HBnB - Proyect

Bryan J Camacho Santiago

Cohort:26
Holbertonschool
Ponce,Puerto Rico

git hub

https://github.com/Bryanjose001/holbertonschool-hbnb

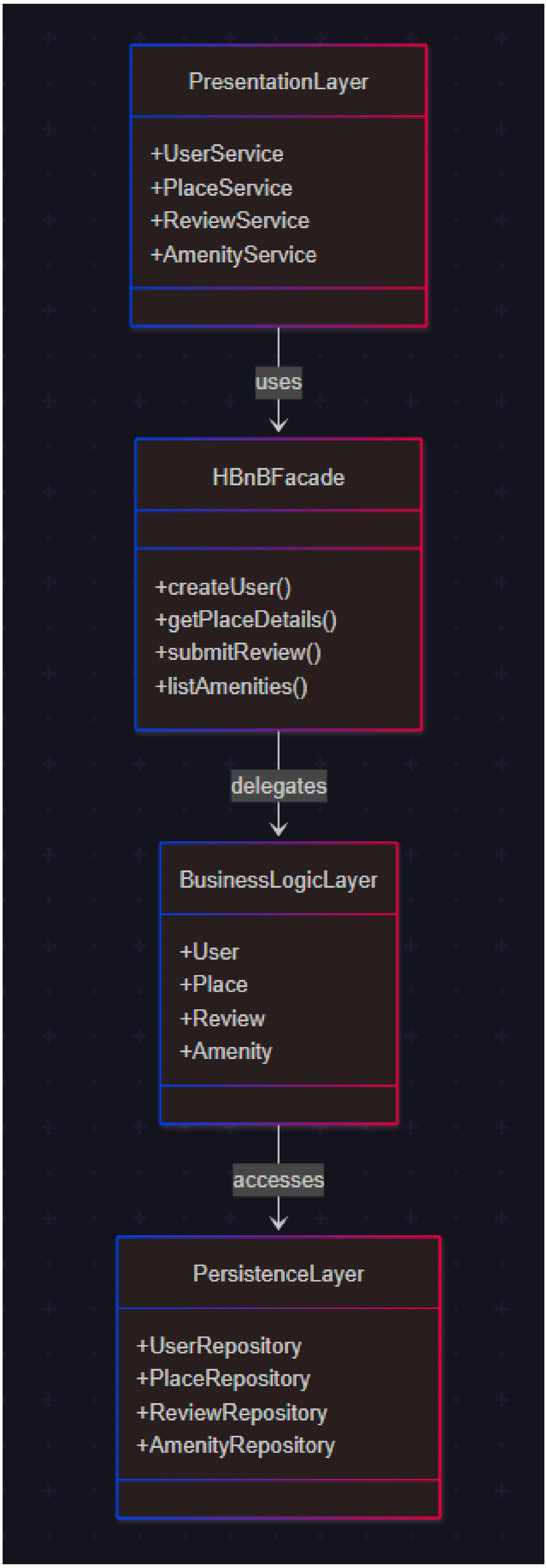
Introduction

This document provides a detailed technical overview of the HBnB project, a platform developed to handle property rentals, user engagement, and review management. It focuses on presenting the system’s overall structure, explaining essential business logic, and mapping out key API interaction workflows. By combining both high-level architecture and specific design elements, the document is intended to support implementation efforts, promote clarity among the development team, and serve as a reliable reference point for all stakeholders. Ensuring accuracy and thoroughness is critical for effective development and long-term maintenance.

High-level Architecture

The diagram illustrates a layered architecture using the Facade pattern to simplify interactions between components. The PresentationLayer contains services like UserService and PlaceService that interact with the central HBnBFacade, which acts as a unified entry point. The HBnBFacade delegates operations to the BusinessLogicLayer, which contains core domain classes (User, Place, etc.). These classes in turn access the PersistenceLayer through repository classes for data management. This structure promotes separation of concerns, maintainability, and modularity.

