Universidad Zaragoza
1542

Trabajo Fin de Grado

Módulo de plataforma de pago multiplataforma para e-commerce basados en Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7

Multiplatform payment plugin for e-commerce based on Magento 1, Magento 2, Prestashop 1.6 and Prestashop 1.7

Autor

Jaime Puig Ortega

Director

David Abad Tomás

Ponente

Francisco Javier Zarazaga Soria

Escuela de Ingeniería y Arquitectura

2024

TÍTULO DEL PROYECTO

Modulo de plataforma de pago multiplataforma para e-commerce basados en Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7

Multiplatform payment plugin for e-commerce based on Magento 1, Magento 2, Prestashop 1.6 and Prestashop 1.7

RESUMEN

Hiberus Tecnología, S.A. es una compañía especializada en la consultoría de negocio y la presentación de servicios tecnológicos y outsourcing. Desde hace más de diez años, esta empresa ha buscado ofrecer el mejor servicio personalizado posible a sus clientes. En el presente proyecto, se aborda el desarrollo de un **plugin personalizado** para un cliente, con el objetivo de crear una **pasarela de pago multiplataforma** que funcione en cuatro entornos de e-commerce principales: **Magento 1**, **Magento 2**, **Prestashop 1.6** y **Prestashop 1.7**.

El proyecto tiene como objetivo principal ofrecer una solución sencilla, eficaz y escalable para la integración de pagos online en múltiples plataformas de comercio electrónico. A través del uso de un **plugin** común y de un código **JavaScript global**, se busca reducir la complejidad del desarrollo en cada una de las plataformas, minimizando el código específico y maximizando la reutilización del código central. El **widget de pago** desarrollado, junto con la integración de la API proporcionada por el cliente, constituye la base de esta pasarela, garantizando pagos seguros y eficientes.

Este trabajo responde a la necesidad de ofrecer una solución que facilite el proceso de pago en línea a los comerciantes, manteniendo la seguridad, flexibilidad y capacidad de expansión, adaptándose a las exigencias de los comercios online actuales.

DECLARACIÓN DE AUTORÍA

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Jaime Puig Ortega

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de

Grado en Ingeniería Informática (Título del Trabajo)

Módulo de plataforma de pago multiplataforma para e-commerce basados en Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7

Multiplatform payment plugin for e-commerce based on Magento 1, Magento 2, Prestashop 1.6 and Prestashop 1.7

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 29 de Noviembre de 2024

Fdo: Jaime Puig Ortega

ÍNDICE

1 INTRODUCCIÓN

1.1 Contexto del Trabajo

Hiberus Tecnología, S.A. [1] es una compañía especializada en la consultoría de negocio y la prestación de servicios tecnológicos y outsourcing en Banca y Seguros, Industria, Logística y Transporte, Medios, Retail y Manofacturing, Sanidad, SATS, Sector Público, y Turismo. Es la compañía de tecnología líder del Valle del Ebro, referente en el mercado Español y en pleno proceso de expansión en el mercado latinoamericano. Hiberus se integra en uno de los principales grupos empresariales del sector de las TIC (Tecnologías de la Información y Comunicación) en España formado por más de 1.000 profesionales y más de 100 M de euros de facturación. Para Hiberus es fundamental acompañarse de destacados profesionales de centros de Investigación y Universidades que sean capaces de aportarles valores destacados que posibiliten un mejor desarrollo de soluciones para sus clientes. En este sentido, la empresa mantiene una activa colaboración con estos centros en un intento de ser receptor de los avances de los mismos de forma directa (mediante la contratación de servicios de consultoría y/o transferencia) o de forma indirecta (acompañándose de ellos en propuestas que se presentan a sus clientes, o contratando a los profesionales formados en los mismos).

Una de las líneas de negocio de Hiberus se centra en el comercio electrónico, ofreciendo servicios de consultoría en diversas plataformas web, como Magento y Prestashop por ejemplo. Sin embargo, el desarrollo en estas plataformas presenta un desafío significativo debido a la coexistencia de múltiples versiones (Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7), cada una con arquitecturas y características diferentes. Esta fragmentación tecnológica complica la creación de soluciones unificadas, incrementando los costes de desarrollo y mantenimiento, y dificultando la implementación de nuevas funcionalidades.

El proyecto aborda esta problemática mediante el desarrollo de un plugin multiplataforma que minimice la cantidad de código específico para cada versión y maximice la reutilización de componentes comunes. Con este enfoque, se busca facilitar el mantenimiento del módulo y mejorar su escalabilidad, permitiendo una evolución más sencilla y eficiente del software en distintos entornos de comercio electrónico.

1.2 Tecnologías y Herramientas Utilizadas

En el desarrollo del módulo de pasarela de pago multiplataforma, se ha implementado un conjunto de tecnologías y herramientas que facilitan la creación, prueba y despliegue de soluciones eficientes y compatibles con las plataformas Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7. A continuación, se detalla el ecosistema tecnológico y las herramientas utilizadas:

1. Plataformas de E-commerce

El comercio electrónico ha crecido exponencialmente en los últimos años, impulsado por la digitalización y cambios en los hábitos de consumo. Este proyecto se centra en la integración de módulos de pago en dos plataformas líderes:

- **Magento:**
  - **Magento 1:** Versión original lanzada en 2008, que sigue siendo utilizada por numerosas tiendas pese a haber perdido soporte oficial en junio de 2020. Su estabilidad y ecosistema de módulos lo mantienen vigente.
  - **Magento 2:** Introducida en 2015, esta versión incluye mejoras en rendimiento, escalabilidad y experiencia de usuario, además de soporte nativo para móviles.
- **Prestashop:**
  - **Prestashop 1.6:** Versión lanzada en 2014, conocida por su interfaz sencilla y funcionalidad adecuada para pequeños negocios. Aunque está quedando obsoleta, sigue siendo ampliamente usada.
  - **Prestashop 1.7:** Versión más reciente (2016), con mejoras significativas en rendimiento, experiencia de usuario y un cambio en su sistema de plantillas (de Smarty a Twig).

2. Lenguajes y Tecnologías de Desarrollo

El desarrollo del módulo ha requerido el uso de varias tecnologías, cada una seleccionada por su compatibilidad y eficacia:

- **PHP:** Lenguaje backend utilizado en ambas plataformas. Proporciona flexibilidad y una integración eficiente con Magento y Prestashop.
- **JavaScript (jQuery):** Utilizado en los scripts compartidos para manejar la lógica del frontend. jQuery facilita las peticiones AJAX, la manipulación del DOM y la compatibilidad con navegadores antiguos.
- **HTML y CSS:** Para garantizar interfaces visuales consistentes y responsive.
- **Composer:** Herramienta de gestión de dependencias clave para Magento 2, que automatiza la instalación y actualización de paquetes.
- **MySQL:** Sistema de bases de datos relacional usado para almacenar productos, clientes y transacciones en ambas plataformas.

3. Entornos de Desarrollo y Despliegue

La consistencia y reproducibilidad del entorno de desarrollo son esenciales:

- **Docker:** Se empleó Docker para crear contenedores ligeros y replicables para cada plataforma, garantizando configuraciones de prueba idénticas a las de producción y eliminando problemas de compatibilidad.
- **Vagrant:** Utilizado en las fases iniciales del desarrollo para entornos virtualizados locales.
- **Entornos Locales:** Todas las pruebas se realizaron en contenedores Docker, simulando configuraciones reales de Magento y Prestashop.

4. Control de Versiones

- **Git:** Herramienta utilizada para el control de versiones, permitiendo un seguimiento eficiente de cambios en el código y facilitando la colaboración.
- **GitHub:** Repositorio remoto empleado para centralizar la gestión del proyecto.

5. Pruebas y Validación

- **Postman:** Utilizado para pruebas de integración con la API de la pasarela de pago. Esta herramienta permitió simular peticiones HTTP, depurar y garantizar la correcta comunicación entre el módulo y la pasarela.
- **Navegadores Web:** Se realizaron pruebas funcionales en Google Chrome y Mozilla Firefox para garantizar una experiencia de usuario consistente en diferentes entornos.

6. Entorno de Desarrollo Integrado (IDE)

- **PhpStorm:** IDE utilizado para el desarrollo, que ofrece funcionalidades avanzadas como autocompletado, análisis de errores, integración con Git y depuración eficiente.

7. Sistemas de Pago en Línea

La integración de pasarelas de pago es un elemento fundamental en cualquier plataforma de e-commerce, ya que facilita transacciones seguras entre clientes y comercios. Estas pasarelas, como PayPal o Stripe, son ejemplos de soluciones ampliamente utilizadas a nivel internacional.

En este proyecto, el módulo se conecta a una API de pasarela de pago mediante protocolos seguros como HTTPS, asegurando el manejo adecuado de datos sensibles, como información de tarjetas de crédito, a través de técnicas como la tokenización. Las APIs REST son la base para la comunicación entre el servidor de la tienda y el servicio de la pasarela de pago, garantizando un flujo confiable y eficiente en las transacciones.

