Polo del Conocimiento

Pol. Con. (Edición núm. 51) Vol. 5, Especial No 1
Noviembre Especial 2020, pp. 151-162
ISSN: 2550 - 682X
DOI: 10.23857/pc.v5i1.1884

Arquitectura de microservicios para compras en línea: caso de uso "ala orden"

Microservices architecture for online shopping: "to order" use case

Arquitetura de microsserviços para compras online: caso de uso "sob encomenda"

Paúl Quevedo-Avila [I]
paul.quevedo.15@psg.ucacue.edu.ec
https://orcid.org/0000-0002-9954-8054

Martin Geovanny Zhindón-Mora [II]
mgzhindonm@ucacue.edu.ec
https://orcid.org/0000-0003-4475-830X

Andrés Sebastián Quevedo-Sacoto [III]
asquevedos@ucacue.edu.ec
https://orcid.org/0000-0001-5585-0270

Correspondencia: paul.quevedo.15@psg.ucacue.edu.ec

Ciencias de la educación
Artículo de revisión

*__Recibido:__ 02 de septiembre de 2020 *__Aceptado:__ 07 de octubre 2020 * __Publicado:__ 06 de noviembre de 2020

I. Ingeniero de Sistemas, Jefatura de Posgrados. Universidad Católica de Cuenca, Cuenca, Ecuador.
II. Ingeniero de Sistemas, Jefe de Tecnologías de la Información, Universidad Católica de Cuenca, Cuenca, Ecuador.
III. Ingeniero de Sistemas, Docente de la Unidad Académica de Tecnologías de la Información y Comunicación (TIC), Jefatura de Posgrados, Universidad Católica de Cuenca, Cuenca, Ecuador.

Resumen

Este trabajo tiene el objetivo de presentar un patrón de diseño para la plataforma !ala orden¡, basado en un arquitectura de software de microservicios, mostrando su definición, bondades y características, con la caracterización del proceso, además, se presentan las capas empleadas en la implementación de esta arquitectura, tales como: capa del cliente y capa del servidor, describiendo cada uno de los microservicios implementados, se menciona la tecnología para la seguridad, acceso y persistencia de los datos, y, para finalizar, se detallan los resultados obtenidos, con el uso de la arquitectura de microservicios, en el diseño y desarrollo de esta plataforma.

Palabras claves: Microservicios; e-commerce; arquitectura; patrón de diseño; escalabilidad.

Abstract

This work has the objective of presenting a design pattern for the platform! To order, based on a microservices software architecture, showing its definition, benefits and characteristics, with the characterization of the process, in addition, the layers used in the implementation of this architecture, such as: client layer and server layer, describing each of the implemented microservices, mentioning the technology for data security, access and persistence, and, finally, the results obtained are detailed , with the use of microservices architecture, in the design and development of this platform.

Keywords: Microservices; e-commerce; architecture; Design pattern; scalability.

Resumo

Este trabalho tem o objetivo de apresentar um padrão de design para a plataforma! Encomendar, com base em uma arquitetura de software de microsserviços, mostrando sua definição, benefícios e características, com a caracterização do processo, bem como as camadas utilizadas na a implementação desta arquitetura, tais como: camada cliente e camada servidor, descrevendo cada um dos microsserviços implementados, mencionando a tecnologia para segurança, acesso e persistência de dados e, por fim, detalham-se os resultados obtidos , com a utilização de arquitetura de microsserviços, na concepção e desenvolvimento desta plataforma.

Palavras-chave: Microsserviços; comércio eletrônico; arquitetura; Padrão de design; escalabilidade.

Introducción

Debido a la situación que atraviesa el mundo ocasionada por la pandemia SARS-CoV-2, y el confinamiento obligatorio dispuesto por varios jefes de estado, existe un aumento del comercio por internet o comercio en línea denominado e-commerce[1]. Esta forma de comercio ayuda al consumidor a obtener el producto conforme a sus expectativas, intereses o gustos, lo que permite agilizar la entrega, el pago y control de los bienes o servicios adquiridos por estos medios digitales de transacciones [2].

