Databases II
Semester 2025-III
Workshop No. 1 — Project Definition and Database Modeling
Personalized E-Commerce Platform

Eng. Jorge Andrés Quiceno S.[1], Eng. Laurent David Chaverra Córdoba[2]
and Eng. Juan David Olmos Corredor[3]

October 18, 2025

Contents

1 Introduction

The ArtisanNexus project proposes the design and development of a high-performance, resilient *Multi-Vendor Marketplace Platform*. The strategic vision is to establish ArtisanNexus as the authoritative digital nexus for unique, customized, and artisanal goods, enabling seamless global commerce between independent creators (Vendors) and discerning consumers (Customers).

The platform manages the entire e-commerce process, from importing and standardizing vendor product data to processing high-volume payments and generating actionable business intelligence. The objective of this architecture is to eliminate the bottlenecks typically associated with monolithic e-commerce architectures by adopting a distributed, data-centric framework.

The core problem addressed by ArtisanNexus lies in the systemic constraints imposed by the multi-vendor paradigm, to handle two primary challenges simultaneously:

1. **Data Heterogeneity and Complexity:** Unlike single-brand e-commerce, the product data ingested from multiple independent vendors lacks a uniform schema. Efficiently storing, retrieving, and searching this dynamic, unstructured data demands flexibility that exceeds the capabilities of a pure Relational Database Management System (RDBMS).
2. **Distributed Transactional Integrity:** The capability for a Customer to execute an atomic checkout involving items from multiple distinct Vendors introduces significant *concurrency control* challenges. Ensuring that inventory reservation, payment authorization, and order finalization succeed or fail as a singular, consistent unit—thereby upholding *ACID* (Atomicity, Consistency, Isolation, Durability)—is critical to preventing organizational revenue leakage and preserving customer trust.

To overcome these constraints, the project is structured around three primary strategic goals:

- **Operational Excellence:** To achieve and maintain a high-availability (HA) operational posture with minimal latency for high-volume, customer-facing interactions.
- **Knowledge Maximization:** To transform raw user activity (clickstreams, purchase history) and sales data into actionable business intelligence (BI) for both platform administrators and Vendors.
- **System Resilience and Scalability:** To engineer a system capable of scaling elastically to accommodate exponential growth in both user count and transactional throughput via database partitioning and distributed architectures.

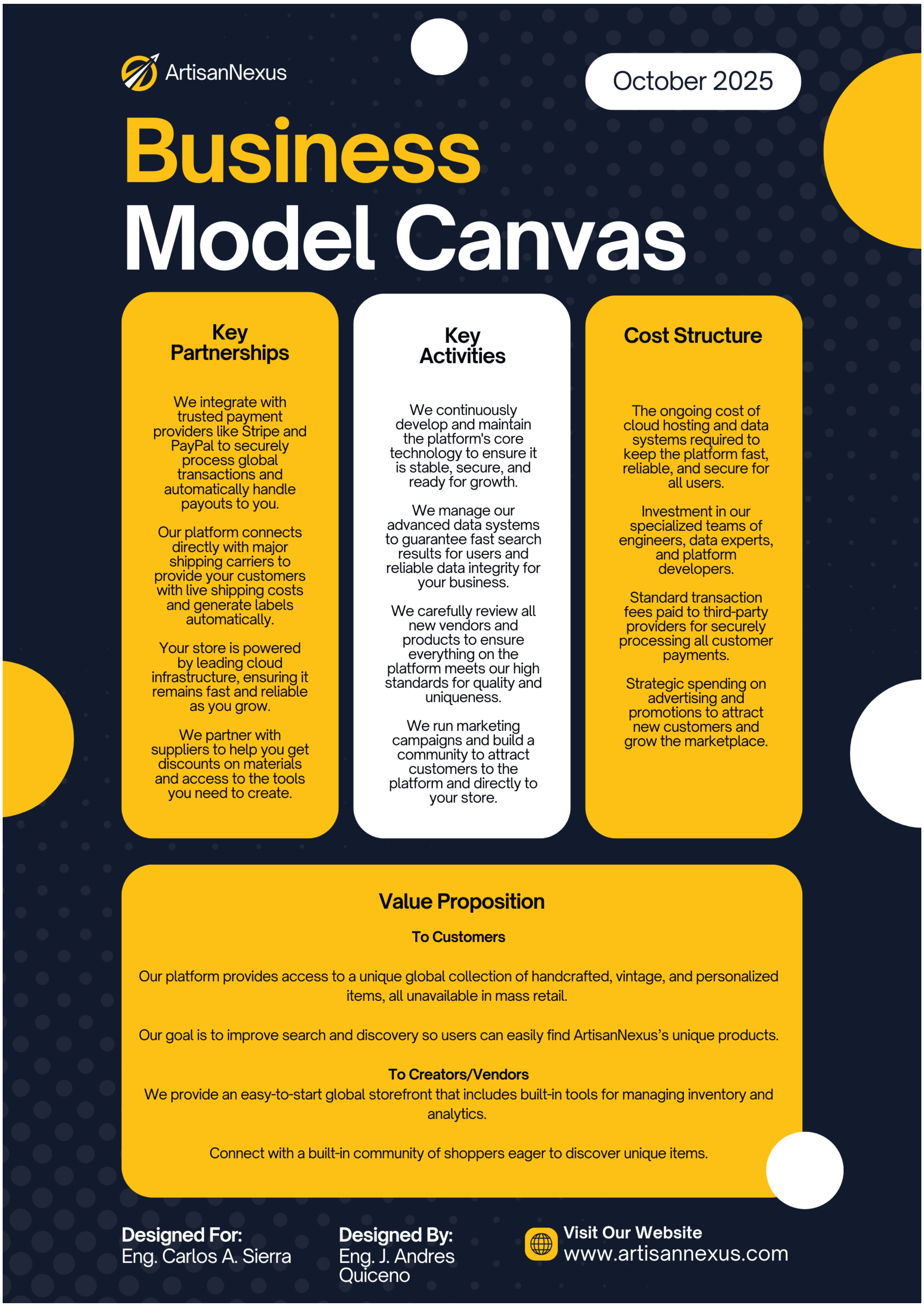
To meet the strategic goals, the system design will leverage a *Microservices Architecture* built on the robust *Java/Spring Boot* framework for the backend, supported by an *Angular* front-end. The core technological objective is the implementation of a *Polyglot Persistence* strategy, which dictates selecting the optimal database technology for each specific service's workload.

This approach directly satisfies the rigorous requirements for modern data-intensive applications:

- **Fast Query Execution** is achieved by dedicating an *Inverted Index NoSQL* solution for search operations, decoupling it from the transactional core.
- **Constant Data Ingestion** and inter-service communication are managed by a *Distributed Streaming Platform*, which processes event data in real-time.

2 Business Model Canvas

The Business Model Canvas describes the value proposition, customer segments, channels, revenue streams, and other key aspects of the application.



ArtisanNexus

October 2025

Business Model Canvas

Key Partnerships

We integrate with trusted payment providers like Stripe and PayPal to securely process global transactions and automatically handle payouts to you.

