Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería
y Ciencias
Ingeniería Electrónica

Monografía de Trabajo de Grado

Desarrollo de una aplicación web para la gestión logística en la industria electrónica colombiana

Santiago Muñoz Bocanegra
Daniel Mamian del Rio

Director
Dr. Luis Eduardo Tobón Llano

17 de julio de 2026

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar al título de Ingeniero Electrónico.

Dr. Mateo López Victoria
Decano Facultad de Ingeniería y Ciencias

Dr. Luis Eduardo Tobón Llano
Director Carrera Ingeniería Electrónica

Dr. Luis Eduardo Tobón Llano
Director Trabajo de Grado

Alfredo Herrera
Jurado 1

Juan Pablo García
Jurado 2

Santiago de Cali, 17 de julio de 2026

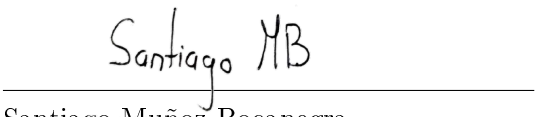
Señores
Pontificia Universidad Javeriana – Cali
Dr. Mateo López Victoria
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

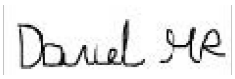
Por medio de la presente nos permitimos presentarle el Trabajo de Grado titulado "Desarrollo de una aplicación web para la gestión logística en la industria electrónica colombiana".

Esperamos que este trabajo reúna todos los requisitos académicos, cumpla el propósito para el cual fue creado y sirva de apoyo para futuros proyectos relacionados con la materia.

Atentamente,



Santiago Muñoz Bocanegra



Daniel Mamian del Rio

Santiago de Cali, 17 de julio de 2026

Señores
Pontificia Universidad Javeriana – Cali
Dr. Mateo López Victoria
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Certifico que el presente Trabajo de Grado titulado "Desarrollo de una aplicación web para la gestión logística en la industria electrónica colombiana", realizado por Santiago Muñoz Bocanegra y Daniel Mamian del Rio, estudiantes de Ingeniería Electrónica, se encuentra terminado y puede ser presentado para su sustentación.

Atentamente,

_____

Dr. Luis Eduardo Tobón Llano
Director Trabajo de Grado

Agradecimientos

En primer lugar, agradecemos a Dios por darnos la fortaleza, la salud y la perseverancia necesarias para culminar este trabajo de grado.

A nuestros padres y a nuestras familias, por su apoyo incondicional, su paciencia y su confianza a lo largo de este proceso. Su respaldo constante fue un pilar fundamental en cada etapa del proyecto.

A nuestros amigos, por su compañía, su ánimo en los momentos difíciles y por recordarnos el valor de seguir adelante cuando el camino se tornaba exigente.

A nuestros profesores, en especial al director de este trabajo, el Dr. Luis Eduardo Tobón Llano, por su orientación, sus observaciones y su acompañamiento académico. Extendemos también nuestro reconocimiento a los docentes que, a lo largo de la carrera, contribuyeron a nuestra formación como ingenieros.

Finalmente, agradecemos a la Pontificia Universidad Javeriana Cali, a sus colaboradores y al personal administrativo y académico que, de distintas maneras, facilitaron el desarrollo de esta investigación y hicieron posible nuestra trayectoria universitaria.

Glosario

API (Application Programming Interface)	Interfaz que permite la comunicación e intercambio de datos entre sistemas mediante reglas y formatos definidos.
Arquitectura Flux	Patrón de arquitectura web con flujo de datos unidireccional y estado centralizado en stores.
Cloud Computing	Modelo que ofrece recursos de computación escalables a través de internet bajo demanda.
Cloud público / privado / híbrido	Modelos de nube: público (compartido), privado (dedicado) e híbrido (combinado).
Componentes electrónicos	Elementos como chips, resistencias o circuitos integrados usados en sistemas electrónicos.
E-commerce (comercio electrónico)	Compra y venta de productos o servicios mediante plataformas digitales.
Gestión logística	Procesos de planificación, almacenamiento y distribución dentro de la cadena de suministro.
Marketplace	Plataforma digital que centraliza productos de múltiples proveedores.
Metodología SCRUM	Metodología ágil basada en iteraciones cortas y entregas incrementales.
NoSQL (bases de datos no relacionales)	Bases de datos con esquemas flexibles que no siguen el modelo relacional tradicional.
Prototipado electrónico	Desarrollo de versiones preliminares de circuitos para validar su funcionamiento.
Proveedor local de componentes	Comerciante nacional que suministra componentes electrónicos.
Scraping web (Web Scraping)	Extracción automatizada de información desde páginas web.
Scraping sintáctico	Extracción basada en la estructura del HTML mediante selectores.
Scraping semántico	Extracción basada en interpretación de contenido mediante análisis avanzado.
SEO (Search Engine Optimization)	Técnicas para mejorar la visibilidad de un sitio web en motores de búsqueda.
SQL (bases de datos relacionales)	Modelo de base de datos estructurado en tablas relacionadas mediante claves.
Store (en arquitectura Flux)	Módulo que gestiona y centraliza el estado de la aplicación.

Resumen

En Colombia, las empresas, estudiantes y demás personas que requieren componentes electrónicos se enfrentaban a dificultades para acceder a inventarios actualizados y opciones de compra, especialmente por la distancia geográfica respecto a los grandes centros de producción a nivel global y por la falta de plataformas locales que centralizaran la oferta disponible. Los costos elevados de importación y la falta de información precisa y actualizada dificultaban tanto el prototipado como la operación diaria de pequeñas y medianas empresas. A partir de esta problemática, se planteó y ejecutó una solución tecnológica orientada a mejorar la accesibilidad y variedad de componentes electrónicos.

En este trabajo de grado se desarrolló una aplicación web que recolectó información de distintos proveedores nacionales e internacionales mediante técnicas de web scraping y uso de APIs, centralizó la oferta de componentes electrónicos y permitió una gestión colaborativa de pedidos. El sistema habilitó el acceso, diversificó la oferta e incentivó la competencia justa en el mercado, con especial atención en la integración de pequeños proveedores. Con ello, se aportó a la mejora de la eficiencia operativa en Colombia y al fortalecimiento de la capacidad de adaptación de las empresas locales en un entorno global y competitivo.

Palabras Clave: componentes electrónicos, inventario, web scraping, gestión colaborativa de pedidos, Colombia.

Abstract

In Colombia, companies, students, and individuals who need electronic components often struggle to access updated inventories and competitive purchasing options. This is mainly due to geographic distance from global production centers and the absence of local platforms that centralize available offerings. High import costs and a lack of precise, current information make prototyping and daily operations more complicated for small and medium-sized businesses. In response to these challenges, there is a clear need for a technological solution that improves accessibility and variety in the procurement of electronic components.

This work proposes the development of a web application designed to gather information from various national and international suppliers through web scraping and API integration. The platform centralizes the supply of electronic components and enables collaborative order management. The goal is to increase access, diversify available options, and promote fair competition, with special focus on including smaller suppliers. By doing so, the system aims to enhance operational efficiency and strengthen the adaptability of local businesses in a competitive, global marketplace.

Keywords: Electronic components, inventory, web scraping, collaborative order management, Colombia

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La industria de componentes electrónicos y semiconductores determina hoy el ritmo de la innovación en múltiples sectores, desde dispositivos de consumo hasta sistemas críticos, pero su cadena de suministro está altamente concentrada geográficamente y expuesta a shocks globales. El liderazgo productivo de empresas como TSMC, con presencia estratégica en Asia y Estados Unidos, ha permitido una coordinación estrecha con grandes clientes, mientras que naciones periféricas quedan lejos de esos centros de decisión y abastecimiento [33]. Esta distancia, física, logística y digital, se traduce en mayores costos, tiempos de entrega extendidos e incertidumbre sobre disponibilidad, lo que afecta la competitividad de actores locales. En el caso colombiano, además, persisten barreras de adopción tecnológica en comercio electrónico y marcos normativos que dificultan el despegue digital del sector, con efectos en eficiencia logística, integración a mercados y costos transaccionales [34]. A nivel regional, la CEPAL advierte que la fragmentación normativa y la débil integración de América Latina agravan la vulnerabilidad de las cadenas de suministro, por lo que es imprescindible fortalecer la disponibilidad y seguridad de insumos estratégicos como los semiconductores [35].

Frente a este panorama, el punto de partida de la presente investigación es que una aplicación web especializada puede contribuir a la gestión logística del abastecimiento electrónico en Colombia. La propuesta busca ofrecer visibilidad consolidada de la variedad y la disponibilidad de componentes, reducir los tiempos de búsqueda y apoyar la toma de decisiones de compra. La experiencia internacional muestra que la adopción de comercio electrónico se asocia con mejoras en desempeño y competitividad de las empresas, aun cuando existan retos regulatorios y de inclusión financiera que deben ser atendidos [34], [36]. Además, los marketplaces globales de componentes (por ejemplo, Octopart, DigiKey, Mouser y Arrow) evidencian el valor de catálogos unificados y buscadores de stock, aunque su uso directo desde Colombia suele verse limitado por costos y tiempos de importación. Esto refuerza la necesidad de soluciones locales que integren datos de múltiples fuentes y prioricen la logística nacional frente al modelo centrado en plataformas globales [37, 38, 39, 40].

El web scraping y la recolección de datos vía API permiten extraer, transformar y disponibilizar información desde fuentes heterogéneas (páginas HTML, documentos PDF y servicios REST), superando las limitaciones del copiado manual y habilitando formatos estructurados (CSV y JSON) para análisis y publicación [14], [41]. En contextos donde el stock cambia con frecuencia y la estandarización de catálogos es dispar, estas técnicas habilitan la construcción de vistas unificadas del mercado. Por su parte, buenas prácticas de diseño y arquitectura, como los principios SOLID y el patrón Flux, favorecen soluciones mantenibles y escalables para interfaces ricas y estados comple-

jos [42], [43]. La selección de almacenamiento (SQL o NoSQL) y el despliegue en la nube aportan elasticidad, disponibilidad y eficiencia en costos para cargas variables propias de la agregación de catálogos y consultas concurrentes [23], [7].

La propuesta se enmarca en un enfoque de ingeniería de software aplicada al dominio logístico-electrónico. Se capturan datos de proveedores nacionales e internacionales, se normalizan y se exponen mediante una aplicación web que mejora la búsqueda, la comparación y la planificación de compras. También se exploran mecanismos de colaboración entre pares, como pedidos conjuntos y listas personalizadas, con el fin de mitigar costos de importación y aprovechar economías de escala cuando el stock local es insuficiente [34], [14].

Capítulo 2

Planteamiento del Problema

Durante las últimas décadas, la industria global de los semiconductores ha experimentado un crecimiento acelerado, convirtiéndose en un pilar fundamental para el desarrollo tecnológico y económico de las naciones. En el panorama de 2026, la cadena de valor se distribuye entre polos de diseño y automatización de diseño electrónico (EDA), fabricación avanzada de obleas (fábricas o *fabs*), y centros de ensamblaje y prueba (OSAT) en distintas regiones. Estados Unidos, la Unión Europea, China, Japón, Corea del Sur y Taiwán concentran capacidades críticas, mientras que economías del sudeste asiático desempeñan un papel relevante en ensamblaje y testeo. Las principales potencias han desplegado iniciativas públicas de inversión (por ejemplo, el CHIPS and Science Act en Estados Unidos, el European Chips Act en la UE, estrategias de autosuficiencia tecnológica en China, y planes sectoriales en Japón y Corea del Sur), lo que intensifica la competencia por capacidad instalada, talento y resiliencia de la cadena de suministro. Empresas como Taiwan Semiconductor Manufacturing Company (TSMC) siguen siendo un referente en manufactura avanzada, con presencia estratégica en Asia y Estados Unidos y relación estrecha con grandes clientes como Apple y Samsung [33].

La Figura 2.1 muestra la evolución de la participación de TSMC frente a otros fabricantes relevantes. Por su parte, la Figura 2.2 sintetiza cómo las principales regiones articulan diseño, fabricación avanzada y ensamblaje, y cómo las políticas públicas buscan regionalizar la producción frente a riesgos geopolíticos. De acuerdo con MicroFabMX, hacia 2026 la demanda global seguirá impulsada por inteligencia artificial, centros de datos, movilidad eléctrica y manufactura avanzada, en un contexto donde Estados Unidos, China, Japón, Corea del Sur y la Unión Europea intensifican inversiones para ganar autonomía en *fabs* y en la cadena de suministro; coexisten, no obstante, presiones por concentración productiva, acceso a materiales críticos (p. ej. tierras raras), brecha de talento y exigencias de sostenibilidad y divulgación de carbono en el diseño industrial [1].

No obstante, países como Colombia se encuentran geográficamente alejados de estos centros de innovación y manufactura, lo que representa un desafío importante para su competitividad en el mercado de semiconductores. La distancia, sumada a la dependencia de la importación de componentes, genera costos elevados, retrasos logísticos y limitaciones para desarrollar ecosistemas tecnológicos sólidos. En este contexto, resulta indispensable diseñar estrategias que permitan a la industria electrónica nacional adaptarse al panorama global y reducir su vulnerabilidad frente a las dinámicas del comercio internacional.

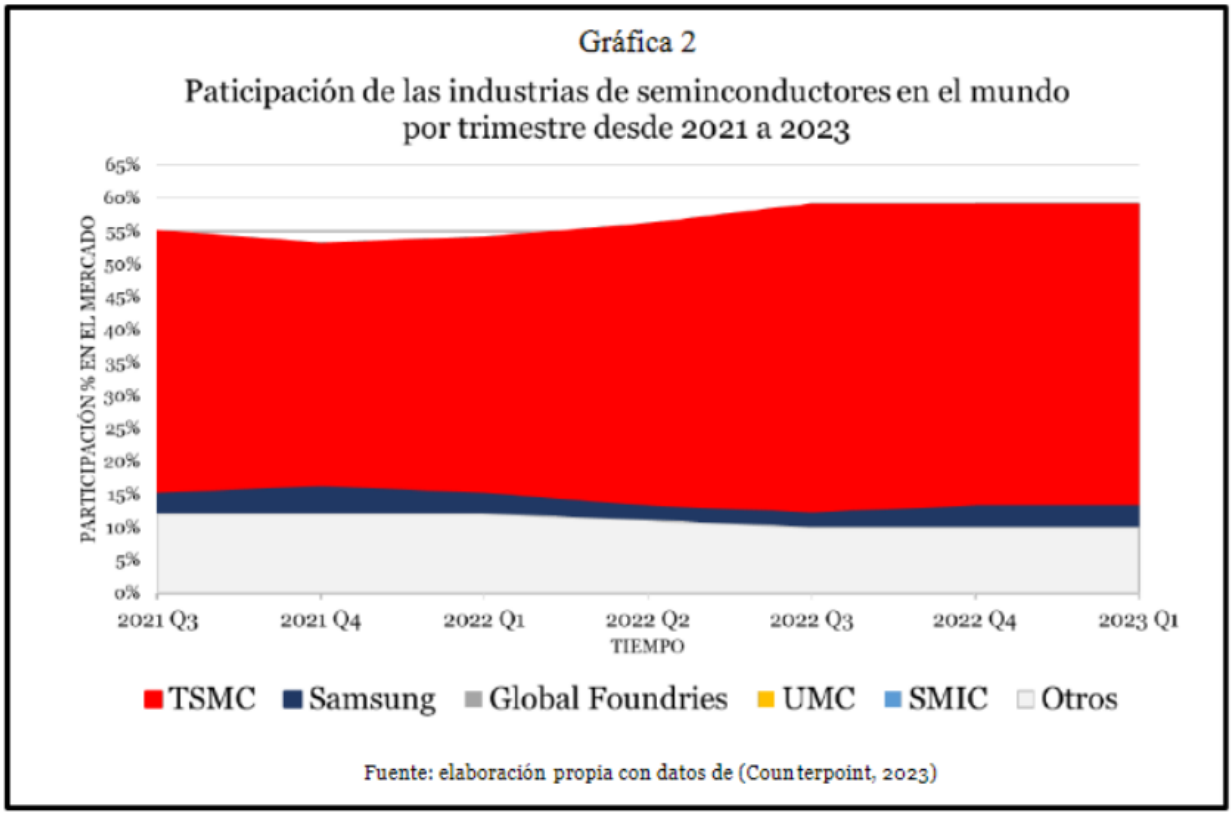


Figura 2.1: Participación de TSMC frente a sus competidores

La falta de adopción de tecnologías de la información en el mercado colombiano de componentes electrónicos y semiconductores agrava este panorama. La carencia de plataformas digitales que ofrezcan acceso a inventarios actualizados limita la visibilidad sobre la disponibilidad y variedad de productos, afectando la planificación y ejecución de proyectos tecnológicos. En consecuencia, la adquisición de componentes suele depender de procesos manuales o intermediarios, lo que incrementa los costos y reduce la eficiencia operativa.

Diversos estudios han señalado el potencial del comercio electrónico como herramienta para dinamizar el mercado de dispositivos electrónicos en Colombia. En [34], se argumenta que la adopción de plataformas digitales puede facilitar la entrada de empresas colombianas a mercados internacionales, optimizar las cadenas de suministro y reducir los costos asociados a la importación. Sin embargo, el mismo estudio advierte que el país enfrenta desafíos significativos, como la ausencia de un marco legal robusto y la existencia de barreras tributarias que dificultan la consolidación del comercio electrónico.

Estas limitaciones han impulsado una tendencia en las empresas colombianas hacia la compra de

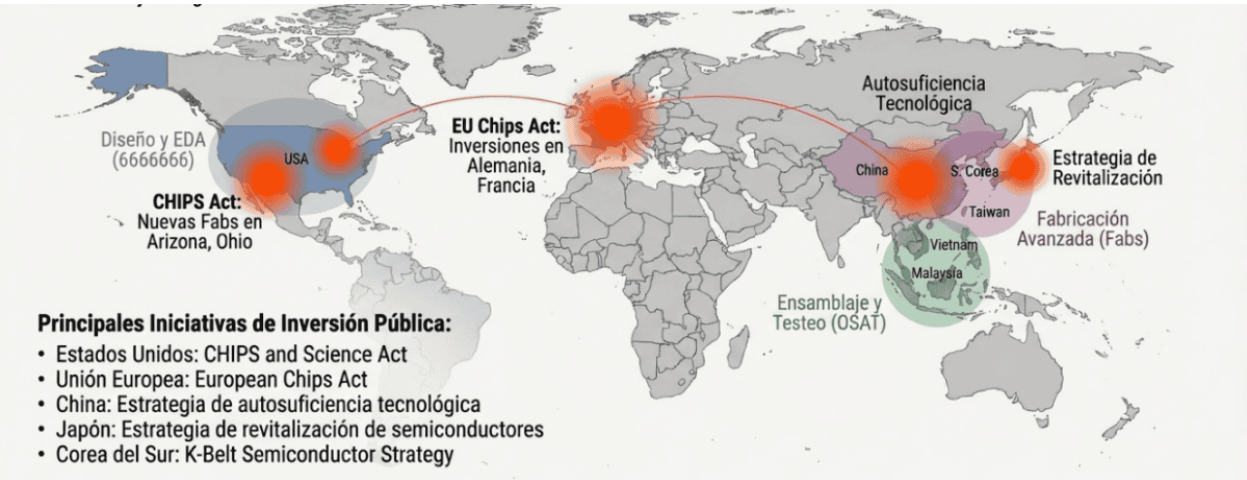


Figura 2.2: Panorama de los principales actores e iniciativas públicas de inversión en semiconductores a nivel mundial (2026). [1]

componentes en mercados extranjeros, incrementando los costos y la complejidad logística. La baja participación en compras en línea dentro de Colombia y América Latina, en comparación con otras regiones del mundo, se debe en parte a la falta de plataformas especializadas, eficientes y adaptadas al contexto local. Esta brecha tecnológica no solo restringe el acceso a insumos, sino que también reduce las oportunidades de innovación y desarrollo en el ámbito electrónico nacional.

A nivel regional, la falta de integración económica y tecnológica en América Latina incrementa la exposición de los países a las fluctuaciones del comercio internacional. Como advierte la CEPAL en [35], la "fragmentación normativa" del comercio global aumenta la vulnerabilidad de los países sudamericanos, que dependen en gran medida de las políticas comerciales y productivas de las potencias tecnológicas. Sin mecanismos que garanticen la estabilidad y la seguridad en la cadena de suministro de semiconductores, países como Colombia corren el riesgo de enfrentar obstáculos estructurales para acceder a los componentes necesarios para su industria electrónica.

Esta situación no solo compromete la competitividad de las empresas locales, sino que también puede tener implicaciones más amplias en la seguridad y confiabilidad de la cadena de suministro en un contexto global cada vez más interdependiente. La falta de soluciones tecnológicas nacionales posiciona a Colombia como un destino poco atractivo para la inversión extranjera en manufactura electrónica y limita su capacidad de innovación en sectores estratégicos.

Aunque existen plataformas internacionales como Octopart, Digi-Key o Mouser, estas no se encuentran adaptadas a las condiciones económicas y logísticas del país. Los altos costos de envío y los extensos tiempos de entrega hacen que su uso sea viable únicamente para compras al por mayor, excluyendo los procesos de prototipado o desarrollo de pequeña escala, que son esenciales en el

ámbito académico y en la investigación tecnológica.

Frente a este panorama, se hace evidente la necesidad de implementar tecnologías que permitan centralizar la información de múltiples proveedores, brindar acceso a inventarios actualizados y habilitar mecanismos de compra colaborativa para reducir costos. De esta manera, se podrían superar las barreras actuales, mejorar la eficiencia operativa y fortalecer la competitividad del sector electrónico colombiano.

En conclusión, el desafío consiste en diseñar una solución tecnológica que no solo responda a las limitaciones logísticas y de información, sino que además promueva la integración entre los actores del mercado. El desarrollo de una solución que facilite la gestión logística de componentes electrónicos en Colombia representa una oportunidad concreta para modernizar el sector, apoyar la innovación local y posicionar al país dentro del ecosistema tecnológico global.

Capítulo 3

Justificación

La adopción de tecnologías web en el mercado colombiano de componentes electrónicos y semiconductores representa una oportunidad estratégica para transformar la gestión de la cadena de suministro. Estas tecnologías permiten obtener visibilidad en tiempo real sobre la disponibilidad de productos, la ubicación de inventarios y la capacidad de respuesta ante variaciones en la demanda, lo cual resulta clave para mejorar la eficiencia operativa y reducir costos.

Asimismo, la implementación de tecnologías web puede facilitar la integración económica regional al optimizar los procesos de adquisición y distribución de componentes. Esto fortalecería la posición de América Latina dentro de la cadena de suministro global y disminuiría su dependencia de redes productivas concentradas en Estados Unidos, especialmente frente a prácticas como el acaparamiento [34], basado en proyecciones de demanda que afectan particularmente a países con baja producción local.

En [35] se evidencia cómo la adopción del comercio electrónico mejora el desempeño empresarial en Indonesia, resaltando beneficios en competitividad e inclusión financiera. Aunque su análisis no se enfoca directamente en el mercado de semiconductores, los resultados permiten inferir el potencial que tendrían estas tecnologías si se aplicaran a este sector en Colombia, especialmente en términos de productividad y modernización.

Por otra parte, la perspectiva de las empresas que fabrican, distribuyen o comercializan componentes electrónicos refuerza la necesidad de implementar tecnologías web. En conversaciones sostenidas con representantes del sector [14], se destacó que la digitalización representa un avance significativo en términos de eficiencia, visibilidad y competitividad.

Estas empresas han enfatizado tres aspectos fundamentales. En primer lugar, la posibilidad de coordinar pedidos conjuntos entre varios usuarios que requieran el mismo componente. Una plataforma digital permitiría identificar coincidencias de demanda y consolidar compras, reduciendo los costos de envío que, en muchos casos, superan el precio del propio componente.

En segundo lugar, resaltaron la importancia del acceso a información estratégica mediante técnicas de *web scraping*, análisis de datos e inteligencia artificial. Estas herramientas permiten recopilar información sobre tendencias del mercado, precios y comportamiento de la competencia, apoyando

la toma de decisiones informadas sobre adquisiciones, estrategias comerciales y gestión de inventarios.

El tercer aspecto se relaciona con la gestión del stock no utilizado. En la industria electrónica es frecuente acumular excedentes de componentes que no llegan a emplearse y ocupan espacio en bodegas. Una aplicación web que permita ofrecer componentes de segunda mano mediante venta o intercambio contribuiría a reducir pérdidas y fomentaría prácticas de economía circular.

Considerar estos factores nos lleva a plantear la necesidad de un sistema basado en tecnologías web. Este sistema podría funcionar como un *marketplace* especializado que no solo permita visualizar productos nuevos y usados, sino que también incluya a pequeños proveedores que hoy carecen de presencia digital y por ende de oportunidades de comercialización.

Esta visión ofrece las siguientes ventajas:

1. **Darles entrada a los proveedores pequeños:** Con una plataforma centralizada, estos proveedores —que muchas veces no tienen cómo mostrarse— podrían llegarle a más gente, hacer visible lo que ofrecen y vender sus productos de forma más fácil y organizada.

2. **Diversificación de ofertas:** Al integrar productos provenientes de pequeños proveedores y componentes usados, se ampliaría la disponibilidad y variedad de opciones para empresas y desarrolladores.

3. **Promoción de una competencia justa:** La inclusión de actores pequeños y productos reutilizados fomentaría la competencia y mantendría precios más competitivos para el consumidor final.

