INGLÉS

**1. Problem Statement**

Develop a global marketplace platform for handcrafted products that removes payment and commission barriers between countries while enhancing the user experience.

The solution will enable customers to purchase handcrafted goods with just a few clicks, supported by an intuitive interface.

**2. Impact and Benefits of the Proposed Solution to the Business**

Business Benefits

* Recurrent and predictable income from each transaction (commission-based model).
* Technological advantage by implementing interoperable payments with ILP and Open Payments, which few platforms are currently using.
* User loyalty thanks to a seamless and fair shopping experience.
* Socially responsible image by supporting financial inclusion and cultural preservation, strengthening the brand.

Business Impact

* Profitability from the first year: with a 5% commission, a steady flow of income is generated, reaching sustainability without relying on subsidies.
* Scalability: the model grows with the number of buyers and artisans without costs increasing at the same rate.
* Competitive positioning: by offering lower commissions than Amazon Handmade or Etsy, the platform becomes more attractive to both artisans and buyers.
* Access to a global market: opens new business opportunities in different countries, expanding both cultural and economic reach.

**3. Technical Solution Description**

The proposed system focuses on optimizing international payments and reducing commissions in the trade of handcrafted products through the integration of the Interledger Protocol (ILP) and the Open Payments standard.

3.1 System Components

I. Interoperable Payment Architecture

* Interledger Protocol (ILP): routes transactions by dividing them into micro-packets that travel from node to node until reaching the destination, ensuring security, traceability, and complete delivery.
* Open Payments: provides a standardized framework to initiate, validate, and settle payments without exposing users’ banking data.

II. Transaction Flow

* The buyer makes the payment in their local currency via their digital wallet.
* The Administrator (intermediary) receives and validates the purchase order, ensuring that the product is properly managed.
* Once delivery is confirmed, the payment is released to the artisan’s wallet through ILP, with automatic currency conversion and real-time confirmation.

III. Security and Transparency

* All transactions are encrypted through the ILP protocol.
* The system ensures that artisans receive the full amount minus a reduced 5% commission, improving their margin compared to traditional platforms.
* The administrator only acts as a validator, without unnecessarily holding funds, which strengthens trust in the platform.

IV. Technical Infrastructure

* Frontend: developed in HTML, CSS, and JavaScript, with an interface optimized for quick navigation and purchases.
* Backend: built with Node.js, responsible for managing authentication, inventory, transactions, and payment protocol connections.
* Database: MySQL/PostgreSQL for secure storage of users, products, and orders.
* Data Exchange: JSON as a lightweight communication format between frontend, backend, and external systems.

**4. Technical Implementation Details**

4.1 Functional and Non-Functional Requirements

Functional:

* International payment processing through ILP + Open Payments.
* Queries on sales, revenue, and transactions.

Non-Functional:

* Security: encryption of transactions and protection of personal data.
* Scalability: ability to support thousands of simultaneous transactions with minimal latency.
* Compatibility: integration with multiple wallets.
* Usability: intuitive and adaptable interface.

4.2 Justification of Technologies Used

* Frontend (HTML, CSS, JavaScript):  
  Chosen for their universal compatibility with browsers, low development cost, and ease of maintenance. They also allow for a lightweight and adaptable interface, scalable to new functionalities without high licensing costs.
* Backend (Node.js):  
  Selected for its high performance in real-time applications and its ecosystem of libraries (NPM), which facilitates integration with external APIs, such as international payment systems. Compared to other frameworks, Node.js offers better scalability and lower resource consumption.
* Database (MySQL/PostgreSQL):  
  Selected for being open source, with large support communities and capable of handling large volumes of structured data. Compared to proprietary databases (e.g., SQL Server or Oracle), they represent a considerable reduction in costs without sacrificing performance or security.
* Data Exchange (JSON):  
  Chosen for its lightness in communication between frontend and backend, and its native compatibility with JavaScript and modern APIs. This improves efficiency and response speed compared to heavier formats like XML.
* Payment Protocols (Interledger + Open Payments):  
  Selected because they eliminate dependency on traditional financial intermediaries, reduce transaction costs, and enable global interoperability of real-time payments — something that no traditional gateway (PayPal, Western Union, banks) guarantees at this level.

