HBnB: Technical Document

# 

HBnB Evolution Project

Nayshka Oliveras Lugo

Cohort: 27

Holberton School - Ponce Campus

Github: LuvliNay - Email: [nayshka1206@gmail.com](mailto:nayshka1206@gmail.com)

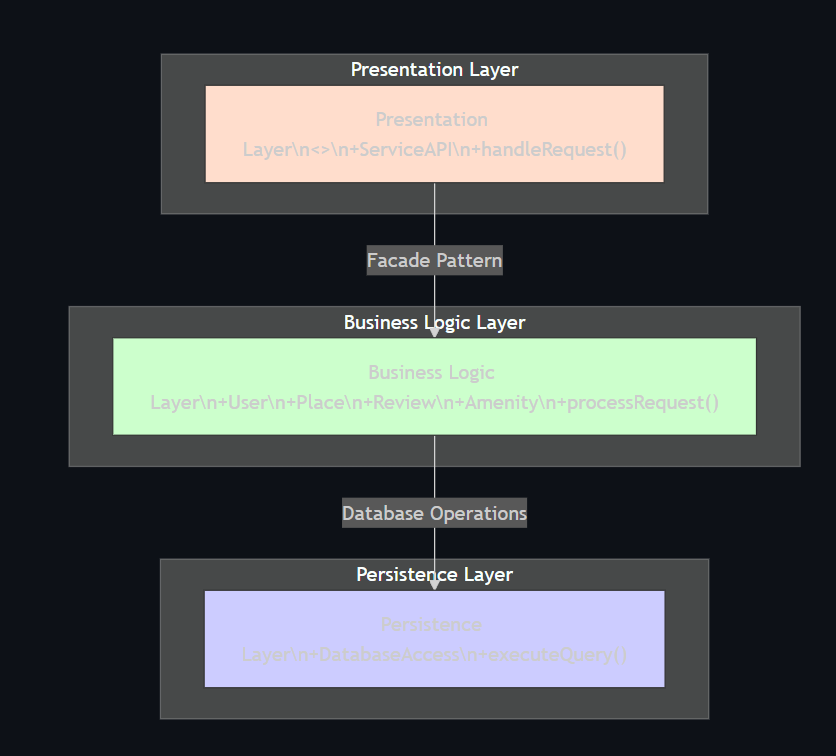
## INTRODUCTION:

This project is a technical documentation that will serve as the foundation for the development of the HBnB Evolution application. This documentation will help in understanding the layered architecture, the detailed design of the business logic, and the interactions within the system. It will serve as a blueprint for a better understanding and as a start for the HBnB project we will complete. We will see the process from these operations: user management, place management, review and amenity management. The tasks you will see here will contain the diagrams of a High-Level Package Diagram, a Detailed Class Diagram for Business Logic Layer, a Sequence Diagram for API Calls, and this Documentation Compilation.

## HIGH-LEVEL ARCHITECTURE (task 0):

The diagram shows a **three-layer architecture** using a **Facade pattern**:

