Contents

HBnB Evolution - Technical Documentation

Table of Contents

---

1. Introduction

HBnB Evolution is a simplified AirBnB-like application that allows users to: - Register and manage user profiles - Create and manage property listings (places) - Leave reviews for places - Search for available places with filters

Main Entities

- **User**: Represents registered users (first_name, last_name, email, password, admin status)
- **Place**: Property listings (title, description, price, latitude, longitude)
- **Review**: User feedback on places (rating, comment)
- **Amenity**: Property features (name, description)

Architecture

The application uses a **3-layer architecture**: - **Presentation Layer** (API): Handles user requests and responses - **Business Logic Layer**: Contains models and business rules - **Persistence Layer**: Manages data storage and retrieval

---

2. High-Level Architecture

The application follows the **Facade Pattern** to simplify communication between layers.

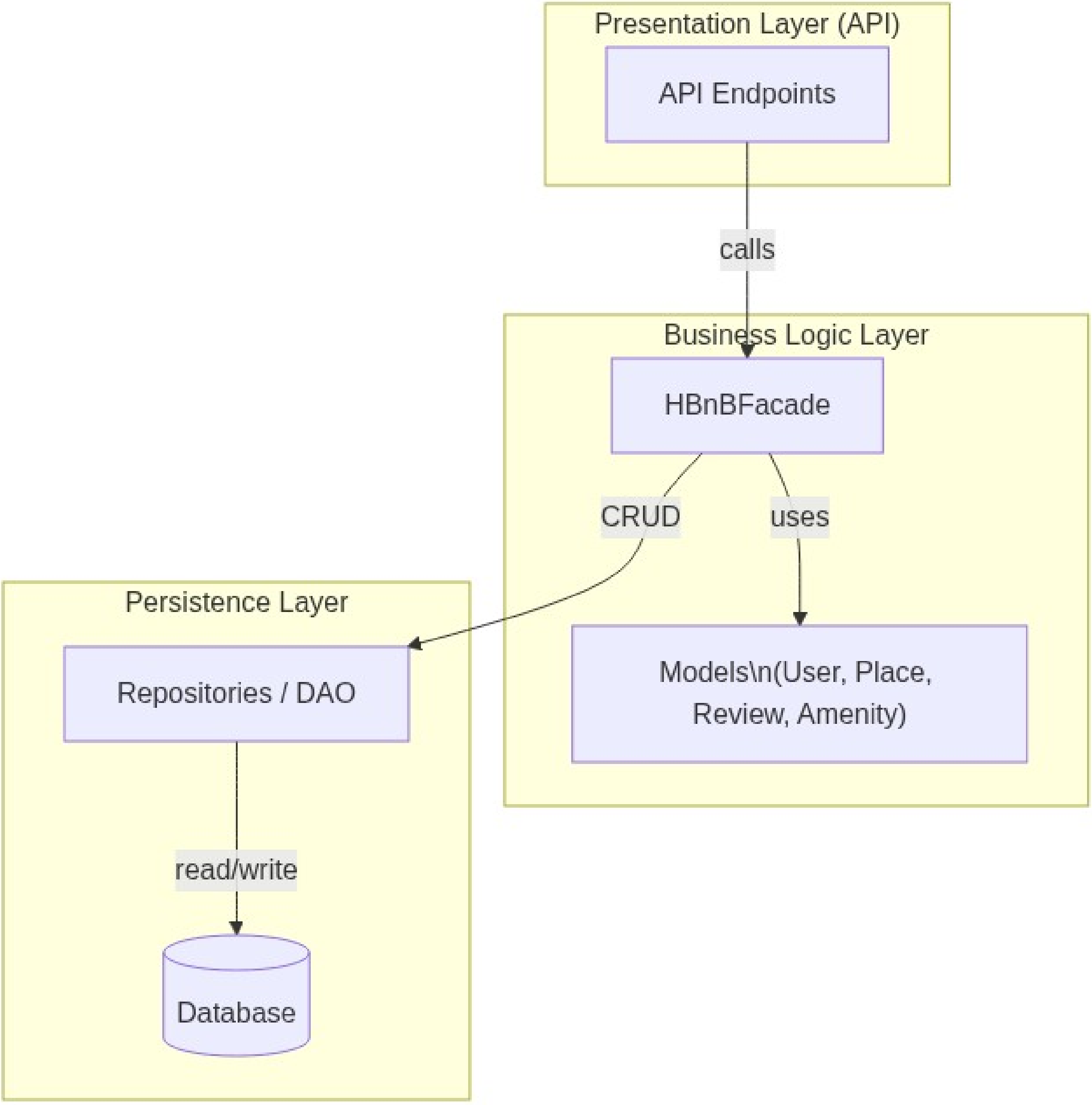

Figure 1: Mermaid Diagram 1

Layer Responsibilities: - **Presentation Layer**: Receives HTTP requests, returns JSON responses - **Business Logic Layer**: Validates data, enforces business rules, manages models - **Persistence Layer**: Handles database operations (CRUD)

---

3. Business Logic Layer

Class Diagram

Entity Descriptions

BaseModel (abstract): - `id`: Unique identifier (UUID) - `created_at`, `updated_at`: Audit timestamps - Common CRUD methods

User: - Attributes: first_name, last_name, email (unique), password (encrypted), admin flag - Can own multiple places, write multiple reviews

Place: - Attributes: title, description, price, latitude, longitude - Belongs to one user (owner) - Contains multiple amenities, receives multiple reviews

Review: - Attributes: rating (1-5), comment - Linked to one user and one place - Business rule: One review per user per place

Amenity: - Attributes: name (unique), description - Can be associated with multiple places

---

4. API Sequence Diagrams

4.1 User Registration

Flow: User submits registration form → System checks if email exists → If available, validates data and creates user account.

Business Rules: Email must be unique, password must be encrypted before storage.

4.2 Place Creation

Flow: User submits place details → System validates data (price, coordinates, etc.) → If valid, creates place and links to owner.

Business Rules: Price must be positive, coordinates must be valid, user must be authenticated.

4.3 Review Submission

Flow: User submits review → System checks if user already reviewed this place → If not, validates and saves review.

Business Rules: One review per user per place, rating must be 1-5, user cannot review their own place.

4.4 Fetching Places List

Flow: User requests places with filters → System validates criteria → Queries database and returns matching places.

Business Rules: Filters must be valid (price range, coordinates, etc.), results can be paginated.

