

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**  
**DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA**  
**CONCEPCIÓN - CHILE**



**DISEÑO E IMPLEMENTACIÓN DEL MÓDULO DE  
VENTAS Y SISTEMA DE VALORACIONES PARA LA  
PLATAFORMA MOTUSCAR**

**JOAQUÍN ANDRES RETAMAL RODRIGUEZ**

**MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO EN INFORMÁTICA**

**Profesor Guía: Cristian Antonio Lara Valenzuela**

**Julio - 2026**



## CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

### 1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

**Tipo de monografía (marcar una opción):** ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

**Título del trabajo:** Diseño e Implementación del Módulo de ventas y valoraciones para MotusCar

**Nombre del candidato(a):** Joaquin Andrés Retamal Rodriguez

**Carrera / Grado:** Ingeniería en Informática

**Campus:** Sede Concepción. **Departamento:** Departamento de Electrónica e Informática

### 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Cristian Lara Valenzuela, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

### 3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

**Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):**

---

---

---

### 4.- FIRMAS

**Profesor(a) guía o director(a) de memoria o tesis:**

CRISTIAN ANTONIO  
LARA VALENZUELA

Firmado digitalmente por  
CRISTIAN ANTONIO LARA  
VALENZUELA  
Fecha: 2026.07.27 23:16:41 -04'00'

**Fecha:** 27/07/2026 **Firma:**

**Estudiante o Candidato(a):**

**Fecha:** 27/07/2026 **Firma:**

*Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.*

## **DEDICATORIA**

Esto se lo dedico a mis padres, mis tios y mi hermana quienes fueron los que siempre me apoyaron y me amaron sin pedir nada a cambio, agradezco su apoyo, en especial en los malos momentos.

A mis compañeros y amigos dentro de la universidad, junto a ellos aprendí un montón y logramos mucho juntos.

Por último, a mis amigos fuera de lo que es el ambiente universitario, de manera indirecta con sus bromas y buenos momentos me han hecho olvidar varios malos ratos de los que he pasado en mi periodo como Universitario.

## **AGRADECIMIENTOS**

En esta sección quiero agradecer a todas las personas que de manera directa y/o indirecta hicieron posible este trabajo.

Para empezar, agradeceré a los múltiples profesores que he tenido durante mi periodo universitario, sus enseñanzas es lo que en primer lugar hicieron posible la creación del proyecto, en este caso y de manera especial los agradecimientos más importantes serán a los Profesores José Carrasco y Cristian Lara, por siempre estar dispuestos a orientarnos y echarnos una mano.

Por último, agradeceré a mis compañeros de equipo en el proyecto con quienes compartí bastante trabajo y responsabilidad.

## RESUMEN

**Resumen**—El presente trabajo describe el diseño e implementación del módulo de ventas y sistema de valoraciones para MotusCar, una plataforma web orientada a solucionar la asimetría de información y falta de confianza entre usuarios y talleres mecánicos en el sector automotriz. El desarrollo se llevó a cabo utilizando el lenguaje Python y el framework Django bajo la arquitectura Modelo-Vista-Template (MVT). El sistema implementado permite centralizar la oferta de repuestos, gestionar stock distribuido mediante algoritmos de agrupación por SKU, simular flujos de pago transaccionales integrados con Webpay y generar documentos tributarios electrónicos. Como resultado principal, se desarrolló un sistema de reputación de "compra verificada", el cual asegura que las valoraciones de los talleres provengan exclusivamente de transacciones reales. Las pruebas funcionales demostraron la viabilidad técnica del módulo, aportando trazabilidad y transparencia, sentando así las bases operativas fundamentales para la plataforma MotusCar.

**Palabras Clave**—Comercio Electrónico; Django; Webpay; Sistema de Reputación; Trazabilidad.

## ABSTRACT

**Abstract**—This work describes the design and implementation of the sales module and rating system for MotusCar, a web platform aimed at mitigating information asymmetry and the lack of trust between users and auto repair shops in the automotive sector. The development was carried out using the Python programming language and the Django framework under the Model-View-Template (MVT) architecture. The implemented system allows centralizing the supply of spare parts, managing distributed stock through SKU grouping algorithms, simulating transactional payment flows integrated with Webpay, and generating electronic tax documents. As a main result, a "verified purchase" reputation system was developed, ensuring that repair shop ratings come exclusively from real transactions. Functional tests demonstrated the technical viability of the module, providing traceability and transparency, thus laying the fundamental operational foundations for the MotusCar platform.

**Keywords**— E-commerce; Django; Webpay; Reputation System; Traceability.

## GLOSARIO

API (Application Programming Interface): Interfaz de Programación de Aplicaciones. Conjunto de definiciones y protocolos que permite que dos aplicaciones de software se comuniquen entre sí.

CSS (Cascading Style Sheets): Lenguaje de diseño gráfico para definir la presentación de un documento estructurado escrito en un lenguaje de marcado como HTML.

Django: Framework de desarrollo web de alto nivel escrito en Python, que fomenta un desarrollo rápido y un diseño limpio y pragmático.

DRF (Django REST Framework): Herramienta potente y flexible para construir APIs Web en Django.

DTE (Documento Tributario Electrónico): Documento digital legalmente válido en Chile que reemplaza a las facturas y boletas de papel, utilizado en el proyecto para la generación de la boleta simulada.

HTML (HyperText Markup Language): Lenguaje de marcado estándar para crear páginas web.

JSON (JavaScript Object Notation): Formato ligero de intercambio de datos, fácil de leer y escribir para humanos y fácil de analizar y generar para máquinas. Utilizado en el proyecto para la comunicación entre el carrito y el backend.

MVT (Model-View-Template): Patrón de arquitectura de software utilizado por Django, que separa la lógica de acceso a datos (Modelo), la lógica de negocio (Vista) y la interfaz de usuario (Template).

ORM (Object-Relational Mapping): Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional.

Python: Lenguaje de programación interpretado de alto nivel, utilizado como base para el desarrollo del backend del sistema.

SEO (Search Engine Optimization): Conjunto de acciones orientadas a mejorar el posicionamiento de un sitio web en la lista de resultados de Google u otros buscadores.

SKU (Stock Keeping Unit): Código único de referencia utilizado para identificar y rastrear un producto específico en el inventario.

UI (User Interface): Interfaz de Usuario. El medio por el cual el usuario y un sistema informático interactúan.

UX (User Experience): Experiencia de Usuario. Se refiere a cómo se siente una persona al interactuar con un sistema, incluyendo la facilidad de uso y la eficiencia.

Webpay: Servicio de pago online de Transbank que permite realizar transacciones seguras con tarjetas de crédito y débito en Chile.

## INDICE DE CONTENIDOS

|                                                        |    |
|--------------------------------------------------------|----|
| RESUMEN .....                                          | 4  |
| ABSTRACT .....                                         | 4  |
| INTRODUCCIÓN .....                                     | 12 |
| CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA .....              | 13 |
| 1.1 Contextualización del problema .....               | 13 |
| 1.2 Situación actual y problemática.....               | 13 |
| 1.3 Actores Involucrados.....                          | 14 |
| 1.4 Consecuencias de la no intervención .....          | 14 |
| 1.5 Propuesta General .....                            | 14 |
| 1.6 Subproyecto: Modulo de ventas y valoraciones ..... | 14 |
| 1.7 Objetivo general del subproyecto .....             | 15 |
| 1.8 Objetivos específicos del subproyecto .....        | 15 |
| CAPÍTULO 2: MARCO CONCEPTUAL .....                     | 16 |
| 2.1 Tecnologías de Desarrollo Backend .....            | 16 |
| 2.1.1 Lenguaje Python .....                            | 16 |
| 2.1.2 Framework Django .....                           | 16 |
| 2.1.3 Arquitectura MVT (Modelo-Vista-Template).....    | 17 |
| 2.2 Tecnologías de Interfaz (Frontend) .....           | 17 |
| 2.3 Integración de Pagos Electrónicos .....            | 18 |
| 2.3.1 Webpay Plus (Transbank).....                     | 18 |
| 2.4 Proceso de Compra y Valoración de Servicios.....   | 18 |
| 2.5 Control de Acceso y Autorización .....             | 19 |
| 2.6 Metodología de Desarrollo .....                    | 19 |
| CAPÍTULO 3: PROPUESTA DE SOLUCIÓN .....                | 20 |