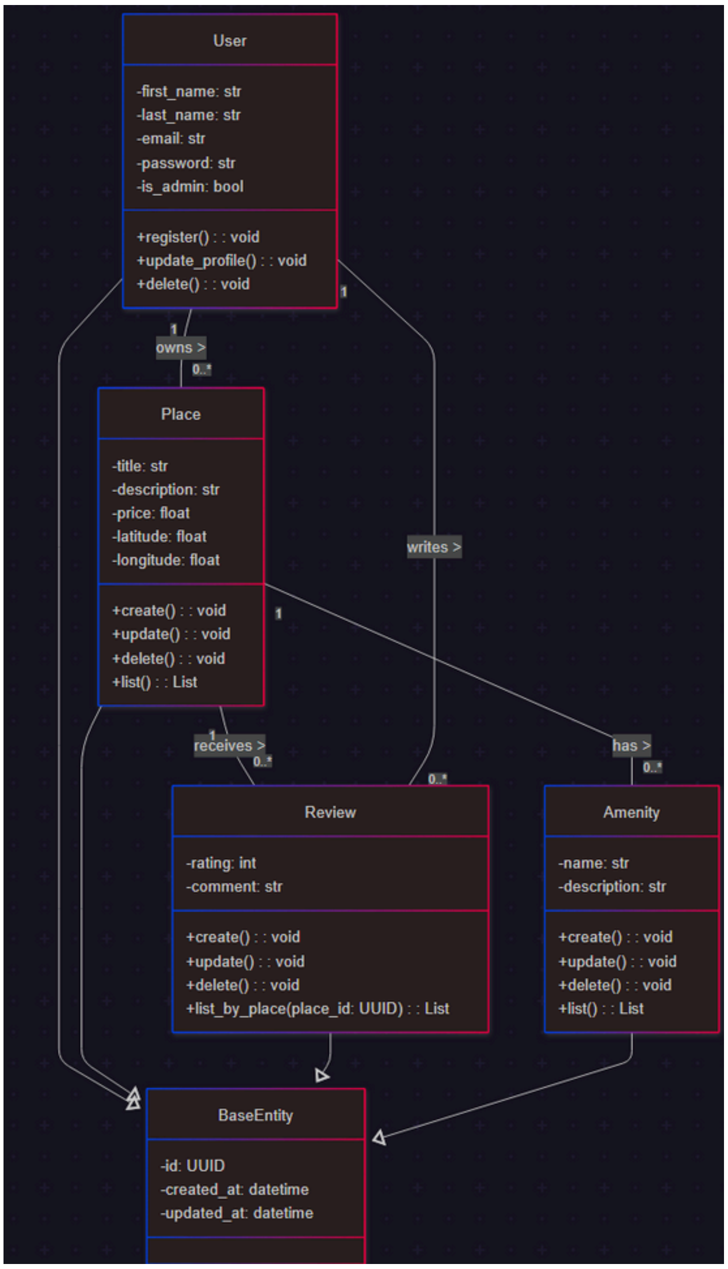
High-Level Package Diagram(task 0)


PresentationLayer
+UserService
+PlaceService
+ReviewService
+AmenityService
uses
HBnBFacade
+createUser()
+getPlaceDetails()
+submitReview()
+listAmenities()
delegates
BusinessLogicLayer
+User
+Place
+Review
+Amenity
accesses
PersistenceLayer
+UserRepository
+PlaceRepository
+ReviewRepository
+AmenityRepository

Business Logic Layer

This class diagram models the core domain entities of the application, all inheriting common fields (id, created_at, updated_at) from a shared BaseEntity. The User class includes personal and authentication attributes, along with methods for account management. A User can own multiple Place instances and write multiple Reviews. Each Place includes location and pricing details and can have many Reviews and Amenity instances associated with it. The Review class allows users to leave ratings and comments on places, while the Amenity class describes features available at a place. This structure promotes reusability, consistency, and clearly defines relationships between entities.