El comercio electrónico ha estimulado que la mayoría de los mercados o negocios adopten este modelo, convirtiendo así al internet en un potencial canal de ventas, puesto que los consumidores pueden acceder y realizar sus pedidos en línea desde un dispositivo electrónico inteligente[3] [4].

Según la Cámara de Comercio del Ecuador [4], tras la llegada del Covid-9 se ha producido un incremento en la frecuencia de compra en línea, como se muestra en la Figura 1.

Figura 1: Tomado de: Transacciones electrónicas en Ecuador durante el Covid-19. En el eje y observamos el porcentaje de incremento de compras en línea.

Fuente: Elaboración propia.

El incremento de ventas o de transacciones electrónicas también ha traído nuevos emprendimientos, los mismos que buscan automatizar y facilitar este tipo de transacciones en línea [5].

Las aplicaciones monolíticas han resultado en varios casos de uso a lo largo del mundo, pero con el paso del tiempo se ha generado problemas al escalar y mejorar su rendimiento, puesto que los cambios constantes en el modelo de negocio originan que se tenga que modificar toda la aplicación, haciéndola más grande y compleja de mantener [6].

Para resolver los problemas que se ocasionan con las arquitecturas tradicionales, se han propuesto modelos mucho más estables y eficientes, como las arquitecturas basadas en microservicios, que plantean la construcción de conjuntos de pequeños servicios independientes, , que se comunican entre sí a través de peticiones HTTP y se ejecutan de forma autónoma, esto permite que sean escalables y fáciles de mantener, y que puedan ser escritos en diferentes lenguajes de programación y por diferentes equipos de desarrollo[7].

Los microservicios se han convertido en una opción de estilo arquitectónico dominante en la industria del desarrollo de software, presentándose como un patrón de arquitectura con énfasis en dividir el sistema en servicios pequeños y livianos que se construyen expresamente para realizar una función comercial muy cohesiva [8], considerándose como un refinamiento y simplificación de la arquitectura orientada a servicios (SOA) [9].

Existen numerosos y diversos principios que presentan este patrón de diseño, entre los que se destacan.

Resilientes

Los microservicios son resistentes y tolerantes a fallos y se pueden solventar rápidamente [10], si algo falla en un microservicio y no están en cascada se puede aislar el problema y el aplicativo continúa operativo, en una arquitectura monolítica, la falla de un componente puede hacer fallar a todo el sistema.

Escalables

Al escalar sistemas monolíticos se requiere escalar la aplicación entera [11], demanda una gran inversión en hardware y complejidad en el código. La capacidad de actualizar y escalar los servicios de forma aislada en un patrón de diseño basado en microservicios es más sencillo [12], se pueden implementar, monitorear y escalar de forma independiente por definición. El ser capaz de escalar horizontalmente un microservicio es fundamental para mejorar el rendimiento del software [13]. Cada microservicio puede implementarse en un servidor con diferentes capacidades de hardware, y pueden escribirse en el lenguaje de programación más apropiado [14]. Si hay un congestionamiento en un microservicio se puede transportar y ejecutar en varios hosts en paralelo [10].

Alta disponibilidad

La alta disponibilidad se logra mediante la capacidad de los microservicios de replicarse y distribuirse entre centros de datos y distancias geográficas, lo que les permite distribuir la carga

y hacer frente a hardware congestionado y con fallas. Otro aspecto relevante se refiere a la actualización y evolución del sistema: si bien la actualización de una aplicación monolítica requiere detenerla y volver a implementarla, esto incurre en tiempos de inactividad posiblemente prolongada, la replicabilidad y la independencia permiten que los microservicios resuelvan el problema [15].

