Análisis y desarrollo de “Endava Marketplace”: una aplicación para la compra y venta de productos usados entre empleados

Ian Marcos Ortiz Duque

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesor
Javier Fernando Botia Valderrama, Doctor en Ingeniería Electrónica

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
Medellín
2023

Dedicatoria

A mi familia, cuyo incondicional apoyo me ha permitido perseverar a lo largo de mi carrera.

Agradecimientos

Mi gratitud se extiende a Endava por brindarme la oportunidad única de realizar mi práctica profesional, así como a todo el equipó que hizo posible el desarrollo de este proyecto. Deseo también agradecer a mis mentores para este proyecto; a Carlos Alzate, su experiencia y disponibilidad han sido invaluables para el aprendizaje y continua mejora a lo largo del proceso, y a Javier Botia, su guianza y recomendaciones han sido fundamentales para el correcto desarrollo de esta práctica. Finalmente, agradezco a la Universidad de Antioquia por brindarme la oportunidad de combinar teoría y práctica a través de esta experiencia.

Tabla de contenido

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

MVP	Minimum viable product
CTO	Chief technology officer
API	Application programming interface
PO	Product owner

RESUMEN

La empresa Endava utiliza el software "Microsoft Teams" como plataforma de comunicación. En esta plataforma, se ha creado un canal llamado "Avisos Clasificados" donde los empleados, conocidos como Endavans, pueden publicar anuncios para vender productos que ya no utilizan. Sin embargo, Teams no está diseñado específicamente para la compra y venta de productos, lo que puede afectar la experiencia de los usuarios. Ante esta situación, se propuso el desarrollo de una nueva plataforma que soluciona estas deficiencias y ofrezca una experiencia más satisfactoria para los Endavans que desean vender sus productos usados.La nueva plataforma se desarrolló utilizando tecnologías ya empleadas por la empresa y siguiendo la metodología SCRUM. Todo esto se llevó a cabo con el objetivo de garantizar la entrega de un producto funcional y escalable. Además del desarrollo de la aplicación, se buscó que los practicantes que participan en el proyecto experimentaran situaciones comunes en proyectos profesionales, como cambios en el alcance y los plazos del proyecto, entre otras circunstancias. Con esto, se logra que los practicantes se adapten no solo al agilismo y a las fases de un proyecto, sino también al ritmo de ejecución del proyecto, mejorando habilidades blandas como la adaptabilidad, la autonomía, la comunicación, la gestión del tiempo, el liderazgo y el trabajo en equipo.

Palabras clave **— Marketplace, aplicación web, desarrollo, frontend**

ABSTRACT

The Endava company uses the "Microsoft Teams" application as its communication platform. In the application, a channel called "Avisos clasificados" has been opened to allow employees, referred to as Endavans, to make publications selling products they no longer use. However, Teams is not a platform designed to sell or purchase products, and because of this, the user experience may not be good. Given this, it has been proposed to develop a new platform that solves the aforementioned shortcomings and also provides a pleasant user experience for the Endavans that seek to sell their used products. In order to guarantee the delivery of a functional and scalable product, the new platform was implemented using technologies already utilized by Endava and following the SCRUM methodology. In addition to the application development, there was the goal of allowing the interns participating in the project to experience common situations found in professional projects, such as changes to a project's scope and deadlines. This makes the interns adapt not only to agile methodologies and project stages but also to the speed at which a project must be carried out, improving soft skills such as adaptability, autonomy, effective communication, time management, leadership, and teamwork.

Keywords **— Marketplace, web application, development, frontend**

I. INTRODUCCIÓN

La plataforma de comunicación interna de Endava es Microsoft Teams. Dentro de esta se ha habilitado un canal en el cual los empleados de la compañía, a los que se les refiere como Endavans, pueden listar productos usados que desean vender. Este canal tiene el beneficio inmediato de la confiabilidad. Los vendedores tienen la certeza de que el comprador es de la misma compañía y probablemente alguien profesional y confiable. Sin embargo, Microsoft Teams no está diseñado para compras y ventas. Por ejemplo, no hay una manera de saber si un producto fue vendido, y tampoco hay una presentación estandarizada de los productos y sus características. Adicionalmente, el canal continúa llenándose de publicaciones, por lo que algunos productos pueden quedar perdidos sin que nadie los vea.

