimado del Pedido**
Actores	Sistema, API MercadoPago
Tipo	Primario
Descripción	En base al volumen y ciudad del Pedido, el sistema calcula el Precio Estimado del Pedido a través de la API de Mercado Pago

Caso de Uso Nº 19	**Guardar Pedido**
Actores	Sistema
Tipo	Primario
Descripción	El sistema guarda el Pedido en la Base de Datos, para su registro correcto y para luego poder notificar a demás usuarios que hay un nuevo Pedido para hacer.

Caso de Uso Nº 20	**Notificar Pedido Nuevo**
Actores	Usuario, Sistema
Tipo	Primario

Descripción	El sistema revisa la tabla necesaria para notificar a los usuarios que pueden estar cerca de la ubicación del nuevo pedido, y se lo notifica vía email y notificación push al Usuario.

Caso de Uso Nº 21	**Cancelar Pedido**
Actores	Usuario
Tipo	Opcional
Descripción	Permite la cancelación del Pedido, validando el día en que se realiza sea anterior al de su fecha de retiro.

Caso de Uso Nº 22	**Notificar Cancelar Pedido**
Actores	Sistema
Tipo	Primario
Descripción	Se notifica a los Usuarios que el pedido fue cancelado, el sistema envía un email a todos los usuarios que pueden estar interesados o ya involucrados con dicho Pedido.

Caso de Uso Nº 23	**Enviar Oferta**
Actores	Usuario
Tipo	Primario
Descripción	El Usuario que actúa como Delivery, envía una oferta al Solicitante, para realizar el Pedido.

Caso de Uso Nº 24	**Responder Oferta con un Mensaje**
Actores	Usuario
Tipo	Opcional
Descripción	El Usuario puede responder la oferta con un mensaje consultando o reclamando alguna cuestión al Delivery.

Caso de Uso Nº 25	**Rechazar Oferta**
Actores	Usuario
Tipo	Opcional
Descripción	Permite rechazar la oferta que el Delivery está proponiendo.

Caso de Uso Nº 26	**Aceptar Oferta**
Actores	Usuarios
Tipo	Primario
Descripción	Permite al Usuario aceptar la oferta que ha emitido el Delivery..

Caso de Uso Nº 27	**Generar Pago de Envío**
Actores	Sistema
Tipo	Primario
Descripción	El sistema genera un precio estimado del Pedido de acuerdo al origen y destino como así también su volumen y peso.

Caso de Uso Nº 28	**Pagar Envío**
Actores	Usuario, Sistema
Tipo	Primario
Descripción	El usuario abona el costo del pedido a través de un link generado por el sistema, para que acceda a Mercado Pago y realice la transacción.

Caso de Uso Nº 29	**Validar Datos de Pago**
Actores	Sistema
Tipo	Primario

Descripción	El Sistema valida los datos de la tarjeta para corroborar si son validos, para de ésta forma controlar el Pago.

Caso de Uso Nº 30	**Enviar Mensajes de Consulta**
Actores	Usuario
Tipo	Secundario
Descripción	Los usuarios pueden enviar mensajes de consulta al Delivery, o viceversa.

Caso de Uso Nº 31	**Notificar Mensaje Nuevo**
Actores	Sistema
Tipo	Primario
Descripción	El Sistema notifica al Usuario que ha recibido un nuevo mensaje a través del Email o Notificación Push, según esté configurado.

Caso de Uso Nº 32	**Generar Código Inicial de Envío**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 33	**Generar Código Final de Envío**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 34	**Notificar Códigos**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 35	**Notificar Oferta Aceptada**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 36	**Ingresar Código Inicial de Envío**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 37	**Validar Código Inicial**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 38	**Indicar Pedido En Proceso**
Actores	Usuario
Tipo	Imprescindible

Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 39	**Notificar Pedido En Proceso**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 40	**Enviar Ubicación en camino.**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 41	**Notificar Ubicación de Envío**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 42	**Ingresar Código Final de Envío**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 43	**Indicar Pedido Completado**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 44	**Notificar Pedido Completado**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 45	**Calificar al Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 46	**Notificar Calificación**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 47	**Registro de Actividades de Usuario**
Actores	Usuario
Tipo	Imprescindible

Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 48	**Registro de Calificaciones**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 49	**Registro de Pagos de Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 50	**Calcular Reputación de Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 51	**Mostrar Pedidos en Mapa**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 52	**Mostrar Pedidos**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 53	**Mostrar Mensajes**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 54	**Editar Datos de Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 55	**Mostrar Datos de Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 56	**Editar Configuración de Notificaciones**
Actores	Usuario
Tipo	Imprescindible

Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 57	**Cuantificar Pedidos en sus Estados**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 58	**Obtener Ubicación Actual de Usuario**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 59	**Filtrar Pedidos por Ubicación Actual**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

Caso de Uso Nº 60	**Visualizar Ruta de Origen a Destino del Pedido**
Actores	Usuario
Tipo	Imprescindible
Descripción	El Usuario puede Registrarse en la Plataforma completando todos los datos que se solicitan..

4.4.1.3. Diagrama de Casos de Uso del Sistema

4.4.2. Diseño del Sistema

4.4.2.1. Arquitectura Del Software

La arquitectura completa del software, está basada en un lenguaje Backend llamado PHP Laravel en su versión 5.4. El cuál es el encargado de contener la lógica de Negocios, interacción entre las vistas, manejo de rutas, seguridad, validaciones y por su puesto integraciòn con Bases de Datos.

Laravel proporciona un sistema MVC, Modelo Vista Controlador, en el cual, el proyecto presenta los diferentes Modelos a partir de cada entidad (tabla de base de datos) con sus campos que lo componen.
A su vez, se contiene diversos Controladores, quienes se encargan de interactuar con la base de datos, y devolver resultados a la Vista.

Es así, como todos las partes se relacionan, en primer lugar el Modelo, luego éste modelo tiene un controlador asociado, y a éste controlador retorna a una Vista. Ésta vista, son llamadas en cierto punto o ingresadas a través de una Ruta, la cual puede tener parámetros o no, dependiendo la función a ejecutar por el controlador.

De ésta forma, permite que el software sea escalable, de una simple manera, creando vistas y modelos con sus controladores correspondientes, dependiendo de lo que se necesite.

Además las vistas (que son la interfaz que ve el Usuario) están desarrolladas en HTML5 y Materialize (Material Design) y a su vez, se aplica lógica de negocios en parte a través de Blade, un lenguaje que permite interactuar con PHP en un vista.

Otra parte importante que tiene que ver con la seguridad de la plataforma, es el uso de Middlewares, que provee Laravel, los cuales son asignados a rutas, siempre y cuando se necesite, dependiendo de qué permisos poseen los usuarios para acceder.

Por ejemplo, si el usuario quiere acceder a *Sus Pedidos*, basta con solo indicar en la ruta que dirige a esa Vista, un middleware con un parámetro *Auth*, por lo tanto, si el Usuario está *deslogueado,* automáticamente se redirecciona a la pantalla de Login, y luego al acceder, vuelve a la vista anterior, que es *Mis Pedidos*.

Con respecto a la Base de Datos, es MySQL y se maneja a través de Laravel también, con un método interesante llamado *Migraciones*, las cuales cada una de ellas son archivos.php que contienen dos métodos *Up()* y *Down()* donde en el primero se indica la tabla que se creó con sus respectivas columnas, índices, claves primarias y demás, y el segundo posee un DropTable si se requiere. Por lo tanto, editando éstas migraciones, y creando nuevas se maneja directamente la estructura de tablas de la base de datos.

En base a las notificaciones vía Mail, se maneja con una arquitectura basada en *Laravel Notifications*, simplemente son archivos que están predefinidos para configurar y por medio de parámetros de conexión a servidores SMTP son enviados y generados los emails.

Por el lado de la aplicación móvil, se maneja una arquitectura basada en *Ionic2,* ya que permite el *Deploy* de las aplicaciones tanto para sistemas operativos iOS como Android.