4. **Apoyo a la innovación:** Facilitar el acceso a productos usados y a nuevos proveedores favorece la experimentación, el prototipado y la generación de nuevas soluciones tecnológicas.

En conjunto, esta propuesta permite abordar las limitaciones actuales del mercado colombiano de componentes electrónicos y semiconductores. Además de mejorar la eficiencia y la competitividad del sector, una plataforma de este tipo puede contribuir de manera significativa a la toma de decisiones a nivel local y global, impulsando el desarrollo tecnológico y fortaleciendo el ecosistema electrónico nacional.

Capítulo 4

Objetivos

4.1. Objetivo General

Desarrollar una aplicación web para la gestión logística de la implementación de sistemas electrónicos, a partir de la disponibilidad y variedad de componentes electrónicos en el mercado.

4.2. Objetivos Específicos

1. Identificar las características que debe tener la aplicación web para lograr la gestión logística de la implementación de sistemas electrónicos, considerando la disponibilidad y variedad de componentes electrónicos en el mercado.

2. Diseñar una arquitectura que soporte adecuadamente los requerimientos funcionales, técnicos y de escalabilidad de la aplicación.

3. Implementar la aplicación con las características especificadas y con la arquitectura seleccionada, garantizando coherencia entre diseño y ejecución.

4. Evaluar el prototipo final mediante métricas de performance, accesibilidad, SEO y buenas prácticas, complementado con pruebas de experiencia de usuario y recolección de feedback para medir la usabilidad y nivel de satisfacción de los usuarios.

Capítulo 5

Marco de Referencia

El siguiente marco de referencia abordará los conceptos y fundamentos teóricos relacionados con la recolección de datos en entornos digitales, incluyendo metodologías, herramientas y tecnologías utilizadas en este proceso.

5.1. Recolección de Datos

En internet, la cantidad de información disponible a través de diversas páginas web es muy extensa. Sin embargo, esta inmensidad conlleva desafíos significativos en la recuperación de datos relevantes. La complejidad de las páginas web, con sus variados formatos como texto, gráficos, audio y video, dificulta la obtención efectiva de información de manera manual. [14]

La extracción manual de datos presenta limitaciones notables, especialmente en la incapacidad de guardar copias de la información de manera eficiente para uso personal. La única opción viable suele ser copiar y pegar manualmente los datos en el disco duro, una tarea tediosa y propensa a errores. Es en este contexto que surge el web scraping como una solución automatizada.

Esta técnica, también conocida como Screen Scraping o Web Scraping, se convierte en un procedimiento esencial para la extracción automatizada de datos desde el código HTML de las páginas web. Utilizando programas especialmente diseñados, los web scrapers extraen información significativa y la almacenan en una base de datos local central o en hojas de cálculo.

5.1.0.1. La necesidad del scraping de sitios web y documentos PDF

El énfasis en la naturaleza no estructurada de los datos en PDF resalta la prevalencia de este formato en la distribución de información en internet. Aproximadamente el 70 por ciento de la información en la web se encuentra en documentos PDF, los cuales presentan un desafío significativo debido a su estructura no uniforme. La información dentro de estos documentos a menudo carece de un formato consistente, lo que dificulta su manejo y análisis. [14]

Por otro lado, los aplicativos web tienen un formato organizado para el usuario pero solo en términos de visualización, pues aunque estas páginas suelen presentar una estructura organizada en términos de código HTML, la información visible en el navegador no siempre se puede extraer de

manera eficiente para uso personal. La incapacidad de las páginas web para ofrecer funcionalidades que permitan a los usuarios guardar una copia estructurada de los datos crea la necesidad de una solución que simplifique este proceso.

Es en este punto donde entra en juego el web scraping como una herramienta esencial. Las herramientas de web scraping, al operar a nivel del código HTML de las páginas web, permiten la conversión de datos en formatos accesibles y reutilizables. Los datos extraídos pueden ser transformados y almacenados en formatos como JSON, XML, CSV, XLS o RSS, proporcionando así a los usuarios la flexibilidad de elegir la estructura de datos que mejor se adapte a sus necesidades.

5.1.0.2. La recolección de datos por medio de API y el web scraping

1. Colección de Datos a través de API: La API (Interfaz de Programación de Aplicaciones) se presenta como un componente fundamental en la colección de datos. Esta interfaz facilita la conexión entre dos aplicaciones, permitiendo el intercambio de datos de manera eficiente. En el contexto del web scraping, la API simplifica el acceso a los datos, ofreciendo una vía estructurada para que los usuarios recuperen la información deseada.

2. Automatización a través de Herramientas de Web Scraping: Se introduce la automatización como una solución clave, y las herramientas de web scraping emergen como la respuesta para superar las limitaciones de la recolección manual. Estas herramientas, impulsadas por programas especialmente codificados, permiten la extracción automática y organizada de datos desde el código HTML de las páginas web.

Hay diferentes formas de realizar esta tarea actualmente, entre ellas están:

1. Programación Manual: Esta forma implica escribir scripts personalizados en un lenguaje de programación (como Python, JavaScript, o Ruby) para acceder y extraer datos específicos de las páginas web.

2. Uso de Frameworks: La utilización de frameworks específicos de web scraping, como Scrapy en Python, simplifica el proceso al proporcionar estructuras y funciones predefinidas para la extracción de datos. Entre los más populares se encuentran BeautifulSoup y Selenium.

3. Herramientas de Extracción Visual: Herramientas como ParseHub o Octoparse permiten a los usuarios extraer datos visualmente sin escribir código, utilizando interfaces gráficas para seleccionar y definir los elementos a extraer.

4. Extensiones del Navegador: Extensiones como BeautifulSoup o Selenium, integradas con navegadores web, facilitan la inspección y extracción de datos directamente desde el navegador, proporcionando flexibilidad y control.

5. APIs de Web Scraping: Algunas plataformas ofrecen APIs dedicadas para realizar web scraping de manera estructurada. Estas APIs simplifican el proceso y permiten la integración de datos en otras aplicaciones.

6. Herramientas Basadas en la Nube: Existen servicios como Import.io o Diffbot que ofrecen soluciones basadas en la nube, donde los usuarios pueden configurar y ejecutar tareas de web scraping sin la necesidad de gestionar infraestructuras locales.

7. Simplificación de URL para la realización de web scraping: La simplificación de las URL es un beneficio adicional proporcionado por la API. Esta simplificación no solo facilita el acceso a los datos raspados, sino que también mejora la usabilidad al proporcionar direcciones web más amigables. Es decir, URL que, de acuerdo con su diseño, permiten llegar de forma predecible a la ruta deseada; por ejemplo: www.somewebsite.com/year/month/day.html. Por medio de esta ruta podemos obtener algún recurso dependiendo de la fecha ingresada como parámetro dentro de la URL.

5.1.0.3. Tipos de web scraping:

Existen diferentes enfoques y técnicas para llevar a cabo el web scraping, cada uno adaptado a las necesidades específicas de la información que se busca extraer. A continuación, se presentan los tipos de web scraping más destacados [14]

- Web Scraping Sintáctico.
  - Selectores de CSS: Utilización de selectores CSS para identificar y extraer elementos específicos de las páginas web usando como identificador las clases que se utilizan para definir las propiedades visuales.
  - Selectores XPath: Empleo de expresiones XPath para la navegación y extracción de datos desde documentos XML o HTML, proporcionando una forma precisa de seleccionar nodos.
- Web Scraping Semántico.
  - Comparación de Datos Extraídos con Recursos Semánticos: Evaluación de la información extraída en comparación con recursos web semánticos como expresiones regulares o expresiones puntuales. Es decir que en este caso se hace la extracción de la data ya sea por medio de comparaciones con cadenas de caracteres puntualmente o por medio de expresiones regulares para la simplificación de esta tarea.
- Análisis de Páginas Web con Visión por Computadora.

- Aplicación de Aprendizaje Automático y Procedimientos de Visión por Computadora: Son herramientas que integran técnicas de aprendizaje automático y procedimientos de visión por computadora para la identificación y extracción de información compleja desde las páginas web. Entre ellas está, por ejemplo, Diffbot, que ofrece un sistema bastante escalable para hacer web scraping sin usar la estructura web de las páginas, sino su parte visual, para así poder extraer datos, tal y como se haría para extraer datos de un PDF, o como se realiza el reconocimiento de placas de vehículos por medio de imágenes.

5.2. Desarrollo

En esta sección se describirán los conceptos fundamentales necesarios para la fase de desarrollo del proyecto. Se abordarán las herramientas, tecnologías y metodologías empleadas, así como su relevancia en la implementación.

5.2.0.1. API

Una API funciona como un contrato entre distintas partes de aplicaciones integradas en el código fuente de la aplicación principal. La comunicación entre aplicaciones se realiza en un lenguaje mutuamente entendido, posiblemente a través de una red. Las APIs otorgan funcionalidad de una aplicación a otra cuando son llamadas.[41]

Una API posibilita el intercambio de datos y funcionalidades entre aplicaciones, sirviendo a desarrolladores externos, socios comerciales y departamentos internos. Los usuarios y desarrolladores no necesitan comprender los detalles específicos del desarrollo de una API.

5.2.0.2. Tipos de API

Existen diferentes categorías de API que desempeñan roles específicos en función de su accesibilidad y propósito. Aquí se detallan cuatro categorías principales [41]

- API Pública (Open API):

  - Características: Es de código abierto y está disponible para uso externo. Su objetivo principal es ser utilizado por desarrolladores externos, la comunidad y terceros.
  - Uso: Facilita la integración de servicios y datos de una organización con aplicaciones externas. Fomenta la innovación y el desarrollo de nuevas aplicaciones que aprovechan la funcionalidad proporcionada por la API.

- API Privada (Interna):

  - Características: Diseñada para uso interno dentro de la organización que la desarrolla. No está destinada a ser compartida con el público en general.
  - Uso: Proporciona una interfaz para que los diferentes servicios y sistemas internos se comuniquen entre sí. Permite una integración eficiente de funciones dentro de la infraestructura interna de la organización.

- API de Socio (Partner API):

  - Características: Compartida selectivamente con socios comerciales específicos. Facilita la colaboración y la integración con socios estratégicos.
  - Uso: Permite a socios comerciales acceder a ciertas funciones o datos de la organización, fortaleciendo las relaciones comerciales. Puede ser utilizada para crear soluciones conjuntas o integraciones personalizadas con socios estratégicos.

- API Compuesta (Composite API):

  - Características: Combina múltiples APIs en una única interfaz.Realiza una solicitud a un servidor a través de múltiples llamadas y recibe una respuesta con la información necesaria.
  - Uso: Simplifica la interacción con sistemas complejos al proporcionar una interfaz unificada.Permite a los desarrolladores realizar operaciones complejas con una única solicitud, mejorando la eficiencia y la experiencia del usuario.

5.2.0.3. Tecnologías y Protocolos de API

Las tecnologías y protocolos utilizados en el desarrollo de API desempeñan un papel crucial en la eficiencia, la flexibilidad y la interoperabilidad de las aplicaciones. Aquí se profundiza en algunas de las tecnologías y protocolos más populares [41]

5.2.0.4. Estilo Arquitectónico REST

- **Descripción:** Separación clara entre el front-end y el back-end, facilitando la escalabilidad y el mantenimiento. Uso de operaciones HTTP estándar como GET, POST, PUT y DELETE para acciones CRUD.

- **Características:**

- Define un conjunto de principios para el diseño de servicios web.
- Se basa en la transferencia de representaciones de recursos a través de HTTP.

- **Uso:** Ampliamente utilizado en el desarrollo de servicios web y API debido a su simplicidad y flexibilidad.

5.2.0.5. SOAP (Simple Object Access Protocol)

- **Descripción:**
  - Un protocolo de mensajería estándar que utiliza XML para la comunicación entre aplicaciones.
  - Se centra en la definición e intercambio de información estructurada.
- **Características:**
  - Mayormente asociado con servicios web que requieren una comunicación más estructurada.
  - Define reglas estrictas para la mensajería y la descripción de servicios a través de WSDL (Web Services Description Language).
- **Uso:** Común en entornos empresariales donde la formalidad y la estructura son críticas.

5.2.0.6. Protocolo de Llamada a Procedimiento Remoto (RPC)

- **Descripción:**
  - Un protocolo que permite a un programa ejecutar procedimientos en otro espacio de direcciones de manera remota.
  - Utiliza solicitudes y respuestas para la comunicación entre cliente y servidor.
- **Características:**
  - Utiliza protocolos de transporte de bajo nivel como UDP o TCP/IP para la transmisión de datos.
  - Proporciona una interfaz simple para la ejecución remota de funciones o procedimientos.
- **Uso:** Común en entornos distribuidos donde la ejecución remota de funciones es esencial.

5.2.0.7. Patrones de Diseño

A continuación se muestra uno de los acrónimos más conocidos para código limpio, que representa cinco principios básicos de la programación orientada a objetos y el diseño [42].

- S (Responsabilidad Única): Una clase debe tener solo una razón para cambiar.
- O (Abierto/Cerrado): Una clase debe estar abierta para su extensión pero cerrada para su modificación.
- L (Sustitución de Liskov): Los objetos de una superclase deben poder ser reemplazados por objetos de una subclase sin afectar la integridad del programa.
- I (Segregación de Interfaces): Ninguna clase debería ser forzada a implementar interfaces que no utiliza.
- D (Inversión de Dependencias): Los módulos de alto nivel no deben depender de módulos de bajo nivel. Ambos deben depender de abstracciones.

5.2.0.8. Patrón Flux [43]

- El patrón de arquitectura Flux es una serie de patrones que busca, a través de ciertas normas y restricciones, mantener un sistema que pueda crecer de manera mantenible. Impone una entrada de datos única y flujo unidireccional de datos, y establece qué capas pueden comunicarse y en qué sentido.
- Flux evita que los controladores y las vistas puedan modificar el estado, entendiendo el estado como los datos que representan la interfaz gráfica de una aplicación en un momento dado. Ya que mantener el estado es algo imprescindible y tiende a volverse complejo a medida que escala la aplicación, Flux argumenta que la mejor forma de hacerlo es limitando quién puede modificar el estado y dónde.
- Para esto, Flux introduce un elemento llamado store, el cual es el punto de entrada de los datos del sistema. Al crear un store, pensar en la estructura que debe llevar la información es un paso obligatorio. Toda la manipulación de datos se da dentro del store. Se puede considerar al store como una fábrica de información donde entran los datos, se estructuran para la necesidad de la aplicación y salen como información. El store controla cómo entran los datos y lleva a cabo los cambios de estado.
- Además de manejar dónde ocurren los cambios de estado, es importante gestionar el orden en que estos se llevan a cabo. En un sistema donde los datos se actualizan asincrónicamente, debemos considerar los comportamientos anómalos debido a dependencias críticas inesperadas en el orden de ejecución de eventos. Flux previene esto asegurándose de que todos los cambios de estado en el store se hagan de manera asincrónica.

- Flux obliga a mantener un flujo de datos unidireccional y por lo tanto elimina la posibilidad de que un componente actualice los datos de una manera que rompa el sistema. De manera que sin importar el tipo de datos, cuando estos entran o se actualizan fluyen desde el store hacia los componentes, y de estos a sus componentes hijos y bajando por el árbol de componentes hasta llegar al final. Manteniendo un flujo unidireccional de datos el problema de entender de donde proviene un cambio de estado se simplifica. Esto es porque siempre se debe originar desde store, haciendo el flujo de datos predecible.

5.3. Infraestructura

5.3.0.1. Almacenamiento

A continuación se presenta un análisis comparativo de las bases de datos SQL y NoSQL. Se incluyen detalles técnicos, ejemplos específicos y consideraciones adicionales para la elección de la base de datos adecuada. [23]

- **Bases de Datos Relacionales (SQL)**:
  - **Modelo de datos:**
    1. Se basa en tablas con filas y columnas.
    2. Las relaciones entre tablas se establecen mediante claves primarias y foráneas.
    3. Las entidades se representan como filas, y sus atributos como columnas.
    4. Se pueden aplicar restricciones de integridad referencial para garantizar la consistencia de los datos.
  - **Características:**
    1. Soporte para transacciones complejas que abarcan múltiples tablas.
    2. Alto nivel de integridad y consistencia de los datos.
    3. Lenguaje de consulta estructurado (SQL) para la recuperación y manipulación de datos.
    4. Amplia variedad de herramientas y frameworks disponibles.
    5. Se pueden aplicar restricciones de integridad referencial para garantizar la consistencia de los datos.
  - **Casos típicos:**
    1. MySQL: Base de datos de código abierto popular para aplicaciones web y empresariales.
    2. PostgreSQL: Base de datos robusta con características avanzadas como la replicación y el particionamiento.
    3. Oracle: Base de datos de alto rendimiento utilizada en grandes empresas.

- **Bases de Datos No Relacionales (NoSQL):**
  - **Tipos de Bases de Datos NoSQL:**
    1. Clave-valor: Almacenan pares clave-valor, ideales para almacenar grandes cantidades de datos simples. Casos típicos: Redis, Memcached, Amazon SimpleDB.
    2. Documentales: Almacenan documentos JSON o XML, permitiendo consultas más complejas. Casos típicos: MongoDB, CouchDB, DocumentDB.
    3. Orientadas a grafos: Almacenan relaciones entre entidades, ideales para redes sociales y análisis de grafos. Casos típicos: Neo4j, Titan, ArangoDB.
    4. Columnas: Almacenan datos en columnas, optimizadas para análisis de datos y agregaciones. Casos típicos: Cassandra, HBase, Apache Accumulo.
    5. Object Storage: Está diseñado para almacenar y recuperar grandes cantidades de datos no estructurados, como archivos multimedia, documentos, copias de seguridad, registros y otros tipos de archivos. Los datos se organizan en .ºbjetos"que consisten en datos y metadatos asociados. Casos típicos: S3 (Simple Storage Service) de Amazon Web Services (AWS) se clasifica como un servicio de almacenamiento en la nube. Dentro de esta categoría, S3 se considera un servicio de almacenamiento de objetos
  - **Características:**
    1. Flexibilidad en el modelo de datos, sin necesidad de estructuras rígidas como las tablas.
    2. Mayor escalabilidad horizontal para manejar grandes volúmenes de datos
    3. Lenguaje de consulta estructurado (SQL) para la recuperación y manipulación de datos.
    4. Rendimiento optimizado para operaciones específicas según el tipo de base de datos.
    5. Se pueden aplicar restricciones de integridad referencial para garantizar la consistencia de los datos.
  - **Casos típicos:**
    1. Cassandra: Utilizada por Facebook para almacenar datos de usuario y publicaciones.
    2. MongoDB: Popular para aplicaciones web y móviles que requieren flexibilidad de datos.
    3. Neo4j: Utilizada por empresas como LinkedIn y Twitter para análisis de redes sociales.

5.3.0.2. Disponibilización

Cloud Computing es un modelo de computación en la nube que ofrece acceso a recursos informáticos como un servicio, a través de Internet. Este modelo permite a las empresas escalar sus recursos de forma rápida y flexible, sin necesidad de invertir en hardware o software propio. El término "nube"se utiliza como una metáfora para Internet, y la computación en la nube se basa en

tecnologías de virtualización y automatización para aprovechar al máximo los recursos disponibles. [7]

1. Evolución:

   a) La idea de la computación en la nube se remonta a la década de 1960, pero no fue hasta principios del siglo XXI que la tecnología maduró lo suficiente para que se hiciera realidad.

   b) En 2006, Amazon lanzó Amazon Web Services (AWS), que se convirtió en el primer proveedor de servicios de computación en la nube a gran escala.

   c) Desde entonces, otros proveedores como Google Cloud Platform (GCP) y Microsoft Azure han entrado en el mercado, y la computación en la nube se ha convertido en una parte esencial de la infraestructura de TI para empresas de todos los tamaños.

2. Tecnologías Relacionadas:

   a) Grid Computing: Es un paradigma de computación distribuida que coordina recursos en red para lograr un objetivo computacional común. La computación en la nube es similar a la computación Grid en que ambas emplean recursos distribuidos para lograr objetivos de nivel de aplicación.

   b) Virtualización: Es una tecnología que abstrae los detalles del hardware físico y proporciona recursos virtualizados para aplicaciones de alto nivel. La virtualización es la base de la computación en la nube, ya que proporciona la capacidad de agrupar recursos informáticos de grupos de servidores y asignar o reasignar dinámicamente recursos virtuales a aplicaciones bajo demanda.

   c) Computación autónoma: Tiene como objetivo construir sistemas informáticos capaces de autogestión, es decir, reaccionar a las observaciones internas y externas sin intervención humana. La computación en la nube presenta ciertas características autónomas como el aprovisionamiento automático de recursos, pero su objetivo es reducir el costo de los recursos en lugar de reducir la complejidad del sistema.

3. Modelos de Cloud Computing:

   La arquitectura del Cloud Computing se puede dividir en 4 capas:

   a) Public Cloud: Los proveedores de servicios ofrecen sus recursos como servicios al público en general.

   b) Private Cloud: También conocidas como clouds internas, están diseñadas para uso exclusivo de una sola organización.

   c) Hybrid Cloud: Es una combinación de modelos de cloud pública y privada que intenta abordar las limitaciones de cada enfoque.

5.4. Seguridad en Entornos Cloud

La seguridad en la nube es un tema importante para las empresas que consideran migrar sus datos y aplicaciones a la nube. Los principales riesgos de seguridad incluyen la seguridad de los datos y la protección de la privacidad. ***Los proveedores de servicios de Cloud Computing deben ofrecer medidas de seguridad*** como el cifrado de datos, el control de acceso y la copia de seguridad para garantizar la seguridad de los datos de los clientes. [7]

1. Cloud Computing vs On-Premise: Se refiere a las soluciones que se instalan en la propia empresa, en servidores y software propios. En la mayoría de casos de sistemas pequeños, el Cloud Computing en comparación con el On-premise tiene ventaja en ahorro de costos, escalabilidad, seguridad y movilidad.

2. Proveedores de Cloud Computing

   a) Amazon Web Services (AWS): Es el proveedor de servicios de computación en la nube más grande del mundo.

   b) Google Cloud Platform (GCP): Ofrece una amplia gama de servicios de computación en la nube, incluyendo Google App Engine, Google Compute Engine y Google Cloud Storage.

   c) Microsoft Azure: Es una plataforma de computación en la nube que ofrece una amplia gama de servicios, incluyendo infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS).

5.5. Proyectos Similares

En el ámbito de la búsqueda y comparación de componentes electrónicos, actores como **Octopart** [40], **Digi-Key** [37], **Mouser** [38] y **Arrow** [39] desempeñan un papel crucial al proporcionar plataformas en línea que permiten a los usuarios acceder a una amplia gama de información sobre productos electrónicos, incluyendo especificaciones técnicas, precios y disponibilidad. Estas plataformas actúan como intermediarios entre los fabricantes de componentes y los clientes finales, facilitando así el proceso de búsqueda y adquisición de componentes para nuestros proyectos de ingeniería y desarrollo.

Las similitudes entre estos actores radican en su objetivo principal: proporcionar una fuente centralizada de información sobre componentes electrónicos y permitirnos a los usuarios buscar y comparar diferentes opciones de manera eficiente. Todas estas plataformas ofrecen una amplia selección de productos de múltiples fabricantes y proveedores, y su funcionalidad principal radica en la capacidad de búsqueda y filtrado para encontrar los componentes específicos que necesitamos para nuestros proyectos.

Sin embargo, lo que distingue a nuestro proyecto es nuestro enfoque en la inclusión de proveedores locales. Mientras que las plataformas mencionadas se centran principalmente en conectar a los usuarios con proveedores globales, nuestra aplicación busca ampliar esta oferta al permitir que los proveedores locales publiquen sus productos en nuestra plataforma. Esto proporciona una ventaja significativa al ofrecernos acceso a una variedad más amplia de opciones, incluyendo productos de fabricantes locales que pueden ofrecer beneficios como tiempos de entrega más rápidos, soporte técnico localizado y precios competitivos.

Por lo cual, aunque las plataformas existentes como Octopart, Digi-Key, Mouser y Arrow ofrecen servicios valiosos en la búsqueda y comparación de componentes electrónicos, nuestro proyecto se diferencia al enfocarse en la inclusión de proveedores locales. Esta estrategia ofrece una ventaja competitiva al proporcionarnos una experiencia más personalizada y adaptada a nuestras necesidades y preferencias del mercado local.

Por otro lado, en el contexto de desarrollo de aplicaciones web para gestión de proyectos, se encontró un caso relevante desarrollado para el Sistema de Investigación, Innovación y Desarrollo Tecnológico (SENNOVA) del SENA en Colombia. Este proyecto comparte similitudes metodológicas y tecnológicas con nuestra propuesta, particularmente en el uso de metodologías ágiles como SCRUM y tecnologías web modernas basadas en JavaScript. La aplicación desarrollada por SENNOVA se enfoca en la gestión integral de proyectos de investigación, implementando módulos para registro, seguimiento y generación de estadísticas, lo cual refleja un enfoque similar al nuestro en términos de centralización y optimización de procesos. Las principales similitudes con nuestro proyecto incluyen: [44]

Uso de metodología SCRUM para desarrollo Implementación de tecnologías web modernas (Vue.js) Desarrollo de aplicación con múltiples módulos funcionales Objetivo de centralizar y optimizar procesos organizacionales

La diferencia principal radica en el contexto específico: mientras el proyecto de SENNOVA se orienta a la gestión de proyectos de investigación, nuestra aplicación se enfoca en la mejora logística del sector de componentes electrónicos. No obstante, ambos proyectos comparten el propósito fundamental de utilizar tecnología web para resolver necesidades organizacionales específicas.

