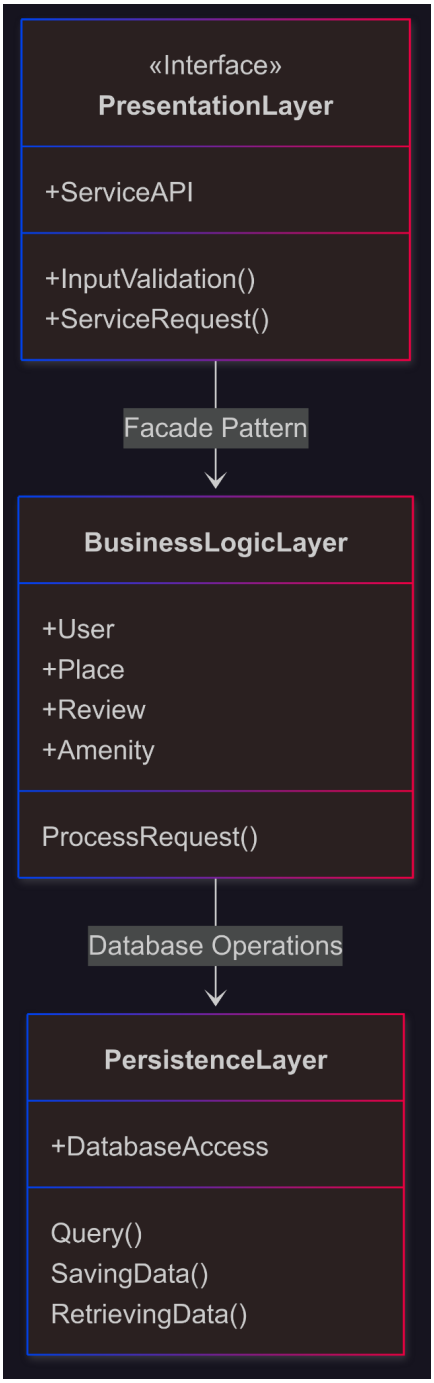
HBnB Evolution - Technical Documentation

This technical document provides a structured overview of the HBnB Evolution project, a simplified Airbnb-like application. It outlines the architecture, entities, and data flow for core features. The system follows a layered architecture: Presentation, Business Logic, and Persistence. This documentation will guide implementation and ensure consistency.

High-Level Architecture



This diagram represents the overall architecture of the HBnB application, broken into three conceptual layers. It serves as a blueprint for how data and control flow through the system, and how responsibilities are cleanly divided using the facade pattern.

The application is divided into three layers:

Presentation Layer:
- API interfaces. This is where processes like input validation and service request are located, what the user or frontend sees/touches. Basically, this layer “talks to the outside world”.

Business Logic Layer:
- Core logic and data models. This layer holds all the rules and actions of the app. Host for the entities like User, Place, Review and Amenity.

Persistence Layer:
- Data handling through repository classes. This part talks to the database. Responsible for saving or getting info from the database.

This design uses the Facade Pattern to isolate APIs from internal complexity, which just means "they don’t talk directly to all internal parts since they use a middleman interface”.

Business Logic Layer - Class Diagram

This diagram details the structure of the main entities (classes) in the Business Logic Layer, including their attributes, methods, and relationships. These classes encapsulate the core data and logic of the HBnB system.

