Workshop N° 2 - 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 for vendors and administrators. The platform emphasizes scalability, performance, and security, using Amazon Web Services (AWS) for hosting, storage, and data processing—ensuring high availability, low latency, and a smooth user experience. Through its combination of technology, usability, and local focus, BogoGo aims to strengthen 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.

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** (AWS): 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

- **AWS cloud infrastructure:** For scalable computing, storage, and deployment.
- **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 (AWS).**
- **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 seconds.

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 AWS Auto Scaling 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, aligning with AWS regional service-level agreements and early-stage operational resources.
- 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. AWS's integrated encryption tools make end-to-end security feasible even for small-scale deployments.
- 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..
- 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 database represents an e-commerce platform designed to manage vendors, products, customers, and their commercial interactions. At the core of the model, the **User** entity stores essential information such as email, password, name, and account status. Each user is associated with a specific **Role**, which defines their permissions in the system (for example, administrator, vendor, or customer). Vendors are users who register products and manage their availability within the platform.

Users interact with the system mainly through the **Order** entity, which records their purchases. Each order connects a **Customer** with one or more **Products**, registering details such as order date, total amount, and status (pending, shipped, delivered, or canceled). Products represent the main assets of the platform and include attributes like name, description, price, stock, size, color, and material.

Products are connected to several complementary entities. The **Category** entity organizes products by type (e.g., clothing, accessories, footwear), allowing customers to filter and explore items easily. The **Image** entity stores the multimedia content of each product, enabling multiple visual representations. Furthermore, each product is directly linked to a **Vendor**, identifying the user responsible for its publication, pricing, and management.

The platform also includes the **Payment** entity, which registers transaction details such as payment method, confirmation status, and transaction date, as well as the **Shipment** entity, which manages delivery information including address, carrier, and tracking code. Additionally, a **Rating** entity allows customers to evaluate a vendor, providing an overall score that reflects product quality and customer satisfaction.

Overall, the model describes a structured environment where users, organized by role, engage in buying and selling fashion products. These products are enriched with detailed metadata, visual resources, and vendor information, while the system carefully records commercial transactions, payments, and deliveries. The design ensures efficient management of products, transparent customer interaction, and controlled access according to user roles and transaction integrity.