En este contexto se planteó la necesidad de desarrollar e implementar una plataforma de comercio electrónico que permita a los pequeños, medianos y grandes comercios ofrecer sus productos o servicios, y para esto resulta indispensable construir esta solución bajo una arquitectura de microservicios, esto debido a que nos encontramos en un entorno volátil, incierto, complejo y ambiguo[16], lo que obliga que la plataforma y sus componentes brinden la oportunidad de que puedan ser ajustadas acorde a esta dinámica.

Metodología

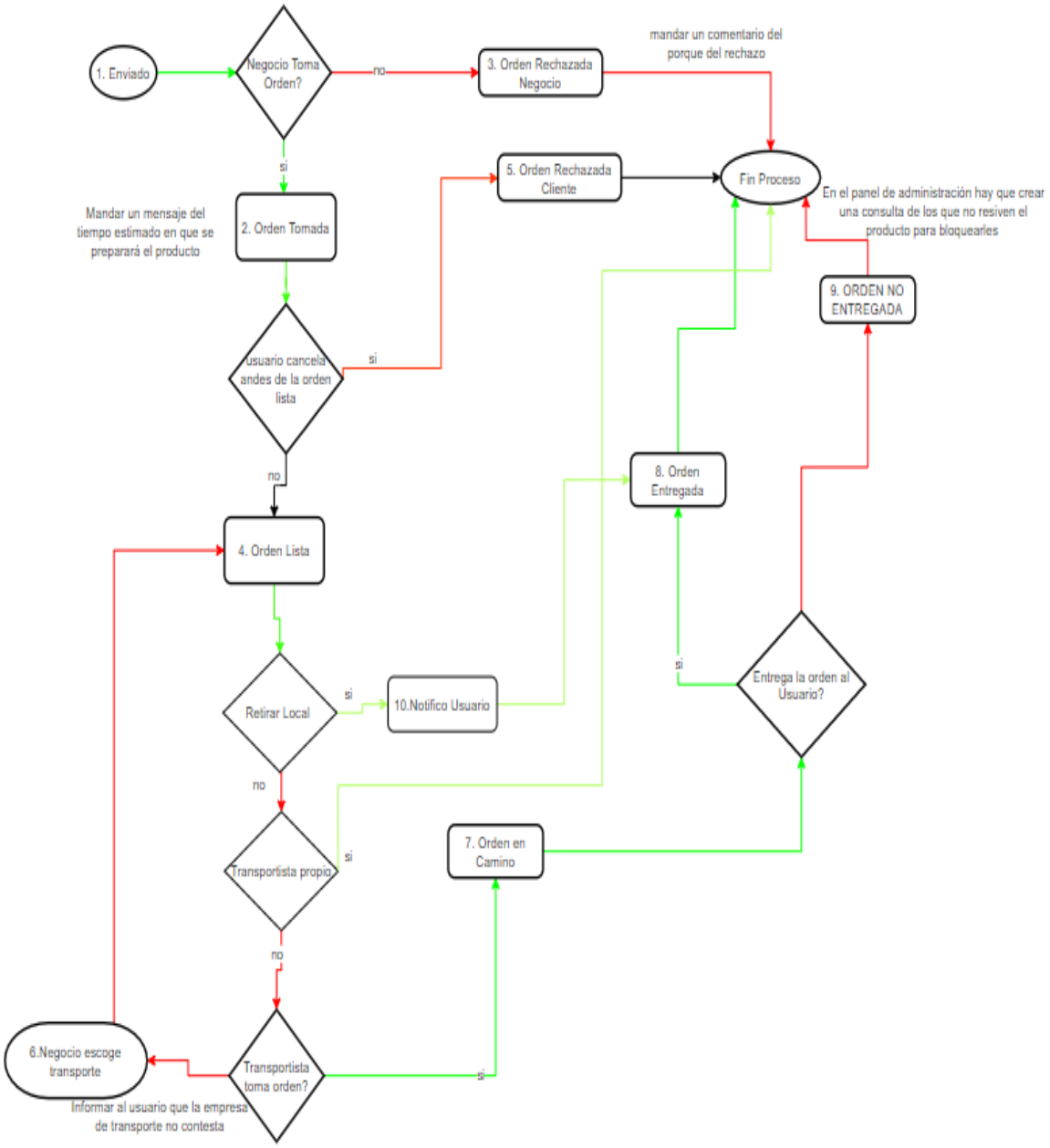
Caracterización del proceso

En esta fase se utilizó un método sistémico, lo que permitió diagnosticar y evaluar lo que conforma el modelo, necesidades y requerimientos, así se logró establecer los respectivos objetivos (relaciones), por ejemplo, para los comerciantes