Ante estas desventajas, se planteó, como ejercicio de aprendizaje, crear una plataforma que brinde una mejor experiencia al usuario, y garantice una interfaz de usuario limpia para facilitar el proceso de compra y publicación de productos. Esta nueva plataforma, llamada “Endava Marketplace”, es una aplicación web que permite crear y administrar ofertas de productos, comprar productos, actualizar automáticamente el estado de una oferta cuando se complete la venta, listar únicamente los productos disponibles, filtrar productos por categorías, buscar productos por nombre, listar los productos comprados y vendidos, entre otras funcionalidades propias de una plataforma de compras y ventas.

Para el desarrollo de la aplicación, se empleó la metodología SCRUM, lo que permitió a los practicantes aprender sobre metodologías ágiles, y de las ceremonias comunes de un proyecto, como son las dailies, sprint plannings, story refinements, entre otros. Siguiendo la metodología, en primera instancia se logró desarrollar un MVP, posterior a esto, se siguieron implementado las demás funcionalidades planteadas para el producto final. En síntesis, se creó un producto que evolucionó gradualmente, y que en cada paso del proyecto, fue creando cada vez más valor para el cliente. Ahora bien, al ser un ejercicio de aprendizaje, no hay planes para la publicación y uso del aplicativo (Esto se definió en mutuo acuerdo con el CTO de la empresa).

II. OBJETIVOS

A. Objetivo general

Desarrollar una plataforma que mejore la experiencia de los Endavans para facilitar la compra y venta de productos usados, siguiendo metodologías y estrategías usadas en la industria.

B. Objetivos específicos

- Analizar las historias de usuario y las tareas necesarias que permitan la creación del servicio.
- Diseñar la arquitectura de la aplicación, seleccionando aquellas tecnologías que mejor se adapten a la solución planteada.
- Desarrollar la aplicación de manera incremental, empleando metodologías ágiles que permitan tener una solución funcional en intervalos definidos de tiempo.
- Validar el producto (Minimum Viable Product) con sus funcionalidades mínimas para garantizar un valor agregado a la empresa.

III. MARCO TEÓRICO

El problema que se intenta resolver con Endava Marketplace surge porque la plataforma de comunicación Teams no ha sido diseñada para albergar un marketplace, es decir, un espacio donde las personas presentan productos que desean vender, y otras los compran.

La solución más accesible para los empleados de Endava es una aplicación web específicamente diseñada para las interacciones propias de un marketplace. Esto se debe a que las aplicaciones web son programas cuyo objetivo principal es prescindir de software o configuraciones especiales del lado del usuario [1]. Simplemente se necesita cualquier navegador web, que son herramientas de uso diario. Estas aplicaciones se dividen, en muchos casos, en dos partes: el backend, que es el código encargado de manejar e implementar servicios regidos por la lógica del negocio, y el frontend, que es el código encargado de las interfaces gráficas mediante las cuales el usuario accede a los servicios proporcionados por el backend.

Para implementar tanto el backend como el frontend, se utilizaron frameworks con el fin de facilitar y agilizar el proceso de desarrollo. Los frameworks son soluciones de alto nivel para la reutilización de segmentos de software, que permiten el uso de funciones comúnmente empleadas y de lógica genérica asociada a las aplicaciones. A su vez, garantizan un mayor nivel de calidad del producto final, ya que gran parte del código proporcionado ya ha sido probado [2]. La facilidad y agilidad en el desarrollo provienen del conjunto de clases e interfaces reutilizables expuestas por el framework, las cuales sirven para realizar las tareas más comunes de las aplicaciones. Por ejemplo, a menudo, un servidor necesita habilitar una URL para que una o varias aplicaciones accedan a sus servicios. Normalmente, esto tomaría bastante código para ser implementado desde cero, pero en este caso, se puede usar un framework que haya elaborado toda la lógica e implementación, y presente a los desarrolladores un simple método que realice todo el trabajo. De esta manera, se logra reducir la cantidad de código que se va a escribir.