Our platform connects directly with major shipping carriers to provide your customers with live shipping costs and generate labels automatically.

Your store is powered by leading cloud infrastructure, ensuring it remains fast and reliable as you grow.

We partner with suppliers to help you get discounts on materials and access to the tools you need to create.

Key Activities

We continuously develop and maintain the platform's core technology to ensure it is stable, secure, and ready for growth.

We manage our advanced data systems to guarantee fast search results for users and reliable data integrity for your business.

We carefully review all new vendors and products to ensure everything on the platform meets our high standards for quality and uniqueness.

We run marketing campaigns and build a community to attract customers to the platform and directly to your store.

Cost Structure

The ongoing cost of cloud hosting and data systems required to keep the platform fast, reliable, and secure for all users.

Investment in our specialized teams of engineers, data experts, and platform developers.

Standard transaction fees paid to third-party providers for securely processing all customer payments.

Strategic spending on advertising and promotions to attract new customers and grow the marketplace.

Value Proposition

To Customers

Our platform provides access to a unique global collection of handcrafted, vintage, and personalized items, all unavailable in mass retail.

Our goal is to improve search and discovery so users can easily find ArtisanNexus's unique products.

To Creators/Vendors

We provide an easy-to-start global storefront that includes built-in tools for managing inventory and analytics.

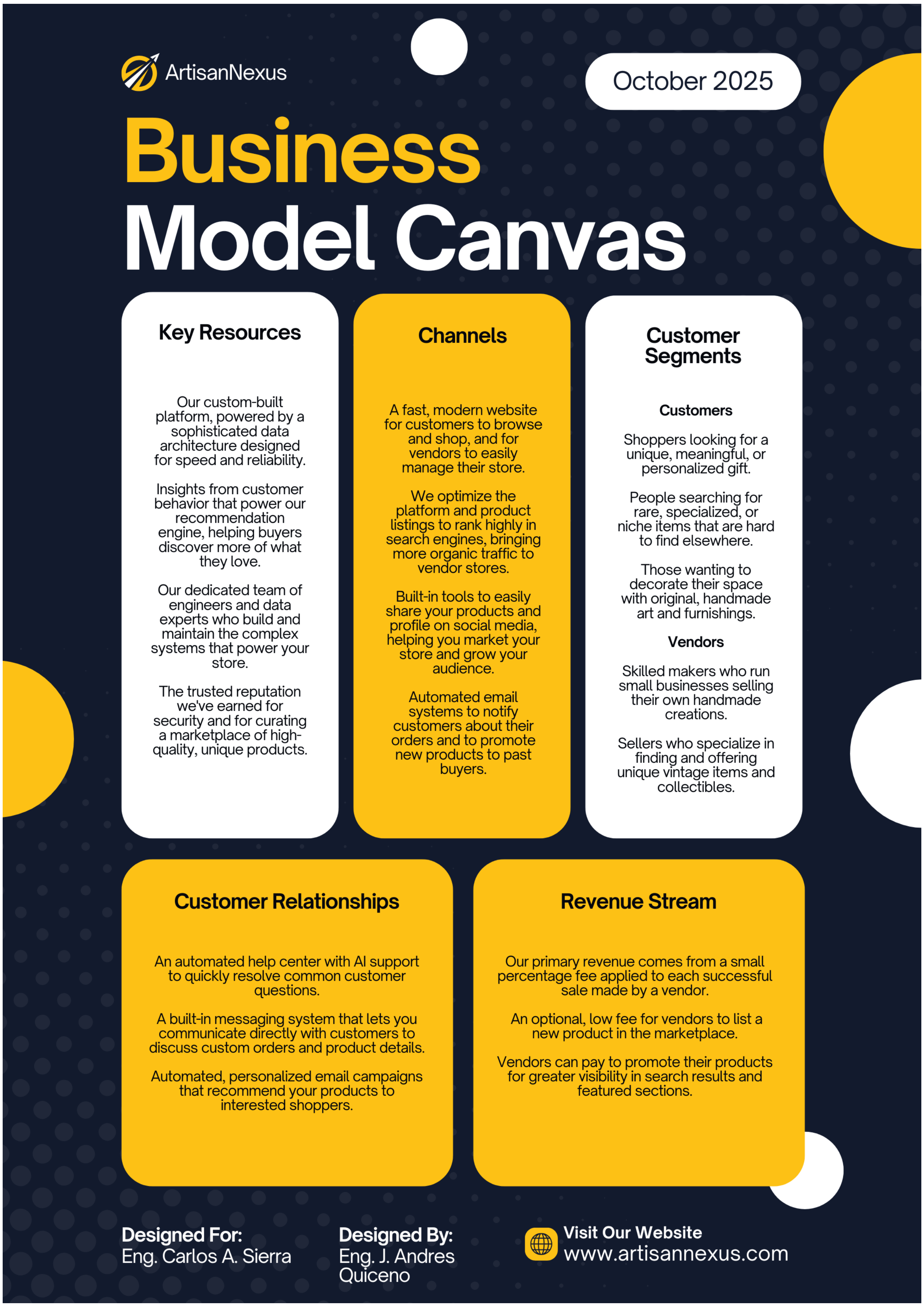
Connect with a built-in community of shoppers eager to discover unique items.

Designed For:
Eng. Carlos A. Sierra

Designed By:
Eng. J. Andres Quiceno

Visit Our Website
www.artisannexus.com

Figure 1: Business Model Canvas for the E-Commerce Platform



ArtisanNexus

October 2025

Business Model Canvas

Key Resources

Our custom-built platform, powered by a sophisticated data architecture designed for speed and reliability.

Insights from customer behavior that power our recommendation engine, helping buyers discover more of what they love.

Our dedicated team of engineers and data experts who build and maintain the complex systems that power your store.

The trusted reputation we've earned for security and for curating a marketplace of high-quality, unique products.

Channels

A fast, modern website for customers to browse and shop, and for vendors to easily manage their store.

We optimize the platform and product listings to rank highly in search engines, bringing more organic traffic to vendor stores.

Built-in tools to easily share your products and profile on social media, helping you market your store and grow your audience.

Automated email systems to notify customers about their orders and to promote new products to past buyers.

Customer Segments

Customers

Shoppers looking for a unique, meaningful, or personalized gift.

People searching for rare, specialized, or niche items that are hard to find elsewhere.

Those wanting to decorate their space with original, handmade art and furnishings.

Vendors

Skilled makers who run small businesses selling their own handmade creations.

Sellers who specialize in finding and offering unique vintage items and collectibles.

Customer Relationships

An automated help center with AI support to quickly resolve common customer questions.

A built-in messaging system that lets you communicate directly with customers to discuss custom orders and product details.

Automated, personalized email campaigns that recommend your products to interested shoppers.

Revenue Stream

Our primary revenue comes from a small percentage fee applied to each successful sale made by a vendor.

An optional, low fee for vendors to list a new product in the marketplace.

Vendors can pay to promote their products for greater visibility in search results and featured sections.