**Presentation Layer**: is as a *facade* to handle requests and hide internal complexity.

**Business Logic Layer**: processes requests and coordinates database access.  


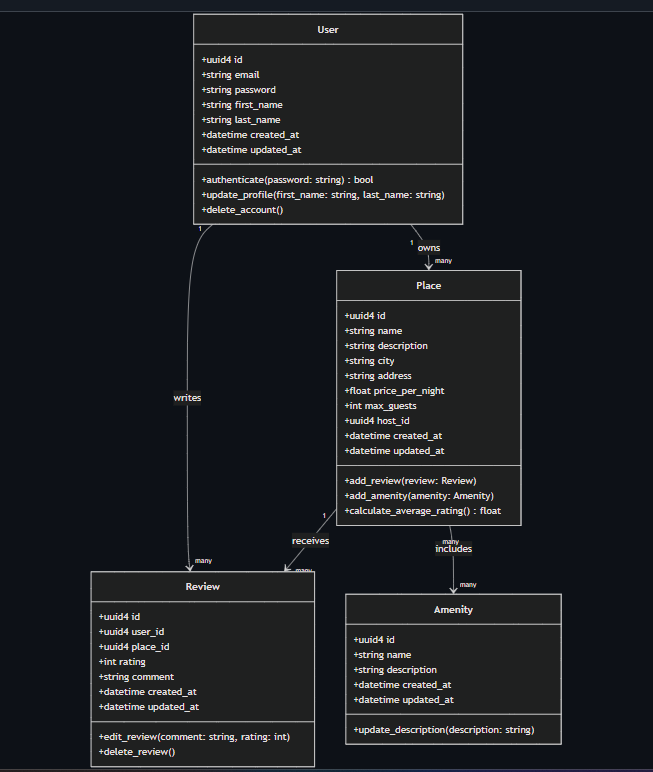
**Persistence Layer**: executes database operations.

The **Facade pattern** simplifies client interaction by exposing only the PresentationLayer.

## BUSINESS LOGIC LAYER (task 1):

This class diagram represents a **layered architecture** with a **Facade-style organization**:

**Business Logic Layer**: includes core classes (User, Place, Review, Amenity) that manage operations, relationships, and rules of the Hbnb system.

**Facade pattern**: though not explicitly shown, a higher-level interface (like a service or API layer) would interact with these classes to simplify access, hiding the complexity of managing users, places, reviews, and amenities.  


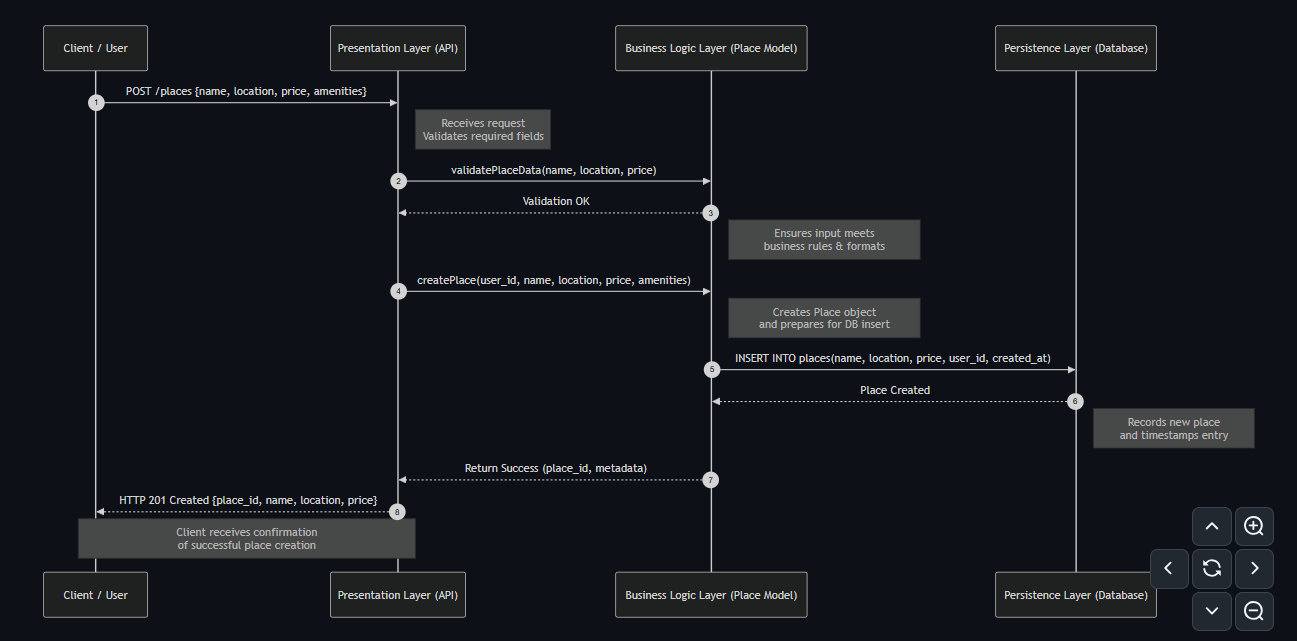
## API INTERACTION FLOW (task 2):

This sequence diagram illustrates a **layered architecture** with a **facade pattern**:

**Presentation Layer (API)**: acts as the **facade**, handling client requests, validating input, and coordinating the workflow.