Sin embargo, para desarrollar el proyecto, no es suficiente con solo escribir código y utilizar frameworks. En todos los proyectos de Endava, se toman medidas para asegurar y mejorar la calidad del código. Es por esto que a lo largo del desarrollo de este proyecto, es vital realizar pruebas unitarias de código tanto para el frontend como para el backend. Las pruebas de software, en particular las pruebas unitarias, son consideradas el pilar del aseguramiento de la calidad del código, ya que ayudan a comprender el diseño del programa, los cambios en sus

componentes, y proporcionan retroalimentación visual que facilita la documentación y la reutilización del código [3].

Es común que se piense que la cantidad de pruebas es directamente proporcional a la calidad del código e inversamente proporcional a la cantidad de defectos en un programa [4]. Por eso, existe el concepto de cobertura de las pruebas, que representa el porcentaje del código cubierto por las pruebas. Este concepto es ampliamente utilizado en la industria, como se evidencia en el hecho de que es habitual buscar una cobertura mínima del 85% en los proyectos de software [4]. Con esto en mente, la cobertura del código fue el eje principal para trabajar y mejorar la calidad del producto final.

IV. METODOLOGÍA

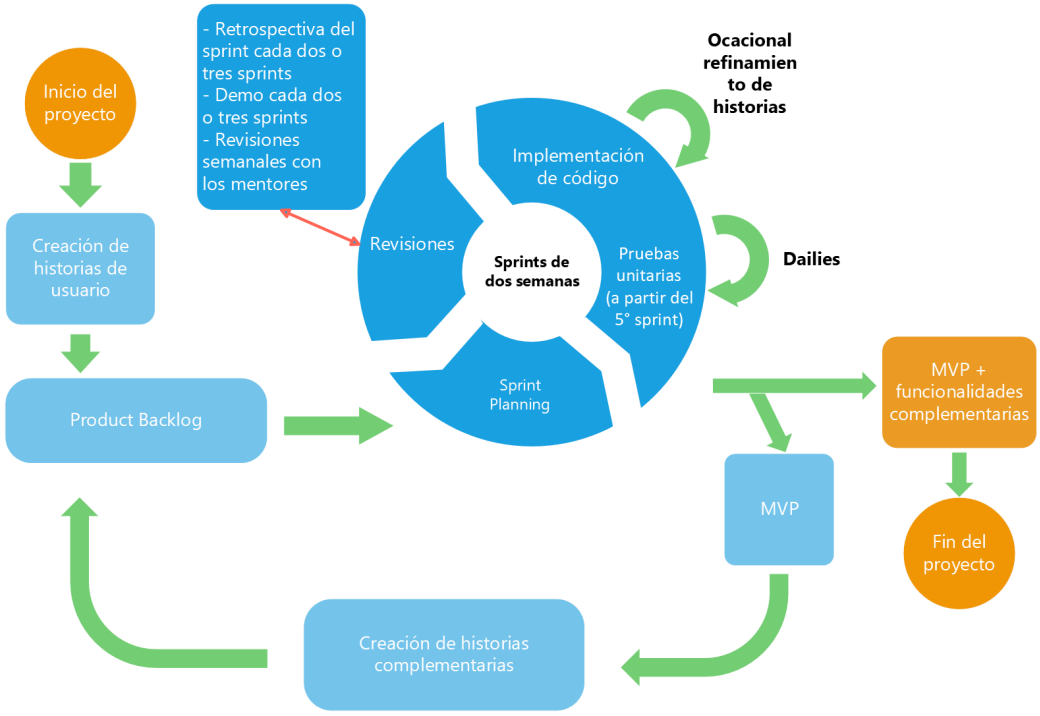

Fig. 1. Metodología SCRUM del proyecto

El proyecto se desarrolló utilizando la metodología ágil SCRUM, que es la metodología empleada para la gran mayoría de los proyectos de la empresa. Esta metodología se centra en dividir el trabajo en historias de usuarios, que son tareas y funcionalidades que, al ser implementadas, pueden generar valor. La mayoría de las historias de usuarios se definieron en las fases iniciales del proyecto; sin embargo, a lo largo del mismo, se agregaron nuevas historias y se modificaron y refinaron otras. Las historias se desarrollan en períodos de tiempo llamados sprints, cada uno con una duración de dos semanas. Al final de cada sprint, se logra añadir más funcionalidades al producto, brindando así una experiencia más completa y amena para el usuario.

La herramienta que facilitó el trabajo con la metodología SCRUM es Jira, una plataforma web que permite crear historias de usuario, añadirlas a los sprints y administrar dichos sprints. Al comienzo del proyecto, en un documento aparte, se definieron el conjunto de historias que abarcaban las funcionalidades del “producto mínimo viable”, o MVP por sus siglas en inglés. Una vez definidas, se crearon las historias en Jira usando el formato “Como [tipo de usuario], quiero…, para qué…”, y se definieron los criterios de aceptación de cada historia usando el