Designed For:
Eng. Carlos A. Sierra

Designed By:
Eng. J. Andres Quiceno

Visit Our Website
www.artisannexus.com

Figure 2: Business Model Canvas for the E-Commerce Platform

3 Requirements

This section specifies the comprehensive requirements for the *ArtisanNexus* Multi-Vendor Marketplace, organized by functional domain and quality attributes, ensuring alignment with advanced database principles (ACID, Polyglot Persistence, Distributed Systems).

3.1 Functional Requirements (FR)

Functional requirements detail the specific behaviors and services the system must provide to its stakeholders (Customer, Vendor, Admin). Each requirement is designed to be independent and clarify system accountability.

- **FR 1.1: Secure User Authentication and Authorization.**
  Requirement: The system shall allow secure registration, login, and identity verification, enforcing role-based access for Customer, Vendor, and Admin roles.
  Acceptance Criteria: The system must successfully authenticate multiple concurrent login requests with a success rate of 98% and enforce role separation such that a Customer cannot access the Vendor dashboard API endpoints.
  Role Association: All Users.

- **FR 1.2: User Profile and Preference Management.**
  Requirement: The system shall allow users to manage their personal profiles, multiple shipping addresses, preferred payment references, and personalization settings.
  Acceptance Criteria: Changes to a user's default shipping address must be reflected in the *Order Service* and available during checkout within 4 seconds of the update being committed.
  Role Association: Customer, Vendor.

- **FR 2.1: Product Catalog Management (Catalog Service).**
  Requirement: The system shall allow a Vendor to create, update, and manage core product listings and dynamic, vendor-defined attributes (e.g., size charts, material composition) within the Catalog Service.
  Acceptance Criteria: A new product listing, including 5 custom attributes, must be successfully persisted in the **MongoDB** catalog and display correctly on the product page.
  Role Association: Vendor.

- **FR 2.2: Real-Time Inventory Management and Concurrency Control.**
  Requirement: The system shall manage product inventory counts in a high-concurrency environment and utilize database mechanisms to prevent overselling of limited stock.
  Acceptance Criteria: Inventory levels, persisted in the *PostgreSQL* Order Service, must adjust immediately when a purchase occurs, preventing any concurrent transaction from confirming a sale that would result in a negative stock count.
  Role Association: Vendor.

- **FR 3.1: High-Speed Faceted Search and Filtering.**
  Requirement: The system shall provide an advanced, high-speed **faceted search capability** across the entire product catalog, supporting full-text queries and filtering by category, price, vendor rating, and dynamic product attributes.
  Rationale & Acceptance Criteria: To ensure a seamless user experience and prevent bounce rates, search results for a complex query (keywords + 3 filters) must be returned from the **Elasticsearch** index with a P99 latency of **under 800 milliseconds**. This

necessitates a specialized NoSQL index store to achieve performance targets for our projected load of 150 RPS.
Role Association: Customer.

- **FR 3.2: Atomic Multi-Vendor Transaction Execution.**
  Requirement: The system shall enable a Customer to perform a checkout for a cart containing items from multiple distinct vendors in a single transaction request.
  Acceptance Criteria: If inventory reservation fails for *any single item* in a multi-vendor cart, the system must trigger an immediate and complete rollback across all associated database updates (inventory, payment holding), and the customer must receive a single, consolidated error message.
  Role Association: Customer.

- **FR 4.1: Order, Cart, and Checkout Lifecycle Management.**
  Requirement: The system shall accurately manage the full lifecycle of a customer order, including cart persistence, checkout finalization, and displaying complete order history.
  Acceptance Criteria: All completed orders must be consistently recorded in the *PostgreSQL Order database* and visible in the Customer's order history within 6 seconds of payment confirmation.
  Role Association: Customer.

- **FR 4.2: Returns, Refunds, and Order History Management.**
  Requirement: The system shall handle customer-initiated return and refund requests, accurately updating the order history, inventory counts, and financial records.
  Acceptance Criteria: A successful return action must immediately adjust the vendor's available inventory count and initiate the financial refund workflow, with the status reflected in the user's order history within 5 seconds.
  Role Association: Customer, Vendor.

- **FR 5.1: Personalized Recommendation Generation.**
  Requirement: The system shall provide tailored product suggestions based on captured user activity and purchase history, supporting collaborative filtering logic.
  Acceptance Criteria: The homepage must display a "Recommended for You" section for a logged-in user with a **relevance rate of 70%** (7 out of 10 recommendations match the user's previously viewed categories) and must be generated within 1000 ms.
  Role Association: System (Internal) / Customer.

- **FR 5.2: A/B Testing Framework for Personalization Effectiveness.**
  Requirement: The system shall support a framework to run simultaneous A/B experiments to test the effectiveness of different recommendation algorithms or personalization strategies.
  Acceptance Criteria: Experiment assignment data must be captured alongside all user events, allowing the Administrator to accurately segment and analyze the conversion performance of each test group.
  Role Association: System (Internal) / Admin.

- **FR 6.1: Granular Event Collection (Data Ingestion).**
  Requirement: The system shall securely and reliably capture and persist all raw user interactions (clickstreams, search terms, views), orders, and core transactions as events.

Rationale & Acceptance Criteria: To feed the recommendation engine and analytics, every product page view event must be successfully published to the *Apache Kafka* data stream, maintaining a consumer lag of under 30 seconds under a projected peak load of **500 events per second**.
Role Association: System (Internal).

- **FR 7.1: Real-Time Vendor Operational Dashboard.**
  Requirement: The system shall provide the Vendor with a real-time dashboard displaying key operational metrics, including inventory alerts and current month's sales transactions.
  Rationale & Acceptance Criteria: To provide timely operational awareness without the cost of true real-time processing, the Vendor Dashboard must display sales metrics derived from the transactional database that are no more than **15 seconds** old.
  Role Association: Vendor.

- **FR 7.2: Analytical Reporting for Business Intelligence (BI).**
  Requirement: The system shall enable an Administrator to generate comprehensive sales and performance reports, aggregated by product category, vendor region, and time period, based on historical data.
  Acceptance Criteria: The analytical query for the monthly sales report must execute against the isolated **BI Data Warehouse** and return a complete result set to the Admin within **20 seconds**.
  Role Association: Administrator.

- **FR 8.0: Data Governance and Lineage Tracking.**
  Requirement: The system shall enforce data governance protocols, including schema consistency across services and the ability to trace data lineage for all critical datasets.
  Acceptance Criteria: The Administrator must be able to verify the source and transformation path of any data field presented in the BI Dashboard (FR 7.1) back to its originating system (PostgreSQL or Kafka event).
  Role Association: System (Internal) / Admin.

3.2 System Analysis

The non-functional requirements (NFRs) for ArtisanNexus are derived from a projected workload model based on a micro-enterprise growth trajectory. This analysis justifies the selection of key metrics for latency, availability, and scalability, framing the entire architectural design.

