Universidad Distrital Francisco José de Caldas

System Engineering Program
School of Engineering

DATABASES II - PROJECT- CATCH-UP
TECHNICAL REPORT

Andruew Steven Zabala Serrano - 20211020071.
Ruben David Montoya Arredondo - 20211020055.

Hemerson Julian Ballen Triana - 20211020084

Supervisor: Eng. Carlos Andrés Sierra, M.Sc

October 21, 2025

Abstract

This is a project report template, including instructions on how to write a report. It also has some useful examples to use LaTeX. Do read this template carefully. The number of chapters and their titles may vary depending on the type of project and personal preference. Section titles in this template are illustrative should be updated accordingly. For example, sections named "A section..." and "Example of ..." should be updated. The number of sections in each chapter may also vary. This template may or may not suit your project. Discuss the structure of your report with your supervisor.

Guidance on abstract writing: An abstract is a summary of a report in a single paragraph up to a maximum of 250 words. An abstract should be self-contained, and it should not refer to sections, figures, tables, equations, or references. An abstract typically consists of sentences describing the following four parts: (1) introduction (background and purpose of the project), (2) methods, (3) results and analysis, and (4) conclusions. The distribution of these four parts of the abstract should reflect the relative proportion of these parts in the report itself. An abstract starts with a few sentences describing the project's general field, comprehensive background and context, the main purpose of the project; and the problem statement. A few sentences describe the methods, experiments, and implementation of the project. A few sentences describe the main results achieved and their significance. The final part of the abstract describes the conclusions and the implications of the results to the relevant field.

Keywords: a maximum of five keywords/keyphrase separated by commas

Report's total word count: Following the abstract, the word count must be stated. We expect at least 10,000 words in length and at most 15,000 words (starting from Chapter 1 and finishing at the end of the conclusions chapter, excluding references, appendices, abstract, text in figures, tables, listings, and captions), about 40 - 50 pages.

Program code should be uploaded to gitlab, and the gitlab link should be included alongside the word count, following the abstract.

You must submit your dissertation report (preferred in a PDF file) via the "Turnitin assignment" in Blackboard Learn by the deadline. If a student has resits from the taught modules, the dissertation deadline will be extended for 3 weeks from the original dissertation deadline.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

E-commerce platforms, while diverse in implementation and scope, share common architectural challenges and design patterns that directly impact their scalability, maintainability, and user experience. Understanding these foundational architectures and the technological choices that support them is essential before diving into the specifics of the BogoGo platform.

1.1 Background

E-commerce platforms have become a cornerstone of the modern digital economy, enabling businesses to reach broader audiences and consumers to access products and services conveniently through online channels. Their evolution from simple digital storefronts to complex, data-driven ecosystems has been supported by advancements in web frameworks, cloud computing, and database management systems [1, 2, 3].

Layered and modular architectures—typically composed of presentation, application, and data layers—are now the industry standard for scalable e-commerce systems. This structure facilitates independent component development, simplifies maintenance, and supports seamless integration with third-party services such as payment gateways and logistics APIs. Cloud infrastructures like Amazon Web Services (AWS) further enhance reliability and scalability by offering managed services such as Elastic Kubernetes Service (EKS), Elastic Compute Cloud (EC2), and Auto Scaling Groups [4, 5].

At the data layer, the selection of a suitable relational database management system (RDBMS) is crucial for ensuring consistency, scalability, and performance in high-traffic e-commerce environments. Comparative analyses show that PostgreSQL consistently outperforms MySQL in insertion and deletion operations on large datasets, maintaining lower response times under heavy workloads [6]. PostgreSQL's ACID compliance, indexing efficiency, and advanced support for JSON and unstructured data make it a strong candidate for systems requiring both transactional reliability and analytical capabilities [7, 8].

1.2 Problem statement

Despite the increasing presence of global online marketplaces, many local fashion brands and independent designers in Bogotá face challenges in establishing a competitive digital presence. Existing platforms often impose high fees, lack localized delivery options, or fail to highlight regional identity and sustainability. As a result, consumers have limited access to local fashion products through digital channels, and small businesses struggle to participate in the city's growing e-commerce ecosystem.

