Workshop N° 3 - Data System Architecture and Information Retrieval

Universidad Distrital Francisco José de Caldas
School of Engineering
Computer Engineering Program

Andruew Steven Zabala Serrano - 20211020071

Ruben David Montoya Arredondo - 20211020055
Hemerson Julian Ballen Triana - 20211020084

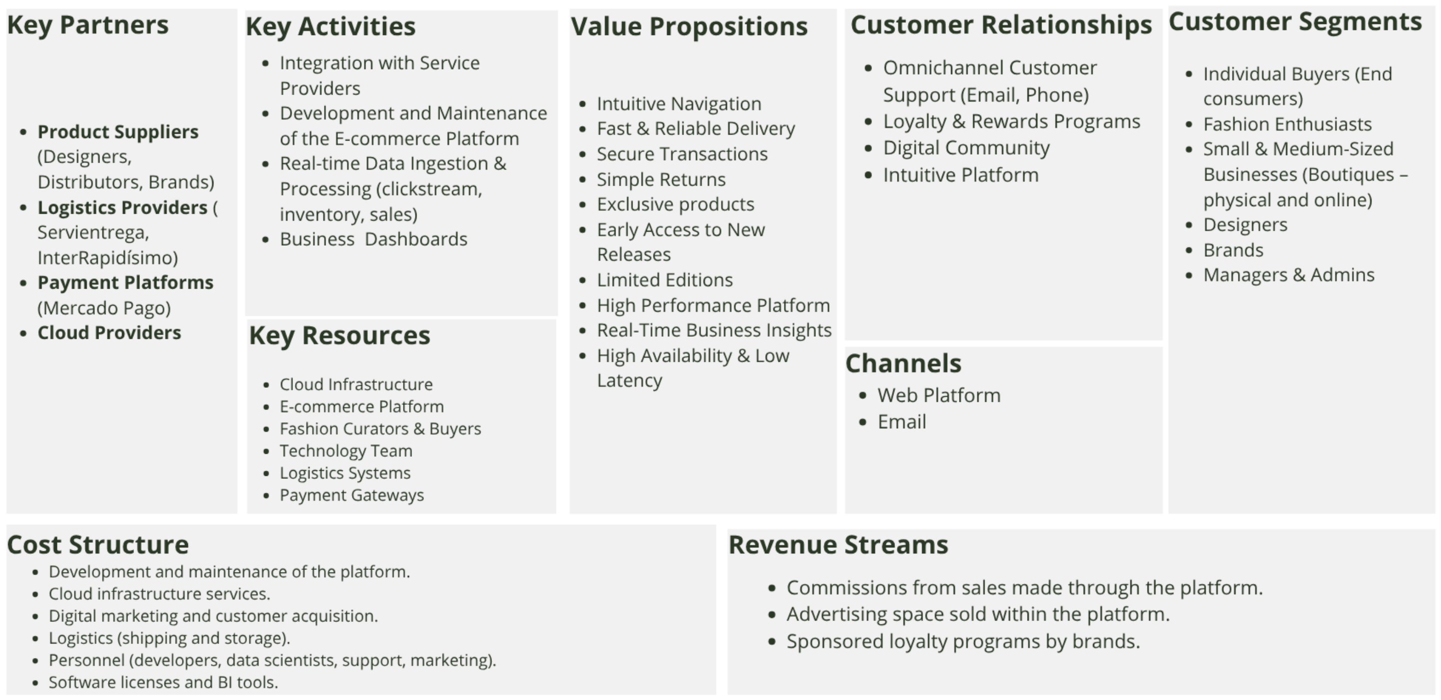
October 2025

1 Introduction

This project proposes the development of BogoGo, a fashion e-commerce platform designed to connect local clothing brands, independent designers, and customers within Bogotá. The platform seeks to promote local fashion culture while offering users a modern, personalized, and efficient online shopping experience. BogoGo provides exclusive access for customers and registered vendors, enabling secure transactions, real-time product management. The system supports multiple user roles: customers can explore collections, purchase items, track orders, and receive personalized suggestions; vendors can upload, update, and manage their products; and administrators oversee analytics, sales performance, and system integrity through a Business Intelligence (BI) dashboard. Core features include fast product search and filtering, localized delivery management within Bogotá, and real-time analytics that support informed decision-making for both vendors and administrators. Designed with scalability, usability, and security in mind, the platform seeks to ensure a seamless experience for all users. By combining intuitive design, efficient workflow management, and a strong focus on local fashion culture, BogoGo aims to enhance the connection between Bogotá's fashion creators and consumers.

2 Canvas Business Model

The following canvas diagram serves as a **contextual overview** of BogoGo. It outlines each key component of the business, providing clarity on how they are connected and which elements play a crucial role in the company's operations.



Key Partners

- **Product Suppliers** (Designers, Distributors, Brands)
- **Logistics Providers** (Servientrega, InterRapidísimo)
- **Payment Platforms** (Mercado Pago)
- **Cloud Providers**

Key Activities

- Integration with Service Providers
- Development and Maintenance of the E-commerce Platform
- Real-time Data Ingestion & Processing (clickstream, inventory, sales)
- Business Dashboards

Key Resources

- Cloud Infrastructure
- E-commerce Platform
- Fashion Curators & Buyers
- Technology Team
- Logistics Systems
- Payment Gateways

Value Propositions

- Intuitive Navigation
- Fast & Reliable Delivery
- Secure Transactions
- Simple Returns
- Exclusive products
- Early Access to New Releases
- Limited Editions
- High Performance Platform
- Real-Time Business Insights
- High Availability & Low Latency

Customer Relationships

- Omnichannel Customer Support (Email, Phone)
- Loyalty & Rewards Programs
- Digital Community
- Intuitive Platform

Channels

- Web Platform
- Email

Customer Segments

- Individual Buyers (End consumers)
- Fashion Enthusiasts
- Small & Medium-Sized Businesses (Boutiques – physical and online)
- Designers
- Brands
- Managers & Admins

Cost Structure

- Development and maintenance of the platform.
- Cloud infrastructure services.
- Digital marketing and customer acquisition.
- Logistics (shipping and storage).
- Personnel (developers, data scientists, support, marketing).
- Software licenses and BI tools.

Revenue Streams

- Commissions from sales made through the platform.
- Advertising space sold within the platform.
- Sponsored loyalty programs by brands.

Model 1. Business model canvas.

2.1 Key Partners

- **Product Suppliers** (Designers, Distributors, Brands): Main source of inventory and catalog.
- **Logistics Providers** (Servientrega, InterRapidísimo): Selected for their
  - National coverage throughout Bogota.
  - Established infrastructure with local logistics experience.
  - Fast delivery capabilities.
  - Extensive pickup and return points.
- **Payment Platforms** (Mercado Pago): Selected for their
  - Multiple payment methods.
  - Secure transactions.
  - Compliant with local regulations.
  - Trusted by Colombian consumers.
- **Cloud Provider**: offers a robust ecosystem of cloud tools aligned with BogoGo's technical and business needs. These include:
  - Scalable infrastructure.
  - Real-time data ingestion and analytics.

  - Machine learning integration.
  - High availability across regions.

2.2 Key Activities

- **Integration with cloud and logistics providers:** Ensures scalability and reliability of the platform.
- **E-commerce platform development and maintenance:** Keeps the platform functional, updated, and optimized.
- **Real-time data ingestion and processing:** Enables immediate, data-driven decision-making.
- **Implementation of BI dashboards:** Visual insights into business performance.