Éste framework, está asociado con Cordova para poder deployar éstas apps.

Las interfaces o diferentes *layout*, están desarrollados en HTML5 en el frontend, con estilos .scss. A su vez, se utiliza AngularJS para la lógica impuesta en el cliente, y también cuenta con *TypeScript*, para la conexión a servicios que llamen a la API para su interacción la misma base de datos de la Web.

Esa API, está alojada en el servidor de la Web, la cual está desarrollada también en PHP LARAVEL, con otro sistema de rutas, pero utilizando los mismos *Controladores*, pero con métodos diferentes y nuevos.

De ésta forma, el objetivo de toda esta arquitectura, es que ambas plataformas tanto Web como Móvil, se mantienen homogéneas e interactúen entre sí, debido a que ambas dependen por la relación de sus Datos de todas las acciones e interacciones que pueda realizar un Usuario.

4.4.2.1.1. Vista de Casos de Uso

Permite la creación e implementación de la Base de Datos, y a su vez planificar el Desarrollo de la plataforma y la aplicación móvil. Contemplando cada Caso de Uso con respecto a la lógica que se debe desarrollar, teniendo en cuenta que va a ir creciendo a medida que sucedan interacciones.

4.4.2.1.2. Vista Lógica

Es la que provee el completo funcionamiento del sistema, contemplando cada actor y las entidades para con sus relaciones. De ésta forma, se puede desarrollar la funcionalidad que implica la solución impuesta anteriormente en el documento.

4.4.2.1.3. Vista de Procesos

Posee el entendimiento y proceso de comunicación e integración entre las partes del proyecto y del desarrollo del mismo. Permitiendo la observación de la concurrencia de todo el sistema.

4.4.2.1.4. Vista de Despliegue

Abarca la parte de Hardware, permitiendo ver la arquitectura física con respecto a Servidores en los que se hospeda la plataforma y aplicación con respecto a los requerimientos que se deben tener en cuenta para sus componentes.

4.4.2.1.5. Vista de Implementación

Es la que permite visualizar todo el trabajo del equipo que lleva a cabo las tareas ya sean de Código o físicas para la puesta en marcha del sistema.

4.4.2.1.6. Vista de Datos

Permite visualizar toda la data guardada y preservada en la Base de Datos MySQL nombrada anteriormente.

Se trabaja con ésta Base de Datos, por el hecho de que encaja perfectamente con su sistema Relacional entre tablas, debido a que fue pensada cada una de las entidades

en base a sus relaciones que pueden ocurrir. Además es realmente fácil obtener la información desde ésta Base y a su vez poder manipularla, ya sea editando o elimando la misma.