## DISEÑO E IMPLEMENTACIÓN DEL MÓDULO DE VENTAS Y SISTEMA DE VALORACIONES PARA LA PLATAFORMA MOTUSCAR

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| 3.1 Visión general de la solución .....                               | 20 |
| 3.2 Arquitectura del módulo de ventas .....                           | 20 |
| 3.3 Diseño del flujo de compra .....                                  | 21 |
| 3.3.1 Entidad Order (Orden de Compra) .....                           | 21 |
| 3.3.2 Entidad OrderItem (Detalle de Orden).....                       | 21 |
| 3.3.3 Especificación de Casos de Uso .....                            | 22 |
| 3.4 Diseño del modelo de datos.....                                   | 28 |
| 3.4.1 Algoritmo de Agrupación por Familia de SKU .....                | 28 |
| 3.4.2 Motor de Búsqueda y Filtrado Multivariable.....                 | 29 |
| 3.4.3 Gestión de Stock Distribuido .....                              | 30 |
| 3.5 Implementación del Flujo de Pagos (Simulación Webpay .....        | 31 |
| 3.5.1 Arquitectura de la Simulación .....                             | 31 |
| 3.6 Gestión del carrito y checkout.....                               | 32 |
| 3.6.1 Generación y Visualización de la Boleta Electrónica Texto ..... | 33 |
| 3.7 Integración con el sistema de valoraciones .....                  | 34 |
| 3.8 Consideraciones finales del capítulo .....                        | 35 |
| CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN .....                           | 36 |
| 4.1 Estrategia de Validación .....                                    | 36 |
| 4.2 Validación del Filtrado y Descubrimiento .....                    | 36 |
| 4.3 Validación de la Lógica Multi-Proveedor (Stock Distribuido) ..... | 37 |
| 4.4 Validación del Ciclo Transaccional y Emisión de Boleta.....       | 37 |
| 4.5 Validación del Sistema de Valoraciones (Verified Purchase).....   | 38 |
| 4.6 Conclusión de la Validación.....                                  | 38 |
| CAPÍTULO 5: CONCLUSIONES .....                                        | 39 |
| 5.1 Conclusiones Generales.....                                       | 39 |
| 5.2 Cumplimiento de Objetivos .....                                   | 39 |
| 5.3 Limitaciones y Trabajo Futuro .....                               | 40 |

|                                  |    |
|----------------------------------|----|
| REFERENCIAS BIBLIOGRÁFICAS ..... | 41 |
|----------------------------------|----|

## INDICE DE FIGURAS

|                                                                                              |    |
|----------------------------------------------------------------------------------------------|----|
| Figura 1. Diseño del Flujo de Compra. Fuente: Elaboración propia. ....                       | 27 |
| Figura 2. Modelo de base de datos relacional del sistema .....                               | 28 |
| Figura 3. Catálogo de productos del módulo de ventas. Fuente: Elaboración propia. ....       | 29 |
| Figura 4. Vista de detalle de producto y stock distribuido. Fuente: Elaboración propia. .... | 30 |
| Figura 5. Simulación del flujo de pago mediante Webpay. Fuente: Elaboración propia. ....     | 31 |
| Figura 6. Gestión del Carrito y Checkout. Fuente: Elaboración propia.....                    | 32 |
| Figura 7. Boleta electrónica generada tras la compra. Fuente: Elaboración propia. ....       | 33 |
| Figura 8. Interfaz de valoración de productos. Fuente: Elaboración propia. ....              | 34 |

## INDICE DE TABLAS

Tabla 1. Especificación del Caso de Uso: Realizar Compra de Repuestos. Fuente: Elaboración propia..... 22

Tabla 2. Caso de Uso: Valorar Producto Adquirido. Fuente: Elaboración propia..... 25

## INTRODUCCIÓN

La industria de servicios mecánicos automotrices se caracteriza por una constante asimetría de información entre los prestadores de servicios y los clientes. Esta brecha genera un clima de desconfianza, dificultando la toma de decisiones informadas por parte de los usuarios y limitando el crecimiento digital de los talleres honestos y de calidad. En respuesta a esta problemática nace el proyecto MotusCar, una iniciativa tecnológica orientada a transparentar este ecosistema.

El presente documento detalla el desarrollo del componente central de esta plataforma: el Módulo de Ventas y Valoraciones. El objetivo de este subproyecto es diseñar un entorno transaccional seguro y unificado que permita a los usuarios localizar repuestos, simular el proceso de pago y, de manera crucial, generar valoraciones de servicio respaldadas por transacciones reales.

La memoria se estructura en cinco capítulos: el Capítulo 1 expone la problemática del sector y los objetivos del proyecto; el Capítulo 2 revisa el marco conceptual y las tecnologías backend y frontend utilizadas (Python, Django); el Capítulo 3 detalla la arquitectura de la solución propuesta, incluyendo el diseño de base de datos y la gestión de stock distribuido; el Capítulo 4 presenta las pruebas funcionales y la validación del sistema; y finalmente, el Capítulo 5 expone las conclusiones del desarrollo y las proyecciones de trabajo futuro.

## **CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA**

### ***1.1 Contextualización del problema***

En Chile y el mundo hispanohablante, la interacción entre dueños de vehículos y talleres mecánicos adolece de baja confianza y escasa transparencia. Estudios revelan que una gran mayoría de los conductores desconfía de los talleres de reparación: por ejemplo, el 85% de los entrevistados en un estudio chileno mostraba desconfianza sobre cobrar un precio justo (La Tercera, 2009). De modo similar, un informe español reciente indica que aproximadamente el 90% de los conductores expresa incertidumbre ante los servicios mecánicos (ABC Motor, 2024). Esta desconfianza surge por la falta de información sobre precios, plazos y la calidad real del servicio: los talleres no siempre comunican de forma clara sus condiciones, lo que deja al usuario a ciegas y a merced de la honestidad del mecánico. Además, los automóviles modernos alertan sobre fallas sin detalle, aumentando la ansiedad del dueño del vehículo (ABC Motor, 2024).

### ***1.2 Situación actual y problemática***

Para los usuarios, la falta de datos confiables implica riesgos reales: pueden ser cotizados de más, recibir un servicio deficiente o incluso inseguro, y sufrir demoras injustificadas. Muchos conductores ven la elección de taller como un “juego de azar” por desconocimiento, tal como señala BioBioChile; la información está tan dispersa que resulta arduo “encontrar al especialista adecuado” (BioBioChile, 2019). Para los talleres, las dificultades incluyen poca visibilidad ante nuevos clientes y dificultad para diferenciarse en base a calidad. Un taller con excelente trayectoria fuera de línea puede pasar inadvertido en internet, pues carece de herramientas SEO y de reputación digital. Además, las reseñas online suelen centrarse en lo anecdótico (mención de un mecánico específico, etc.), sin medir aspectos objetivamente críticos como cumplimiento de plazos o atención al cliente (BioBioChile, 2019). En resumen, ambos actores sufren: los usuarios siguen desconfiando (y podrían dejar de llevar su auto al taller o elegir pésimas opciones), mientras los talleres honestos pierden negocios ante la incertidumbre general.

### ***1.3 Actores Involucrados***

Los principales participantes son los usuarios/clientes (dueños de autos), los talleres mecánicos y las plataformas digitales. Los usuarios desean un servicio confiable y claro en precios y plazos. Los talleres buscan aumentar su clientela y demostrar su calidad, pero muchos carecen de presencia digital o mecanismos de reputación formales. Las plataformas online (Google, Facebook, portales especializados) actúan como intermediarias de información, pero actualmente no integran ni verifican los datos críticos (hora de entrega, calidad técnica, atención). Estas brechas entre actores dificultan la transparencia en el sector.