2.3 Key Resources

- **Technology Team:** developers.
- **Custom e-commerce platform (mobile optimized):** Mobile-optimized, customized online store.
- **Fashion buyers and curators:** Experts in trends, curation, and selection.
- **Logistics integration systems:** Integrated supply chain and delivery partners.
- **Payment gateway integrations:** Seamless integration with multiple providers.

2.4 Value Propositions

- **Intuitive Navigation:** A user-friendly interface that makes shopping easy.
- **Fast and Reliable Delivery:** Optimized logistics through national partners.
- **Secure Transactions:** Multiple payment options with strong security.
- **Simple Returns:** Hassle-free return process available nationwide.
- **Exclusive products:** Unique products not found in mainstream retailers.
- **Early Access to New Releases:** Priority access to new collections.
- **Limited Editions:** Special collections and capsule drops.
- **High Performance Platform:** Fast, reliable, and scalable online shopping.
- **Real-Time Business Insights:** For sales and inventory management.
- **High Availability and Low Latency:** Fast query execution and seamless performance.

2.5 Customer Relationships

- **Omnichannel support (email, phone):** Customers can reach out via their preferred channel.
- **Loyalty and rewards programs:** Encourages repeat purchases and increases customer lifetime value.
- **Online fashion community:** Keeps customers engaged between purchases.
- **Easy-to-navigate platform:** Enables customers to shop independently and find what they need easily.

2.6 Customer Segments

- **Individual buyers** (End consumers): People shopping for personal use.
- **Fashion enthusiasts and early adopters:** Trend-savvy early adopters.
- **Small and medium-sized clothing businesses**(Boutiques – physical and online): Looking for unique inventory sources.
- **Emerging and established designers:** Emerging and established designers looking to grow their presence.
- **Brands looking to expand digitally:** Established labels aiming to expand digitally.
- **Managers and admins needing sales tools:** Professionals seeking analytics and sales tools.

2.7 Channels

- **Web platform:** Main e-commerce site.
- **Social media:** Email.

2.8 Cost Structure

- **Platform development and maintenance.**
- **Cloud infrastructure or hosting infrastructure**
- **Digital marketing and customer acquisition.**
- **Logistics (shipping, warehousing).**
- **Personnel.**
- **Software licenses and BI tools.**

2.9 Revenue Streams

- **Commissions on sales.**
- **Advertising space within the platform.**
- **Sponsored loyalty programs.**

3 Requirements

The following requirements provide a more precise delineation of the problem context, emphasizing the users' perspective rather than a system-centered approach. Furthermore, the non-functional requirements are supported by theoretical foundations that justify the proposed timeframes, which have been established based on rational criteria rather than arbitrary or empirical estimations.

3.1 Functional (FR)

User & Account Management

- Users (clients, vendors, and admins) must be able to create accounts and log in securely.
- Users (clients, vendors, and admins) must be able to manage their accounts.

Product & Catalog Management

- Vendors must be able to create, update, and delete products with attributes that them select (size, color, images).
- Clients must be able to search and filter products and providers by multiple attributes (name, category, price, rating, availability).

Vendors rate

- Clients must be able to rate vendors, after receive an order.
- Vendors, Clients and Administrators must be able to see the rate of a vendor.

Shopping & Transactions

- Customers must be able to to add/remove products from the shopping cart.
- Customers must be able to select multiple payment gateways.
- Vendors and Customers must be able to receive order statuses: confirmed, processing, delivered, returned.
- Clients must be able to rate vendors.
- Vendors must be able to receive status updates from the administrator regarding their products.

Mobile Friendliness

- Customers and vendors must be able to access the platform across multiple devices and platforms.

3.2 Non-Functional Requirements (NFR)

Performance & Fast Queries

- According to the Google Web Vitals guidelines, the homepage should load in less than 2.5 s [1].

Scalability

- Handle up to 500 concurrent users: BogoGo's infrastructure planning begins with a realistic growth model. Although there is no established average, the estimated number of users during the platform's first year of operation is approximately 5,000. Based on this projection, it is expected that around 10% of users will be active during peak hours, resulting in approximately 500 concurrent sessions.This approach allows the system to start with minimal resources and scale horizontally as demand increases. The estimate of 500 concurrent users reflects the anticipated load during BogoGo's early operational phase, while using an Auto Scaling approach ensures that the platform can expand seamlessly without requiring architectural redesign.

Availability & Reliability

- The system shall maintain 99% uptime: The 99% uptime target balances reliability with infrastructure costs[2].
- The system shall tolerate node failures with automatic failover.

- The system shall ensure no data loss in case of pipeline failure.

Security

- All sensitive data must be encrypted in transit and at rest:Data protection is critical for transactions and user trust.
- Only system administrators can assign roles and change access permissions.

Usability

- Customers must be able to complete purchases within 3 steps (browse → cart → checkout): Reducing purchase steps improves conversion rates and customer satisfaction, especially for mobile users, who represent over 73% of e-commerce traffic in Bogotá, according to the 2023 Financial Inclusion Report by Banca de las Oportunidades [4].
- BI dashboards must provide interactive and drill-down visualizations.

Maintainability

- The system shall use a microservices architecture for modular updates.
- The backend must be documented to simplify future modifications.

4 User Stories

4.1 Customer Stories

- CU01: Customer account registration

Title:Customer account registration	**Priority: high**	**Estimate: 3 days**
User Story: As a customer, I want to create an account so that I can securely access my purchase history.		
Acceptance Criteria: • Given I am on the registration page, when I enter valid personal data (name, email, password), then my account is created and I receive a confirmation email • Given I try to register with an already used email, when I submit the form, then I see an error message.		

- CU02: Product browsing and search

Title:Product browsing	**Priority: high**	**Estimate: 1 week**
User Story: As a customer, I want to browse and search for products so that I can quickly find what I need.		

Acceptance Criteria: • Given I am on the search bar, when I enter a keyword, then the system returns matching products within 2 seconds.   • Given I apply filters (price range, category), when I search, then only products matching those filters appear

- CU03: Shopping cart management

Title:Shopping cart management	**Priority: high**	**Estimate: 3 days**
User Story: As a customer, I want to add products to a shopping cart so that I can purchase multiple items at once.		
Acceptance Criteria: • Given I select a product, when I click "Add to cart," then the item appears in my cart. • Given I have items in my cart, when I update quantity, then the total is recalculated instantly.		

- CU04: Secure checkout and payment

Title:Secure checkout and payment	**Priority: high**	**Estimate: 1 week**
User Story: As a customer, I want to check out and pay securely so that I can complete my order successfully.		
Acceptance Criteria: • Given I am in the checkout flow, when I choose a valid payment method and confirm, then the order is created, and I receive confirmation. • Given I enter invalid card details, when I confirm, then the system rejects the payment and shows an error.		

- CU05: Order status tracking

Title:Order status tracking	**Priority: medium**	**Estimate: 3 days**

User Story: As a customer, I want to track the status of my order so that I know when it will arrive.
Acceptance Criteria: • Given I am logged in, when I view "My Orders," then I can see the current status (Confirmed, Processing, Delivered).

- CU06: Client rates vendor

Title:Client rates vendor	**Priority: medium**	**Estimate: 3 days**
User Story: As a client, I want to be able to rate vendors after receiving a service or product,so that I can provide useful feedback and help other users make informed decisions.		
Acceptance Criteria: • Given I am a client who has completed a transaction with a vendor,when I go to the vendor's profile and submit a rating, then the rating is saved and displayed on the vendor's profile. • Given I am rating a vendor, when I select a star rating from 1 to 5, then the system accepts and stores the rating and comment. • Given I already rated a vendor for a specific transaction, when I try to rate the same transaction again, then I see a message saying I have already submitted a rating.		

