HBnB Evolution – Technical Documentation

Summary

This document provides a comprehensive overview of the architecture and design for the HBnB Evolution application, an AirBnB-inspired platform. It includes high-level architecture, detailed domain models, and key API workflows, serving as a blueprint for subsequent development phases.

1. Global Architecture

1.1 Layered Architecture Diagram

![Architecture Diagram application is organized into three main layers:

- Presentation Layer: Handles HTTP requests/responses, API endpoints, and data serialization.
- Business Logic Layer: Contains core business rules, data validation, and domain models.
- Persistence Layer: Manages database access, data storage, and retrieval.

Facade Pattern:
Each layer communicates with the next via a Facade, simplifying interactions and decoupling implementation details.

2. Domain Model

2.1 UML Class Diagram

![Class Diagram 2.2 Main Entities

- User:

  Attributes: id, first_name, last_name, email, password_hash, is_admin, created_at, updated_at

  Methods: authenticate(), update_profile()

- Place:

Attributes: id, title, description, price, latitude, longitude, owner_id, created_at, updated_at

Methods: add_amenity(), remove_amenity()

- Review

Attributes: id, place_id, user_id, rating, comment, created_at, updated_at

- Amenity:

Attributes: id, name, description, created_at, updated_at

2.3 Relationships

- One User owns many Places
- One Place has many Reviews
- Many Places can have many Amenities (many-to-many)

3. API Workflows (Sequence Diagrams)

3.1 User Registration

![User Registration Sequence](user_sequence.m sends POST /users with registration data.
2. API validates input and forwards to business logic.
3. Business logic checks for unique email and creates the user.
4. Persistence layer saves the user to the database.
5. API returns confirmation to the client.

3.2 Place Creation

![Place Creation Sequence](place_sequence sends POST /places with place details.
2. API validates and delegates to business logic.
3. Business logic verifies the owner and creates the place.
4. Persistence layer saves the place.
5. API returns the created place to the client.

3.3 Review Submission

![Review Submission Sequence](review_sequence sends POST /reviews with review data.
2. API and business logic validate and process the review.
3. Review is saved and confirmation is returned.

3.4 Fetching List of Places

![Fetch Places Sequence](fetch_places_sequence sends GET /places.
2. API forwards the request to business logic.
3. Business logic queries the persistence layer.
4. List of places is returned to the client.

4. Business Rules and Constraints

- All entities have unique IDs and timestamps (created_at, updated_at).
- Users can be regular or administrators.
- Places must have valid geolocation and non-negative prices.
- Reviews must be linked to existing users and places.
- Amenities are managed centrally and can be linked to multiple places.

5. References

- Mermaid.js Documentation

- UML Class Diagram Tutorial
- UML Sequence Diagram Tutorial

6. File Structure

part1/

```
├── API-Architecture.mmd      # High-level architecture diagram (Mermaid)
├── Classe-UML.md             # UML class diagram (Mermaid or Markdown)
├── user_sequence.mmd         # User registration sequence diagram
├── place_sequence.mmd        # Place creation sequence diagram
├── review_sequence.mmd       # Review submission sequence diagram
├── fetch_places_sequence.mmd # Fetch places sequence diagram
└── README.md                 # This documentation file
```

This documentation is intended to guide the development team, ensure alignment on system design, and serve as a reference throughout the HBnB Evolution project.