Holberton SCHOOL

-

HBnB Documentation

1. Introduction

This technical document outlines the foundational architecture and core business logic of the HBnB Evolution project. It describes a scalable, user-driven system for listing, discovering, and reviewing places. The document serves as a blueprint to ensure consistent, maintainable, and extensible implementation across the application.

The scope of this document includes:
- High-level package architecture
- Detailed class descriptions for the Business Logic Layer
- Sequence descriptions for core API interactions
- Key design decisions and architectural justifications

High-Level System Architecture

The HBnB Evolution system follows a three-layer architecture that separates concerns and improves maintainability. These layers are Presentation Layer, Business Logic Layer, and Persistence Layer. Communication between the Presentation Layer and Business Logic Layer is handled through a Facade pattern, which provides a unified interface and hides internal complexity.

1. Presentation Layer

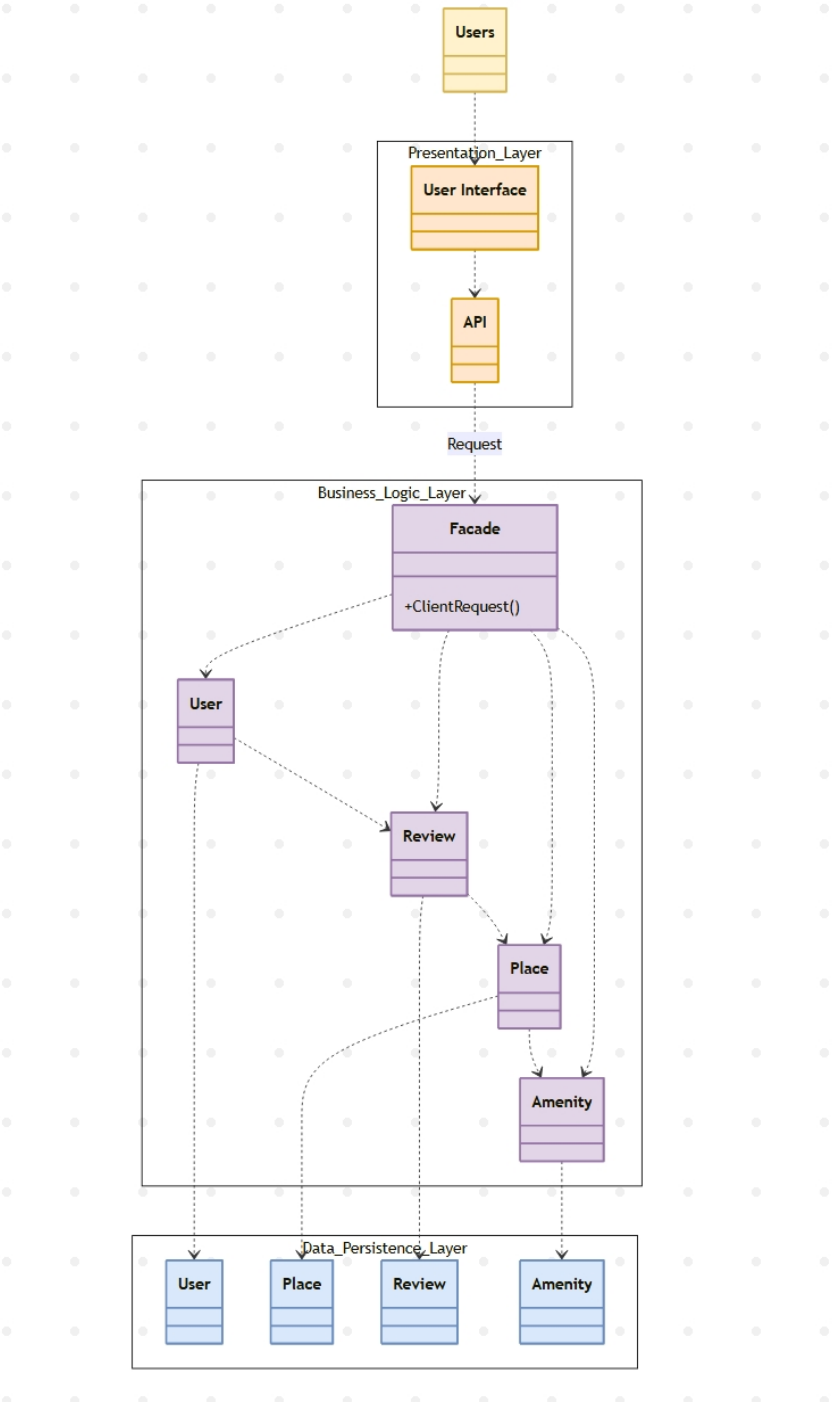
The Presentation Layer is responsible for handling all user interactions. It receives HTTP requests from users, validates basic input, and forwards requests to the Business Logic Layer via the Facade. This layer does not contain business rules and focuses solely on request handling and response formatting.