8. Evolución y Tendencias del E-commerce

El comercio electrónico sigue evolucionando rápidamente:

- Tecnologías como Progressive Web Apps (PWA), Machine Learning para análisis predictivo y Headless Commerce están transformando las plataformas de e-commerce.
- Magento y Prestashop han adoptado estas tendencias, garantizando soporte para interfaces modernas y flexibles que mejoran la experiencia del usuario.

9. Consideraciones de Seguridad

La seguridad es una prioridad en el desarrollo del módulo:

- Uso de protocolos HTTPS para todas las comunicaciones.
- Cumplimiento con normativas PCI DSS y encriptación SSL para proteger datos financieros.
- Implementación de buenas prácticas, como autenticación de dos factores (2FA), para salvaguardar la información de los usuarios.

Justificación de las Herramientas Seleccionadas

La elección de tecnologías y herramientas se basó en:

- **Compatibilidad:** PHP, MySQL y jQuery son nativos en Magento y Prestashop, garantizando integraciones fluidas.
- **Eficiencia en Desarrollo:** Git, PhpStorm y Docker proporcionaron un flujo de trabajo ágil y reproducible.
- **Robustez en Pruebas:** Postman permitió validar la integración con la pasarela de pago de manera confiable.

En conjunto, estas herramientas y tecnologías han permitido desarrollar un módulo consistente, seguro y adaptable a diferentes entornos de producción.

1.3 Motivación y problema que se aborda

Motivación

El comercio electrónico es un sector en constante crecimiento, impulsado por la digitalización de los negocios y la creciente demanda de los consumidores por realizar compras en línea. Sin embargo, uno de los aspectos más críticos para el éxito de una tienda online es la experiencia de pago. La integración de una pasarela de pago rápida, segura y fácil de usar es clave para maximizar las conversiones y mejorar la satisfacción del cliente.

Existen varias **alternativas en el mercado** para integrar pasarelas de pago en plataformas de e-commerce como Magento y Prestashop. Algunas de las opciones más populares incluyen:

- **Módulos nativos de PayPal y Stripe**: Estas soluciones ofrecen módulos oficiales para múltiples plataformas, proporcionando una integración rápida y confiable. Sin embargo, suelen ser limitadas en términos de personalización y, en muchos casos, requieren configuraciones adicionales específicas para cada plataforma, lo que incrementa el tiempo de implementación.
- **Módulos desarrollados por terceros**: Empresas como **Mollie** y **Authorize.Net** ofrecen integraciones para diferentes plataformas de e-commerce. Si bien estos módulos pueden ser bastante completos, a menudo vienen con dependencias propias, diferentes requisitos para cada plataforma y, en ocasiones, tarifas adicionales por el uso de sus servicios.
- **Pasarelas de pago universales**: Herramientas como **Adyen** o **Braintree** ofrecen soluciones que intentan ser "plug and play" para varias plataformas, pero aún requieren desarrollos específicos para cada entorno, lo que implica que el mantenimiento y las actualizaciones deben realizarse individualmente para cada módulo de e-commerce.

A pesar de estas alternativas, se observan limitaciones comunes:

1. **Falta de consistencia**: La implementación varía entre plataformas, lo que genera experiencias de usuario dispares y obliga a realizar adaptaciones específicas para cada entorno.
2. **Complejidad en el mantenimiento**: Dado que las integraciones no comparten un núcleo común, cualquier cambio en la pasarela de pago requiere modificaciones en cada módulo específico, incrementando el esfuerzo de mantenimiento.
3. **Costos de desarrollo**: Crear y actualizar módulos diferentes para cada versión de las plataformas de e-commerce eleva los costos de desarrollo y aumenta el tiempo necesario para poner el módulo en funcionamiento.

Problema que se Aborda

El problema principal abordado en este proyecto es la **fragmentación en el desarrollo y mantenimiento** de pasarelas de pago para múltiples plataformas de e-commerce. Las diferencias significativas en la arquitectura de Magento 1, Magento 2, Prestashop 1.6 y

Prestashop 1.7 requieren el desarrollo de módulos separados, lo que incrementa la complejidad y el tiempo de implementación.

Además, la necesidad de mantener actualizados varios módulos, cada uno con su propia lógica de negocio y requisitos de integración, aumenta el riesgo de errores y de inconsistencias en la experiencia del usuario final. Este enfoque fragmentado no solo implica más trabajo para los desarrolladores, sino que también puede impactar negativamente en el rendimiento y la estabilidad de las tiendas online.

Solución Propuesta

Para resolver estas limitaciones, el proyecto propone desarrollar un **módulo de pasarela de pago multiplataforma** basado en un enfoque centralizado mediante el uso de **scripts de JavaScript reutilizables**. El objetivo es crear módulos ligeros para cada una de las plataformas (Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7), en los que la mayor parte de la lógica de negocio y funcionalidades se delegue a scripts de JavaScript compartidos. Este enfoque ofrece varias ventajas:

- **Consistencia en la experiencia de usuario**: Al centralizar la lógica en scripts reutilizables, se garantiza que el proceso de pago sea el mismo en todas las plataformas, ofreciendo una experiencia uniforme a los clientes.
- **Mantenimiento simplificado**: Los cambios en la lógica de la pasarela de pago pueden realizarse directamente en los scripts JavaScript globales, reduciendo la necesidad de modificar cada módulo individualmente para cada plataforma.
- **Reducción de costos y tiempo de desarrollo**: Al reutilizar código y minimizar las diferencias entre los módulos específicos de cada plataforma, se reduce significativamente el esfuerzo necesario para el desarrollo e implementación.
- **Escalabilidad**: El uso de tecnologías modernas de JavaScript permite manejar las interacciones con la pasarela de pago de forma asíncrona, utilizando técnicas como **AJAX** o **Fetch API**, mejorando el rendimiento y la capacidad de respuesta de la aplicación.

En comparación con las alternativas del mercado, esta solución ofrece una mayor eficiencia y escalabilidad, al reducir la redundancia en el desarrollo y facilitar el mantenimiento continuo. La propuesta se orienta a crear una **base de código común** que simplifique la implementación de nuevas funcionalidades y garantice una experiencia de pago optimizada en cualquier plataforma de e-commerce compatible.

1.4 Alcance, objetivos y limitaciones

Alcance

El alcance de este proyecto se centra en el desarrollo de un **módulo de pasarela de pago multiplataforma** para **Magento 1**, **Magento 2**, **Prestashop 1.6** y **Prestashop 1.7**. El objetivo es proporcionar una integración simple y eficiente que permita procesar pagos en línea a través de una interfaz unificada y fácil de implementar en estas plataformas.

El módulo estará compuesto por dos partes principales:

1. **Plugins ligeros específicos para cada plataforma**:
    - Desarrollo de un módulo específico para cada una de las versiones de Magento y Prestashop, asegurando la compatibilidad con sus arquitecturas y requerimientos.
    - Implementación de las interfaces necesarias para la configuración y gestión de la pasarela de pago desde el panel de administración de cada plataforma.
2. **Scripts de JavaScript reutilizables**:
    - Creación de un script de JavaScript que centralice la lógica de pago, manejando aspectos como la validación de datos, comunicación con la API de la pasarela de pago y respuesta asíncrona al usuario.
    - El mismo script será utilizado por los diferentes plugins, garantizando consistencia en la experiencia de usuario y simplificando el mantenimiento del código.

El alcance incluye:

- La integración con una **API REST** de una pasarela de pago específica.
- Pruebas de funcionalidad y rendimiento en los entornos de Magento y Prestashop mencionados.
- Documentación técnica para la instalación y configuración del módulo en cada plataforma.

No se incluye en el alcance del proyecto la personalización avanzada de la interfaz de pago para cada tienda, la integración con otras pasarelas de pago no especificadas inicialmente o el soporte para versiones futuras de Magento y Prestashop más allá de las especificadas (1.6 y 1.7 para Prestashop y 1 y 2 para Magento).

Objetivos

Objetivo General

Desarrollar un **módulo de pasarela de pago multiplataforma** que sea fácil de integrar en tiendas online basadas en **Magento 1**, **Magento 2**, **Prestashop 1.6** y **Prestashop 1.7**, proporcionando una experiencia de pago uniforme y eficiente para los usuarios.

Objetivos Específicos

1. **Simplificar la integración de la pasarela de pago**:
    - Crear plugins ligeros para cada plataforma, que faciliten la instalación y configuración del módulo por parte de los administradores de tiendas online.
2. **Centralizar la lógica de negocio en scripts de JavaScript reutilizables**:
    - Desarrollar un úninco script global que gestione la validación, comunicación con la pasarela de pago y respuesta al cliente de manera uniforme, independientemente de la plataforma utilizada.
3. **Garantizar la seguridad y cumplimiento de normativas**:
    - Implementar medidas de seguridad, como el uso de HTTPS, tokenización y cumplimiento de estándares **PCI DSS**, para asegurar el manejo seguro de datos sensibles durante el proceso de pago.
4. **Optimizar el rendimiento y la experiencia de usuario**:
    - Utilizar técnicas modernas de desarrollo web, principalmente **AJAX**, para ofrecer una experiencia de pago fluida y sin interrupciones.
5. **Documentar el proceso de desarrollo e implementación**:
    - Generar documentación clara y detallada sobre la instalación, configuración y uso del módulo para facilitar su adopción por parte de los administradores de tiendas.