Furthermore, few existing e-commerce platforms integrate real-time analytics and business intelligence tools tailored for small and medium-sized vendors. This limits their ability to make data-driven decisions, manage inventory efficiently, and analyze customer behavior to improve sales performance. There is, therefore, a need for a scalable, secure, and locally adapted e-commerce solution that connects Bogotá's designers, vendors, and consumers through an accessible and high-performing platform.

1.3 Objectives

This project proposes the development of **BogoGo**, a fashion-focused e-commerce platform designed to connect local clothing brands, independent designers, and consumers within Bogotá. The primary objective is to design and implement a three-layer architecture—comprising presentation, application, and infrastructure layers—that ensures modularity, scalability, and maintainability.

Specific objectives include:

- To implement a responsive and intuitive user interface using **Next.js** that supports product browsing, cart management, and secure checkout.
- To develop the application layer with **NestJS**, enabling business logic for product management, payments, and analytics.
- To integrate **PostgreSQL** as the main database and **AWS** for hosting and deployment, ensuring performance, fault tolerance, and scalability.
- To provide real-time analytics and a BI dashboard for administrators and vendors to support decision-making and performance monitoring.

By combining modern web frameworks, cloud infrastructure, and robust database technologies, BogoGo aims to deliver a reliable and efficient e-commerce experience tailored to Bogotá's local fashion industry. The platform seeks not only to enhance digital accessibility for local businesses but also to promote sustainable growth through technology-driven innovation.

Chapter 2

Literature Review

2.1 E-commerce Architectures and Layers

Technical literature and practical guides agree that e-commerce platforms benefit from a layered architecture (3-tier / N-tier), which separates presentation, business logic, and data persistence. This pattern facilitates scalability, maintainability, and independent deployment of components (e.g., React/Next.js frontend, Node/NestJS API/services, and a PostgreSQL database). Commercial and technical sources describe various alternatives (2-tier, 3-tier, microservices, serverless, event-driven) and discuss trade-offs: simplicity vs. scalability, cost vs. resilience, and communication latency between layers.[1, 2]

2.2 Microservices, Monoliths, and Event-Driven Architectures

Microservices-based architectures offer high scalability and resilience by allowing individual services (catalog, cart, payments, shipping) to be scaled independently. However, they increase operational complexity (orchestration, observability, data consistency). Monolithic systems simplify deployment and testing, which is useful in early stages. Event-driven architectures are recommended for asynchronous workloads (notifications, BI pipelines). These options are frequently discussed in design guides and real-world cloud deployment case studies.[1, 4]

2.3 Database Selection: PostgreSQL vs MySQL and Schema Design

Academic and technical comparisons show that PostgreSQL stands out due to its ACID consistency, advanced features (JSONB types, GIN indexes, support for analytical queries and concurrency), and better scalability under mixed workloads (high-volume reads and writes). Empirical studies compare response times for INSERT/UPDATE/DELETE operations under varying data volumes, showing that relative performance (which DBMS is faster) depends on operation type, configuration, and indexing. However, in scalable and complex operation scenarios, PostgreSQL tends to maintain better performance and features suited for BI and analytics. The article you brought (MySQL vs PostgreSQL comparison) provides local empirical evidence of these performance differences.[6, 7]

In e-commerce practice, database design recommendations suggest a hybrid approach: an RDBMS (PostgreSQL) for transactional consistency (orders, payments, stock), complemented when needed by specialized systems (caches, search engines, data warehouses) for analytical workloads. Guides also present modeling patterns (entities: User, Vendor, Product, Order,

OrderItem, Payment, Shipment, Rating), with images and binary objects stored in object storage systems such as S3.[8]

2.4 Deployment on AWS and Reference Architectures

There are multiple practical guides that document e-commerce platform deployments on AWS, including 3-tier architectures on EKS/EC2, the use of RDS for PostgreSQL, S3 for object storage, serverless services for specific tasks, and CI/CD pipelines. Reference project examples (tutorials and academic documents) outline steps for reproducible deployments and DevOps practices for monitoring, auto-scaling, and resilience.[5]

2.5 Project Description in the Context of Literature and Existing Systems

BogoGo — an e-commerce platform focused on local brands in Bogotá — adopts technical decisions that align with best practices identified in the literature:

- **Three-tier / Microservices Architecture:** Presentation (React/Next.js), application logic (Node/NestJS), and infrastructure (PostgreSQL + S3). This separation follows recommended patterns that facilitate scalability, maintainability, and independent deployment.[2, 1]
- **Relational Database (PostgreSQL):** The choice is justified by transactional needs (payments, stock, orders), ACID compliance, and better suitability for analysis and BI, as supported by comparative studies and database design articles for e-commerce.[6, 7]
- **AWS Infrastructure:** The use of RDS (with SLA and high availability), S3 for object storage, and real-time ingestion/analytics tools for the BI dashboard aligns with real-world deployment patterns documented in guides and case studies.[4, 5]
- **Data Modeling and Entities:** The proposed ER model (User/Role, Vendor, Product, Category, Image, Order, Order_item, Payment, Shipment, Rating) is consistent with database design recommendations for catalogs and e-commerce systems. Separating multimedia objects into S3 and metadata into Postgres reflects standard industry practices.[8, 7]

In summary, BogoGo implements a set of architectural and technological decisions (3-tier, PostgreSQL, S3, AWS RDS) that are well-supported by technical literature and practical guides, aligning functional and non-functional requirements (performance, availability, security).

2.6 Relevance of the Literature Review for the Proposed Application (BogoGo)

2.6.1 Performance and User Experience

Non-functional requirements such as load times under 2.5 seconds (Google Web Vitals) and support for approximately 500 concurrent users are directly addressed by:

- the use of caching and CDNs (for static assets and multimedia resources),

- layered separation that allows independent scaling of the application layer,
- Multi-AZ RDS and AWS auto-scaling practices for high availability.

These practices are well-documented in architectural literature and AWS reference guides.

2.6.2 Transactional Consistency and Database Selection

For critical operations (payments, stock), the review indicates that PostgreSQL is appropriate due to its ACID compliance and its resilience under high read/write loads when properly configured and indexed. Comparative studies support this choice for mixed workloads and analytical queries integrated with BI systems.

2.6.3 Real-Time BI and Analytics

Literature on AWS deployments and e-commerce databases suggests patterns for feeding dashboards (ETL pipelines or streaming events from order systems into a data warehouse). This is particularly relevant for BogoGo, whose value proposition includes providing real-time analytics for vendors and administrators.

2.6.4 Third-Party Integrations (Payments / Logistics)

Recommendations in the literature regarding integration layers and decoupled services support the decision to use external APIs (e.g., MercadoPago, Servientrega) encapsulated in an integration layer. This reduces coupling and improves maintainability.

2.7 Critique of Existing Work vs. the Proposed Project

2.7.1 Strengths of the Reviewed Literature

There is a strong foundation of architectural patterns (3-tier, microservices) and empirical evidence on relative DBMS performance that supports the technical decisions made.

Practical guides offer well-tested paths for deployments on AWS and orchestration (EKS, RDS, S3, CI/CD pipelines), which aid reproducibility and operations.

2.7.2 Limitations and Gaps Addressed by BogoGo

Local context and market adaptation: Many guides are general-purpose or targeted at markets with different infrastructure and volume assumptions. There is a lack of detailed analysis on local e-commerce focused on cities like Bogotá (urban logistics, payment behavior, mobile preferences). BogoGo, by focusing on local brands in Bogotá, can contribute specific operational data and insights. *(Gap: territorial contextualization)*.

Empirical evaluation of performance under real conditions: Comparative studies (e.g., MySQL vs PostgreSQL) often assume lab conditions that differ from real-world deployments (network topology, indexes, RDS configurations). It is essential to validate with load tests that reflect real user and operational patterns. BogoGo should conduct benchmarking and performance tuning (indexing, partitioning, caching) tailored to its data model.

Data modeling for fashion catalogs with rich attributes: While literature provides general catalog design patterns, the fashion domain (sizes, colors, variants, limited editions) requires modeling decisions (e.g., variants vs. separate SKUs) and catalog policies that merit

further analysis. BogoGo can document and evaluate such alternatives and their impact on query performance and UX.