4.3 Security Considerations (Hardening)

To ensure the protection of users, data, and transactions, the following measures are adopted:

* Encryption of sensitive data (passwords, payment information) using secure hashing algorithms (e.g., bcrypt, SHA-256).
* Protection against common attacks (SQL Injection) through strict input validation and data sanitization.

**5. Prototyping**

5.1 Initial Prototype

The first prototype corresponds to a wireframe showing the general structure of the product catalog.

* Each product has its image, name, price, and a button to add it to the cart.
* The user can access the cart at any time through the “View Cart” button.
* This design aimed to validate usability and the arrangement of elements before implementing the final version.

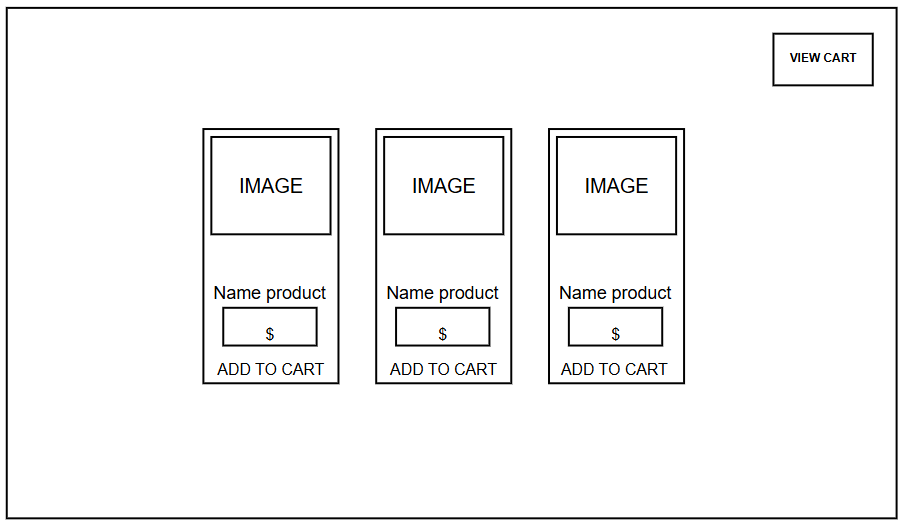


Illustration 1. Initial prototyping design  
Source: Own elaboration.

5.2 Complete Functional Prototype

The advanced prototype was developed with an improved graphical interface (colors, visual style) and with the implemented functionalities, including:

* Adding products to the shopping cart.
* Viewing the cart with a list of selected items.
* Approving the purchase by confirming quantity and total amount.
* Confirming payment through the Interledger Protocol (ILP), with real-time validation.
* Full transaction flow from product selection to payment confirmation.

The functional prototype not only represents the interface but also integrates business logic, database communication, and initial security (hardening) tests.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

Illustration 2. Complete functional prototype  
Source: Own elaboration.

**6. Source Code and Database Script**

6.1 Backend (BE), Frontend (FE) and Database Scripts (DB)

The system uses a central table called *pedido* (order), where all transactions carried out on the platform are recorded, from order creation to settlement with the artisan.

CREATE TABLE pedido (

id int(11) NOT NULL AUTO\_INCREMENT,

Transaccion varchar(50) NOT NULL,

Producto varchar(50) NOT NULL,

Cantidad int(11) NOT NULL,

Total decimal(10,2) NOT NULL,

wallet\_address\_artesano varchar(100) NOT NULL,

Status varchar(50) NOT NULL,

Divisa varchar(50) NOT NULL,

fecha\_creacion timestamp NOT NULL DEFAULT current\_timestamp(),

PRIMARY KEY (id)

) ENGINE=InnoDB AUTO\_INCREMENT=29

DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

Design justification:

* The table centralizes all the necessary information to process orders and transactions in a single record.
* The inclusion of *fecha\_creacion* allows for auditing and chronological ordering of operations.
* The *Status* field facilitates tracking the order lifecycle (pending → confirmed → settled).
* The use of *wallet\_address\_artesano* removes dependence on traditional banking information, aligning with the philosophy of the ILP + Open Payments–based system.