Limitaciones

El proyecto presenta algunas limitaciones derivadas de su naturaleza y de las tecnologías involucradas:

1. **Compatibilidad limitada con versiones específicas**:
    - El módulo está diseñado exclusivamente para funcionar en **Magento 1**, **Magento 2**, **Prestashop 1.6** y **Prestashop 1.7**. No se garantiza su correcto funcionamiento en versiones anteriores o posteriores a las mencionadas, ni en plataformas de e-commerce distintas.
2. **Dependencia de la API de la pasarela de pago**:
    - La solución está construida en torno a una API REST de la pasarela de pago seleccionada. Cualquier cambio o actualización significativa en dicha API podría requerir modificaciones en los scripts de JavaScript y en los plugins específicos de cada plataforma.
3. **Funcionalidades avanzadas no incluidas**:

- El proyecto se centra en proporcionar una integración básica y funcional de la pasarela de pago. Funcionalidades avanzadas, como pagos en varias divisas, suscripciones o pagos recurrentes, no están contempladas en esta versión inicial del módulo.

4. **Limitaciones en la personalización de la interfaz de pago**:
    - Dado que el enfoque del proyecto es mantener la solución lo más simple y genérica posible, no se incluirán opciones extensivas de personalización visual de la interfaz de pago dentro del módulo. Los administradores de tiendas deberán realizar personalizaciones adicionales manualmente si así lo desean.
5. **Pruebas limitadas en entorno real**:
    - El proyecto realizará pruebas de integración y rendimiento en entornos de prueba para cada plataforma, pero el comportamiento en entornos de producción puede variar según la configuración específica de cada tienda online (plugins adicionales, temas personalizados, etc.).

En resumen, el proyecto tiene como objetivo proporcionar una solución sencilla y efectiva para la integración de una pasarela de pago multiplataforma, simplificando el desarrollo y mantenimiento a través del uso de scripts de JavaScript reutilizables, aunque presenta ciertas limitaciones derivadas de la compatibilidad y el alcance definido.

1.5 Esquema general de la memoria del proyecto

En esta sección se detalla la estructura de la memoria del proyecto, organizada en cinco capítulos principales, donde se aborda desde la contextualización y desarrollo hasta las conclusiones y bibliografía. El esquema es el siguiente:

Capítulo 2: Trabajo Desarrollado

En este capítulo se detalla el desarrollo del proyecto desde los requisitos del sistema, especificando tanto los funcionales como los no funcionales, hasta los aspectos técnicos más complejos. Se describe la arquitectura del módulo, destacando la integración de plugins y scripts JavaScript reutilizables, junto con el diseño del sistema a través de diagramas y la implementación de la lógica. Además, se analiza la magnitud del trabajo realizado, considerando tiempo, recursos y funcionalidades, y se exponen los principales desafíos técnicos enfrentados y sus soluciones.

Capítulo 3: Lecciones Aprendidas y Conclusiones

En este capítulo se recogen los aprendizajes del proyecto, identificando los conocimientos técnicos y de gestión adquiridos. Además, se proponen mejoras y futuras ampliaciones del módulo, como soporte para más plataformas y nuevas funcionalidades. Finalmente, se presentan las conclusiones, evaluando el cumplimiento de los objetivos y el impacto del trabajo realizado.

Capítulo 4: Bibliografía

En esta sección se listan todas las fuentes de información utilizadas para la realización del proyecto, incluyendo artículos, documentación técnica, libros y recursos online que han sido referenciados.

Capítulo 5: Anexos

El último capítulo incluye materiales adicionales que complementan la memoria del proyecto, tales como:

- **Manual de Instalación del Módulo**: Instrucciones detalladas para la correcta instalación y configuración del módulo de pasarela de pago en Magento y Prestashop.
- **Código Fuente Relevante**: Fragmentos de código y ejemplos que ilustran la implementación de las funcionalidades clave.

2 TRABAJO DESARROLLADO

2.1 Requisitos del sistema

Requisitos Funcionales	
RF1	El módulo debe ser compatible con las cuatro plataformas de e-commerce seleccionadas: Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7.
RF2	El sistema debe permitir a los usuarios realizar pagos online de manera rápida y efectiva, integrando una pasarela de pago que facilite transacciones seguras.
RF3	El módulo debe ofrecer la opción de fraccionar los pagos, permitiendo a los usuarios elegir realizar pagos en cuotas mensuales.
RF4	Se debe proporcionar un **widget** que pueda ser insertado fácilmente en los sitios web de las tiendas para implementar la funcionalidad de la pasarela de pago.
RF5	El módulo debe estar preparado para adaptarse al uso en múltiples idiomas, aunque en la versión inicial solo se incluya el idioma base. Esto facilita su escalabilidad internacional.
RF6	El módulo debe minimizar la cantidad de código nativo específico para cada plataforma (Magento y Prestashop) y maximizar el uso del **script JavaScript** compartido, lo que simplifica el mantenimiento y la actualización del sistema.

Tabla 1. Requisitos funcionales

Requisitos No Funcionales	
RNF1	El sistema debe garantizar transacciones seguras, cumpliendo con los estándares de seguridad para el manejo de datos sensibles, como el cumplimiento de normativas PCI DSS para la protección de la información de tarjetas de crédito.
RNF2	El módulo debe ser compatible con los navegadores modernos más utilizados, incluyendo **Google Chrome**, **Mozilla Firefox**, **Microsoft Edge** y **Safari**, garantizando una experiencia de usuario consistente.
RNF3	La solución debe ser escalable, es decir, estar diseñada para poder **incrementar la funcionalidad** en el futuro sin requerir una reestructuración completa del módulo. Esto permite añadir nuevas características o integraciones con otras plataformas de pago.
RNF4	El código debe estar optimizado para no afectar significativamente el rendimiento del sitio web en el que se integra, evitando tiempos de carga prolongados o problemas de usabilidad.
RNF5	El módulo debe ser fácil de instalar y mantener, facilitando actualizaciones futuras tanto para el código del módulo específico de cada plataforma como para los scripts JavaScript compartidos.
RNF6	El entorno de desarrollo debe estar preparado para funcionar en configuraciones locales utilizando **Docker**, permitiendo replicar un entorno similar al de producción para pruebas y ajustes.

Tabla 2. Requisitos no funcionales

2.2 Arquitectura y diseño software del sistema

La arquitectura conceptual de este proyecto se centra en definir los componentes clave del módulo de pasarela de pago multiplataforma y cómo interactúan para proporcionar una experiencia de pago unificada en las distintas plataformas de e-commerce: Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7.