Detailed Class Diagram for the Business Logic Layer (task 1)

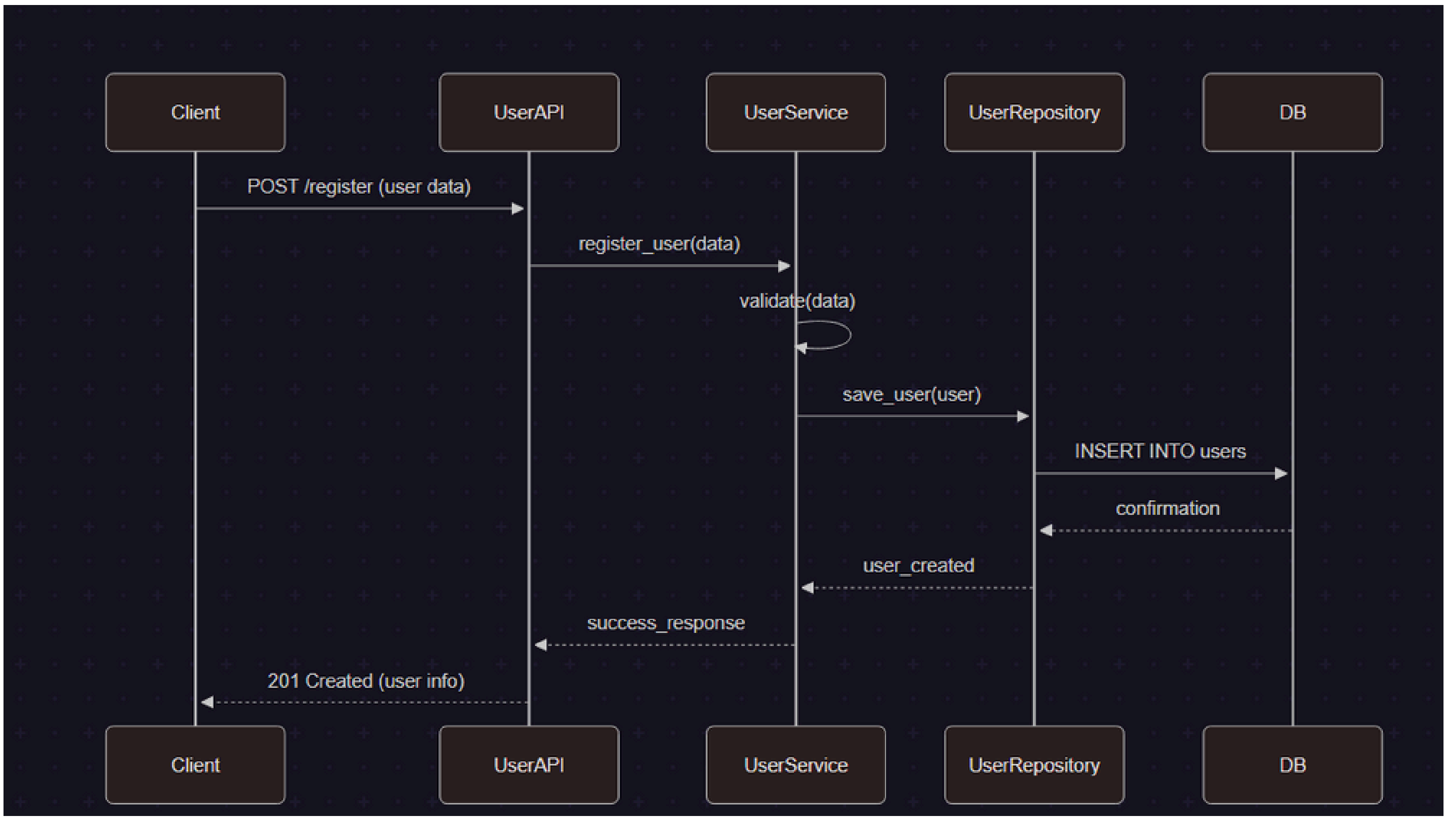


API Interaction Flow

Sequence Diagrams for API Calls(task 2)