Finalmente se encontró un último caso similar pero enfocado al desarrollo de una aplicación web de comercio electrónico basada en tecnologías modernas como ReactJS y Firebase, en respuesta a la creciente demanda de soluciones tecnológicas que optimicen la experiencia de compra en línea. La propuesta incluye funcionalidades clave como búsqueda, filtrado, carrito de compras, integración de pasarelas de pago, y recomendaciones personalizadas basadas en el historial de compra del usuario. Estos elementos buscan mejorar la usabilidad y seguridad y dan como base un stack tecnológico y arquitectura como referencia que puede ser útil para nuestra actual propuesta de tesis. [32]

Capítulo 6

Metodología

En respuesta al problema de dispersión del mercado electrónico, costos de importación y poca visibilidad de proveedores locales, se desarrolló un prototipo web que agrupa información de disponibilidad, normaliza datos de fuentes heterogéneas y ofrece búsqueda, comparación y mecanismos de colaboración (p. ej. tableros para coordinar compras).

El desarrollo de la aplicación se realizó siguiendo un enfoque modular, priorizando la escalabilidad, la mantenibilidad del código y la integración entre los distintos componentes del sistema. Para ello se definieron criterios de diseño basados en principios de arquitectura moderna como separación de responsabilidades y desacoplamiento entre capas, alineados con las recomendaciones vigentes en ingeniería de software [45]. Esta sección documenta lo que se hizo para construir y validar el prototipo, incluyendo los diagramas que guiaron la implementación.

6.1. Diagramas de arquitectura y de datos utilizados en la implementación

Durante el diseño y la construcción del sistema se elaboraron diagramas que permitieron fijar criterios comunes entre backend, datos y despliegue. La Figura 6.1 resume la arquitectura modular del servicio backend; la Figura 6.2 muestra el modelo entidad–relación implementado en MySQL para soportar inventario, usuarios, tableros colaborativos y relaciones con el motor de búsqueda.

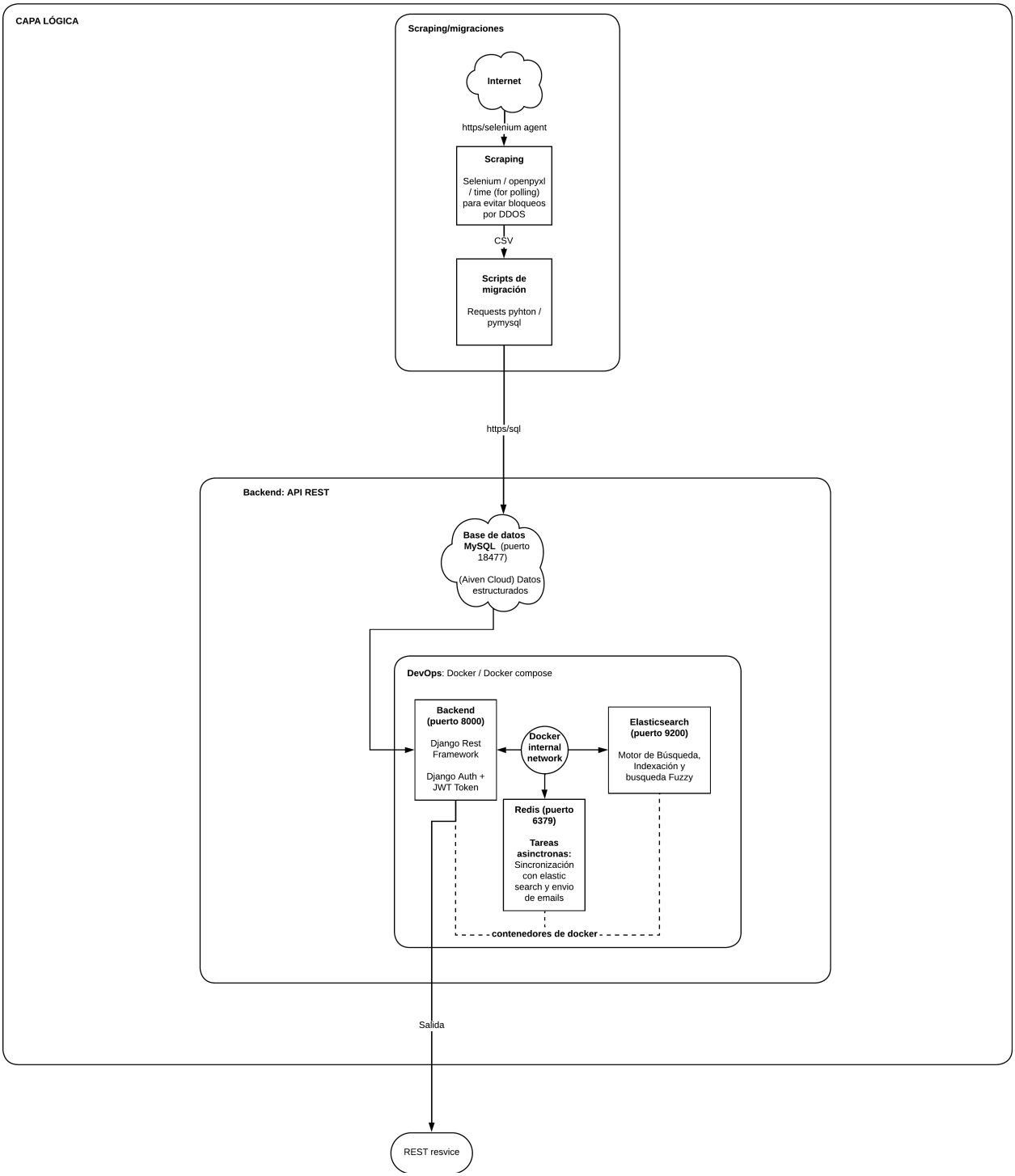


Figura 6.1: Diagrama de arquitectura modular del backend [2].

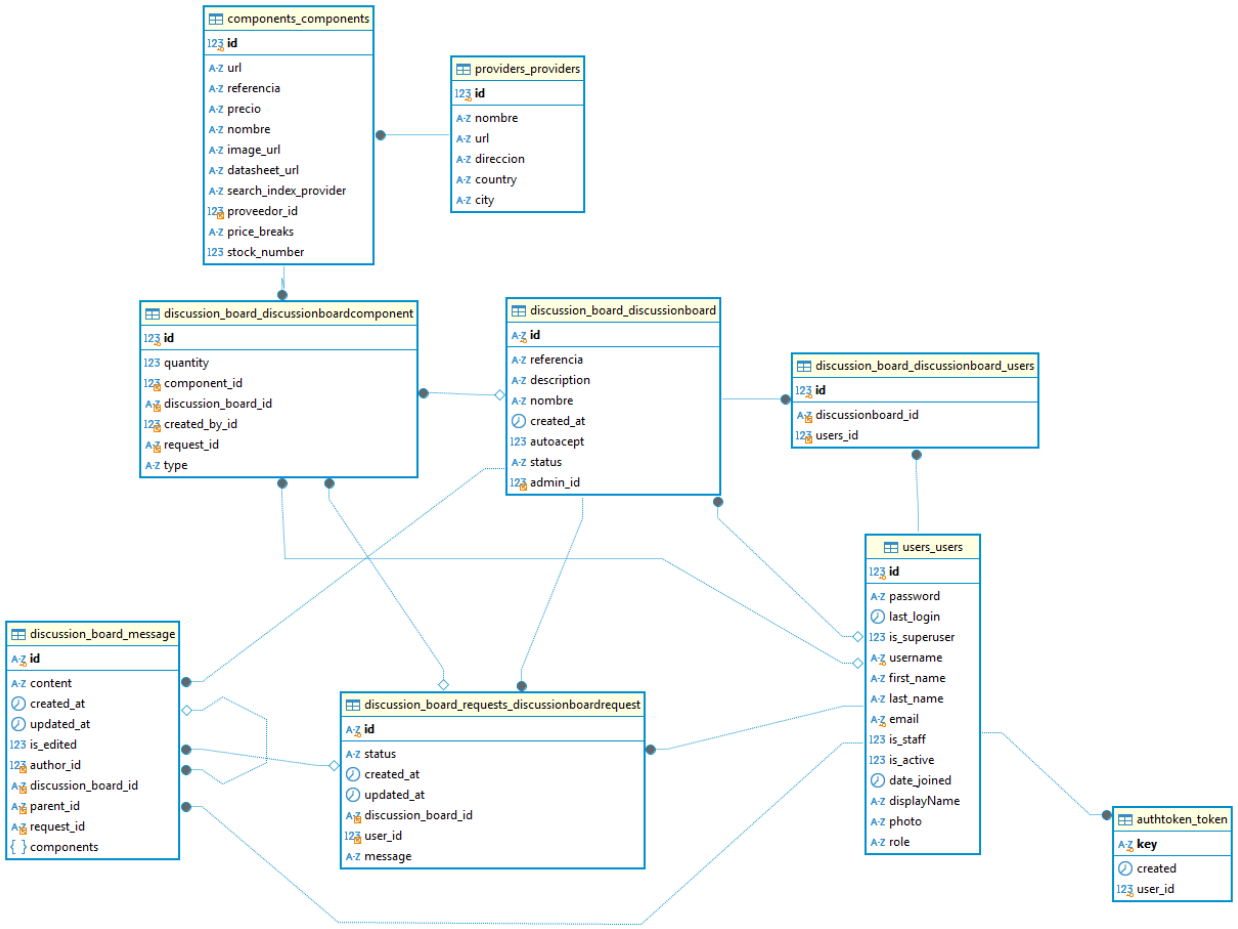


Figura 6.2: Diagrama entidad–relación del modelo de datos implementado.

6.2. Metodología aplicada al desarrollo del prototipo

El proyecto se ejecutó como un proceso de ingeniería de software orientado al problema de gestión logística del abastecimiento de componentes electrónicos. A diferencia de un anteproyecto, lo que sigue resume **las actividades que se llevaron a cabo**, en el orden en que estructuraron el trabajo.

Actividades realizadas

1. **Caracterización del problema y de la solución software.** Se identificaron las características que debía tener la aplicación web para apoyar la gestión logística frente a la disponibilidad y variedad de componentes en el mercado.

   a) Se investigó el mercado de componentes electrónicos y la oferta de proveedores locales e

internacionales.

b) Se compararon plataformas existentes y se delimitaron funcionalidades que la aplicación debía cubrir.

c) Se priorizaron áreas de mejora tecnológica abordables con una aplicación web.

d) **Diseño de diagramas de flujo orientados al problema.** Tras el análisis previo se fijó un flujo objetivo en tres pasos—**búsqueda/selección de componentes → tableros colaborativos → consolidación de pedidos**—que conecta requisitos con el diseño de pantallas.

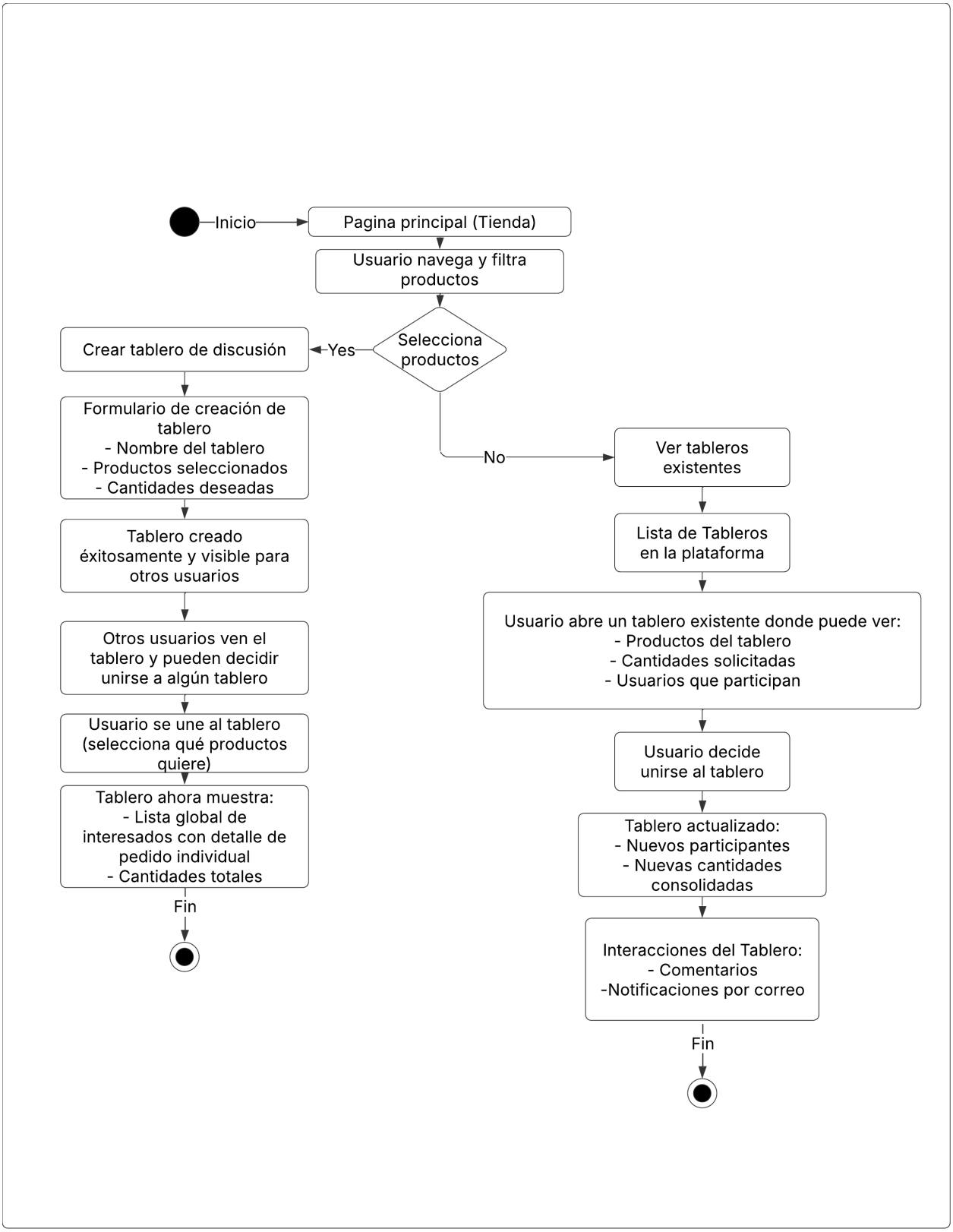


Figura 6.3: Diseño del flujo objetivo del usuario en la aplicación web.

La Figura 6.3 resume ese recorrido para coordinar pedidos en conjunto y cerrar con solicitudes agrupadas que reducen fricción logística.

2. **Diseño de arquitectura y selección tecnológica.** Se definió una arquitectura que soportara los requerimientos funcionales y no funcionales (rendimiento, SEO, accesibilidad, mantenibilidad).

   a) Se contrastaron tecnologías de desarrollo web con documentación y comunidad activa.

   b) Las decisiones de despliegue, persistencia, recolección automatizada de datos y presentación web se fundamentaron en la Sección 6.4 y en las tablas multicriterio 6.2 a 6.7, alineadas con los requisitos del prototipo y con la viabilidad en tiempo del trabajo de grado.

   c) Se tuvo en cuenta el **tiempo de desarrollo** estimado de adoptar un stack u otro (curva de aprendizaje, configuración inicial, integración entre servicios y pruebas), de modo que la elección no dependiera solo del desempeño teórico, sino también de la **viabilidad en el calendario** del trabajo de grado.

   d) Se valoró la **experiencia previa** del equipo con las tecnologías candidatas: en la práctica se priorizaron herramientas que ya se dominaban (p. ej. ecosistema Python/Django en backend y React/Next.js en frontend), lo que redujo riesgos de retrabajo y aceleró la implementación frente a opciones igualmente válidas en papel pero menos conocidas para el grupo.

   e) Se fijaron los diagramas de la sección anterior como referencia de implementación para el equipo.

3. **Implementación incremental por épicas.** Se construyó el prototipo operativo con la arquitectura elegida.

   a) Se montó el entorno de desarrollo y despliegue con contenedores.

   b) Se organizó el trabajo en épicas que agrupaban flujos completos de negocio (búsqueda, tableros colaborativos, autenticación, integración con proveedores, etc.).

   c) Se desarrollaron y validaron las épicas de forma secuencial, comprobando en cada ciclo el cumplimiento de los requisitos asociados al flujo entregado.

4. **Evaluación del prototipo.** Se midió el resultado frente a criterios técnicos y de uso.

   a) Se aplicaron auditorías con Lighthouse sobre el frontend (rendimiento, accesibilidad, buenas prácticas y SEO).

   b) Se realizaron pruebas con usuarios del sector y encuestas para recoger percepción de usabilidad y utilidad.

6.3. Investigación del mercado y comparativa

Como parte de la caracterización del dominio (primer bloque de la metodología), se revisó el papel de los agregadores y distribuidores globales más utilizados en ingeniería electrónica. **Octopart** funciona principalmente como motor de búsqueda y comparación de fichas técnicas y existencias entre múltiples fabricantes y distribuidores [40]. **Digi-Key**, **Mouser** y **Arrow** operan como catálogos de alto volumen con logística internacional, fuerte cobertura de marcas globales y flujos de compra orientados a importación y a pedidos que no siempre se adaptan a montos bajos o a plazos cortos en Colombia [37, 38, 39]. Esas fortalezas son valiosas para cubrir el *long tail* del mercado electrónico: muchas referencias minoritarias o de nicho más allá de los pocos códigos de muy alto volumen, cuya relevancia proviene de agrupar en catálogo un abanico enorme de demanda dispersa [46]. Aun así, tienden a **omitir** un eje explícito: visibilizar y ordenar la oferta de **proveedores locales**, facilitar **compras conjuntas** y reducir la **fricción logística nacional** para pymes, laboratorios y formación. La Tabla 6.1 resume la comparación que orientó el diseño funcional del prototipo.

Criterio	**Plataformas globales (Octopart, Digi-Key, Mouser, Arrow)**	**Plataforma desarrollada (con foco en Colombia)**
Rol frente al usuario	Catálogo/búsqueda masiva y comparación de fichas y stock a escala mundial; la compra suele cerrarse con distribución internacional.	Búsqueda unificada con intención de **acercar** inventarios nacionales y extranjeros en un solo flujo de decisión de compra.
Proveedores locales	No es el valor central; la oferta local compite en visibilidad con el catálogo global.	Se priorizó dar cabida a **proveedores nacionales** y a datos que reflejen tiempos y condiciones de abastecimiento en el país.
Logística y escala de compra	Costos y plazos de importación; mínimos y procesos pensados para volúmenes mayores.	Mecanismos de **agrupación** (p. ej. tableros colaborativos) para repartir carga logística y acercar compras de menor escala a condiciones más viables.
Intermediación	Enlace a distribuidores globales; poca curación de contexto local.	Énfasis en **comercio electrónico local**: reducir fricción entre oferta nacional y demanda académica o industrial ligera.
Funcionalidades colaborativas	No son el núcleo del modelo de negocio de estos actores.	Se incorporaron flujos de **discusión, solicitudes y coordinación** alrededor de pedidos, alineados con compras conjuntas.

Cuadro 6.1: Comparación entre plataformas globales de referencia y el enfoque de la solución desarrollada.

En síntesis, la investigación de mercado concluyó que las plataformas globales resuelven bien la **información técnica y el acceso al catálogo planetario**, pero dejan un vacío para Colombia en

impulso al comercio local, **visibilidad de proveedores nacionales** y **herramientas de coordinación logística** entre pares. El prototipo se orientó a cubrir ese vacío sin pretender reemplazar el inventario mundial, sino **complementarlo** con mayor **cercanía y articulación nacionales**, aspectos que los grandes agregadores no explicitan en su propuesta de valor.

6.4. Selección tecnológica

Lo que sigue presenta, **por capas**, la comparación multicriterio de alternativas. En cada tabla se define un conjunto de criterios con **pesos** w_i (suma igual a 1,00), se asigna a cada tecnología candidata una **puntuación** s_{ij} en escala del 1 (muy desfavorable) al 5 (muy favorable) y se calcula la **puntuación ponderada** $\sum_i w_i s_{ij}$ (máximo teórico 5,00). Los pesos reflejan la prioridad del trabajo de grado: viabilidad en tiempo, costo académico, adecuación al dominio y reproducibilidad. Las matrices quedaron registradas en las Tablas 6.2 (despliegue), 6.3 (recolección con navegador), 6.4 (API), 6.5 (persistencia), 6.6 (búsqueda) y 6.7 (interfaz).

6.4.1. Selección de tecnologías de despliegue e integración

La Tabla 6.2 resume la evaluación entre **Docker + Docker Compose**, un **despliegue tradicional** (servidores o VM, proxy inverso y configuración manual) y **Kubernetes**. La opción con mayor puntuación ponderada se adoptó para el prototipo.

Criterio	Peso	Docker + Compose	Despliegue tradicional	Kubernetes
Reproducibilidad del entorno entre máquinas	0,25	5	2	4
Simplicidad operativa para el calendario del TG	0,25	5	3	2
Facilidad de red interna entre microservicios	0,20	5	3	5
Costo de licencias (enfoque académico)	0,15	5	5	5
Preparación para escalado futuro en producción	0,15	3	3	5
Puntuación ponderada ($\sum w_i s_{ij}$)	**1,00**	**4,70**	**3,05**	**4,00**

Cuadro 6.2: Evaluación multicriterio de opciones de despliegue e integración (escala 1–5 por celda). [3, 4, 5, 6, 7, 8]

6.4.2. Selección de tecnologías para recolección de datos

La Tabla 6.3 cubre la rama de automatización con **navegador real** (sitios con renderizado dinámico). Las peticiones HTTP con `requests` y parseo HTML liviano se reservaron cuando no hace falta ejecutar un navegador completo.

Criterio	Peso	Selenium	Playwright	Puppeteer
Alineación con experiencia previa del equipo (Python)	0,30	5	3	2
Costo de licencias / stack libre	0,20	5	5	5
Adecuación a sitios con mucho JavaScript	0,25	4	5	5
Documentación y ejemplos para el caso de uso	0,15	5	4	3
Rendimiento y consumo de recursos del crawler	0,10	3	5	4
Puntuación ponderada	**1,00**	**4,55**	**4,25**	**3,70**

Cuadro 6.3: Evaluación multicriterio de herramientas de automatización de navegador para recolección de datos (escala 1–5). [9, 10, 11, 12, 13, 14]

6.4.3. Selección de tecnologías de backend

Marco para API y lógica de negocio

La Tabla 6.4 contrasta **Django + Django REST Framework**, **FastAPI** y **Spring Boot**.

Criterio	Peso	Django + DRF	FastAPI	Spring Boot
Integración ORM, migraciones y autenticación	0,25	5	3	4
Tiempo estimado de adopción en el TG	0,25	5	4	2
Madurez de convenciones REST y comunidad	0,20	5	4	5
Costo de licencias / ecosistema abierto	0,15	5	5	5
Rendimiento en escenarios altamente concurrentes	0,15	3	5	4
Puntuación ponderada	**1,00**	**4,70**	**4,05**	**3,85**

Cuadro 6.4: Evaluación multicriterio de marcos para API backend (escala 1–5). [15, 16, 17, 18, 19]

Sistema gestor de base de datos

La Tabla 6.5 compara **MySQL**, **PostgreSQL** y **MongoDB**.

Criterio	Peso	MySQL	PostgreSQL	MongoDB
Adecuación al modelo de datos relacional del dominio	0,35	5	5	2
Disponibilidad de hosting académico / gratuito	0,30	5	4	5
Madurez de transacciones ACID para inventario y pedidos	0,20	5	5	4
Flexibilidad de esquema frente a cambios exploratorios	0,15	3	4	5
Puntuación ponderada	**1,00**	**4,70**	**4,55**	**3,75**

Cuadro 6.5: Evaluación multicriterio de sistemas de persistencia (escala 1–5). [20, 21, 22, 23, 24]

Motor de búsqueda de texto completo

La Tabla 6.6 compara **Elasticsearch**, **Apache Solr** y **Meilisearch**.

Criterio	Peso	Elasticsearch	Solr	Meilisearch
Búsqueda textual, filtros y tolerancia a errores tipográficos	0,30	5	5	4
Documentación y ejemplos de integración	0,25	5	4	3
Costo operativo y complejidad de despliegue para el TG	0,20	4	3	5
Costo de licencias / capa abierta usable en el prototipo	0,15	4	5	5
Encaje con indexación incremental desde MySQL	0,10	5	4	3
Puntuación ponderada	**1,00**	**4,65**	**4,25**	**4,00**

Cuadro 6.6: Evaluación multicriterio de motores de búsqueda (escala 1–5). [25, 26, 27]

6.4.4. Selección de tecnologías de frontend

La Tabla 6.7 contrasta **Next.js** (React con SSR/SSG integrados) con una **SPA con React y Vite** y con **Angular** (enfoque basado en TypeScript, componentes y enrutador propio, con posibilidad de SSR según configuración del proyecto) [31], en un contexto de catálogo con muchas rutas y necesidad de SEO.