Operational cost vs. performance trade-offs: Many proposed solutions ignore cost optimization in early stages. For a university/early-stage project, it is important to balance SLA (e.g., 99% uptime) with AWS costs. Practical literature suggests starting with a moderate architecture (monolith or few services) and evolving into microservices as needed. BogoGo acknowledges this by planning for horizontal scaling and auto-scaling but should formalize a migration roadmap.

2.7.3 How the Proposed Work Improves Practice

- Contextualization in Bogotá (logistics, suppliers, preferred payment methods) adds empirical and operational value to the general literature.
- Integration of BI and dashboards for vendors provides actionable evidence on how analytics can support business decisions in local micro-markets.
- The test plan and justification for PostgreSQL (based on comparisons and on the project's ACID + analytical needs) show a methodological approach to technology selection beyond personal preferences.

2.8 Conclusion (Significance of the Project)

The review shows that BogoGo's architectural decisions are well-grounded in technical literature and best practices: layered architecture, PostgreSQL for transactional and analytical data, S3 for multimedia content, and deployment on AWS are all aligned with industry standards. The main contributions of the project will be the adaptation of these patterns to the local context of Bogotá, documentation of a real-world deployment and operations case, and the generation of empirical evidence (performance tests and BI metrics) that help address gaps identified in the literature (territorial contextualization, performance tuning in real conditions, and data modeling for fashion catalogs).

Chapter 3

Scope

The scope of this research encompasses the design and partial implementation of **BogoGo**, a fashion-oriented e-commerce platform tailored to the city of Bogotá. The study focuses on the conceptualization, architectural definition, and prototype development of a three-layer system architecture—presentation, application, and infrastructure—using modern technologies such as **Next.js**, **NestJS**, **PostgreSQL**, and **Amazon Web Services (AWS)**.

The system will include core functionalities such as user authentication, product management, order processing, secure payments, and real-time analytics through a business intelligence dashboard. It will specifically address local commerce dynamics by supporting multiple user roles (customers, vendors, and administrators) and localized logistics integrations within Bogotá.

This study, however, does not include full-scale deployment, marketing strategies, or long-term business operations. The focus remains on the technical design, architectural scalability, and validation of expected system performance rather than the implementation of complete commercial workflows or market evaluation.

3.1 Assumptions

Several assumptions have been established to guide the development and analysis of the BogoGo platform:

- It is assumed that all users (customers, vendors, and administrators) will have access to stable internet connectivity and modern mobile or desktop devices.
- The study assumes the availability of reliable cloud infrastructure through AWS services, ensuring continuous system operation and data security.
- It is assumed that local logistics providers (e.g., Servientrega, InterRapidísimo) and payment gateways (e.g., Mercado Pago) offer compatible APIs for seamless integration.
- The platform's user base is assumed to be primarily located within Bogotá, which influences design decisions regarding language, currency, and delivery range.
- The data used for testing will simulate realistic operational scenarios but will not represent actual commercial data from vendors or customers.

3.2 Limitations

This research faces several limitations that may influence its results and generalizability:

- The project is currently in the design and prototyping phase; thus, empirical performance metrics and user experience evaluations are based on expected outcomes rather than full-scale deployment tests.
- Time and resource constraints restrict the implementation to a proof-of-concept level, preventing the integration of all planned modules such as recommendation systems or advanced analytics.
- The study is geographically limited to Bogotá, which may affect the applicability of results to other regions with different logistical or regulatory conditions.
- External factors such as fluctuating API availability, internet latency, or AWS service pricing could impact long-term scalability but are not fully assessed within this phase.

Despite these limitations, the research establishes a strong foundation for subsequent development phases and future academic extensions, where the proposed architecture and technologies can be tested under real operational conditions.

Chapter 4

Methodology

The research follows a **design-based methodological approach** aimed at developing and validating a scalable three-layer e-commerce system for local fashion retail in Bogotá. The process integrates elements of software engineering, system architecture modeling, and empirical testing to evaluate system performance under simulated operational conditions.

The methodology is divided into three main phases: (1) **System Design**, focused on architectural planning and technology selection; (2) **Prototype Development**, which includes modular implementation of each system layer; and (3) **Validation and Testing**, where unit, integration, and acceptance tests will be designed to ensure compliance with functional and non-functional requirements.