- CU07: Customer account registration

Title:Customer account registration	**Priority: high**	**Estimate: 3 days**
User Story: As a customer, I want to create an account so that I can securely access my purchase history.		
Acceptance Criteria: • Given I am on the registration page, when I enter valid personal data (name, email, password), then my account is created and I receive a confirmation email • Given I try to register with an already used email, when I submit the form, then I see an error message.		

- CU08: Client views vendor rating

Title: Client views vendor rating	**Priority: High**	**Estimate: 1 day**
User Story: As a client, I want to view a vendor's rating before making a purchase, so that I can make an informed decision based on other users' feedback.		
Acceptance Criteria: • Given I am browsing a product, when I view the vendor's profile or product page, then I can see their average rating and number of reviews. • Given a vendor has no ratings yet, when I view their profile, then I see a message indicating "No ratings yet".		

4.2 Vendor Stories

- VS01: Product listing management

Title:Product listing management	**Priority: high**	**Estimate: 4 days**
User Story: As a vendor, I want to create and update product listings so that customers always see accurate information.		
Acceptance Criteria: • Given I am a vendor, when I add a new product, then it appears in the product catalog. • Given I update stock to "0," when customers browse, then the product is marked as "Out of Stock."		

- VS02: Sales analytics dashboard

Title:Sales analytics dashboard	**Priority: medium**	**Estimate: 1 week**
User Story: As a vendor, I want to view my sales analytics so that I can make better business decisions.		
Acceptance Criteria: • Given I am logged in as a vendor, when I open the analytics dashboard, then I see daily/weekly/monthly sales charts.		

- VS03: Vendor views their own rating

Title: Vendor views their own rating	**Priority: Medium**	**Estimate: 1 day**
User Story: As a vendor, I want to see my average rating based on client feedback, so that I can understand how my services are being perceived and improve if necessary.		
Acceptance Criteria: • Given I am logged into my vendor dashboard, when I navigate to my profile, then I can view my current average rating and total number of reviews. • Given a new rating is submitted, when I refresh my dashboard, then my average rating is updated accordingly.		

- VS04: Vendor product status updates

Title: Vendor product status updates	**Priority: Medium**	**Estimate: 2 days**
User Story: As a vendor, I want to receive updates from the administrator about the status of my submitted products, so that I can track their approval or rejection.		
Acceptance Criteria: • Given I have submitted a product for review, when the administrator updates its status, then I receive a notification with the updated status. • Given a product has been approved or rejected, when I log in to my vendor dashboard, then I can view the current status of all submitted products. • Given I receive a status update, when I click on the notification, then I am redirected to the detailed product page.		

4.3 Administrator Stories

- AS01: Sales and customer behavior reports

Title:Sales and customer behavior reports	**Priority: high**	**Estimate: 1 week**
User Story: As an administrator, I want to generate sales and customer behavior reports so that I can evaluate performance.		

Acceptance Criteria:
• Given I select a time range, when I generate a report, then I receive a PDF/CSV file with sales metrics.

- AS02: Vendor and customer management

Title:Vendor and customer management	**Priority: high**	**Estimate: 4 days**
User Story: As an administrator, I want to manage vendors and customers so that I can maintain a secure and trusted marketplace.		
Acceptance Criteria: • Given I am an admin, when I deactivate a vendor, then their products become unavailable.		

- AS03: Administrator views vendor rating

Title: Administrator views vendor rating	**Priority: Medium**	**Estimate: 1 day**
User Story: As an administrator, I want to access the ratings of all vendors so that I can monitor vendor performance and address issues if necessary.		
Acceptance Criteria: • Given I am logged in as an administrator, when I access the vendor management panel, then I can view each vendor's average rating and review count. • Given I view a specific vendor, when I open their detail page, then I see a list of individual ratings and comments left by clients.		

5 Database Architecture

- **High-Level Architecture Overview:**
  The BogoGo platform operates as an e-commerce system built on a three-layer architecture that separates the user interface, backend business logic, and data infrastructure. Its data model manages users—assigned to roles such as admin, vendor, or customer—who interact through products, orders, payments, and vendor ratings. Vendors create and manage product listings enriched with categories

and images, while customers place orders and evaluate their experience. All commercial interactions, including transactions and delivery statuses, are carefully recorded to ensure transparency and controlled access. Instead of relying on long textual descriptions, the platform's structure and data interactions are illustrated through a Data Flow Diagram (DFD) and a BPMN process model, offering a clearer and more operational understanding of how the system functions.

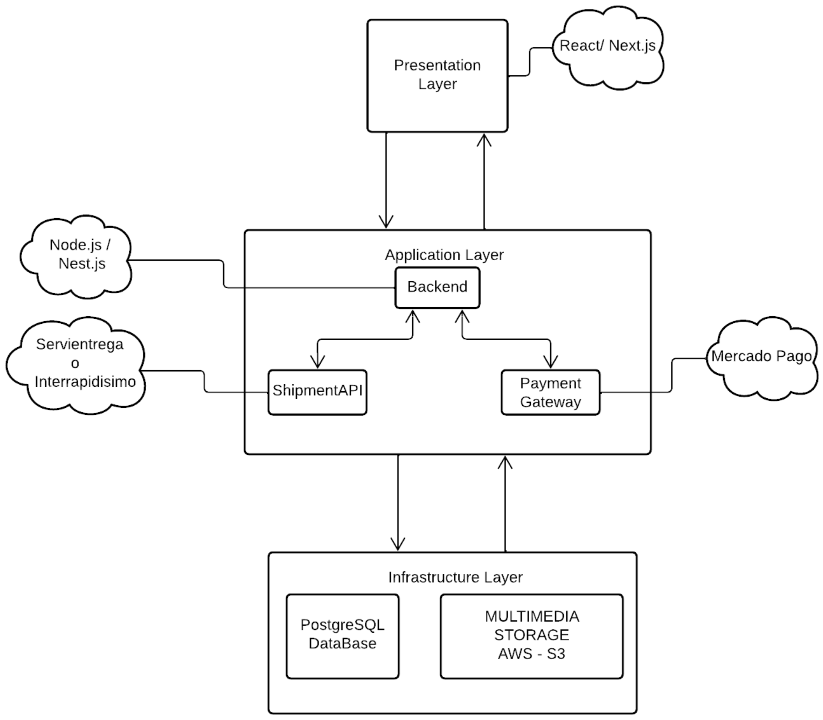
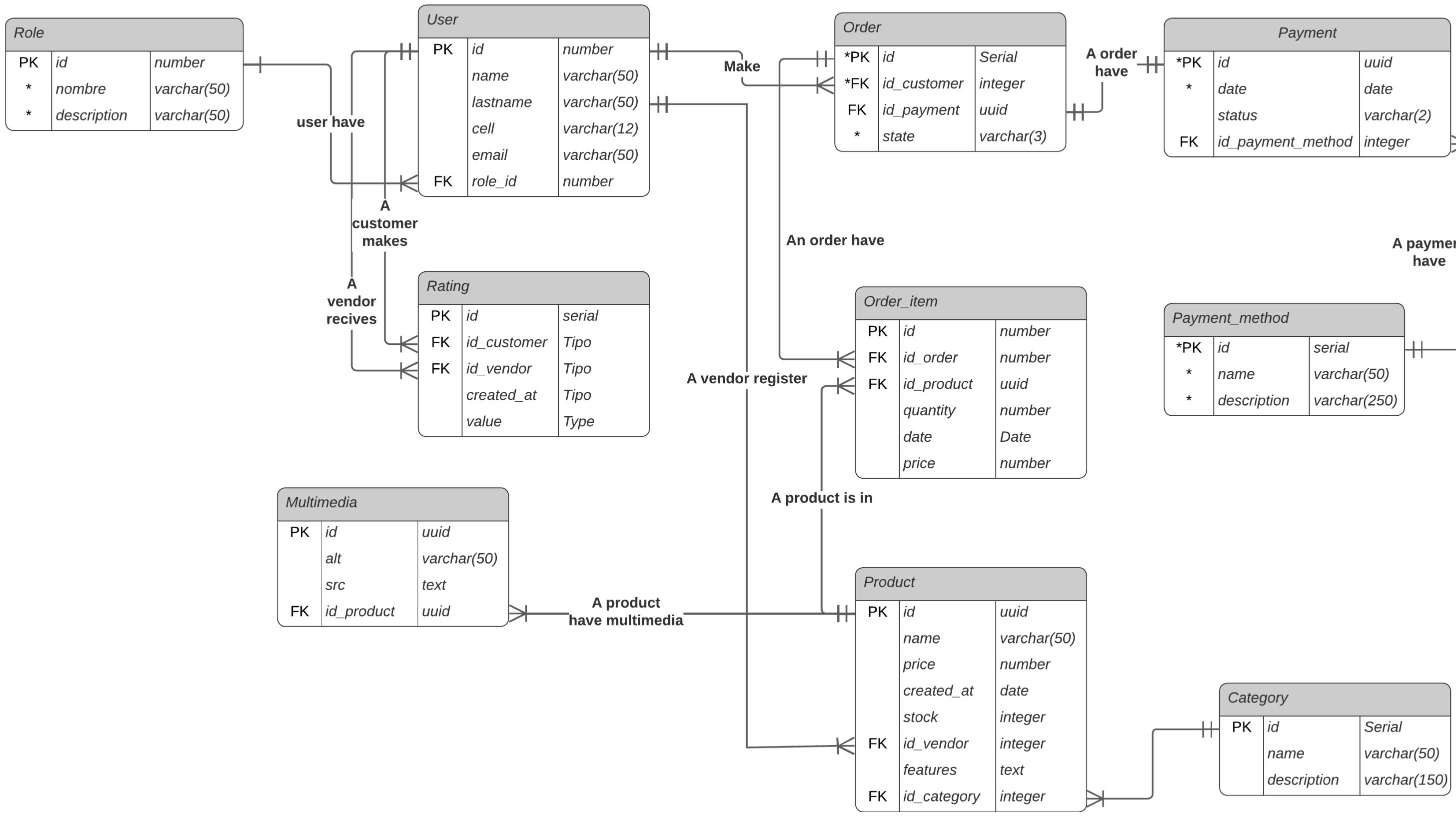