formato Gherkin. Posteriormente, para cada historia se estimó cuánto tiempo y esfuerzo se necesitaría para completarla, y, una vez estimadas, se dividieron en subtareas y se asignaron a alguien del equipo. Finalmente, se seleccionaron algunas historias para ser añadidas al primer sprint, teniendo en cuenta que el tiempo estimado para realizarlas no podía superar las dos semanas del sprint. Una vez iniciado el primer sprint, el proceso de crear historias se volvió mucho más flexible. A medida que se desarrollaba la aplicación, surgieron nuevas tareas y necesidades; quien identificara una era responsable de crear la respectiva historia o tarea en Jira y compartirla con el equipo.

Durante todo el proceso, hubo cuatro ceremonias adicionales que contribuyeron al buen desarrollo del proyecto. En primera instancia, están los *sprint plannings*, que son reuniones al final de cada sprint en las que se seleccionan y distribuyen las historias del siguiente sprint. Asimismo, se estiman las tareas nuevas, y ocasionalmente se estiman historias ya creadas. En segundo lugar, estuvieron los refinamientos de historias, que ocurrieron ocasionalmente cuando el equipo sentía que las próximas historias a trabajar no eran claras, o cuando alguien sentía que era necesario esclarecer el rumbo del proyecto. El tercer tipo de ceremonia realizada fueron los *sprint retrospectives*. Realizados cada dos o tres sprints, buscaban hacer una retrospectiva que identificara qué se hizo bien, qué aspectos del trabajo y del equipo podrían mejorarse, y estrategias para implementar estas mejoras. Por último, estuvieron las *demos*, que son presentaciones en las que se demuestra el estado y funcionamiento del producto. Estas presentaciones fueron las que más impacto tuvieron en el proyecto, porque fueron en estos espacios donde el *product owner*, que es la persona que tiene una visión clara del producto final, realizó su retroalimentación. Gracias a sus contribuciones, y a las de los mentores de los practicantes, se pudo asegurar que el producto tuviera la forma y funcionalidades esperadas, y cumpliera con los estándares técnicos y de calidad planteados por Endava.

Una vez que comenzaron los sprints, después de definir el MVP y sus historias de usuario pero antes de implementar el código, se diseñó la arquitectura y se seleccionaron las tecnologías para la aplicación. Bajo la guía de los mentores de los practicantes, se optó por implementar una arquitectura monolítica, donde el frontend y el backend son dos partes separadas de la aplicación pero están alojadas en el mismo servidor; su comunicación se realiza a través de peticiones a una API del backend.

