

# **Databases II**

Semester 2025-III

Workshop No. 1 — Project Definition and Database Modeling

**Julian Camilo Espinosa Morales**  
Cod. 20191020073

**Pablo Alejandro Montaña Moreno**  
Cod. 20201020090

**Juan Carlos Duarte Sandoval**  
Cod. 20212020149

# Workshop Scope and Objectives

## Business Model Canvas

### 1 Project Summary

A multi-module web platform for short-term stays and experiences around Colombia (similar to Airbnb). The architecture is based on independent microservices, which can be implemented in different programming languages. The system integrates both relational storage for structured data (users, bookings, payments) and non-relational/object storage for unstructured large media (photos, videos).

The platform supports real-time data ingestion, fast query execution, recommendation engines, business intelligence dashboards, multi-region access, and high availability.

### 2 Business Model Canvas

<b>Key Partners</b>  <b>Payment processors and gateways (Stripe, PayU, Wompi): handle secure transactions, refunds, and PCI compliance.</b>  <b>Local tourism offices and travel agencies:</b> provide verified local experiences and increase platform credibility.  <b>CDN and object-storage providers (AWS S3, Cloudflare):</b> ensure fast and reliable delivery of photos and videos.  <b>Cleaning, verification, and property-inspection services:</b> maintain quality and trust of listings.  <b>Technology partners (search engines, recommendation algorithms, analytics tools):</b> support innovation and scalability.  	<b>Key Activities</b>  Operating the web/mobile platform Host verification and trust/safety checks Managing bookings, payments, and refunds Media upload, transcoding, and delivery Developing recommendation  	<b>Value Propositions</b>  Localized and verified listings covering Colombian cities, rural stays, and unique experiences not found on global platforms. Secure, transparent booking and payment flows that reduce fraud and build user trust. Personalized recommendations and smart search based on traveler preferences, location, and seasonality.  Modular microservice architecture enabling rapid feature development by independent teams and easier scalability. High-quality media delivery so guests can confidently view listings with detailed photos and videos.  	<b>Customer Relationships</b>  Self-service portal for travelers and hosts 24/7 support chat Trust & safety verification and reviews system Automated processes to keep costs low while maintaining reliability while ensuring reliability.  	<b>Customer Segments</b>  Travelers (domestic and international) looking for trustworthy, localized stays and experiences. Hosts (property owners) seeking to monetize their spaces with secure and easy-to-use tools. Local experience/activity providers offering tours, workshops, or services complementary to stays.  
<b>Cost Structure</b>  <b>Infrastructure costs:</b> hosting databases, storage, CDN, security, and monitoring. <b>Engineering and development:</b> salaries of developers, designers, DevOps, and data scientists. <b>Customer support operations:</b> 24/7 chat, verification team, dispute handling. <b>Payment gateway fees and chargebacks for processing transactions.</b> <b>Marketing and partnerships:</b> campaigns, affiliate payouts, and tourism board collaborations.  	<b>Revenue Streams</b>  <b>Booking fees:</b> percentage-based commission on each reservation from travelers or hosts. <b>Host subscriptions or premium listings:</b> monthly or annual plans offering higher visibility and extra features. <b>Featured placement and advertising:</b> paid options for hosts or local businesses to appear in top search results. <b>Ancillary services:</b> partnerships with travel insurance, transport providers, or experiences generating referral income.  			

Figure 1: Workshop canvas

## **2.1 Value Propositions**

- Localized, verified listings for Colombia (cities, rural stays, unique experiences).
- Secure booking and payment flows.
- Recommendations tailored to traveler preferences and location.
- Modular platform enabling rapid feature development by different teams.

## **2.2 Customer Segments**

- Travelers (domestic & international).
- Hosts (property owners).
- Local experience providers.
- Admins and managers.

## **2.3 Channels**

- Web application.
- Mobile web.
- Partner integrations (travel agencies).
- Social media.
- Local tourism boards.

## **2.4 Customer Relationships**

- Self-service portal.
- 24/7 support chat.
- Trust & safety verification.
- Reviews.