es ofrecer sus productos/servicios y mejorar sus ventas, mientras que para la empresa de repartos (delivery) se determinó su proceso de negocio; finalmente, para los usuarios es tener una mejor experiencia al momento de realizar una compra.

Para determinar las necesidades de las partes involucradas que implican este modelo, se trazó de manera primaria el uso de fuentes, como: Observación (experiencia personal del autor), uso de entrevistas a los comerciantes, empresas de reparto, expertos en proyectos y usuarios en general.

Con la información recolectada, se estableció presentar un diagrama de flujo, como el mostrado en la figura 2, para luego elaborar una guía para el direccionamiento del proyecto.

Figura 2: Diagrama de flujo de la plataforma “ala orden”.



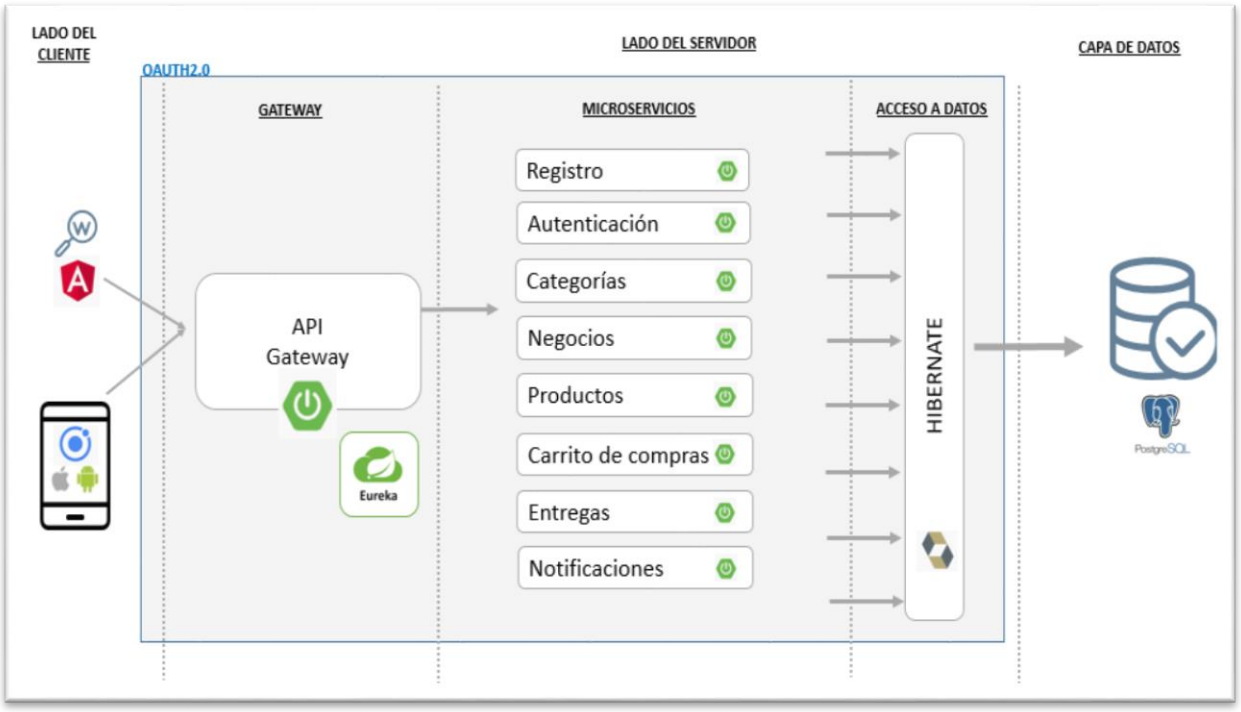
Fuente: Elaboración propia.