**Business Logic Layer (Place Model)**: enforces rules, creates objects, and prepares data for persistence.

**Persistence Layer (Database)**: stores the data and returns confirmation.

The facade hides the complexity of business logic and database operations, providing a simple interface for clients.

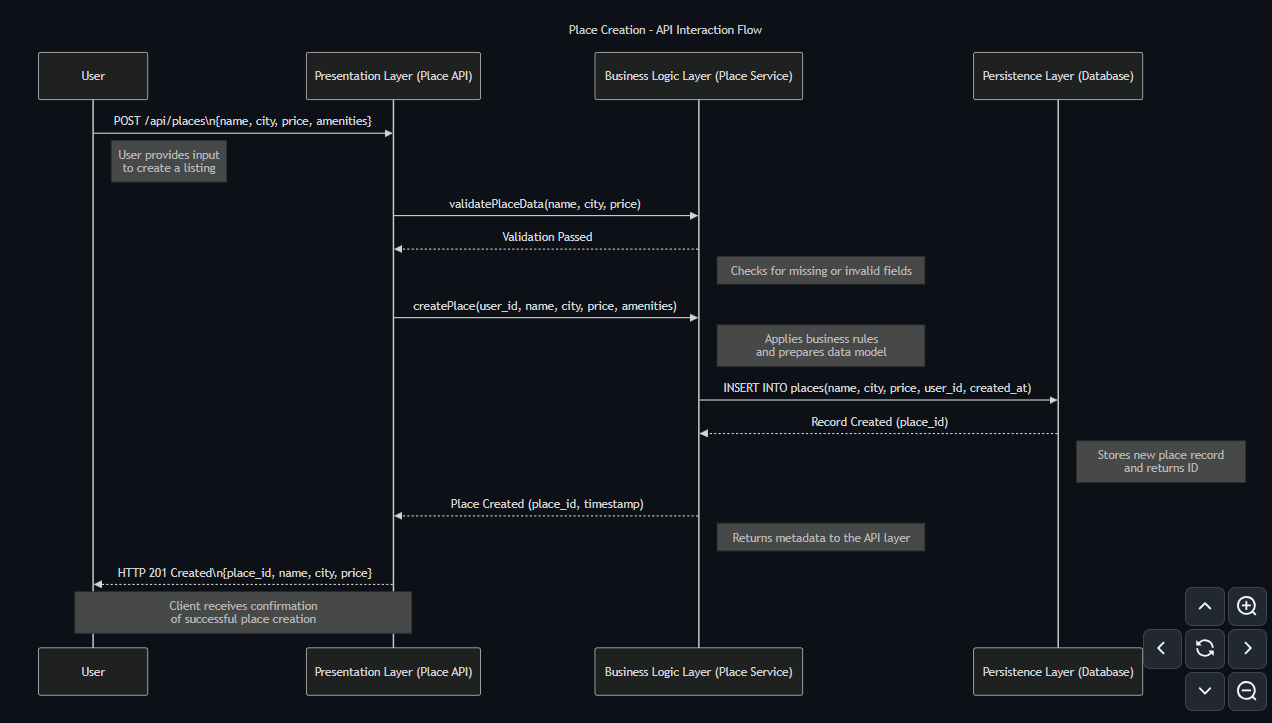
## PLACE CREATION (task 2\_2):

This sequence diagram shows a **layered architecture** for creating a new Place listing:

**Presentation Layer (API)**: acts as the facade, handling client requests, validating input, and coordinating the process.