### ***1.4 Consecuencias de la no intervención***

Si no se aborda este problema, la brecha de confianza seguirá creciendo. Los usuarios podrían evitar talleres formales (recurriendo a soluciones caseras o inseguras) y gastar más en repuestos defectuosos. Para los talleres, la falta de credibilidad general estanca su crecimiento y perjudica al rubro automotriz en su conjunto. En el peor de los casos, un accidente o mal servicio grave podría manchar la reputación colectiva del sector. Desde la perspectiva de la economía digital, existe además un costo de oportunidad: el nivel de desconfianza actual genera ineficiencias (clientes potenciales mal dirigidos), algo que una plataforma adecuada podría mitigar.

### ***1.5 Propuesta General***

Con base en lo anterior, el objetivo general de *MotusCar* es diseñar y desarrollar una plataforma web que aumente la visibilidad y la confianza en talleres mecánicos de calidad.

### ***1.6 Subproyecto: Módulo de ventas y valoraciones***

Dentro de la propuesta general de *MotusCar*, este trabajo se focaliza en el desarrollo del módulo de ventas y valoraciones de servicio. En particular, se aborda el diseño e implementación de las funcionalidades que permiten a los usuarios:

- adquirir servicios o repuestos ofrecidos por talleres mecánicos a través de la plataforma, y
- registrar valoraciones asociadas a dichas compras, una vez que el servicio ha sido efectivamente prestado.

El propósito de este subproyecto es aportar a la transparencia y confianza en el ecosistema de talleres mecánicos, proporcionando mecanismos formales para registrar transacciones y evaluar la experiencia del usuario. De esta forma, el módulo se integra como un componente central de MotusCar, conectando la lógica de negocio relacionada con las compras con un sistema de reputación basado en valoraciones verificadas.

### ***1.7 Objetivo general del subproyecto***

Desarrollar el módulo de ventas y valoraciones de servicios del proyecto MotusCar, implementando un sistema que permita registrar transacciones y habilitar valoraciones asociadas únicamente a experiencias reales, con el fin de promover la transparencia, la mejora continua de los talleres mecánicos y una toma de decisiones más informada por parte de los usuarios.

### ***1.8 Objetivos específicos del subproyecto***

- 1- Analizar los procesos relacionados con la compra de servicios mecánicos y su posterior valoración, para identificar requerimientos del módulo de valoraciones.
- 2- Diseñar modelo de interfaces, procesos y datos de la funcionalidad definida en el ámbito de los procesos de compra y valoración de servicios de talleres mecánicos
- 3- Implementar el módulo de ventas y sistema de valoraciones para la plataforma MotusCar

## CAPÍTULO 2: MARCO CONCEPTUAL

### *2.1 Tecnologías de Desarrollo Backend*

Para la construcción de la lógica del servidor y la gestión de datos, se seleccionaron tecnologías robustas y ampliamente documentadas en la industria del desarrollo web.

#### *2.1.1 Lenguaje Python*

El proyecto está construido sobre Python, un lenguaje de programación de alto nivel interpretado, conocido por su legibilidad y eficiencia. Su amplia colección de bibliotecas estándar y de terceros facilita la implementación de lógica compleja con menos líneas de código en comparación con lenguajes como Java o C++. En el contexto de este proyecto, Python permite una manipulación ágil de los datos de ventas y una integración fluida con servicios externos.

#### *2.1.2 Framework Django*

Se utiliza **Django**, un framework web de "baterías incluidas" que fomenta un desarrollo rápido y un diseño limpio y pragmático. Django se encarga de gran parte de la complejidad del desarrollo web (seguridad, manejo de base de datos, autenticación), permitiendo centrarse en escribir la lógica de negocio de MotusCar en lugar de reinventar la rueda. Características clave utilizadas:

- **ORM (Object-Relational Mapping):** Permite interactuar con la base de datos utilizando objetos de Python en lugar de escribir consultas SQL puras.
- **Sistema de Autenticación:** Provee gestión segura de usuarios, grupos y permisos desde el primer momento.

### **2.1.3 Arquitectura MVT (*Modelo-Vista-Template*)**

Django implementa una variación del patrón MVC denominada MVT. Esta arquitectura separa claramente las responsabilidades dentro de la aplicación:

- **Modelo (Model):** Es la fuente definitiva de información sobre los datos (ej. Producto, Order). Contiene los campos y comportamientos esenciales de los datos almacenados.
- **Vista (View):** Describe *qué* datos deben mostrarse. Se encarga de la lógica de negocio, procesando las solicitudes HTTP y devolviendo las respuestas adecuadas (ej. filtrar productos por región).
- **Plantilla (Template):** Describe *cómo* se presentan los datos. Son archivos HTML que reciben información de las vistas y la renderizan para el usuario final.

### **2.2 Tecnologías de Interfaz (*Frontend*)**

La capa de presentación, con la que interactúan los usuarios finales, se construyó utilizando estándares web modernos para garantizar compatibilidad y rendimiento.

- **HTML5 Semántico:** Estructura el contenido de manera lógica, facilitando la accesibilidad y el posicionamiento en buscadores (SEO).
- **CSS3:** Se utiliza para el diseño visual y la adaptabilidad (Responsive Design), asegurando que la plataforma sea utilizable tanto en computadoras de escritorio como en dispositivos móviles.
- **JavaScript (Vanilla):** Se implementó lógica del lado del cliente para mejorar la interactividad sin depender de librerías pesadas. Funcionalidades clave como el carrito de compras en tiempo real y los menús desplegables se gestionan directamente con JavaScript nativo.

### ***2.3 Integración de Pagos Electrónicos***

Dada la naturaleza transaccional del módulo de ventas, es fundamental definir el estándar de pagos utilizado en Chile.

#### ***2.3.1 Webpay Plus (Transbank)***

Webpay es la pasarela de pagos más utilizada en el comercio electrónico chileno. Funciona bajo un modelo de redirección: el comercio (MotusCar) envía al usuario a un sitio seguro de Transbank para ingresar sus datos bancarios, y Transbank devuelve al usuario al comercio con un token de aprobación o rechazo. En este proyecto, se implementó un **Simulador (Mock) de Webpay** que replica fielmente este flujo de redirección y manejo de tokens, permitiendo validar el ciclo completo de la orden sin requerir credenciales bancarias de producción.

### ***2.4 Proceso de Compra y Valoración de Servicios***

El módulo de valoraciones se enmarca en un flujo secuencial estricto para garantizar la veracidad de la información.

1. **Búsqueda y Selección:** El usuario localiza el repuesto o servicio.
2. **Transacción (Checkout):** Se procesa el pago y se genera una Orden (Order).
3. **Habilitación (Verified Purchase):** El sistema detecta la existencia de una orden pagada.
4. **Valoración:** Solo entonces se desbloquea la interfaz para calificar.

Este control evita votos espurios y garantiza la credibilidad de las reseñas, ya que una valoración auténtica proviene necesariamente de una experiencia real respaldada por una transacción en base de datos.

## ***2.5 Control de Acceso y Autorización***

La integridad del sistema exige implementar mecanismos de seguridad en dos niveles:

1. **Autenticación:** Verificación de identidad mediante el sistema de usuarios de Django (django.contrib.auth).
2. **Autorización Lógica:** No basta con estar logueado; el sistema verifica permisos específicos de negocio. Por ejemplo, en la vista guardar\_valoracion, se ejecuta una consulta que cruza el ID del usuario con el historial de órdenes (Order) para validar que efectivamente adquirió el producto que intenta calificar. Si la validación falla, la acción es rechazada.

## ***2.6 Metodología de Desarrollo***

Para organizar el trabajo se utilizó un enfoque híbrido con prácticas ágiles. El desarrollo se estructuró en iteraciones cortas, permitiendo probar y ajustar funcionalidades clave de manera incremental.

1. **Fase 1:** Diseño de Modelos de Datos (Backend).
2. **Fase 2:** Implementación de Vistas y Lógica de Negocio.
3. **Fase 3:** Desarrollo de Interfaces (Templates) y experiencia de usuario.
4. **Fase 4:** Integración de componentes complejos (Simulador de Pagos y Valoraciones).

Esta metodología permitió detectar errores de lógica (como la duplicidad de reseñas) en etapas tempranas y corregirlos antes de la validación final.

## CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

### *3.1 Visión general de la solución*

La solución tecnológica desarrollada para el proyecto MotusCar consiste en una plataforma web integral basada en la arquitectura *Model-View-Template* (MVT) proporcionada por el framework Django. Dentro de este ecosistema, el presente trabajo de título se centra específicamente en el **Módulo de Ventas y Gestión Transaccional**.

Este módulo no solo funciona como un catálogo digital, sino que actúa como el motor de ingresos del modelo de negocio, integrando la lógica de comercio electrónico (selección de repuestos) con la lógica de servicios (talleres). El sistema permite resolver la problemática de la dispersión de información, centralizando la oferta de múltiples proveedores y talleres en una interfaz unificada que gestiona desde la búsqueda avanzada hasta la simulación de la pasarela de pago.

### *3.2 Arquitectura del módulo de ventas*

El desarrollo se adhiere a los principios de diseño de software modular. La estructura del módulo ventas se diseñó para operar de manera desacoplada del resto de aplicaciones del sistema (como inventario o usuarios), comunicándose únicamente a través de las interfaces de modelos de datos.

La arquitectura interna del módulo se divide en tres capas lógicas:

Capa de Datos (Models): Define la estructura de persistencia para las órdenes de compra y los ítems transaccionales, asegurando la integridad referencial con los usuarios y proveedores.

Capa de Lógica de Negocio (Views): Implementada en el archivo `views.py`, concentra las reglas de filtrado, agrupación de productos, paginación y la simulación del flujo de pagos.

Capa de Presentación (Templates): Utiliza el motor de plantillas de Django (DTL) para renderizar interfaces dinámicas (`carrito.html`, `checkout.html`, `detalle.html`) que responden al estado de la sesión del usuario.

### **3.3 Diseño del flujo de compra**

Para soportar el flujo transaccional, se diseñó un esquema de base de datos relacional optimizado para la trazabilidad de las ventas. A continuación, se detallan las entidades principales implementadas en `models.py`:

#### **3.3.1 Entidad Order (Orden de Compra)**

Esta entidad actúa como el encabezado de la transacción. Se diseñó con una flexibilidad clave: el campo `user` se definió como *nullable* (`null=True`, `blank=True`). Esto permite soportar dos modalidades de compra:

- **Usuarios Registrados:** La orden se vincula directamente a una cuenta de usuario existente (`settings.AUTH_USER_MODEL`).
- **Usuarios Invitados (Guest Checkout):** El sistema permite realizar compras sin registro previo, capturando datos esenciales a través de los campos `email_contacto` y `nombre_contacto` para la gestión post-venta.

Además, el modelo incluye campos de estado (`pagado`), información logística (`direccion_envio`, `region`, `comuna`) y auditoría temporal (`creado`).

#### **3.3.2 Entidad OrderItem (Detalle de Orden)**

Debido a que los precios de los repuestos pueden fluctuar en el tiempo, no es suficiente referenciar al producto. La entidad `OrderItem` implementa un patrón de "snapshot" (instantánea), almacenando una copia del `precio_unitario` y el `sku` en el momento exacto de la transacción. Esto garantiza que el historial de ventas permanezca inalterable ante futuros cambios de precios en el catálogo maestro.

### 3.3.3 Especificación de Casos de Uso

#### 3.3.3.1 Caso de Uso - Realizar Compra de Repuestos

**Tabla 1. Especificación del Caso de Uso: Realizar Compra de Repuestos. Fuente: Elaboración propia.**

|                               |                                                                                                                                                                                                        |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del Caso de Uso</b> | <b>Realizar Compra de Repuestos (Checkout)</b>                                                                                                                                                         |
| <b>Actor Principal</b>        | Cliente (Usuario Registrado)                                                                                                                                                                           |
| <b>Descripción</b>            | Permite al usuario adquirir los productos seleccionados en su carrito, ingresando datos de despacho y completando el pago simulado.                                                                    |
| <b>Precondiciones</b>         | <ol style="list-style-type: none"><li>1. El usuario debe haber iniciado sesión.</li><li>2. El carrito de compras no debe estar vacío.</li><li>3. Los productos deben tener stock disponible.</li></ol> |
| <b>Flujo Normal</b>           | <b>Acción del Actor</b><br><br><ol style="list-style-type: none"><li>1. Selecciona la opción "Ir a Pagar" desde el carrito.</li></ol>                                                                  |

|                               |                                                                                                                                                                                                                                     |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del Caso de Uso</b> | <b>Realizar Compra de Repuestos (Checkout)</b>                                                                                                                                                                                      |
|                               | <p>3. Completa el formulario con datos de despacho (Región, Comuna, Dirección).</p> <p>5. Selecciona "Pagar con Webpay".</p> <p>7. En el simulador, selecciona "Aprobar Pago".</p> <p>9. Visualiza la Boleta de Venta generada.</p> |
| <b>Respuesta del Sistema</b>  | <p>2. Valida stock y redirige a la vista de Checkout.</p> <p>4. Valida campos obligatorios y calcula costos de envío (si aplica).</p> <p>6. Redirige al ambiente seguro (Simulador Webpay).</p>                                     |

|                               |                                                                                                                                                                                                                                                                                                  |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del Caso de Uso</b> | <b>Realizar Compra de Repuestos (Checkout)</b>                                                                                                                                                                                                                                                   |
|                               | <p>8. Procesa la transacción, descuenta stock y genera el registro en base de datos (Order).</p> <p>10. Envía comprobante al correo del usuario.</p>                                                                                                                                             |
| <b>Flujos Alternativos</b>    | <p><b>A1. Stock Insuficiente:</b> Si al momento de pagar el stock cambió, el sistema notifica al usuario y bloquea el pago.</p> <p><b>A2. Pago Rechazado:</b> Si el usuario selecciona "Rechazar" en el simulador, el sistema redirige al carrito con mensaje de error y no genera la orden.</p> |

### 3.3.3.2 Caso de Uso – Valorar Producto

Tabla 2. Caso de Uso: Valorar Producto Adquirido. Fuente: Elaboración propia.

|                               |                                                                                                                                                                    |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del Caso de Uso</b> | <b>Valorar Producto Adquirido</b>                                                                                                                                  |
| <b>Actor Principal</b>        | <b>Cliente (Comprador Verificado)</b>                                                                                                                              |
| <b>Descripción</b>            | <b>Permite a un usuario calificar y comentar un producto que ha comprado y pagado exitosamente.</b>                                                                |
| <b>Precondiciones</b>         | <b>1. El usuario debe estar autenticado.</b><br><br><b>2. Debe existir una orden con estado pagado=True vinculada al usuario que contenga el SKU del producto.</b> |
| <b>Flujo Normal</b>           | <b>Acción del Actor</b><br><br><b>1. Accede a la ficha técnica de un producto que ya compró (o desde el link de la boleta).</b>                                    |

|                               |                                                                                                                                                                                                                                                                                                 |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del Caso de Uso</b> | <b>Valorar Producto Adquirido</b>                                                                                                                                                                                                                                                               |
|                               | <p><b>3. Selecciona una calificación (1 a 5 estrellas) y escribe un comentario.</b></p> <p><b>5. Confirma la acción pulsando "Publicar Opinión".</b></p>                                                                                                                                        |
| <b>Respuesta del Sistema</b>  | <p><b>2. Verifica en segundo plano si existe la orden de compra. Si es positivo, habilita el formulario de valoración.</b></p> <p><b>4. Recibe los datos.</b></p> <p><b>6. Guarda la valoración en la base de datos (Valoracion) y actualiza la vista para mostrar la reseña publicada.</b></p> |
| <b>Excepciones</b>            | <b>E1. Intento de Duplicado: Si el usuario ya valoró el producto, el sistema muestra su reseña anterior y ofrece la opción de "Editar" en lugar de crear una nueva.</b>                                                                                                                         |