Figure 1: First version of high level architecture diagram



Role		
PK	id	number
*	nombre	varchar(50)
*	description	varchar(50)

User		
PK	id	number
	name	varchar(50)
	lastname	varchar(50)
	cell	varchar(12)
	email	varchar(50)
FK	role_id	number

Order		
*PK	id	Serial
*FK	id_customer	integer
FK	id_payment	uuid
*	state	varchar(3)

Payment		
*PK	id	uuid
*	date	date
	status	varchar(2)
FK	id_payment_method	integer

Rating		
PK	id	serial
FK	id_customer	Tipo
FK	id_vendor	Tipo
	created_at	Tipo
	value	Type

Order_item		
PK	id	number
FK	id_order	number
FK	id_product	uuid
	quantity	number
	date	Date
	price	number

Payment_method		
*PK	id	serial
*	name	varchar(50)
*	description	varchar(250)

Multimedia		
PK	id	uuid
	alt	varchar(50)
	src	text
FK	id_product	uuid

Product		
PK	id	uuid
	name	varchar(50)
	price	number
	created_at	date
	stock	integer
FK	id_vendor	integer
	features	text
FK	id_category	integer

Category		
PK	id	Serial
	name	varchar(50)
	description	varchar(150)

Figure 2: Second Version of Entity Relationship Diagram

- **Main components:**
  The database system for the BogoGo platform is designed under a **3-layer architecture** model, supporting both transactional and basic analytical workloads, with scalability and high availability as priorities to handle big data and distributed system requirements.

  1. **Presentation Layer:**
     – User interface (web) interact with the backend services via REST APIs.
  2. **Application Layer:**
     – Manages business logic, requests, authentication, and communication with the database services and the other external services used within the application.
     – **Payment Gateways:** We want to use the "Mercado Pago" API for all of the payment's transactions, This is due to its well-established platform in the country and its wide catalog of payment methods.
     – **Shipping API's:** It is used to manage the shipping information and delivery of products.

     The application layer handles the core business logic, coordinating how information flows between the presentation layer and the infrastructure layer. It manages requests, authentication, and the overall execution of operations within the platform. This layer ensures that processes related to orders, products, payments, and shipments are executed reliably and consistently, serving as the central link that connects users with the system's data and functionality.
     Technologies:
     – Node.js or NestJS
       ∗ Node.js was considered for BogoGo's backend because it is fast, flexible, and familiar to the development team, allowing efficient development of the platform. Its event-driven, non-blocking architecture makes it well-suited for handling multiple simultaneous operations, such as user logins, product searches, and order processing. Node.js also provides

a wide ecosystem of packages through npm—such as Express for routing, Mongoose for MongoDB integration, Axios for API requests, and bcrypt for secure password management—which helps streamline development and simplifies common backend tasks. This combination of performance, flexibility, and ecosystem support makes Node.js a practical and reliable option for BogoGo.

∗ NestJS was also considered because it builds on Node.js while providing a more structured and modular architecture. Developed with TypeScript, NestJS encourages the creation of organized, maintainable, and scalable code. Its features—such as dependency injection, modular controllers, and built-in support for REST and GraphQL APIs—facilitate the development of clean and robust backend services. Additionally, NestJS integrates smoothly with databases like PostgreSQL and MongoDB, and supports libraries such as TypeORM for database management, Passport for authentication, and class-validator for input validation. These capabilities make NestJS a strong option for building a well-structured and maintainable backend for BogoGo.

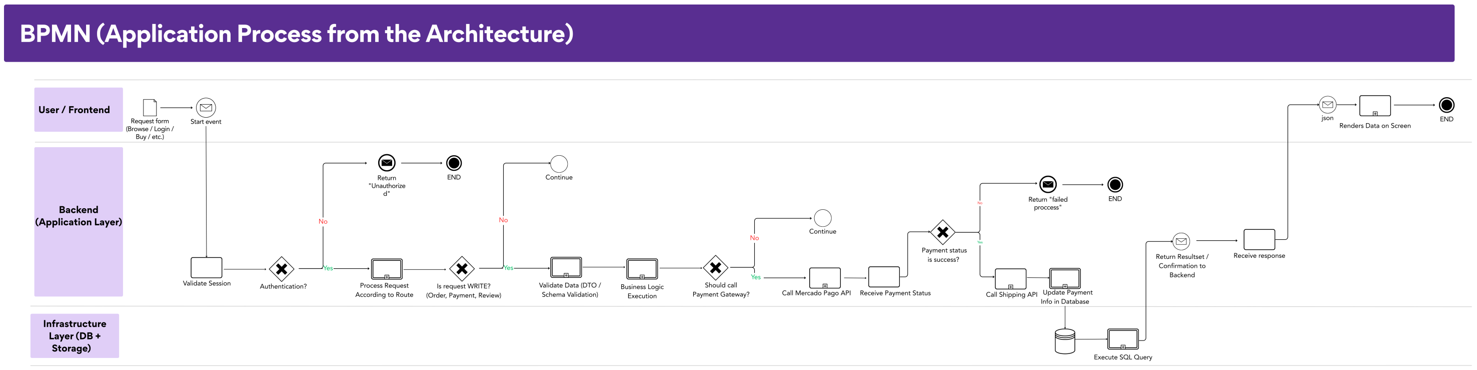

BPMN (Application Process from the Architecture)
User / Frontend
Backend (Application Layer)
Infrastructure Layer (DB + Storage)
Request form (Browse / Login / Buy / etc.)
Start event
Validate Session
Authentication?
No
Yes
Return "Unauthorize d"
END
Process Request According to Route
Is request WRITE? (Order, Payment, Review)
No
Yes
Continue
Validate Data (DTO / Schema Validation)
Business Logic Execution
Should call Payment Gateway?
No
Yes
Continue
Call Mercado Pago API
Receive Payment Status
Payment status is success?
No
Yes
Return "failed proccess"
END
Call Shipping API
Update Payment Info in Database
Execute SQL Query
Return Resultset / Confirmation to Backend
Receive response
json
Renders Data on Screen
END

