HBnB Project – Comprehensive Technical Documentation

Ryota et Simon

1. Introduction

This technical document serves as a detailed blueprint for the HBnB project, designed as a property rental platform. The objective is to provide a clear, structured, and professional guide for implementation. This document compiles all architectural and design diagrams—including the high-level architecture, the business logic class diagram, and API interaction flows—with accompanying explanations. It will be used as a reference throughout the project's development lifecycle.

2. High-Level Architecture

The following diagram illustrates the high-level layered architecture of the HBnB project, leveraging the Facade Pattern to simplify the interactions between layers.

The architecture follows a clean separation of concerns:
 - The Presentation Layer contains the API controllers for User, Place, and Review.
 - The Facade Layer simplifies interactions by exposing high-level interfaces.
 - The Business Logic Layer manages all application logic.
 - The Persistence Layer encapsulates direct database operations.
 This layered approach ensures molecularity and maintainability.

TASK0

PresentationLayer

=== API Requests ===
+UserController
+PlaceController
+ReviewController

API Requests

FacadeLayer

=== Facade Pattern ===
+UserFacade
+PlaceFacade
+ReviewFacade

Searching logic methodes

BusinessLogicLayer

=== All Methods ===
+UserService
+PlaceService
+ReviewService
+User
+Place
+Review
+Amenity

DataBase Operations

PersistanceLayer

=== Date Base ===
+UserRepository
+PlaceRepository
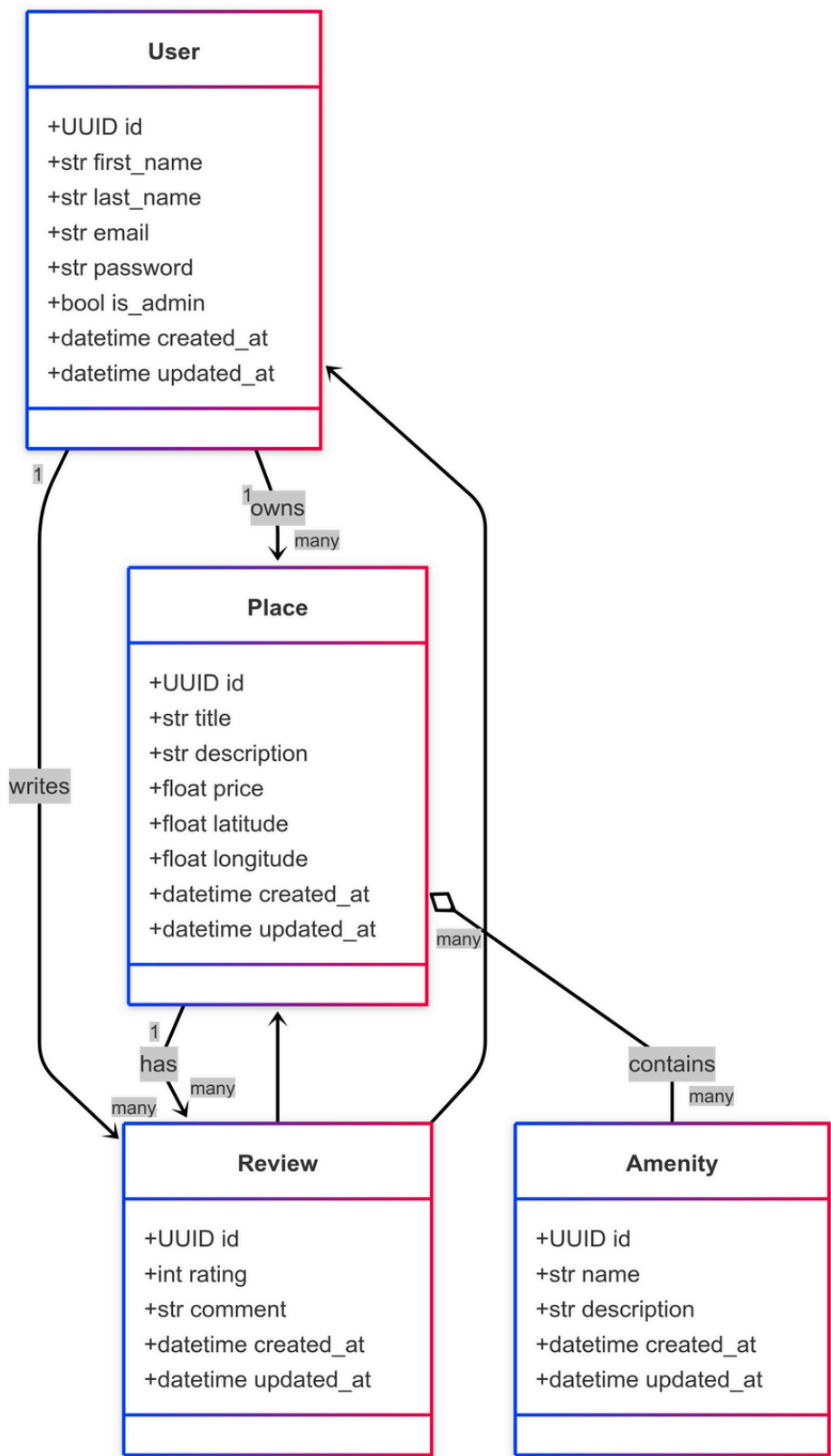+ReviewRepository
+AmenityRepository

