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

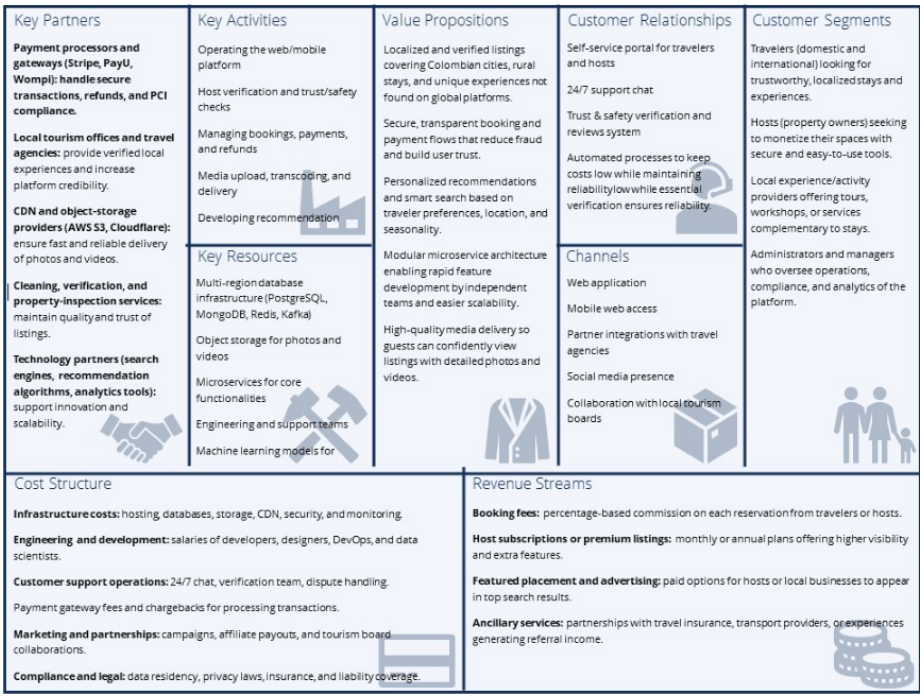

Figure 1: Workshop canva

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.	• Search returns listings matching city, date availability, and guest count. • Results include price, ratings, thumbnail image, and booking button. • Booking flow prevents double-booking and completes payment via gateway.
Host - Create Listing	As a host, I want to create a listing with photos and availability so I can receive bookings.	• Host can create listing with title, description, price, amenities, calendar. • Host can upload photos/videos; upload returns URLs and thumbnails. • Calendar prevents overlapping availability ranges.
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.	• Sign up verifies email; social sign in creates user record with provider. • Tokens (JWT) are issued with appropriate expiration and refresh flow.
Admin - Analytics Dashboard	As an admin, I want to see monthly revenue and occupancy so I can evaluate business health.	• Dashboard shows revenue, bookings, top cities; data updated daily. • Admin can filter by date range and city. • Data queries for dashboard complete under 5 seconds.

System - Media Delivery	As a system, I want media stored in an object store and served via CDN so page load is fast.	• Photos/videos are uploaded to object storage; metadata saved in DB. • Media URLs served via CDN; thumbnails generated automatically. • Large videos support resumable uploads and background transcoding.
Guest - Review Host	As a guest, I want to leave a review after my stay so others can benefit.	• Guest can post one review per completed booking. • Reviews are attached to listing and host profile; average rating updates.

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).

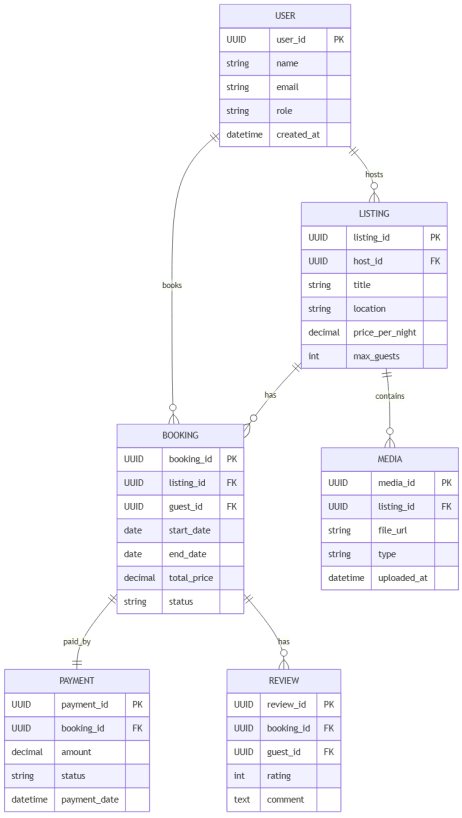

Figure 2: ER Diagram Model

5 Information Requirements

The system must retrieve several key types of information to support the main business operations, decision-making, and user interactions. These requirements are directly linked to the user stories defined for guests, hosts, administrators, and the system itself.

Booking and Availability Information: The platform must retrieve available listings based on city, date range, and number of guests. This information ensures accurate search results and prevents double bookings, supporting the core experience for travelers.

Host and Listing Data: The system must display and manage listings, including titles, descriptions, prices, and media. Hosts can access their listings, update availability, and review booking performance.

User Authentication and Profiles: Information related to user accounts, including email, name, and role (guest, host, or admin), must be retrievable to enable login, personalized dashboards, and access control.

Reviews and Ratings: The platform must retrieve and display reviews linked to specific listings and hosts, providing reputation metrics that influence booking decisions and customer trust.

Payments and Financial Reports: The system must access detailed payment data such as booking amounts, payment dates, and status (pending, completed, refunded). Administrators require aggregated financial information, including total revenue per month, commission income, and payment success rates, to evaluate business performance and detect anomalies.