3.2.1 Summary of Key Metrics

Table 1: Summary of Key Performance and Operational Metrics

Metric	**What it Measures**	**Why it's Important**	**Example from Document**
P99/P50 Latency	Speed of individual requests	User experience & consistency	Search P99 $\leq$ 800ms
RPS (Requests Per Second)	Backend load capacity	Scalability of microservices	150 RPS at peak
Concurrent Users	Simultaneous active load	Database & service stress test	50 concurrent users
EPS (Events Per Second)	Data ingestion volume	Scalability of the event pipeline	500 EPS
Availability Percentage	System uptime	Reliability & user trust	99.5% available
RTO (Recovery Time Objective)	Disaster recovery speed	Business continuity	RTO $<$ 2 hours
Consumer Lag	Data processing delay	"Real-timeness" of insights	Lag $<$ 30 seconds
Data Freshness	Timeliness of displayed data	Decision-making quality	Data $\leq$ 15 sec old
Relevance Rate	Recommendation quality	User engagement & sales	70% relevance
Success Rate	Operation reliability	System stability	98% login success rate

3.2.2 Expected Load and Workload Profile

We estimate the initial operational parameters and project the target metrics based on a realistic two-year growth model for a micro-enterprise:

- **Target Active User Base (Year 2):** 5,000 registered users.
- **Peak Concurrent Users (PCU):** Estimated at 1% of the active base, leading to a PCU of **50 concurrent users** (users actively performing transactions/searches).
- **Peak Request Volume (API):** During peak hours (e.g., promotional events), the system must handle up to **150 requests per second** (RPS), with a split of approximately 80% read operations (search, catalog views) and 20% write operations (cart updates, order placement).
- **Data Ingestion Rate:** Approximately **500 events per second** (EPS) from clickstream data during peak usage (e.g., product views, searches, scroll activity).

3.3 Non-Functional Requirements (NFR)

Non-functional requirements specify quality attributes essential for the system's operational viability, scalability, and compliance, with a focus on distributed system characteristics.

- **NFR 1.0: High Availability and Fault Tolerance.**
  Requirement: The core transactional services must be highly available and resilient to single-node failures using database replication and automated failover.
  Rationale & Acceptance Criteria: To balance user trust with operational cost for a micro-enterprise, the **Order Service (PostgreSQL)** must maintain an overall service availability of **99.5%** (approx. 44 hours of downtime/year), and failover from a Primary database node to a Replica must complete within 5 minutes.

- **NFR 2.0: Horizontal Scalability and Partitioning.**
  Requirement: The architecture must support scaling horizontally to accommodate growth in user base, products, and transactional volume.
  Rationale & Acceptance Criteria: To ensure the platform can handle unexpected growth or flash sales without a costly re-architecture, the system must demonstrate the ability to process **50** concurrent order submissions without service interruption. This validates the initial design using **database sharding** (e.g., by Vendor ID) for high-volume tables, a core principle of our polyglot persistence strategy.

- **NFR 3.0: Data Consistency and Isolation (Concurrency Control).**
  Requirement: Strict transactional correctness must be maintained in the financial and inventory core to prevent anomalies like Lost Updates and Dirty Reads.
  Acceptance Criteria: The transactional core (PostgreSQL) must enforce at least the *Repeatable Read* isolation level for all critical operations (e.g., inventory debit), ensuring that concurrent transactions from our projected 50 peak concurrent users do not compromise data integrity.

- **NFR 4.0: Multi-Location Data Storage and Disaster Recovery.**
  Requirement: Key data must be replicated to a secondary region to ensure business continuity in case of a regional outage.
  Rationale & Acceptance Criteria: For a micro-enterprise with a concentrated user base, active-active geo-replication is cost-prohibitive. The requirement is scaled to disaster recovery. The **MongoDB Product Catalog** and **PostgreSQL** must have asynchronous replicas in a separate geographic region, with a Recovery Time Objective (RTO) of **under 2 hours**.

- **NFR 5.0: Cost Efficiency and Resource Management.**
  Requirement: Data and compute resources must be managed efficiently, utilizing tiered storage strategies based on data access frequency.
  Acceptance Criteria: Historical transaction data older than 180 days must be migrated from the high-cost OLTP (PostgreSQL) cluster to a lower-cost analytical store (OLAP/-Data Warehouse) to manage costs effectively at our projected scale.

- **NFR 6.0: Security, Encryption, and Compliance.**
  Requirement: All sensitive data, including customer PII and payment references, must be encrypted both in transit and at rest.
  Acceptance Criteria: All API traffic between the Angular frontend and the Spring Boot backend must enforce TLS 1.2 or higher, and the PII fields in the database must be encrypted using AES-256.

- **NFR 7.0: Maintainability and Observability.**

Requirement: The system must provide comprehensive logging, metrics tracking, and alerting to enable clear diagnosis and support for operations teams.
Acceptance Criteria: System operators must receive an automated alert within 10 minutes of any critical service dependency (PostgreSQL, Kafka, Elasticsearch) exceeding 80% of its allocated CPU or memory resources.

4 Improved User Stories

The user stories describes the main use cases from the user's perspective, including the related actions, the necessary conditions to do these actions, and the expected results of the actions.

To improve the latest release, we created a user story for each action, replacing the previous generic user stories that grouped several related actions (such as the one describing complete product management, which is a complete CRUD). In addition, in the acceptance criteria, we sought to detail the consequences of requests, including valid and invalid cases (e.g., successful login vs. failed login), more accurately representing the flow of actions from the user's perspective.

Title: **Register**	Priority: **High**	Estimation: **5 h**
User Story: As a end-user or admin, I want to register in the system, so that I can authenticate to access the app resources and do various operations, according to my permissions level.		
Acceptance Criteria: • Given I am on the registration page, when I enter a unique email, a strong password, my full name, select a role (End-User or Admin), and submit the form, then my account is created, a success message is shown, and I am redirected to the login page. • Given I am on the registration page, when I enter an email that is already registered in the system and submit the form, then the account is not created, and I see an error message: "This email is already registered". • Given I am on the registration page, when I enter a password that does not meet the strong password policy (less than 8 characters, no numbers/special characters) and submit the form, then the account is not created, and I see an error message explaining the password requirements. • Given I am on the registration page, when I leave a mandatory field (email, password, name, surname) empty and submit the form, then the account is not created, and I see error messages indicating the mandatory fields. • Given I am on the registration page, when I enter an email with an invalid format and submit the form, then the account is not created, and I see an error message: "Please enter a valid email address". • Given I am on the registration page, when I successfully submit the form, then the system assigns the correct permissions and default access level based on the selected role (End-User or Admin).		

Table 2: End User and Admin user story for register

Title: **Authentication**	Priority: **High**	Estimation: **5 h**
User Story: As a end-user or admin, I want to authenticate before the system, so that I can access the app resources and do various operations, according to my permissions level.		
Acceptance Criteria: • Given I am on the login page, when I attempt to log in with a valid username/email and password combination, then the app must redirect me to the appropriate landing page, depending on my user role. • Given I am on the login page, when I attempt to log in with an invalid username/email and password combination, then the app must show me an error message indicating the username/email don't exists or the password is incorrect. • Given I am on the login page, when I fail to log in three times with the same username/email, then the system must email me asking if it is really me, or if I want to block the authentication to prevent someone stole my account. • Given I am on the login page, when I click the "I forgot my password" link, then the system must email me indicating the steps to start the reset password process.		

