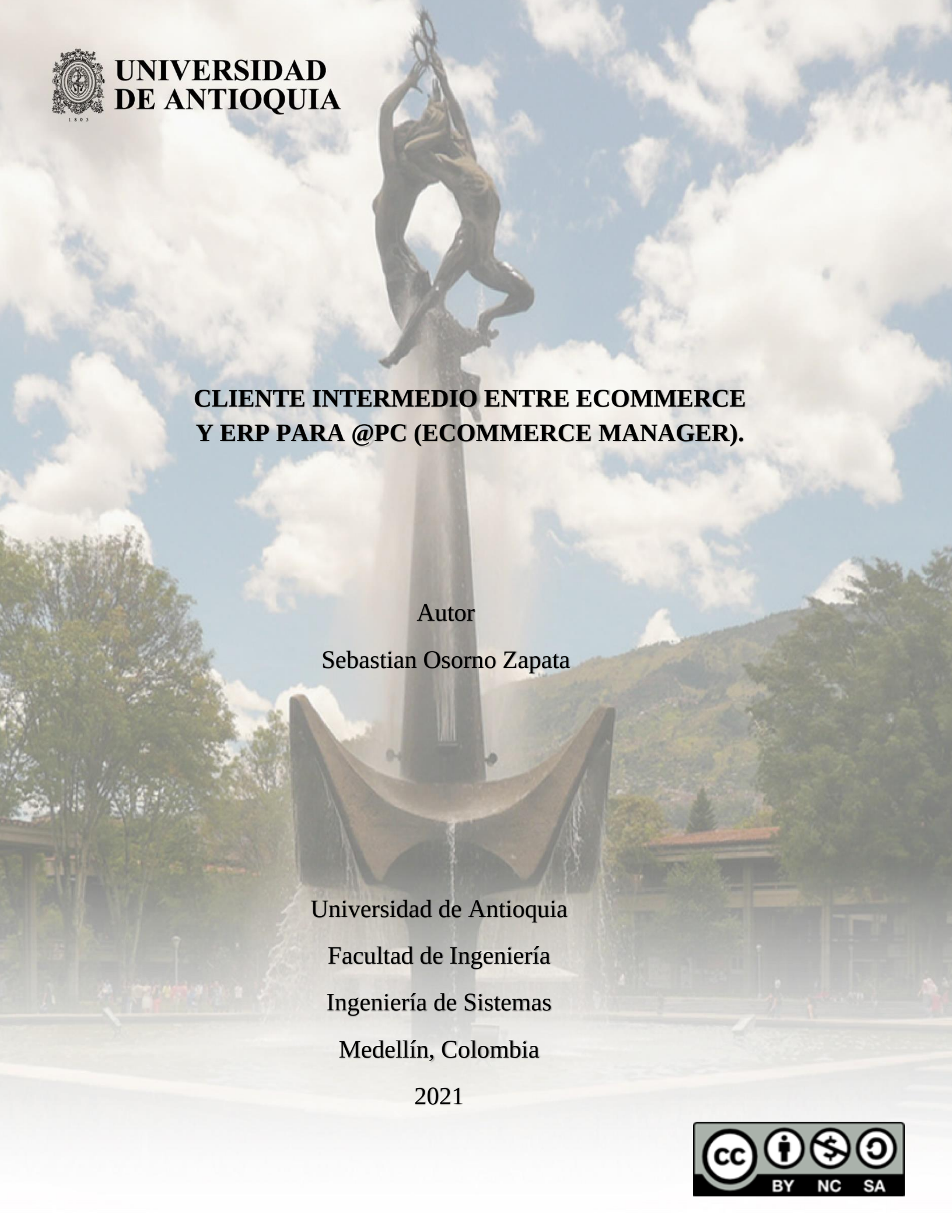

UNIVERSIDAD DE ANTIOQUIA

CLIENTE INTERMEDIO ENTRE ECOMMERCE Y ERP PARA @PC (ECOMMERCE MANAGER).

Autor

Sebastian Osorno Zapata

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Colombia

2021

Cliente intermedio entre Ecommerce y ERP para @PC (Ecommerce Manager).

Sebastian Osorno Zapata

Informe de práctica presentado como requisito parcial para optar al título de:
Ingeniero de Sistemas

Asesores:
Astrid Duque Ramos, Doctora en Ingeniería Informática
Jonathan Diosa Giraldo, Magister en Ingeniería

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Colombia
2021

Cliente intermedio entre Ecommerce y ERP para @PC (Ecommerce Manager).