Criterio	Peso	Next.js	React SPA (Vite)	Angular
Soporte nativo a SSR/SSG y rutas tipo catálogo	0,30	5	2	4
SEO y primer pintado sin integraciones ad hoc	0,25	5	3	4
Tiempo estimado de adopción en el TG	0,25	5	5	3
Costo de licencias / ecosistema abierto	0,10	5	5	5
Estructuración modular de la interfaz (componentes y rutas)	0,10	5	5	5
Puntuación ponderada	**1,00**	**5,00**	**3,60**	**3,95**

Cuadro 6.7: Evaluación multicriterio de enfoques de frontend (escala 1–5). [28, 29, 30, 31, 32]

6.5. Diseño y desarrollo del backend

El sistema adoptó principios del patrón **Modelo–Vista–Controlador (MVC)** de manera distribuida, en línea con las prácticas habituales para aplicaciones web con backend robusto y frontend independiente [2]. Las tablas 6.4, 6.5 y 6.6 documentan la selección multicriterio del marco de API, del sistema gestor de base de datos y del motor de búsqueda. De esta forma, el *modelo* se implementó en el backend mediante Django REST Framework y su ORM [16], el *controlador* mediante los *viewsets* y los serializadores expuestos en la API REST, y la *vista* correspondió al frontend desarrollado en React, encargado de la interfaz de usuario y la visualización de los datos [29].

6.5.1. Criterios de diseño y arquitectura

El backend se construyó siguiendo principios de modularidad y separación estricta de responsabilidades, apoyándose en el estilo arquitectónico REST. Entre los criterios empleados se destacan:

- **Separación de responsabilidades**: división clara entre lógica de negocio, acceso a datos, serialización y endpoints REST. [45]
- **Estandarización**: uso del protocolo REST como convención de comunicación [47].
- **Escalabilidad**: integración con servicios externos como Elasticsearch [25].
- **Automatización de despliegue**: contenedorización completa utilizando Docker [3].

6.5.2. Base de datos y gestión de migraciones

Como sistema de gestión de bases de datos se utilizó **MySQL**, alojado en un proveedor cloud gratuito denominado **Aiven** [24]. La decisión se alinea con la comparación de la Tabla 6.5 y con la documentación del producto [20]. Esta elección se justificó además por disponibilidad, integración

con servicios externos y compatibilidad con Django [15].

La administración de migraciones se llevó a cabo utilizando el módulo nativo de Django `django.migrations`, el cual permite definir, versionar y aplicar cambios estructurales sobre el esquema de forma controlada, siguiendo las recomendaciones oficiales del framework para la evolución incremental de modelos [15].

Adicionalmente, se desarrollaron **scripts en Python** [48] para realizar migraciones de datos desde archivos Excel hacia la base de datos, empleando librerías como `pandas` [49] para el procesamiento estructurado de la información. Estas herramientas resultaron esenciales para cargar catálogos iniciales de componentes electrónicos y normalizar datos provenientes de proveedores.

6.5.3. Indexación y búsqueda con Elasticsearch

Para mejorar la experiencia de búsqueda dentro de la plataforma, se implementó un motor de búsqueda basado en **Elasticsearch** [25], en coherencia con la evaluación de la Tabla 6.6. Este sistema permitió:

- realizar búsquedas de alta velocidad,
- aplicar indexación avanzada para componentes electrónicos,
- integrar tolerancia a errores tipográficos mediante *fuzzy search*.

La indexación se construyó mediante un proceso automatizado que sincroniza los registros provenientes de la base de datos MySQL hacia los índices de Elasticsearch.

6.5.4. Web scraping y adquisición de datos

El sistema dependió de la recopilación de información proveniente de sitios de diferentes proveedores de componentes electrónicos. La elección entre automatización de navegador y peticiones HTTP directas se fundamentó en la Tabla 6.3 y en la Sección 6.4. En la implementación se emplearon dos enfoques complementarios:

- **Web scraping dinámico con Selenium** [9], en sitios donde el contenido se renderiza mediante JavaScript o requiere interacción del usuario.
- **Peticiones HTTP y parseo con Python**, mediante librerías como `requests` y parseo HTML cuando procedía [12, 13], cuando la respuesta HTML era suficiente sin ejecutar un navegador completo.

Estos scripts permitieron obtener datos como precios, disponibilidad, características técnicas y enlaces a fichas de producto.

6.5.5. Contenedorización con Docker

Todos los servicios del backend incluyendo Django, MySQL y Elasticsearch— se montaron con **Docker** y **Docker Compose** [50, 4], en línea con la decisión de la Tabla 6.2. En la práctica esto aportó:

- entornos reproducibles,
- aislamiento entre servicios y resolución por nombre en la red interna de contenedores [5],
- facilidad de despliegue y portabilidad entre máquinas de desarrollo y servidores [3].

6.6. Diseño y desarrollo del frontend

El frontend fue desarrollado empleando **Next.js** [28], un framework de React conocido por ofrecer renderizado del lado del servidor (SSR) y renderizado del lado del cliente (CSR) según las necesidades de la aplicación. La motivación frente a una SPA exclusivamente en React y frente a un enfoque Angular [31] se resume en la Tabla 6.7. En conjunto, SSR/SSG y el enrutado del framework permitieron optimizar el tiempo de carga inicial, mejorar el SEO y acercar la experiencia a la de un catálogo electrónico con muchas páginas generadas a partir de datos del backend.

6.6.1. Criterios de diseño y usabilidad

El diseño del frontend se rigió por:

- **Experiencia de usuario (UX)**: interfaz clara para búsquedas, navegación y visualización de componentes.
- **Rendimiento**: uso de SSR para páginas críticas, y CSR para componentes interactivos.
- **Estilado visual**: uso de Tailwind CSS [51].
- **Accesibilidad**: validación con herramientas como *Lighthouse* [52].

6.6.2. Diseño del flujo objetivo del usuario

En la fase de diseño del frontend se definió el **flujo objetivo del usuario**: secuencia deseada de pantallas y decisiones (búsqueda y consulta de componentes, autenticación, tableros colaborativos y acciones asociadas al pedido) que el prototipo debía soportar con claridad y pocas desviaciones. Ese recorrido sirvió de guía para priorizar rutas en Next.js, estados de carga y mensajes de error antes de implementar el detalle visual. La Figura 6.3 resume dicho diseño; las capturas del capítulo de resultados muestran la materialización de ese flujo en la interfaz construida.

6.6.3. Estructura del frontend

Las principales tecnologías utilizadas incluyen:

- **React Hooks**: manejo de estado, efectos, formularios y lógica compartida [53].
- **Next.js SSR/CSR**: híbrido entre contenido pre-renderizado y contenido interactivo.
- **TailwindCSS**: sistema de estilos utilitario que permitió un desarrollo rápido y mantenible.
- **Fetch API**: utilizada para consumir los endpoints REST expuestos por Django [54].

6.6.4. Manejo de sesiones y autenticación

En lo referente a la autenticación y la gestión de sesiones, se implementó un sistema basado en JSON Web Tokens (JWT), pero apoyado en el mecanismo estándar de *Django Authentication* [55] para la validación de credenciales y la generación inicial de sesiones seguras. Este enfoque permitió mantener la solidez del sistema de autenticación nativo de Django, al tiempo que se integraba un esquema tokenizado adecuado para aplicaciones web modernas. El token firmado se almacenó en `localStorage` para persistir entre sesiones del navegador, mientras que Django Auth se encargó del proceso de verificación de usuario, hashing de contraseñas y control interno de seguridad.

Adicionalmente, se incorporaron controles para verificar la caducidad del token, renovar sesiones cuando fuera necesario y proteger rutas privadas tanto en el backend como en el frontend. Estas medidas garantizaron que el acceso a recursos sensibles estuviera debidamente restringido y que las sesiones activas mantuvieran un nivel adecuado de seguridad sin comprometer la experiencia del usuario.

6.6.5. Integración y flujo de datos

La comunicación entre frontend y backend siguió los principios REST, utilizando rutas claramente definidas para operaciones CRUD, búsquedas avanzadas y actualización de datos. Los flujos principales incluyeron:

- autenticación del usuario,
- búsqueda de componentes y tableros de discusión mediante Elasticsearch,
- visualización de especificaciones y disponibilidad, y tabla de precios,
- gestión de tableros de discusión.

El manejo del estado del navegador incluyó el uso de:

- **LocalStorage** para almacenamiento de preferencias,
- **Hooks personalizados** para centralizar la lógica de datos.

6.7. Integración general del sistema

El proceso de integración se desarrolló de manera incremental. Cada servicio fue probado de forma independiente y posteriormente integrado mediante pruebas funcionales y de API. La arquitectura general se construyó bajo una estructura desacoplada en la cual:

- el backend expuso los servicios REST,
- el frontend consumió dichos servicios,
- Elasticsearch actuó como motor de búsqueda,
- MySQL almacenó los datos estructurados,
- Docker orquestó todos los servicios.

6.7.1. Repositorios usados en la implementación

Como soporte de la metodología aplicada, el desarrollo se gestionó en repositorios separados para backend y frontend:

- **Backend:** `https://github.com/SMB1998/logistics`
- **Frontend:** `https://github.com/danielmamian99/gadgetgalaxy`

6.8. Validación del sistema

En primera instancia, se empleó la herramienta *Lighthouse* [52], integrada en Google Chrome, para realizar auditorías automatizadas centradas en cuatro pilares fundamentales: rendimiento, accesibilidad, buenas prácticas y SEO. Estas evaluaciones permitieron identificar oportunidades de optimización en el renderizado de páginas, la carga de recursos, la estructura del DOM y la adaptación a usuarios con limitaciones de accesibilidad. Las métricas proporcionadas por Lighthouse sirvieron como línea base para ajustes iterativos en el frontend, especialmente en lo relacionado con tiempos de respuesta, peso de los componentes visuales y mejora en la semántica HTML utilizada en la interfaz.

Complementariamente, se realizaron pruebas con usuarios pertenecientes al campo de la ingeniería electrónica. Estas pruebas consistieron en la ejecución guiada de escenarios clave de la aplicación, tales como la búsqueda de componentes, la visualización de detalles técnicos, la creación de tableros de discusión y la interacción con las funcionalidades de listas personalizadas. Durante estas sesiones, se registraron observaciones sobre la claridad de la navegación, la facilidad de uso, la coherencia del flujo visual y el tiempo necesario para completar tareas específicas.

Finalmente, se aplicaron encuestas estructuradas a los participantes con el fin de recopilar retroalimentación cualitativa y cuantitativa respecto a su experiencia general con la plataforma. Dichas encuestas incluyeron preguntas sobre usabilidad percibida, utilidad de las funcionalidades, nivel de satisfacción, posibles mejoras y percepción del valor agregado del sistema para el sector de componentes electrónicos. Los resultados de estas encuestas permitieron identificar patrones comunes de uso, dificultades recurrentes y elementos que requerían refinamiento para mejorar la experiencia de usuario.

6.8.1. Aplicación del cuestionario de validación con usuarios

Como parte del proceso metodológico de evaluación, se aplicó un cuestionario a usuarios del sector electrónico, incluyendo estudiantes, desarrolladores y profesionales que interactúan con proveedores de componentes. El objetivo del instrumento fue recopilar evidencia cualitativa y cuantitativa sobre usabilidad, accesibilidad, claridad visual y utilidad percibida del prototipo.

El formulario se diseñó con escalas de 1 a 5, opciones de selección y preguntas cerradas orientadas a medir:

- la frecuencia de dificultades al buscar o comparar componentes en el mercado colombiano;
- la utilidad percibida de una plataforma que centralice inventarios y precios;
- el interés en compras conjuntas para reducir costos logísticos;
- la relevancia de funcionalidades adicionales, como venta o intercambio de componentes no utilizados;
- la facilidad de navegación, la claridad visual y la accesibilidad de la interfaz;
- la intuición en la ubicación de acciones clave;
- la satisfacción general con la experiencia de uso.

Los resultados derivados de este instrumento se analizan en el capítulo de Resultados y se apoyan en las gráficas del cuestionario presentadas en dicho capítulo.

Capítulo 7

Resultados y Discusión

7.1. Resultado final flujos frontend

El flujo principal de la aplicación se estructura en los siguientes pasos, mediante los cuales se atiende el problema de coordinar compras de componentes electrónicos entre varios usuarios. A través de este proceso, los participantes pueden consolidar sus necesidades en un solo tablero de compra y generar un pedido conjunto, obteniendo mejores condiciones económicas tanto por **descuentos por volumen** como por la **reducción de costos de envío** al gestionar una única orden compartida.

Paso 1. Pantalla de inicio y acceso al catálogo. El flujo del frontend inicia cuando el usuario accede al dominio de la aplicación (en este caso, `localhost`). En esta pantalla inicial se visualiza el **catálogo de productos**, el **buscador de componentes** y los **accesos directos para crear un tablero de discusión o consultar los tableros existentes**. Esta vista funciona como punto de entrada principal para la navegación y la ejecución de las acciones más relevantes de la plataforma (ver Figura 7.1).

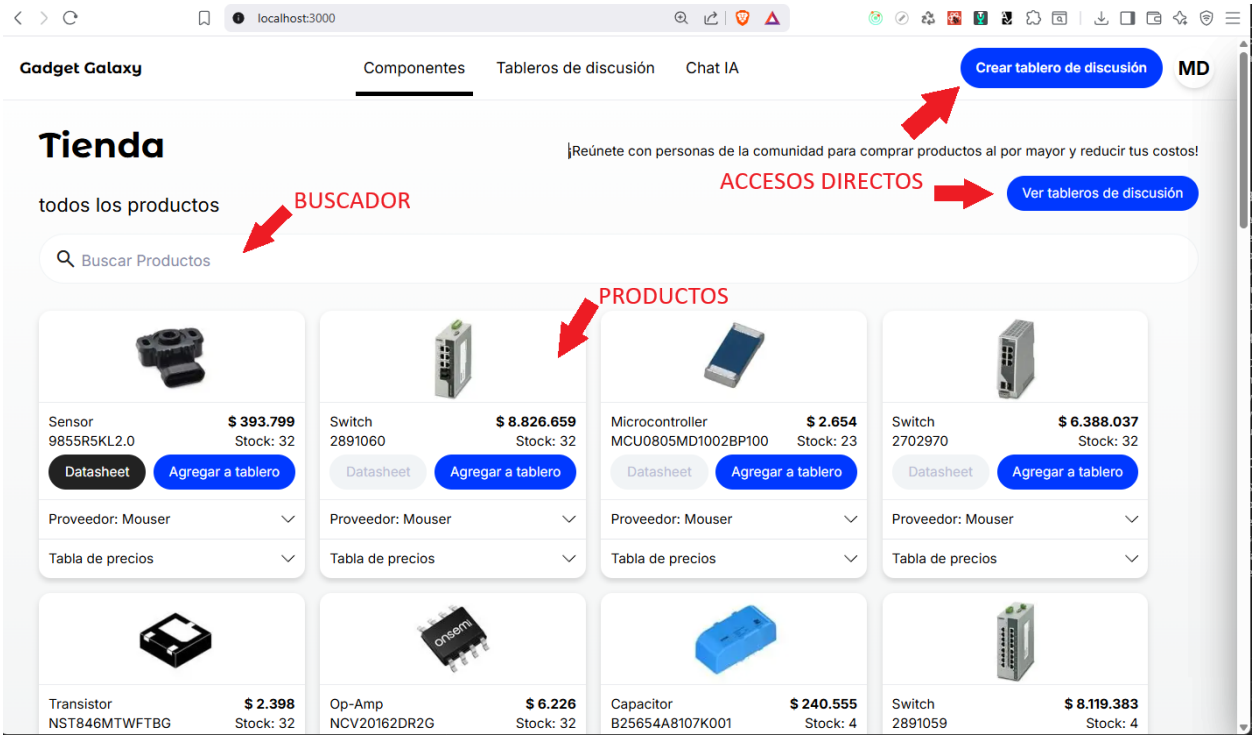


Figura 7.1: Flujo principal: pantalla de inicio con catálogo de productos, buscador y accesos a tableros de discusión

Paso 2. Consulta de productos y selección de interés. Desde la pantalla de inicio, el usuario puede buscar componentes y consultar información detallada de cada producto, incluyendo **datasheet**, **precio** y **cantidad disponible en stock**. También puede acceder directamente al sitio del proveedor para ampliar la información técnica y revisar disponibilidad en el catálogo (Figura 7.3). Cada producto incluye un botón de **Agregar a tablero**: al pulsarlo, el componente se incorpora a una selección provisional, de forma análoga a un carrito, con el fin de marcar los ítems de interés para una compra colaborativa. Posteriormente, al usar la acción **Crear tablero de discusión**, la interfaz despliega un **panel lateral** donde se listan los productos que fueron agregados mediante ese botón, de modo que el usuario revisa y ajusta lo que llevará al tablero antes de completar la creación (ver Figura 7.2).

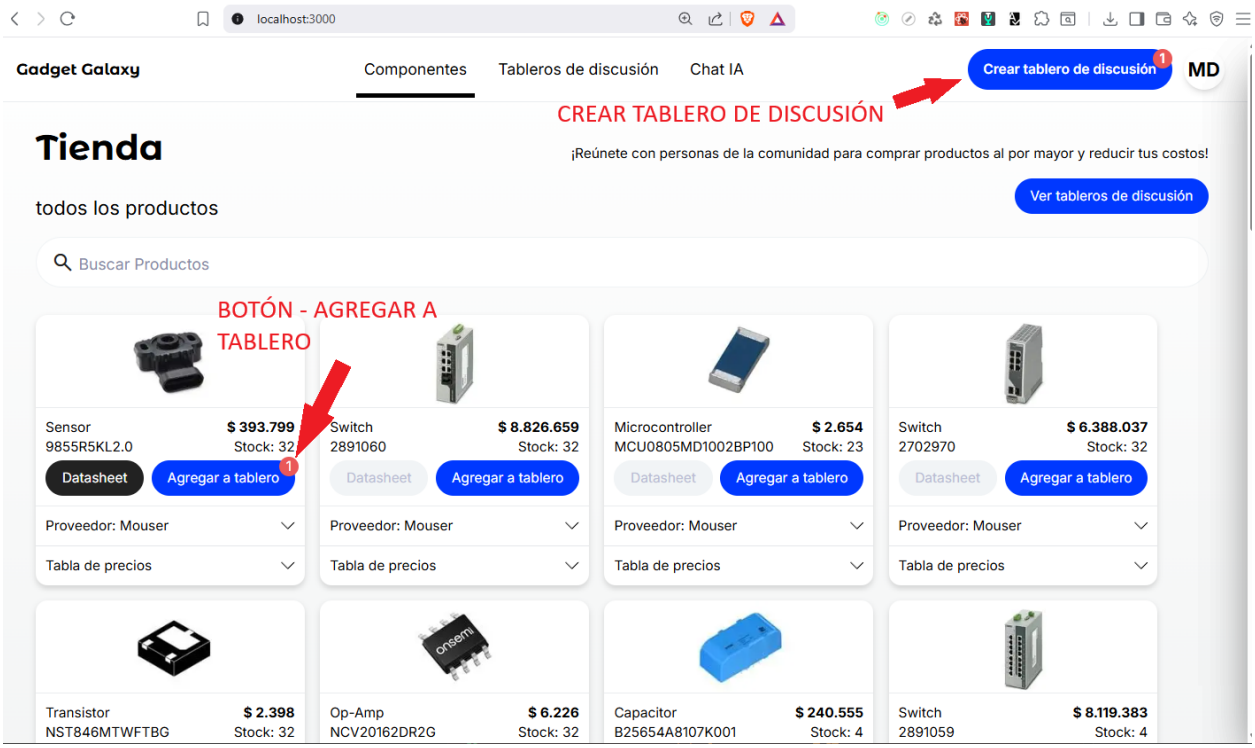


Figura 7.2: Flujo principal: botón Agregar a tablero en los productos y vínculo con el panel lateral al crear tablero de discusión

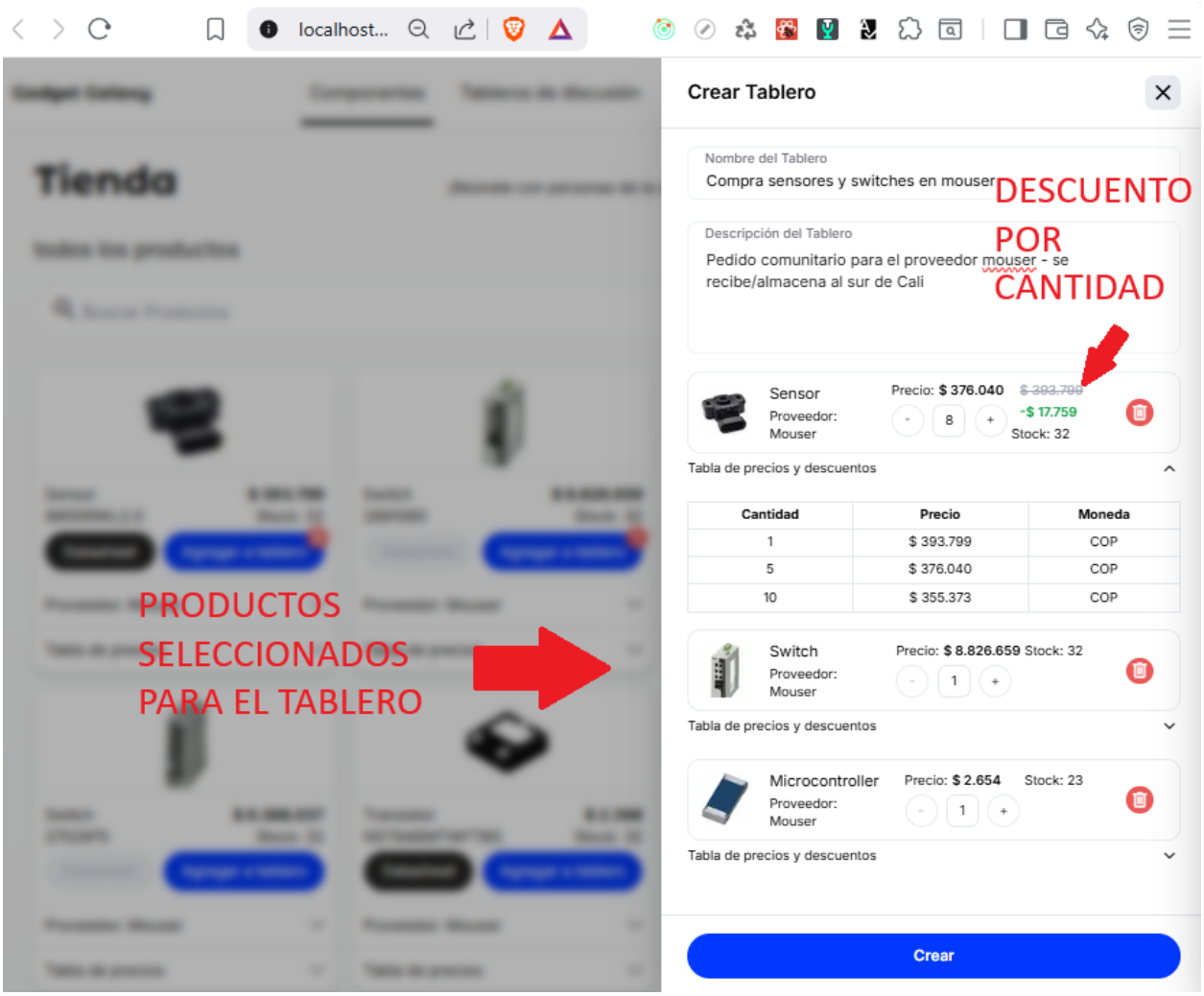


Figura 7.3: Flujo principal: consulta de componentes mediante buscador, catálogo y datos de proveedores

Paso 3. Configuración inicial del tablero. Al agregar componentes y seleccionar la opción de crear tablero, se despliega un panel lateral (slider) donde el usuario puede asignar **nombre** al tablero, una **descripción o tema de discusión**, revisar los productos seleccionados, ajustar cantidades y visualizar descuentos aplicables. En la Figura 7.4 se resaltan mediante una flecha los **campos de entrada** correspondientes al nombre del tablero y al texto de discusión.

