HBnB Evolution - Technical Documentation

Introduction

This document outlines the technical architecture, design, and interaction flow of the HBnB Evolution application. It serves as a comprehensive guide for the development and implementation phases, detailing the system’s structure, core business logic, and API interactions.

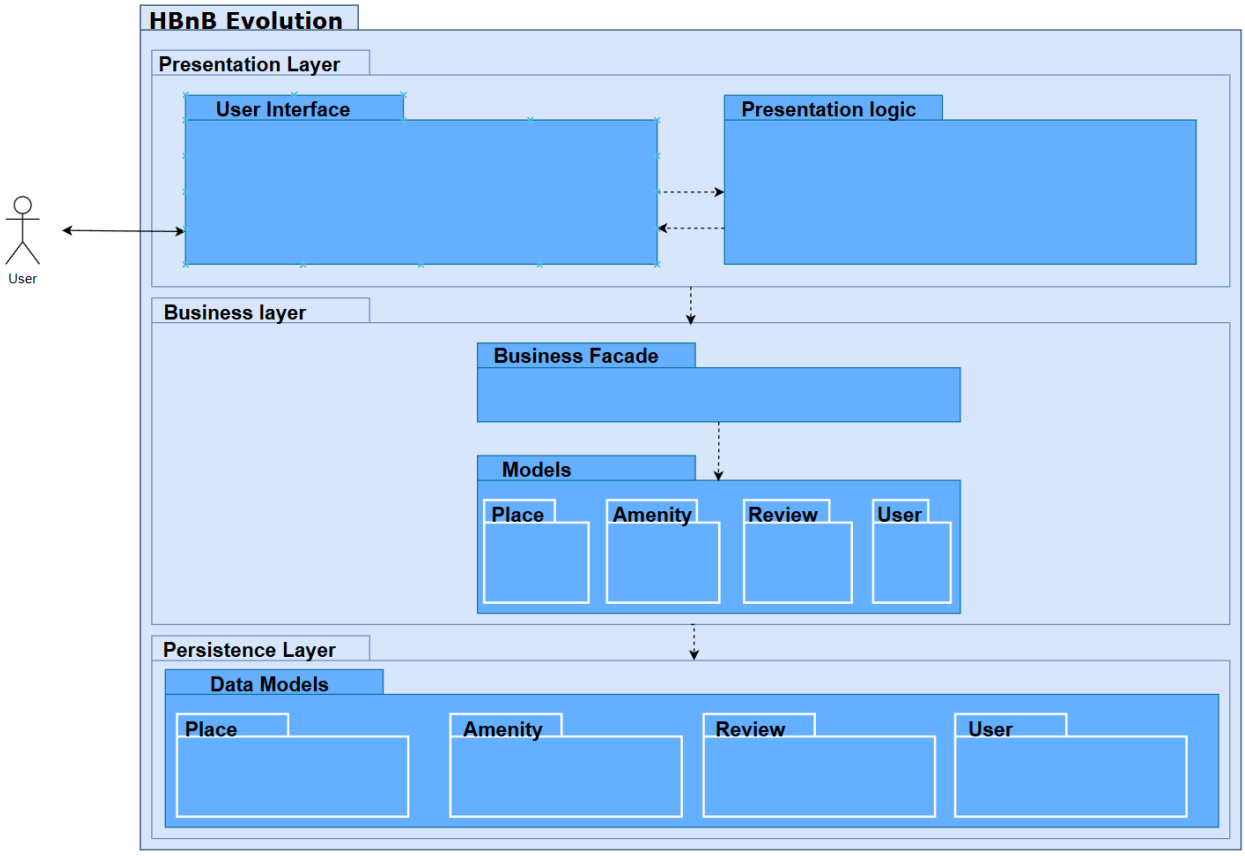
High-Level Architecture

Package Diagram

The HBnB Evolution application follows a three-layered architecture comprising:

1. **Presentation Layer (Services, API)**
    - Handles user interactions and exposes API endpoints.
    - Communicates with the Business Logic Layer via the Facade Pattern.
2. **Business Logic Layer (Models)**
    - Contains core business rules and logic.
    - Manages the application's core entities: User, Place, Review, and Amenity.
    - Interacts with the Persistence Layer for data storage and retrieval.
3. **Persistence Layer**
    - Responsible for data management and database operations.
    - Provides data access objects (DAOs) and repositories.