Informe práctica empresarial

Autor:
Sebastian Osorno Zapata

Asesores:
Astrid Duque Ramos
Jonathan Diosa Giraldo

Universidad de Antioquia

Facultad de Ingeniería

Departamento de sistemas

Medellín, Colombia

Abril 2021

Índice

Índice de figuras

1. Resumen

Las diferentes dinámicas sociales y económicas en el mundo, han generado un crecimiento exponencial de la web y en especial del comercio electrónico(ecommerce). La digitalización de los usuarios es inminente y cada vez más empresas se adhieren a este nuevo paradigma de comunicación y comercio, en la cual se utiliza la información como su mejor aliado y la automatización de procesos, por medio de la tecnología, como herramienta para aumentar sus ingresos. El grupo empresarial @PC se compone de diferentes tipos de empresas como Experimentality, Kaskey, entre otras; @PC como compañía se dedica a la distribución de tecnología, software y hardware, siendo la modalidad online, uno de sus canales de distribución más importantes, y este incluye diversas plataformas de venta como Vtex, Linio, Mercado Libre, Etc. La empresa cuenta con un sistema de gestión de tiendas online, donde se pueden visualizar y gestionar todas las órdenes que se generan y actualizar su inventario, siendo Mercado Libre y Vtex las dos tiendas principales para la compañía. Además, posee un equipo ecommerce el cual usa constantemente la herramienta que integra ambos sistemas, el ERP(Enterprise Resource Planning - sistemas de planificación de recursos empresariales) y las diferentes tiendas, esta herramienta está desarrollada en tecnologías muy antiguas y presenta constantes impedimentos en el acoplamiento de nuevas tiendas, la concurrencia o actualización de la misma; por lo anterior, surgió la necesidad de facilitar a su área de ecommerce la administración de sus diferentes tiendas online en el proceso postventa de cara al ERP, integrando desde la facturación de la orden hasta el envío de la misma y la actualización de inventarios en las diferentes tiendas. Para el proceso de prácticas académicas se definió un proyecto que se encargó de desarrollar e implementar el backend para un sistema que sirve de cliente intermedio y que permite la gestión de órdenes de las tiendas Ecommerce (Vtex - Mercado Libre) para @PC. Este proyecto contó con dos frentes de trabajo, frontend y backend, los cuales interactuaron para lograr la completa integración y funcionalidad del sistema, dentro de los parámetros del alcance definidos para la práctica y permitiendo a futuros equipos continuar dicha actualización de manera desacoplada. En nuestro caso y para este proyecto, se abordó solo el backend del sistema con todo su proceso de análisis, definición y desarrollo.

Palabras clave: Ecommerce, Digital, Web, Tienda online, Integración, Gestión de ventas, Comercio electrónico, Workflow automation, Automation ecommerce management

2. Introducción

El comercio electrónico es definido como todos aquellos aspectos de los procesos comerciales generalmente identificados como una plataforma basada en la web y que se ha convertido en una de las plataformas más importantes para las compañías, permitiendo que se realicen varios tipos de transacciones comerciales en ella, y que ha crecido como un conjunto dinámico de tecnologías para cada una de las partes en el proceso de venta, a través del cual las aplicaciones y las empresas cambian a la forma digital y se da en su mayoría a través de Internet [15]. Además, un modelo comúnmente encontrado dentro del comercio electrónico es aquel que incluye las etapas de difusión, venta y post venta; y todas éstas, forman parte fundamental en el proceso de venta, las cuales implican procesos ya estipulados en la estructura de comercio electrónico y que se han ido refinando con el paso del tiempo. Y aunque éste no es considerado una camisa de fuerza, su efectividad generalmente comprobada, conlleva a ser adoptado por las compañías [20].

En este proyecto nos centraremos en la automatización de la etapa de postventa, la cual aporta una mejora específica en el flujo de venta, ya que de ésta depende en gran medida el grado de satisfacción del usuario y a su vez la capacidad de la compañía de escalar todos sus procesos [17].La automatización o estructuración de los procesos en esta etapa, generalmente es sustentada por las compañías con diferentes aplicativos y herramientas tecnológicas que permiten a sus equipos desempeñar labores efectivas o concentrarse en otras tareas que generen mayor beneficio a los objetivos corporativos [2].

Los procesos de automatización hacen parte primordial de los ideales de @PC y sus objetivos de escalabilidad, pero la herramienta usada hasta el momento dentro de la compañía llamada manager ecommerce, se desarrolló en tecnologías muy antiguas como angularJS [10] y usando mínimos patrones de diseño, arquitectura y escalabilidad. Debido a esta manera de implementar el sistema, algunas de las actualizaciones del mismo son complejas o técnicamente inviables, además de generar dependencia del equipo de desarrollo para cambiar información de uso constante o para realizar un soporte oportuno. Por ende y al ver este tipo de situaciones, surgió la necesidad de mejorar estos procesos, no solo de cara a las parte comercial y gestión, sino también de la parte técnica del mismo, su proyección y su estabilidad. Una vez identificada la necesidad del equipo y las principales falencias del proyecto anterior, se definió un nuevo proyecto con los puntos anteriores como parte de sus objetivos y con el fin de implementar esta mejora.