**7. Additional Documentation**

7.1 Technical Documentation

1. **Introduction**  
PaDuki is an e-commerce system designed to facilitate the sale of handcrafted products.

Its main innovation lies in the use of the Interledger API, which provides several key benefits:

1. Interoperable payments
2. Micropayments
3. Transaction speed
4. Cost reduction
5. Security and decentralization

**2. System Objectives**

General Objective  
Develop an inclusive e-commerce platform that facilitates the sale of handcrafted products in national and international markets, using the Interledger API to enable fast, secure, and accessible payments. Its aim is to empower artisans and preserve the cultural value of their creations in a globalized world.

Specific Objectives:

* Facilitate artisans’ digitalization  
  PADUKI provides a user-friendly and intuitive platform where artisans can publish, manage, and sell their products without requiring advanced technological experience.
* Enable accessible global payments  
  Thanks to the integration of the Interledger API, artisans receive international payments instantly, with reduced costs and without depending on traditional banking barriers.
* Ensure security and trust  
  The system includes robust authentication, permission management, and logging, ensuring that every transaction is transparent and reliable for both buyers and sellers.
* Connect cultures globally  
  Through the platform, buyers from all over the world can discover and purchase unique handcrafted pieces, promoting cultural exchange and appreciation of handmade art.
* Optimize buying and selling processes  
  The system ensures speed and efficiency in each transaction thanks to its architecture based on data caching, information synchronization, and basic services that guarantee product availability at all times.
* Promote the sustainability of artisanal commerce  
  PADUKI rescues the value of handmade goods versus industrialized production, ensuring that economic benefits go directly to creators, thus strengthening local economies.

This technical document is aimed at developers with experience in web technologies, particularly those skilled in:

* Node.js: for building business logic, API services, and server management.
* HTML and CSS: for developing and customizing user interfaces (UI/UX).
* API integration: especially the Interledger API, which allows the management of secure, low-cost international payments.

**3. System Architecture**

System architecture diagrams

Figure 1 shows the proposed architecture for the PaDuki e-commerce system, structured in layers that provide greater organization, scalability, and security. Each layer fulfills a specific role within the purchase and payment flow.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Illustration 3. System Architecture Diagram  
Source: Own elaboration

4. Technologies Used

Table 1 presents the main technologies used in the development of the PaDuki e-commerce platform, along with their respective functions within the system.

|  |  |
| --- | --- |
| Technology | Description / Function |
| HTML | Structures the e-commerce web pages, displaying products and artisan information. |
| CSS | Defines the visual style and appearance of the platform, making the interface attractive and consistent with the project’s identity. |
| JavaScript | Adds interactivity to the frontend, handles events, validates forms, and enables communication with the backend. |
| Node.js | Backend that manages users, products, authentication, transactions, and business logic, executing JavaScript on the server. |
| Visual Studio Code | Development environment used to write, debug, and organize all project code (frontend and backend). |
| MySQL | Relational database that securely stores information about products, artisans, users, and transactions in a structured way. |
| JSON | Data exchange format between frontend and backend, allowing lightweight and efficient information transfer. |
| Interledger | International payment system that connects different digital money networks, enabling fast, secure, and low-cost transactions for artisans and clients. |

5. Main Modules

I. Backend (pagos.js)  
The *pagos.js* file is the core of the backend system. It is responsible for launching the server with Express, enabling CORS, serving static files, and defining the main API routes (endpoints).

Figure 1 shows the backend flow, detailing how the server processes frontend requests, manages GET/POST routes, and communicates with the Open Payments network to handle transactions.

Diagrama

El contenido generado por IA puede ser incorrecto.

Illustration 4. Backend flow  
Source: Own elaboratio

II. Frontend – Main Page (client.js)

The *client.js* script controls the interactivity of the *index.html* page. Its main responsibility is to allow the user to add products to the cart. For this, it uses an internal map (*productWallets*) that links products to wallets, manages the cart in *sessionStorage*, and generates notifications through the *showToast()* function.