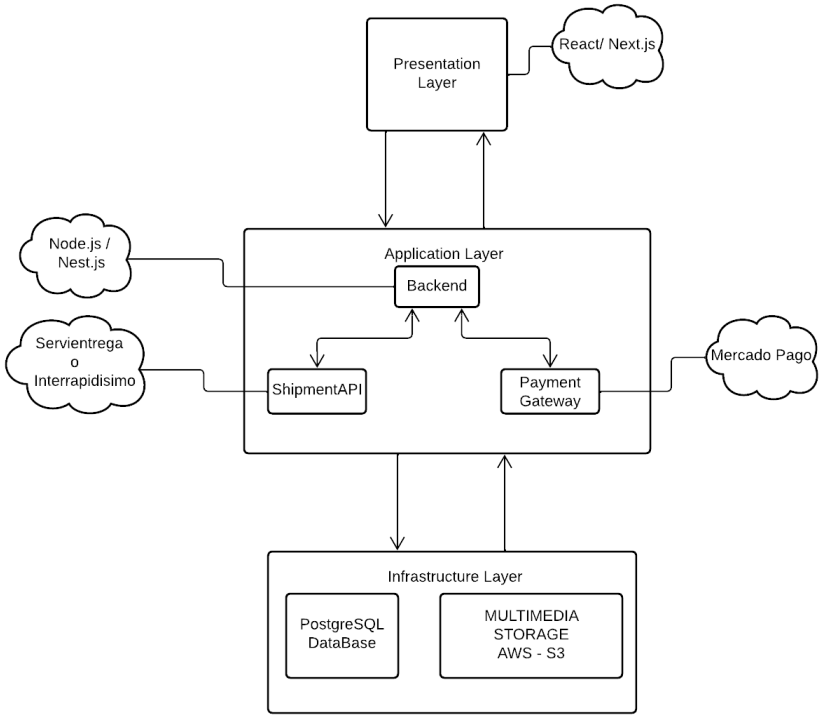

Figure 1: First version of high level architecture diagram

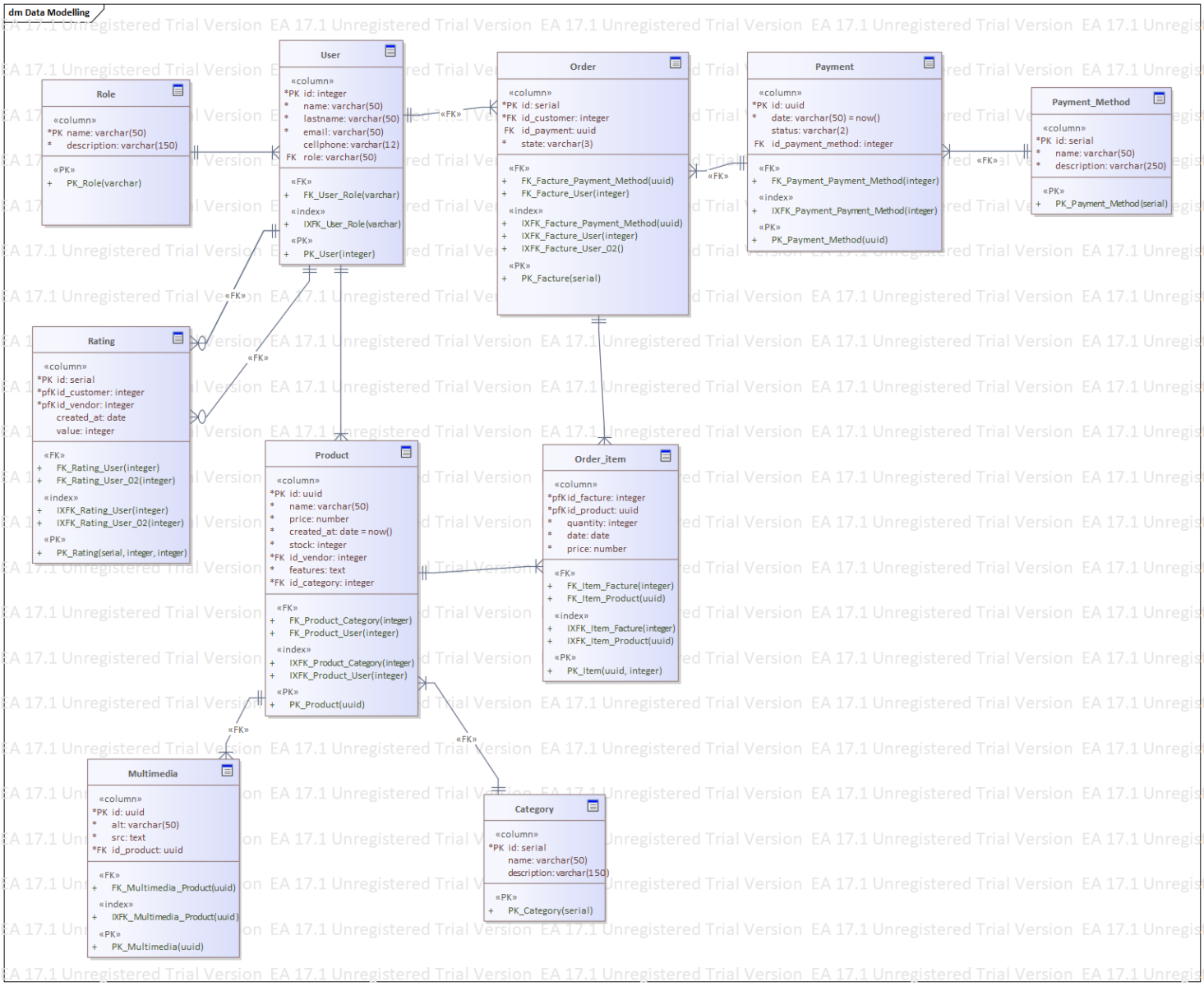

Figure 1. Second Version of Entity Relationship Diagram

- **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.

     This layer delivers the user-facing interface, whether through web or mobile applications. It communicates with backend services via REST APIs, enabling seamless interaction between users and the system while keeping business logic separate from the client side.
     Technologies:
     - React or Next.js

    * React was chosen for the frontend because it enables the creation of fast, dynamic, and responsive user interfaces. Its component-based architecture allows developers to build reusable and modular UI elements such as product cards, shopping carts, and checkout forms, improving maintainability and consistency across the platform.
    * Next.js was chosen for BogoGo's frontend because it combines efficiency with familiarity for the development team. Since it builds on React —a framework the team already knows well— it allows faster development and easier collaboration. Its built-in tools for routing and optimization make the website load quickly and provide a smooth experience for users browsing fashion products. Additionally, Next.js makes it simple to deploy and scale the platform on AWS, ensuring that BogoGo can grow without major technical changes in the future.