### 3.3.3.3 Diseño del flujo de compra

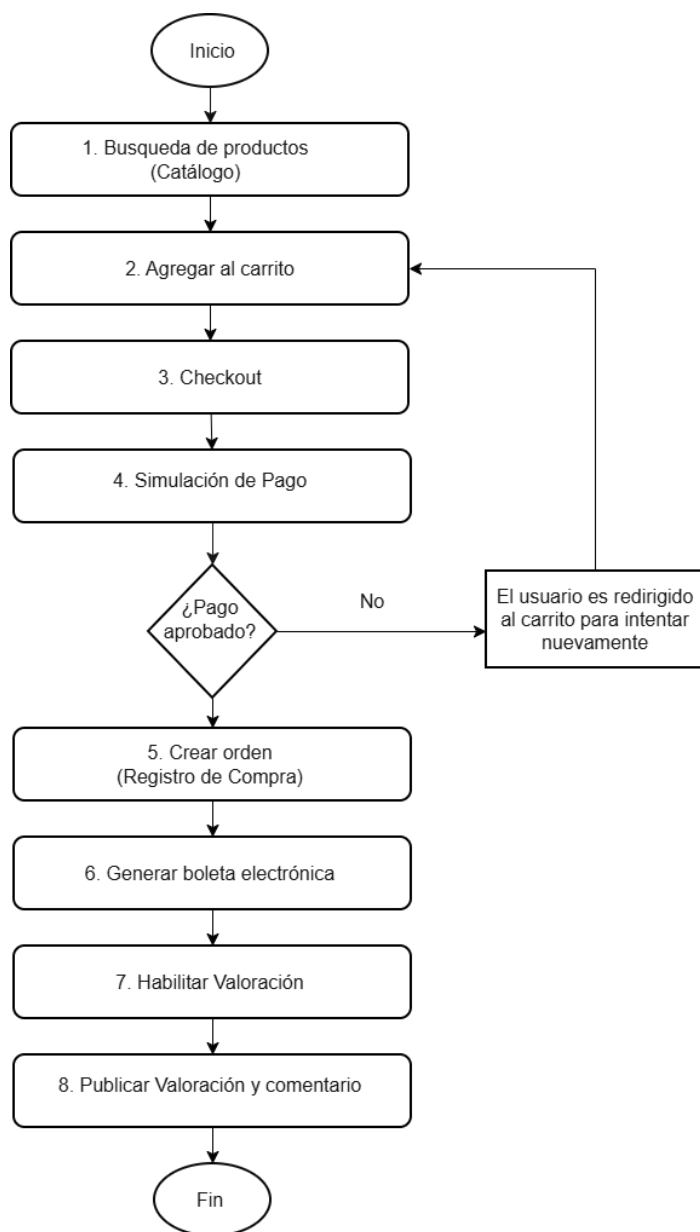


**Figura 1. Diseño del Flujo de Compra. Fuente: Elaboración propia.**

### 3.4 Diseño del modelo de datos

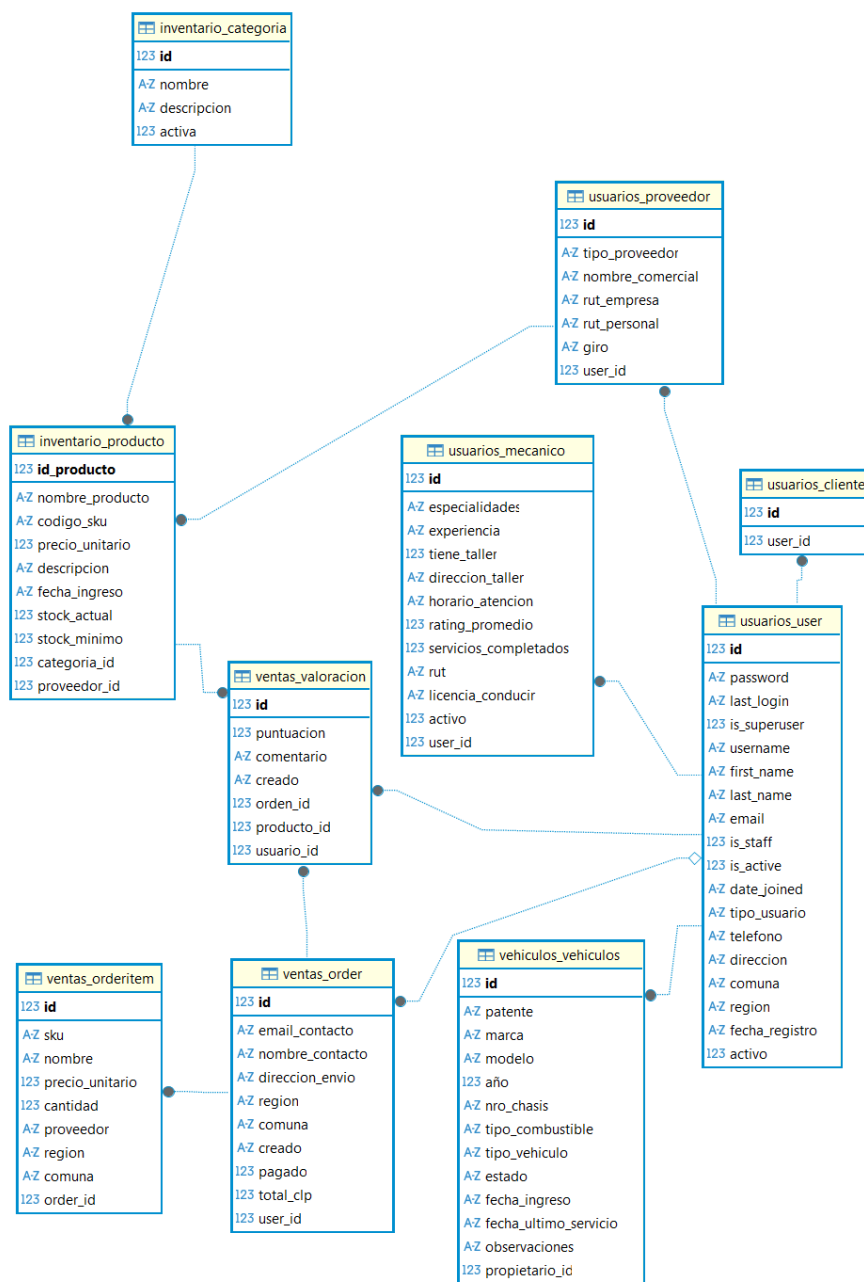


Figura 2. Modelo de base de datos relacional del sistema

### 3.4.1 Algoritmo de Agrupación por Familia de SKU

Uno de los desafíos técnicos detectados fue la redundancia visual en el catálogo cuando múltiples proveedores venden el mismo repuesto, o cuando existen variantes mínimas del mismo ítem.

Para resolver esto, se implementó en views.py una lógica de deduplicación basada en el SKU. La función auxiliar base\_sku(sku: str) analiza el código del producto y extrae su raíz identificadora. Posteriormente, en la vista principal ventas\_home, el sistema filtra y agrupa los resultados para mostrar una única "tarjeta de producto" por familia, mejorando significativamente la experiencia de usuario (UX) al evitar listados repetitivos.

### 3.4.2 Motor de Búsqueda y Filtrado Multivariable

Se desarrolló un motor de búsqueda robusto utilizando objetos Q de Django para consultas complejas. El sistema permite filtrar simultáneamente por:

- **Texto Libre:** Búsqueda insensible a mayúsculas (icontains) sobre el nombre del producto y el código SKU.
- **Ubicación Geográfica:** Filtros dinámicos por Región y Comuna, cruzando datos con la ubicación del proveedor (proveedor\_\_user\_\_region).
- **Rango de Precios y Categorías:** Validación de entradas numéricas para filtrar por precio\_min y precio\_max.