Para las tecnologías de desarrollo, se decidió utilizar Java como lenguaje de programación para el backend, en conjunto con el framework Spring Boot. En el frontend, se trabajó con JavaScript y se emplearon dos frameworks: Vue.js para la lógica y maquetación, y Tailwind CSS para la estilización de las interfaces de usuario. Por otro lado, para la persistencia de datos de la aplicación, se optó por utilizar una base de datos relacional, empleando el motor PostgreSQL.

Adicionalmente, fue necesario seleccionar otras tecnologías que facilitaran el correcto trabajo colaborativo. Para este fin, se utilizó Git como herramienta para mantener versiones del código. Este código se almacenará en un repositorio remoto en la plataforma Bitbucket, la cual está diseñada y optimizada para trabajar en conjunto con Jira. Además, Bitbucket cuenta con herramientas para el despliegue continuo del código.

Una vez seleccionadas y configuradas las tecnologías, así como los entornos de desarrollo durante el primer sprint, comenzó el trabajo de implementar las funcionalidades e interfaces de usuario definidas para el MVP. En los ocho sprints que tomó implementar el MVP, se logró incorporar funcionalidades para crear y administrar ofertas de productos, comprar productos, realizar preguntas acerca de los productos, actualizar automáticamente el estado de una oferta cuando se complete la venta, cancelar ventas, listar únicamente los productos disponibles, filtrar productos por categorías y buscar productos por nombre. Además, se crearon interfaces gráficas donde los usuarios pueden ver los datos de su perfil, y listar los productos publicados, comprados y vendidos. Posterior a la implementación del MVP, se continuó mejorando el producto y añadiendo más características que generan valor.

Debido a la inexperiencia del equipo de desarrollo, durante las primeras semanas del proyecto se les permitió escribir código sin redactar las pruebas unitarias correspondientes. Sin embargo, a partir del quinto sprint, tanto el equipo del backend como el del frontend comenzaron a redactar dichas pruebas, tanto para el código ya implementado como para todo código que se fuera escribiendo. La meta para el proyecto fue tener un 80% de cobertura del código, según lo establecido por el PO. La definición de este porcentaje fue importante, ya que la cobertura de código se tomó como el principal indicador cuantitativo de la calidad del producto. A pesar de que existen muchas más métricas para medir y asegurar la calidad del producto, al ser este un proyecto con fines de aprendizaje, se optó por limitar la complejidad del proyecto y mantener una sola métrica de calidad: **la cobertura de pruebas unitarias del código**, que tiende a ser la métrica más común en los proyectos.

V. RESULTADOS

Se logró implementar una arquitectura monolítica por capas, integrando las tecnologías y frameworks propuestos. En esta arquitectura, el backend y el frontend son partes separadas de la aplicación; sin embargo, se encuentran alojados en el mismo servidor y se comunican a través de peticiones HTTP que el cliente, o usuario final, realiza al servidor. El backend se ha diseñado siguiendo una arquitectura de capas, donde cada capa tiene una responsabilidad específica y solo se comunica con capas adyacentes. Por su parte, el frontend, en lugar de dividirse por capas, busca dividirse en componentes. Esta elección se realiza con el objetivo de facilitar la reutilización de elementos de la interfaz gráfica y mejorar la mantenibilidad del código. Como buena práctica, se busca que los componentes solo se encarguen de la parte visual de la aplicación, y cualquier lógica, ya sea para validación o conexiones, se separa de los componentes en módulos aparte.