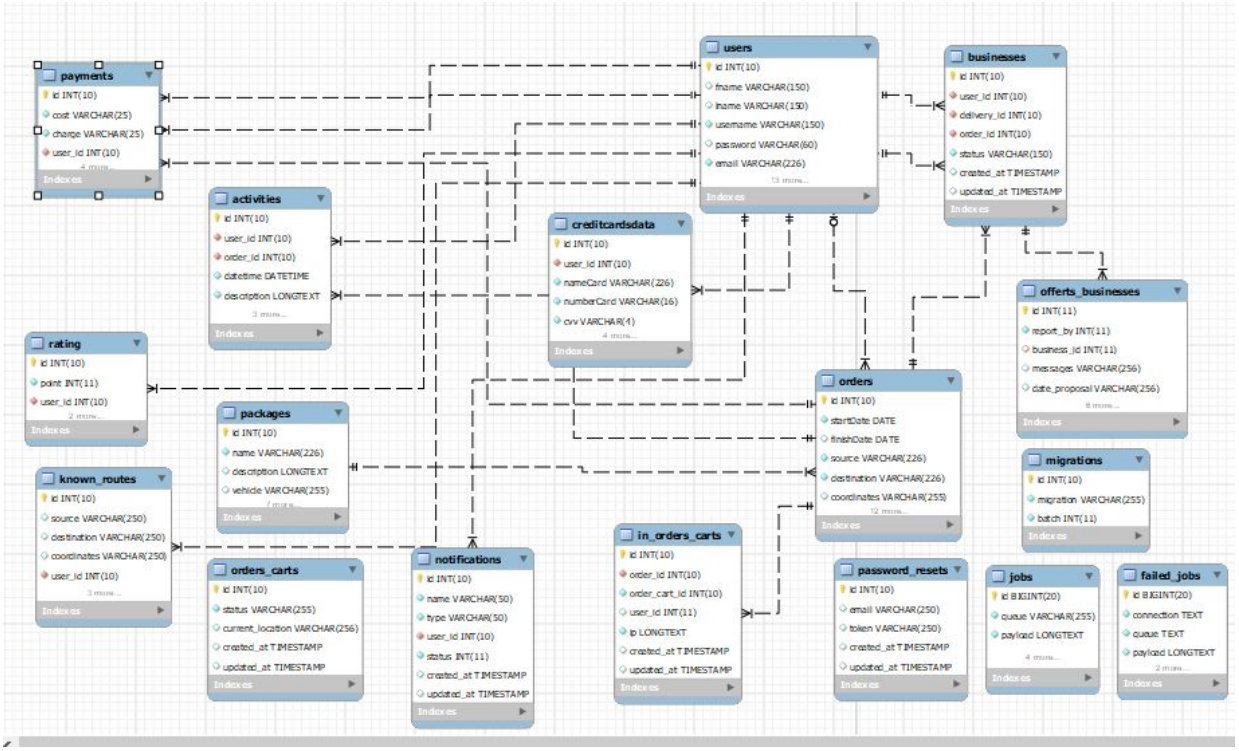
4.4.2.2. Tamaño y Performance

Es importante la implementación de buenos Servidores con características que incluyan todo lo necesario como para soportar una Base de Datos bastante cargada (*BigData*) y además la plataforma completa Web y Móvil, que soporte PHP Laravel, Ionic2, AngularJS, y sus componentes para que funcionen sus librerías de cada plataforma, como lo es *composer*, que permite la instalación y descarga de todas las librerías que el proyecto y Laravel impongan, y a su vez *npm*, para hacer lo mismo pero con la plataforma móvil y Ionic2.

La base de datos, está pensada para que soporte las relaciones y las completas interacciones de los Usuarios al mismo tiempo, ya que va a poseer gran concurrencia, es por eso que además se manejan *Jobs*, los cuales sirven para ejecutar acciones en otros hilos, que son programadas a través de CronTabs, los mismos hospedados en el Servidor, por lo cual también deben ser soportados por el mismo.

De ésta forma, el tamaño de la plataforma web y móvil es grande, y se pensó ésta arquitectura y metodologías para abarcar su completo funcionamiento, sin restricciones ni topes que perjudiquen a las interacciones de los usuarios en tiempo real, con o sin concurrencia.

4.4.2.3. Modelo de Datos



4.4.2.4. Resumen de Decisiones de Diseño

4.5. Conclusión

Como conclusión se observa, con respecto a los conceptos abarcados, se han logrado los objetivos mencionados al comienzo, con respecto a los requerimientos de la plataforma y proyecto Envioentregas.

Durante el proceso, se definieron e instauraron diferentes metodologías teóricas, obtenidas de fuentes relatadas en la materia y aprendidas como profesionales. Lo cual permite tener en cuenta la solución para aplicarla hoy en día en nuestra vida, además de resolver problemas coherentes y de problemática palpable como lo es la logística , permite visualizar los campos de acción marcados anteriormente en el proyecto.

A su vez, el proyecto permite una ampliación del modelo diseñado a otras realidades, dada la difusión de la problemática y la contextualización global que se vive hoy en la actualidad.

Se han aplicado conceptos y contenido tecnológico y científico, en el caso de investigación de tecnologías, ponderaciones, planificación de estructuras, métodos de desarrollo y correcciones.

Los diagramas presentados como referencias para la comprensión del diseño del sistema se entiende que permiten al lector tener una idea de las pretensiones y objetivos
perseguidos por el sistema que se busca implementar en las próximas etapas.