Figure 3: BPMN diagram

This BPMN diagram illustrates the end-to-end flow of an application request, including authentication, routing, data validation, business logic execution, external API integration, database access, and response rendering. [1]

3. **Infrastructure Layer (Data Storage & External Services):**
   – **Relational Database:** A relational SQL database is used to store structured information such as users, products, transactions, and analytical data. Its consistency and reliability make it suitable for managing the core operational data of the e-commerce platform.
   – **Object Storage:** An object storage service is used to store multimedia assets such as product images, banners, and videos, allowing efficient handling of large static files independently from the transactional database.

The infrastructure layer combines multiple storage solutions and external services to support the platform's operations. A relational database stores structured data such as users, roles, vendors, products, orders, and ratings. Digital assets, including images and videos, are managed through an object storage solution like . Additionally, the platform integrates with external services such as payment gateways (e.g., Mercado Pago) and shipping APIs to handle transactions and product deliveries efficiently. This layer guarantees scalability, reliability, and secure management of all operational data.

Technologies:

– Relational Databases
  ∗ A relational database system is selected due to its robustness, well-defined structure, and full ACID compliance, which ensures data integrity, consistency, and security during transactions.

[1] The original BPMN diagram format is available at: https://drive.google.com/file/d/1cYVWJxSEuKPPu_E12hZfGIyM-wIgys_Q/view?usp=sharing

For a district-level clothing e-commerce platform, where orders, inventory, users, and financial transactions must be managed reliably, a relational model provides structured, normalized, and easily queryable data. Given the moderate volume of structured transaction data, high frequency of updates (orders, inventory changes, user interactions) and the need for complex joins (for example: sales by product, customer behaviour), a relational database is highly appropriate. Its capacity for vertical scaling and efficient querying makes it suitable for an academic project that aims to simulate real commercial operations.

- Object Storage Service
    * An object storage service is proposed for managing product images, promotional banners, and other multimedia content. Because the volume of multimedia assets (images, videos) is large, their structure is unstructured or semi-structured, and the frequency of updates is relatively low (assets are uploaded less often, but read many times), a dedicated object storage model is optimal. The object storage system allows efficient, cost-effective management of large binary files, separation of static content from transactional data, and supports scalability in storing and delivering heavy static loads without impacting the transactional database.

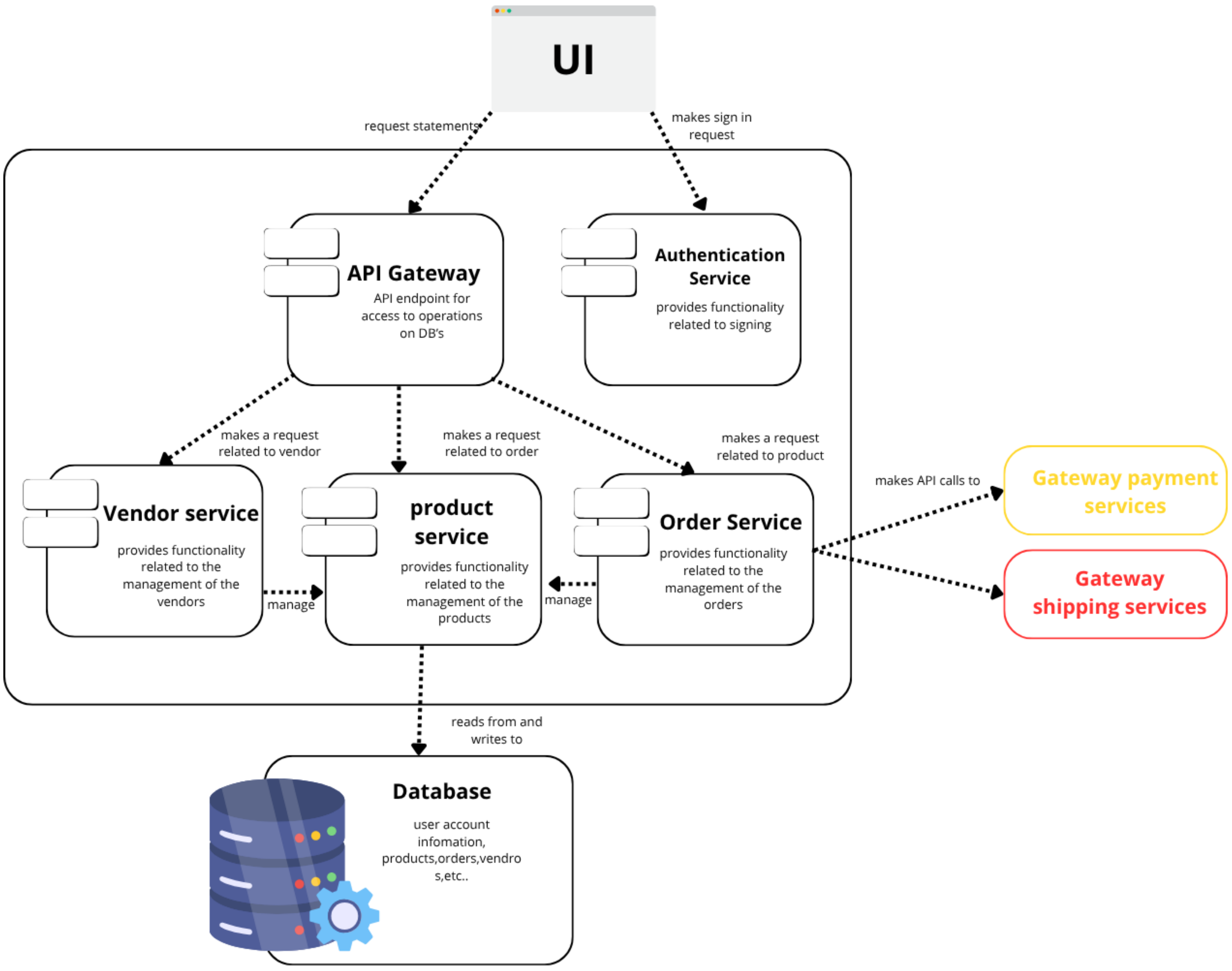

UI
request statements
makes sign in request
API Gateway
API endpoint for access to operations on DB's
Authentication Service
provides functionality related to signing
makes a request related to vendor
makes a request related to order
makes a request related to product
Vendor service
provides functionality related to the management of the vendors
product service
provides functionality related to the management of the products
Order Service
provides functionality related to the management of the orders
manage
manage
makes API calls to
Gateway payment services
Gateway shipping services
reads from and writes to
Database
user account infomation, products,orders,vendros,etc..

Figure 4: Data flow diagram

The DFD provides a high-level representation of how users, vendors, external services, and internal components exchange data across the three layers.