Figure 2 presents the interaction flow on the main page, from when the user clicks “Add to Cart” to the cart update and customer notification.

Diagrama

El contenido generado por IA puede ser incorrecto.

Illustration 5. Main page  
Source: Own elaboration

III. Frontend – Cart Page (cart-client.js)

The *cart-client.js* file manages the logic of the cart page. It handles the visual representation of the cart (*updateCartUI()*), controls button events (+, -, delete), and manages the payment process via the */iniciar-pago* and */finalizar-pago* endpoints.

Figure 3 shows the cart page flow, highlighting the interface update cycle, data submission to the backend, and the wait for payment confirmation through polling.

Diagrama

El contenido generado por IA puede ser incorrecto.

Illustration 6. Cart page  
Source: Own elaboration

IV. Administrator Interface – Transaction Management

The administrator interface centralizes transaction management. From this panel, the administrator can list, filter, and search for transactions, as well as execute payments to providers. Once a transaction is executed, the money is sent to the provider’s wallet, and the status changes to “Completed.”

Figure 5 presents the administrator interface flow, from viewing transactions to the successful execution of an operation and the update of the status in the table.

Diagrama

El contenido generado por IA puede ser incorrecto.Illustration 7. Admin interface flow  
Source: Own elaboration

**7.2 User Manuals**

1. User Manuals - Client

This manual describes the process that an end-user (buyer) must follow to make a purchase within the platform.

Steps to use:

1. Browse product catalog
   * Upon entering the system, the client can view the complete catalog of available handcrafted products.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

1. Select product and add to cart
   * The client chooses the desired product and presses the “Add to Cart” button.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

* + One or more items can be added before proceeding to payment.

1. Access the shopping cart
   * By clicking on the cart icon in the upper right corner, the client accesses the list of selected products.

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

1. Confirm products and quantity
   * The system displays the chosen items along with the quantity.
   * The client can modify the number of units or remove products.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Proceed to payment
   * Click the “Pay Now” button to start the purchase process.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

1. Payment confirmation via Interledger
   * A confirmation window for the Interledger Protocol (ILP) appears.
   * The client must select “Accept” to validate the transaction or “Decline” to cancel the purchase.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

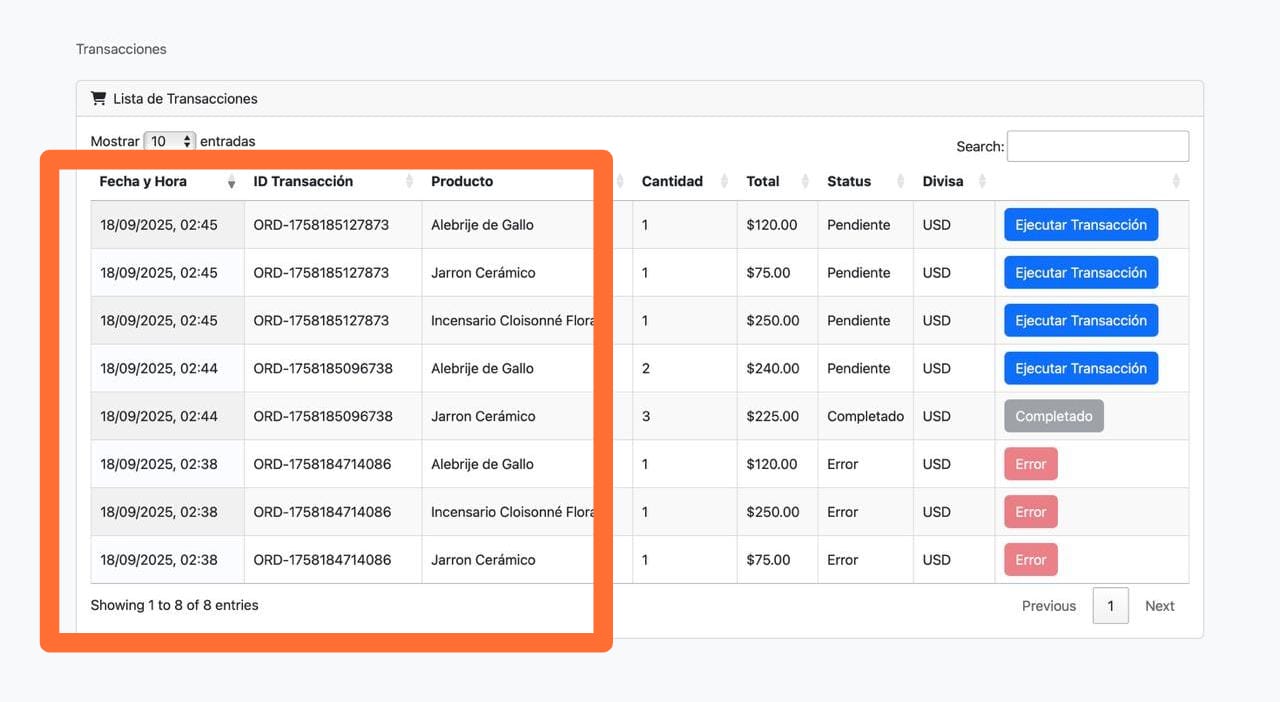
1. Payment notification
   * The system notifies the client that the payment has been successfully completed.