Arquitectura

La arquitectura de la plataforma ¡ala orden!, cuenta con un patrón de diseño basado en microservicios. En la figura 3 se muestra el diseño que conforma la arquitectura de la plataforma ¡ala orden!

Pol. Con. (Edición núm.51) Vol. 5, Especial No 1, noviembre 2020, pp. 151-162, ISSN: 2550 - 682X

Figura 3: Diagrama del diseño de la arquitectura del sistema "ala orden".



Fuente: Elaboración propia.

Lado del cliente

Esta capa, hace referencia frontend la misma que cuenta con los siguientes framework's de desarrollo angular para desarrollo web, ionic para el desarrollo de aplicaciones móviles híbridas , bootstrap y angular material para el diseño de interfaces responsivos adaptables a los dispositivos inteligentes basados en el estándar html5 y css3 , y, por último, la integración de openstreetmap como proveedor de mapas en línea.

Lado del servidor Backend

En el lado del servidor, la arquitectura del sistema propuesto está dividida en 4 secciones que se describen a continuación:

- Seguridad

Se encarga de gestionar los roles y permisos de acceso a la aplicación. Este proceso es ejecutado por el sistema en dos pasos: un proceso de registro donde se identifica al usuario y un proceso de verificación cada vez que el usuario inicia una nueva sesión en la aplicación otorgándole un token, según los roles asignado se tiene acceso a los microservicios que son mapeados de acuerdo a perfiles pre establecidos como: administrador, publicador, transporte, transportista y usuario en general.

El estándar de seguridad utilizado en la aplicación es Open Authorization , que permite flujos de autorización para las API implementadas en los microservicios.

- Gateway

En esta sección de la Arquitectura, se implementa un proceso de puerta de enlace ente los requerimientos de consumo del cliente y por otra parte el servidor. Eureka hace el descubrimiento y balanceo de carga de los microservicios de la aplicación, transformando la petición según lo requerido por el cliente.

- Microservicios

En la Tabla 1 se detalla cada uno de los microservicios implementados en la lógica de negocios mediante el framework Spring Boot, cada uno de estos se definen independientes entre sí.

Tabla 1: Detalle de los microservicios implementados.

Microservicio	Objetivo
Registro	Permite a los usuarios realizar el registro de su información en la plataforma.
Autenticación	Contiene la información del usuario y roles, la autenticación en el sistema se realiza mediante el ingreso de su usuario y contraseña; Aquí se utiliza la herramienta Oauth2.0 el cual brinda token al cliente que permitirá el acceso a los microservicios según su rol.
Categorías	Permite crear un CRUD[1] de las categorías (rol administrador).
Negocios	Permite administrar la información de los comercios (rol publicador).
Productos	Este microservicio permite realizar un CRUD de los productos de cada negocio (rol publicador).
Carrito de compras	Es uno de los principales del sistema, ya que es el encargado de la gestión de pedidos en línea por parte de los consumidores, notificando así a los negocios y a las empresas de delivery por cada solicitud de compra en la plataforma.