- **Data Flow Between Layers:**
  The interaction across layers follows a structured sequence:

  1. **Presentation Layer**: The user initiates an operation such as browsing products, logging in, or placing a purchase.
  2. **Application Layer**: This layer is responsible for interpreting and processing the user's request. Its functions include:
     – Receiving the incoming request
     – Validating authentication through JWT
     – Executing the appropriate business logic
     – Communicating with the relational database, object storage, or external APIs

3. **Infrastructure Layer**: The infrastructure layer handles data persistence and external service responses. Its main operations include:
   - Executing SQL queries
   - Providing product media and static assets from object storage
   - Returning payment confirmations or shipping information from integrated external services

After the required operations are completed, the application layer assembles the final response and returns it to the Presentation Layer, ensuring secure, accurate, and consistent handling of all e-commerce functionalities.

6 Information Requirements

Types of Data Returned by the System. The system provides the following types of data in response to user actions and queries:

- **Available Products List:** Returns a list of products currently available in the catalog, as defined in the Product and Catalogue Management requirement (User Story CU-02).
- **Top-Selling Products Report:** Provides a report of the best-selling products, as described in User Story VS-02.
- **Sales Behavior Report:** Offers insights into sales trends and patterns, following the specifications of User Story AS-01.
- **Product Status:** Indicates whether products are active or inactive, in accordance with User Story CU-05.
- **Product Management Confirmation:** Returns success or failure messages for product administration actions, as referenced in User Story VS-01.
- **Vendor Rating:** Provides the rating score of a selected vendor, based on User Stories CU-08, VS-03, and AS-03.

7 Query Proposals

7.1 Available Products List

Purpose: Retrieves all active products currently available for purchase, including price, stock, category, and vendor. Only products with stock greater than zero are shown.

```
SELECT
  p.id,
  p.name,
  CAST(p.price AS numeric) AS price,
  p.stock,
  p.created_at,
  c.name AS category,
  u.id AS vendor_id,
  (u.name || ' ' || u.lastname) AS vendor_name,
  p.features
FROM "Product" p
LEFT JOIN "Category" c ON p.id_category = c.id
LEFT JOIN "User" u ON p.id_vendor = u.id
WHERE p.stock > 0
ORDER BY p.created_at DESC;
```

7.2 Top-Selling Products Report

Purpose: Generates a ranking of the best-selling products in a given date range, including total units sold and total revenue.

```
SELECT
  p.id,
  p.name,
  SUM(oi.quantity) AS total_units_sold,
  SUM(CAST(oi.price AS numeric) * oi.quantity) AS total_revenue
FROM "Order_item" oi
JOIN "Product" p ON oi.id_product = p.id
WHERE oi.date BETWEEN :start_date AND :end_date
GROUP BY p.id, p.name
ORDER BY total_units_sold DESC
LIMIT :limit_n;
```

7.3 Sales Behavior Report

Purpose: Provides monthly aggregated sales data (orders, units sold, revenue) to analyze sales trends over time.

```
SELECT
  date_trunc('month', oi.date)::date AS month,
  COUNT(DISTINCT oi.id_facture) AS orders_count,
  SUM(oi.quantity) AS total_units_sold,
  SUM(CAST(oi.price AS numeric) * oi.quantity) AS total_revenue
FROM "Order_item" oi
WHERE oi.date BETWEEN :start_date AND :end_date
GROUP BY month
ORDER BY month;
```

7.4 Product Status

Purpose: Determines if a product is active or inactive based on stock availability.

```
SELECT
  p.id,
  p.name,
  p.stock,
  CASE
    WHEN p.stock > 0 THEN 'active'
    ELSE 'inactive'
  END AS status,
  p.created_at
FROM "Product" p
ORDER BY status DESC, p.name;
```

7.5 Product Management Confirmation

Purpose: Creates a new product and immediately returns its key fields for API confirmation using `RETURNING`.

```
INSERT INTO "Product"
(id, name, price, created_at, stock, id_vendor, features, id_category)
VALUES (:id, :name, :price, :created_at, :stock, :id_vendor, :features, :id_category)
RETURNING id, name, stock, id_vendor;
```

7.6 Vendor Rating

Purpose: Calculates average rating and total reviews per vendor, ordering them by best performance.

```
SELECT
  r.id_vendor,
  (u.name || ' ' || u.lastname) AS vendor_name,
  COUNT(r.id) AS num_ratings,
  ROUND(AVG(r.value)::numeric, 2) AS avg_rating
FROM "Rating" r
JOIN "User" u ON r.id_vendor = u.id
GROUP BY r.id_vendor, u.name, u.lastname
ORDER BY avg_rating DESC;
```

7.7 Performance and Optimization Considerations

Although the proposed queries are appropriate and aligned with the project requirements, several performance strategies are considered to ensure scalability in scenarios with large datasets:

– **Indexing strategy:** B-tree indexes are created on the most frequently filtered and joined columns, such as `"Product"(stock, created_at)`, `"Order_item"(date, id_product)`, and `"Rating"(id_vendor)`. These indexes improve the performance of listing, reporting, and aggregation queries.

– **Pagination and result limits:** Read-heavy queries like the *Available Products List* are combined with `LIMIT` (and optionally keyset pagination) to avoid loading excessive rows into memory and to keep response times predictable.

– **Pre-aggregation and reporting:** For analytical reports such as *Top-Selling Products* and *Sales Behavior*, the system can employ materialized views or summary tables (e.g., daily or monthly sales) that are periodically refreshed. This reduces the computational cost of recalculating aggregates over large order histories.

– **Hybrid storage and NoSQL complements:** Although the core transactional data is managed in a relational database, the system may use NoSQL technologies as a complement. For example, a key–value store or in-memory cache (e.g., Redis) can be used to cache frequently requested product lists or vendor ratings, and a document or search engine (e.g., Elasticsearch) can support advanced text search on product features without overloading the primary SQL database.

8 Concurrency Analysis

This section identifies scenarios within the BogoGo platform where concurrent access to shared data may occur, describes potential consistency problems (e.g., race conditions, deadlocks), and proposes mechanisms such as isolation levels, pessimistic and optimistic locking, and idempotency to ensure transactional integrity.

Case 1: Concurrent Checkout and Stock Handling

During the checkout process, multiple customers may attempt to purchase the same clothing item simultaneously. Since products on BogoGo include attributes such as size and limited stock, concurrent writes may occur when confirming purchases. These interactions, if not managed properly, can lead to incorrect stock deduction or duplicated orders, especially when customers retry payment requests during latency or network delays.

Concurrency Risks:

- Race conditions leading to overselling when multiple transactions read the same stock value.
- Lost updates where valid stock reductions are overwritten by stale data.
- Duplicate checkout confirmation requests causing repeated order creation and incorrect inventory reduction.

Proposed Solutions:

- Use `SELECT FOR UPDATE` to apply pessimistic locks on stock rows during checkout.
- Increase isolation level to `SERIALIZABLE` to prevent uncommitted visibility and concurrent write conflicts.
- Utilize idempotency keys to avoid processing duplicate confirmation attempts.
- Add integrity constraints such as `CHECK(stock >= 0)` to reject invalid inventory states.

Case 2: Vendor Updates to Pricing and Catalog While Customers Browse

Vendors can modify product information in real time, including stock levels, pricing, and multimedia content. Meanwhile, many customers may be browsing the catalog and filtering products to make purchasing decisions. As these updates and reads happen concurrently, inconsistencies may appear in prices or availability if customers load outdated product data during checkout.

Concurrency Risks:

- Dirty reads where customers temporarily view uncommitted catalog modifications.
- Inconsistency between the displayed price and the final price confirmed at checkout.
- Phantom reads where items or sizes appear or disappear between catalog queries.