La ruta del proyecto fue desarrollada a través de la aplicación de metodologías ágiles de desarrollo como SCRUM [23], y contó con dos frentes de trabajo, frontend y backend, y como se menciono anteriormente la implementación del nuevo sistema se llevó a cabo por consumos de APIs [14] de las plataformas (ecommerce y ERP) que interactúan, lo cual permite una trazabilidad de manera efectiva y centraliza todas las tareas de la venta ecommerce en un solo sistema, disminuyendo así los tiempos de gestión para cada tienda, al igual que la mano de obra requerida para la administración.

Como parte de las limitaciones iniciales y aunque se cuenta con una herramienta ya creada, debido a las dificultades técnicas planteadas anteriormente, este sistema se construyó desde el inicio, lo que requirió una mayor cantidad de trabajo de lo esperado y por ende, solo se incluyó en la propuesta el módulo de gestión de órdenes para Vtex y MercadoLibre, sin procesos de rastreo o envío, en virtud de cumplir los objetivos de las practicas académicas, los requerimientos de calidad del software y el tiempo estimado en la propuesta. Por otra parte, durante el proceso de desarrollo y en reuniones con el equipo ecommerce de la compañía, se detectó un inconveniente adicional en el ERP, ya que este tiene ciertas limitaciones técnicas, trayendo más operatividad e incluso problemas que no se podrían automatizar, como la facturación electrónica, por la falta de datos desde las tiendas debido a los acuerdos contractuales de los Marketplace.

3. Objetivos

3.1. Objetivo general

Optimizar el proceso de gestión, trazabilidad y administración de las tiendas online, llevado a cabo por el equipo ecommerce de @PC, disminuyendo así los tiempos de gestión para cada tienda, al igual que la mano de obra requerida para la administración.

3.2. Objetivos específicos

- Analizar y definir los flujos de procesos del sistema dentro del negocio.
- Desarrollar e implementar el backend para un sistema que servirá de cliente intermedio y que permitirá la gestión de órdenes de las tiendas Ecommerce (Vtex - Mercado Libre) para @PC en sus módulos de órdenes, orientados por las reglas del negocio.
- Integrar el sistema con las diferentes aplicaciones que interactúen en el proceso.
- Validar los requerimientos técnicos y de negocio.

4. Marco teórico

Como parte del proyecto, se presentarán en esta sección una serie de conceptos técnicos para el entendimiento del proceso realizado en la elaboración de la herramienta. Para el sistema, en la parte de back, se usó el lenguaje JavaScript con el entorno NodeJS, acompañado de una base de datos no relacional la cual solo almacena los logs y estados de los procesos. Posterior a esto, la aplicación se virtualizó en un contenedor Docker, automatizando este proceso apoyados en Circle CI y GitHub, y a su vez, el marco de trabajo en el que se basa la gestión del proyecto es SCRUM, aplicado por medio de la herramienta Jira:

4.1. NodeJS

Es una tecnología desarrollada en el 2009 por Ryan Dahl, la cual está basada en el motor V8 de Google y que hace de este, un lenguaje del lado del servidor, potente y fácil de usar [7]. Además, actúa como un entorno de ejecución de JavaScript enfocado a trabajar con eventos asíncronos, y es utilizado principalmente en el desarrollo de aplicaciones web. Su importancia reside en la capacidad de actuar como un entorno no bloqueante, es decir, no existen en general funciones que realicen operaciones de entrada y salida directamente, lo que lo hace ideal para sistemas con una alta escalabilidad. NodeJS es una plataforma muy estable y cuenta con un sistema de versionamiento constante y estructurado, lanzando nuevas funcionalidades en estado de pruebas durante 6 meses, hasta que las mismas se convierten en versiones abiertas para uso general, lo que proporciona a los desarrolladores y autores de librerías el tiempo para actualizar su soporte a las nuevas versiones. [18].

4.2. Base de datos no relacional

También conocida como NoSQL [11], las bases de datos no relacionales son en sí, un sistema de gestión de datos diferente al tradicional SQL y que nace como respuesta a las necesidades de la industria, donde se requerían sistemas de alta concurrencia de lectura y baja latencia de escritura, alta escalabilidad y disponibilidad, entre otras. Basadas en este tipo de sistema, se han creado diferentes bases de datos en la industrias como CouchDB, cassandra, RavenDB, entre otras. Para nuestro sistema, usaremos MongoDB [16], la cual fue creada por Dwight Merriman en 2007 y es definida como base de datos documental, lo que significa que almacena datos en forma de documentos tipo JSON (JavaScript Object Notation) y será implementada en NodeJS por medio de una librería llamada Mongoose.

4.3. Despliegue continuo

Es un conjunto de mejoras para el despliegue de código a la hora de construir software, y hereda de una practica llamada entrega continua [3], la cual, tiene como objetivo automatizar el flujo de implementación del código generalmente en los ambientes productivos. A diferencia de su predecesora, el despliegue continuo incluye aspectos como la automatización de pruebas, lo que completa el flujo de despliegue de código y se encarga de validar el correcto funcionamiento del sistema, una vez esté integrado [22].