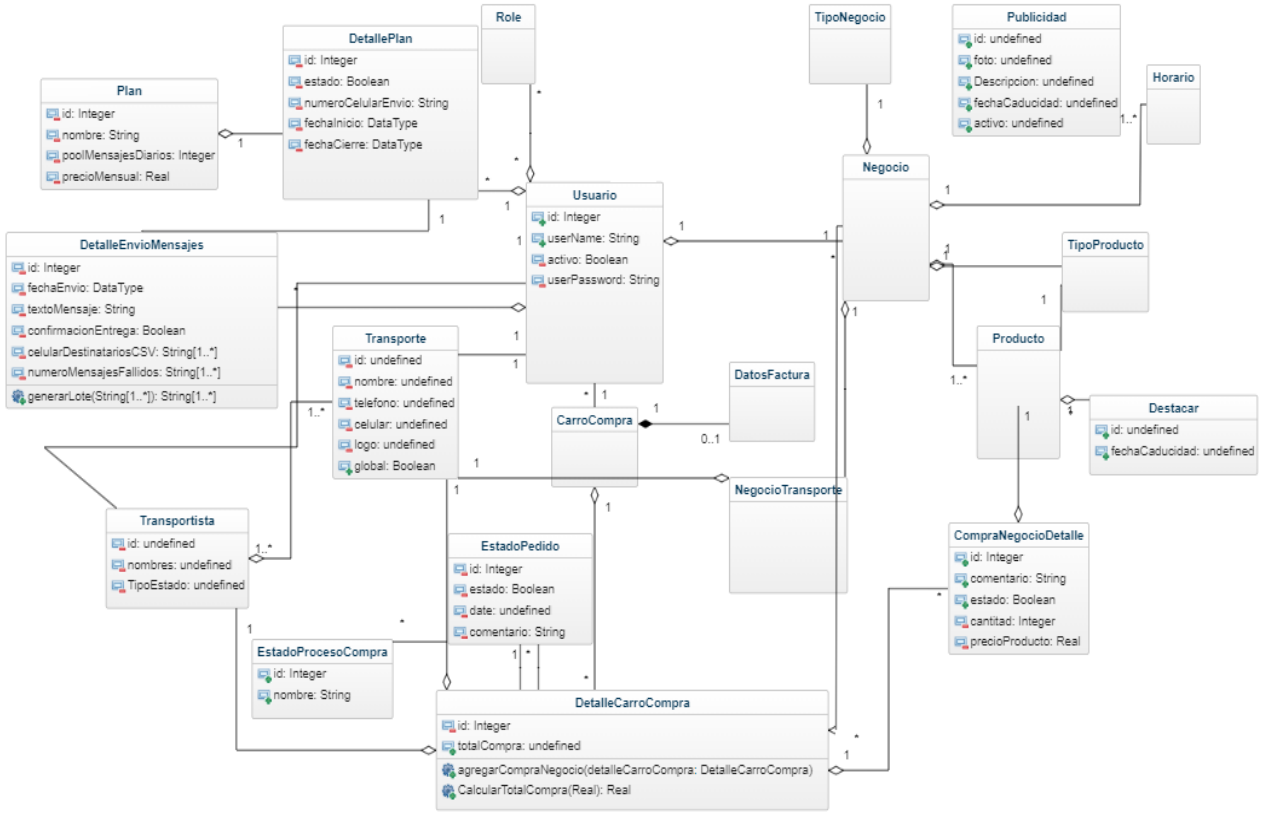
[1] Es el acrónimo de "Crear, Leer, Actualizar y Borrar", que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software

Entregas	Permite la gestión (entregas a domicilio, asignación de un transporte) por cada solicitud de compra (rol transporte).
Notificaciones	Se encarga de realizar las notificaciones (notificación de compra, estado del pedido y promociones) a los usuarios, este servicio cuenta con una integración con las API's de chat-api[2].

- Acceso a datos

Esta capa se implementó a través del ORM como se muestra en la figura 4, con el framework hiberbate, que permite la persistencia de los datos en una base de datos relacional y al mismo tiempo permite emplearla programación orientada a objetos (POO) en la aplicación.

Figura 4: Mapeo objeto relacional de la plataforma ¡ala orden!