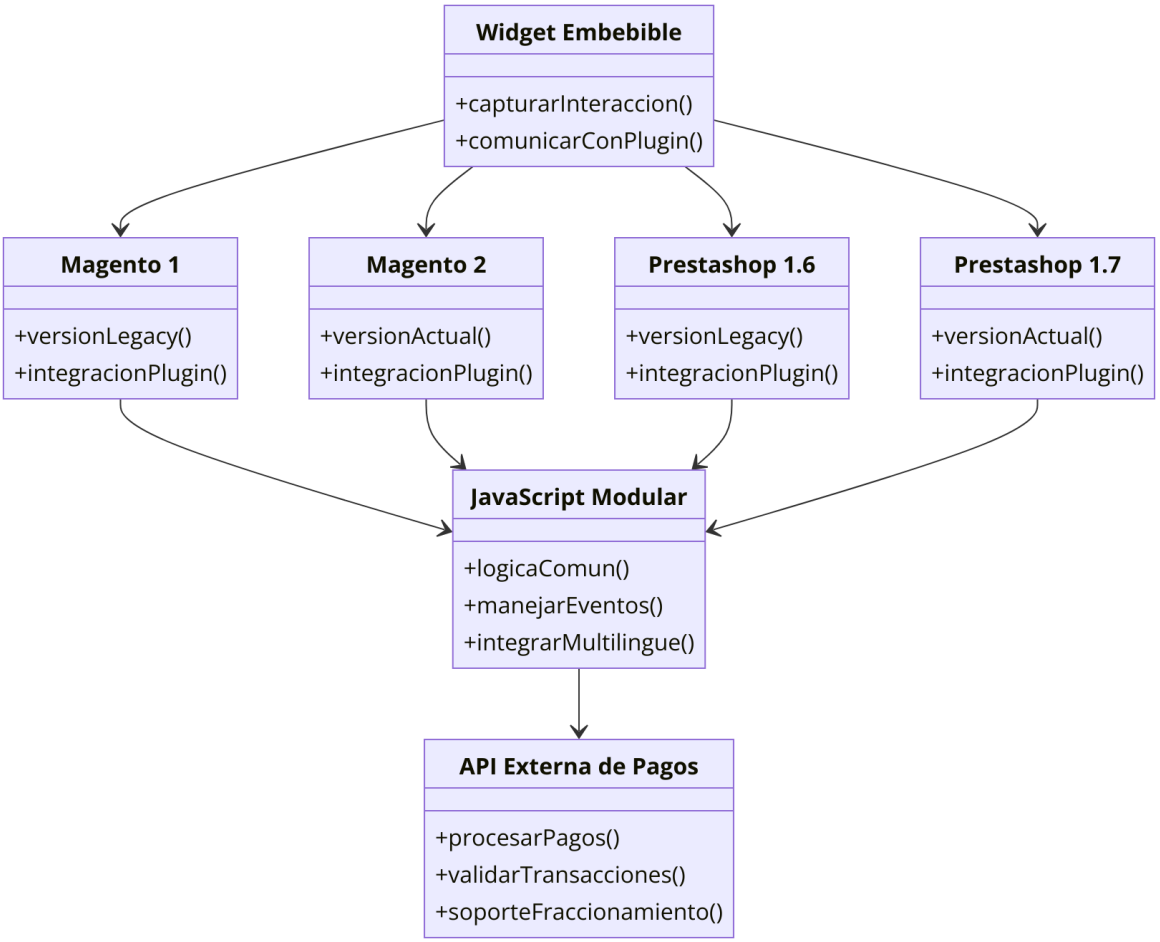

Diagrama 1. Arquitectura Conceptual del Sistema

El diseño está orientado a maximizar la reutilización de código, utilizando scripts JavaScript compartidos para manejar la lógica de pago, mientras que los plugins específicos de cada plataforma sirven como conectores que facilitan la integración con la tienda online. De esta manera, se minimiza el esfuerzo de mantenimiento al concentrar la lógica central del sistema en componentes globales, reduciendo las diferencias entre las implementaciones específicas de cada plataforma.

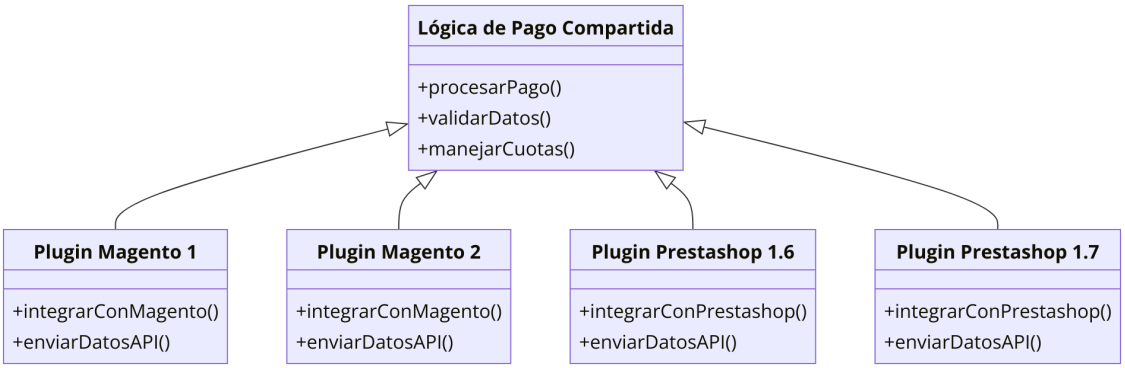

Diagrama 2. Componentes Globales

Los principales módulos incluyen: **plugins para cada plataforma**, un **conjunto de scripts JavaScript reutilizables**, y la **integración con la API de la pasarela de pago**, permitiendo manejar transacciones de manera segura y eficiente. A continuación, se detallan los componentes de esta arquitectura y sus interacciones.

Componentes del Sistema:

El diseño del módulo se basa en una arquitectura modular que facilita la integración multiplataforma y maximiza la reutilización de componentes. A continuación, se describen los principales elementos del sistema y sus responsabilidades:

1. **Plugin de la Plataforma (Magento y Prestashop):**
   Cada plataforma de e-commerce (Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7) cuenta con un plugin específico que actúa como intermediario entre la tienda online y los componentes centrales de la pasarela de pago.
   - Este plugin interactúa con las APIs de la plataforma y gestiona la comunicación con el frontend mediante scripts JavaScript.
   - Además, incluye el widget de pago que se inserta en las tiendas para que los usuarios realicen transacciones directamente en la interfaz de la tienda.
2. **Script JavaScript Compartido:**
   La lógica principal del sistema reside en un script JavaScript reutilizable que gestiona la experiencia del usuario y las llamadas a la API de la pasarela de pago.
   - Este script permite centralizar funcionalidades críticas como la validación de datos y las solicitudes a la API.
   - Su diseño compartido minimiza las diferencias entre plataformas y facilita el mantenimiento.
3. **API de la Pasarela de Pago:**
   El sistema se conecta con una API de pasarela de pago externa para realizar transacciones, validar pagos y recibir notificaciones de estado.
   - La comunicación con la API utiliza protocolos seguros (HTTPS) y técnicas como la tokenización para proteger los datos de los usuarios.

En el **Diagrama de Paquetes** se ilustra cómo estos componentes se agrupan lógicamente en diferentes módulos y cómo interactúan entre sí.

- **Plugins de Plataforma:** Cada uno es un paquete independiente que contiene las implementaciones específicas para Magento y Prestashop, incluyendo adaptadores para interactuar con el sistema central.
- **Módulo Central:** Agrupa el script JavaScript reutilizable y las conexiones a la API de la pasarela de pago, destacando la reutilización de código.
- **Dependencias Externas:** Como la API de la pasarela de pago y bibliotecas JavaScript necesarias, se representan como paquetes externos que interactúan con el sistema.

Este enfoque modular garantiza una separación clara de responsabilidades, reduce las dependencias y facilita la escalabilidad del sistema.

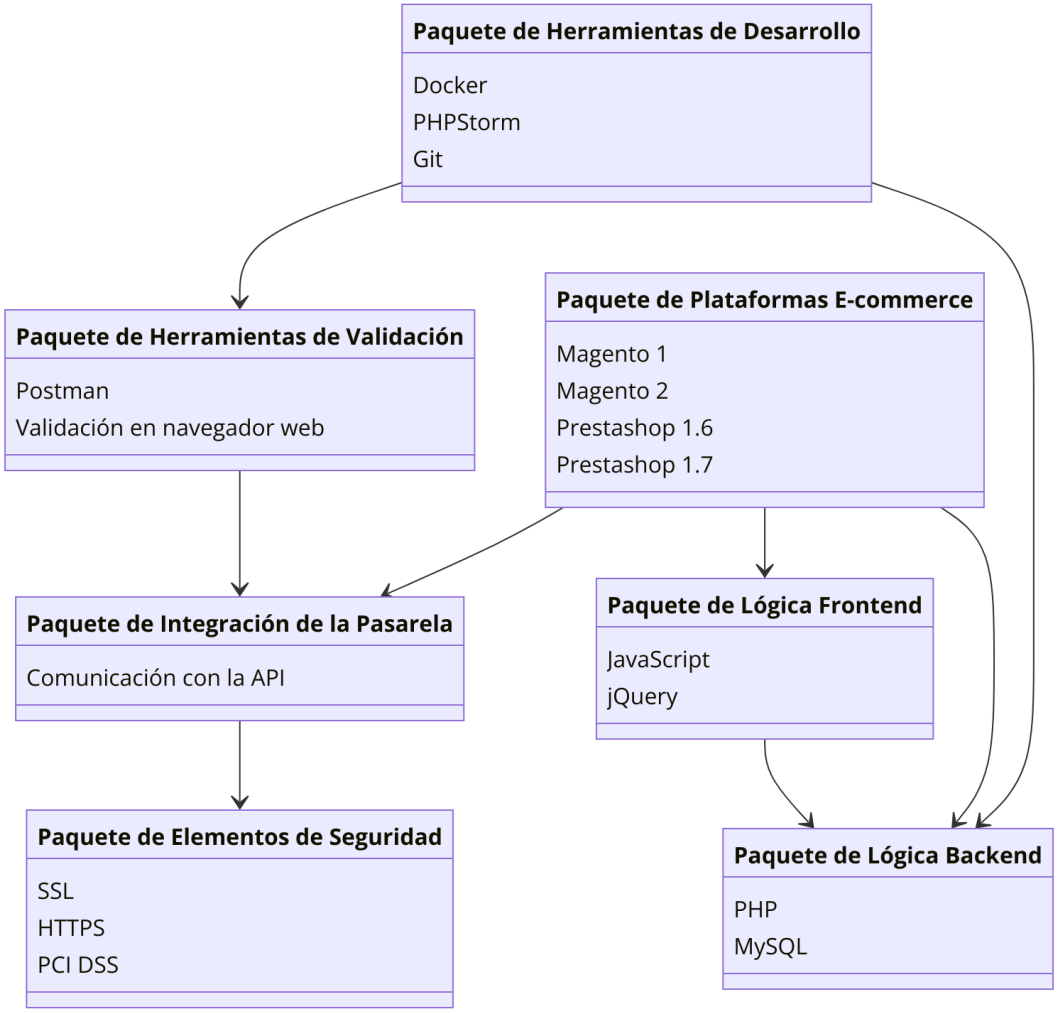

Diagrama 3. Paquetes del proyecto

Flujo de Trabajo General:

1. El usuario selecciona el método de pago en la tienda online.
2. El plugin de la plataforma carga el widget de la pasarela de pago mediante los scripts JavaScript compartidos.
3. El usuario ingresa sus datos de pago y selecciona si desea fraccionar el pago.
4. Los datos se envían a la API de la pasarela de pago.
5. La API valida la transacción y devuelve el resultado al frontend.
6. El resultado de la transacción se muestra al usuario y se actualiza el estado en el backend de la tienda.