4.4. Integración continua

Práctica que consiste en una integración frecuente del código y sus cambios por parte de los desarrolladores del proyecto, permitiendo tener nuevas funcionalidades de manera constante y facilitando la fusión del código cuando se trabaja de forma conjunta. Generalmente se cuenta con herramientas encargadas de validar el adecuado funcionamiento del código dentro del proyecto y sistemas de

control de versionamiento, que facilitan la orquestación del la estructura y la inclusión de las nuevas entregas [6].

4.5. CircleCI

Es una herramienta de automatización que controla el flujo de despliegue, la validación del proyecto y su funcionamiento. CircleCI se encarga de estar a la espera de nuevas entregas de desarrollo dentro del repositorio de código, para comenzar el flujo de integración y pruebas predefinidas por medio de un archivo de configuración, para luego, ejecutar e instalar la aplicación [4].

4.6. Github

Es una plataforma pensada para alojar código de manera distribuida y que permite el control de versiones para el desarrollo de software, Github tiene la capacidad de conectarse con otros aplicativos por medio de integraciones nativas que posibilitan la automatización de procesos como la integración y el despliegue. Además, permite realizar un adecuado control de versiones y ambientes por medio del proceso llamado gitflow, el cual determina una serie de reglas y metodologías que permiten mejorar la estructura, eficiencia y consistencia del proyecto [8].

4.7. Docker

Es un sistema de código abierto que permite automatizar el despliegue, y se basa en la idea de generar contenedores para encapsular las aplicaciones. Los contenedores son en sí, una abstracción en la capa de la aplicación que empaqueta el código y las dependencias, y permite que dicho sistema se ejecute de forma rápida y confiable de un entorno informático a otro [21].

4.8. SCRUM

Es un conjunto flexible de actividades que combina herramientas y técnicas conocidas para generar ciclos de desarrollo iterativo e incrementales [23], fue creada por Ikujiro Nonaka e Hirotaka Takeuchi y que hace referencia a un tipo de formación que se realiza en el rugby para aumentar la productividad y flexibilidad del equipo [24]. SCRUM es una de las más utilizadas por las compañías en la actualidad, tiene una enorme comunidad tras ella y está muy bien documentada [9]. Una de las características de SCRUM es la creación de ciclos breves para el desarrollo, los cuales se conocen como iteraciones, pero para nuestro proyecto fueron denominados como “Sprint” [19]. SCRUM proviene desde el enfoque de metodologías ágiles, la cuales generalmente son resumidas en su manifiesto inicial, donde se determinan cuatro puntos claves para el éxito y mejora en la gestión de proyectos, los cuales son: Individuos e interacciones sobre procesos y herramientas, software de trabajo sobre documentación completa, colaboración

con el cliente sobre la negociación de contratos y responder al cambio sobre seguir un plan.

4.9. Jira

Esta es una herramienta pensada para facilitar el proceso de administración, gestión y progreso de los proyectos de software principalmente, pero a lo largo de los años se ha ampliado a una serie de productos que permiten a las compañías integrar todo tipo de soluciones a sus procesos. Jira ha implementado a sus soluciones flujos que son altamente compatibles con las metodologías ágiles de desarrollo como SCRUM y hacen de esta herramienta un recurso ideal para la gestión de proyectos de desarrollo [1].

5. Metodología

El proyecto se abordó en dos etapas, la primera consistió en definir las bases del mismo y todas aquellas tareas verticales utilizadas en el desarrollo. La segunda, está relacionada con la metodología empleada para el seguimiento, SCRUM. La cual implementó cuatro grandes fases dentro de cada iteración(Sprint) y que se aplicó de la misma manera en cada ciclo de desarrollo, permitiendo la correcta validación del riesgo, tiempos de desarrollo y niveles de prioridad para cada funcionalidad.

5.1. Etapas del proyecto:

5.1.1. Etapa inicial:

- **Elegir las tecnologías actuales que más se acoplan a este tipo de sistemas y a las reglas de negocio.** Como parte de este proceso inicial se tuvo en cuenta las tecnologías de uso más común dentro de la compañía, por su fácil mantenimiento, soporte y desarrollo a largo plazo. Además, se esperaba que las mismas fueran actuales, con buenas proyecciones de soporte, que tuvieran una curva de aprendizaje que no afectara los tiempos de ejecución, con facilidad de estructuración, cambio y escalabilidad, pensando en el crecimiento del proyecto y en el acoplamiento de nuevas funciones. Basados en esto, las tecnologías elegidas fueron NodeJS como framework de javascript para la codificación del sistema, MongoDB como motor de bases de datos, docker como plataforma de virtualización, CircleCI como herramienta de integración continua y Linux como sistema operativo de despliegue.