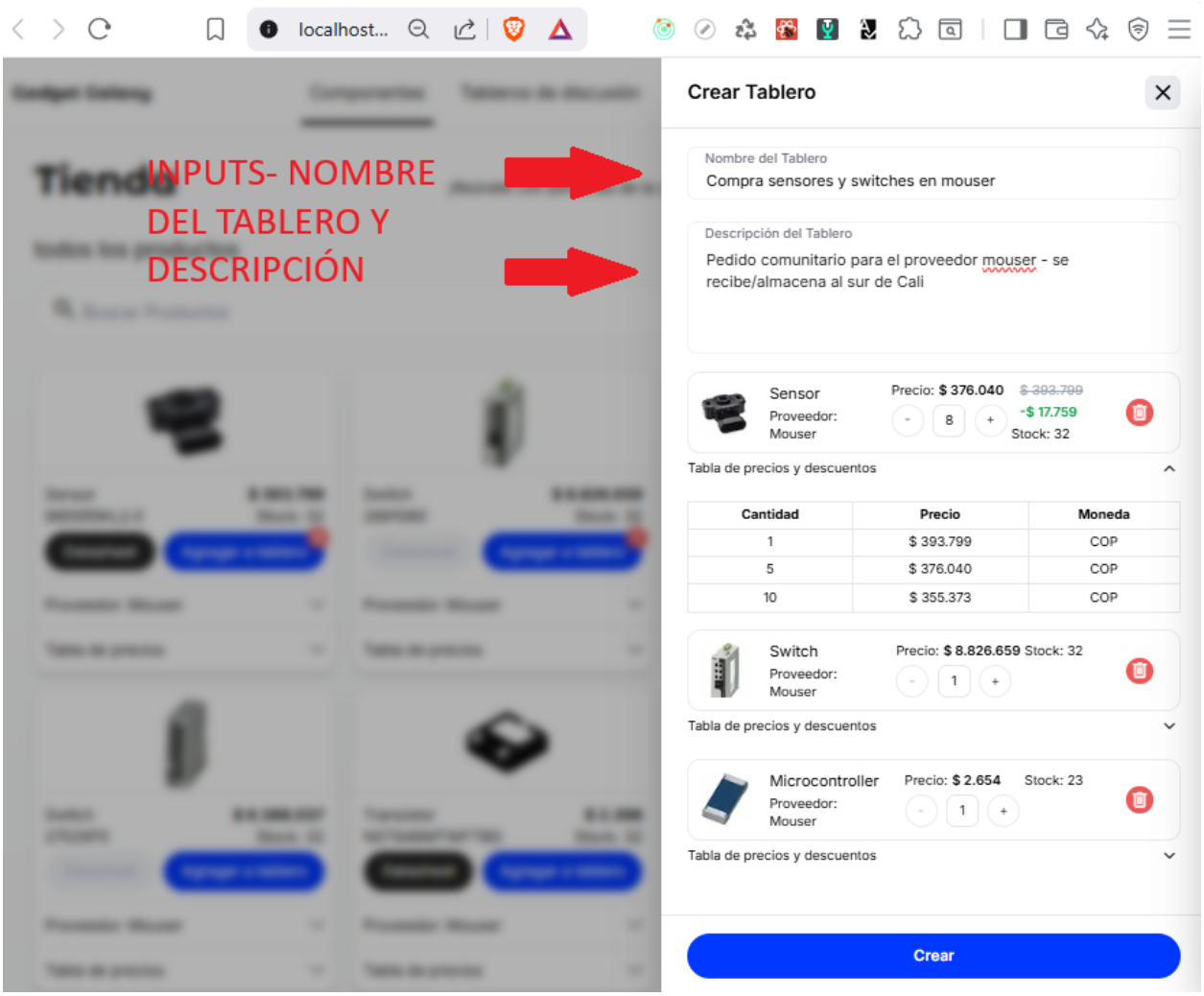


Figura 7.4: Flujo principal: detalle del panel de creación con indicación de los campos para nombre del tablero y descripción de la discusión

Paso 4. Validación de autenticación. Si el usuario no ha iniciado sesión, la interfaz presenta la solicitud de credenciales o registro para continuar con el flujo de creación y participación en tableros (Figura 7.5).

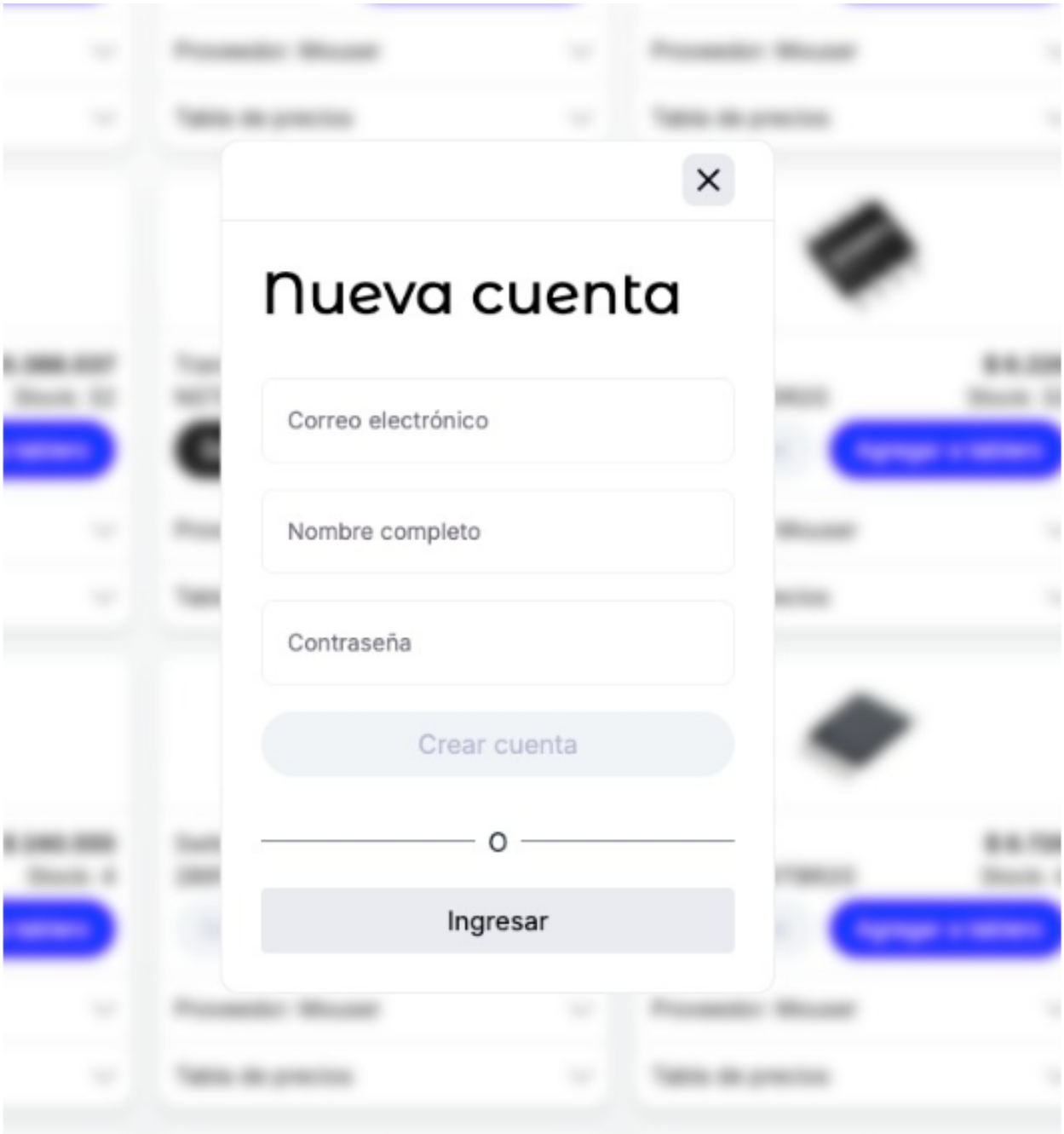


Figura 7.5: Flujo principal: autenticación requerida para continuar en el flujo colaborativo

Paso 5. Creación del tablero y vista principal. Una vez creado el tablero, el usuario accede a su vista principal, donde se visualizan los componentes agregados y la sección de interacción colaborativa (Figura 7.6).

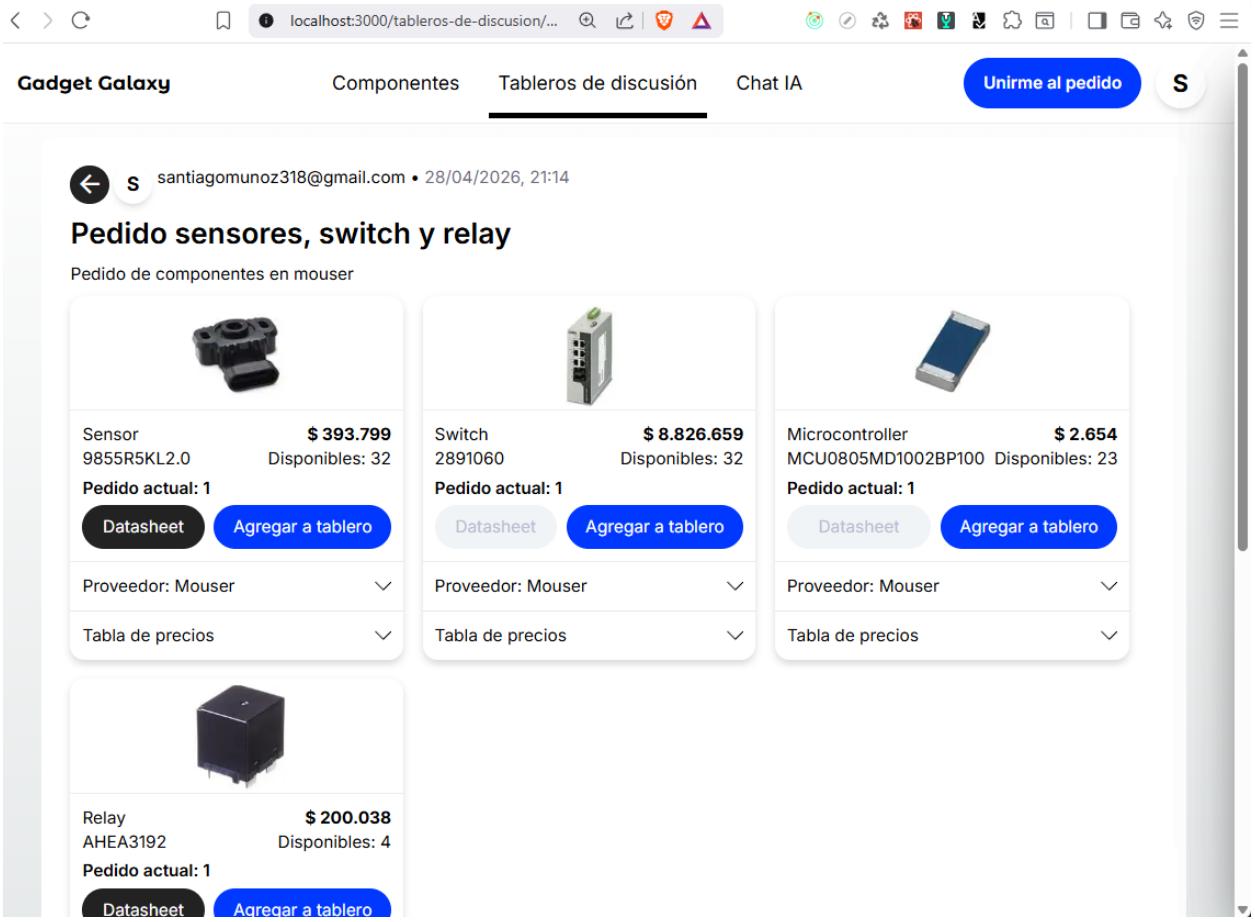


Figura 7.6: Flujo principal: vista principal del tablero con productos y espacio de interacción

Paso 6. Búsqueda de tableros disponibles. La plataforma permite consultar y ubicar tableros mediante un listado general y la barra de búsqueda (Figura 7.7).

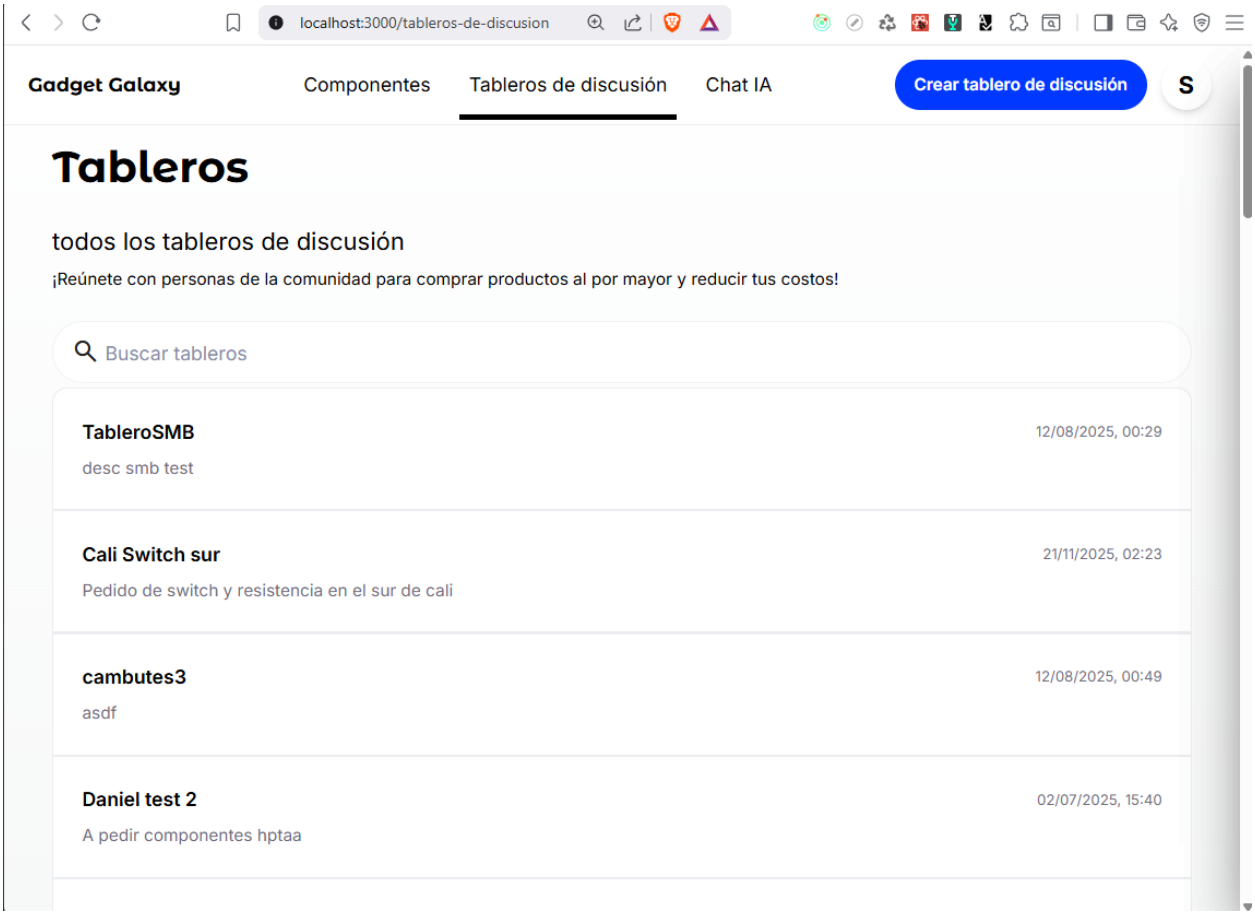


Figura 7.7: Flujo principal: listado y búsqueda de tableros de discusión disponibles

Paso 7. Solicitud de ingreso a tableros existentes. Al seleccionar un tablero, el usuario puede solicitar unirse y proponer componentes o cantidades para el pedido conjunto (Figura 7.8).

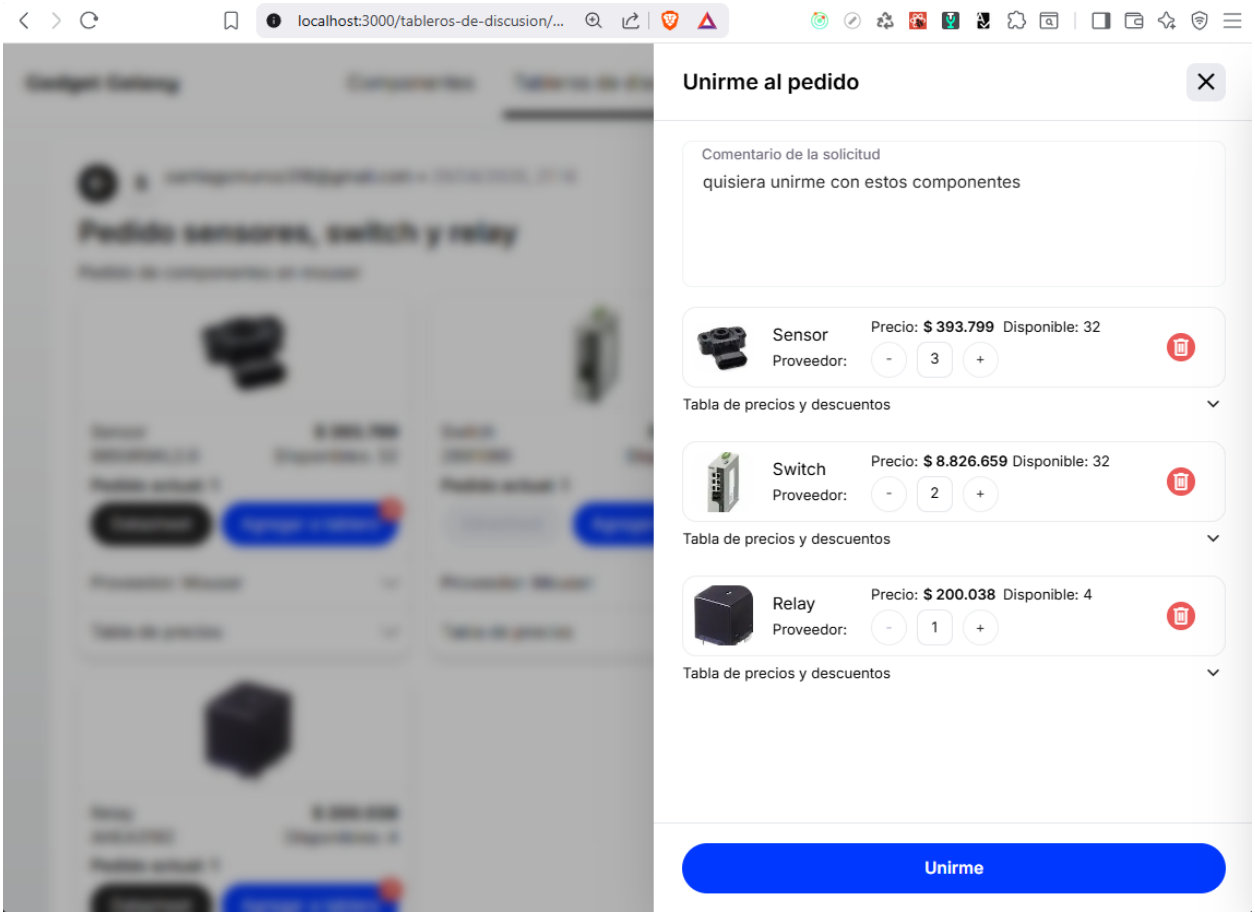


Figura 7.8: Flujo principal: formulario de solicitud de ingreso y propuesta de componentes

Paso 8. Revisión administrativa de solicitudes. Las solicitudes generadas se visualizan en la sección de comentarios, donde el administrador puede aprobarlas o rechazarlas (Figura 7.9).

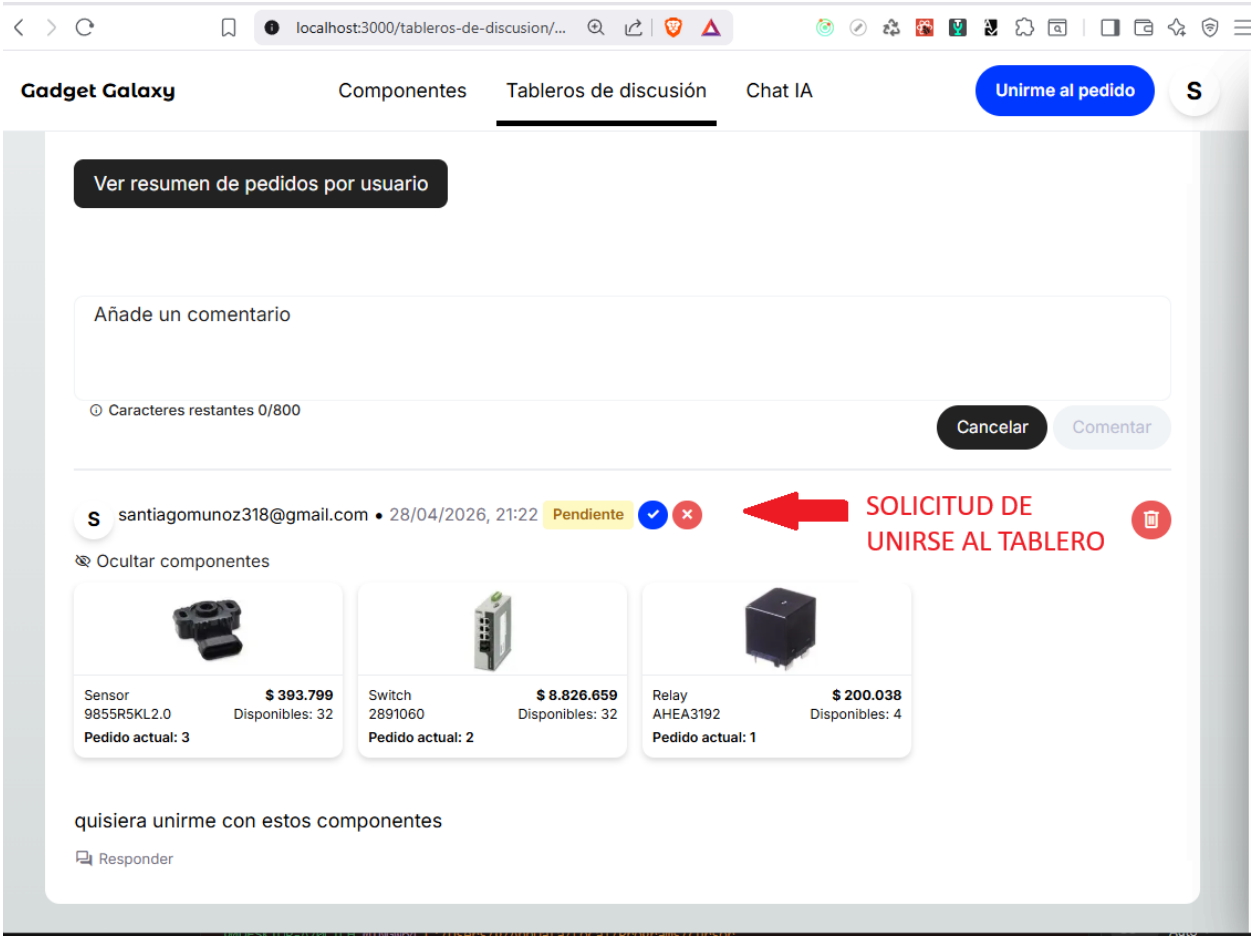


Figura 7.9: Flujo principal: revisión de solicitudes con estado pendiente y decisiones del administrador

Paso 9. Retroalimentación del administrador. El administrador puede registrar comentarios para justificar aprobaciones, rechazos o solicitudes de ajuste (Figura 7.10).

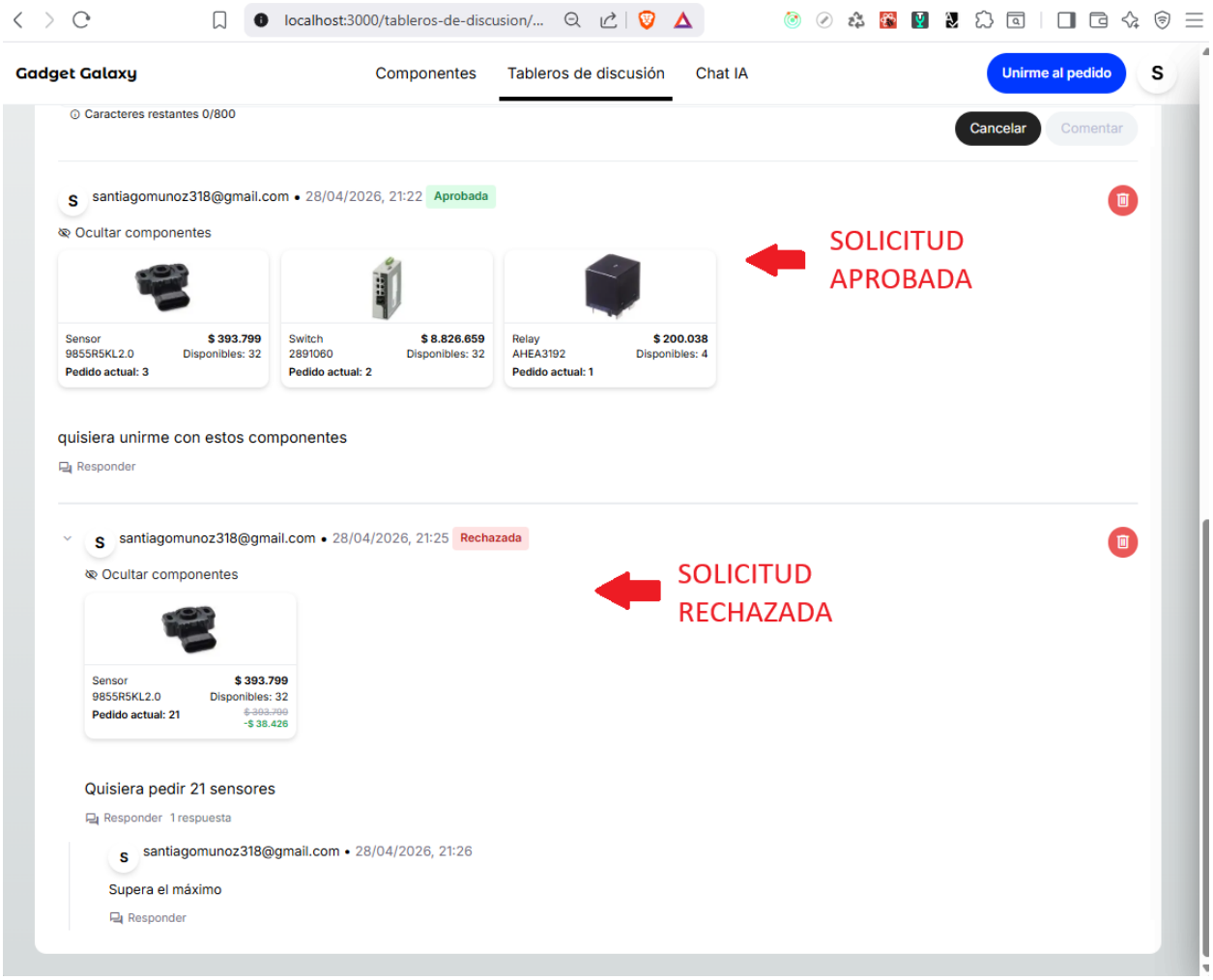


Figura 7.10: Flujo principal: historial de comentarios y decisiones sobre solicitudes de ingreso

Paso 10. Visualización de pedidos aprobados. Tras la aprobación, se consolida el resumen de pedidos por participante dentro del tablero (Figura 7.11).