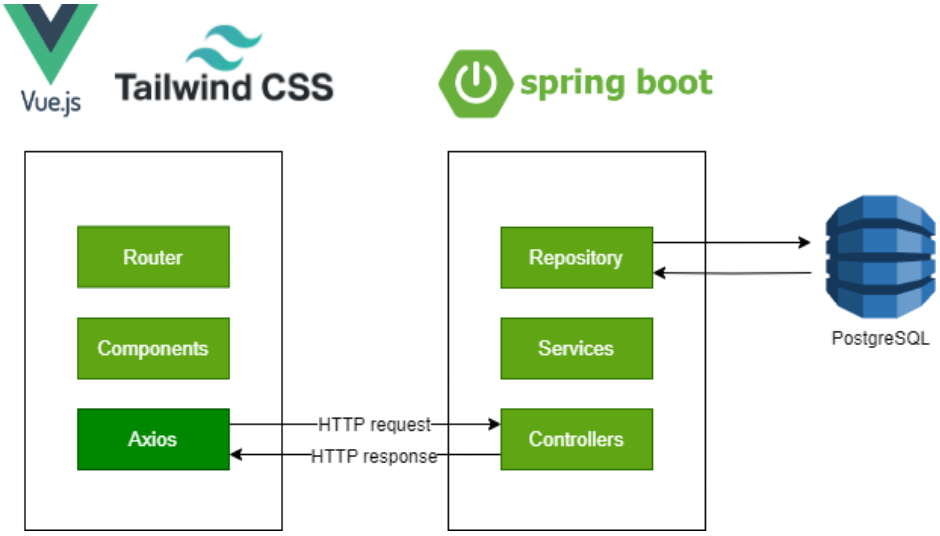

Fig. 2. Arquitectura y tecnologías de la aplicación

Adicionalmente, se logró crear las funcionalidades definidas para el MVP ya mencionadas en la metodología. Posteriormente, se implementan funcionalidades complementarias, tales como:

- Notificaciones para vendedores y compradores.
- Uso de videos para las publicaciones.
- Borradores de publicaciones.
- Búsqueda y filtrado en el listado de productos vendidos y comprados de un usuario.

- Sección de preguntas y respuestas dentro de las publicaciones.

Como complemento a las funcionalidades implementadas, se logró aproximadamente un 77% de cobertura de pruebas unitarias en el código del backend y un 86% de cobertura en el código del frontend. En los anexos A y B se presentan los informes de cobertura del frontend y del backend, respectivamente.

Por último, cabe mencionar que durante el tiempo de desarrollo del proyecto, se llevaron a cabo actividades destinadas a enriquecer el aprendizaje de los practicantes. Entre estas actividades hubo conferencias en las que los practicantes debían exponer, en inglés, sobre temas de desarrollo web, como los lenguajes HTML y CSS. Además, los practicantes tuvieron la oportunidad de participar en retos de desarrollo centrados en el frontend. Tanto el desarrollo de los retos como la retroalimentación recibida permitieron profundizar en los conocimientos de accesibilidad, el framework Vue.js y las buenas prácticas de HTML. Finalmente, también se llevaron a cabo simulacros de entrevistas de trabajo, los cuales contribuyeron a preparar a los practicantes para entrevistas técnicas destinadas a roles de desarrollo.

VI. ANÁLISIS

Se observa en los resultados que se logró superar la meta planteada de cobertura del código. Aunque este es un hito frecuente e importante en los proyectos, es fundamental destacar que la cobertura es solo un paso para asegurar la calidad del código. El siguiente paso es asegurarse de que dichas pruebas cubran todos los escenarios en los que se va a encontrar la aplicación. Sin embargo, no hay forma de cuantificar y medir qué tantos casos cubren las pruebas, ya que cada aplicación es diferente.

A pesar de lo ya mencionado, es correcto afirmar que la aplicación cumple con los requisitos de calidad planteados para el proyecto. El proyecto alcanza la cobertura propuesta, que es la única métrica de calidad planteada, y esta cobertura se mide con herramientas aprobadas por los mentores, quienes tienen experiencia en otros proyectos de la empresa. Si a esto se le suma la correcta implementación de la arquitectura sugerida y las validaciones hechas en revisiones al código, se concluye que la aplicación también cumple con los patrones de escalabilidad propuestos al principio del proyecto, logrando la meta de desarrollar una plataforma funcional y escalable.