## **2.5 Revenue Streams**

- Booking fees (percentage-based).
- Host subscription/premium listings.
- Featured placement.
- Advertising.

## **2.6 Key Activities**

- Platform operation.
- Host verification.
- Payments.
- Recommendations.
- Marketing.

## **2.7 Key Resources**

- Multi-region database infrastructure.
- Object storage.
- Microservices.
- Machine learning models for recommendations.

## **2.8 Key Partners**

- Payment processors.
- Local tourism offices.
- CDN/object-storage providers.
- Cleaning/verification services.

## **2.9 Cost Structure**

- Infrastructure (databases, storage, CDN).
- Engineering.
- Customer support.
- Payment fees.

# **3 Requirements Documentation**

## **3.1 Functional Requirements (selected & prioritized)**

- User registration & login (email/social auth, roles: guest/host/admin).
- Listing CRUD for hosts (create listing, upload photos/videos, set availability & price).
- Search & filters by location, date, guests, price, amenities.

- Booking/reservation flow with atomic availability checks and payment integration.
- Payments & refunds via external payment gateway (PCI compliant).
- Reviews & ratings for hosts and guests.
- Media management: upload, transcode, store photos/videos, CDN delivery.
- Recommendation engine: personalized suggested listings.
- Admin dashboard & BI: occupancy, revenue, fraud alerts, KPIs.
- Notifications: email/SMS/push for booking confirmations, reminders.
- Multi-region deployment (Colombia + optional international).
- Audit & logging for compliance and debugging.

### 3.2 Non-Functional Requirements

- **Performance:** Search queries return first page within 300 ms for common queries; booking availability check <200 ms for single listing.
- **Scalability:** Horizontal scaling for services; DB partitioning/sharding for large tables.
- **Availability:** 99.95% SLA for booking & payment services; failover across regions.
- **Consistency:** Strong consistency for booking/availability; eventual consistency allowed for analytics and recommendations.
- **Security:** OAuth2/JWT, password hashing, input sanitization, PCI-DSS compliance for payments, data encryption at rest & in transit.
- **Maintainability:** Microservice architecture; clear API contracts; CI/CD pipelines.
- **Compliance:** Data residency & local laws (Colombia), GDPR-style user controls if international.

Role	User Story	Acceptance Criteria
------	------------	---------------------

Guest - Search & Book	As a guest, I want to search for listings by city and date so that I can book a stay.	<ul style="list-style-type: none"> <li>Search returns listings matching city, date availability, and guest count.</li> <li>Results include price, ratings, thumbnail image, and booking button.</li> <li>Booking flow prevents double-booking and completes payment via gateway.</li> </ul>
Host - Create Listing	As a host, I want to create a listing with photos and availability so I can receive bookings.	<ul style="list-style-type: none"> <li>Host can create listing with title, description, price, amenities, calendar.</li> <li>Host can upload photos/videos; upload returns URLs and thumbnails.</li> <li>Calendar prevents overlapping availability ranges.</li> </ul>
User - Authentication	As a user, I want to log in using email/password or social auth so I don't need to remember another account.	<ul style="list-style-type: none"> <li>Sign up verifies email; social sign in creates user record with provider.</li> <li>Tokens (JWT) are issued with appropriate expiration and refresh flow.</li> </ul>
Admin - Analytics Dashboard	As an admin, I want to see monthly revenue and occupancy so I can evaluate business health.	<ul style="list-style-type: none"> <li>Dashboard shows revenue, bookings, top cities; data updated daily.</li> <li>Admin can filter by date range and city.</li> <li>Data queries for dashboard complete under 5 seconds.</li> </ul>

System - Media Delivery	As a system, I want media stored in an object store and served via CDN so page load is fast.	<ul style="list-style-type: none"> <li>Photos/videos are uploaded to object storage; metadata saved in DB.</li> <li>Media URLs served via CDN; thumbnails generated automatically.</li> <li>Large videos support resumable uploads and background transcoding.</li> </ul>
Guest - Review Host	As a guest, I want to leave a review after my stay so others can benefit.	<ul style="list-style-type: none"> <li>Guest can post one review per completed booking.</li> <li>Reviews are attached to listing and host profile; average rating updates.</li> </ul>

