UNIVERSIDAD DE ANTIOQUIA

Sistema visualizador de productos de canales e-commerce

Autor(es)

Andrés Felipe Isaza Arboleda

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas

Medellín, Colombia

2021

Sistema visualizador de productos de canales e-commerce

Andrés Felipe Isaza Arboleda

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Ingeniero de Sistemas

Asesores (a):
Astrid Duque Ramos, Doctora en Ingeniería Informática
Frank Alexis Castrillon Giraldo, Ingeniero de Sistemas

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

Índice

Resumen

Desde hace algunos años la interacción entre individuos a través de las tecnologías de la información ha incrementado exponencialmente, existen miles de millones de dispositivos en el mundo con la capacidad de conectarse a internet. Con este auge, las compañías se están adaptando a ofrecer sus productos y servicios a través de internet, ya que es un mercado que está en constante crecimiento, y con el impacto que ha tenido la pandemia en todo el mundo, disponer de canales virtuales se convirtió en una necesidad para las compañías. Para dar solución a esta necesidad, algunas grandes compañías ofrecen productos y servicios B2B (business-to-business o empresa a empresa) a pequeñas y medianas empresas para que estas puedan comercializar sus productos y servicios, una de estas soluciones son las tiendas e-commerce. El grupo empresarial @PC se encarga de comercializar equipos y accesorios tecnológicos a través de diferentes canales, ya sea físico, con puntos de venta en diversos lugares de Antioquia, por vía telefónica, o por medios digitales como Mercadolibre y VTEX. Como ya se ha mencionado, la demanda de canales virtuales de comercialización ha aumentado, y para @PC no ha sido la excepción, forzandolos a responder a esta demanda con un equipo de e-commerce que gestiona todas las ventas de la compañía en los canales virtuales. Sin embargo, debido a que la compañía posee diversos canales de comercialización electrónicos, surgió la necesidad que el equipo de e-commerce esté sincronizado con estos canales para gestionar los productos en dichos canales, este proceso se torna muy lento ya que los miembros del equipo deben estar revisando cada una de las tiendas para garantizar que haya disponibilidad de los productos y evitar perder ventas, por lo que surgió la necesidad de desarrollar un sistema que permitiera al equipo de e-commerce conocer la disponibilidad de los productos en las tiendas de Mercadolibre y VTEX de forma rápida, y además se pudiera adaptar a los demás canales en el futuro.

Palabras clave: E-commerce, Aplicación web, React, Node, Frontend, Backend, API.

Introducción

@PC Mayorista es una compañía dedicada a la comercialización de artículos tecnológicos, para llevar a cabo esta misión disponen de diferentes canales de venta digitales, como son Mercadolibre, Linio y Buyruru, además dispone de algunos puntos de venta físicos. La compañía se apoya de su equipo de e-commerce que se encarga de impulsar las ventas por medio de sus canales digitales, el equipo comercial necesita conocer el inventario de los productos que se encuentran publicados en las tiendas en todo momento para garantizar la disponibilidad y asesorar a los clientes, actualmente el proceso para consultar el inventario de productos se realiza consultando en cada uno de los canales digitales y mediante una hoja de cálculo que es generada por el ERP de la compañía, esto hace que su trabajo no sea muy eficiente puesto que deben repetir esta tarea varias veces al día, tantas veces como necesiten conocer esta información. Conociendo este problema, surge la necesidad de desarrollar una aplicación que agrupe todo el inventario de productos de las tiendas y el ERP de la compañía y exponga dicha información de forma consistente al equipo de e-commerce, con el fin de que la información sea consultada rápidamente y así facilitar la labor de este equipo.

Objetivos

Objetivo General

Permitir que el equipo de E-commerce de @PC Mayorista pueda consultar y visualizar los productos expuestos a la venta en los canales de comercialización de la compañía.

Objetivos específicos

- Proponer e implementar una arquitectura para el sistema que cumpla con las necesidades de la compañía.
- Diseñar un flujo de integración continua y entrega continua que permita realizar desplegar el sistema
- Desarrollar un sistema de información que permita al equipo e-commerce consultar el inventario de productos en los canales digitales Mercadolibre y Buyruru.
- Diseñar e implementar una API bajo los principios REST que exponga los servicios web necesarios para consultar los productos.
- Diseñar e implementar una interfaz de usuario que permita visualizar los productos.

Marco Teórico

En esta sección se definirán los principales conceptos utilizados durante la elaboración del proyecto. Para el diseño y arquitectura del sistema se eligió el modelo cliente/servidor, el cliente en sistema es una aplicación de frontend construida con la librería ReactJS y el servidor es una aplicación backend desarrollada con el framework de Node conocido como Nest, este backend fue construido como una API REST que expone los servicios web para que los datos puedan ser consultados por el frontend. A nivel de proceso se trabajó bajo el marco de las metodologías ágiles, y la metodología utilizada fue scrum.