Facade Pattern: The Presentation Layer interacts with the Business Logic Layer through a unified interface, simplifying communication and decoupling the layers.



Business Logic Layer

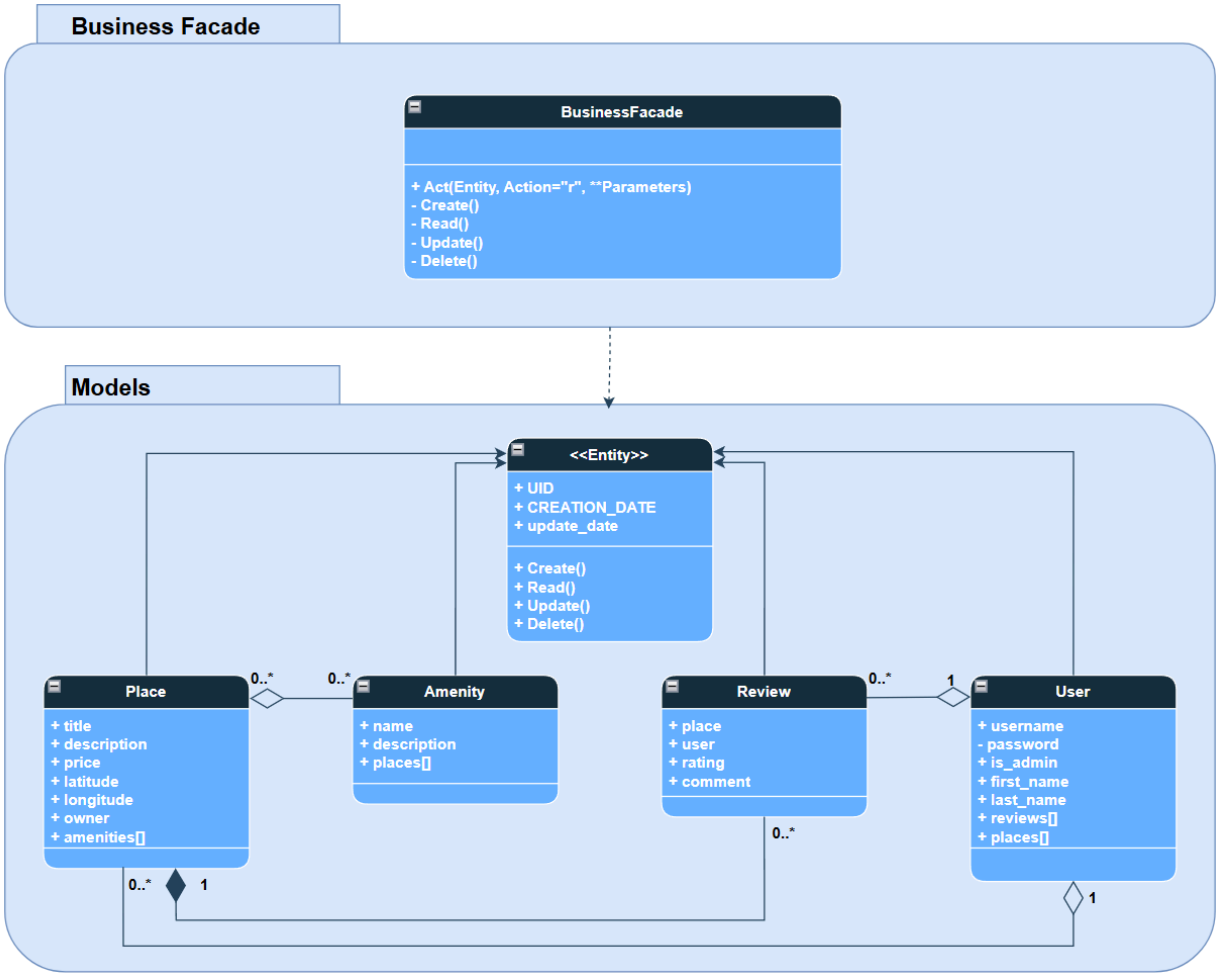
Class Diagram

The Business Logic Layer consists of the following core entities:

1. **User**
    - Attributes: id (UUID), first_name, last_name, email, password, is_admin (boolean), created_at, updated_at
    - Methods: register(), update_profile(), delete()
2. **Place**
    - Attributes: id (UUID), title, description, price, latitude, longitude, owner_id (User), created_at, updated_at
    - Methods: create_place(), update_place(), delete_place(), list_places()
3. **Review**
    - Attributes: id (UUID), place_id (Place), user_id (User), rating, comment, created_at, updated_at
    - Methods: submit_review(), update_review(), delete_review(), list_reviews_by_place()
4. **Amenity**
    - Attributes: id (UUID), name, description, created_at, updated_at
    - Methods: add_amenity(), update_amenity(), delete_amenity(), list_amenities()

Relationships:

- A User can own multiple Places.
- A Place can have multiple Reviews and Amenities.
- A Review is associated with one User and one Place.

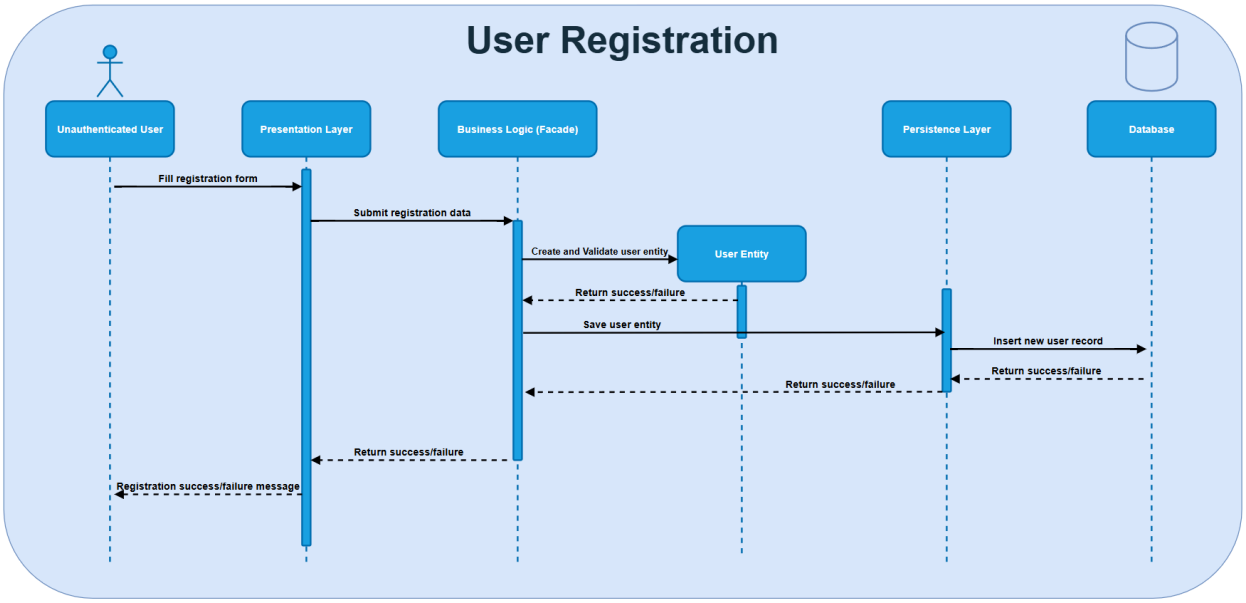


API Interaction Flow

Sequence Diagrams

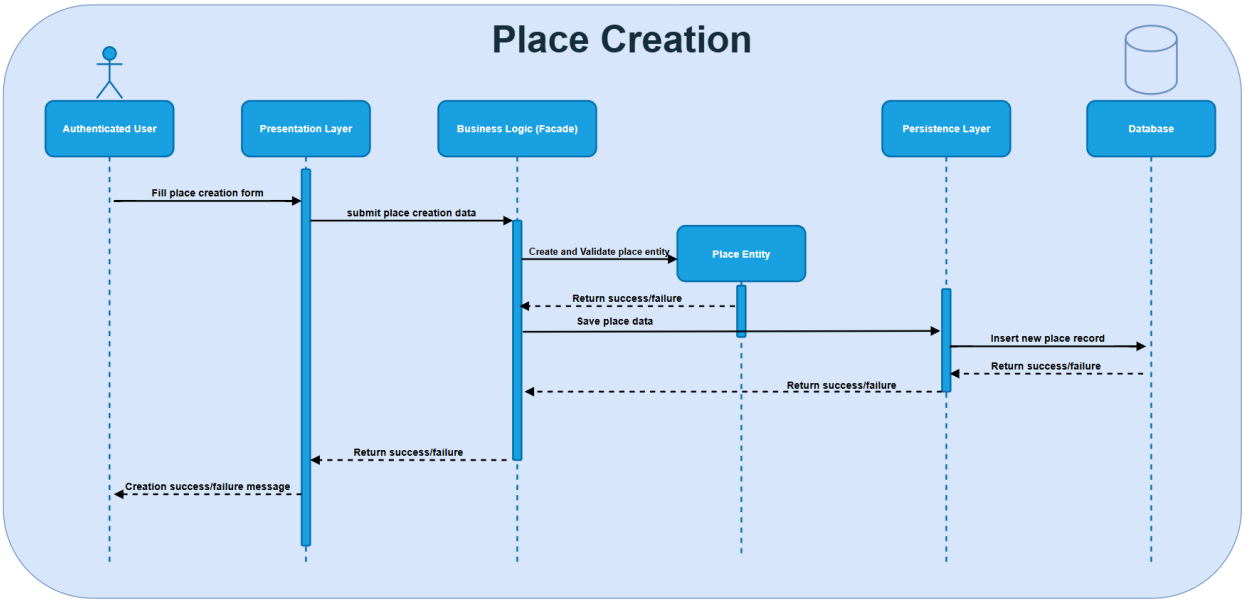
1. User Registration

- User sends registration data to the API.
- API validates the data and forwards it to the Business Logic Layer.
- Business Logic processes the registration and interacts with the Persistence Layer to store the new user.
- Success or error response is sent back to the user.