Media and Performance Metrics: Information about uploaded photos and videos, including their URLs and upload times, must be retrieved for media management. Additionally, the system should collect analytics data such as most-viewed listings and popular destinations to improve recommendations and marketing strategies.

These information requirements support the platform's core objectives defined in the business model and user stories: ensuring reliable booking processes, transparent financial tracking, and personalized user experiences while maintaining performance and scalability.

6 Query Proposals

The proposed platform integrates both relational and non-relational databases to support different use cases:

- **Relational (PostgreSQL)**: transactional data consistency (users, listings, bookings, payments, reviews).
- **NoSQL (MongoDB)**: fast document retrieval and denormalized views for search, recommendations, and caching.

6.1 Relational Database Queries (PostgreSQL)

Query 1 — Retrieve the most popular listings by city
Purpose: Identify the top listings in each city based on the number of bookings.
Business goal: Helps the platform recommend top-rated or most-booked stays to users.

```
SELECT
    l.city,
    l.title AS listing_name,
    COUNT(b.booking_id) AS total_bookings
FROM listings l
JOIN bookings b ON l.listing_id = b.listing_id
-- Maybe implementing another states list
WHEre b.status = 'confirmed'
GROUP BY l.city, l.title
ORDER BY l.city, total_bookings DESC
LIMIT 10;
```

Insight: Provides marketing and analytics teams with data on trending destinations in order gather information and improving the service.

Query 2 — Calculate total earnings per host
Purpose: Determine total revenue generated by each host through completed bookings.
Business goal: Enables hosts to visualize their performance and supports payout operations.

```
SELECT
    u.name AS host_name,
    SUM(p.amount) AS total_earnings
FROM users u
JOIN listings l ON u.user_id = l.host_id
JOIN bookings b ON l.listing_id = b.listing_id
JOIN payments p ON b.booking_id = p.booking_id
WHERE p.status = 'completed'
GROUP BY u.name
ORDER BY total_earnings DESC;
```

Insight: Shows which hosts contribute the most revenue and supports their financial reporting, something that might help hosts a lot with finantial info.

Query 3 — Get guest booking history
Purpose: Retrieve all past and current bookings for a specific user.
Business goal: Supports usser dashboards and customer service operations.

```
SELECT
    b.booking_id,
    l.title AS listing,
    b.start_date,
    b.end_date,
    b.status,
```

```
    p.amount AS payment_amount
FROM bookings b
JOIN listings l ON b.listing_id = l.listing_id
LEFT JOIN payments p ON b.booking_id = p.booking_id
WHERE b.guest_id = 'exampleUserUid'
ORDER BY b.start_date DESC;
```

Insight: Provides personalized views of guest activity.

6.2 Non-Relational Database Queries (MongoDB)

Query 1 — Retrieve denormalized listing document
Purpose: Fetch a full listing document with host, location, and reviews embedded.
Business goal: Enables fast rendering of listing pages with minimal joins.

```
db.listings.find(
  { "listing_id": "eexamplelist:)" },
  {
    title: 1,
    location: 1,
    price_per_night: 1,
    "host.name": 1,
    "reviews.comment": 1,
    "reviews.rating": 1,
    "media.file_url": 1
  }
);
```

Insight: Supports high-performance reads in the web frontend.

Query 2 — Find top-rated listings
Purpose: Retrieve listings with average rating above 4.5 stars.
Business goal: Drives recommendation and discovery modules.

```
db.listings.aggregate([
  { $unwind: "$reviews" },
  { $group: {
      _id: "$listing_id",
      title: { $first: "$title" },
      avg_rating: { $avg: "$reviews.rating" }
  }},
  { $match: { avg_rating: { $gte: 4.5 } }},
  { $sort: { avg_rating: -1 }},
  { $limit: 10 }
]);
```

Insight: Quickly identifies top-quality listings without heavy relational joins.

Query 3 — Track user activity logs
Purpose: Retrieve recent actions from the event stream stored in MongoDB (e.g., viewed listings, searches).
Business goal: Enhances personalization and activity tracking for analytics.

```
db.user_activity.find(
  { "user_id": "exampleuseruid" },
  { action: 1, listing_id: 1, timestamp: 1 }
).sort({ timestamp: -1 }).limit(20);
```

Insight: Provides behavioral data for analytics and recommendation systems, something that can be scalable in the future to a better algorithm.

6.3 Summary Table

Query Type	Database	Purpose	Output Example
Top listings by city	PostgreSQL	Identify popular destinations	City + Listing + Count
Host earnings report	PostgreSQL	Financial summary	Host + Total revenue
Guest booking history	PostgreSQL	Personalized dashboard	Booking list
Listing document	MongoDB	Quick read access	JSON listing
Top-rated listings	MongoDB	Recommendations	JSON top 10
User activity log	MongoDB	Behavioral insights	Recent actions

6.4 Integration Insight

Relational (PostgreSQL) supports transactions, consistency, and reliable data for payments and bookings.
NoSQL (MongoDB) enables fast document retrieval for liistings, search, and recommendations.
Together, they can help us to fulfill both real-time and analytical needs of the platform.

7 References

References

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

[10] w3schools. (n.d.). MongoDB Tutorial. Retrieved from https://www.w3schools.com/mongodb

[11] w3schools. (n.d.). PostgreSQL Tutorial. Retrieved from https://www.w3schools.com/postgresql

[12] CodeWithHarry(Youtube). (n.d.). MongoDB Tutorial in 1 Hour (2024). Retrieved from https://www.youtube.com/watch?v=J6mDkcqU_ZE