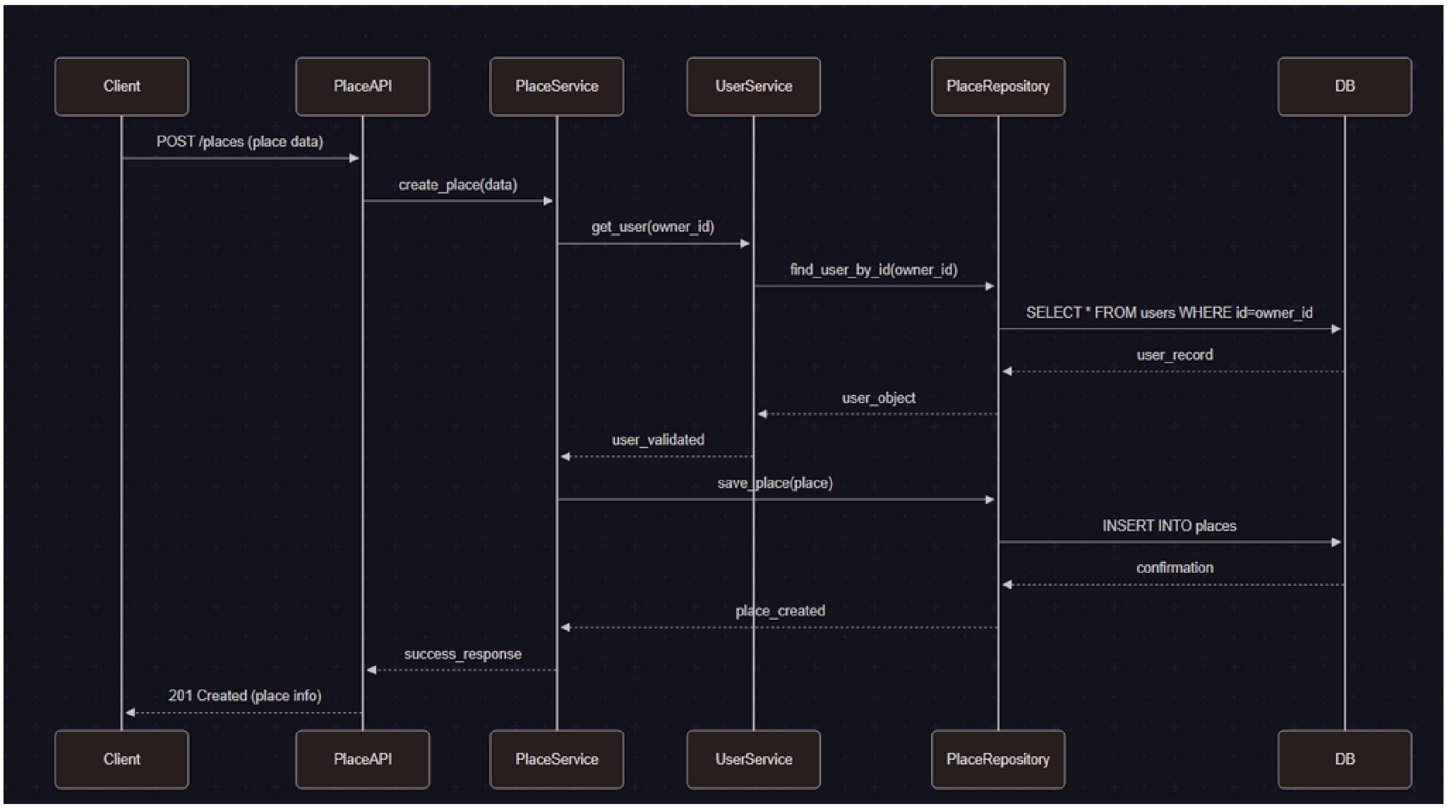
Task-2-1User Registration

This sequence diagram outlines the user registration process. The Client initiates a POST /register request with user data to the UserAPI. The API forwards this to the UserService, which first validates the input. Upon successful validation, the service calls the UserRepository to persist the new user. The repository interacts with the Database to insert the user record, receives confirmation, and returns control to the service. The service responds with a success message to the API, which finally returns a 201 Created response with user information to the client. This flow clearly separates responsibilities across layers and ensures data validation and persistence.