2. Business Logic Layer

The Business Logic Layer is the core of the application. It contains the Facade class and the main domain models: User, Place, Review, and Amenity. This layer applies business rules, validations, and coordinates interactions between models and the persistence layer.

3. Persistence Layer

The Persistence Layer is responsible for data storage and retrieval. It abstracts database operations from the Business Logic Layer and performs create, read, update, and delete operations for all entities in the system.



Business Logic Layer and Class Descriptions

User:

Represents individuals who use the platform.
Attributes: id, first_name, last_name, email, password, is_admin, created_at, updated_at.
Methods: register(), update(), delete().
A User can own multiple Places and write multiple Reviews.

Place:

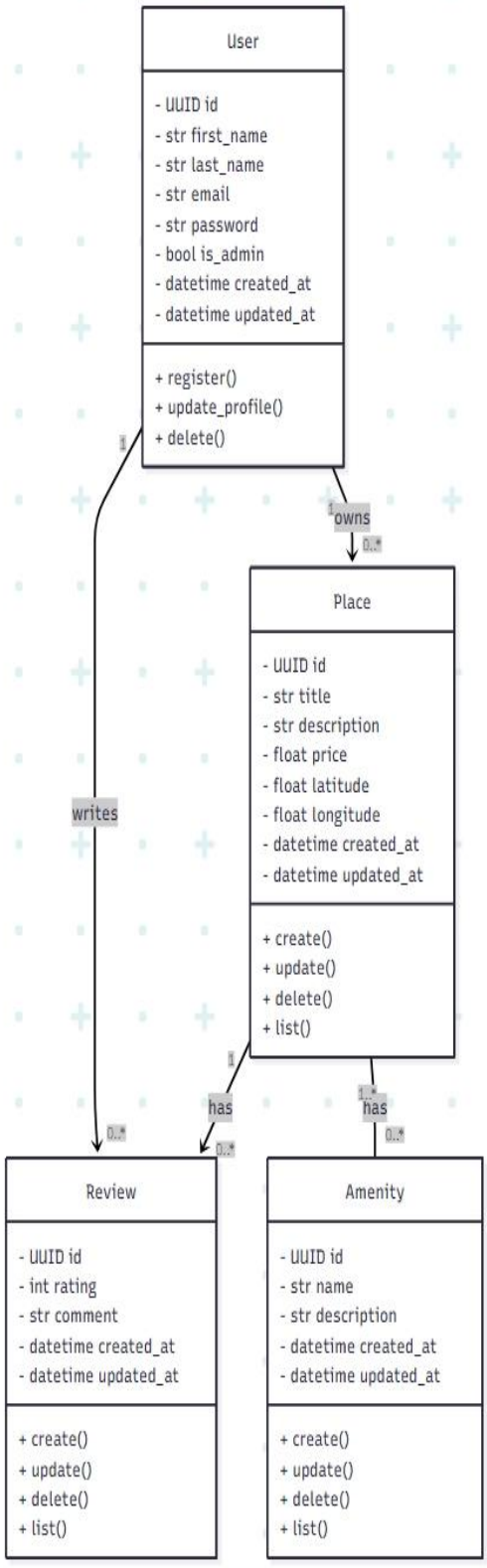
Represents properties listed by users.
Attributes: id, title, description, price, latitude, longitude, created_at, updated_at.
Methods: create(), update(), delete(), list().
A Place belongs to one User, can have multiple Reviews, and multiple Amenities.

Review:

Represents feedback left by users on places.
Attributes: id, rating, comment, created_at, updated_at.
Methods: create(), update(), delete(), list().
A Review is written by one User and belongs to one Place.

Amenity:

Represents features or services associated with places.
Attributes: id, name, description, created_at, updated_at.
Methods: create(), update(), delete(), list().
A Place can have many Amenities, and an Amenity can belong to many Places.



Design Decisions

- UUIDs are used for unique identification across systems.
- Timestamps are included for auditing purposes.
- CRUD operations are standardized across entities.
- Directed associations prevent circular dependencies.
- Encapsulation ensures data integrity and controlled access.

Sequence Flow Description

The system processes API calls in a structured sequence. A request begins at the Presentation Layer, passes through the Business Logic Layer for validation and processing, and reaches the Persistence Layer for data storage or retrieval. Once completed, responses propagate back up to the user. This applies to user registration, place creation, review submission, and fetching lists of places.