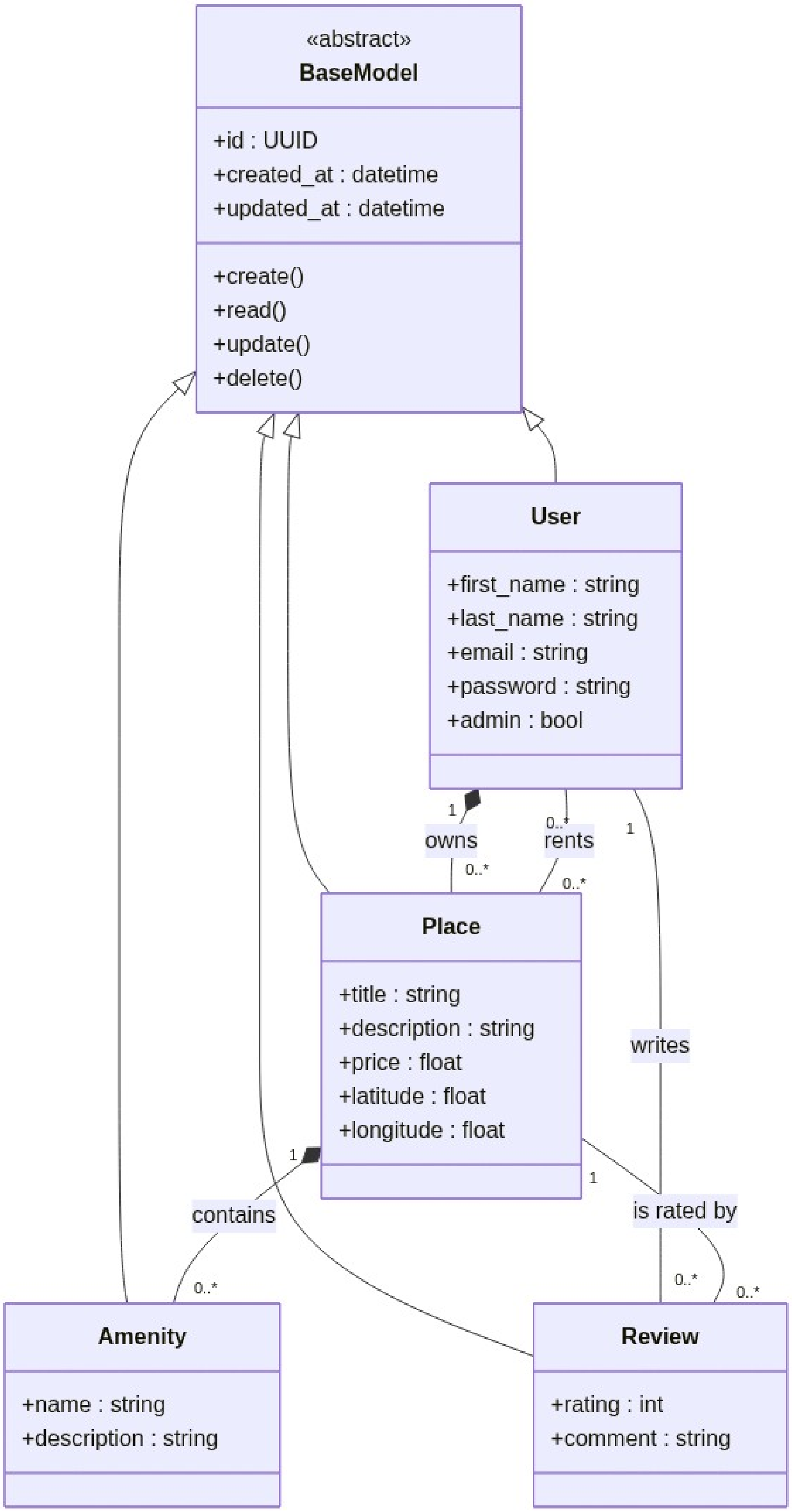

Figure 2: Mermaid Diagram 2

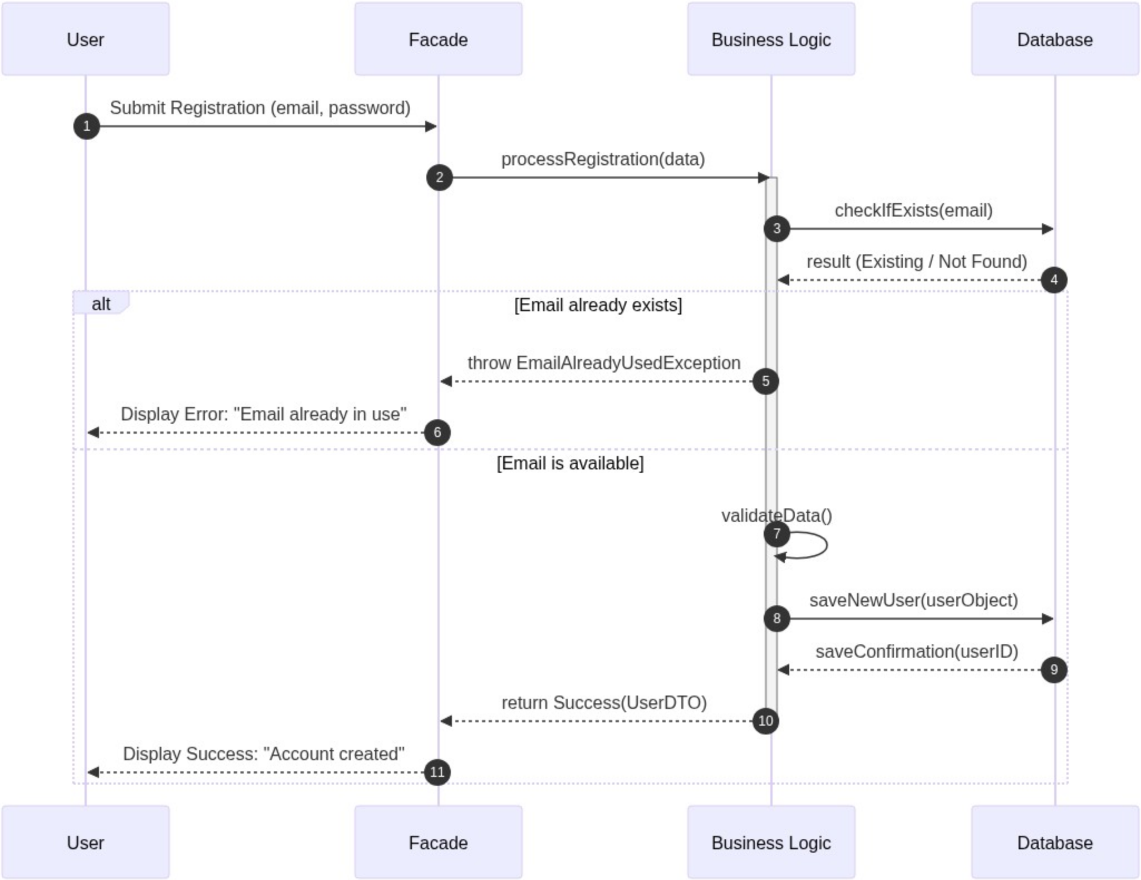

Figure 3: Mermaid Diagram 3

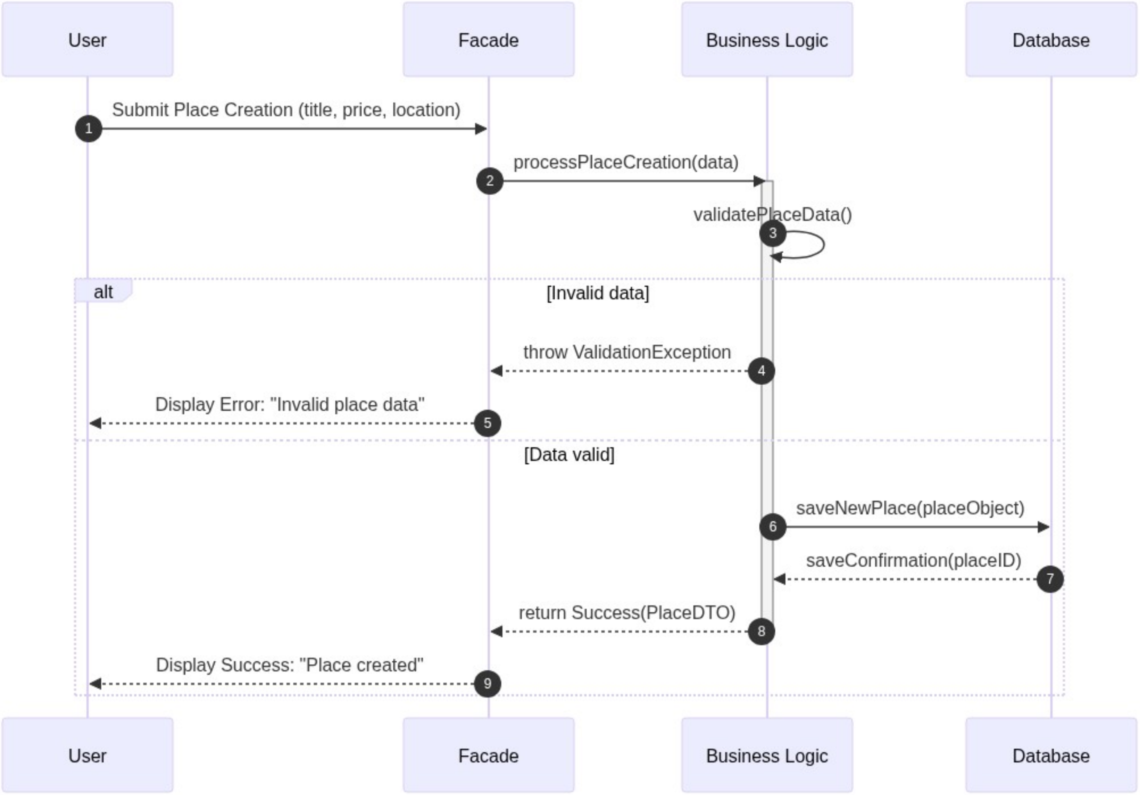
User
Facade
Business Logic
Database
1
Submit Place Creation (title, price, location)
2
processPlaceCreation(data)
3
validatePlaceData()
alt
[Invalid data]
4
throw ValidationException
5
Display Error: "Invalid place data"
[Data valid]
6
saveNewPlace(placeObject)
7
saveConfirmation(placeID)
8
return Success(PlaceDTO)
9
Display Success: "Place created"
User
Facade
Business Logic
Database