3. Business Logic Layer

This diagram details the class structure for the Business Logic Layer, showing key entities and their relationships.

Entities include:
 - User: Manages user credentials and roles.
 - Place: Describes properties listed by users.
 - Review: Stores user reviews for each place.
 - Amenity: Lists amenities associated with places.
 The diagram illustrates relationships such as 'User owns Place', 'Place has Reviews', and 'Place contains Amenities'.
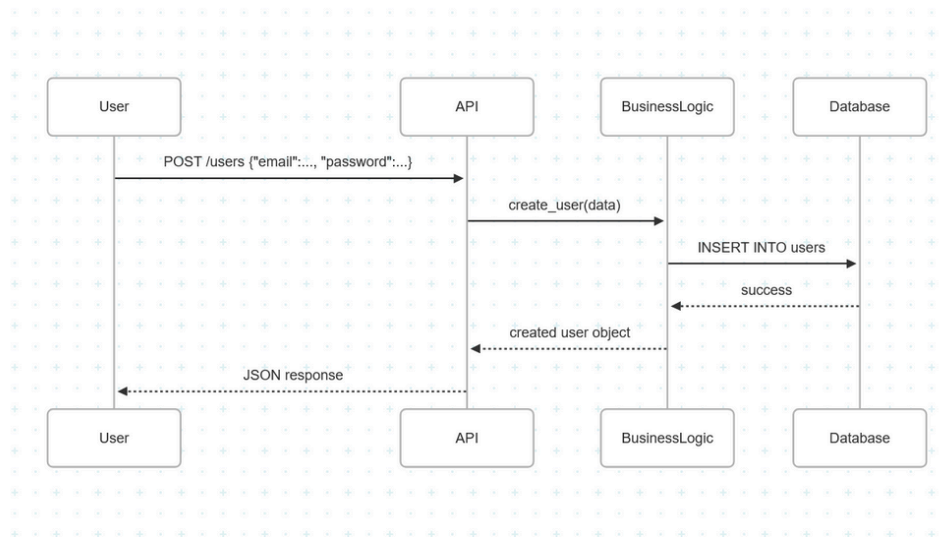
User

+UUID id
+str first_name
+str last_name
+str email
+str password
+bool is_admin
+datetime created_at
+datetime updated_at

Place

+UUID id
+str title
+str description
+float price
+float latitude
+float longitude
+datetime created_at
+datetime updated_at

Review

+UUID id
+int rating
+str comment
+datetime created_at
+datetime updated_at

Amenity

+UUID id
+str name
+str description
+datetime created_at
+datetime updated_at

1 owns many
1 has many
writes many
contains many

4. API Interaction Flow

This section presents sequence diagrams for typical API interactions in the system.