Table 3: End User and Admin user story for authentication

Title: **Create Product**	Priority: **High**	Estimation: **3 h**
User Story: As a supplier, I want to create new products with details like name, description, price, and image so that I can add them to my catalog for customers to see and purchase.		
Acceptance Criteria: • Given I am on the "Add New Product" page, when I enter valid data for all mandatory fields (name, description, price, stock) and submit the form, then the product is saved, I see a success message, and I am redirected to the product list or catalog view. • Given I am on the "Add New Product" page, when I try to submit the form without filling in a mandatory field, then the product is not created, and I see an error message highlighting the missing mandatory fields. • Given I am on the "Add New Product" page, when I enter an invalid price or invalid stock, then the product is not created, and I see an appropriate error message. • Given I am on the "Add New Product" page, when I successfully create a product, then the new product appears in my product catalog/list and is marked as available for end-users.		

Table 4: Supplier user story for creating a product

Title: **Edit Product**	Priority: **High**	Estimation: **3 h**
User Story: As a supplier, I want to edit the details of my existing products so that I can correct mistakes, update prices, or change stock levels.		
Acceptance Criteria: • Given I am viewing my product list, when I select a product to edit, then I am taken to an edit form with the product's current details pre-filled. • Given I am editing a product, when I change some details (like description or price) with valid data and save the form, then the product is updated, I see a success message, and the changes are immediately reflected in the product catalog. • Given I am editing a product, when I try to save the form with invalid data (like an empty name or a negative price), then the product is not updated, and I see an error message explaining the validation failure. • Given I am editing a product, when I change the product's status to "Inactive" or "Out of Stock" and save, then the product is no longer visible or purchasable by end-users in the storefront.		

Table 5: Supplier user story for editing a product

Title: **Delete Product**	Priority: **Mid**	Estimation: **2 h**
User Story: As a supplier, I want to delete products that I no longer wish to offer so that they are permanently removed from the catalog.		
Acceptance Criteria: • Given I am viewing my product list, when I select the option to delete a specific product, then a confirmation dialog appears, asking me to confirm the deletion. • Given the deletion confirmation dialog is open, when I confirm that I want to delete the product, then the product is permanently removed from the system, and I see a success message. • Given the deletion confirmation dialog is open, when I cancel the action, then the dialog closes, the product is not deleted, and I am returned to the product list. • Given a product has been successfully deleted, when I view the product catalog or search for the deleted product, then the product is no longer present.		

Table 6: Supplier user story for deleting a product

Title: **Search and View Products**	Priority: **Mid**	Estimation: **2 h**
User Story: As a supplier, I want to search and filter through my list of products so that I can quickly find specific items I need to manage.		
Acceptance Criteria: • Given I am on my product management page, when I enter a search term (e.g., product name or SKU) into the search bar, then I see a list of products whose names or details match the search term. • Given I have performed a search, when the search returns no results, then I see a clear message: "No products found matching your search". • Given I am on my product management page, when I use a filter (by status "Active", "Inactive", or by category) then the product list updates to show only the products that match the selected filter. • Given I am viewing a list of products (either all, searched, or filtered), when I scroll or navigate through the list, then I can see key details for each product like its name, main image, price, and stock level at a glance.		

Table 7: Supplier user story for searching and viewing products

Title: **Custom Products Searching**	Priority: **High**	Estimation: **10 h**
User Story: As a end-user, I want to do search by various criteria, so that the app throw me custom results based in my app interactions and history.		
Acceptance Criteria: • Given I am on the products searching page, when I input a name in the search bar, then the app must show me all the products which name contains the one I submitted, prioritizing brands and categories that I previously review or buy. • Given I am on the products searching page, when I input a name in the search bar that no one product contains, then the app must show me clearly a message indicating no products where found. • Given I am on the products searching page, when I click the advanced search button, then the app must display inputs to filter the results by branch, category, minimum price, maximum price, or product specifications (depending on the selected category). • Given I am on the products searching page, when I input an advanced search, then the app must show me all the products which fulfill the specified filters. • Given I am on the products searching page, when I input an advanced search that no one product fulfill, then the app must show me clearly a message indicating no products where found.		

Table 8: User story for custom search

Title: **Profile Personalization**	Priority: **Mid**	Estimation: **2 h**
User Story: As a end user, I want to personalize my profile, so that I can obtain better recommendations.		
Acceptance Criteria: • Given I'm on my profile page, when I input changes on my personal data (name, location, etc), then the app must validate it. • Given I'm on my profile page, when I submitted valid changes on my personal data, then the app must save these changes, updating my profile. • Given I'm on my profile page, when I submitted valid changes on personal data that is being used to build custom recommendations, like my location, then the app must change its recommendations. • Given I'm on my profile page, when I submitted invalid changes on my personal data, like future birth dates or empty names/surnames, then the app must update the edit form, indicating the errors.		

Table 9: User story for profile customization

Title: **Supplier Personalization**	Priority: **Mid**	Estimation: **2 h**
User Story: As a supplier, I want to personalize my profile, so that I can show my business's changes.		
Acceptance Criteria: • Given I'm on my profile page, when I input changes on my business data (name, location, etc), then the app must validate it. • Given I'm on my profile page, when I submitted valid changes on my business data, then the app must save these changes, updating my profile. • Given I'm on my profile page, when I submitted invalid changes on my business data, like future birth dates or empty names/surnames, then the app must update the edit form, indicating the errors.		

Table 10: User story for profile customization

Title: **Manage Shopping Cart**	Priority: **High**	Estimation: **3 h**
User Story: As an end user, I want to add products to a shopping cart, view its contents, and remove items, so that I can review my selections before purchasing.		
Acceptance Criteria: • Given I'm viewing the products list, when I click the "Add to Cart" button for a product, then the product is added to my shopping cart. • Given I'm anywhere on the site, when I click the shopping cart icon, then the app must show me the shopping cart page, with the items currently added. • Given I'm viewing the shopping cart page, when I click an item's trash button, then the item is removed from the cart.		

Table 11: End user story for managing the shopping cart

Title: **Update Cart Quantities**	Priority: **High**	Estimation: **2 h**
User Story: As an end user, I want to adjust the quantities of items in my shopping cart, so that I can buy the exact amount I need.		
Acceptance Criteria: • Given I'm viewing the shopping cart page, when I click the subtract/add buttons for an item, then the item's quantity is updated, and the final price is recalculated accordingly. • Given I'm viewing the shopping cart page, when I click the add button, then the app must validate if the requested quantity is available in stock. • Given the requested quantity exceeds available stock, when I try to add more units, then the system must show an alert: "There are not enough items in stock. Only [X] available."		

Table 12: End user story for updating item quantities in the cart

Title: **Initiate Checkout**	Priority: **High**	Estimation: **2 h**
User Story: As an end user, I want to proceed from my shopping cart to the checkout, so that I can finalize my order and make a payment.		
Acceptance Criteria: • Given I'm on the shopping cart page, when I click the "Checkout" or "Finish Buy" button, then the app redirects me to the checkout/payments page. • Given I'm on the payments page, when I click the cancel button, then the app redirects me back to the shopping cart page.		