Modelo cliente/servidor

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. Tanto el Cliente como el Servidor son entidades abstractas que pueden residir en la misma máquina o en máquinas diferentes (Marini, 2012).

Frontend

En el contexto de la web, el frontend es la parte que el usuario ve e interactúa, como los menús, los formularios de contacto, etc. Para diseñar y desarrollar dicha interfaz web frontend, hay que utilizar ciertas herramientas y tecnologías, que suelen ser una combinación de HTML, CSS y JavaScript, todo ello controlado por el navegador. (Abdullah & Zeki, 2014)

Backend

Cuando hablamos del frontend y backend, nos referimos al modelo cliente-servidor. En este orden de ideas, el backend es la parte que se ejecuta en un servidor en algún lugar de la nube y, entre múltiples cosas, es responsable de tratar los datos en términos generales. Esto significa que es responsable de recibir peticiones de los clientes para que les proporcionen datos, para que puedan visualizarlo (Filipova & Vilão, 2018). Además de esto, el backend tiene más responsabilidades, como manejar la lógica principal en las aplicaciones, garantizar la seguridad e integridad de los datos, entre otras tareas.

API

Una API, en inglés, Application Programming Interface es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo, otorga flexibilidad, simplifica el diseño, la administración y el uso de las aplicaciones. (¿Qué es una API?, 2021)

REST

REST (Representational State Transfer) significa Transferencia de Estado Representacional. Es un término acuñado por Roy Fielding para referirse a un estilo arquitectónico de software. REST es un híbrido (unión) estilo que tiene como objetivo inducir ciertas arquitecturas propiedades que son importantes para los sistemas hipermedia distribuidos. Estas propiedades incluyen usabilidad, simplicidad, escalabilidad y extensibilidad. Introduciendo estas propiedades con compensaciones cuidadosamente consideradas, REST intenta minimizar la latencia y las comunicaciones de red, y al mismo tiempo, maximizando la independencia y escalabilidad de implementaciones de

componentes, incluyendo el agente de usuario, el proxy pasarela, servidor de origen y varios conectores, en sistemas hipermedia distribuidos. (Li & Chou, 2011)

Javascript

JavaScript es un lenguaje de programación que se puede aplicar a un documento HTML y usarse para aplicar interactividad dinámica en las páginas web. Fue inventado por Brendan Eich en 1995, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla.
("Fundamentos de JavaScript | MDN", 2021)

Typescript

Typescript es un lenguaje de código abierto desarrollado y mantenido por Microsoft y basado en Javascript, una de las herramientas más utilizadas del mundo, y le provee definiciones de tipos estáticos que otorgan un mecanismo para describir la forma de un objeto, proporcionando una mejor documentación y permitiendo que Typescript valide que el código está funcionando correctamente mientras se está escribiendo. (Bierman et al., 2014)

React

React es una librería de Javascript para crear interfaces de usuario que fue lanzada en 2013. React fue creado originalmente por Facebook para resolver los desafíos que implica el desarrollo de interfaces de usuario complejas con conjuntos de datos que cambian con el tiempo. Esto no es una tarea trivial y no sólo debe ser mantenible, sino sino también escalable para trabajar a la escala de Facebook, estos trabajaban bajo la arquitectura MVC (Modelo Vista Controlador) (Gackenheimer, 2015), React cambió la forma en que se crearon estas aplicaciones haciendo algunos atrevidos avances en el desarrollo web.

Patrón de componentes de presentación y contenedores

Es un patrón para la separación de responsabilidades en aplicaciones del lado del frontend, los componentes contenedores poseen la lógica y el estado de una parte específica de la aplicación, además de que definen el comportamiento que van a tener sus componentes hijos cuando el usuario ejecuta un evento. Los componentes de presentación simplemente presentan el contenido al usuario y notifican a su padre cuando ocurre algún evento. (Abramov, 2019)

Node

Node.js es un entorno de ejecución de Javascript del lado del servidor. Se basa en la implementación del motor de Google Chrome llamado V8, elaborado por Google (Node.js, 2021). V8 y Node están implementados principalmente en los lenguajes de programación C y C++, centrándose en el rendimiento y el bajo consumo de memoria. Pero, mientras que V8 soporta principalmente el lenguaje Javascript en el navegador (sobre todo en Google Chrome), Node tiene como objetivo soportar procesos de servidor procesos de servidor de larga duración.