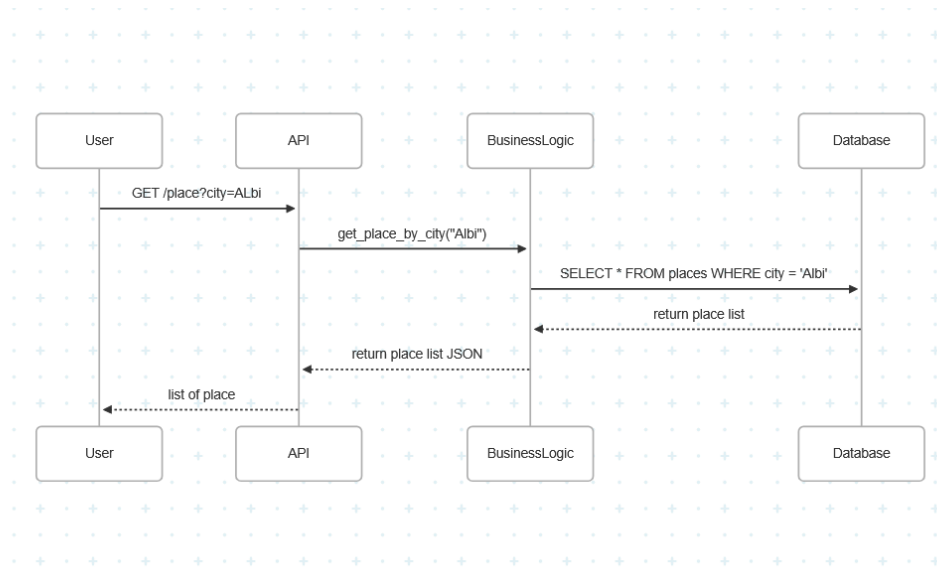## 4.1 Create User

The UserController receives a POST request to create a user. The Facade Layer calls the relevant service to insert the new user into the database, and a JSON response confirms creation.
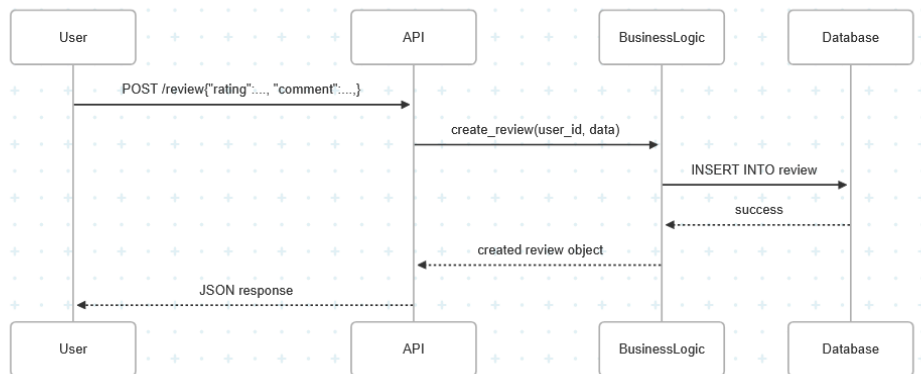

4.2 Create Place

A POST request `/places` is used to create a new place. The request is handled similarly via the API to Facade to Business Logic and then inserted into the database.

4.3 Create Review

A POST request `/review` with rating and comment data is handled by the ReviewController. The business logic layer processes the request and inserts the data into the review table in the database. A confirmation JSON is returned.

4.4 Retrieve Places by City

A GET request /place?city=Albi is sent to retrieve places by city. The API calls the Facade Layer, which queries the Business Logic and retrieves data from the database. The response is returned as a JSON list of places.