Gracias a la retroalimentación compartida por el Product Owner (PO), se tiene la confianza de que la plataforma genera valor para la empresa y ofrece una buena experiencia de usuario. El frontend presenta interfaces gráficas que se consideraron limpias, útiles y de uso intuitivo, donde los usuarios pueden realizar todas las tareas que la plataforma ofrece. Por su parte, el backend y su API separan la lógica de negocio de la aplicación y facilitan servicios de almacenamiento, lo que permite al frontend ser lo más liviano y rápido posible. Con esto, se logra que el usuario tenga una experiencia ágil e intuitiva a la hora de comprar, vender o administrar sus publicaciones y productos.

VII. CONCLUSIONES

Se cumplieron exitosamente los objetivos propuestos para el proyecto, implementando aquellas funcionalidades que se plantearon con el fin de brindar una buena experiencia de usuario. Entre los factores que facilitaron este resultado se encuentran:

- La naturaleza iterativa de la metodología SCRUM, que permitió adaptarse a cambios y corregir errores de manera ágil.
- La correcta creación y validación de las historias de usuario, que aclaró el alcance del proyecto, y permitió la correcta estimación de los tiempos de desarrollo.
- La selección guiada de arquitectura y tecnologías, que aseguraron crear una aplicación escalable.
- La frecuente retroalimentación y validación tanto de mentores como del PO, del código, las funcionalidades, y el estado del MVP.

Por su parte, la realización de pruebas unitarias fue un factor decisivo para garantizar la calidad del código. Ayudan a entender y mejorar la lógica de la aplicación, así como a asegurar la estabilidad y escalabilidad del código en caso de cambios futuros. Se espera, incluso si hay modificaciones en la implementación, que el código continúe produciendo los mismos resultados.

La sumatoria de estos elementos, en conjunto con el énfasis en el trabajo en equipo, la comunicación asertiva, y el aprendizaje, hacen que se tenga la confianza de que la aplicación tiene todas las características deseadas por la empresa.

VIII. RECOMENDACIONES

Un aspecto del proyecto en el que es necesario profundizar, si se quiere crear una plataforma más robusta y complementar el aprendizaje de los practicantes, es la exploración de métricas de calidad adicionales a las pruebas unitarias. Temas como el rendimiento, la optimización para motores de búsqueda, la detección de defectos, entre otros, podrían ser abordados en ambientes controlados.

Por otra parte, las funcionalidades implementadas en este proyecto fueron diseñadas para brindar una experiencia centrada en la compra y venta de productos de segunda mano. Sin embargo, se pueden proponer funcionalidades que amplíen el alcance de la plataforma y la conviertan en un e-commerce, tales como subastas, comparación de productos, sistema de reportes de ventas, etc.

Finalmente, un reto que puede presentarse en el futuro es el replanteamiento de las tecnologías con las que se ha implementado la aplicación, especialmente las del frontend. Java con el framework de Spring Boot es más que capaz de operar en aplicaciones de pequeña y gran escala; sin embargo, Vue.js es diferente. Aunque Vue.js es más que suficiente para la escala actual del proyecto, donde no se espera un número limitado de usuarios y transacciones, es posible que con el tiempo y el crecimiento de la aplicación sea necesario implementar un framework más robusto, como Angular, por ejemplo.

REFERENCIAS

[1] J. Conallen, “Modeling web application architectures with UML,” *Communications of the ACM*, vol. 42, no. 10, pp. 63–70, 1999. doi:10.1145/317665.317677

[2] R. A. Santelices and M. Nussbaum, “A framework for the development of videogames,” *Software: Practice and Experience*, vol. 31, no. 11, pp. 1091–1107, 2001. doi:10.1002/spe.403

[3] L. Gren and V. Antinyan, "On the Relation Between Unit Testing and Code Quality," *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria, 2017, pp. 52-56, doi: 10.1109/SEAA.2017.36.

[4] T. W. Williams, M. R. Mercer, J. P. Mucha and R. Kapur, "Code coverage, what does it mean in terms of quality?," *Annual Reliability and Maintainability Symposium. 2001 Proceedings. International Symposium on Product Quality and Integrity (Cat. No.01CH37179)*, Philadelphia, PA, USA, 2001, pp. 420-424, doi: 10.1109/RAMS.2001.902502.