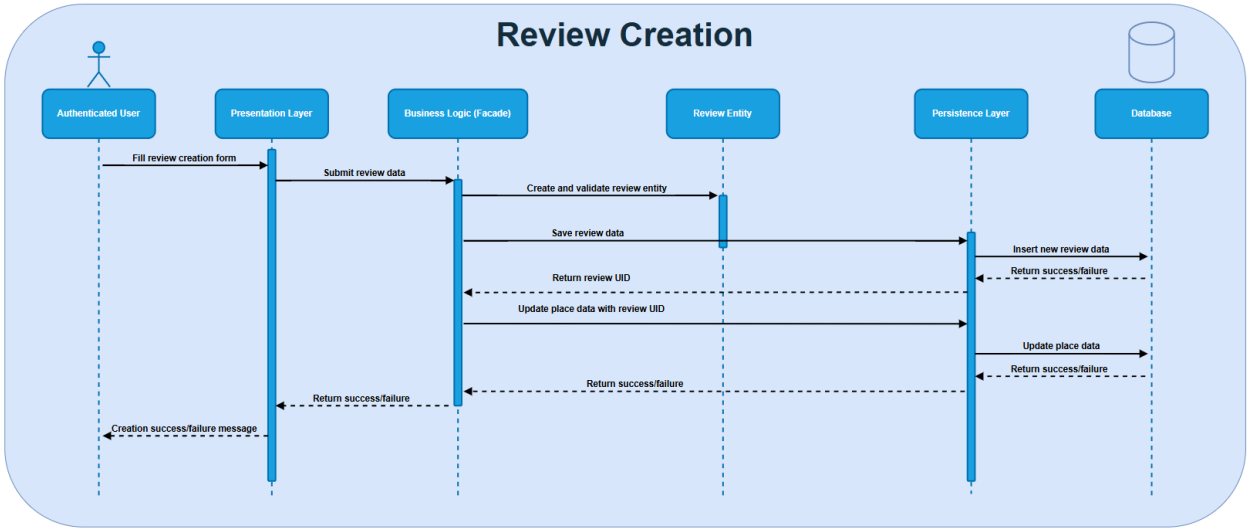
2. Place Creation

- User submits place details to the API.
- API validates and forwards the data to the Business Logic Layer.
- Business Logic creates the place and stores it via the Persistence Layer.
- API returns the newly created place details to the user.



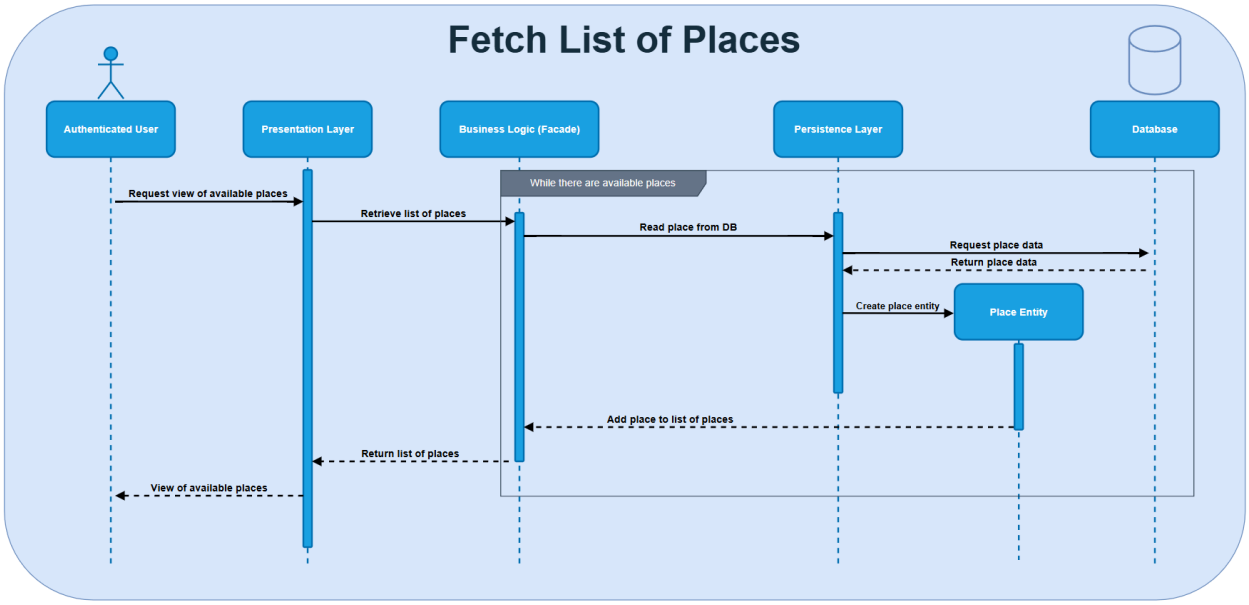
3. Review Submission

- User submits a review through the API.
- API validates and sends the review data to the Business Logic Layer.
- Business Logic saves the review in the database via the Persistence Layer.
- API returns a confirmation response.



4. Fetching a List of Places

- User requests a list of places.
- API processes the request and queries the Business Logic Layer.
- Business Logic retrieves data from the Persistence Layer.
- API returns the list of places to the user.



Conclusion

This technical documentation provides a detailed blueprint for the HBnB Evolution application, covering its architecture, core business logic, and API interactions. It serves as a guide for the implementation phase and ensures a consistent understanding of the system's design and functionality.