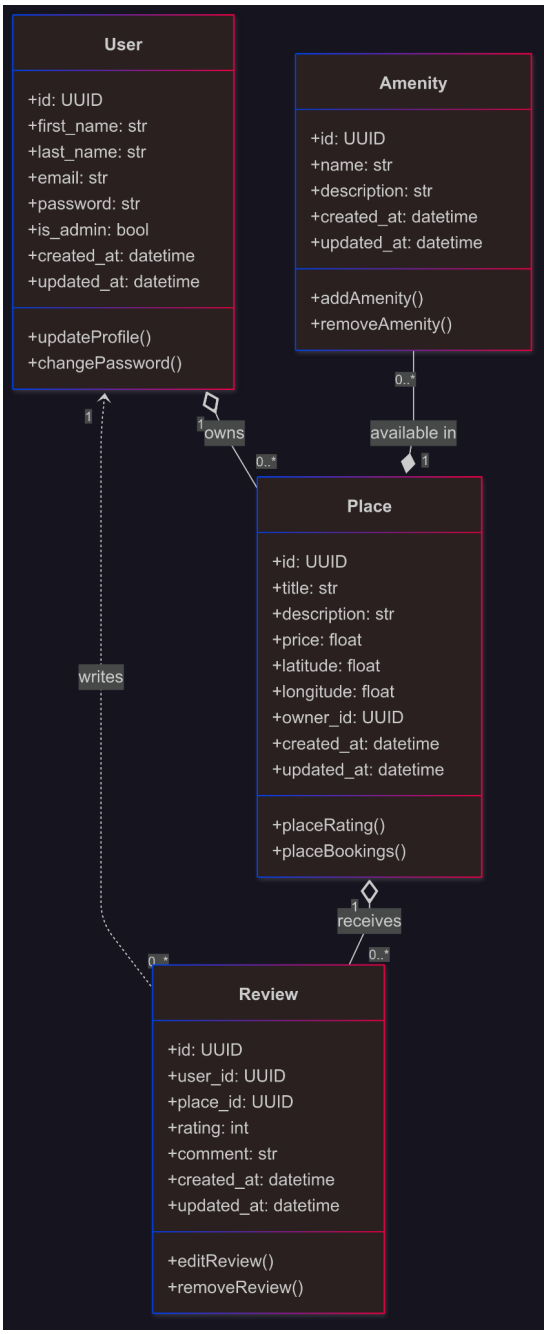
Entities Overview:

User
Represents a person using the application.
- Attributes include standard user info, admin status, and timestamps.
- Methods allow for updating a profile and changing passwords.
- A user can own multiple places and write multiple reviews.

Place
Represents a property listed by a user.
- Stores descriptive info, price, and location coordinates.
- owner_id links it to a User.
- Methods handle ratings and bookings.
- A place can receive multiple reviews and contain multiple amenities.

Review
Represents user feedback for a place.
- Tied to both a User and a Place.
- Includes a numeric rating and a comment.
- Methods support editing and removing the review.

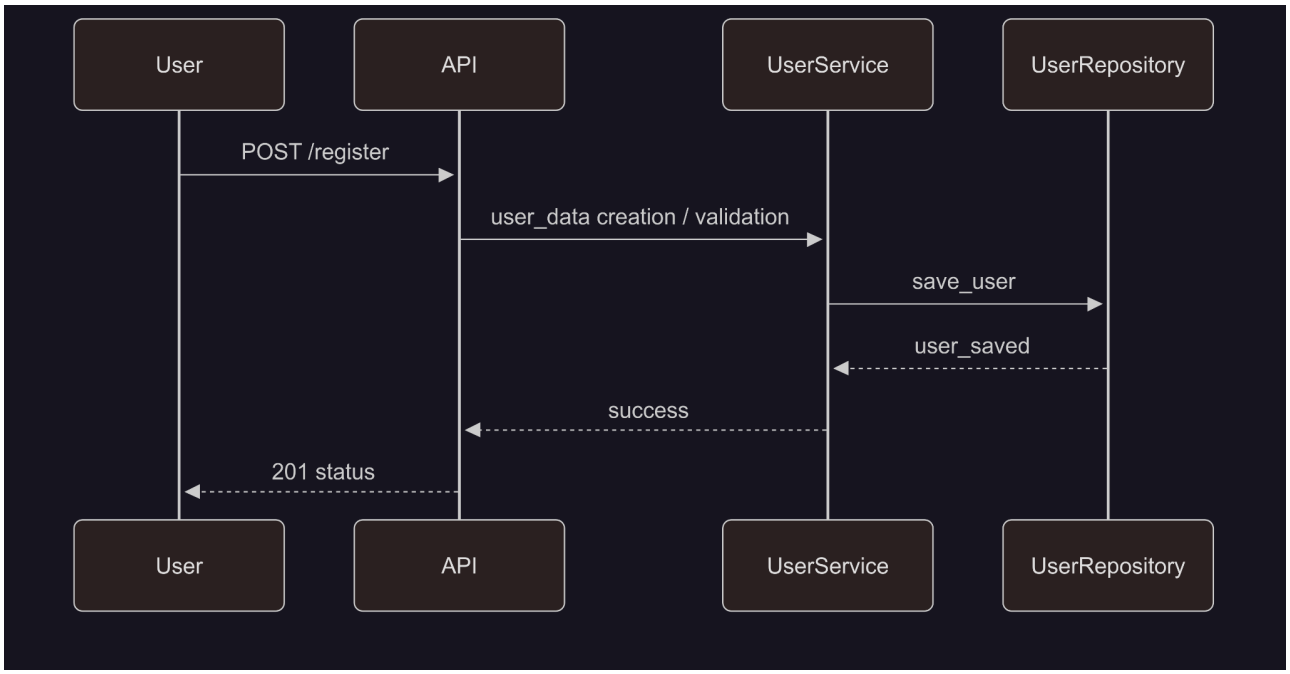
Amenity
Represents features like Wi-Fi, pool, etc.
- Can be linked to multiple places.
- Methods allow adding or removing amenities.