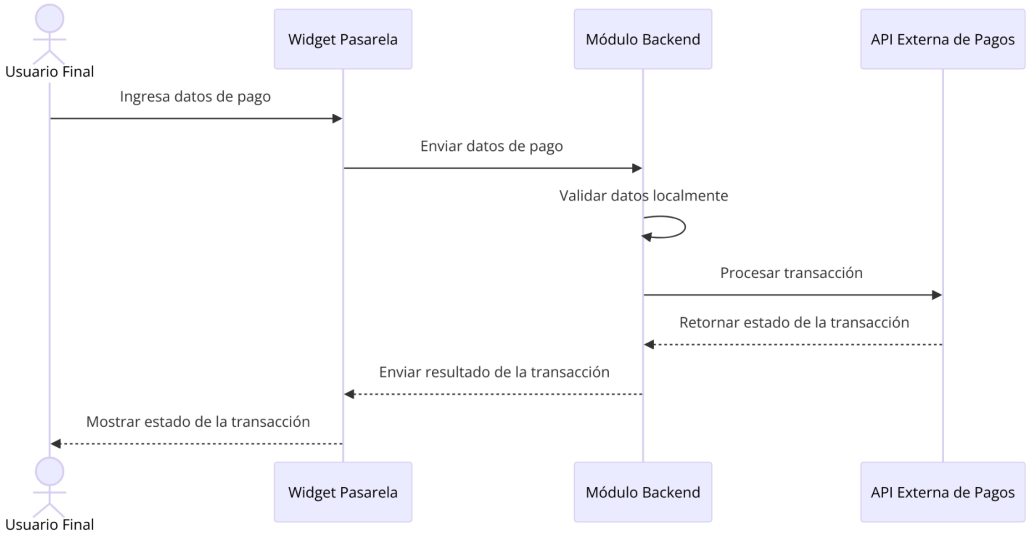

Diagrama 3. Secuencia de una transacción

Arquitectura de Despliegue

La arquitectura de despliegue se centra en cómo se distribuyen los componentes del sistema en diferentes entornos y servidores. Aquí se especifica el entorno de ejecución, las dependencias y los servicios necesarios para el funcionamiento del módulo.

Entornos de Despliegue:

1. **Entorno Local de Desarrollo:**
    - Se utiliza **Docker** para crear entornos de desarrollo consistentes. Cada plataforma (Magento o Prestashop) se despliega en un contenedor Docker, lo que facilita la replicación del entorno de producción y las pruebas.
    - El servidor local ejecuta PHP y MySQL, con los scripts JavaScript sirviendo la lógica de la pasarela de pago.

2. **Entorno de Producción:**
    - En producción, el módulo se integra en tiendas reales de e-commerce alojadas en servidores propios o en servicios de alojamiento web (cloud hosting).
    - La integración debe funcionar con servidores PHP y bases de datos MySQL, los cuales son comúnmente usados por Magento y Prestashop.
    - Los scripts JavaScript se sirven desde un CDN o desde el propio servidor de la tienda para garantizar tiempos de carga rápidos.

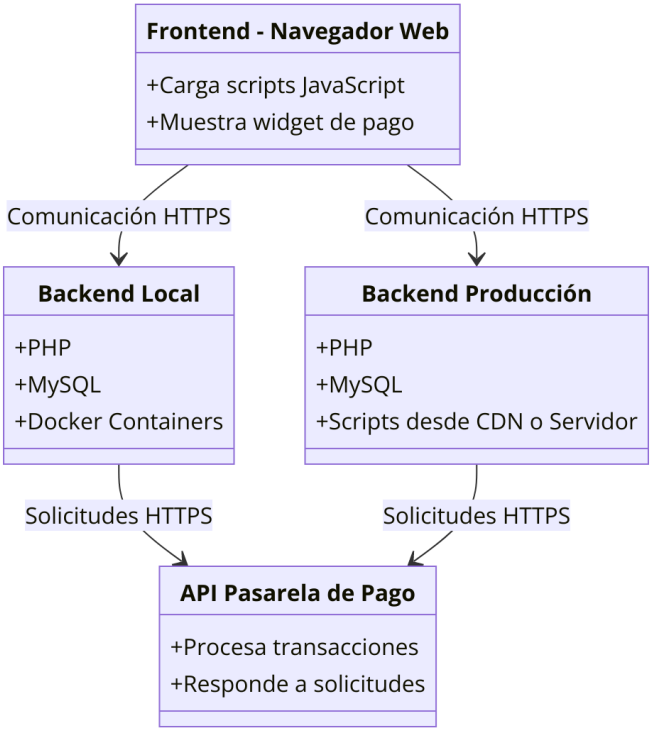

Diagrama 4. Arquitectura de Despliegue

Diagrama de Despliegue (Explicativo):

- **Frontend (Cliente):** Navegador web del usuario que carga los scripts JavaScript y muestra el widget de pago.
- **Backend de la Plataforma:** Servidor PHP ejecutando Magento o Prestashop, que gestiona el plugin de la pasarela de pago.
- **API de la Pasarela de Pago:** Servicio externo al que se realizan las solicitudes de transacción.
- **Contenedores Docker:** En el entorno local, se utilizan contenedores Docker para replicar el backend de la plataforma y la base de datos.

Consideraciones de Seguridad y Rendimiento:

- **HTTPS:** Todas las comunicaciones, tanto del frontend como del backend con la API de la pasarela de pago, deben realizarse a través de HTTPS para proteger los datos.
- **Optimización de Scripts:** Los scripts JavaScript deben estar minificados para mejorar el rendimiento y reducir los tiempos de carga.
- **Escalabilidad:** La arquitectura debe permitir la escalabilidad horizontal, permitiendo añadir más instancias del servidor backend si la carga de usuarios aumenta.

2.3 Diseño de la interfaz de usuario

La navegación de la interfaz de usuario está diseñada para ser intuitiva y sencilla. El widget de pago se insertará en las páginas de producto o en el carrito de compras de las tiendas de e-commerce, permitiendo a los usuarios iniciar el proceso de pago con un solo clic. El diseño del widget se adapta automáticamente a diferentes tamaños de pantalla (responsive), asegurando que sea fácil de usar tanto en dispositivos de escritorio como móviles. Los usuarios serán guiados a través de un flujo claro: introducción de los datos de pago, selección de la modalidad de pago fraccionado si es aplicable, y finalmente, la confirmación del pago.

El diseño de la base de datos del sistema es relativamente simple, ya que gran parte de los datos de las transacciones se gestionan a través de la API de la pasarela de pago. Sin embargo, el sistema utiliza una base de datos MySQL para almacenar configuraciones del módulo y el historial de transacciones, como el estado de los pagos y las opciones de fraccionamiento. Las tablas principales incluyen **transacciones**, **configuraciones** y **historial de pagos**. La base de datos se integra con la base de datos de cada plataforma de e-commerce para almacenar el estado del pago en el sistema de gestión de pedidos.

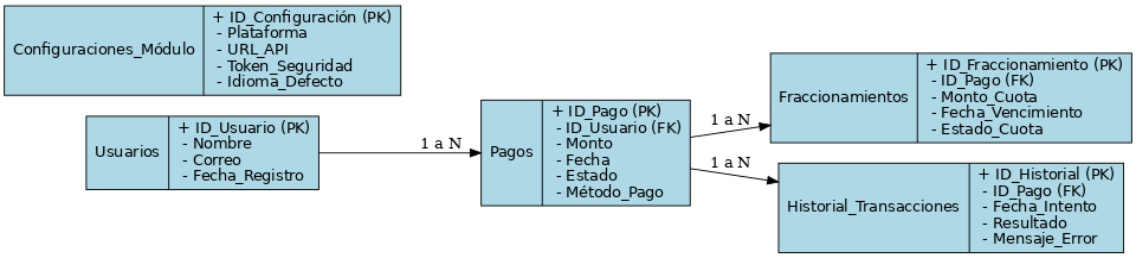

Diagrama 5. Modelo de datos

2.4 Dimensión del trabajo realizado

El proyecto se dividió en varias fases:

- Análisis y diseño (2 semanas)
- Desarrollo (7 semanas)
    - Primero, se hizo el desarrollo del módulo en Magento 2 para utilizarlo como base del proyecto (1.5 semanas).
    - El siguiente paso fue el desarrollo de la funcionalidad Javascript que abarca el grueso del proyecto (2.5 semanas).
    - La última parte del desarrollo fue la inclusión y adaptación del proyecto a las otras tres plataformas establecidas, así como ajustes a la sección Javascript para garantizar el correcto funcionamiento de manera global (3 semanas).
- Validación y pruebas (2 semanas)
- Entrega y documentación (2 semanas).