2. User Manuals **- Administrator**

This manual describes the functions of the administrator within the system, responsible for validating payments and managing settlement to artisans.

Steps to use:

1. Receipt of total payments
   * The administrator views the received payments, which correspond to the sum of products purchased by the client.
   * The system shows the product–artisan relationship, indicating which seller should receive each portion of the payment.



1. Validation of product delivery
   * Once confirmed that the client has received the order, the administrator authorizes the execution of the transaction toward the sellers.

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

* + This validation ensures transparency in the operation.

1. Transaction confirmation to artisans
   * The administrator selects the “Accept” option.
   * The system automatically settles the corresponding amount into the seller’s wallet, deducting only the established commission (5%).

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

1. Final notification
   * The artisan receives real-time confirmation that the funds are already available in their wallet.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

7.3 Data Dictionary

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| Transaccion | varchar(50) | Primary key. Identifier of the transaction made by the client to the administrator’s wallet. |
| Producto | varchar(50) | Product name. |
| Cantidad | int(100) | Quantity of the product. |
| Total | decimal(10,2) | Total amount to be transferred to the artisan’s wallet. |
| Wallet\_address\_artesano | varchar(100) | ID or wallet address of the artisan. |
| Status | varchar(50) | Status of the transfer to the artisan’s wallet. |
| Divisa | varchar(50) | Currency in which the client made the transaction. |

**8. Conclusions**  
General Conclusions

* The developed system demonstrates that it is possible to create an inclusive, scalable, and secure platform that enables artisans to participate in global e-commerce with lower entry barriers.
* The integration of ILP (Interledger Protocol) and Open Payments provides a clear competitive advantage over traditional platforms by reducing transaction costs and ensuring real-time interoperable payments.
* A balance was achieved between technical, economic, and social feasibility, confirming that the solution is not only profitable but also contributes cultural value and financial inclusion.
* The functional prototype validated usability and user experience through a simple interface, confirming that buyers can complete a transaction in just a few clicks and artisans receive their payments transparently.

Specific Conclusions

* The 5% commission model proved to be fairer than that of global commercial platforms, creating incentives for both artisans and buyers.
* Using a single order table simplified the initial development and consolidated key transaction data, facilitating auditing and future scalability.
* The user manual with screenshots adds value by making system adoption easier for both clients and administrators.
* The validation flow through the Administrator ensures trust in the purchasing and payment process, strengthening system transparency.
* Economic feasibility was verified: with a base of 10,000 annual buyers, the system can generate sustainable revenue from the first year.
* At the cultural and social level, the platform becomes a channel for global visibility of craftsmanship, contributing to the preservation and promotion of local traditions.