Table 13: End user story for initiating the checkout process

Title: **Process Payment and Create Order**	Priority: **High**	Estimation: **3 h**
User Story: As an end user, I want to securely pay for the items in my cart using a saved or new payment method, so that I can complete my purchase and acquire the products.		
Acceptance Criteria: • Given I'm on the payments page, when I click the confirm button, then the app must let me select one of my saved payment methods, or let me register a new one if none are available. • Given I have selected a payment method, when I confirm the order, then the system must validate if the payment method has the necessary funds. • Given the selected payment method has the necessary funds, when the payment is processed, then the order is confirmed, the products' inventory is updated, and I see an order confirmation screen. • Given the selected payment method has insufficient funds, when I try to confirm the order, then the transaction is canceled, and I see an alert: "The selected payment method does not have the necessary funds. Please select or register another method."		

Table 14: End user story for processing payment and creating an order

Title: **Returns and refunds**	Priority: **High**	Estimation: **6 h**
User Story: As an end user I want to request product refunds or changes, so that I can have my money back or a new product, in case the acquisition don't fulfill my expectations.		
Acceptance Criteria: • Given I'm on anywhere in the site, when I click the order history button, then the system redirect me to my order history page. • Given I'm on my order history page, when I select any finished order prior 30 days from today, then the system must show me the option to refund or return its products. • Given I'm on the order history page, when I request to refund or return any finished order prior 30 days from today, then the system must show me the order product list, allowing me to select the products I going to refund or return. • Given I'm on the refund form, when I click the "Finish" button, then the system must request the refund/request to each one supplier of the selected products.		

Table 15: User story for refunds and returns

Title: **Business Intelligence Dashboard**	Priority: **Mid**	Estimation: **32 h**
User Story: As an Admin, I want to access a central dashboard that displays key business metrics and user preferences, so that I can make data-driven decisions to facilitate business administration.		
Acceptance Criteria: • Given I am an authenticated admin user, when I navigate to the admin section, then I can access a "Dashboard" page that displays various data visualizations (e.g., charts, graphs, tables). • Given I am on the dashboard page, when the page loads, then I can see a section showing the top 5 most purchased and the top 5 least purchased products, along with their purchase counts. • Given I am on the dashboard page, when the page loads, then I can see a section showing sales statistics by product category, indicating which categories are the most and least profitable. • Given I am on the dashboard page, when the page loads, then I can see a chart displaying the hours of the day and days of the week with the highest and lowest user interaction (e.g., logins, page views, purchases). • Given I am on the dashboard page, when the page loads, then I can see a ranked list of suppliers based on their performance, showing the most successful (e.g., by total sales volume) and least successful suppliers. • Given any of the dashboard data sections, when I view the information, then the data presented must be pulled and updated in real-time (or near real-time) from the live database.		

Table 16: Admin user story for the business intelligence dashboard

5 Database Architecture

Presentation of the initial database architecture for the project

5.1 High-Level Architecture

(This section has been improved with respect to the last version in Workshop 1, specifying the sources, and the app structure.)

Our high level architecture is based on two main parts a Date Lake that is going to use a No SQL technology (yet to be defined) that recollect information from:

- User interactions: Simulating the data stream by extracting web information.
- User behaviors: Collect information from a variety of user actions.
- Partners: Supplier users representing a particular brand, who are in charge of providing data about their catalog.

On the other hand, a Date Warehouse divided in 3 parts for different functionalities:

- Products: segment designed to manage the showing, filtering, and searching of products.
- Sales: main segment for the sale logic.
- User patterns: segment designed to manage the recommendation system.

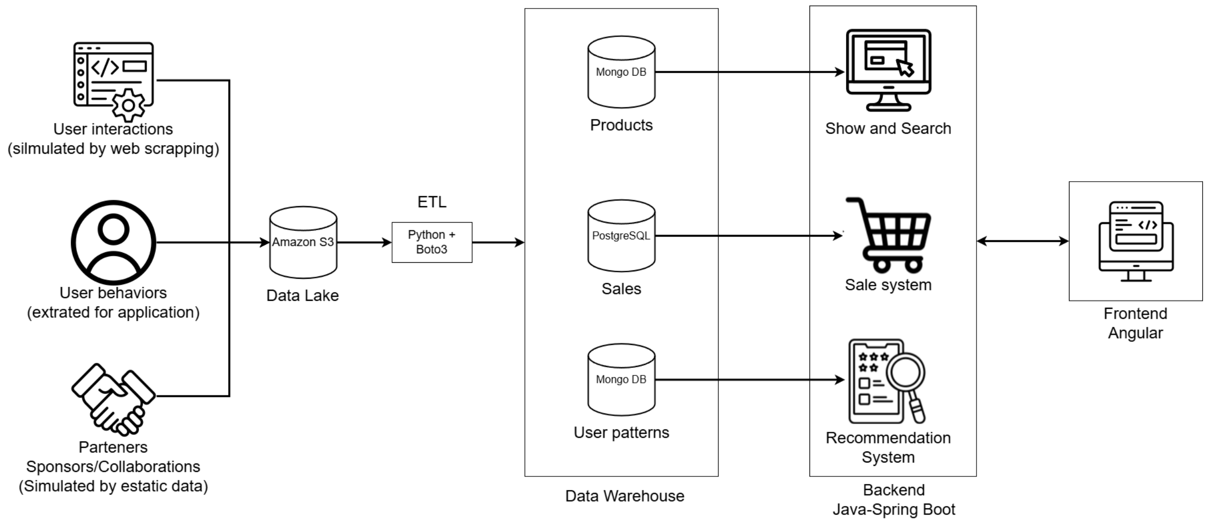


Figure 3: High Level DB Diagram

5.2 Data Flow

As shown in Figure 2, our flow data start with 3 main sources:

- User interactions: Simulating by web scraping, this raw data will be in HTML files.
- Behavioral patterns: Extracting for our application, these data will be given in JSON files.
- Partners: Working as a static base products, these data will be in CSV files.

After that, there is the ETL implemented with pandas and Boto3 in Python to process these raw data to put them in their corresponding data base:

- Products database: Feeding by sources: user interaction and partners.
- User Patterns Database: Feeding by the user behavior source.
- Sales: feeding by the internal application working.

5.3 Updated version of ER Diagram

The next diagram addresses the relational part of the database corresponding to the sale's section. The core of the sales database consists of three entities: user, product, and sale. The sale entity has a master/detail structure, where each detail refers to a product. Additionally, the user entity has a seller relationship with the offered products and a buyer relationship with completed sales.

This section has been improved with respect to the last version in Workshop 1, updating the diagram format, and describing better the main relationships.