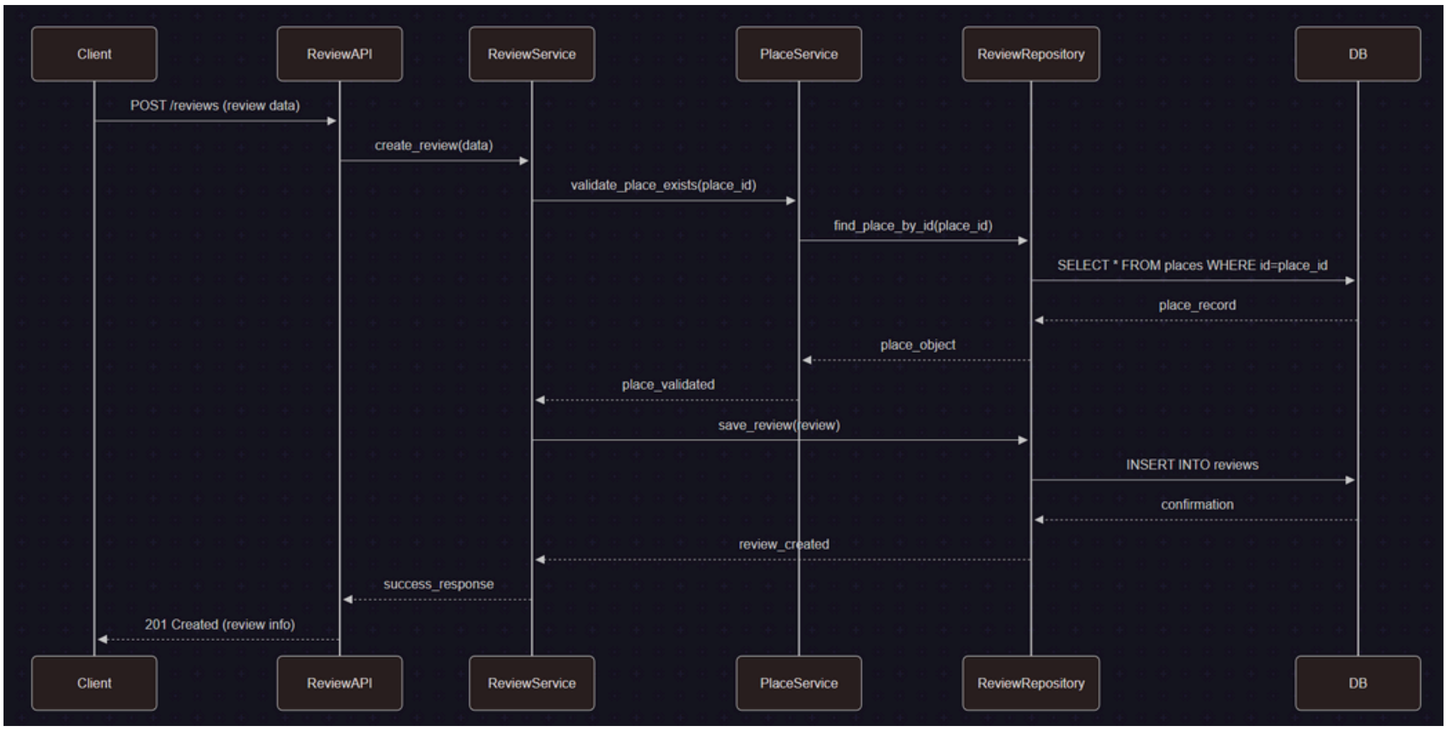
Task-2-2Place Creation

This sequence diagram represents the flow for creating a new place. The Client sends a POST /places request with place data to the PlaceAPI, which delegates the request to PlaceService. To ensure the place owner is valid, PlaceService calls UserService to fetch the user by owner_id. The UserService interacts with the PlaceRepository, which queries the Database for the user record. Once validated, PlaceService saves the new place by calling the repository, which inserts the record into the database. After confirmation, a success response is sent back through the layers, and the client receives a 201 Created status with the new place information.