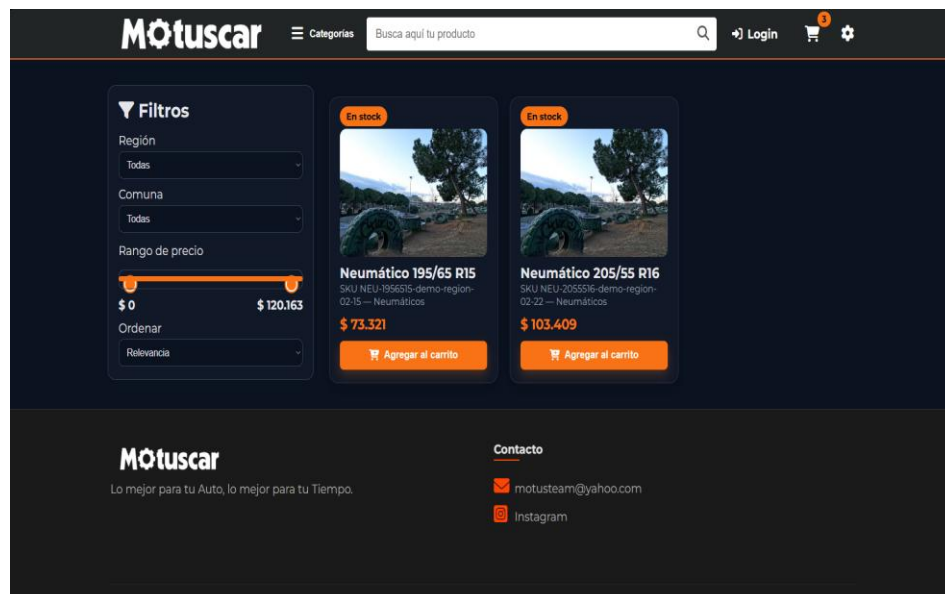


Figura 3. Catálogo de productos del módulo de ventas. Fuente: Elaboración propia.

3.4.3 Gestión de Stock Distribuido

En la vista de detalle (producto\_detalle), el sistema no solo muestra el producto, sino que agrega dinámicamente todas las variantes disponibles en el mercado. El algoritmo agrupa las variantes en un diccionario por\_region, permitiendo al usuario visualizar qué talleres específicos tienen el repuesto en stock y a qué precio, facilitando la decisión de compra basada en la geolocalización.

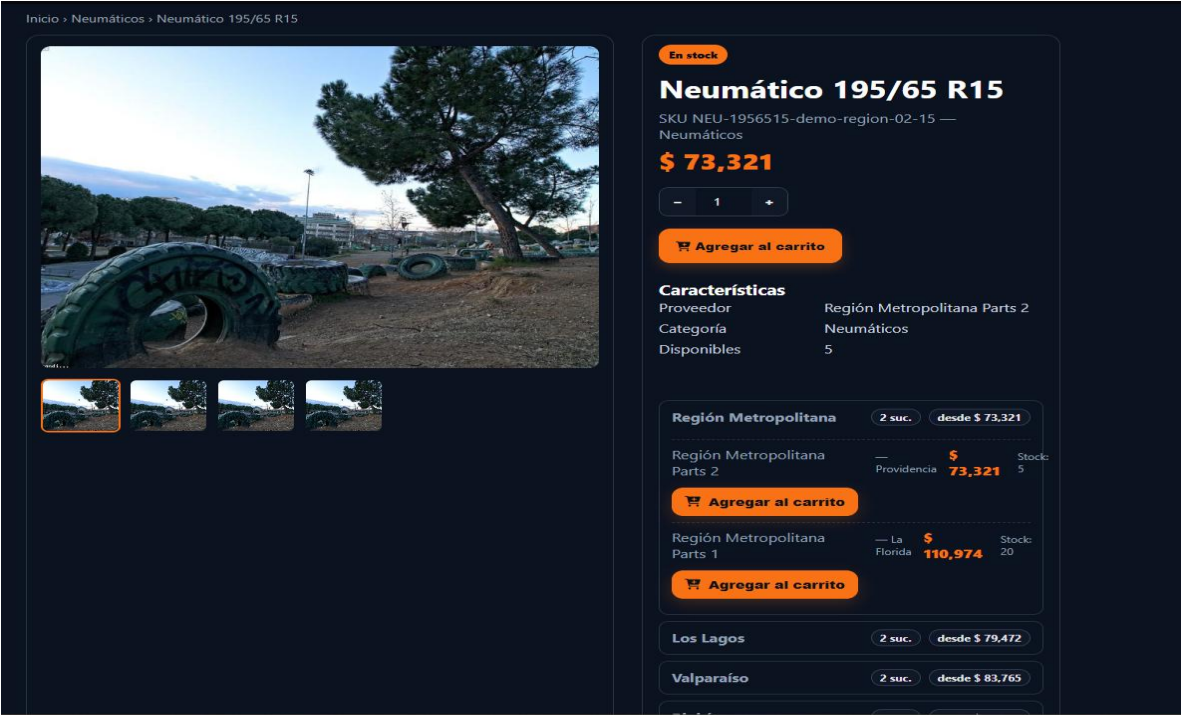


Figura 4. Vista de detalle de producto y stock distribuido. Fuente: Elaboración propia.

### ***3.5 Implementación del Flujo de Pagos (Simulación Webpay)***

Dado el contexto académico del proyecto, se implementó un "Mock" (simulador) de pasarela de pago que emula el comportamiento de Transbank Webpay, sin realizar transacciones bancarias reales, pero validando el ciclo de vida completo de la orden.

#### ***3.5.1 Arquitectura de la Simulación***

El flujo de pago se gestiona mediante el uso de variables de sesión (`request.session`) para mantener la seguridad y persistencia temporal de los datos durante la redirección entre el comercio y la pasarela simulada.

1. **Inicialización (`webpay_init`):** Recibe el payload del carrito en formato JSON y el total de la orden. Estos datos se inyectan en la sesión del usuario bajo la clave `mock_order`.
2. **Interfaz de Pasarela:** Se renderiza una vista intermedia que permite al usuario seleccionar el resultado de la operación ("Aprobar" o "Rechazar"), emulando la interfaz bancaria.
3. **Confirmación (`webpay_sim_ok`):** Si el pago es aprobado, el sistema recupera los datos de la sesión, procesa la orden y utiliza el módulo `EmailMultiAlternatives` de Django para generar y enviar un comprobante de compra HTML al correo del usuario.

#### **Simulador Webpay**

Total a pagar: \$45,078

Items: 3

Aprobar pago

Rechazar pago

**Figura 5. Simulación del flujo de pago mediante Webpay. Fuente: Elaboración propia.**

### 3.6 Gestión del carrito y checkout

La capa de presentación se construyó utilizando HTML5 semántico y CSS3, con un diseño responsivo adaptado a dispositivos móviles. Se implementó una gestión de estado del carrito de compras del lado del cliente (Client-side Cart) utilizando localStorage y JavaScript vanilla.

Esto permite una interacción fluida donde el usuario puede agregar productos, modificar cantidades y eliminar ítems sin necesidad de recargar la página (AJAX-free), sincronizando los datos con el servidor únicamente en el momento del checkout mediante un campo oculto (input type="hidden") que serializa el carrito a JSON.

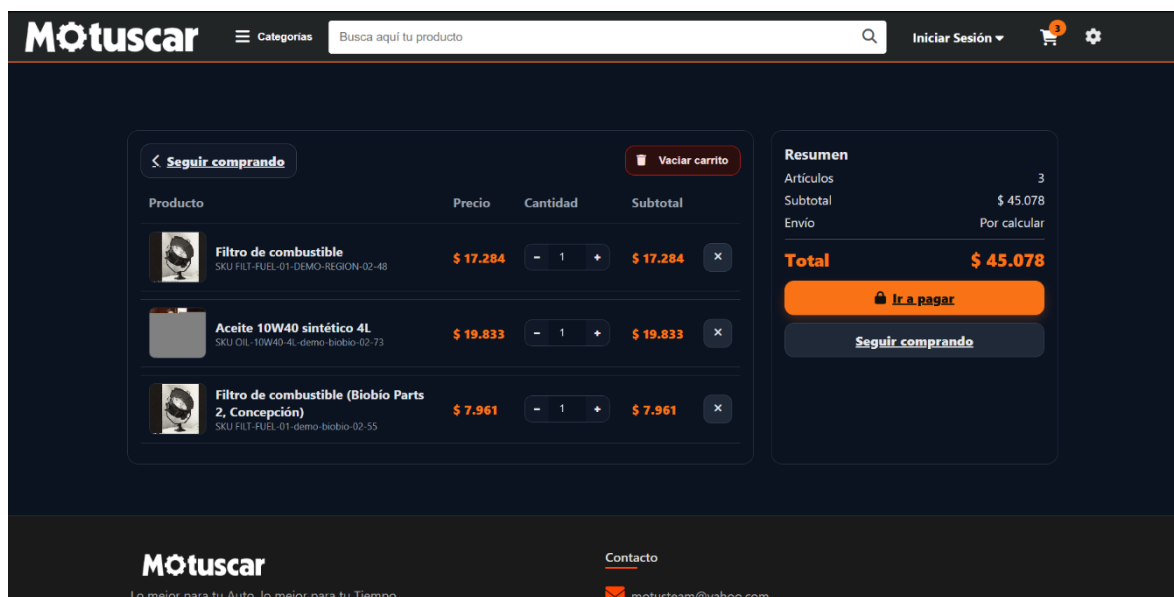


Figura 6. Gestión del Carrito y Checkout. Fuente: Elaboración propia.