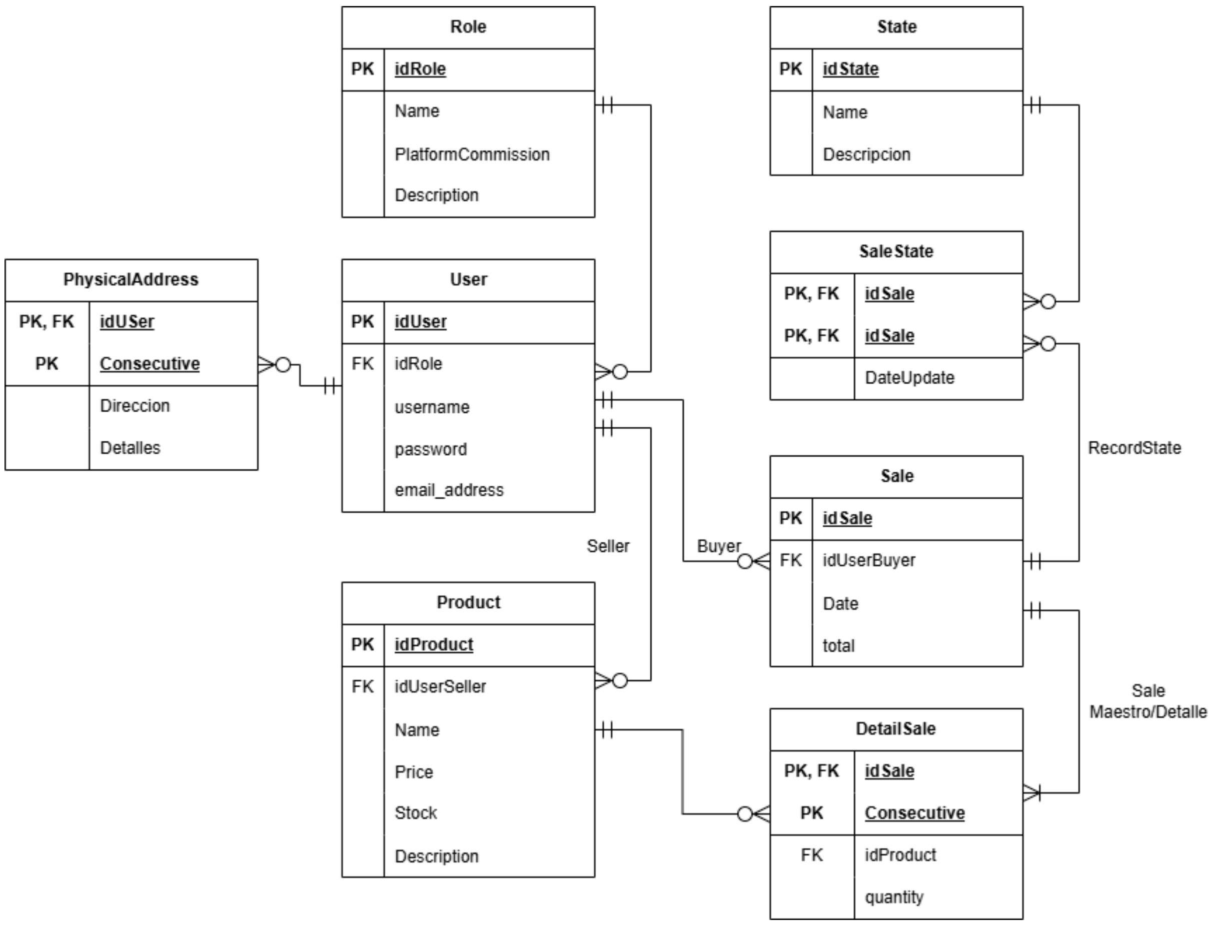


Figure 4: Entity-Relationship Diagram v.2.0

5.4 Storage Solutions

We propose a hybrid storage in order to reduce implementation's costs, potentially using a cloud storage solution for the data lake and an on-premise solution for a part of the data warehouse due to our familiarity with these alternatives.

5.5 Stack Technologies

- Data Lake in Amazon S3: Amazon S3 was chosen due to its high availability and durability. S3's scalability allows the system to handle large volumes of structured and unstructured data seamlessly. besides that it does this at a low price.
- ETL with boto3 in Python: Python was chosen for its simplicity, flexibility, and rich data processing ecosystem, including libraries such as Pandas, SQLAlchemy, and Requests. Python also provides fast prototyping and easy data manipulation, which makes it ideal for these tasks.
- Databases in PostgreSQL and Mongo DB:
  - PostgreSQL: proposed for transactional and operational data related to sales, orders, and system logic. It was chosen for its strong ACID compliance and advanced querying; PostgreSQL balances structure, flexibility, and power effectively.
  - MongoDB (NoSQL): used for product catalogs and user behavior patterns. It was

selected for its schema-less design, high performance in read-heavy operations, and easy integration.

- Backend in Spring Boot (Java): The backend is built with Spring Boot, a mature and enterprise-grade Java framework. It was selected for its reliability, scalability, and strong integration with relational and non-relational databases.
- Front end in Angular: We chose Angular for the front-end because it offers a robust and modular framework for building scalable web applications. Its Typescript foundation improves code maintainability and early error detection. Angular also works very well in conjunction with the backend technology chosen (Spring Boot).

6 Information Requirements

The information requirements describe the main types of information our system must retrieve, and the data sources where this information comes from, based on the functional requirements and user stories of the system. The platform's polyglot persistence strategy dictates the optimal data store for each retrieval pattern, balancing query flexibility with transactional integrity.

6.1 IR1 – User and Authentication Data

IR1.1 **Retrieve user profile and credentials from the User Service.**
Requirement: To authenticate users and populate profile information during login and session management.
Acceptance: The system successfully validates credentials and returns user roles (Customer, Vendor, Admin) and core profile data (e.g., name, email) without delay.
Data Source: PostgreSQL.

IR1.2 **Retrieve user permissions and role-based access controls from the User Service.**
Requirement: To authorize actions across the platform, ensuring users can only access resources and perform operations permitted by their role.
Acceptance: The UI dynamically adjusts based on the user's role, and API endpoints correctly deny unauthorized access attempts.
Data Source: PostgreSQL.

6.2 IR2 – Product Catalog and Search Data

IR2.1 **Retrieve structured product inventory from the Catalog Service.**
Requirement: To manage stock levels, vendor assignments, and availability for the transactional core.
Acceptance: The checkout process can reliably verify stock of items from multiple vendors.
Data Source: PostgreSQL.

IR2.2 **Retrieve structured product pricing from the Catalog Service.**
To calculate order's base pricing, discounts and taxing.
Acceptance: The checkout process can reliably calculate the total cost for an order, including items from multiple vendors.
Data Source: PostgreSQL.

IR2.3 **Retrieve unstructured product specifications and attributes from the Search Service.**
Requirement: To power the flexible product catalog where different categories (e.g., T-Shirts, Processors) have varying attributes (e.g., size, RAM).

Acceptance: The product detail pages correctly display all category-specific attributes, and the search engine can index and filter based on these dynamic fields.
Data Source: MongoDB.

IR2.4 **Retrieve products for user searching and browsing from the Search Service.**
Requirement: To provide fast, flexible, and faceted search results based on product names, descriptions, and dynamic attributes.
Acceptance: Users can perform full-text searches and apply filters by various product specs, receiving sub-second response times for their queries.
Data Source: MongoDB.

6.3 IR3 – Transaction and Order Data

IR3.1 **Retrieve order history and status from the Order Service.**
Requirement: To provide customers with their purchase history and vendors with their sales records.
Acceptance: Users can view a detailed history of their orders, including items, prices, dates, and current status (Processing, Shipped, and so on).
Data Source: PostgreSQL.