Durante esta fase se asignaron recursos y se establecieron los plazos de entrega.

La planificación final del proyecto se ajustó en función de las dificultades técnicas encontradas al integrar la pasarela de pago en las distintas plataformas. Aunque inicialmente se esperaba que el desarrollo tomara 6 semanas, se extendió a 8 semanas debido a la complejidad de la integración con las diferentes versiones de Magento y Prestashop. A pesar de esto, se cumplió con los objetivos establecidos, y se entregó la versión final con todas las funcionalidades requeridas, incluyendo la documentación técnica completa.

A lo largo del desarrollo del proyecto, se dedicaron aproximadamente **500 horas** de trabajo, distribuidas entre las siguientes fases:

- Análisis de requisitos y diseño: 80 horas.
- Desarrollo e implementación de la pasarela de pago: 260 horas.
- Pruebas y validación: 80 horas.
- Documentación y entrega final: 80 horas.

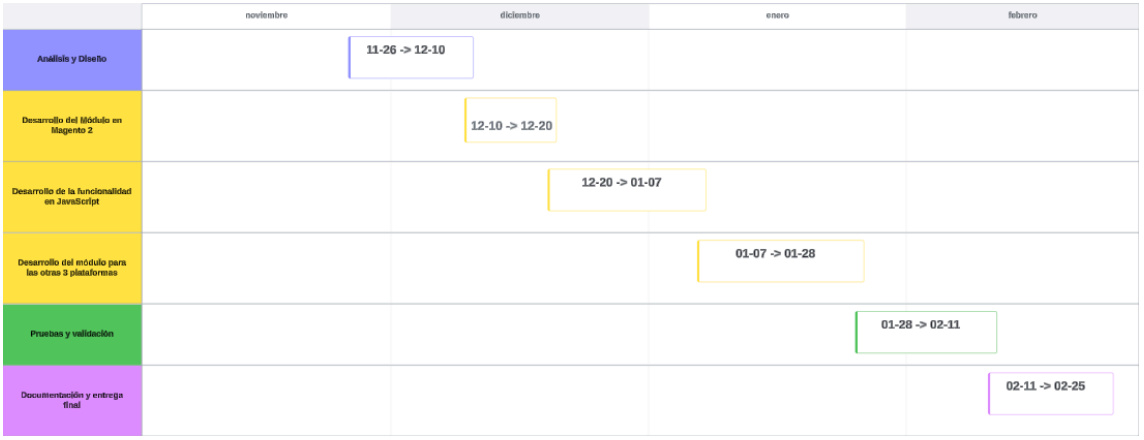

Diagrama 6. Cronograma de esfuerzos

2.5 Aspectos más complejos abordados

A lo largo del desarrollo de este proyecto, varios aspectos presentaron retos técnicos que fueron abordados de diferentes maneras. Sin embargo, el aspecto más complejo y desafiante fue la creación de una solución **JavaScript** que pudiera funcionar de manera universal en las distintas plataformas de e-commerce deseadas. La necesidad de tener un único script que pudiera integrarse y comportarse de la misma forma en todas estas plataformas fue el principal reto técnico enfrentado.

Desarrollo del script JavaScript multiplataforma

El principal desafío en este proyecto fue lograr que el **script JavaScript** que implementa la lógica de pago funcionara de forma consistente en todas las plataformas mencionadas. Si bien los módulos específicos para Magento y Prestashop fueron relativamente sencillos, la integración del script de pago que debía ser común para todas las plataformas presentó diversas dificultades. Cada plataforma tiene su propia estructura, métodos de integración y requisitos específicos, lo que hace que la implementación de un código globalizado no sea una tarea trivial.

La solución a este desafío consistió en desarrollar un código JavaScript lo más independiente posible de las plataformas. Para ello, se utilizó una serie de técnicas y patrones que permitieron minimizar la dependencia de las particularidades de cada plataforma. Entre las estrategias empleadas, se destacan las siguientes:

- **Uso de jQuery**: Para garantizar la compatibilidad y la interacción con los diferentes elementos del DOM en las distintas plataformas, se utilizó **jQuery**. Esta librería es ampliamente soportada por las plataformas mencionadas y facilitó la manipulación de elementos HTML y la ejecución de funciones relacionadas con la pasarela de pago.
- **Abstracción de la lógica de negocio**: La lógica de los pagos y las interacciones con la API de pago fue abstraída en funciones JavaScript que no dependían del entorno de la plataforma. Esto permitió que el código fuera fácilmente reutilizable sin importar la plataforma que estuviera ejecutándolo.
- **Adaptación al entorno de cada plataforma**: Aunque el código JavaScript es universal, se realizaron adaptaciones mínimas en cada plataforma para asegurarse de que el widget y el script se integraran adecuadamente con las interfaces de usuario y las bases de datos específicas de Magento y Prestashop. Estos ajustes fueron principalmente de integración y no de funcionalidad.

Al margen de este script globalizado, cada plataforma cuenta también con scripts simples para preparar la configuración y el widget, que han sido adaptados a las particularidades de cada uno de las 4 plataformas a la hora de incorporar elementos JavaScript.

Desarrollo del Widget de pago

El desarrollo del **widget de pago** que se inserta en las páginas del e-commerce también presentó algunos desafíos, aunque estos estuvieron más relacionados con la falta de

experiencia en el área de desarrollo **front-end**. El widget debe ser sencillo, eficiente y visualmente atractivo para los usuarios, lo que implicó gestionar la correcta integración del HTML, CSS y JavaScript en el contexto de cada plataforma.

Aunque no se trató de una dificultad técnica tan grande como la parte de JavaScript multiplataforma, el trabajo con el **diseño de la interfaz** y la **adaptación a los diferentes temas visuales** de Magento y Prestashop requirió una curva de aprendizaje, especialmente en cuanto a cómo integrar el widget de manera que no alterara la estética ni el funcionamiento de las tiendas.

Comunicación con la API de pago

Otro aspecto importante, aunque no tan complejo, fue la integración con la **API de pago proporcionada por el cliente**. La pasarela de pago y sus especificaciones fueron entregadas como una interfaz lista para ser consumida por el módulo, lo que simplificó considerablemente esta parte del proyecto. El módulo simplemente se encargó de enviar los datos a la API y gestionar las respuestas (como confirmación o error).

Aunque este aspecto fue relativamente sencillo, resultó fundamental para el éxito del proyecto, ya que toda la funcionalidad de procesamiento de pagos depende de esta API. El principal reto aquí fue garantizar la seguridad y la correcta transmisión de los datos sensibles durante la transacción, lo cual se resolvió utilizando **protocolos de seguridad estándar** como HTTPS y asegurando que los datos se manejaron de acuerdo con las mejores prácticas de protección de información.

En resumen, los **aspectos más complejos** del proyecto se centraron principalmente en la creación de un **script JavaScript universal** que fuera capaz de integrarse correctamente con las cuatro plataformas de e-commerce mencionadas. Esto requirió una solución que minimizara las dependencias específicas de cada plataforma y permitiera una integración sin fricciones en todos los entornos. Aunque otros aspectos como el desarrollo del widget de pago y la integración con la API fueron también importantes, estos representaron menos complejidad técnica, aunque contribuyeron de manera significativa al resultado final del proyecto.

3 LECCIONES APRENDIDAS Y CONCLUSIONES

3.1 Conocimientos adquiridos

A lo largo del desarrollo de este proyecto, se adquirieron una serie de conocimientos y habilidades tanto en el ámbito técnico como en la gestión de un proyecto de desarrollo de software. A continuación, se destacan los principales aprendizajes que resultaron de la implementación de la pasarela de pago multiplataforma.

Desarrollo y gestión de código multiplataforma

Uno de los aprendizajes más significativos fue el desarrollo de un **sistema común** que funcionara de manera eficiente en plataformas de e-commerce diferentes como Magento y Prestashop. Este reto permitió profundizar en el concepto de **abstracción de código**, buscando soluciones que minimizaran las diferencias entre entornos y maximizaran la reutilización del código. Se aprendió a escribir un **JavaScript limpio y reutilizable**, que pudiera integrarse de manera eficiente en distintas plataformas sin depender de características específicas de estas.

El desafío de **integrar una única lógica de negocio** en un entorno multiplataforma permitió mejorar las habilidades en la creación de **código desacoplado**. Esta experiencia enseñó la importancia de tener en cuenta las limitaciones y particularidades de cada plataforma, sin sacrificar la flexibilidad y la escalabilidad del sistema.