Nest

Nest es un framework para crear aplicaciones del lado del servidor en la plataforma Node. Nest usa principalmente Typescript, pero también permite usar JavaScript. Nest provee capas de abstracción de librerías existentes para el desarrollo de aplicaciones de servidor en Node.js (Sabo, 2020). NestJS impone un estilo para desarrollar aplicaciones, es decir, te exige desarrollar funcionalidades de una forma específica. Esto es una gran ventaja, ya que este framework estimula el uso de buenas prácticas de programación, además, si tenemos esta cuenta que Nest está basado en TypeScript, las aplicaciones construidas con este framework, modulares, sostenibles y escalables (Sabo, 2020). NestJS mezcla componentes de programación funcional, programación orientada a objetos, y programación reactiva ("NestJS - A progressive Node.js framework", 2021).

Integración continua

La integración continua (CI) es un proceso que ayuda a que los desarrolladores fusionen los cambios que introducen en el código para incorporarlos a un repositorio compartido con frecuencia. Una vez que el desarrollador envía los cambios implementados al repositorio, se validan con la ejecución de distintos niveles de pruebas automatizadas (generalmente, pruebas de unidad e integración) para verificar que los cambios no hayan dañado la aplicación. Si una prueba automática detecta un conflicto entre el código nuevo y el actual, la CI facilita la resolución de esos errores con frecuencia y rapidez. (“¿Qué son la integración/distribución continuas (CI/CD)?”, 2021)

Entrega continua

El objetivo de la entrega continua es tener una base de código que pueda implementarse en un entorno de ejecución en cualquier momento, ya sea un ambiente de pruebas o de producción. La entrega continua depende de que la integración continua ya esté incorporada a su canal de desarrollo. (“¿Qué son la integración/distribución continuas (CI/CD)?”, 2021)

Jira

JIRA es un sistema de seguimiento de incidencias desarrollado por Atlassian Corporation a partir de 2002. Es el más comúnmente utilizado para el seguimiento de errores de software, pero gracias a sus avanzadas características de personalización, es muy adecuado para otros tipos de sistemas de tickets (órdenes de trabajo, mesas de ayuda, etc.) y para la gestión de proyectos, (Atlassian, 2021).

Metodologías ágiles

El objetivo de las metodologías ágiles surgen con el objetivo de ofrecer una alternativa a los procesos de desarrollo de software tradicionales, se inspiran en un conjunto de valores que fueron resultado de una reunión de expertos en el desarrollo de software, (que posteriormente fundaron *The Agile Alliance*) de estos valores se generan unos principios que diferencian un proceso ágil de uno tradicional (Canós, et al., 2003)**.** Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

(The Agile Manifesto, 2021)

SCRUM

Scrum es un marco de trabajo 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. En Scrum se realizan entregas parciales del producto mediante ciclos temporales cortos y de duración fija. El Product Owner (que representa a los clientes del proyecto) se encarga de priorizar las tareas de cada sprint (iteración de trabajo) usando como criterio el valor que aportan a los receptores del proyecto, 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 ("Qué es SCRUM", 2021). En la Figura 1 se observan los procesos que se llevan a cabo en la metodología scrum.