IR3.2 **Retrieve transaction and commission records from the Transaction Service.**
Requirement: Calculate commission per vendor and process payouts to vendors to facilitate financial reconciliation.
Acceptance: The admin dashboard can accurately report total revenue, commission earned, and individual vendor balances.
Data Source: PostgreSQL.

6.4 IR4 – Business Intelligence and Analytics Data

IR4.1 **Retrieve user interaction events from the Analytics Event Stream.**
Requirement: To capture semi-structured data on user behavior, such as page views, product clicks, and search queries, for analysis.
Acceptance: The Business Intelligence Dashboard can report on the most and least intense interaction hours and days, as well as popular products.
Data Source: MongoDB.

IR4.2 **Retrieve aggregated sales and performance metrics from the Data Warehouse.**
Requirement: To compile key business metrics for dashboard visualizations, such as top-selling products/categories and vendor performance rankings.
Acceptance: The admin dashboard displays real-time charts for top products, sales by category, and supplier rankings, pulling from aggregated data sources.
Data Source: PostgreSQL is necessary to obtain the sales data, MongoDB is necessary for obtain interaction data by products and products categories.

7 Query Proposals

Below is a query proposal for each information requirement, detailing how the information can be extracted.

7.1 IR1 – User and Authentication Data

Data provided: This query gets the basic information about a user and his role name as long as the credentials given (username and password) match the information stored.

```
SELECT U.username, U.emailAddress, R.name
FROM User U
    JOIN Role R ON U.idRole = R.idRole
WHERE U.username = <username> AND U.password = <password>;
```

7.2 IR2 – Product Catalog and Search Data

7.2.1 IR 2.1 – Retrieve structured product inventory from the Catalog Service

Data provided: This query gets the basic information about the products and their sellers.

```
SELECT P.name,
    P.description,
    P.price,
    P.stock,
    U.username AS seller
FROM Products P
    JOIN Users U ON P.idSeller = U.idUser;
```

7.2.2 IR 2.2 – Retrieve structured product pricing from the Catalog Service

Data provided: This query retrieve the products (and their prices) related with a sale, allowing to calculate the sale's total price.

```
SELECT P.name, P.price, DS.quantity
FROM Product P, DetailSale DS
WHERE DS.idProduct = P.idProduct
AND DS.idSale = <idSale>;
```

7.2.3 IR 2.3 – Retrieve unstructured product specifications and attributes from the Search Service

Data provided: This query retrieves all the relevant information for a product given.

```
db.products.find(
    { name: <Name>},
    {
        _id: 0,
        categories: 1,
        attributes: 1
    }
)
```

7.2.4 IR 2.4 – Retrieve products for user searching and browsing from the Search Service

Data provided: This query retrieves all the relevant information for a product given.

```
db.products.find(
    { name: <Name>},
    {
        _id: 0,
        categories: 1,
        attributes: 1,
    }
)
```

7.3 IR3 – Transaction and Order Data

7.3.1 IR 3.1 – Retrieve order history and status from the Order Service

Data provided: These queries retrieve all the necessary information to list a particular user's sales, including the status and products for each sale.

SQL Queries:

- Query to retrieve the sales from a user:

```
SELECT S.idSale
FROM Sale S
WHERE S.idBuyer = <idUser>;
```

- Query to retrieve the products of a sale:

```
SELECT P.name, P.price , DS.quantity
FROM Product P, DetailSale DS
WHERE DS.idProduct = P.idProduct
AND DS.idSale = <idSale>;
```

- Query to retrieve the state of a sale:

```
SELECT S.idSale , St.name as state , SST.dateUpdate
FROM Sale S, SaleStatus SS, State ST
WHERE S.idSale = SS.idSale
AND SS.idState = ST.idState
ORDER BY SST.dateUpdate desc
LIMIT 1;\\
```

7.3.2 IR 3.2 – Retrieve transaction and commission records from the Transaction Service

Data provided: This query retrieves the information necessary to calculate the amount sold and its percentage corresponding to the platform.

```
    SELECT U.username as seller,
        R.plataformCommission,
        P.name as product,
        DS.quantity,
        P.price
    FROM User U, Role R, Product P, DetailSale DS
    WHERE U.idUser = <idUser>
        AND DS.idProduct = P.idProduct
        AND P.idSeller = U.idUser
        AND U.idRole = R.idRole;
```

7.4 IR4 – Business Intelligence and Analytics Data

7.4.1 IR 4.1 – Retrieve user interaction events from the Analytics Event Stream

Data provided: This query retrieves the recent activity stream for a specific user, including pages visited, products viewed, and searches performed, for detailed behavioral analysis.

```
db.user_activity.find(
    { user_id: <idUser> },
    {
        _id: 0,
        timestamp: 1,
        activity_type: 1,
        page_url: 1,
        product_viewed: 1,
        search_query: 1
    }
).sort( { timestamp: -1 } ).limit(50)
```

7.4.2 IR 4.2 – Retrieve aggregated sales and performance metrics from the Data Warehouse

Data provided: These queries retrieve aggregated data to power the admin dashboard, showing top-selling products, sales by category, and best-performing suppliers.

SQL Queries:

- Query to retrieve the top 5 best-selling products:

```
    SELECT P.name, SUM(DS.quantity) as total_units_sold
    FROM Product P
        JOIN DetailSale DS ON P.idProduct = DS.idProduct
        JOIN Sale S ON DS.idSale = S.idSale
    WHERE S.date >= <start_date>
    GROUP BY P.idProduct, P.name
```

```
ORDER BY total_units_sold DESC
LIMIT 5;
```

- Query to retrieve sales statistics by product category:

```
SELECT C.name as category,
       COUNT(DISTINCT S.idSale) as number_of_orders,
       SUM(DS.quantity * P.price) as total_revenue
FROM Category C
    JOIN Product P ON C.idCategory = P.idCategory
    JOIN DetailSale DS ON P.idProduct = DS.idProduct
    JOIN Sale S ON DS.idSale = S.idSale
WHERE S.date >= <start_date>
GROUP BY C.idCategory, C.name
ORDER BY total_revenue DESC;
```

- Query to retrieve a ranked list of suppliers by sales performance:

```
SELECT U.username as supplier_name,
       COUNT(DISTINCT S.idSale) as total_orders,
       SUM(DS.quantity * P.price) as total_sales_volume
FROM User U
    JOIN Product P ON U.idUser = P.idSeller
    JOIN DetailSale DS ON P.idProduct = DS.idProduct
    JOIN Sale S ON DS.idSale = S.idSale
WHERE U.idRole = <vendor_role_id>
  AND S.date >= <start_date>
GROUP BY U.idUser, U.username
ORDER BY total_sales_volume DESC;
```

8 References

References

[1] Osterwalder, A. & Pigneur, Y. Business Model Generation. Wiley, 2010.

[2] Kimball, R. & Ross, M. The Data Warehouse Toolkit. Wiley, 2013.

[3] Ricci, F., Rokach, L., Shapira, B. Recommender Systems Handbook. Springer, 2015.