### 3.6.1 Generación y Visualización de la Boleta Electrónica Texto

Tras la aprobación del pago en la simulación, el sistema no solo notifica al usuario, sino que genera en tiempo real un documento de respaldo transaccional: la **Boleta de Venta**.

Este componente fue diseñado siguiendo los estándares visuales de los Documentos Tributarios Electrónicos (DTE), e incluye:

- **Encabezado:** Con el logotipo corporativo y el número de folio (simulado).
- **Datos del Cliente:** Nombre y correo asociado a la compra.
- **Detalle de la Transacción:** Lista desglosada de productos adquiridos, mostrando nombre, variante (si aplica) y precio unitario.
- **Totalización:** Monto final pagado, coincidente con la suma de los subtotales.

La implementación técnica renderiza esta boleta como una vista HTML limpia, pensada para que el usuario pueda imprimirla o guardarla como PDF, cerrando así el ciclo de experiencia de usuario con una confirmación formal.

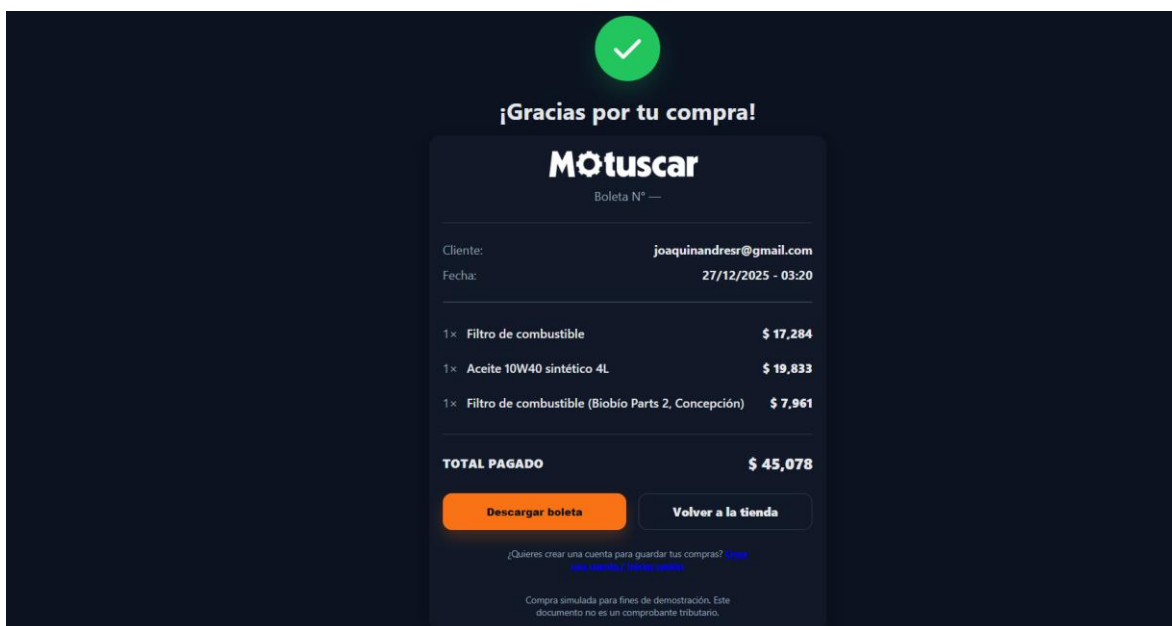


Figura 7. Boleta electrónica generada tras la compra. Fuente: Elaboración propia.

### 3.7 Integración con el sistema de valoraciones

Las órdenes de compra registradas por el módulo de ventas constituyen la base para habilitar el sistema de valoraciones del proyecto MotusCar. La existencia de una orden asociada a un usuario permite validar que una valoración corresponde a una experiencia real.

De esta forma, el módulo de ventas se integra conceptualmente con el sistema de valoraciones, asegurando que solo los usuarios que hayan realizado una compra puedan emitir reseñas sobre talleres o servicios.

La información generada por las órdenes de compra constituye el insumo principal para el módulo de valoraciones, permitiendo validar que una calificación solo pueda ser emitida por usuarios que hayan completado una transacción.

The figure consists of two screenshots of a product review interface. The top screenshot shows a form for writing a review. It has a dark blue background with a light blue border. At the top, there is a checkmark icon and the text "¡Compraste este producto!". Below this, it says "Cuéntanos qué tal te funcionó:". There is a star rating selector showing "★★★★★ (Excelente)". Below the stars is a text input field with the placeholder "Escribe tu opinión aquí...". At the bottom of the form is an orange button labeled "Publicar Opinión". The bottom screenshot shows the review submitted by a user. It has a dark blue background with a light blue border. At the top, there is a checkmark icon and the text "¡Gracias por tu opinión!". Below this, it says "Ya has valorado este producto con 5 estrellas." and "Muy bueno". There is a link "▶ Editar mi reseña". Below this is the user's name "joaquinandresr" and the text "Muy bueno". At the bottom, it says "30 Dec 2025 - Compra verificada ✓".

Figura 8. Interfaz de valoración de productos. Fuente: Elaboración propia.

### ***3.8 Consideraciones finales del capítulo***

El módulo de ventas desarrollado cumple con el objetivo de simular un flujo completo de comercio electrónico, validando la lógica de negocio asociada a la compra de productos y servicios automotrices. La solución implementada es coherente con el alcance del proyecto y sienta las bases para futuras extensiones, tales como la integración con pasarelas de pago reales, la persistencia completa del carrito en el backend y la incorporación de reportes de ventas.

## CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

### ***4.1 Estrategia de Validación***

Para garantizar la fiabilidad del módulo de ventas desarrollado, se ejecutó una estrategia de validación basada en pruebas funcionales de caja negra (*Black-box testing*). El objetivo principal fue verificar que los flujos críticos de negocio —desde el descubrimiento del producto hasta la post-venta— se comportaran según los requisitos definidos, asegurando la integridad de los datos transaccionales y la correcta experiencia de usuario.

### ***4.2 Validación del Filtrado y Descubrimiento***

Se sometió a prueba el motor de búsqueda y filtrado del catálogo para asegurar que los usuarios puedan localizar repuestos específicos de manera eficiente.

- **Prueba de Filtrado Geográfico:** Se verificó que, al seleccionar una región específica (ej. "Biobío"), el sistema ocultara automáticamente los productos que no tuvieran stock en dicha zona, actualizando la vista en tiempo real.
- **Prueba de Rango de Precios:** Se validó el funcionamiento del control deslizante (*slider*) de precios, confirmando que el sistema filtra correctamente los productos cuyo valor unitario excede los límites definidos por el usuario.

Como se evidencia en la **Figura** (ver Capítulo 3), la interfaz responde correctamente aplicando los filtros de manera acumulativa, permitiendo una segmentación precisa del catálogo.

#### **4.3 Validación de la Lógica Multi-Proveedor (Stock Distribuido)**

Uno de los componentes más complejos del sistema es la capacidad de gestionar el mismo SKU desde múltiples orígenes. Se realizó una prueba de validación sobre la vista de detalle del producto para confirmar la disponibilidad por zonas.