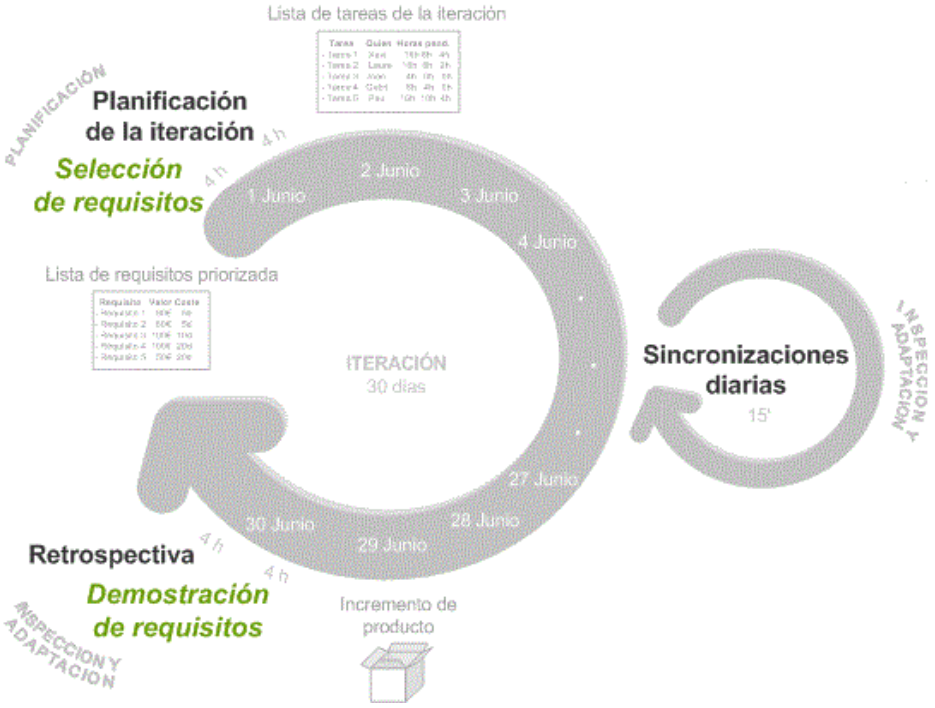

Figura 1. Metodología SCRUM

Metodología

Para el desarrollo del proyecto se utilizó una metodología compuesta de 5 fases, como se muestra en la Figura 2, estas fases son: Análisis, Diseño, Desarrollo, Pruebas y Entrega, las fases de análisis y diseño se llevaron a cabo una única vez, mientras que las fases de Desarrollo, Pruebas y Entrega se realizaron de forma iterativa, a continuación se detallan cada una de las fases.

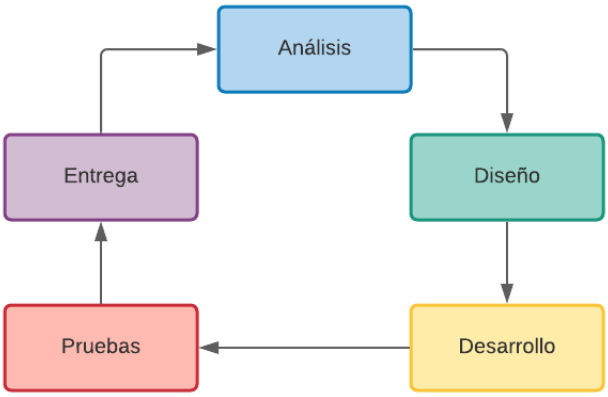

Figura 2. Metodología de desarrollo de software

En la fase de análisis se escucharon las necesidades del equipo de e-commerce de @PC, con base en las necesidades expresadas se construyeron los requerimientos de la aplicación utilizando historias de usuario.

Una vez obtenidos los requerimientos, estos son analizados con el objetivo de diseñar una solución que se adapte a las necesidades expuestas, en la primera iteración de la fase de diseño se eligieron las arquitecturas del sistema para aplicación de frontend y la aplicación de backend, además se determinaron las tecnologías a usar y se detallaron las funcionalidades a implementar, en las posteriores iteraciones de la fase de diseño, se analizaron las funcionalidades a implementar y se diseñaron las soluciones específicas para cada una de estas.

En la fase de desarrollo, se inició el proceso iterativo que caracteriza la metodología scrum, en esta fase se fueron tomando las descripciones de las funcionalidades a implementar y se construyeron historias de usuario que fueron desarrolladas de forma continua.

En la fase de pruebas, se realizaron pruebas manuales y automatizadas (pruebas unitarias), las pruebas manuales consisten en probar funcionalmente el software desde la perspectiva del usuario final, y las pruebas automatizadas se elaboraron con el fin de garantizar la calidad del software desde su implementación.

Por último, en la fase de entrega, se desplegaron las funcionalidades desarrolladas y testeadas en un ambiente pre-productivo mediante un flujo de integración continua y despliegue continuo que se encargaba de tomar el código fuente, compilarlo y ejecutarlo desde un servidor, las entregas se hicieron a medida que las tareas fueron finalizadas, esto con el fin de que el equipo de e-commerce diera una retroalimentación del software y propusiera mejoras en caso de ser necesario.

Resultados y análisis

En la Figura 3 se presenta de forma general el flujo que sigue el sistema, se puede observar que el sistema está compuesto por 2 componentes frontend y backend que se comunican mediante el protocolo HTTP para compartir la información. El componente

backend se encarga de la comunicación con las APIs de los ecommerce Mercadolibre y Buyruru VTEX, para esto también usa el protocolo HTTP.