Figure 4: Mermaid Diagram 4

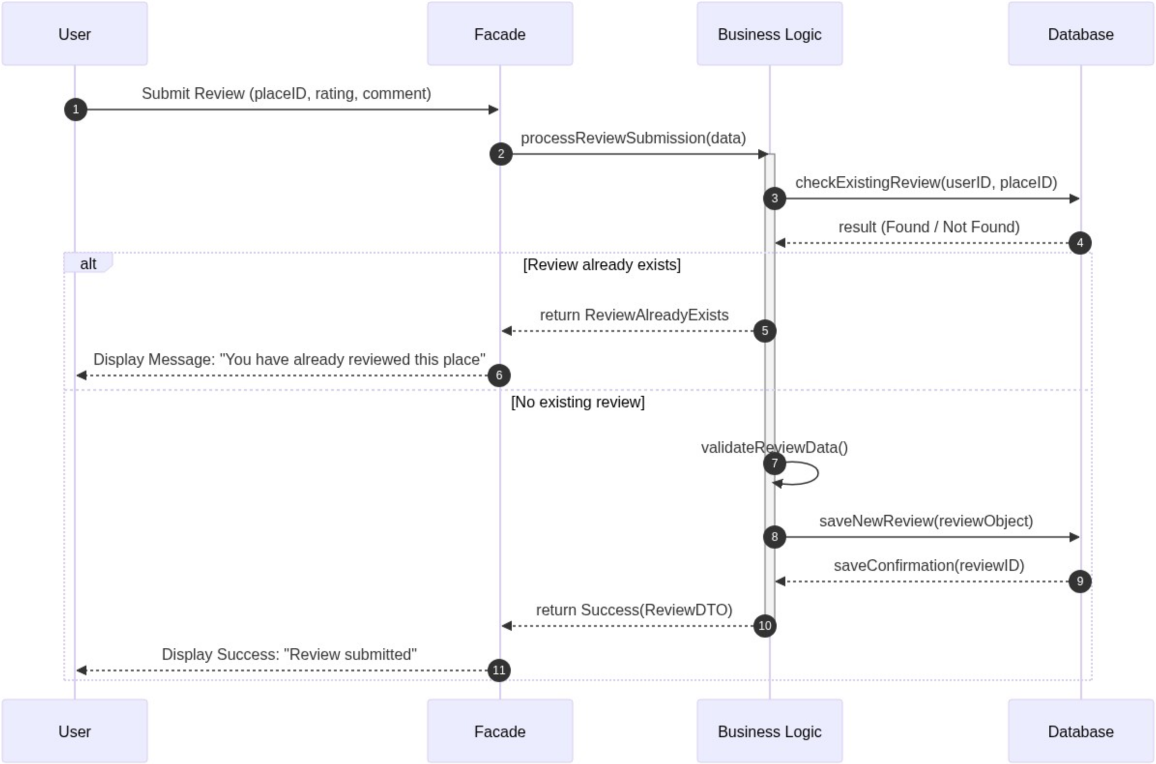
User
Facade
Business Logic
Database
1 Submit Review (placeID, rating, comment)
2 processReviewSubmission(data)
3 checkExistingReview(userID, placeID)
4 result (Found / Not Found)
alt
[Review already exists]
5 return ReviewAlreadyExists
6 Display Message: "You have already reviewed this place"
[No existing review]
7 validateReviewData()
8 saveNewReview(reviewObject)
9 saveConfirmation(reviewID)
10 return Success(ReviewDTO)
11 Display Success: "Review submitted"
User
Facade
Business Logic
Database