**Business Logic Layer (Place Service)**: enforces rules, creates the Place object, and prepares data for storage.

**Persistence Layer (Database)**: stores the Place record and returns its ID.

It provides a clear, modular flow for creating a place while hiding internal complexity from the user through the API facade.

## REVIEW CREATION (task 2\_3):

This sequence diagram illustrates a **layered architecture** with a **facade pattern** for submitting a review:

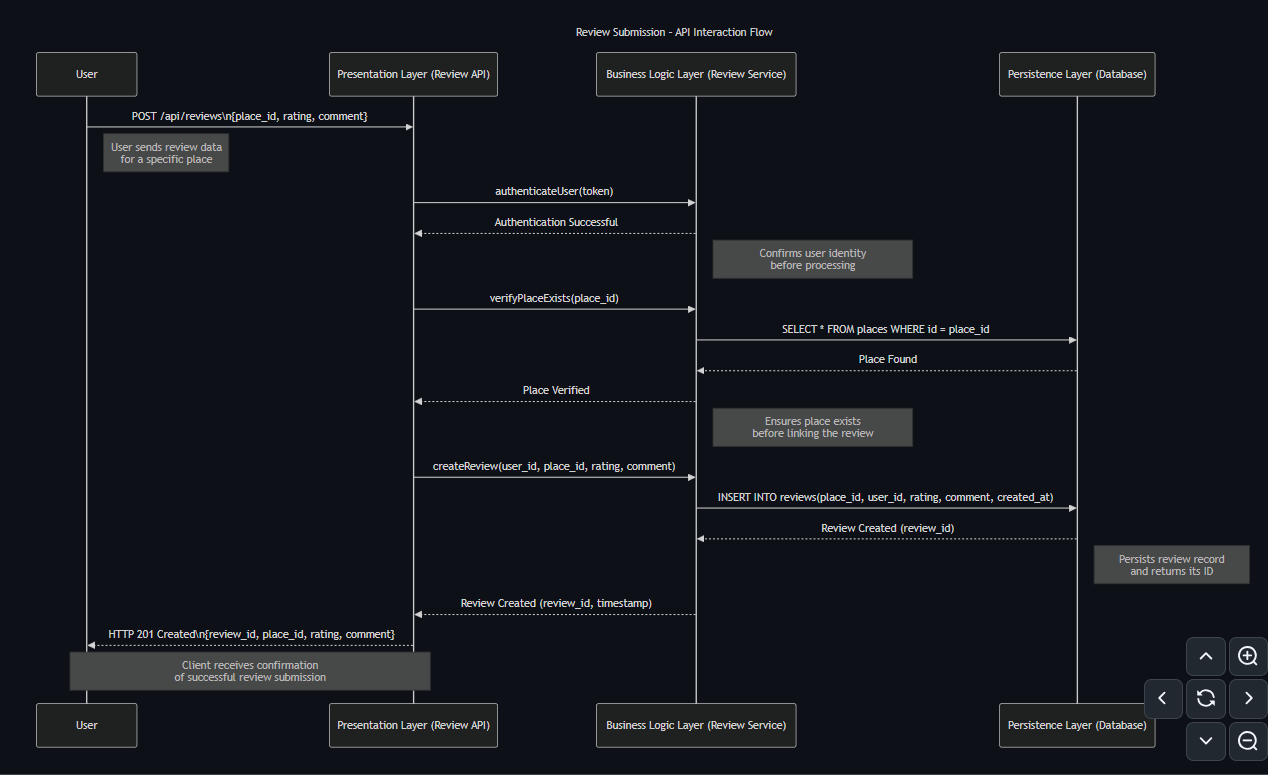
**Presentation Layer (Review API)**: acts as the **facade**, handling client requests, authentication, and coordinating the workflow.

**Business Logic Layer (Review Service)**: enforces rules, verifies the place, and creates the Review entity.

**Persistence Layer (Database)**: stores the review and returns its ID.

It manages review submission in a structured way, hiding complexity from the user while ensuring validation, authentication, and proper data storage.