En cuanto a las herramientas y definiciones asumidas para unificar lo que parece en principio ser un conjunto de conceptos sueltos e inconexos en una herramienta concreta y unificada, donde todos los componentes tienen por objetivo contribuir a solucionar una problemática específica.

5. CUARTA PARTE: CONCRECIÓN DEL MODELO

5.1. Introducción

5.2 Implementación

5.2.1 Recursos de terceros utilizados en la implementación

5.2.1.1 JQuery

jQuery es una biblioteca multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargar, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery(). Es por eso que se

validó el uso de jQuery para la plataforma web debido a que en muchas oportunidades, refresco de estado de pantalla, o mostrar mensajes de error y ocultarlos, necesitamos el uso de jQuery para poder utilizar AJAX o simplemente aplicar condiciones de jQuery, para que funcione correctamente.

5.2.1.2 SDK API Facebook

El inicio de sesión con Facebook permite iniciar sesión en una aplicación o un sitio web de forma cómoda, rápida y segura.

Para implementar el inicio de sesión con Facebook con el SDK para JavaScript, necesitas un identificador de la aplicación de Facebook, que puedes crear y recuperar en el panel de aplicaciones. También es necesario algún sitio donde alojar archivos HTML.

En el proyecto, se utiliza tanto para la web como para la aplicación móvil, debido a que su uso es esencial para la rapidez del Login del Usuario en la plataforma, permitiendo a su vez, que gracias a Facebook Analytics se pueda observar y realizar un análisis sobre el uso o estadísticas sobre usuarios, navegando en la plataforma, cuanto uso tiene cada pantalla, el tiempo que el Usuario navega usualmente, de que país proviene, qué dispositivo o PC utiliza, sistema operativo y demás.

A su vez, gracias a lo explicado anteriormente, logramos identificar quizas problemas que sin el análisis de Facebook Analytics no podríamos realizar, es por eso la utilidad del Facebook Login API

5.2.1.3 API Google Maps

La implementación de la API Google Maps tanto en la plataforma web como móvil para la App, ofrece todo tipo de servicio de Geolocalización y Geográficos, que permiten al Usuario y al funcionamiento del sistema, trabajar con coordenadas para lograr orientar al usuario en donde está, buscar rutas para lograr realizar Pedidos cercanos o no, permite el cálculo matemático para buscar una mejor ruta de fácil

acceso y menor recorrido, da muchísimo potencial al proyecto, debido a que se puede visualizar un mapa, trabajar con GPS, hacer de lo que parece abstracto, sea real e ilustrativo para los Usuarios.

Además, incluye otras APIs como las siguientes:

- La API de Directions, que ofrece instrucciones para llegar en automóvil en 199 países, le permite ayudar a sus usuarios a encontrar la forma de llegar a su tienda, hotel y otros destinos.

- Obtenga acceso ilimitado

Permita que sus usuarios vean a dónde se dirigen, incluso antes de llegar a destino, gracias a las imágenes de gran precisión visual de Street View.

- La API Google Maps JavaScript, permite visualizar los mapas, interactuar con *Markers,* listado de direcciones del mundo para lograr encontrar la que el Usuario prefiera, y demás. ésta API permite que sea usada tanto en una web, como en Ionic, ya que está basada en JavaScript y puede ser utilizada fácilmente en ambas plataformas del Deploy (Android y iOS)

5.2.1.4 API Firebase

Firebase es un conjunto de herramientas orientadas a la creación de aplicaciones de alta calidad, al crecimiento de los usuarios y a ganar más dinero. Personalmente describo la plataforma como una suite de diferentes aplicaciones que nos harán más fácil el desarrollo de nuestra aplicación

Sus características fundamentales están divididas en varios grupos, las cuales podemos agrupar en:

- Analíticas: Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.

- Desarrollo: Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firebase, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.

- Crecimiento: Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.

- Monetización: Permite ganar dinero gracias a AdMob.