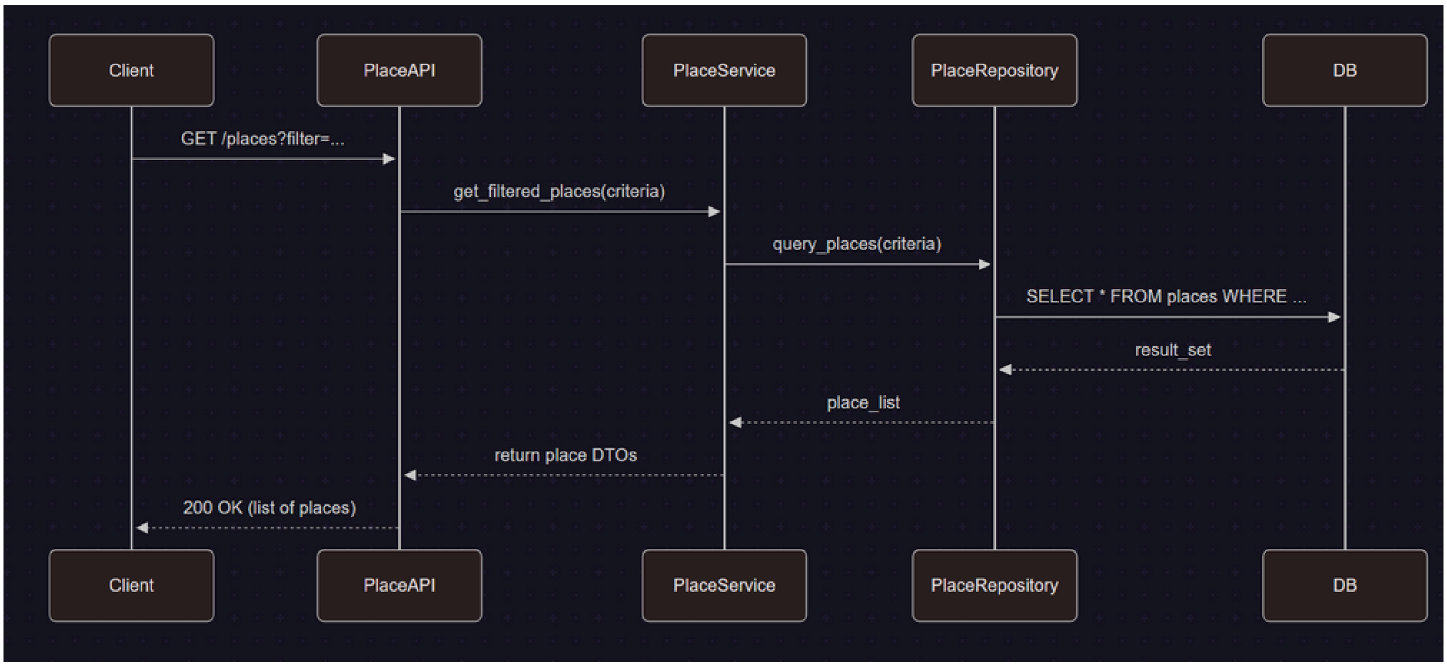
Task-2-3Review Submission

This sequence diagram illustrates the process of submitting a new review. The Client sends a POST /reviews request with review data to the ReviewAPI, which passes the data to the ReviewService. To ensure the review is associated with a valid place, the service calls PlaceService to validate the existence of the given place_id. The PlaceService queries the ReviewRepository, which interacts with the Database to retrieve the place record. Once the place is validated, ReviewService proceeds to save the review via the repository, which performs an INSERT into the reviews table. After database confirmation, the success response flows back to the client with a 201 Created status and the review details.



Task 2-4 Fetching a List of Places

This sequence diagram shows the flow for retrieving a filtered list of places. The Client sends a GET /places request with filter parameters to the PlaceAPI, which delegates the request to the PlaceService. The service calls the PlaceRepository to query the database using the provided criteria. The repository executes a SQL SELECT statement against the Database and retrieves the matching records. These results are returned as a list to the service, which transforms them into DTOs (Data Transfer Objects) and sends them back through the API. The client receives a 200 OK response along with the filtered list of places.



Conclusion

This technical document offers a structured and in-depth overview of the HBnB project, capturing its architectural foundation, essential business logic, and key API workflows. It begins by establishing a well-defined layered architecture that promotes modular design and clear separation of responsibilities—starting from a Presentation Layer interfacing through a facade to the core Business Logic Layer, which communicates with the underlying Persistence Layer.

The class diagram for the Business Logic Layer breaks down the main components—User, Place, Review, and Amenity—highlighting their properties and interconnections that drive the system’s core features. Complementing this, the sequence diagrams for core use cases like user registration, place creation, review handling, and filtered search provide a step-by-step depiction of how components interact across layers to process data and execute logic.

Serving as both a conceptual roadmap and practical reference, this document supports all phases of the HBnB project—from development and testing to future updates. By presenting both high-level structure and detailed behavior, it fosters alignment among team members, reinforces consistent implementation, and ensures the system remains scalable, maintainable, and efficient.