- **Implementar patrones arquitectónicos que permitieran la mantenibilidad del sistema y definir la arquitectura backend que permitiera un bajo acoplamiento y una alta escalabilidad, respecto a los sistemas integrales externos.** Para nuestro sistema, se tuvieron en cuenta los temas técnicos y los aspectos del contexto, costos, mano

de obra para mantener dicho sistema y demás factores. En este caso se deseaba desarrollar un cliente intermedio, el cual es un concepto que inicialmente viene desde el inicio de los sistemas distribuidos, permitiendo así comunicar múltiples aplicativos de manera transparente y homogénea, además de otras características según el tipo de sistemas involucrados[13]. Basados en la definición anterior, se seleccionaron dos arquitecturas compatibles. La primera es la arquitectura cliente servidor, la cual recomienda dividir las cargas de trabajo de los diferentes componentes, y la otra es la arquitectura por capas o niveles, la cual determina que cada capa del sistema se debe encargar de una responsabilidad o función y que provee dicha información a las demás capas, sin que ellas tengan que estar al tanto de dicha abstracción. Como resultado y entendiendo que el sistema debe estar en armonía con la parte de frontend, se decidió usar una arquitectura híbrida (Figura 1), ya que éstas comparten ciertas características y se facilita el aprovechamiento de los puntos fuertes de cada una, aplicando los lineamientos resultantes en temas como la estructuración del proyecto, sus módulos y componentes, al dividirlos en subgrupos que se encargan de un proceso especifico como por ejemplo las rutas, los servicios, las configuraciones, etc o en la definición de responsabilidades entre el front y el back (Figura 2).

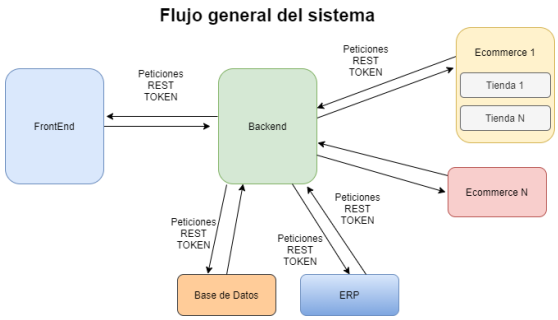

Figura 1: Flujo general de interacción backend - Arquitectura

- **Definir parámetros de seguridad y comunicación de datos entre los diferentes sistemas.** Para nuestro sistema se definieron dos métodos de seguridad principales desde el back y que solo permiten el uso de estos servicios a personas autorizadas por la compañía, ya que la información gestionada no debe ser divulgada públicamente. Como primera medida, el sistema solo está disponible desde la red interna de la compañía, desplegado en uno de los servidores físicos de la misma y solo recibe peticiones de esta red, ya que no se cuenta a nivel de servidor con una ip pública expuesta que pueda acceder a los puertos del aplicativo; para los empleados que laboran de manera virtual, se tienen implementados protocolos de conexión VPN con credenciales propias asignadas a cada uno de ellos y

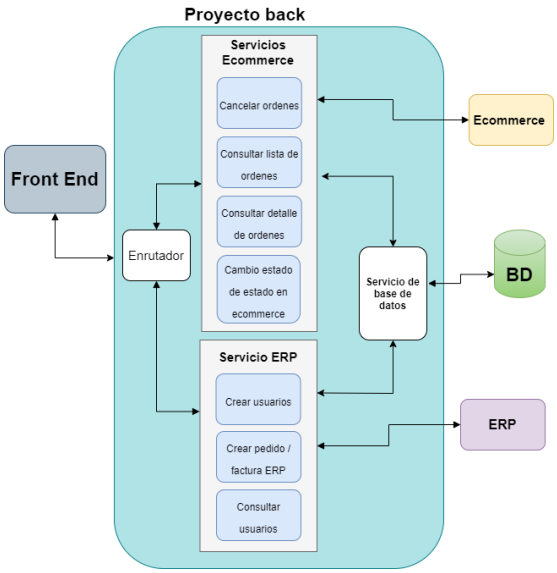

Figura 2: Diagrama estructura proyecto Backend

que permiten dicha conexión desde otros lugares. Como una medida extra se definieron reglas de consumos de los servicios, los cuales solo pueden ser usados si se posee un usuario y token que lo permita, incluso estando dentro de la red de la compañía.

- **Definir estándares de codificación, nombramiento de variables, archivos y endpoints para el backend.** Ya que en la actualidad no existe ningún estándar aceptado por toda la comunidad o determinado para Javascript, para la definición de variables, nombramientos de archivos o definición de métodos, se optó por tomar como punto de referencia, las guías de estilos de organizaciones reconocidas que en ocasiones son mantenidas por la misma comunidad de desarrolladores javascript [5]. Como complemento a esta referencia, se usó el resultado de investigaciones de algunos autores como Gabor Lóki y Péter Gál [12], en los casos donde la guías de estilos anteriores no determinaban como proceder en situaciones como la definición de modelos y estructuras del proyecto, y así evitar inconsistencias de nombramiento.