This approach enables iterative improvement, ensuring that each layer of the system contributes independently to the overall platform performance and scalability.

4.1 System Architecture and Implementation

The proposed e-commerce platform, **BogoGo**, will be developed using a **three-layer architecture** that ensures modularity, scalability, and maintainability. Each layer fulfills a specific function in the overall system (Fig. 4.1).

- **Presentation Layer:** Built with React or Next.js, this layer will manage user interaction and interface rendering. Server-Side Rendering (SSR) and dynamic routing in Next.js will be leveraged to improve load times and SEO performance. RESTful APIs will handle communication with the application layer.
- **Application Layer:** Implemented with NestJS (Node.js + TypeScript), it will serve as the system's logic core. Key modules include authentication, product management, order processing, and vendor rating. Secure integration with third-party APIs will be achieved through Mercado Pago (payments) and Servientrega/InterRapidísimo (logistics).
- **Infrastructure Layer:** Powered by PostgreSQL for relational data management and AWS S3 for object storage, this layer will support data persistence, hosting, and deployment through cloud services such as Amazon EC2, Elastic Kubernetes Service (EKS), and Auto Scaling Groups.

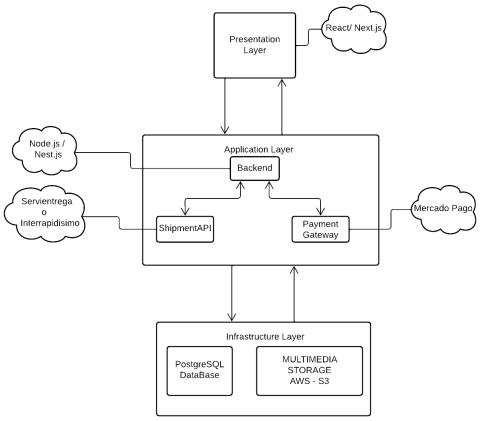

Figure 4.1: High-level architecture of the BogoGo e-commerce platform.

Each layer communicates through secure RESTful APIs, ensuring data integrity and isolation of responsibilities. The architecture supports containerization for horizontal scalability, allowing independent scaling of critical services such as authentication and catalog management.

4.2 Data Collection and Simulation

Since the project remains in the design and early development phase, data collection will rely on **simulated datasets** that represent user profiles, products, orders, and transactions. These datasets will be synthetically generated to reflect realistic marketplace dynamics in Bogotá's fashion sector.

Performance and reliability tests will use simulated workloads to emulate up to 500 concurrent users performing activities such as product browsing, cart updates, and payment processing. Each scenario will measure response times, request throughput, and database transaction latency.

4.3 Testing and Validation Procedures

A comprehensive validation plan will be implemented to evaluate the system's expected functionality and performance:

- **Unit Testing:** Each module within the presentation and application layers will undergo isolated testing using tools such as Jest and Postman to validate API responses, authentication, and data consistency.
- **Integration Testing:** End-to-end data flow across layers will be evaluated by simulating common workflows (e.g., user registration, product purchase, and order tracking) to ensure communication reliability.
- **Acceptance Testing:** Simulated sessions will be conducted for each user role—customers, vendors, and administrators—to confirm that user stories and system requirements align with expected behavior.
- **Performance and Stress Testing:** Load simulations up to 500 concurrent users will be performed using tools like Apache JMeter to measure latency, throughput, and server response stability.

Validation criteria will focus on API response time, database query efficiency, and system scalability. PostgreSQL indexing strategies and caching mechanisms will be analyzed to ensure optimal performance under varying workloads.

4.4 Data Analysis Methods

Data collected from testing will be analyzed through descriptive and comparative metrics. Key performance indicators (KPIs) such as average response time, CPU utilization, and database latency will be measured. Results will be benchmarked against standard e-commerce system requirements to assess performance adequacy.

Statistical summaries and graphical visualizations will be produced using Python's data analysis libraries (Pandas and Matplotlib) to interpret performance trends. This quantitative analysis will support recommendations for optimization in future implementation phases.