Diagram below:

Diagram of the review creation:

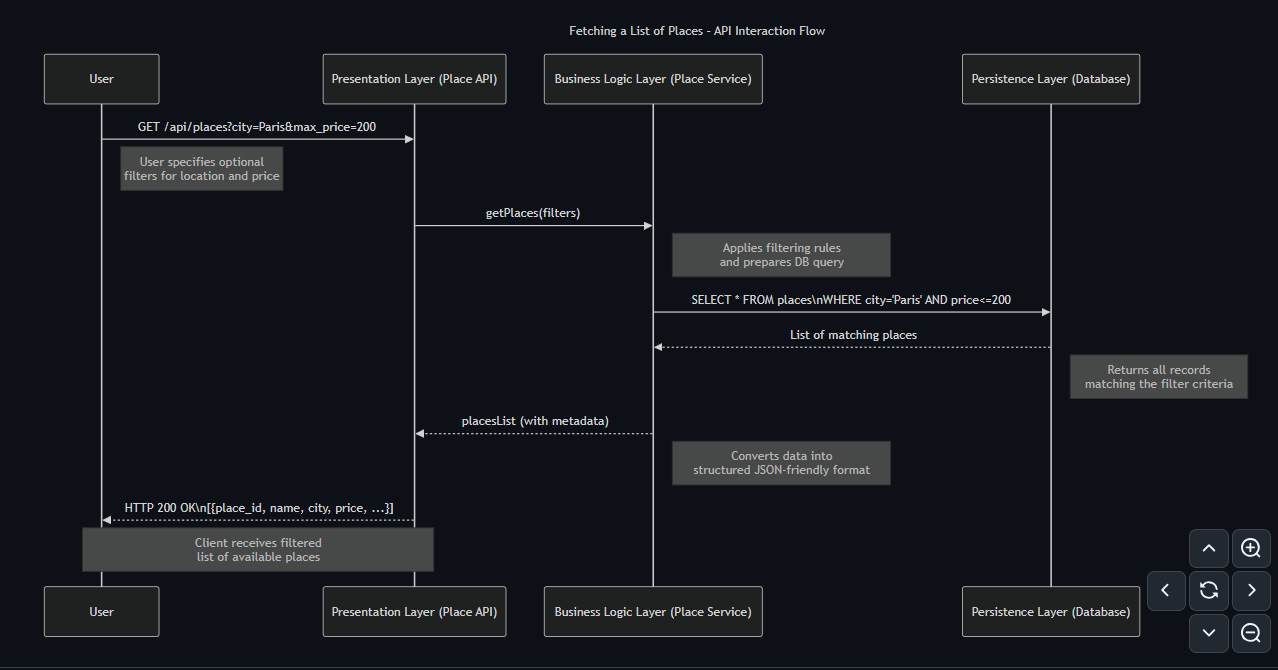
## PLACE SEARCH (task 2\_4):

This sequence diagram shows a **layered architecture** with a **facade pattern** for fetching places:

**Presentation Layer (Place API)**: acts as the **facade**, handling client requests, forwarding filters, and returning results.

**Business Logic Layer (Place Service)**: applies filtering rules, queries the database, and formats the data.

**Persistence Layer (Database)**: retrieves the matching place records.

It provides a clean, modular way for users to get a filtered list of places while hiding internal query and formatting complexity.

## CONCLUSION:

This document provides a clear and structured overview of the HBnB project, outlining its system design and overall blueprint. It explains how the different components work together and how data flows throughout the application. Key features such as Users, Places, Reviews, and Amenities are detailed, along with their relationships and interactions. The included diagrams enhance understanding and serve as valuable guidance during development, testing, and maintenance. Overall, this document supports the creation of a well-organized, scalable, and maintainable system that mirrors the architectural style of the AirBnB platform while enabling dynamic and interactive functionality.