Trabajo con tecnologías y herramientas de desarrollo

Además de los conocimientos adquiridos en términos de programación, este proyecto también permitió familiarizarse y mejorar la destreza con diversas **herramientas de desarrollo**. En particular, el uso de **jQuery** como librería para interactuar con los distintos elementos de las plataformas facilitó la comprensión de cómo funcionan los navegadores y cómo optimizar el código para que funcione de manera eficiente en todos los entornos. También se aprendió a configurar y utilizar entornos de desarrollo con **Docker**, lo cual proporcionó una visión más clara de cómo manejar entornos aislados y mejorar la **portabilidad** de las aplicaciones.

Desarrollo de widgets front-end y mejora en habilidades de diseño de interfaces

Otro aspecto importante del proyecto fue el desarrollo del **widget de pago** que debía integrarse con el sitio web de manera que no alterara la experiencia del usuario. A lo largo de este proceso, se adquirió una mayor comprensión de las **principales técnicas de desarrollo front-end**. Aunque este reto no fue tan complejo como otros aspectos, representó un proceso de aprendizaje importante, especialmente en lo relacionado con el **trabajo con HTML, CSS y JavaScript** para la creación de componentes interactivos que se adaptaran a diferentes estructuras de diseño.

Integración con APIs externas y gestión de seguridad

Un área clave de conocimiento adquirido fue la **integración de la pasarela de pago** con la API externa proporcionada por el cliente. Este proceso implicó entender cómo comunicarse con una API externa de manera segura, cómo manejar **transacciones de datos sensibles** y cómo garantizar la seguridad y confidencialidad de la información. Aunque la integración fue relativamente sencilla, este ejercicio permitió conocer las mejores prácticas para la **protección de datos** en la web, incluyendo el uso de HTTPS y la validación de la seguridad en cada paso de la transacción.

Gestión de proyectos y aprendizaje de la planificación

Finalmente, el trabajo en este proyecto permitió mejorar las habilidades en **gestión de proyectos**, especialmente en lo que respecta a la **planificación de tareas** y la **organización del trabajo** en función de los plazos. A lo largo del proceso, fue fundamental identificar las tareas más complejas y distribuir el tiempo de forma adecuada para abordarlas de manera eficiente. Además, se adquirió experiencia en la **documentación** del proceso de desarrollo, lo que permitió explicar de manera clara los aspectos técnicos del sistema, lo cual resulta crucial tanto para la colaboración en equipo como para la presentación final del proyecto.

En resumen, este proyecto no solo permitió profundizar en el desarrollo de sistemas multiplataforma y la creación de aplicaciones web, sino que también brindó una comprensión más amplia de cómo gestionar de manera eficiente un proyecto de software desde el punto de vista técnico y organizativo. Las habilidades adquiridas en **JavaScript**, **integración con APIs externas**, **seguridad web** y **gestión de proyectos** serán de gran valor en futuras iniciativas y proyectos relacionados con el desarrollo de software.

3.2 Ideas Futuras

A medida que la pasarela de pago multiplataforma avanza, existen varias **líneas de mejora** y **expansión** que podrían explorar en el futuro para incrementar su funcionalidad, robustez y facilidad de uso. Algunas de las principales ideas para el futuro incluyen:

Ampliación de plataformas soportadas

Actualmente, el módulo está diseñado para funcionar en cuatro plataformas de e-commerce principales: Magento 1, Magento 2, Prestashop 1.6 y Prestashop 1.7. Sin embargo, con el tiempo podría ser conveniente **expandir la compatibilidad** a otras plataformas de e-commerce populares como WooCommerce, Shopify o OpenCart. Esto permitiría llegar a una mayor base de usuarios y diversificar las opciones de integración para los comercios online.

Mejoras en la interfaz de usuario del widget de pago

Si bien el **widget de pago** desarrollado es funcional y cumple con su propósito, siempre existe margen para mejorar la **experiencia del usuario**. En futuras versiones, se podría invertir en un diseño más atractivo y moderno, con opciones de personalización adicionales que permitan a los comerciantes adaptar el widget a la estética de su tienda. Además, se podrían añadir funcionalidades como la opción de **pago en un solo clic** o la integración de opciones de **autocompletado** de datos para simplificar aún más el proceso de pago.

Optimización de la seguridad y cumplimiento normativo

Dado que las pasarelas de pago manejan información sensible, siempre hay espacio para mejorar la **seguridad**. En el futuro, sería conveniente incorporar nuevas **medidas de protección** como la autenticación **2FA (Autenticación en dos pasos)** o la **tokenización de tarjetas de crédito** para reducir riesgos de fraude. También sería importante estar al tanto de la **evolución de las normativas** internacionales de protección de datos, como el **RGPD** (Reglamento General de Protección de Datos) en Europa, y adaptar el sistema para cumplir con los estándares más recientes.

Soporte para múltiples idiomas y monedas

Aunque en la versión inicial no se incluyeron funciones multilingües ni soporte para múltiples monedas, **la internacionalización** podría ser un aspecto clave a abordar en el futuro. La pasarela de pago podría ser adaptada para **soportar varios idiomas** y permitir a los comercios operar en diferentes **monedas locales**, lo cual es crucial para tiendas online que operan en mercados internacionales. Esto no solo mejoraría la usabilidad de la pasarela, sino que también ampliaría su aplicabilidad a nivel global.

Mejora del rendimiento y escalabilidad

Aunque el sistema actual es eficiente, siempre se puede mejorar en cuanto a **rendimiento** y **escalabilidad**. A medida que el volumen de transacciones aumente, sería necesario optimizar ciertos aspectos del código, como la gestión de solicitudes concurrentes o la minimización de latencias en la comunicación con la API de pagos. En este sentido, también se podría explorar el uso de **caching** o la implementación de una **arquitectura orientada a microservicios**, que permitiría una mayor flexibilidad y escalabilidad del sistema.

Soporte para nuevas tecnologías de pago

A medida que el mercado de pagos online continúa evolucionando, sería valioso adaptar la pasarela para aceptar **nuevas tecnologías de pago**, como pagos con **criptomonedas** o la integración con **billeteras digitales** como **Apple Pay**, **Google Pay** o **PayPal**. La integración con estos métodos de pago podría atraer a una base de usuarios más amplia y aprovechar las nuevas tendencias en el sector de los pagos online.

3.3 Conclusiones

El desarrollo del módulo de pasarela de pago multiplataforma ha sido un desafío interesante que ha permitido explorar y profundizar en varios aspectos clave del desarrollo de software, desde la integración de sistemas hasta el diseño de una interfaz de usuario efectiva y segura. El proyecto ha cumplido con el objetivo principal de proporcionar una solución sencilla y efectiva para permitir pagos online a través de diferentes plataformas de e-commerce (Magento y Prestashop), destacando la capacidad de desarrollar un sistema que minimice la dependencia de código específico de cada plataforma, aprovechando una solución basada en JavaScript.

Uno de los principales logros del proyecto ha sido la creación de un sistema común que permita el uso de un código **global** para las plataformas soportadas, optimizando el esfuerzo de desarrollo y manteniendo una arquitectura modular que facilita futuras expansiones y adaptaciones. Esta solución ha demostrado la viabilidad de integrar una funcionalidad compleja como una pasarela de pago en diversas plataformas sin necesidad de realizar grandes modificaciones específicas para cada una.

Además, el trabajo realizado ha permitido adquirir valiosos conocimientos en diversas tecnologías y herramientas, como **JavaScript**, **jQuery**, **Docker** y la integración con **APIs externas**, lo que ha contribuido al desarrollo tanto de habilidades técnicas como de gestión de proyectos. También ha quedado claro que la **seguridad** en las transacciones online es un aspecto crucial en este tipo de sistemas, y el proyecto ha logrado abordar este aspecto mediante la integración de prácticas recomendadas en cuanto a protección de datos.

Aunque el proyecto ha cumplido con los requisitos establecidos y ha alcanzado los objetivos propuestos, también han quedado identificadas áreas de mejora y expansión que podrían explorarse en el futuro. La **internacionalización**, la **mejora de la experiencia de usuario** y la **adaptación a nuevas tecnologías de pago** son aspectos que podrían enriquecer aún más la solución y hacerla más robusta y competitiva en un mercado en constante evolución.

En resumen, este proyecto no solo ha permitido desarrollar una solución técnica funcional, sino que también ha servido como una experiencia de aprendizaje significativa en cuanto a la resolución de problemas complejos y la mejora continua en el desarrollo de software. Se considera que esta pasarela de pago tiene un alto potencial para evolucionar y adaptarse a nuevas necesidades, ofreciendo una base sólida para futuras implementaciones y adaptaciones en el ámbito de los pagos online.