- **Crear modelos y estándares de almacenamiento de información para la base de datos.** Como parte del proceso de creación de dichos modelos y dentro de la metodología de almacenamiento de una base de datos no relacional, las diferentes fuentes de información se subdividieron en colecciones, las cuales se determinaron según el tipo de datos a almacenar, una de ellas es logs de errores y la otra, avance de la gestión de cada orden para evitar conflictos en situaciones de fallo y contar con in-

formación general de las gestiones para llevar un control de los procesos. Dicha estructura facilita la depuración y gestión de almacenamiento de la base de datos en caso de ser requerido, ya que la misma permite ser filtrada según la colección y en algunos casos, según el número de orden o plataforma ecommerce.

5.1.2. Etapa de iteraciones:

- **Diseño y análisis:** Se elicitó y estructuró los requisitos funcionales y no funcionales del sistema.
- **Codificación y desarrollo:** Se implementaron procesos de despliegue continuo e integración continua para el backend.
- **Integración:** Se integraron los módulos de órdenes para plataformas ecommerce más usadas por @PC (Vtex- Mercado Libre).
- **Despliegue:** Se desplegó y aprovisionó el sistema en sus propios contenedores, con sus métodos de automatización y dentro de los servidores de la compañía.

6. Resultados y análisis

Al finalizar la ejecución del proyecto propuesto inicialmente para esta práctica empresarial, se obtuvo como resultado una herramienta en su etapa inicial, pero con una arquitectura ya definida y la capacidad de ser integrada al antiguo sistema, permitiendo así el cambio progresivo de dicha herramienta y abarcando gradualmente cada uno de sus módulos actuales en nuevas etapas del desarrollo.

Para este proyecto como uno de los acuerdos iniciales entre los desarrolladores y negocio, se definió la base de las rutas de consumo como un estándar y que varía según el ambiente al cual se esté apuntando, se determinaron y suministraron a las áreas de desarrollo involucradas los respectivos token, métodos, cargas de consumo y demás bases para iniciar el proyecto, además de que se definieron los puertos y rutas usados para que la aplicación fuera expuesta; por temas de confidencialidad no se exponen dichos datos en el presente documento.

6.1. Rutas de consumo

Los servicios expuestos en la aplicación se encargan de exponer y recibir la información para el frontend, internamente el sistema realiza solicitudes y cambios de información a los sistemas externos como el ERP o la base de datos y después de esto, gestiona la información para retornar una respuesta adecuada.

6.1.1. Vtex

- **Obtener lista de ordenes:** Por medio de esta ruta se obtiene un listado de ordenes según, los criterios enviados como parámetros de consulta.

Método: GET /manager/environment/vtex/getOrdersList

- **Obtener detalle de orden:** Por medio de esta ruta se obtiene un el detalle completo de una orden, según el orderId enviado como parte de la petición.

  Método: GET /manager/environment/vtex/getOrder/orderId

- **Cambiar orden a estado handling:** Por medio de esta ruta se cambia el estado de una orden a handling, según el orderId enviado como parte de la petición.

  Método: POST /manager/environment/vtex/startHandling/orderId

- **Cambiar orden a estado invoice:** Por medio de esta ruta se cambia el estado de una orden a invoice, según el orderId enviado como parte de la petición.

  Método: POST /manager/environment/vtex/invoiceNtfVtex/orderId

6.1.2. Mercado Libre

- **Obtener lista de tiendas:** Por medio de esta ruta se obtiene un listado de las tiendas habilitadas, según los criterios enviados como parámetros de consulta.

  Método: GET /manager/environment/mercLibre/getStores

- **Obtener lista de ordenes:** Por medio de esta ruta se obtiene la lista de ordenes disponibles para gestionar, los criterios enviados como parámetros de consulta. Además, se cuenta con un parámetro de ruta opcional, el cual permite realizar búsquedas de ordenes por tiendas especificas según su id de tienda.

  Método: GET /manager/environment/mercLibre/getOrdersListByStore/store

6.1.3. ERP

- **Crear usuario en el ERP:** Por medio de esta ruta, se crea a un usuario como cliente en los sistemas de la compañía, para que luego pueda ser asociada una factura.

  Método: POST /manager/environment/ERP/createUserERP

- **Obtener lista de tiendas:** Por medio de esta ruta se crea un pedido en el ERP, y a su vez se genera la factura correspondiente.

  Método: POST /manager/environment/ERP/invoiceOrderERP

Al inicio del proyecto, se definieron las estructuras de almacenamiento de la base de datos y sus modelos. Por lo que quedó subdividida en dos colecciones diferentes, la primera es “StepController”, la cual contiene dos campos principales “orderId” como clave única para cada documento y “steps” el cual se

encarga de almacenar el avance de la gestión de la orden y los cambios de sus estados. Por otro lado se tienen la colección "ErrorType", la cual se encarga de almacenar los registros de errores, en caso de presentarse, y así tener registro del mismo; esta colección esta compuesta por documentos que contienen los siguientes campos, "orderId" nuevamente como clave única de la gestión que se esta realizando, "status" y "code" como códigos de errores ya predefinidos, "message" que almacena el texto del error según la etapa en la que se presente y "data" que guardará los datos en manejo en el momento del error.