##### **Resultados de la prueba:**

1. El sistema identificó correctamente que el producto estaba disponible en múltiples regiones (Biobío, Valparaíso, Los Lagos).
2. Al desplegar la región "Biobío", se listaron correctamente dos proveedores distintos con precios y stocks independientes.
3. Se verificó que el botón "Agregar al carrito" captura el precio específico de la oferta seleccionada (sucursal) y no un precio genérico.

#### **4.4 Validación del Ciclo Transaccional y Emisión de Boleta**

Se ejecutó una prueba de extremo a extremo (*End-to-End*) para validar el flujo completo de compra.

1. **Checkout:** Se completó el formulario de despacho, validando que el sistema exige los campos obligatorios (RUT, Teléfono) antes de habilitar el pago.
2. **Simulación de Pago:** Se utilizó el simulador Webpay para aprobar la transacción exitosamente.
3. **Validación de la Boleta:** Tras la aprobación, se verificó la correcta generación del documento de término (Boleta de Venta).
  - **Integridad de Ítems:** Se comprobó que la boleta desplegara exactamente los productos seleccionados.
  - **Integridad Aritmética:** Se validó que la suma de los subtotales coincidiera con el total pagado.
  - **Enlace de Post-Venta:** Se verificó que los nombres de los productos en la boleta funcionaran como hipervínculos para redirigir al usuario al sistema de valoración.

#### ***4.5 Validación del Sistema de Valoraciones (Verified Purchase)***

Finalmente, se puso a prueba la lógica de restricciones para las reseñas de productos, asegurando que solo clientes reales puedan opinar.

##### **Caso A: Usuario sin comprar**

- **Acción:** Se accedió a la ficha de un producto no comprado con un usuario nuevo.
- **Resultado:** El formulario de valoración no se mostró, cumpliendo con la regla de negocio.

##### **Caso B: Usuario con compra confirmada**

- **Acción:** Tras completar el flujo de pago y volver al producto desde la boleta.
- **Resultado:** El sistema habilitó el bloque "¡Compraste este producto!" permitiendo ingresar una calificación y comentario.

##### **Caso C: Prevención de duplicados**

- **Acción:** El usuario intentó valorar nuevamente el mismo producto.
- **Resultado:** El sistema detectó la existencia de una reseña previa y, en lugar de duplicarla, mostró el mensaje de confirmación con los datos de la opinión existente, permitiendo solo su edición.

#### ***4.6 Conclusión de la Validación***

Las pruebas realizadas demuestran que el Módulo de Ventas cumple satisfactoriamente con los requisitos funcionales. La integración entre el catálogo, la lógica de selección de proveedores, la emisión de boleta y el sistema de reputación verificado opera de manera fluida, garantizando la trazabilidad completa del ciclo de vida de una venta.

## CAPÍTULO 5: CONCLUSIONES

### *5.1 Conclusiones Generales*

El desarrollo del módulo de ventas y valoraciones para MotusCar culminó con la implementación exitosa de un ecosistema transaccional funcional y seguro. La utilización de la arquitectura MVT (Modelo-Vista-Template) mediante el framework Django demostró ser una decisión técnica acertada, permitiendo un desarrollo ágil y un desacoplamiento efectivo entre la lógica de negocio, la persistencia de datos y la interfaz de usuario.

A través de esta solución, se logró abordar directamente la problemática de la asimetría de información en el sector automotriz. La plataforma no solo actúa como un catálogo centralizado con gestión de stock distribuido, sino que integra un flujo de validación crítico: el sistema de compras verificadas. Al vincular estrictamente la capacidad de emitir una valoración a la existencia de una orden de compra pagada, se elimina la posibilidad de reseñas espurias, aportando la transparencia y confianza que tanto los talleres como los usuarios requieren.

### *5.2 Cumplimiento de Objetivos*

Los objetivos específicos planteados al inicio del subproyecto fueron alcanzados a cabalidad en la implementación técnica:

**Diseño e Implementación de Base de Datos:** Se estructuró un modelo relacional robusto que garantiza la trazabilidad mediante la captura de precios históricos (Snapshot) en la entidad OrderItem y el registro de la transacción en la entidad Order.

**Desarrollo de API y Flujo de Pagos:** Se construyó un entorno transaccional que incluye un motor de búsqueda multivariable, la gestión asíncrona del carrito de compras y la integración de un simulador (Mock) de la pasarela Webpay Plus, emulando exitosamente el ciclo de aprobación y rechazo bancario.

**Sistema de Valoración Restringida:** Se logró asegurar que las calificaciones dependan de experiencias reales. La lógica de negocio implementada consulta el historial del usuario y

cruza los identificadores de producto (SKU base) para habilitar la interfaz de comentarios únicamente si existe una orden confirmada, previniendo además la duplicidad de reseñas mediante el uso de operaciones atómicas en la base de datos.

### **5.3 Limitaciones y Trabajo Futuro**

Si bien el módulo desarrollado es completamente funcional dentro del alcance definido, existen oportunidades de mejora para su eventual paso a un entorno de producción masivo:

- **Integración Productiva de Pasarelas de Pago:** La principal limitación actual es el uso de un simulador de pagos. El trabajo futuro inmediato contempla la integración con la API real de Transbank (Webpay Plus) utilizando credenciales de producción para procesar transacciones bancarias verídicas.
- **Escalabilidad del Sistema de Reseñas:** A medida que la plataforma crezca, será necesario implementar paginación dinámica (Lazy Loading) para las valoraciones en la vista de detalles del producto, además de un panel de moderación para administrar comentarios que infrinjan las normas comunitarias.
- **Despliegue y Contenerización:** Se proyecta la migración de la arquitectura actual hacia un entorno basado en contenedores (Docker), facilitando su despliegue continuo en servicios en la nube para asegurar la alta disponibilidad del catálogo y el módulo transaccional.

## REFERENCIAS BIBLIOGRÁFICAS

- ABC Motor. (2024). *El 90% de los conductores desconfía del servicio de un mecánico*. Recuperado de <https://www.abc.es/motor/reportajes/conductores-desconfia-servicio-mecanico-20240602090000-nt.html>
- BioBioChile. (2019). *DesRueda: la startup chilena que ayuda a los usuarios a buscar el taller mecánico ideal para su auto*. Recuperado de <https://www.biobiochile.cl/noticias/ciencia-y-tecnologia/internet/2019/07/18/desrueda-la-startup-chilena-que-ayuda-a-los-usuarios-a-buscar-el-taller-mecanico-ideal-para-su-auto.shtml>
- Django Software Foundation. (2024). *Django Documentation: Models and Databases*. Recuperado de <https://docs.djangoproject.com/en/stable/topics/db/models/>
- Eby, K. (2017). *¿Cuál es la diferencia? Agile vs Scrum vs el método de cascada vs Kanban*. Smartsheet. Recuperado de <https://es.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban>
- La Tercera. (2009). *Servicios: El síndrome del taller mecánico*. Recuperado de <https://www.latercera.com/revista-que-pasa/1-1487-9-servicios-el-sindrome-del-taller-mecanico/>
- Python Software Foundation. (2024). *The Python 3.12 Tutorial*. Recuperado de <https://docs.python.org/3/tutorial/index.html>
- Schwaber, K., & Sutherland, J. (2020). *La Guía de Scrum*. Scrum.org. Recuperado de <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf>
- Torresburriel Estudio. (2021). *Buenas prácticas en el diseño de valoraciones y sistemas de reputación*. Torresburriel WeBlog. Recuperado de <https://www.torresburriel.com/weblog/2021/04/23/buenas-practicas-en-el-diseno-de-valoraciones/>
- Transbank Developers. (2024). *Documentación Oficial Webpay Plus*. Recuperado de <https://www.transbankdevelopers.cl/documentacion/webpay-plus>
- Laudon, K. C., & Traver, C. G. (2023). *E-commerce 2023–2024: Business, Technology, Society* (18th ed.). Pearson.
- Django Software Foundation. (2024). *Writing your first Django app*. Recuperado de <https://docs.djangoproject.com/>