4 BIBLIOGRAFÍA

[1] **Hiberus Tecnología, S.A.**. Consultoría de negocio y servicios tecnológicos. Disponible en: https://www.hiberus.com

[2] **Magento Open Source Documentation**. *Magento DevDocs*. Adobe Inc. Disponible en: https://developer.adobe.com/commerce

[3] **Prestashop Developer Guide**. *Prestashop DevDocs*. Prestashop S.A. Disponible en: https://devdocs.prestashop-project.org/

[4] **JavaScript: The Definitive Guide**. *David Flanagan*. O'Reilly Media, 2011. Disponible en: https://www.oreilly.com

[5] **jQuery Documentation**. jQuery Foundation. Disponible en: https://api.jquery.com/

[6] **Docker: Up & Running**. *Karl Matthias y Sean P. Kane*. O'Reilly Media, 2015. Disponible en: https://www.oreilly.com

[7] **Vagrant: Up and Running**. *Mitchell Hashimoto*. O'Reilly Media, 2017. Disponible en: https://www.oreilly.com

[8] **Git Documentation**. Git Project. Disponible en: https://git-scm.com/doc

[9] **MySQL 8.0 Reference Manual**. Oracle Corporation. Disponible en: https://dev.mysql.com/doc/

[10] **PHPStorm Documentation**. JetBrains. Disponible en: https://www.jetbrains.com/phpstorm/guide/

[11] **Postman API Platform Documentation**. Postman, Inc. Disponible en: https://learning.postman.com/

[12] **Patrones de Diseño: Elementos de Software Reutilizable Orientado a Objetos**. *Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides*. Pearson Education, 1995. Disponible en: https://www.pearson.com

[13] **MDN Web Docs: Seguridad en Aplicaciones Web**. *Mozilla Foundation*. Disponible en: https://developer.mozilla.org/es/docs/Web/Security

[14] **The Twelve-Factor App**. *Heroku*. Disponible en: https://12factor.net/

5 Anexo I. Guia de Instalación del Plugin

Instalación Manual del Módulo en un Entorno Existente

Para instalar el módulo en un entorno ya existente, sigue los siguientes pasos:

1. **Descomprimir el módulo:**
   Extraer el contenido del archivo ZIP proporcionado.
2. **Copiar los archivos a las rutas correspondientes:**
   - Magento: Copiar los archivos a la carpeta `app/code/` según la versión de Magento.
   - Prestashop: Copiar la carpeta del módulo a `modules/`.
3. **Activar el módulo:**
   - En Magento, ejecutar el comando: `php bin/magento setup:upgrade`
   - En Prestashop, activar desde el panel de administración.
4. **Limpiar la caché de la plataforma:**
   Para garantizar que los cambios se reflejen inmediatamente.

6 Anexo III. Trazas de código y de peticiones

Ejemplo JSON de peticion a la API


```
{
  "enabled": true,
  "js_version": "1.0",
  "version": "1",
  "transactionId": "616740F5-6486-4733-9F31-76DCEC284ACF",
  "date": 20181228,
  "widgetSections": [
    {
      "section": "homepage",
      "dom": "banner_4",
      "position": "before",
      "dom_var_price": null,
      "content": "homepage",
      "dom_lblPayment": null,
      "dom_btnPopup": null,
      "dom_btnPopupInc": null,
      "dom_btnPopupDec": null,
      "dom_lblPopupPayment": null,
      "dom_lblPopupTerm": null,
      "dom_cntPopupInfo": null,
      "dom_btnPopupClose": null,
      "dom_btnInc": null,
      "dom_btnDec": null,
      "dom_btnInfo": null,
      "dom_lblTerm": null,
      "dom_btnToolTip": null,
      "dom_cntToolTip": null
    },
    {
      "section": "catalog",
      "dom": ".catalog-category-view .products-grid .price-box .regular-price",
      "position": "before",
      "dom_var_price": ".product-item-price .price",
      "content": "catalog",
      "dom_lblPayment": "lblPayment",
      "dom_btnPopup": "btnPopup",
      "dom_btnPopupInc": "btnInc",
      "dom_btnPopupDec": "btnDec",
      "dom_lblPopupPayment": "lblPayment",
      "dom_lblPopupTerm": "lblTerm",
```

```
    "dom_cntPopupInfo": "lblInfo",
    "dom_btnPopupClose": "btnX,btbVolver",
    "dom_btnInc": null,
    "dom_btnDec": null,
    "dom_btnInfo": null,
    "dom_lblTerm": null,
    "dom_btnToolTip": null,
    "dom_cntToolTip": null
  },
  {
    "section": "product",
    "dom": ".catalog-product-view .product-view .price-info .price",
    "position": "after",
    "dom_var_price": "price",
    "content": "product",
    "dom_lblPayment": "lblPayment",
    "dom_btnPopup": null,
    "dom_btnPopupInc": null,
    "dom_btnPopupDec": null,
    "dom_lblPopupPayment": null,
    "dom_lblPopupTerm": null,
    "dom_cntPopupInfo": null,
    "dom_btnPopupClose": null,
    "dom_btnInc": "btnInc",
    "dom_btnDec": "btnDec",
    "dom_btnInfo": "btnInfo",
    "dom_lblTerm": "lblTerm",
    "dom_btnToolTip": "btnToolTip",
    "dom_cntToolTip": "lblToolTip"
  },
  {
    "section": "checkout",
    "dom": "final_price",
    "position": "after",
    "dom_var_price": "price",
    "content": "checkout",
    "dom_lblPayment": "lblPayment",
    "dom_btnPopup": null,
    "dom_btnPopupInc": null,
    "dom_btnPopupDec": null,
    "dom_lblPopupPayment": null,
    "dom_lblPopupTerm": null,
    "dom_cntPopupInfo": null,
    "dom_btnPopupClose": null,
    "dom_btnInc": "btnInc",
```

```
      "dom_btnDec": "btnDec",
      "dom_btnInfo": "btnInfo",
      "dom_lblTerm": "lblTerm",
      "dom_btnToolTip": "btnToolTip",
      "dom_cntToolTip": "lblToolTip"
    }
  ],
  "financialOffer": [
    {
      "termTotal": 3,
      "termDeferred": 0,
      "minAmount": 1000,
      "maxAmount": 60,
      "openingFeeType": "N",
      "openingFeePct": 0,
      "openingFeeFixAmt": 0,
      "tin": 0
    },
    {
      "termTotal": 6,
      "termDeferred": 0,
      "minAmount": 1000,
      "maxAmount": 60,
      "openingFeeType": "N",
      "openingFeePct": 0,
      "openingFeeFixAmt": 0,
      "tin": 0
    },
    {
      "termTotal": 9,
      "termDeferred": 0,
      "minAmount": 1000,
      "maxAmount": 60,
      "openingFeeType": "N",
      "openingFeePct": 0,
      "openingFeeFixAmt": 0,
      "tin": 0
    },
    {
      "termTotal": 12,
      "termDeferred": 0,
      "minAmount": 1000,
      "maxAmount": 60,
      "openingFeeType": "N",
      "openingFeePct": 0,
```

```
        "openingFeeFixAmt": 0,
        "tin": 0
      }
   ]
}
```

Fragmento del JS global de operaciones con la API

```
/***
* funcion global para caclular los datos financieros y los textos de info
***/
function pepper (vPepperData){
        return pepperCalculator(vPepperData);
}

/***
* funcion que recibe el JSON en bruto y lo convierte a objetos para que se interprete por JavaScript
***/
function pepperJsonParser(vPepperData){
        // Parseamos JSON de entrada para convertirlo en un objeto
        var pepperData = JSON.parse(vPepperData);
        // separamos objeto en oferta comercial y en textos
        return{
                financialOffer : pepperData.calculator.financial_offer,
                texts : pepperData.calculator.eco_texts
        };
}

/***
* funcion base que retorna todos los valores desaeados
***/
function pepperCalculator(vPepperData) {
        // llamamos a la funcion de parseado
        var jsonData = pepperJsonParser(vPepperData);
        // variables "globales" y objetos de uso intero
        var financialOffer = jsonData.financialOffer;
        var texts = jsonData.texts;
        var len = financialOffer.length; // imporante para saber el tamaño del objeto JSON de entrada para recorrerlo

        // Inicializacion de clases
        var cFunctionalities = pepperFunctionalities();

        // Ordenamos el objeto con la oferta comercial por la cuota
```

```
financialOffer = financialOffer.sort(fSortByTerms);

// Funcion que usaremos para ordenar nuestra oferta comercial
function fSortByTerms(a, b) {
  if (a.term_total === b.term_total) {
    return 0;
  }
  else {
    return (a.term_total < b.term_total) ? -1 : 1;
  }
}

// funcion base que dvuelve todos los calculos
function fCalculator(amount, term){
        var dataCalc = fGetDataCalc(amount, term);

        return   pepperMapper().ReturnDataMapper(dataCalc);
}

// funcion para calcular el texto del toolTip
function fGetHTMLToolTip(amount, term){
        // validamos datos de entrada
        var message = null;
        var messages = [];

        message = cFunctionalities.ValidateDataEntry("amount", amount);
        if (message !== null){
                messages.push(message);
        }

        message = cFunctionalities.ValidateDataEntry("term", term);
        if (message !== null){
                messages.push(message);
        }

        if (messages.length > 0){
                cFunctionalities.ThrowJsException(messages,
"fGetHTMLToolTip");
        }

        var calc = fCalculator(amount, term);
        var HTMLToolTip = texts.html_tool_tip;
        HTMLToolTip = fReplaceTextToolTip(HTMLToolTip, calc);
        return HTMLToolTip;
}
```