## 4 Initial Database Architecture

### 4.1 High-level Database Architecture

The proposed architecture is an hybrid architecture which combines the Microservices Architecture with an Distributed Architecture. This integrates multiple storage and processing technologies to balance transactional integrity, scalability, and analytical needs:

- **Relational DB (OLTP):** PostgreSQL (primary) for core transactional data requiring ACID properties such as users, listings metadata, bookings, payments, reviews, and availability. *Reason: ensures transactional integrity (e.g., bookings) and strong consistency for reservations.*
- **Object Storage (S3-compatible):** Used to store raw photos and videos, generating links for CDN delivery.
- **NoSQL / Document Store:** MongoDB or DynamoDB for flexible media metadata, search facets, denormalized listing views, and session data.
- **Search Engine:** Elasticsearch / OpenSearch to support full-text search and fast faceted queries.
- **Cache:** Redis for hot lookups such as user sessions, rate limiting, and short-term availability caching.

- **Event Streaming:** Apache Kafka for ingestion of events (uploads, bookings) and to feed Business Intelligence (BI) and recommendation pipelines.
- **Data Lake / Analytics:** Parquet files stored on object storage, processed by Apache Spark for BI and Machine Learning (recommendations).
- **BI Layer:** Precomputed aggregates stored in a columnar store (e.g., ClickHouse), or served via a star-schema OLAP warehouse (e.g., BigQuery, Snowflake, or PostgreSQL + cubes).

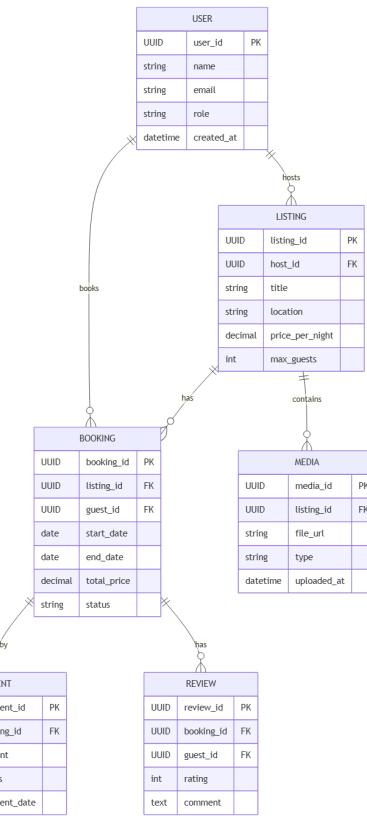


Figure 2: ER Diagram Model

## 5 References

### References

- [1] Corporate Finance Institute. (n.d.). Business model canvas examples. Retrieved from <https://corporatefinanceinstitute.com/resources/>

[management/business-model-canvas-examples/](https://management/business-model-canvas-examples/)

- [2] PostgreSQL Global Development Group. (n.d.). PostgreSQL: The world's most advanced open source database. Retrieved from <https://www.postgresql.org/>
- [3] MongoDB, Inc. (n.d.). MongoDB — The application data platform. Retrieved from <https://www.mongodb.com/>
- [4] Redis Ltd. (n.d.). Redis — In-memory data store & cache. Retrieved from <https://redis.io/>
- [5] The Apache Software Foundation. (n.d.). Apache Kafka — Distributed event streaming platform. Retrieved from <https://kafka.apache.org/>
- [6] Amazon Web Services. (n.d.). Amazon Simple Storage Service (S3). Retrieved from <https://aws.amazon.com/s3/>
- [7] Elastic N.V. (n.d.). Elasticsearch — The official search & analytics engine. Retrieved from <https://www.elastic.co/elasticsearch/>
- [8] Snowflake Inc. (n.d.). Snowflake — Data cloud. Retrieved from <https://www.snowflake.com/>
- [9] Google Cloud. (n.d.). BigQuery — Serverless, highly scalable data warehouse. Retrieved from <https://cloud.google.com/bigquery>