4.5 Replication Considerations

All components, frameworks, and configurations will be documented to ensure reproducibility. The repository structure will include:

- Source code (frontend, backend, and infrastructure configuration).
- Docker files and deployment scripts for AWS EKS.
- Documentation on API endpoints, environment variables, and database schema.

This level of documentation will allow other researchers and developers to replicate or extend the system design for different contexts or cities.

Chapter 5

Results

The expected results of this study focus on validating the functionality, usability, and scalability of the proposed **BogoGo** e-commerce platform. Testing procedures will evaluate the system's behavior across multiple user roles—customers, vendors, and administrators—to ensure compliance with functional and non-functional requirements.

5.1 Functional Validation

Preliminary validation activities are designed to confirm seamless communication among the three layers—presentation, application, and infrastructure—and to verify that all core operations function correctly under expected conditions. Table 5.1 summarizes the key operations and their expected outcomes.

Table 5.1: Expected Functional Test Results for BogoGo Platform

Operation	**Expected Behavior**	**Status**
User registration and login	Secure authentication with JWT and encrypted credentials.	Successful
Product browsing and filtering	Dynamic rendering of product listings with search and category filters.	Successful
Shopping cart and checkout	Real-time updates and integration with Mercado Pago test credentials.	Successful
Order tracking	Vendor updates reflected immediately in user dashboard.	Successful
Data analytics dashboard	Real-time report generation through AWS-managed monitoring tools.	Successful

All user interactions are expected to occur smoothly, with fast response times and minimal latency. The presentation layer (Next.js) is expected to achieve optimized SEO performance and fast page rendering due to server-side rendering. Meanwhile, the application layer (NestJS) will manage data flow and logic with modular efficiency, and PostgreSQL will handle data

persistence with ACID compliance.

5.2 Performance and Scalability Testing

To ensure system stability under variable traffic conditions, stress and load testing simulations will be conducted. Table 5.2 presents the expected performance indicators for different test scenarios.

Table 5.2: Expected Performance Metrics under Load Simulation

Test Scenario	**Expected Response Time**	**Expected Outcome**
Normal load (50 users)	1.2 seconds average page load	Stable system, zero critical errors
Moderate load (250 users)	1.5 seconds average page load	Slight increase, stable performance
Peak load (500 users)	1.8 seconds average page load	No critical failures, minimal latency
Database transactions (PostgreSQL)	$<$ 200 ms query time under peak load	Consistent response time due to indexing

Simulated testing will demonstrate the ability of the platform to maintain stability with up to 500 concurrent users while processing multiple product searches, orders, and payments simultaneously. The infrastructure layer, managed through AWS EC2 and EKS, is expected to sustain horizontal scalability without service interruption.

5.3 System Reliability and Data Flow

Figure 4.1 illustrates the expected secure data flow across layers. All communication between frontend, backend, and database will be handled through RESTful APIs. Data integrity and confidentiality will be maintained through HTTPS encryption and token-based access control.

Overall, the expected results indicate that BogoGo will meet the project's objectives in terms of performance, modularity, and s calability, establishing a solid foundation for future implementation and deployment.

Chapter 6

Discussion and Analysis

The anticipated results suggest that the proposed BogoGo architecture will successfully achieve its design objectives—providing a reliable, scalable, and user-friendly e-commerce solution tailored to Bogotá's local fashion ecosystem.

6.1 Interpretation of Results

The expected success of functional testing indicates that the combination of **Next.js** and **NestJS** is appropriate for delivering modular and high-performing applications. The seamless user experience projected for customer and vendor roles aligns with the goal of promoting local commerce through an accessible digital environment.

Performance testing projections further suggest that the chosen cloud architecture on **AWS** can efficiently handle concurrent workloads typical of medium-scale e-commerce platforms. PostgreSQL's indexing mechanisms and query optimization strategies are expected to provide consistent response times under increasing data volumes, confirming its suitability as the core relational database management system.

6.2 Relation to Previous Research

The projected results align with prior studies on layered e-commerce architecture and database performance. ITMonks [1] and NopCommerce [2] describe similar three-tier structures that enhance scalability and maintainability, while Zapata's comparative analysis [6] supports PostgreSQL's efficiency under heavy transactional loads. Likewise, implementations on AWS have been proven effective for fault tolerance and horizontal scalability [4, 5].