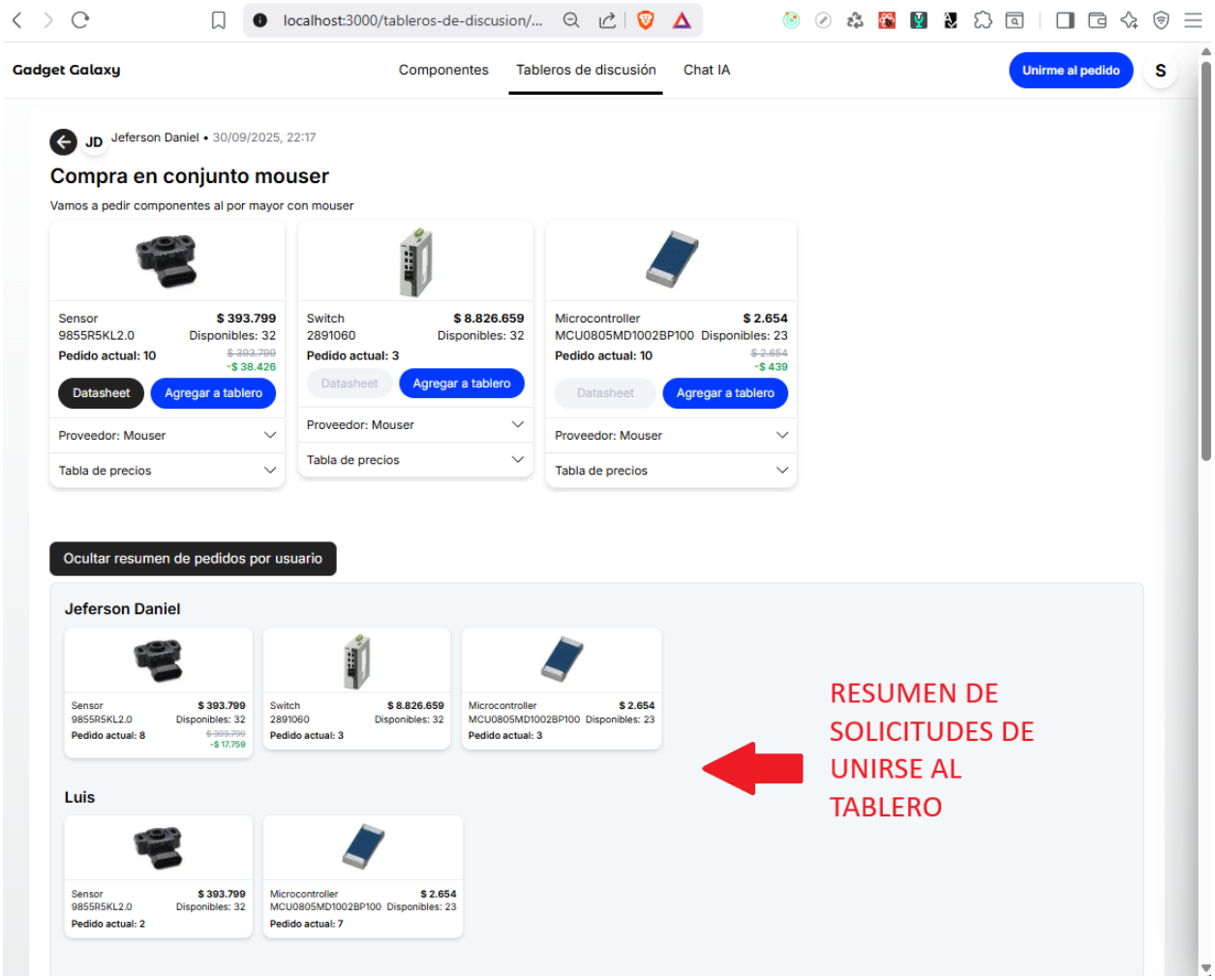


Figura 7.11: Flujo principal: consolidado de pedidos aprobados por participante

Paso 11. Consulta de perfil y trazabilidad de participación. Finalmente, el usuario puede acceder a su perfil para visualizar los tableros en los que ha participado y su historial de actividad (Figura 7.12). Con esta trazabilidad de participación disponible, puede continuar en nuevos tableros, repetir pedidos y mantener seguimiento de su actividad colaborativa.

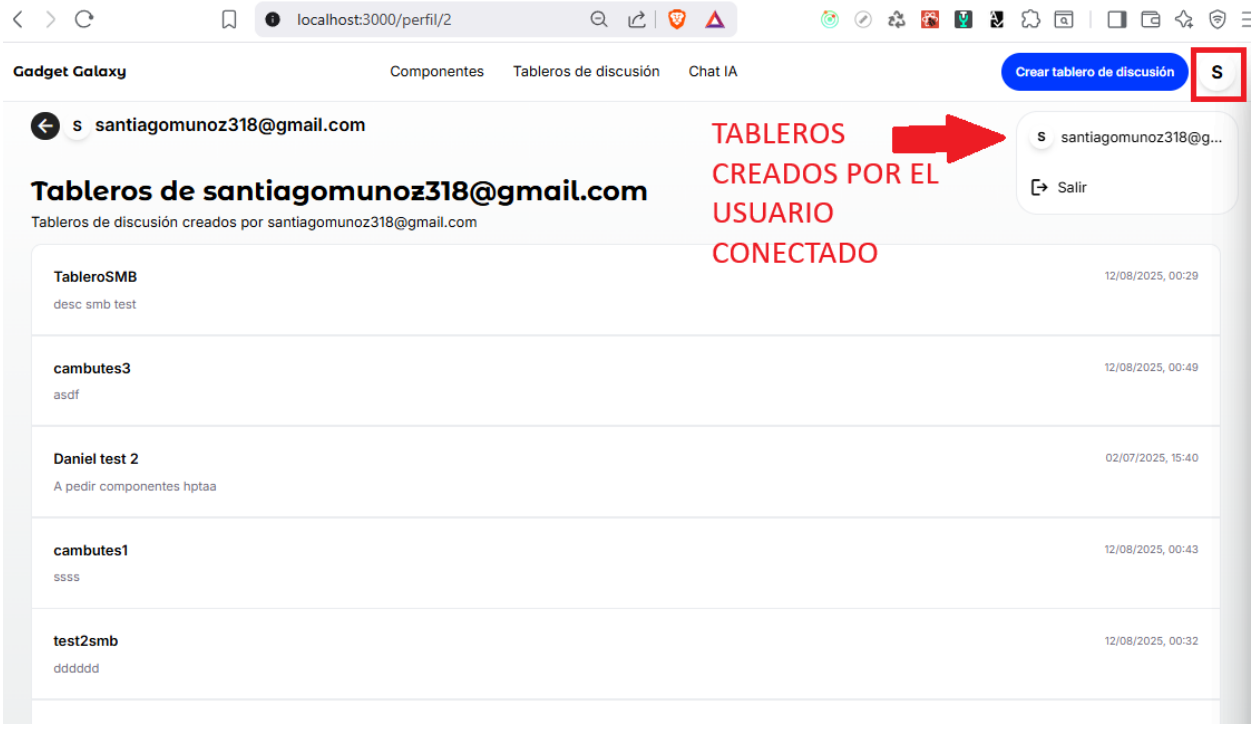


Figura 7.12: Flujo principal: perfil del usuario con listado de tableros y trazabilidad de participación

Recorrido guiado de onboarding

Como complemento al flujo principal, la aplicación incluye un recorrido guiado (*onboarding*) para usuarios nuevos. Este asistente contextual presenta, de forma secuencial, las acciones clave para participar en compras colaborativas: crear tableros, iniciar sesión, consultar tableros activos, agregar productos y leer descuentos por volumen.

En el paso 1 se destaca el botón **Crear tablero de discusión**, explicando su función para organizar compras colaborativas desde la tienda principal (Figura 7.13).

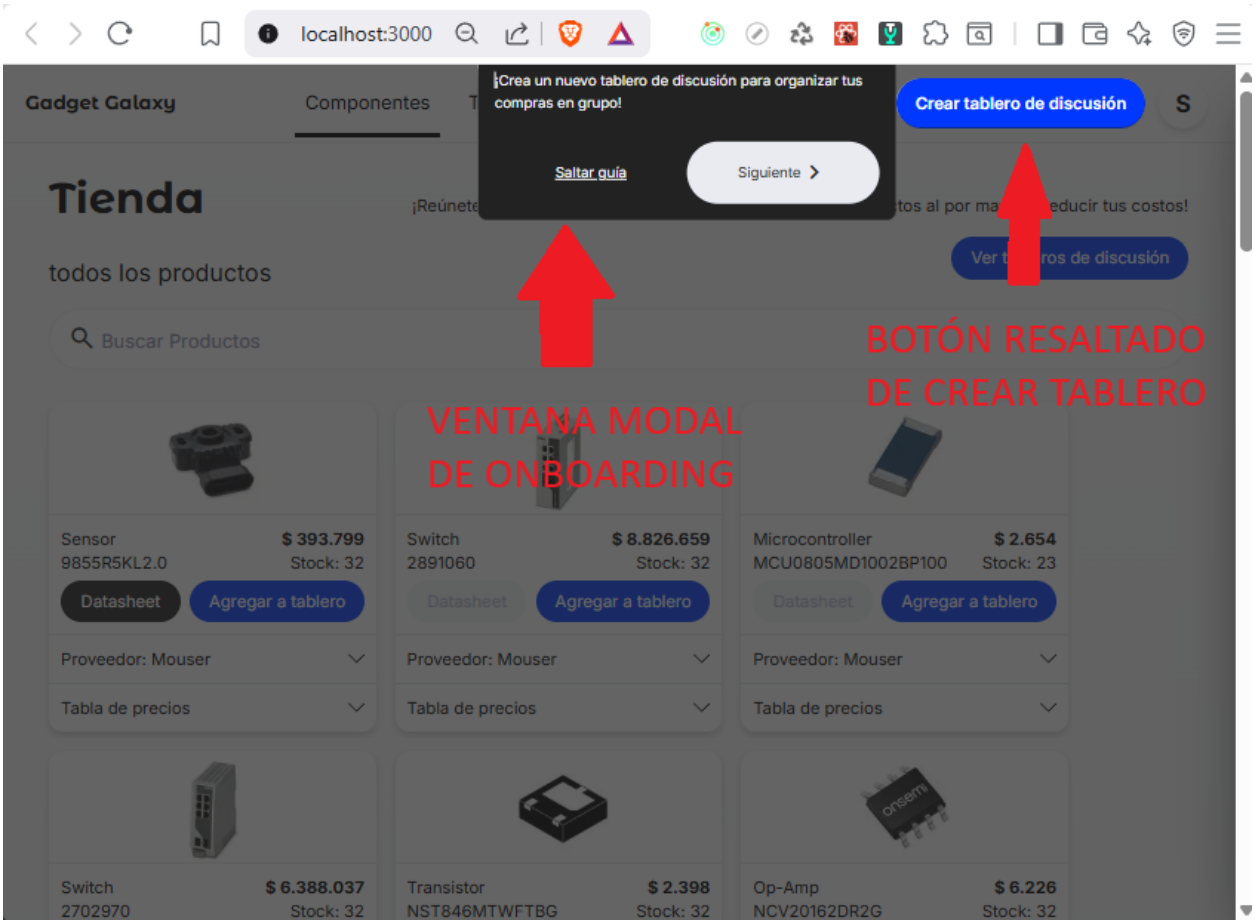


Figura 7.13: Onboarding: introducción al botón para crear tableros de discusión

En el paso 2 se indica la necesidad de **iniciar sesión** para guardar tableros y participar activamente en la comunidad (Figura 7.14).

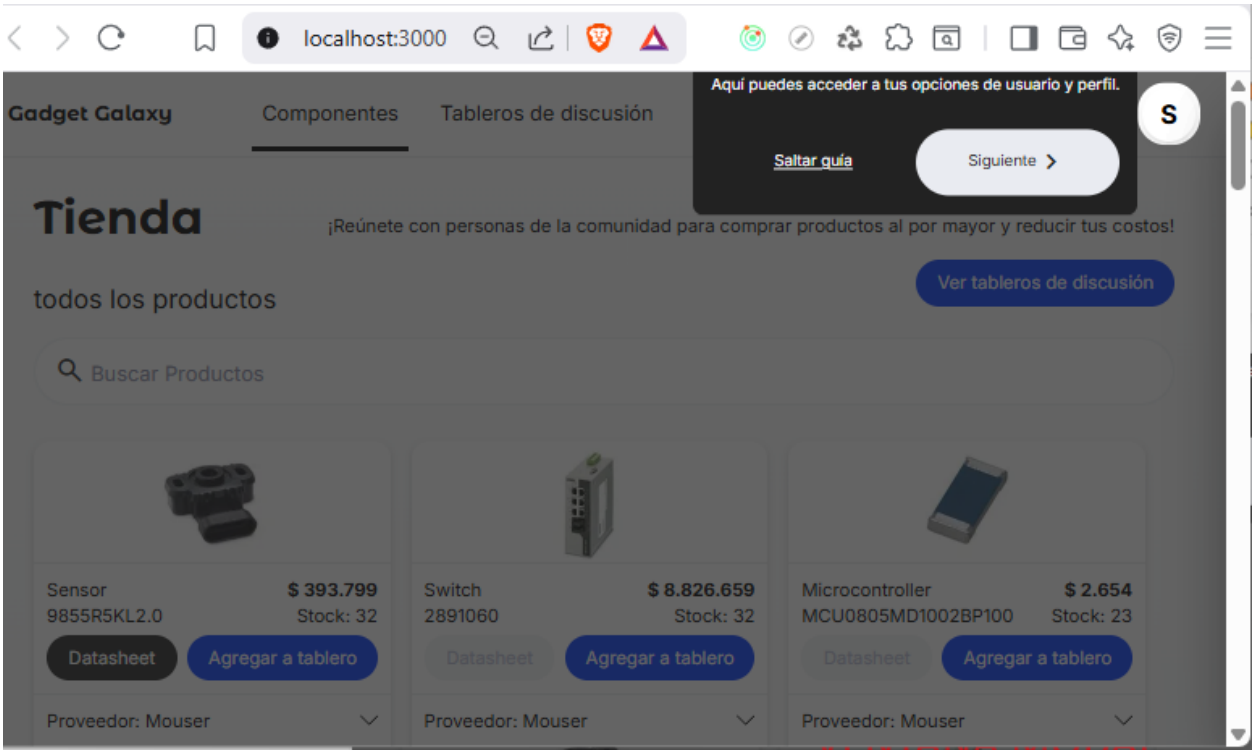


Figura 7.14: Onboarding: guía sobre autenticación e inicio de sesión

En el paso 3 se destaca la pestaña de **tableros de discusión**, desde donde se consultan los tableros activos dentro de la plataforma (Figura 7.15).

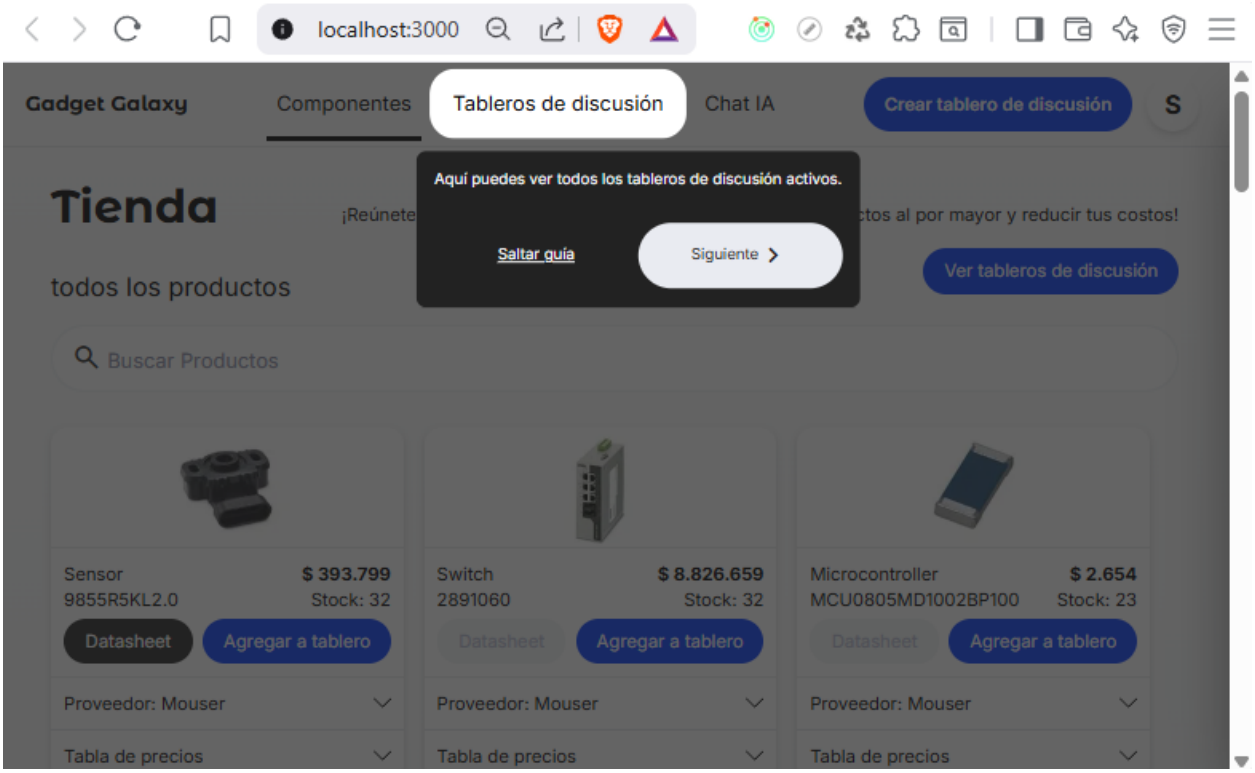


Figura 7.15: Onboarding: acceso a la sección de tableros de discusión activos

En el paso 4 se explica cómo **agregar productos al tablero** para planear compras en grupo (Figura 7.16).

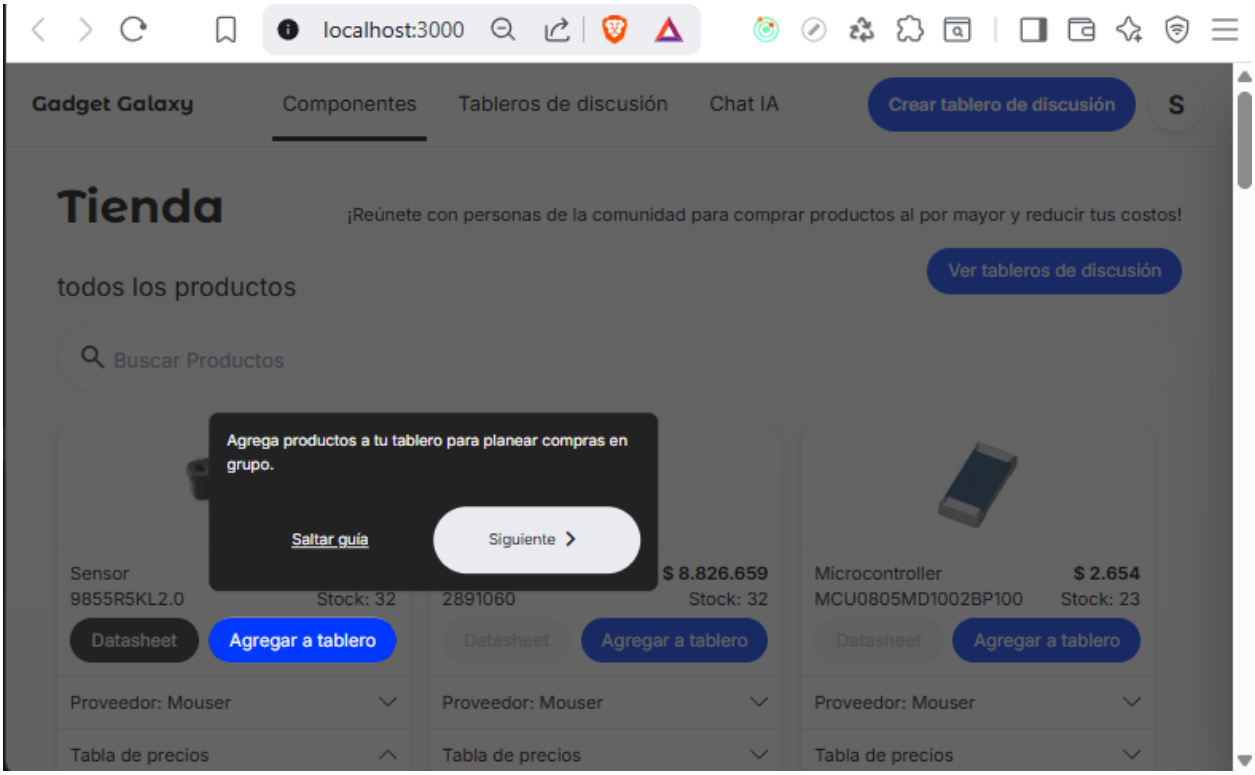


Figura 7.16: Onboarding: procedimiento para agregar productos a un tablero

En el paso 5 se presenta la lectura de **tablas de precios por volumen**, permitiendo identificar descuentos según la cantidad solicitada (Figura 7.17).

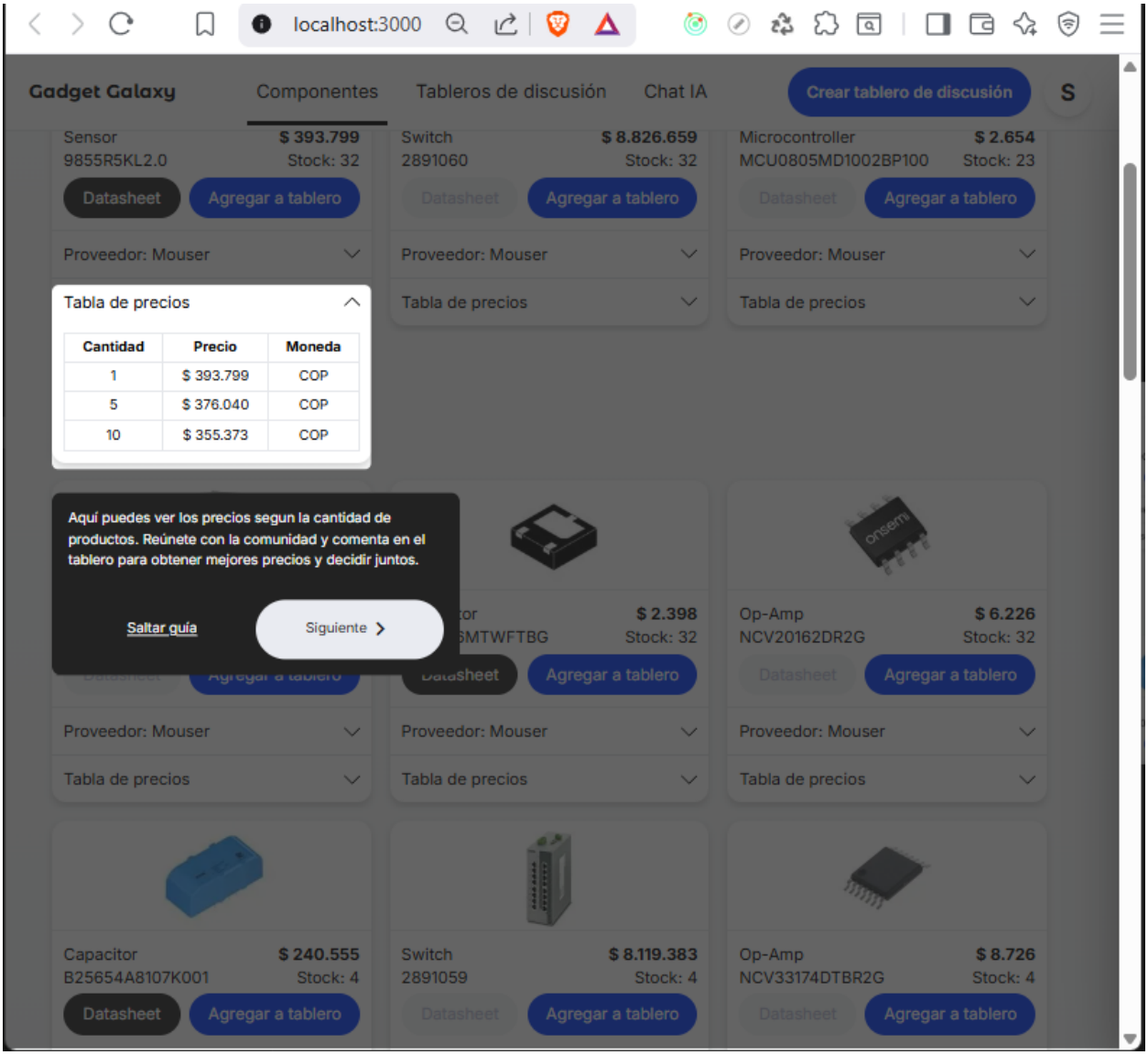


Figura 7.17: Onboarding: lectura de escalas de precios y descuentos por volumen

En el paso 6 finaliza el recorrido guiado y se retorna a la **vista general del catálogo**, dejando al usuario listo para navegar de forma autónoma por la plataforma (Figura 7.18).