La API de Firebase además permite:

- Autenticación: Es un servicio que nos simplifica el inicio de sesión y la gestión de la misma en nuestra aplicación. Si la usamos en aplicaciones web es extremadamente fácil de configurar, sobretodo si usamos el proveedor de Google, aún así si usamos otros de los disponibles (Correo/Contraseña, Teléfono, Facebook, Twitter, GitHub, Anónimo) también es muy fácil, sólo es un paso más en el caso de las redes.

- Almacenamiento: Este servicio es muy bueno para aplicaciones que requieran guardar archivos del usuario. También nos sirve si queremos subir estáticos ya que existe un botón desde la interfaz o podemos programar algo. En mi caso lo he usado para subir imágenes desde un formulario y no he tenido ningún tipo de problemas. Como la base de datos, tiene reglas que podemos configurar.

- Hosting: Este servicio es uno de mis favoritos. Con una colección de estáticos (o de archivos que han pasado ya el proceso de build) podemos subir una aplicación y esta automágicamente contará con SSL y HTTP2. Si tenemos una app con Angular o Firebase podemos hacer un build desde nuestros ordenadores y subir estos archivos generados y nos funcionarán sin

problemas. Si necesitamos un Backend tendríamos que subirlo a otro lugar o hacer uso de las Cloud Functions.

También cabe destacar que al hacer deploy de tu aplicación esta hará parte del CDN de Firebase y se replicará en servidores a lo largo de todo el mundo, disminuyendo el tiempo de transferencia desde estos a el ordenador de tu visitante.

- Cloud Functions(BETA): Cloud Functions es un producto bastante reciente de Firebase que tiene como objetivo la transformación de nuestro código del backend en pequeñas piezas del mismo(funciones). Estas funciones son creadas en NodeJS y se suben a la vez que hacemos deploy. Al crearse generan una URL a la que podemos llamar desde AJAX para que se ejecute el código pertinente.

- Laboratorio de tests para Android: Los desarrolladores de Android se habrán planteado la encrucijada que produce muchas veces a la hora de probar una aplicación. Existen múltiples modelos de dispositivos con diferentes versiones y diseño del OS, así como diferentes sensores. Este servicio nos ofrece la posibilidad de probar los tests de nuestra aplicación en los entornos que configuremos(Dispositivo, versión del OS)

- Notificaciones: Este servicio, como su nombre nos indica, nos permitirá gestionar el envío de notificaciones a nuestros usuarios con la diferencia de que estas podrán ser programadas acorde a diferentes parámetros.

Acá nos detenemos en el último servicio que nos provee la API de Firebase en la aplicación Móvil, debido a que nos permite mediante Firebase Cloud Messages (FCM) enviar notificaciones a los dispositivos en determinadas acciones de la plataforma, ya sea, para notificar sobre un pedido, notificar una oferta, mensajes, y demás.
Éste método es realizado a través de Firebase, mediante el Token del dispositivo que se logea en el plugin, por lo tanto el mismo debe correr en un servidor Express desarrollado en NodeJS por su arquitectura, es una de las formas de que funcione.

5.2.1.5 API Mercado Pago

La API de Mercado Pago, permite a la plataforma calcular el costo estipulado del envío, haciendo un cálculo a través del Código Postal del Origen y Destino del Pedido, el volumen de lo que se envía, más el Peso del mismo. Es así que facilita al cálculo o conexión con algún web services para lograr lo mismo, y además ese precio, orienta a Usuario en parte, con consecuencia al gasto que puede causar.

Además, otra función crucial que ejecuta esta API, es crear un link de preferencia, para acceder a las formas de pago, es el servicio más importante que ofrece, ya que nos ahorra todo tipo de validaciones de pago, seguridad en los datos del Usuario, performance con respecto a la plataforma, ya que corre desde el servidor de MercadoLibre, y ahorro en horas de trabajo para desarrollar un algoritmo que efectúe ese tipo de acciones.

5.2.3. Vista Lógica

5.2.4. Modelo de Despliegue

5.2.5 Diseño de Interfaz

5.2.6 Política de Backup

5.2.7 Política de Seguridad

5.3 Pruebas