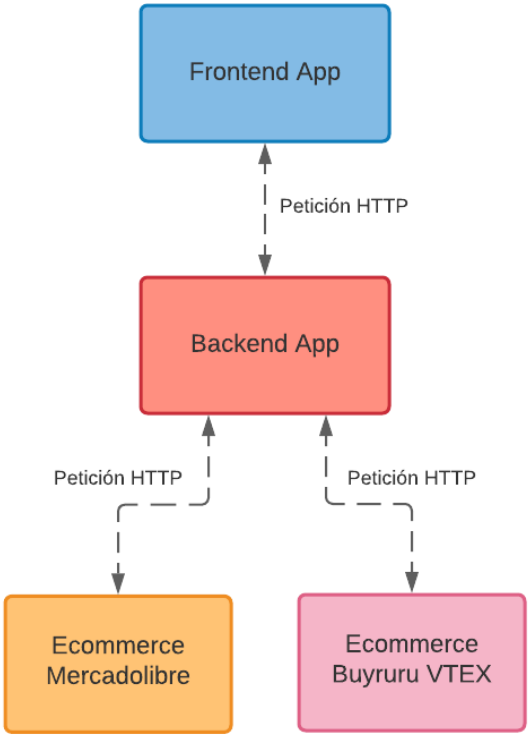

Figura 3. Flujo general del sistema

Para el desarrollo del frontend se utilizó la librería React, esta herramienta permite crear componentes y encapsular el comportamiento de cada uno de estos para que sigan el principio de responsabilidad única. React es una librería muy versátil ya que permite desarrollar aplicaciones frontend bajo múltiples patrones, para este caso en particular, se utilizó el patrón de componentes contenedores y de presentación para separar las responsabilidades de cada componente y que cada uno de estos tuviera una tarea específica. A continuación se detalla la responsabilidad de cada uno de los componentes.

- **App:** Este es el componente principal de cualquier aplicación construida con React, es el punto de entrada de la aplicación.
- **Navbar:** Se encarga de presentar un menú lateral para cambiar de pantalla (pensado para futuras implementaciones).
- **Productos:** Este es el componente inteligente que se encarga de realizar la comunicación con el backend para obtener la información de los productos
- **Buscador de Productos:** Se encarga de recibir una cadena de búsqueda ingresada por el usuario y enviar dicha cadena al componente padre.
- **Loading:** Se encarga de presentar un ícono de espera para notificar al usuario que la información está en proceso de carga.

- **Listado de Productos:** Se encarga de recibir la información de los productos y mostrarlos en pantalla.
- **Tipo de visualización:** Se encarga de notificar al componente padre la forma como se va visualizar la información, de acuerdo a la decisión del usuario.
- **Listado tipo tarjeta:** Se encarga de presentar los productos en modo tarjetas.
- **Listado tipo tabla:** Se encarga de presentar los productos en modo tabla.
- **Producto:** Se encarga de presentar la información de un producto.
- **Detalle Producto:** Se encarga de presentar información adicional de un producto.

En la Figura 4 se presenta la arquitectura del frontend y la jerarquía y relación entre los componentes.