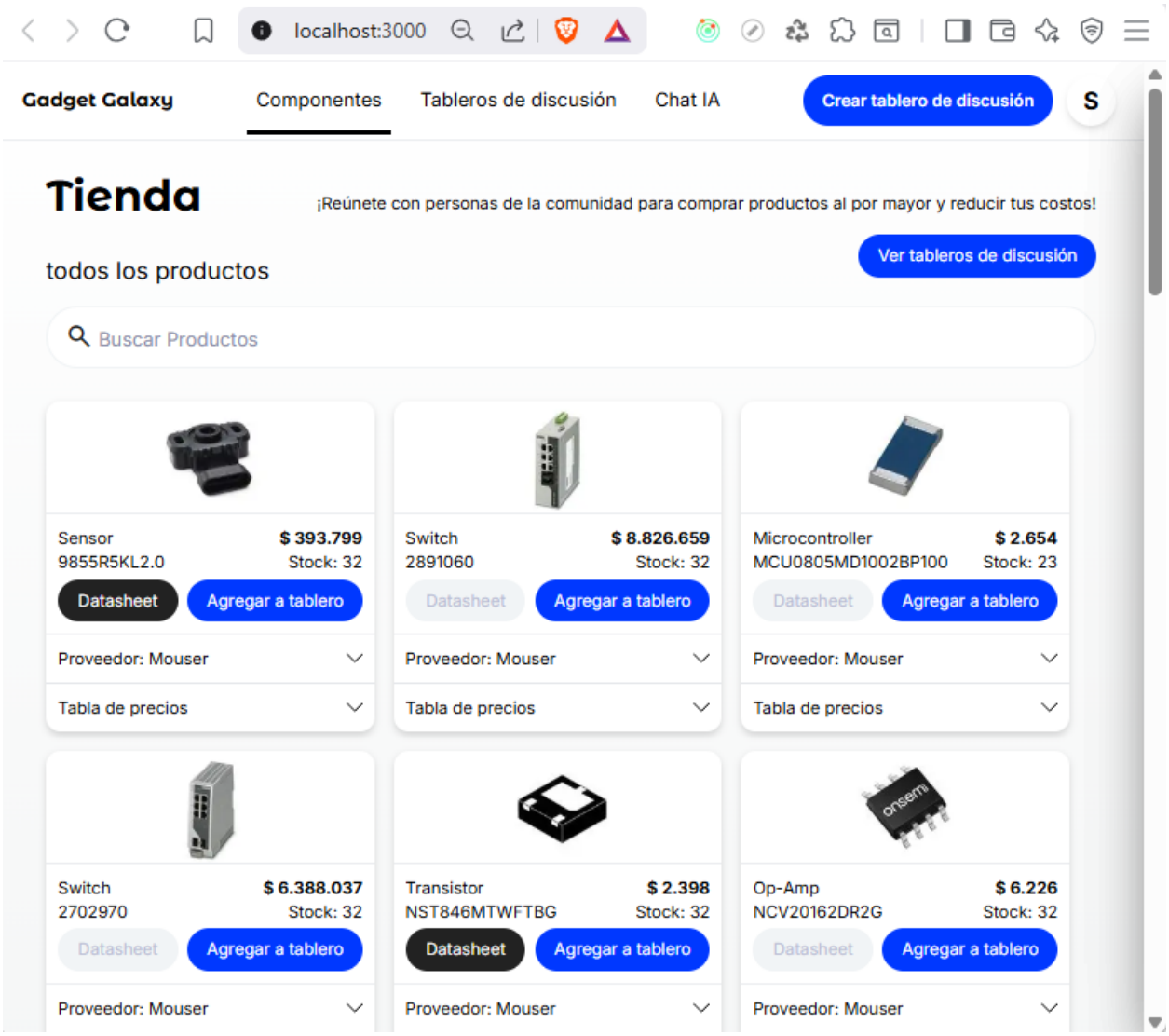


Figura 7.18: Onboarding: cierre del recorrido y vista general del catálogo de componentes

7.2. Análisis de métricas de rendimiento mediante Lighthouse

Con el fin de evaluar el desempeño técnico de la aplicación desarrollada, se realizó un análisis utilizando la herramienta Lighthouse de Google. Los resultados obtenidos evidencian un alto nivel de optimización en los principales indicadores de calidad web.

En la categoría de **Performance**, la aplicación obtuvo una puntuación de 99/100, lo que indica un rendimiento sobresaliente. Las métricas asociadas respaldan este resultado: un First Contentful Paint (FCP) de 0.5 s, un Largest Contentful Paint (LCP) de 0.8 s, un Total Blocking Time (TBT)

de 0 ms y un Cumulative Layout Shift (CLS) igual a 0. Estos valores demuestran tiempos de carga rápidos, ausencia de bloqueos en el hilo principal y estabilidad visual completa durante la renderización.

En cuanto a **Best Practices**, se alcanzó una puntuación de 100/100, lo que confirma el cumplimiento de estándares modernos de desarrollo web, incluyendo buenas prácticas de seguridad, uso adecuado de recursos y correcta configuración técnica.

La categoría de **Accessibility** obtuvo una puntuación de 84/100, lo que indica un nivel adecuado de accesibilidad, aunque con oportunidades de mejora relacionadas posiblemente con contraste de colores, etiquetas ARIA o estructuración semántica del contenido.

Por su parte, el indicador de **SEO** alcanzó una puntuación de 82/100, evidenciando una base sólida en términos de indexación y visibilidad en motores de búsqueda, aunque susceptible de optimización adicional mediante mejoras en metadatos, descripciones y estructuración del contenido.

En conjunto, los resultados obtenidos validan que la arquitectura implementada cumple con los requisitos de rendimiento, estabilidad y buenas prácticas planteados en los objetivos del proyecto, estableciendo una base técnica adecuada para su escalabilidad y futura implementación en entornos productivos.

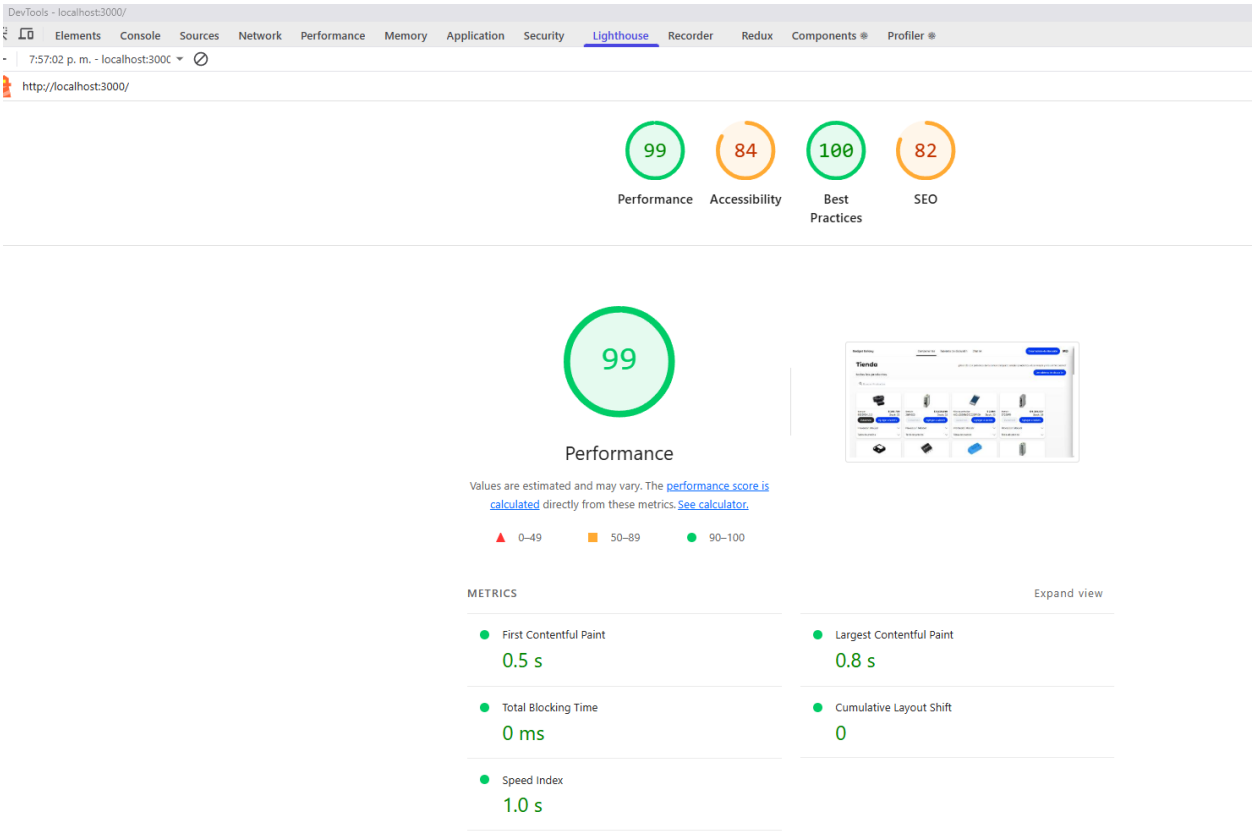


Figura 7.19: Primera evaluación de calidad web mediante Lighthouse

7.2.1. Mejoras respecto a las métricas iniciales

Tras la primera corrida (Figura 7.19), las categorías con mayor brecha frente al rango excelente (90–100) fueron **SEO** (82/100) y **Accesibilidad** (84/100). Con base en los hallazgos y recomendaciones del propio informe de Lighthouse, se implementó un conjunto de ajustes en el frontend (Next.js) y se volvió a auditar la aplicación.

SEO (aproximadamente 80 % → 100 %). Lo central fue sustituir títulos y descripciones genéricos o repetidos por metadatos útiles y alineados con Gadget Galaxy y con el propósito de la tienda. El contenido principal incorpora metadata de Next.js (título y descripción orientados a componentes electrónicos y a la comunidad de usuarios), lo que mejora cómo los buscadores y las redes interpretan la página. En el *layout* raíz se reforzó la capa de descubrimiento y de vista previa al compartir: descripción, Open Graph (título, URL, imagen, tipo de sitio, configuración regional), Twitter Card e indicaciones de indexación para robots y Googlebot. Ese conjunto suele elevar con fuerza la puntuación de SEO en Lighthouse al cubrir metadatos faltantes, *previews* en redes y señales básicas de indexación. Alcanzar 100/100 implica, en la práctica, que desaparecieron los avisos críticos habituales de ese bloque (por ejemplo documento sin meta descripción, *title* débil

o problemas de enlaces sin texto discernible según las reglas de la auditoría aplicada).

Accesibilidad (aproximadamente 80 % → 96 %). El trabajo se orientó a la semántica HTML y a nombres comprensibles para lectores de pantalla y navegación por teclado. En inicio y catálogo se usan regiones `section` con `aria-labelledby` asociadas a un `h1` visible, y una región identificable para el listado de productos. En la paginación, botones y páginas cuentan con `aria-label` descriptivos y `aria-current` en la página activa; los íconos meramente decorativos llevan `aria-hidden` para evitar información redundante. En fichas de producto, bloques plegables usan `aria-expanded` y `aria-controls` para enlazar el control con el panel. Los campos de formulario reciben `aria-label` coherente con la etiqueta visible cuando corresponde; íconos tipográficos se exponen como imagen con `role=ïmg"` y nombre accesible; donde aplica, separadores usan `role="separator"`. El resultado 96/100 y no 100/100 suele deberse a avisos residuales: contraste puntual en texto o bordes, estados de foco en componentes personalizados, orden de lectura dentro de un modal o algún control interactivo cuyo nombre no satisface reglas estrictas (contraste, `tabindex`, jerarquía de encabezados, *landmarks*, etc.).

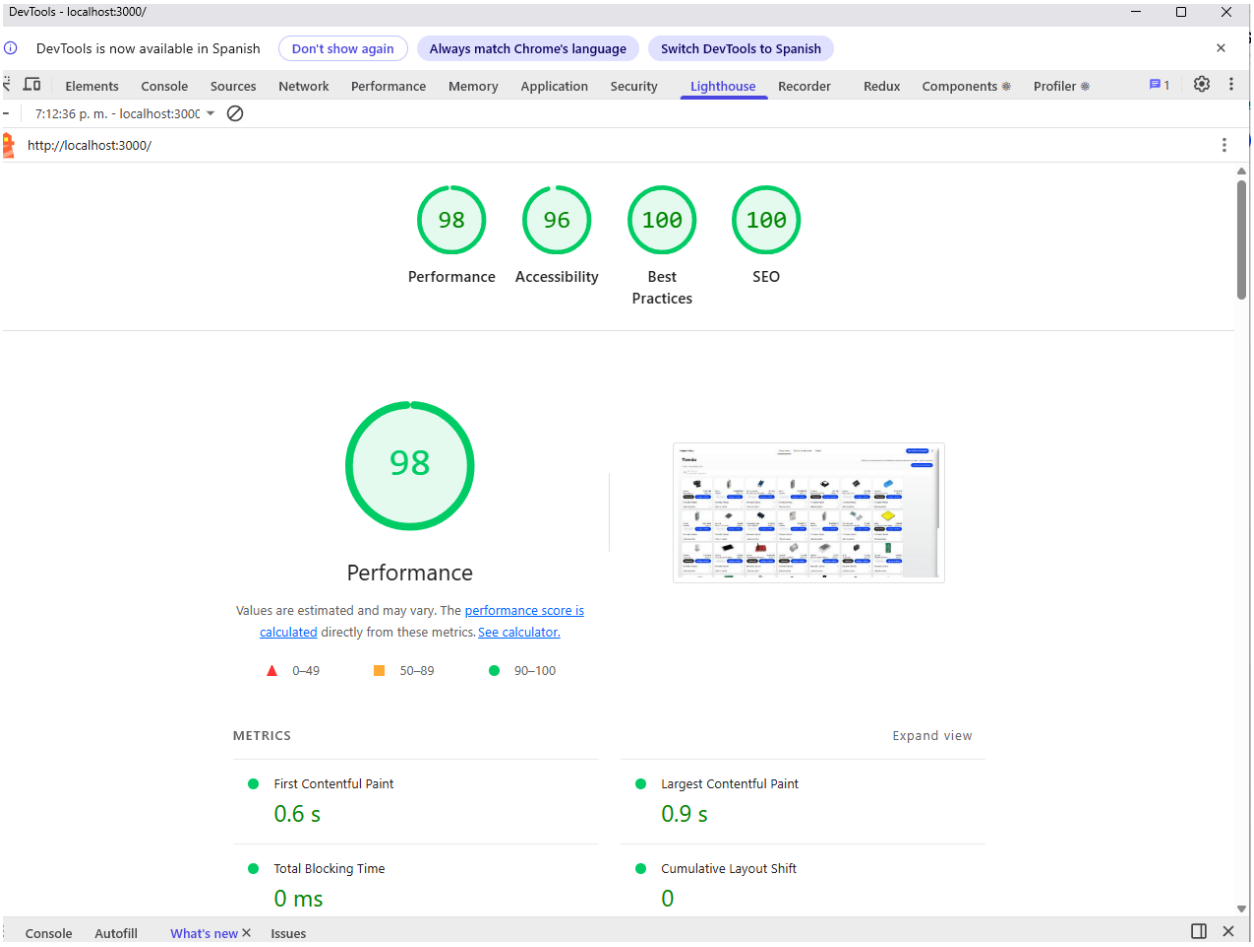


Figura 7.20: Segunda evaluación Lighthouse tras mejoras en SEO, accesibilidad y revisión de rendimiento

En la segunda auditoría (Figura 7.20), **Performance** alcanzó **98/100**, con First Contentful Paint 0.6 s, Largest Contentful Paint 0.9 s, Total Blocking Time 0 ms y Cumulative Layout Shift 0; **Best Practices** se mantuvo en **100/100**; **Accesibilidad** pasó a **96/100**; y **SEO** a **100/100**, en línea con las intervenciones descritas.

Análisis de resultados del cuestionario de validación

La encuesta se aplicó a una muestra de **9 participantes**. En la pregunta sobre dificultad para encontrar o comparar componentes en el mercado colombiano, el **77.8 %** reportó que esto ocurre **frecuentemente o siempre** (44.4 % siempre y 33.3 % frecuentemente), y el 22.2 % indicó que sucede **a veces**, sin respuestas en las categorías de baja dificultad (Figura 7.21). Esto confirma la vigencia del problema abordado por el prototipo.

Respecto a la utilidad de una plataforma que centralice inventarios y precios, el **100 %** calificó con **5/5** (Figura 7.22). De forma consistente, el **100 %** respondió "sí, definitivamente" a la opción de coordinar compras conjuntas para reducir costos de envío (Figura 7.24). En términos funcionales, estos dos resultados respaldan el valor de las capacidades de búsqueda unificada y coordinación colaborativa implementadas.

En relación con funcionalidades de expansión, el **88.9 %** manifestó interés en vender o intercambiar componentes no utilizados, frente a un 11.1 % que no lo consideró (Figura 7.23), lo que sugiere una línea de evolución viable hacia módulos de economía circular o mercado secundario.

Sobre la información prioritaria al buscar componentes, los criterios con mayor preferencia fueron **precio**, **proveedor** y **tiempo de entrega** (cada uno con 77.8 % de selección), seguidos de **disponibilidad** (66.7 %) y **ficha técnica** (55.6 %), mientras que **reseñas de otros usuarios** obtuvo 22.2 % (Figura 7.25). Este patrón es coherente con un contexto de compras técnicas en el que el costo, la fuente de abastecimiento y la viabilidad logística pesan más que variables sociales de recomendación.

En usabilidad y experiencia de uso, la navegación obtuvo calificaciones entre 4 y 5 (**77.8 % en 5/5** y **22.2 % en 4/5**; **promedio 4.78/5**; Figura 7.26), el diseño visual alcanzó **100 % en 5/5** (Figura 7.27), la accesibilidad visual registró la misma distribución 4–5 (**promedio 4.78/5**; Figura 7.28), y la intuición de acciones principales fue **100 % en 5/5** (Figura 7.29).

Finalmente, la satisfacción general mostró **77.8 % en 5/5** y **22.2 % en 4/5**, equivalente a un **promedio de 4.78/5** (Figura 7.30). En conjunto, los resultados evidencian una percepción favorable del prototipo tanto en utilidad práctica como en experiencia de usuario, con oportunidades de mejora centradas en ampliar funciones colaborativas de poscompra.

Gráficas de resultados del cuestionario

Figura 7.21: Dificultad para encontrar o comparar componentes electrónicos en el mercado colombiano.

En esta pregunta, el 77.8 % de la muestra indicó que la dificultad ocurre "frecuentemente" o "siempre" (44.4 % y 33.3 %, respectivamente), mientras que el 22.2 % respondió "a veces" y 0 % se ubicó en categorías de baja dificultad.

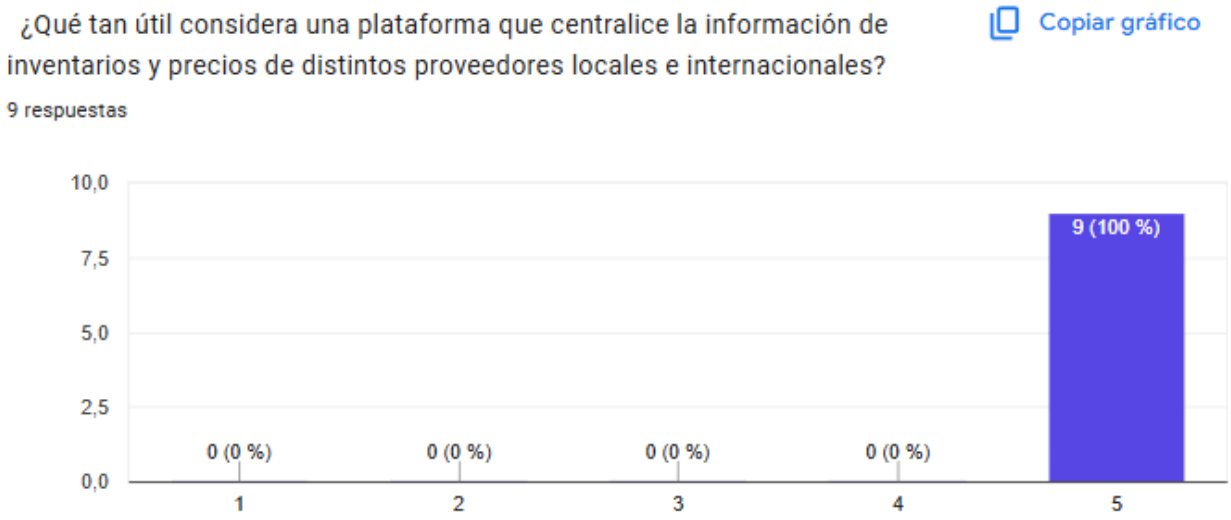


Figura 7.22: Utilidad percibida de una plataforma que centralice inventarios y precios de distintos proveedores.

El 100 % de los participantes calificó esta utilidad con 5/5.

Figura 7.23: Interés en vender o intercambiar componentes electrónicos no utilizados.

El 88.9 % manifestó interés en vender o intercambiar componentes no utilizados, mientras que el 11.1 % no mostró ese interés.

Figura 7.24: Valor percibido de coordinar compras conjuntas para reducir costos de envío.

El 100 % de la muestra seleccionó la opción "sí, definitivamente" para coordinar compras conjuntas y reducir costos de envío.

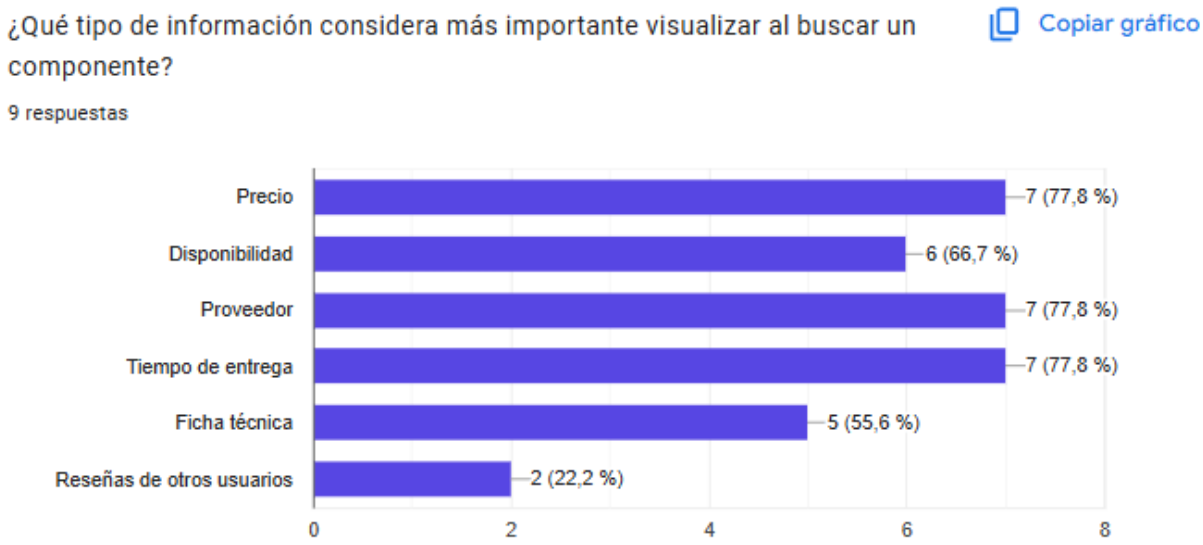


Figura 7.25: Información considerada más importante al buscar un componente.

Los criterios con mayor selección fueron precio, proveedor y tiempo de entrega (77.8 % cada uno); en segundo nivel estuvieron disponibilidad (66.7 %) y ficha técnica (55.6 %), y reseñas de usuarios registró 22.2 %.

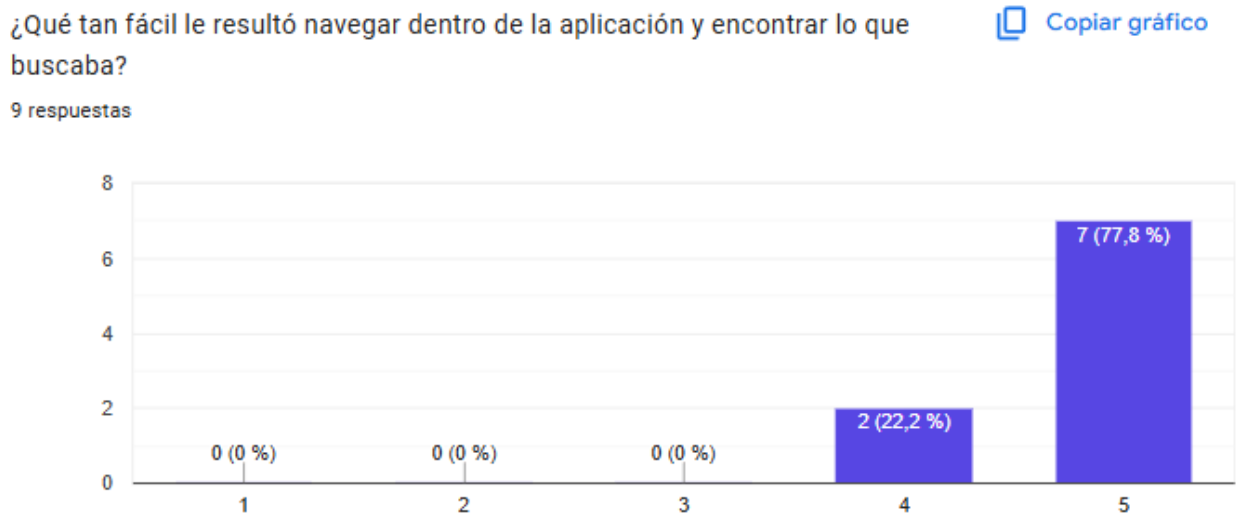


Figura 7.26: Facilidad de navegación dentro de la aplicación.