[5] Vue.js - El framework utilizado para el desarrollo del frontend. Sitio web: https://vuejs.org/

[6] Tailwind CSS - El framework utilizado para los estilos del frontend. Sitio web: https://tailwindcss.com/

[7] Spring Boot - El framework utilizado para el desarrollo del backend. Sitio web: https://spring.io/projects/spring-boot

[8] PostgreSQL - El sistema de gestión de bases de datos relacional empleado en el proyecto. Sitio web: https://www.postgresql.org/

ANEXOS

Anexo A. Reporte de pruebas unitarias del frontend

TABLA I
COBERTURA DE PRUEBAS DEL FRONTEND

Archivos	% Declaraciones de código	% Rama	% Funciones	% Líneas
Todos los archivos	94.27	95.41	67.24	94.27
components	100	100	100	100
BasicSpinner.vue	100	100	100	100
FilterForm.vue	100	100	100	100
GenericModal.vue	100	100	100	100
NotificationBox.vue	100	100	100	100
ProductCard.vue	100	100	100	100
SkeletonCard.vue	100	100	100	100
TableBase.vue	100	100	100	100
TeamsChatLink.vue	100	100	100	100
components/Images	97.8	100	100	97.8
ImageSelector.vue	95.91	100	100	95.91
ImageSelectorList.vue	100	100	100	100
components/Inputs	91.21	80.55	63.15	91.21
FormButton.vue	100	100	100	100
ImageInput.vue	92.2	85.71	60	92.2
ImageInputList.vue	100	85.71	100	100
SelectInput.vue	100	100	100	100
StarSVG.vue	100	50	100	100
StarsInput.vue	72.56	50	0	72.56
TextInput.vue	94.79	86.66	100	94.79
components/Menus	100	100	100	100
DropdownMenu.vue	100	100	100	100
LinkListItem.vue	100	100	100	100
ProductListItem.vue	100	100	100	100

composables	98.83	100	94.73	98.83
useAdminUser.js	100	100	100	100
useForm.js	100	100	100	100
useNotification.js	100	100	100	100
useUserList.js	95.91	100	80	95.91
utils	76.66	97.19	61.7	76.66
axios.js	70.14	94.91	51.72	70.14
cookies.js	78.57	100	66.66	78.57
strings.js	100	100	100	100
userSession.js	79.67	100	70	79.67
views	100	98.07	52	100
AdminPanel.vue	100	100	100	100
CategoriesManagement.vue	100	100	100	100
ListingDetail.vue	100	100	100	100
LoginPage.vue	100	100	100	100
LogoutPage.vue	100	100	100	100
MainPage.vue	100	100	100	100
NewListing.vue	100	100	37.5	100
NotFoundPage.vue	100	100	100	100
PurchaseHistory.vue	100	100	100	100
PurchasedItem.vue	100	100	100	100
SalesHistory.vue	100	100	100	100
UserDashboard.vue	100	100	100	100
UserManagement.vue	100	83.33	50	100
UserProfile.vue	100	100	100	100

Anexo B. Reporte de pruebas unitarias del frontend

TABLA II

COBERTURA DE PRUEBAS DEL BACKEND

Archivos	% Clases	% Métodos	% Líneas
Todos los archivos	90	73	69
Controllers	100	75	79
AuthController	100	100	100
EndavanController	100	66	75
ListingController	100	77	80
SaleController	100	71	75
Models	80	63	70
Endavan	100	100	100
Listing	100	100	100
ListingCategory	100	42	50
ListingStatus	100	60	66
Question	0	0	0
Sale	100	100	100
SaleStatus	100	50	57
Service	100	84	71
EndavanService	100	75	54
ListingCategoryService	100	100	100
ListingService	100	66	25
ListingStatusService	100	72	48
SaleService	100	83	42
SaleStatusService	100	75	80