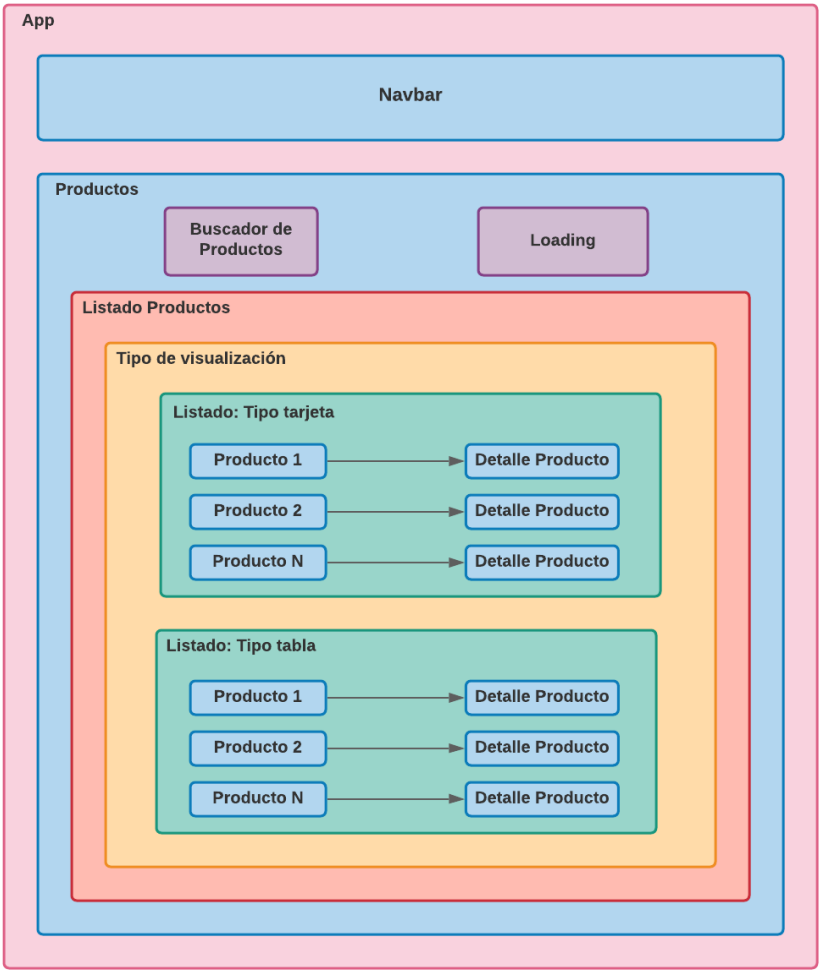

Figura 4. Arquitectura frontend del sistema

Como componente backend del sistema se construyó una REST API utilizando la herramienta Nest, la ventaja que entrega esta herramienta es su arquitectura robusta y pensada en la escalabilidad y mantenimiento del sistema. La API expuesta se compone de 3 recursos:

- Obtener todos los productos:
  GET /products

- Obtener todos los productos por cadena de búsqueda:
  GET /products?query=**{cadena}**

- Obtener un producto:
  GET /products/**{id}**?shop=**{tienda}**

El punto de entrada del backend es una petición HTTP realizada por el cliente, la capa controlador recibe esta petición y se encarga de decidir la acción a tomar, este componente se comunica con la capa de servicios que se encarga de ejecutar la lógica específica de acuerdo a la acción recibida, esta capa se encarga de comunicarse mediante peticiones HTTPs con cada una de las APIs de los e-commerce con los cuales está integrado el sistema. Debido a que los e-commerce exponen API diferentes e independientes, se realizó un servicio por cada e-commerce, esto permite que la arquitectura se adapte a futuras integraciones con otros ecommerce, independientemente de cómo sea su API. La Figura 5. presenta el flujo descrito anteriormente.

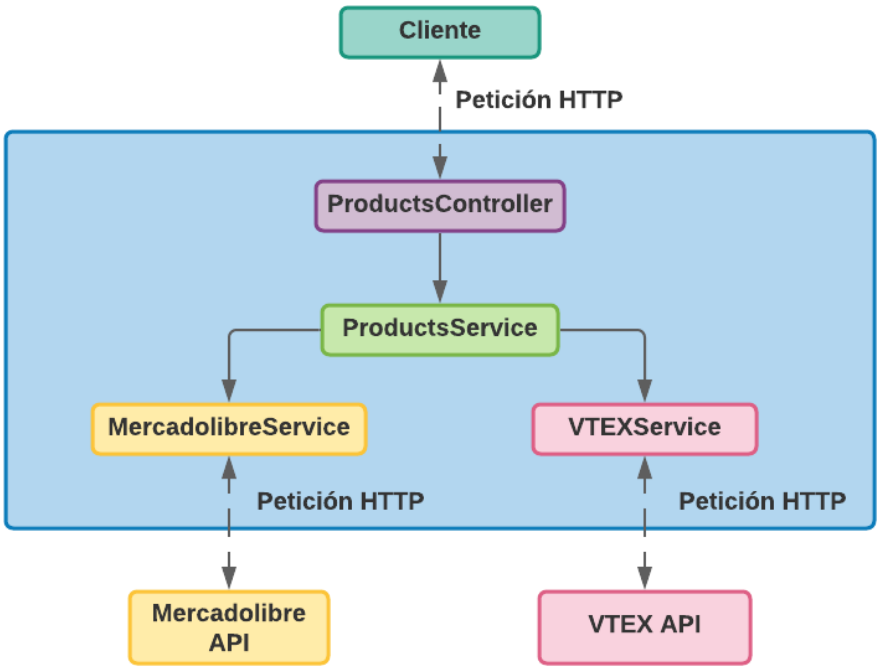

Figura 5. Arquitectura backend del sistema

En la Figura 6. se puede observar la vista de la aplicación, en la parte superior posee un campo que permite escribir una cadena de caracteres de búsqueda, al lado derecho hay un botón que permite enviar esta cadena a través de una petición hacia el backend, la petición responde un listado de productos que son almacenados en el estado de la aplicación, para posteriormente ser mostrados en forma de tarjetas. Estas tarjetas poseen algunos datos del producto: Imagen, nombre, precio unitario, cantidad disponible, tienda en la cual está publicado el producto y un enlace hacia la web de cada producto.