Fuente: Elaboración propia.

Resultados y discusión

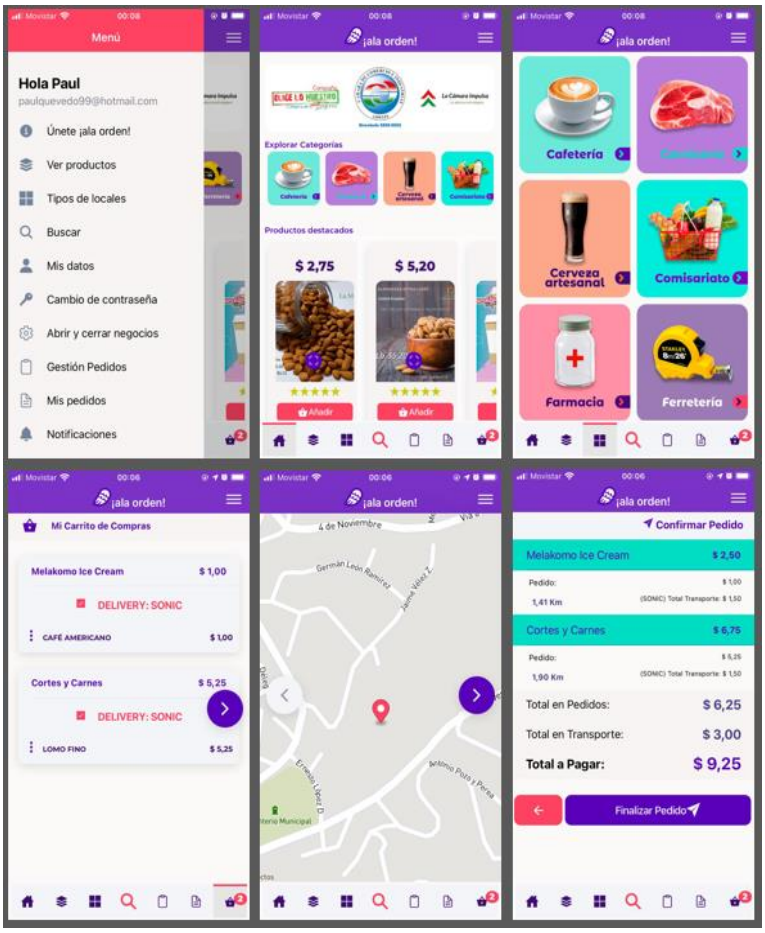
[2] https://chat-api.com/

La implementación de un patrón arquitectónico de microservicios permitió realizar cambios en cualquier microservicio, debido a fallos o por cambios requeridos, siendo estos transparente para los clientes que consume dichos servicios. Esto permite un escalamiento rápido y eficiente, también permite implementar mejoras y añadir nuevas funcionalidades a la aplicación de forma ágil y sin interrupciones.

Se ha conseguido en el caso de que se presente problemas estos se puedan tratar de manera aislada, debido a que si se afecta un microservicio, los demás servicios de la aplicación pueden continuar con el control de las solicitudes lo cual posibilita escalar horizontalmente.

La figura 5, muestra las interfaces de la aplicación móvil ¡ala orden!, usando ionic como framework y como microservicios como arquitectura de software.

Figura 5: Interfaces móvil



Fuente: aplicación móvil ¡ala orden!

Esta plataforma se encuentra disponible en la web ingresando a www.alaorden.io, y para los dispositivos móviles mediante sus tiendas oficiales como google play y app store, denominada !ala orden¡.

Conclusiones

Se ha investigado e implementado un modelo arquitectónico basado en microservicios para la plataforma ¡ala orden!, que evidenció la utilidad de dicho patrón de diseño, ya que permitió adaptar a la plataforma múltiples cambios que se presentaron al momento del desarrollo, implementación y primera fase de puesta en operación.