Las respuestas se ubicaron entre 4/5 (22.2 %) y 5/5 (77.8 %), con promedio de 4.78/5.

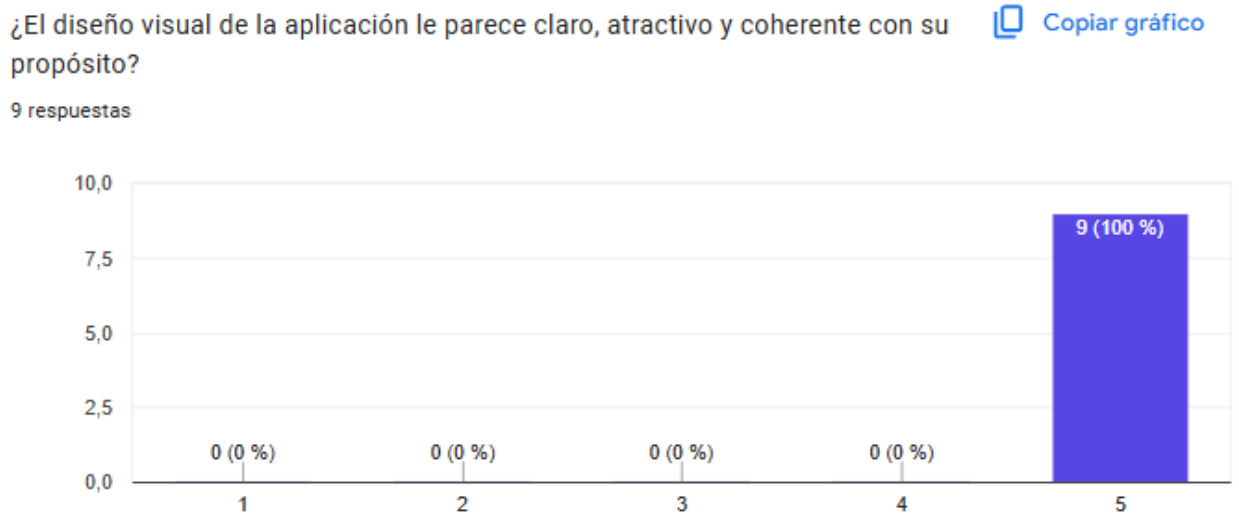


Figura 7.27: Claridad y coherencia del diseño visual de la aplicación.

El 100 % de los participantes calificó la claridad y coherencia visual con 5/5.

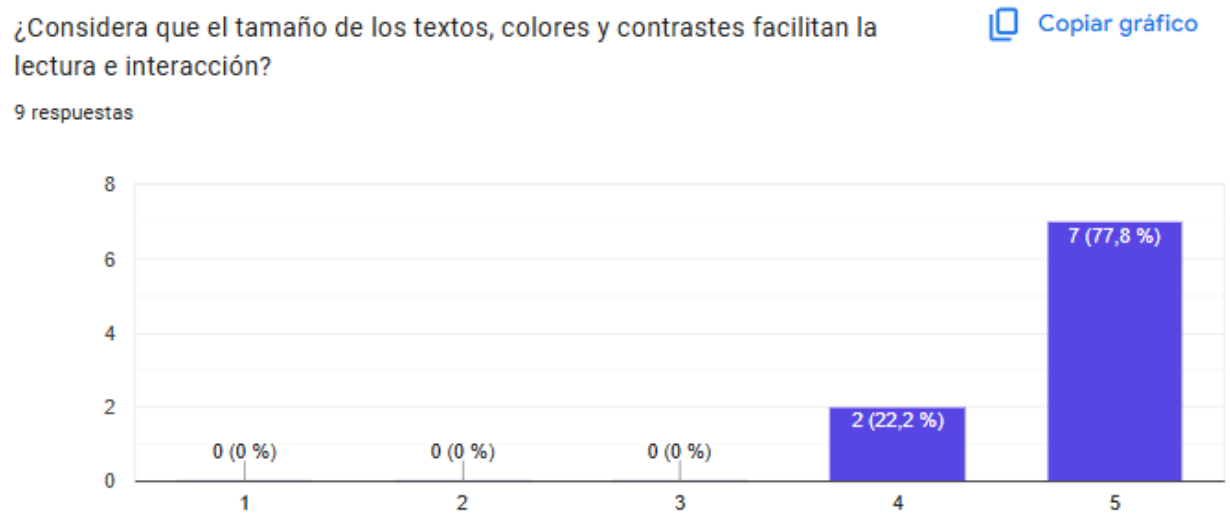


Figura 7.28: Accesibilidad visual: tamaño de textos, colores y contrastes.

La accesibilidad visual obtuvo calificaciones entre 4/5 (22.2 %) y 5/5 (77.8 %), con promedio de 4.78/5.

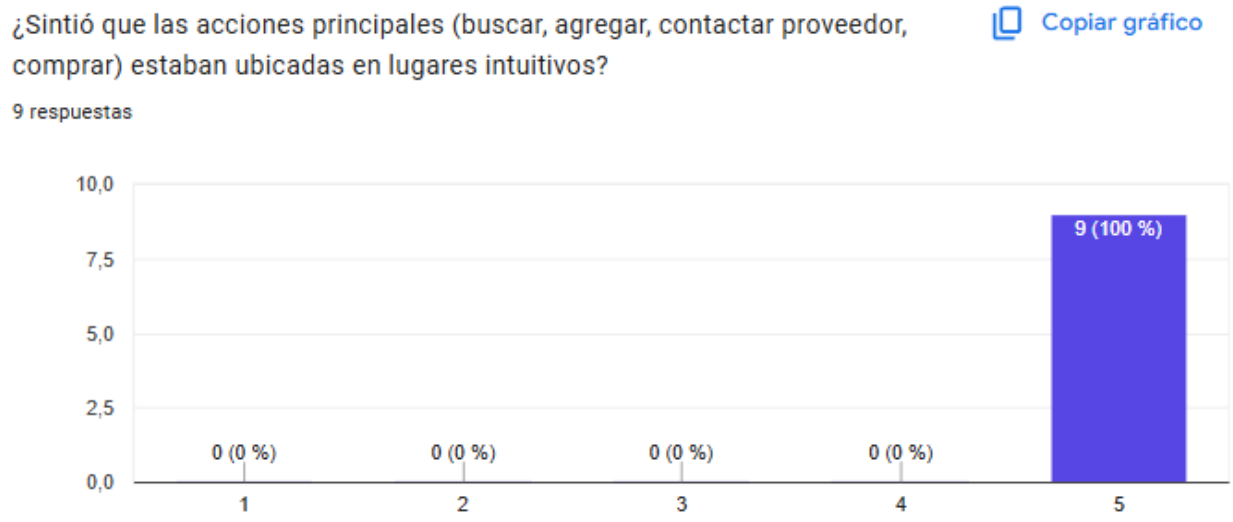


Figura 7.29: Intuición en la ubicación de las acciones principales (buscar, agregar, contactar, comprar).

El 100 % de la muestra calificó con 5/5 la intuición en la ubicación de acciones principales.

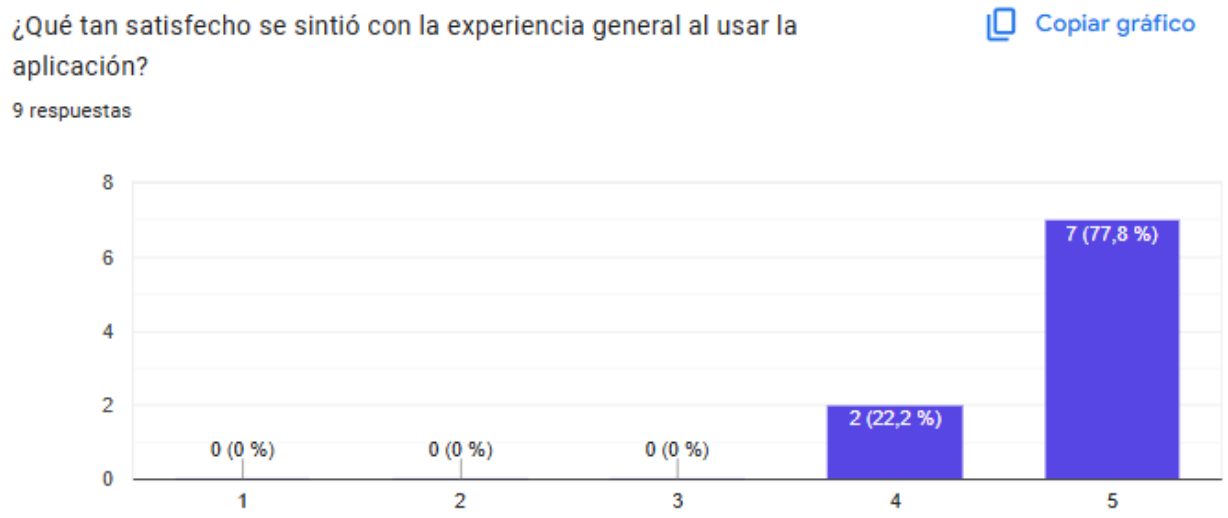


Figura 7.30: Satisfacción general con la experiencia de uso de la aplicación.

La distribución fue 77.8 % en 5/5 y 22.2 % en 4/5, con promedio general de 4.78/5.

Capítulo 8

Conclusiones

1. Mediante la investigación del mercado con actores globales y locales, la revisión del problema del abastecimiento de componentes y el levantamiento de requerimientos funcionales, se definieron como características clave: centralización de inventarios y precios, comparación de proveedores, acceso a ficha técnica, soporte para compras colaborativas y visibilidad de disponibilidad. Con ello se pudo establecer, qué funcionalidades debía incorporar la aplicación para apoyar la gestión logística.

2. Usando el análisis de alternativas tecnológicas y el diseño de una arquitectura modular con separación de responsabilidades (frontend (Nextjs), backend(Django Rest Framework), servicios de integración (Docker) y capa de datos (MySQL)), se construyó una propuesta técnica coherente con los requerimientos funcionales, técnicos y de escalabilidad. De esta forma se cumplió el objetivo de definir una arquitectura implementable y consistente con el alcance del proyecto.

3. Mediante la implementación del prototipo web, la integración de fuentes de datos por scraping y APIs, y la construcción del flujo colaborativo (búsqueda de componentes, agregar a tablero, creación de tableros y gestión de solicitudes), se materializó el diseño en una solución funcional. Con esto se resolvió el objetivo de garantizar coherencia entre lo diseñado y lo ejecutado en el sistema final.

4. Mediante auditorías con Lighthouse y sucesivas implementaciones acordes a las observaciones de la herramienta, se obtuvieron resultados medibles en dos corridas: en la primera, **Performance 99/100**, **Best Practices 100/100**, **Accesibilidad 84/100** y **SEO 82/100**; tras priorizar metadatos y descubrimiento (Open Graph, Twitter Card, robots), semántica y atributos ARIA en el frontend, la segunda auditoría registró **Performance 98/100**, **Best Practices 100/100**, **Accesibilidad 96/100** y **SEO 100/100**. En conjunto, el uso de Lighthouse como guía orientó cambios concretos y permitió verificar el incremento en SEO y accesibilidad y un rendimiento alto y estable. Complementariamente, en la validación con usuarios (9 participantes) se observó alta percepción de utilidad para compras conjuntas y satisfacción promedio de **4.78/5**. Con ello se cumplió el cuarto objetivo al contrastar desempeño técnico, usabilidad y aceptación del prototipo.

Capítulo 9

Recomendaciones

- Con una mayor asignación de presupuesto y recursos técnicos, la aplicación desarrollada podría evolucionar hacia una etapa productiva, considerando que surge de una necesidad real identificada en reuniones con empresarios y actores del sector de componentes electrónicos. A lo largo de la investigación quedó en evidencia un desequilibrio entre dos tipos de oferta: los catálogos globales, amplios pero poco adaptados a costos de envío, tiempos de llegada y escalas típicas del prototipado; y la oferta local, dispersa en múltiples canales con escasa centralización digital. Orientar parte de los recursos hacia la incorporación gradual de inventarios nacionales visibles en la misma interfaz contribuiría a corregir ese desbalance, de modo que la plataforma no reproduzca únicamente la lógica de grandes distribuidores internacionales sino que refuerce la capacidad de decisión frente al mercado colombiano, en línea con la centralización de información y los flujos colaborativos ya implementados en el prototipo.

- Resulta pertinente establecer alianzas formales con proveedores locales para actualización de inventarios, validación de la información y fortalecimiento del ecosistema nacional. En las entrevistas se registró interés en comercializar o truequear stock no utilizado; integrar ese canal con reglas claras (trazabilidad, condición del bien, garantías y disputas) acercaría la plataforma a prácticas reales junto con acuerdos que fijen frecuencias de actualización y calidad mínima de fichas ante un mercado local aún muy mediado manualmente.

- Explorar la expansión de la plataforma a otros sectores es viable donde concurran fragmentación de proveedores, compras frecuentemente descentralizadas y escasa interoperabilidad de catálogos, como puede ocurrir en insumos para manufactura ligera, equipamiento de laboratorio o automatización industrial. Para que ese salto sea sostenible conviene no limitarse a duplicar el catálogo electrónico: los grandes integradores o marketplaces globales pueden aportar referencia y amplitud, pero la capa que habilita condiciones reales de entrega, soporte y confianza sigue dependiendo de actores cercanos al comprador. Plantear desde el inicio reglas claras sobre los datos, acuerdos con quienes venden y cómo comprobar que la información sea confiable—respetando impuestos, trámites y envíos en el país donde se aplique—ayudaría a mantener lo esencial de este trabajo: reunir la oferta para que sea visible, permitir comparar y agrupar pedidos, sin que lo único nuevo sea cambiar de sector únicamente de nombre.

Capítulo 10

Anexos

10.1. Anexo 1 – Reunión con Nicolás Velásquez - CEO DeepSea

Durante la reunión con Nicolás Velásquez, fundador y CEO de DeepSea, nuestro equipo le presentó una idea de tesis sobre una aplicación para dar visibilidad a los productos de proveedores locales de componentes electrónicos. Nicolás nos planteó algunas dudas sobre el interés del mercado en los componentes electrónicos, pues el proceso de adquisición de estos funciona más mediante proveedores que ofrecen directamente los componentes a las empresas, en lugar de que las empresas busquen a los proveedores.

Nos mencionó que desde su perspectiva como empresario en Colombia, el desarrollo electrónico no es rentable debido a los altos costos. Explicó que aquí es más conveniente comprar y vender, en lugar de desarrollar. Sin embargo, Nicolás destacó que el hardware en el futuro podría experimentar un boom similar al del software. Además, compartió una idea interesante para un modelo de negocio que podría ser beneficioso para nuestra propuesta y acorde a las necesidades de su empresa.

Nicolás mencionó que DeepSea produce en otros países, pero realiza prototipos en Colombia. El costo de pedir componentes para prototipos en Colombia es muy alto, alrededor de 100 dólares por pieza, sin importar el tamaño. Esto eleva significativamente los costos de envío. Propuso la idea de una plataforma que reúna a personas interesadas en pedir los mismos componentes, para hacer pedidos conjuntos y así reducir costos. Además, nos contó que muchas veces las empresas se quedan con stock inutilizado, por lo que sería útil una forma de comerciar componentes viejos o usados, o incluso hacer trueques.

Finalmente, nos habló de una empresa que tiene 1000 tarjetas electrónicas sin utilizar y que está gastando dinero en almacenamiento. Esta situación refuerza la necesidad de nuestra aplicación y del modelo de negocio sugerido.

En resumen, Nicolás Velásquez destacó las dificultades del desarrollo electrónico en Colombia debido a los altos costos, sugiriendo en cambio un modelo de negocio que facilite pedidos conjuntos y el comercio de componentes usados. Este modelo podría ser viable y útil para nuestra aplicación propuesta, dado el contexto de empresas con stock inutilizado y la potencial expansión del mercado de hardware en el futuro.

10.2. Anexo 2 – Reunión con Sergio Echeverry - Comerciante

Sergio Echeverry es un comerciante caleño, graduado como ingeniero electrónico en la Pontificia Universidad Javeriana de Cali, el cual trabaja actualmente como importador de mercancías.

En primer lugar hablamos con Sergio acerca de la dificultad de conseguir dispositivos electrónicos para el desarrollo de prototipos, ya que a diferencia de otros productos como ropa, alimentos, suplementos deportivos y demás, estos son menos sencillos de adquirir.

Sergio dijo que en primer lugar los proveedores de componentes electrónicos a nivel nacional en su mayoría, no cuentan con tecnologías web que les permitan tener mejor visibilidad de sus productos, normalmente cuentan con tiendas físicas, lo cual hace el ejercicio de buscar componentes menos sencillo, ya que implica un mayor gasto de tiempo y recursos de transporte.

Preguntamos a Sergio: ¿según su experiencia por qué considera que ocurre esto con el mercado de componentes electrónicos a nivel nacional?

Sergio dijo que parte de los proveedores de dispositivos que cuentan con estas tecnologías suelen ser grandes empresas fuera del país como digikey o mouser, lo cual implica un problema en los costos ya que los componentes electrónicos en promedio suelen ser muy baratos, y tan solo el costo de envío puede ser incluso más de 10 veces el valor del mismo.

Al ser los componentes electrónicos tan económicos hace que no puedan ser aplicables técnicas de e-commerce como dropshipping, ya que el mismo envío puede superar el valor del producto. Además existen muchas variedades y referencias de chips en el mercado, y mantener actualizado el stock en una aplicación para empresas pequeñas y medianas de este sector es un problema.

Ante esta problemática explicamos a Sergio la aplicación que pensamos desarrollar, donde se hará una centralización de los diferentes marketplace de componentes que hay a nivel nacional y local, donde el usuario podrá encontrar desde una aplicación web todos los productos disponibles en diferentes tiendas como mouser, digikey, microelectronicos, didácticas electrónicas, etc. Esto permitirá al usuario rápidamente dar una barrida al mercado y encontrar las mejores opciones de compra. En caso de no haber disponibilidad del producto, los usuarios por medio de la aplicación, pueden ponerse de acuerdo para hacer pedidos al por mayor y reducir costos, o ayudar a los proveedores locales a saber que productos les conviene importar.

Sergio dio su opinión sobre la aplicación diciendo que es un modelo de negocio bastante bueno, aunque daría mejores ganancias si se aplicara a otros productos, sin embargo reconoció que sería un aporte importante para el mercado electrónico en Colombia.

Bibliografía

[1] Red Mexicana de Microfabricación y Semiconductores (MicroFabMX), "Panorama de los semiconductores para 2026: mercado, riesgos y cadenas de suministro," 2025, publicado dic. 2025. Consultado en abr. 2026.

[2] E. Lumba and A. Waworuntu, "Implementation of model view controller architecture in object oriented programming learning," *International Journal of New Media Technology*, vol. 8, no. 2, pp. 102–108, 2022.

[3] Docker Inc., "Docker documentation," 2025, accessed: 2025-02-10.

[4] ——, "Docker compose documentation," 2025, accessed: 2025-12-09.

[5] ——, "Docker networking overview," 2025, consultado en abr. 2026.

[6] F5 Inc., "Nginx reverse proxy," 2025, consultado en abr. 2026.

[7] C. A. P. Acosta, "Despliegue de aplicación on-premise en cloud computing utilizando servicios de aws." 2020.

[8] Cloud Native Computing Foundation, "Kubernetes documentation," 2025, consultado en abr. 2026.

[9] Selenium Project, "Selenium — browser automation documentation," 2025, accessed: 2025-12-09.

[10] Microsoft Corporation, "Playwright documentation," 2025, consultado en abr. 2026.

[11] Google Chrome Team, "Puppeteer documentation," 2025, consultado en abr. 2026.

[12] K. Reitz and R. Contributors, "Requests: Http for humans," 2025, python HTTP library; Accessed: 2025-12-09.

[13] L. Richardson, "Beautiful soup documentation," 2025, consultado en abr. 2026.

[14] V. Singrodia, A. Mitra, and S. Paul, "A review on web scrapping and its applications," in *2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, Jan. 2019, pp. 1–6.

[15] Django Software Foundation, "Django documentation," 2025, accessed: 2025-02-10.

[16] Encode OSS, "Django rest framework documentation," 2025, accessed: 2025-02-10.

[17] S. Ramírez, "Fastapi documentation," 2025, consultado en abr. 2026.

[18] VMware Inc., "Spring boot reference documentation," 2025, consultado en abr. 2026.

[19] ——, "Spring framework documentation," 2025, consultado en abr. 2026.

[20] Oracle Corporation, "Mysql reference manual," 2025, consultado en abr. 2026.

[21] PostgreSQL Global Development Group, "Postgresql documentation," 2025, consultado en abr. 2026.

[22] MongoDB Inc., "Mongodb manual," 2025, consultado en abr. 2026.

[23] J. E. S. Carnedas, "Analisis comparativo de dos bases de datos sql y dos bases de datos no sql," 2014.

[24] Aiven Ltd., "Aiven cloud database hosting documentation," 2025, accessed: 2025-02-10.

[25] ElasticSearch Team, "Elasticsearch documentation," 2025, accessed: 2025-02-10.

[26] Apache Software Foundation, "Apache solr reference guide," 2025, consultado en abr. 2026.

[27] Meilisearch SAS, "Meilisearch documentation," 2025, consultado en abr. 2026.

[28] Vercel Inc., "Next.js documentation," 2025, accessed: 2025-12-09.

[29] Meta Platforms Inc., "React official documentation," 2025, accessed: 2025-02-10.

[30] Vite Team, "Vite: Next generation frontend tooling," 2025, consultado en abr. 2026.

[31] Google LLC, "Angular documentation," 2025, consultado en abr. 2026.

[32] V. R. Shwini Yerlekar, "Developing an e-commerce web application with reactjs and firebase," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 2023.

[33] L. F. Martínez-Zapata, "El mercado de semiconductores, su demanda, su escasez y su logística. Taiwan Semiconductor Manufacturing Company (TSMC) desde el enfoque de la organización industrial (escrito en 2021; datos actualizados a 2023)," 2023.

[34] CEPAL, "Hacia la transformación del modelo de desarrollo en américa latina y el caribe: producción, inclusión y sostenibilidad," 2022.

[35] I. A. Rosnita Wirdiyanti, "How does e-commerce adoption impact micro, small, and medium enterprises' performance and financial inclusion? evidence from indonesia," *Springer Nature*, 2022.

[36] V. F. Zamora, "El comercio electrónico en colombia: Barreras y retos de la actualidad." 2017.

[37] Digi-Key Electronics, "Catálogo DigiKey," 2026.

[38] Mouser Electronics, "Sitio corporativo regional Latin America," 2026.

[39] Arrow Electronics, "Catálogo corporativo Arrow," 2026.

[40] Octopart Inc., "Octopart: buscador y comparador de componentes electrónicos," 2026.

[41] F. N. Iqbal, "A brief introduction to application programming interface (api)." 2023.

[42] T. E. Vamsi Madasu, Trinadh Venna, "Solid principles in software architecture and introduction to resm concept in oop." *Journal of Multidisciplinary Engineering Science and Technology (JMEST).*, 2015.

[43] R. S. Morales, "Arquitectura flux aplicada a sistemas de gestión de contenidos." 2022.

[44] J. A. M. C. Omar Fernando Chaparro Bayona, "Desarrollo de un aplicativo web de gestión de proyectos para el área de investigación sennova," *Vía Innova, 10 (1), 5-12*, 2022.

[45] Y. Liu and R. Patel, "Modern modular architecture patterns for scalable web systems," *Future Internet*, vol. 17, no. 11, p. 496, 2025.

[46] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More*. New York: Hyperion, 2006.

[47] C. Rodríguez, M. Baez, F. Daniel, F. Casati, J. C. Trabucco, L. Canali, and G. Percannella, "Rest apis: A large-scale analysis of compliance with principles and best practices," in *Proceedings of the 16th International Conference on Web Engineering (ICWE 2016)*, 2016, pp. 21–39.

[48] Python Software Foundation, "Python programming language," 2024.

[49] pandas Development Team, "pandas: Python data analysis library," 2025, accessed: 2025-02-10.

[50] Docker Inc., "Docker documentation," 2025, accessed: 2025-12-09.

[51] Tailwind Labs, "Tailwind css documentation," 2025, accessed: 2025-12-09.

[52] Google Chrome Developers, "Lighthouse: Automated website quality testing tools," 2025, accessed: 2025-12-09.

[53] Meta Platforms Inc., "React hooks — official documentation," 2025, accessed: 2025-12-09.

[54] M. Contributors, "Fetch api — web fundamentals documentation," 2025, accessed: 2025-12-09.

[55] Django Software Foundation, "Django authentication system — official documentation," 2025, accessed: 2025-12-09.