Proposed Solutions:

- Apply optimistic concurrency control using versioning or timestamps to detect conflicting product updates.
- Confirm final product price and availability in the order service before payment execution.
- Use caching or read replicas for product browsing operations to reduce lock contention and improve performance.

Case 3: Order State Updates With External Payment and Shipping Services

Once a customer initiates a payment, several asynchronous processes may attempt to update the same order record. For example, the payment gateway callback may confirm payment while the customer manually retries the payment confirmation due to a delayed response. Additionally, logistics services may update delivery status concurrently with order management functions. Uncontrolled parallel updates can lead to invalid state transitions or duplicate shipments.

Concurrency Risks:

- State conflicts that cause invalid status transitions such as reverting from `PAID` to `CANCELLED`.
- Duplicate processing of shipment creation or payment confirmation events.
- Deadlocks when external and internal services update dependent records in different orders.

Proposed Solutions:

- Enforce a state machine model to allow only valid order status transitions.
- Define a consistent lock ordering strategy, such as always updating payment first and then order state.
- Implement Saga or transactional outbox patterns to ensure reliable integration with external APIs.
- Apply idempotency tokens to guarantee that repeated callbacks or retries do not generate duplicate operations.

9 Parallel and Distributed Database Design

To ensure high performance, availability, and scalability, BogoGo's architecture adopts distributed data management and parallel processing strategies aligned with its operational and analytical demands.

9.1 Distributed Transactional Architecture

The core database follows a **primary–replica** configuration:

- **Primary OLTP node**: Executes all write operations with strong consistency (orders, inventory, payments).
- **Read replicas**: Serve search, browsing, and ratings queries in parallel, reducing load and latency.
- **Caching layer**: Frequently accessed catalog and rating data is stored in-memory to accelerate responses.

Most operational workloads—search, browsing, vendor ratings—are read-heavy. Routing these queries to replicas and cache ensures low latency under concurrent demand, while the primary node preserves strong consistency for financial transactions. Replication also contributes to system continuity by allowing failover if the primary database becomes unavailable, supporting availability targets and seamless user experience.

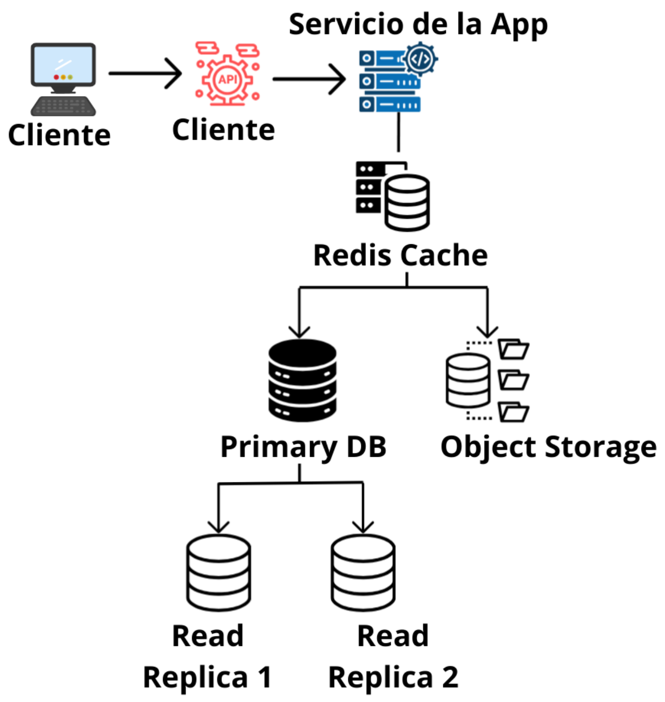


Figure 5: Distributed OLTP architecture with primary node, read replicas and caching layer for horizontal scalability.

9.2 Partitioning and Scalability Strategy

To optimize performance as data grows:

- **Time-based partitioning** for orders and order items: improves recent-query performance and enables parallel scans during reporting.
- **Logical sharding by region or vendor group**: supports growth in local inventory and delivery operations with improved data locality.

These strategies allow horizontal scaling without redesigning the schema.

Order information grows rapidly over time and is frequently queried by date in sales reporting and order tracking. Partitioning order data by time improves response speed and naturally enables parallel processing during analytical workloads. Sharding by Bogotá regions or vendor clusters reduces cross-node contention and aligns database growth with the platform's geographical expansion in local delivery and supply chains.

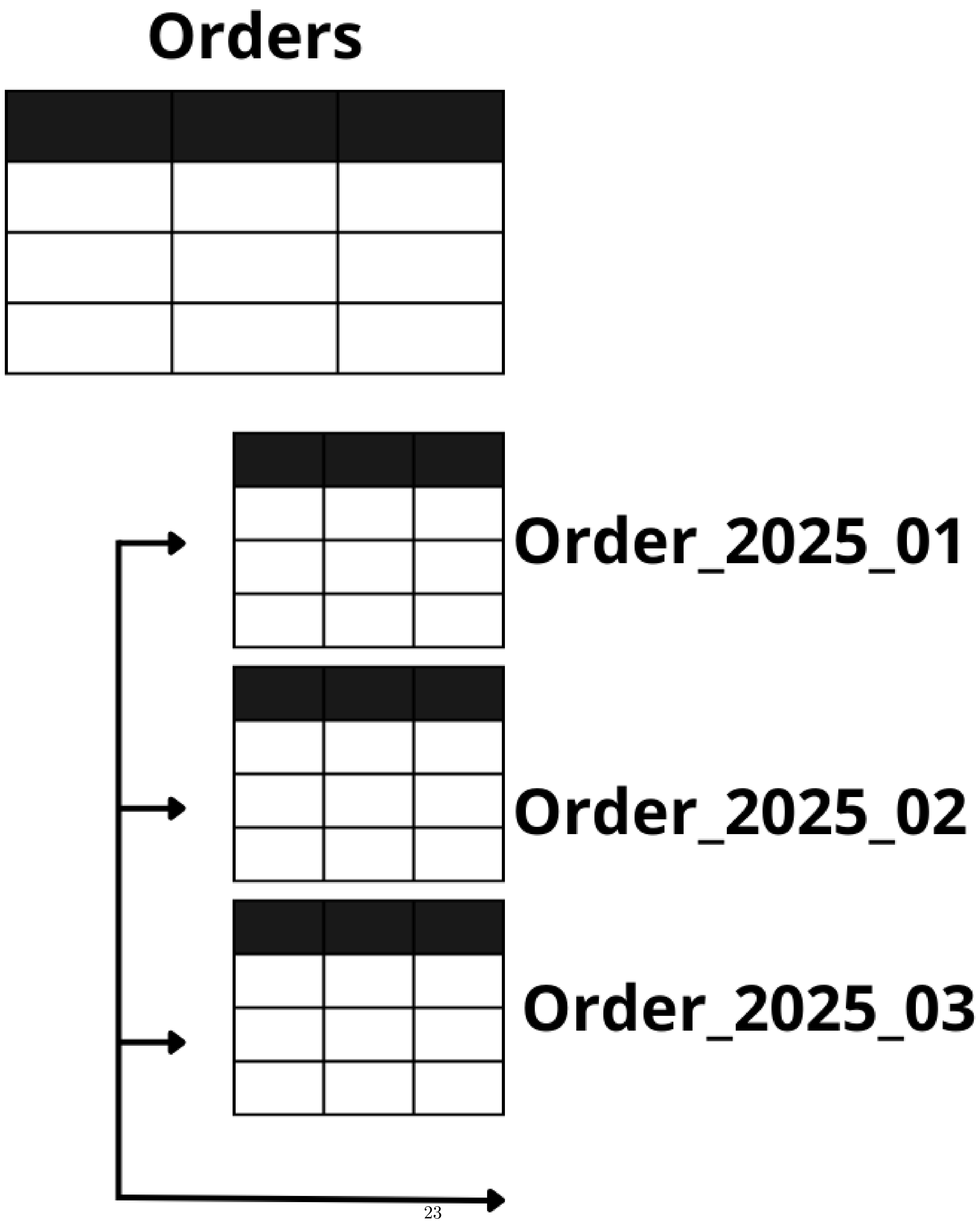


Figure 6: Time-based table partitioning enabling parallel query execution and optimized recent data retrieval.