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 chosen for BogoGo's backend because it is fast, flexible, and familiar to the development team, allowing us to build the platform efficiently. Its event-driven architecture makes it ideal for handling multiple simultaneous requests, such as user logins, product searches, and order processing. Node.js also has a rich 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 help speed up development and simplify common tasks. This familiarity, combined with its ability to scale and integrate seamlessly with AWS, makes Node.js a practical and reliable choice for BogoGo's backend.
        * NestJS was chosen for BogoGo's backend because it combines the speed and flexibility of Node.js with a structured, modular architecture that makes the system easier to maintain and scale. Built with TypeScript, NestJS helps the team write more reliable and organized code while supporting features like dependency injection, modular controllers, and built-in support for REST and GraphQL APIs. It also integrates smoothly with databases (PostgreSQL, MongoDB) and AWS services, while common packages such as TypeORM for database management, Passport for authentication, and class-validator for input validation accelerate development and enforce best practices. This makes NestJS ideal for building a robust, maintainable, and scalable backend for BogoGo.

3. **Infrastructure Layer (Data Storage & External Services):**
    - **Relational Database:** *PostgreSQL* is used as the main database, here is stored the users, products, rates, and all analytical data that it's used within de project.
    - **Object Storage:** For storing digital multimedia files such as PNGs, JPGs, SVGs, and even videos in MP4 format, we propose an S3-based solution within the AWS environment.

    The infrastructure layer combines multiple storage solutions and external services to support the platform's operations. A relational database, PostgreSQL, 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 AWS S3. 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:

- PostgreSQL
  - PostgreSQL was chosen as BogoGo's main database for its reliability, full ACID compliance, and strong analytical capabilities. It not only ensures secure handling of transactions but also supports advanced data analysis, enabling insights into sales, customer behavior, and product performance. This allows BogoGo to centralize operational and analytical data in one scalable system, simplifying data management and supporting real-time, data-driven decisions.
- Amazon S3
  - Amazon S3 is selected as BogoGo's object storage solution for its durability, scalability, and cost-efficiency in managing multimedia content. It supports the platform's need to store and serve large volumes of product images, banners, and videos, while tiered storage options help optimize performance and cost. This makes S3 ideal for handling the visual demands of a fashion-focused e-commerce platform.

- **Data Flow Between Layers:**
  The e-commerce platform follows a three-layer architecture, where each layer communicates with the others, ensuring that only the required information is sent and received.

  Data flow begins at the **Presentation Layer**, where different users interact with the system and issue requests. These requests, which can involve both read and write operations, are sent to the **Application Layer**. Here, the application layer interprets the requests and routes them to the appropriate service. For example, operations related to order payments or product shipments are directed to the corresponding external APIs, and the results are then returned to the presentation layer.

  For requests involving application data, the application layer translates client requests into queries directed to the **Infrastructure Layer**, which encompasses the relational database and object storage services. The infrastructure layer processes these requests, retrieves the required data, and sends it back to the application layer and then to de presentation one. When necessary, it also queries the object storage to retrieve multimedia files or other digital assets, ensuring that all data returned to the client is complete and accurate.

  This structured flow guarantees consistent, reliable, and secure handling of all user interactions, transactions, and data retrieval across the platform.

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;
```

8 Changes from Workshop 1

The context of the project was narrowed down, which allowed for greater precision and understanding, moving away from being a general or vague project. An introduction was added, supporting the project's scope definition and providing a summary of what BogoGo is, the context in which it operates, and the goals of both the project and the workshops. The Canvas model underwent major content changes—initially it was just a list without context. It is now conceived as a comprehensive summary of the project's context, serving as a guide to the business, and giving each section a direct link to the problem space. The functional requirements were rewritten, shifting from a system-centered perspective to a user-centered one, and they were also limited to the actual scope of our project. The non-functional requirements were also refined, aligning them with a deeper theoretical framework that supports the various values presented, rather than being based on arbitrary assumptions. The user stories were reformulated to reflect the new project boundaries and the changes made to both functional and non-functional requirements. Finally, regarding the database architecture, we decided to change much of the structure that was proposed in the previous workshop, as there were elements that were unnecessary for the project and added more complexity. We opted to use a layered architecture that allows us to simplify information transactions for a clearer separation of responsibilities, which helps us to better identify the scope of each main component, also taking into account the time limit for the development of the project. In addition, a justification was added to the architecture explaining why these layers were chosen to solve the problem at hand. A larger image was also added to better visualize the relationships in the E-R model.

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/`