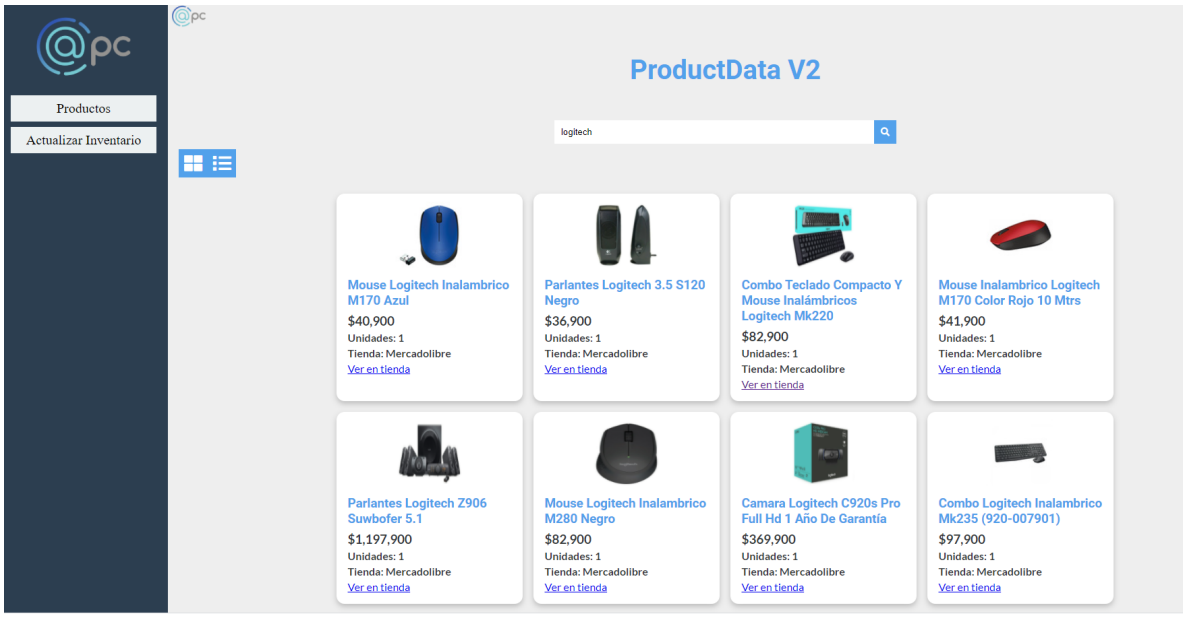

Figura 6. Vista de productos en forma de tarjetas

En la Figura 7. Se presenta la vista de productos en modo de tabla, esta vista se construyó con el objetivo de hacer más fácil la visualización de productos cuando no es necesario ver la imagen. Como se puede observar, cada fila de la tabla es un producto que contiene su identificador, nombre, precio unitario, cantidad disponible y una imagen que hace referencia a la tienda en la cual está publicado el respectivo producto.

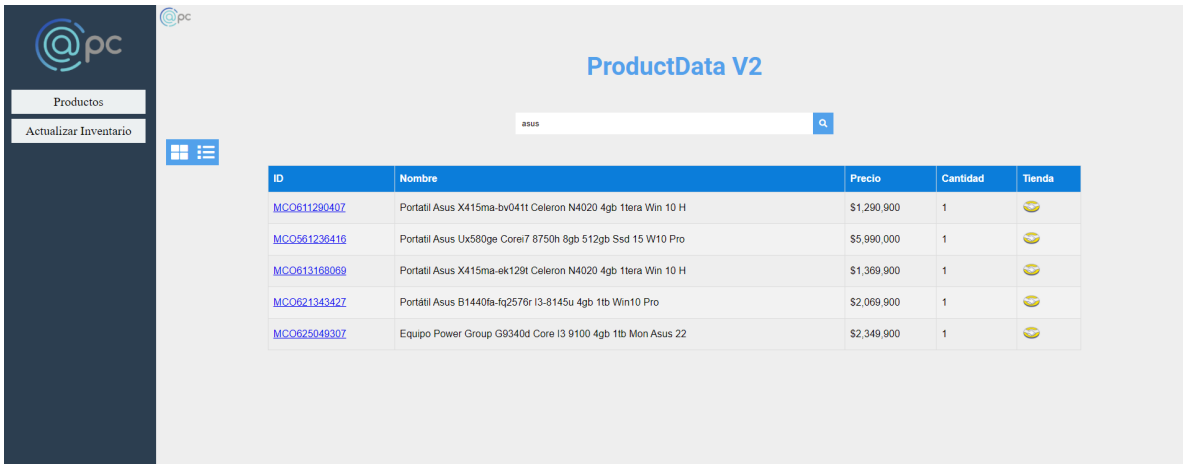

Figura 7. Vista de productos en forma de tabla

Durante todo el proceso de desarrollo del sistema se tuvieron varios impedimentos que no permitieron realizar la fase de entrega del sistema de forma fluida, el problema principal fue la integración con la infraestructura de la compañía, ya que por temas de seguridad se debía hacer uso de una VPN para conectarse con su servidor mediante el protocolo de comunicación PPTP, no obstante, se presentaron problemas con la configuración de la VPN debido al protocolo utilizado, esto impidió que el sistema fuera desplegado en el servidor de la compañía desde el computador utilizado.