En el proceso de mejora del sistema respecto a la herramienta anterior, se implementó un proceso de integración continua y despliegue continuo, por medio de la unión de varias herramientas usadas dentro de la compañía, que son CircleCI, GitHub y Docker. Este proyecto se encuentra alojado en GitHub y posee una configuración de los flujos de trabajo del despliegue dentro de circleCI como lo muestra la figura 3, los cuales, están definidos para la rama Master en ambiente productivo y en la rama develop para el ambiente de desarrollo. El proceso de automatización y despliegue se lleva cabo en el servidor de la compañía, dentro de un contenedor Docker y allí se encarga de exponer los servicios construidos, por medio de la virtualización, de manera aislada y bajo una configuración predeterminada (Figura 4). En la figura 5, se presenta un diagrama general de flujo seguido por el despliegue y los puertos correspondientes según la rama.

```yaml
workflows:
  version: 2
  build-deploy:
    jobs:
      - build:
          filters:
            branches:
              only:
                - develop
                - master
      # Deploy on Develop
      - deploy-dev:
          requires:
            - build
          filters:
            branches:
              only: develop
      # Deploy on Production
      - deploy-pro:
          requires:
            - build
          filters:
            branches:
              only: master
```

Figura 3: Configuración de ramas en CircleCI

Después de haber definido las bases del sistema, se procedió a codificar e implementar lo que se había planteado anteriormente, permitiendo así, visualizar

```
FROM node
ADD . /usr/src/integrador-ecommerce-apc
WORKDIR /usr/src/integrador-ecommerce-apc
RUN npm install
CMD ["npm", "start"]
```

Figura 4: Configuración archivo Docker

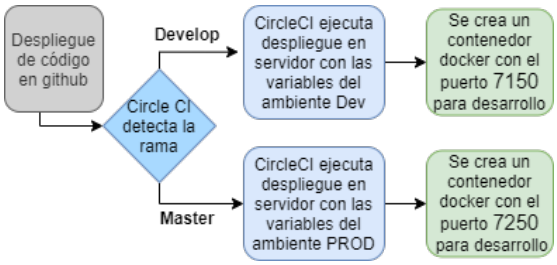

Figura 5: Flujo de despliegue automatizado

algunos temas que no se contemplaron dentro de la estimación inicial, pero que hacen parte del proceso natural del desarrollo de software. Para este caso, se encontró que la integración de clientes de cara al ERP no era la esperada por el sistema planteado, ya que esta sufrió cambios recientes por la adopción de la factura digital dentro de la compañía y que a su vez algunas tiendas por temas de confidencialidad, no están entregando datos de sus clientes con sus distribuidores, datos que al final son requeridos para este tipo de facturación. Por lo anterior, se hizo necesario plantear una opción dinámica, tanto en la respuesta del servicio ante un consumo, como en la interacción de frontend para el usuario, lo que ocasionó algunos retrasos en el proyecto y que al final hacen parte del trabajo futuro propuesto para este sistema.

7. Conclusiones

Al finalizar este proyecto, se puede identificar que los objetivos planteados inicialmente fueron cumplidos en cada una de las fases, y que la herramienta representa una parte importante de las funciones del equipo ecommerce de @pc, ya que esta nueva versión disminuye los tiempos de indisponibilidad y reprocesos manuales, los cuales aumentaban el tiempo de gestión y disminuían la efectividad de respuesta.Y por otro lado, se obtuvo un cumplimiento satisfactorio del análisis, desarrollo, integración y validación, soportado en los recursos y resultados obtenidos.

Se puede resaltar que herramientas como Circle CI, github y docker, juegan un papel importante dentro de este proceso de desarrollo, actuando como

facilitadores en la implementación de los nuevos proyectos y permitiendo a los desarrolladores enfocarse en otros aspectos como las definiciones de arquitectura o la implementación de nuevas funcionalidades.

Se propone como trabajo futuro una nueva fase de proyecto, que permita desarrollar más funcionalidades y la estandarización del proceso, ajustado a los nuevos lineamientos de la facturación electrónica, junto a la integración de nuevos módulos como catalogo o inventario.

Como parte de las lecciones aprendidas y hablando del proceso de desarrollo bajo un marco de metodologías ágiles, se pudo detectar que este tipo de practicas permiten mejorar todo el flujo de desarrollo de principio a fin, pero que aun así es un proceso continuo de mejora y de trabajo en equipo, donde se incluyan a las personas involucradas con la aplicación y que debe ser riguroso en su implementación, para disminuir en gran medida los errores de estimación y desarrollo del proyecto. Además, en un proceso de aprendizaje personal como este, se pudo notar que la experiencia en el campo ayuda ampliamente en temas tan complejos como lo es estimar, y que contar con un equipo que posea madurez en el tema y capacidad, agrega un nivel de estabilidad significativa al proyecto.