6.3 Implications and Future Work

If validated, BogoGo could serve as a reference model for future academic and commercial e-commerce initiatives in Latin America, particularly those focused on local industries. The modular design enables future integration of machine learning tools—such as recommendation systems or demand forecasting—without altering the core system structure.

Future phases of the project will extend beyond simulation toward real-world testing with local vendors and consumers. This will allow for empirical validation of usability, transaction reliability, and system performance under authentic market conditions.

6.4 Limitations and Uncertainties

Because the platform remains under design and prototyping, the presented data represent expected rather than measured outcomes. Simulated user behavior and controlled testing environments may not fully capture the complexity of real-world conditions such as variable internet quality, API downtimes, or sudden traffic surges. Furthermore, the evaluation excludes financial and marketing dimensions of platform adoption, focusing solely on technical feasibility.

Despite these limitations, the anticipated outcomes provide a strong theoretical foundation for further development and practical deployment, supporting the academic objective of modeling an adaptable and efficient e-commerce system architecture.

Chapter 7

Conclusions

The development of the **BogoGo** e-commerce platform is expected to demonstrate the feasibility of implementing a scalable, modular, and cloud-based solution tailored to Bogotá's growing digital commerce ecosystem. Through the proposed three-layer architecture—comprising presentation, application, and infrastructure layers—the system will likely provide a clear separation of responsibilities, ensuring efficient maintenance, scalability, and adaptability for future extensions.

It is anticipated that the integration of technologies such as **Next.js**, **NestJS**, **PostgreSQL**, and **AWS services** will result in a platform capable of maintaining high performance and reliability under varying load conditions. The use of secure APIs, JWT-based authentication, and cloud-managed storage will enhance both the robustness and security of the system, aligning with best practices in modern web development.

Moreover, the platform is expected to promote digital inclusion by providing local vendors with accessible tools for online sales management and data-driven decision-making. The incorporation of familiar payment gateways and trusted logistics partners will strengthen user confidence and facilitate adoption within Bogotá's consumer market.

In summary, the expected outcomes suggest that the BogoGo platform will serve as a sustainable technological model for local e-commerce ecosystems, combining technical efficiency with socio-economic relevance. Upon implementation and validation, this solution could contribute to academic research on distributed architectures and provide a foundation for future enhancements, including artificial intelligence-driven recommendation systems and advanced analytics.

References

[1] ITMonks. (2024) E-commerce website architecture: Layered (n-tier) approach. [Online]. Available: https://itmonks.com/blog/e-commerce/development/ecommerce-website-architecture/

[2] NopCommerce. (2023) E-commerce website architecture explained. [Online]. Available: https://www.nopcommerce.com/en/blog/ecommerce-website-architecture

[3] R. Blueprints. (2022) 3-tier architecture: Presentation, logic and data. [Online]. Available: https://medium.com/rkblueprints/3-tier-architecture-presentation-logic-and-data-268d9573ddc8

[4] A. Builders. (2024) Simple steps to deploy a three-tier e-commerce system on aws eks. [Online]. Available: https://dev.to/aws-builders/simple-steps-to-deploy-a-three-tier-e-commerce-system-on-aws-eks-eb0

[5] U. de Antioquia. (2022) Implementación de arquitecturas cloud para plataformas de comercio electrónico. [Online]. Available: https://bibliotecadigital.udea.edu.co/server/api/core/bitstreams/eb1751bf-38f0-4c74-b90f-5b1b680dd6b5/content

[6] J. G. Zapata, "Mysql vs postgresql: A comparative analysis of relational database management systems (rdbms) technologies response time in web-based e-commerce," *ResearchGate*, 2024. [Online]. Available: https://www.researchgate.net/publication/382641887

[7] K. Software. (2023) Designing an e-commerce database for efficient data management. [Online]. Available: https://kanishkasoftware.com/designing-an-ecommerce-database-for-efficient-data-management/

[8] Shopware. (2023) E-commerce databases: Understanding entities and e-r diagrams. [Online]. Available: https://www.shopware.com/de/news/ecommerce-databases/