La independencia de cada uno de los microservicios posibilitó un desarrollo, mantenimiento y despliegue más eficiente, ya que el enfoque esta puesto en esta actividad y en ese contexto en específico.

Este patrón de diseño se presenta como una alternativa moderna que brinda flexibilidad, escalabilidad e independencia a momento de diseñar y construir software.

En este trabajo se ha evidenciado la razón por la que se ha visto un incremento de nuevas herramientas y frameworks, para el diseño, construcción e implementación de aplicativos basado en microservicios debido a las bondades y beneficios que brinda esta arquitectura de software.

Referencias

1. A. Bhatti, H. Akram, H. M. Basit, A. U. Khan, and S. M. Raza, “E-commerce trends during COVID-19 Pandemic,” vol. 13, no. 2, pp. 1449–1452, 2020.
2. L. Periolo et al., “Marketing + internet = e-commerce : oportunidades,” Cuad. Econ. y Dir. la Empres., vol. 13, no. 16, p. 118, 2013, doi: 10.1016/s1138-5758(07)70081-6.
3. J. Thönes, “Microservices,” IEEE Softw., vol. 32, no. 1, 2015, doi: 10.1109/MS.2015.11.
4. S. Dakduk, R. Dicarlo, L. Ottati, and A. Portalanza, “Transacciones electrónicas en Ecuador durante el Covid- 19.,” Cámara Ecuatoriana Comer. Electrónico, vol. 1du, no. 1, p. 16, 2020.
5. R. Marcelo, M. Hinojosa, Í. Omar, M. Pazmiño, H. Patricio, and D. Solís, “Emprendimiento y marketing durante el aislamiento social por la pandemia Entrepreneurship and marketing during social isolation due to the pandemic,” pp. 0–1.
6. B. Familiar and B. Familiar, Microservice Architecture. 2015.

7. D. Barrios Contreras, “Arquitectura de Microservicios,” Tecnol. Investig. y Acad., vol. 6, no. 1, pp. 36–46, 2018.
8. “Microservices Guide.” https://martinfowler.com/microservices/ (accessed Oct. 01, 2020).
9. M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar, and M. Steinder, “Performance evaluation of microservices architectures using containers,” Proc. - 2015 IEEE 14th Int. Symp. Netw. Comput. Appl. NCA 2015, pp. 27–34, 2016, doi: 10.1109/NCA.2015.49.
10. D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation,” IEEE Cloud Comput., vol. 4, no. 5, pp. 22–32, 2017, doi: 10.1109/MCC.2017.4250931.
11. D. Maya, E., y López, “Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web,” Septima Conf. Dir. Tecnol. Inf. 2017, no. July, p. 12, 2017, [Online]. Available: http://dspace.redclara.net:8080/bitstream/10786/1277/1/93 Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.pdf.
12. D. Namiot and M. Sneps-Sneppe, “‘On microservices Architecture’ International journal of open information technologies.,” Int. J. Open Inf. Technol., vol. 2, no. 9, pp. 24–27, 2014, [Online]. Available: http://injoit.org/index.php/j1/article/view/139.
13. A. Brogi, A. Canciani, D. Neri, L. Rinaldi, and J. Soldani, “Towards a reference dataset of microservice-based applications,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10729 LNCS, pp. 219–229, 2018, doi: 10.1007/978-3-319-74781-1_16.
14. “Scaling Microservices at Gilt with Scala, Docker and AWS.” https://www.infoq.com/news/2015/04/scaling-microservices-gilt/ (accessed Oct. 01, 2020).
15. A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, “From Monolithic to Microservices: An Experience Report from the Banking Domain,” IEEE Softw., vol. 35, no. 3, pp. 50–55, 2018, doi: 10.1109/MS.2018.2141026.
16. L. James and N. Bennett, “What VUCA Really,” Harv. Bus. Rev., vol. 92, no. February, p. 2014, 2014