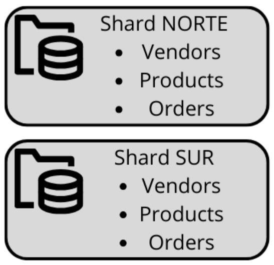


Figure 7: Logical sharding strategy aligned with Bogotá zone expansion and delivery optimization.

9.3 Analytics and Parallel Reporting

A **dedicated analytical node** (or reporting schema) receives periodic ETL loads from transactional data and stores pre-aggregated sales metrics and vendor performance indicators.

- BI dashboards and administrative reports query this node.
- ETL tasks execute in parallel by date or vendor segments to avoid impacting OLTP performance.

Analytical queries such as revenue trends, top-selling products, and vendor performance require full-table scans and aggregations that would degrade the responsiveness of transactional operations. Moving these workloads to an analytical node allows dashboards to remain interactive while ensuring that order placement and inventory updates remain stable and responsive.

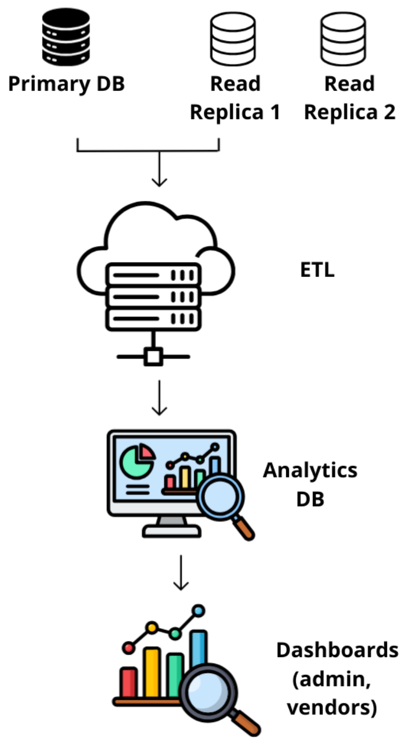


Figure 8: Analytical workload offloading through periodic ETL to a dedicated reporting database.

9.4 Query Distribution

- **Browsing and product search**: replicas + cache for low-latency response.
- **Checkout and inventory updates**: primary node to guarantee consistency.
- **Sales and vendor analytics**: analytical node for heavy aggregations.

This separation isolates critical transactions while boosting analytical throughput.

Each major user story is supported by the component best suited for its workload type: customer browsing benefits from parallel read scaling, checkout processes remain strictly consistent, and sales insights operate independently of core purchasing flows. This alignment ensures business priority operations remain fast and reliable.

9.5 Scalability and Availability Benefits

- Horizontal read scaling through replication and caching.
- Improved reporting performance with partitioning and analytical offloading.
- Failover capabilities via replica promotion.
- Low-latency experience for customers across Bogotá.

This architecture positions BogoGo to support increasing catalog size, vendor expansion, and data-driven decision-making without sacrificing operational efficiency.

By combining replication, caching, partitioning, and analytical offloading, the platform meets its scalability and availability goals while maintaining predictable performance as traffic and stored data increase.

These capabilities enable continuous growth in shoppers, vendors, and catalog inventory while preserving service quality and uptime commitments.

10 Performance Improvement Strategies

BogoGo implements several performance optimization strategies based on parallelism and distributed data processing. These techniques enhance query responsiveness, increase system throughput, and maintain a smooth user experience during peak demand.

10.1 Horizontal Read Scaling Through Replication

Multiple read replicas are used to distribute browsing, search, and vendor rating queries:

- Allows parallel execution of high-volume read traffic.
- Reduces latency by removing read load from the primary transactional node.
- Ensures continuity of service in case of primary node degradation.

Trade-offs:

- Eventual consistency in replicas may briefly show outdated stock or vendor information.
- Requires monitoring and automated failover mechanisms to preserve availability.

10.2 Data Partitioning for Parallel Query Execution

Temporal partitions for orders and order items improve performance as data grows:

- Queries targeting recent activity scan only the most relevant partitions.
- Historical reporting can leverage parallel scanning across partitions.
- Facilitates ETL parallelization during analytical processing.

Trade-offs:

- Cross-partition joins may become more expensive if not properly indexed.
- Partition strategy must be periodically reviewed as data patterns evolve.

10.3 Caching Layer for Hot Data Acceleration

Frequently accessed product lists and vendor ratings are cached in-memory:

- Eliminates repetitive queries to the database.
- Minimizes response time during peak browsing hours.
- Supports sudden traffic spikes triggered by promotions or new releases.

Trade-offs:

- Stale cache data must be invalidated when stock or ratings change.
- Cache capacity planning is required to avoid memory exhaustion.

10.4 Challenges and Mitigation

While the strategies above significantly improve system responsiveness, distributed performance introduces operational complexity:

- **Consistency vs. availability:** asynchronous replication requires validation rules to prevent overselling.
- **Operational visibility:** monitoring, logging, and automated recovery become essential.
- **Workload balancing:** intelligent routing strategies are needed to maximize parallel use of resources.

BogoGo addresses these challenges by maintaining strict consistency for transactional updates in the primary node, while parallelizing reads and analytical workloads as much as possible. This approach ensures fast user interactions with the catalog and dashboards, while preserving the integrity of sales and inventory data.

11 Changes from Workshop 2

All mentions of the use of AWS (Amazon Web Services) were completely removed from the different sections of the document. Additionally, proper referencing was added for the metrics established in the non-functional requirements, which had already been consulted from various sources to support them, but had not been referenced previously, making them seem like "assumptions."
The ER diagram was redesigned using the Lucidchart tool to improve the visualization of the diagram in the document. During the redesign process, changes were made to the structure of the entities to provide greater clarity and better organization of the attributes defined for each entity. A more detailed justification was also added regarding the reasons for implementing the technologies in terms of the database. In the "Non-functional requirements" section, references were added to the pages and reports that were used as a reference to establish the metrics mentioned above. In addition, in the "Database Architecture" section, the total amount of text that was part of the explanations was reduced and a BPMN diagram was added instead to provide a simpler and/or easier-to-understand flow for readers.

References

[1] "Web Vitals | Articles | web.dev." Web.dev. Accessed October 18, 2025. [Online]. Available: `https://web.dev/articles/vitals?hl=es-419`

[2] "Amazon RDS Service Level Agreement." Amazon Web Services, Inc. Accessed October 18, 2025. [Online]. Available: `https://aws.amazon.com/es/rds/sla/`

[3] "Theseus: E-commerce Conversion Rate Optimization Based on Customer Satisfaction and User Experience Surveys — Case Study: Company X." Ammattikorkeakoulut – Theseus. Accessed October 18, 2025. [Online]. Available: `https://www.theseus.fi/handle/10024/355395`

[4] "E-commerce in Bogotá — Invest in Bogotá." Investinbogota.org. Accessed October 18, 2025. [Online]. Available: `https://es.investinbogota.org/e-commerce/`