Además, se realizó una propuesta de integración con el ERP de la compañía para darle mayor valor al sistema entregado, sin embargo, no fue posible llevar a cabo dicha integración por la conexión con la VPN y debido a una actualización aplicada al ERP de la compañía, las API cambiaron y no fue posible hallar documentación de cómo realizar la integración.

Conclusiones

El uso de la metodología ágil scrum permitió diseñar, implementar y probar las funcionalidades durante todo el proceso de desarrollo del sistema, esto favoreció el hallazgo de errores, oportunidades de mejora y nuevos requerimientos en todas la etapas.

Los frameworks y librerías utilizados para el desarrollo del sistema otorgan muchas herramientas para implementar funcionalidades fácil y rápidamente, el uso de React y su ecosistema de librerías facilitó las tareas de desarrollo del frontend. El framework Nest permitió construir la API REST bajo una arquitectura que favorece la escalabilidad y mantenibilidad de la aplicación, esto pensando en las nuevas implementaciones que pueda tener al sistema a futuro.

Como trabajo futuro se propone realizar la integración con el ERP de la compañía para obtener el inventario de la compañía, esta integración daría un gran valor agregado al sistema desarrollado ya que el ERP es la fuente principal de información de la compañía.

La universidad entrega muchos conceptos y herramientas útiles para realizar análisis y diseño de sistemas, en las primeras fases del proyecto los conceptos de levantamiento de requisitos fueron altamente útiles para entender las necesidades que se debían satisfacer con el sistema desarrollado, sin embargo, durante las posteriores fases de desarrollo del sistema, las herramientas y conceptos brindados por la universidad fueron poco utilizados ya que se encuentra desactualizados con las prácticas actuales en la industria de desarrollo de software, sería de mucha utilidad reforzar estos conceptos en los cursos del programa para mejorar la curva de aprendizaje de los estudiantes y prepararlos de mejor manera de cara a la industria de software actual.

Bibliografía

Marini, E. (2012). El modelo cliente/servidor.

Abdullah, H. M., & Zeki, A. M. (2014). Frontend and Backend Web Technologies in Social Networking Sites: Facebook as an Example. 2014 3rd International Conference on Advanced Computer Science Applications and Technologies. Published. https://doi.org/10.1109/acsat.2014.22

Filipova, O., & Vilão, R. (2018). Backend Development. Software Development From A to Z, 101–131. https://doi.org/10.1007/978-1-4842-3945-2_5

¿Qué es una API? (2021). RedHat. Recuperado 16 de Junio de 2021, de https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Li, L., & Chou, W. (2011). Design and Describe REST API without Violating REST: A Petri Net Based Approach. 2011 IEEE International Conference on Web Services. Published.

Fundamentos de JavaScript | MDN. (2021). Recuperado 8 de Julio de 2021, de https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

Bierman, G., Abadi, M., & Torgersen, M. (2014). Understanding TypeScript. ECOOP 2014 – Object-Oriented Programming.

Gackenheimer, C. (2015). Introduction to React (English Edition) (1st ed.). Apress.

Abramov, D. (2019, 17 febrero). Presentational and Container Components - Dan Abramov. Medium. https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing.

Node.js. (2021). Recuperado 16 de Junio de 2021, de https://nodejs.org/en/

Sabo, M. (2020). NestJS (Doctoral dissertation, Josip Juraj Strossmayer University of Osijek. Department of Mathematics. Chair of Applied Mathematics. Computer Science Research Group).

RedHat. (2021). ¿Qué son la integración/distribución continuas (CI/CD)? Recuperado 8 de julio de 2021, de https://www.redhat.com/es/topics/devops/what-is-ci-cd

Atlassian (2021). ¿Para qué se utiliza Jira Software? Recuperado 8 de mayo de 2021, de https://www.atlassian.com/es/software/jira/guides/use-cases/what-is-jira-used-for

NestJS - A progressive Node.js framework. (2021). Recuperado 8 de mayo de 2021, de https://nestjs.com/

Canós, J., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia, Valencia, 1-8.

The Agile Manifesto. (2021). Manifiesto por el Desarrollo Ágil de Software. Manifiesto por el Desarrollo Ágil de Software. Recuperado 10 de mayo de 2021, de https://agilemanifesto.org/iso/es/manifesto.html

Qué es SCRUM. (2021). Recuperado 8 de Julio de 2021, de https://proyectosagiles.org/que-es-scrum/