Figure 5: Mermaid Diagram 5

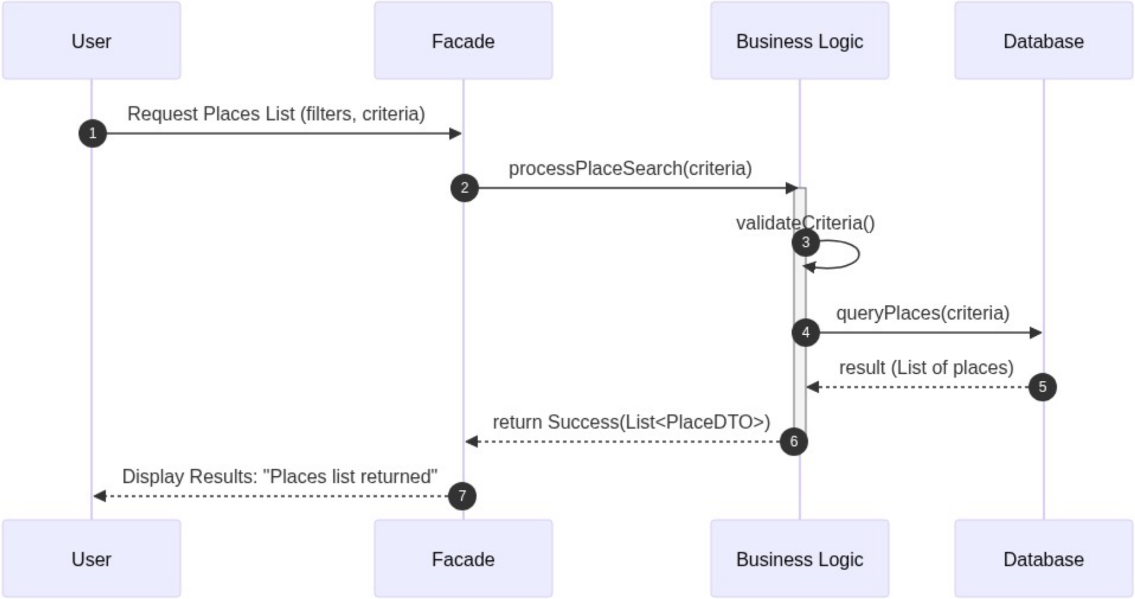
User
Facade
Business Logic
Database
1 Request Places List (filters, criteria)
2 processPlaceSearch(criteria)
3 validateCriteria()
4 queryPlaces(criteria)
5 result (List of places)
6 return Success(List<PlaceDTO>)
7 Display Results: "Places list returned"
User
Facade
Business Logic
Database

Figure 6: Mermaid Diagram 6

5. Conclusion

Summary

This documentation provides the architectural foundation for the HBnB Evolution application, covering: - Three-layer architecture with Facade pattern - Entity relationships and business rules - API interaction flows for core operations

Key Components

- **Entities**: User, Place, Review, Amenity (all inherit from BaseModel)
- **Relationships**: Composition (User-Place), Association (User-Review, Place-Review)
- **Communication**: Unidirectional flow through layers (API → Facade → Repository → Database)

Document Version: 1.0
Last Updated: February 12, 2026 **Author**: Florian ROOSEBEKE, Tom VIEILLEDENT