Las limitaciones encontradas en el desarrollo de este proyecto fueron: la dificultad inicial sobre la definición del alcance del mismo, la estimación, la propuesta y la disponibilidad de tiempo de ejecución, por los tiempos determinados dentro de los lineamientos para las practicas laborales. Requiriendo así, varias redefiniciones de alcance del proyecto con apoyo de la asesora.

Es importante resaltar el papel de la academia en este proceso, al brindar las bases adecuadas para afrontar este tipo de situaciones, en los que se debe tomar decisiones trascendentales desde la parte técnica y corporativa; pero por otro lado, surge la necesidad continua de que la misma academia se encuentre en constante evolución, en el mejoramiento de sus practicas educativas y a la vanguardia del mercado que avanza a pasos agigantados. Dentro de las líneas de aprendizaje que más influyeron en la correcta ejecución de este proyecto, esta la de desarrollo de software, la cual por medio de sus cursos orientados a la industria, a las tecnologías actuales más usadas y las nuevas tendencias, permiten al estudiante experimentar lo más similar a la industria posible, dentro de un ambiente académico.

Referencias

[1] Atlassian. *About Jira [Accessed 23 May 2021]*. 2021. URL: `https://www.atlassian.com/es/software/jira/guides/getting-started/overview#jira-software-hosting-options`.

[2] Amit Basu y Akhil Kumar. «Research commentary: Workflow management issues in e-business». En: *Information Systems Research* 13.1 (2002), págs. 1-14.

[3] Lianping Chen. «Continuous delivery: Huge benefits, but challenges too». En: *IEEE Software* 32.2 (2015), págs. 50-54.

[4] Inc. Circle Internet Services. *About CircleCI [Accessed 23 May 2021]*. 2021. URL: https://circleci.com/docs/2.0/about-circleci/.

[5] MDN contributors. 2021. URL: https://developer.mozilla.org/en-US/docs/MDN/Guidelines/Code_guidelines/JavaScript.

[6] Paul M Duvall, Steve Matyas y Andrew Glover. *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[7] S. Evans. *Nodejs Essentials*. CreateSpace Independent Publishing Platform, 2016. ISBN: 9781540404640. URL: https://books.google.com.co/books?id=brgeMQAACAAJ.

[8] Inc. GitHub. *About GitHub [Accessed 23 May 2021]*. 2021. URL: https://github.com/about.

[9] Oscar Tinoco Gómez, Pedro Pablo Rosales López y Julio Salas Bacalla. «Criterios de selección de metodologías de desarrollo de software». En: *Industrial data* 13.2 (2010), págs. 70-74.

[10] Brad Green y Shyam Seshadri. *AngularJS*. .ºReilly Media, Inc.", 2013.

[11] Jing Han y col. «Survey on NoSQL database». En: *2011 6th international conference on pervasive computing and applications*. IEEE. 2011, págs. 363-366.

[12] Gábor Lóki y Péter Gál. «JavaScript Guidelines for JavaScript Programmers». En: (2018).

[13] FRANCISCO DE ASIS LOPEZ FUENTES. *Sistemas distribuidos*. 1.ª ed. Universidad Autónoma Metropolitana, Cuajimalpa, 2015.

[14] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. .ºReilly Media, Inc.", 2011.

[15] Nurhizam Safie Mohd Satar, Omkar Dastane y Muhamad Yusnorizam Ma'arif. «Customer value proposition for E-Commerce: A case study approach». En: *International Journal of Advanced Computer Science and Applications* 10.2 (2019).

[16] Inc. MongoDB. *MongoDB [Accessed 20 May 2021]*. 2021. URL: https://www.mongodb.com.

[17] Tahir M Nisar y Guru Prabhakar. «What factors determine e-satisfaction and consumer spending in e-commerce retailing?» En: *Journal of retailing and consumer services* 39 (2017), págs. 135-144.

[18] FUNDACIÓN NODE.JS. *Acerca — Node.js. Node.js [online]. 2019. [Accessed 20 May 2021]*. 2019. URL: https://nodejs.org/es/about/.

[19] Paulo Pereira, Paula Torreão y Ana Sofia Marçal. «Entendendo Scrum para gerenciar projetos de forma ágil». En: *Mundo PM* 1 (2007), págs. 3-11.

[20] Zheng Qin y Zheng Qin. *Introduction to E-commerce*. Vol. 2009. Springer, 2009.

[21] Babak Bashari Rad, Harrison John Bhatti y Mohammad Ahmadi. «An introduction to docker and analysis of its performance». En: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), pág. 228.

[22] Akond Ashfaque Ur Rahman y col. «Synthesizing continuous deployment practices used in software development». En: *2015 Agile Conference*. IEEE. 2015, págs. 1-10.

[23] Ken Schwaber. «Scrum development process». En: *Business object design and implementation*. Springer, 1997, págs. 117-134.

[24] Hirotaka Takeuchi e Ikujiro Nonaka. «The new new product development game». En: *Harvard Business Review* 64.1 (1998), pág. 321.