Key Relationships:
- User owns Place: 1-to-many Relationship (Aggregation).
- User writes Review: 1-to-many Relationship(Aggregation).
- Place receives Review: 1-to-many Relationship(Aggregation).
- Place has many Amenity 1-to-many Relationship(Composition).

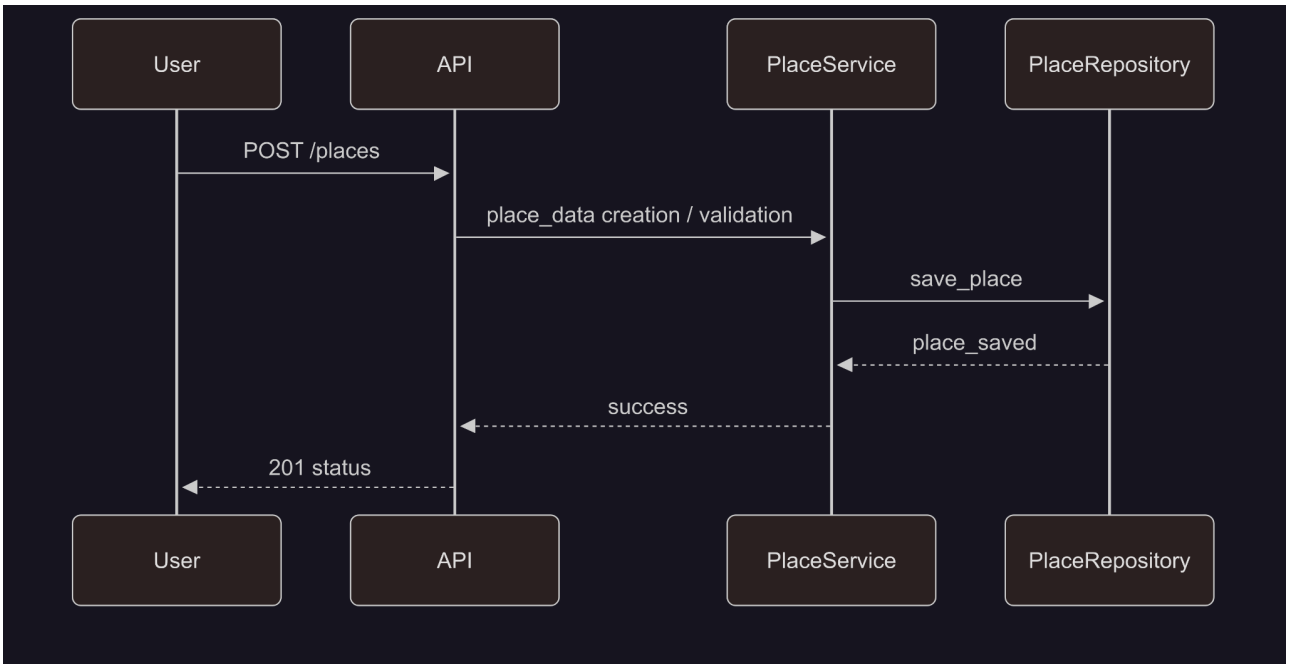
API Interaction Flow - Sequence Diagrams

Shows the step-by-step process of what happens during an API request. They bring the Package Diagram to life by showing how real actions flow through the system.



User Registration

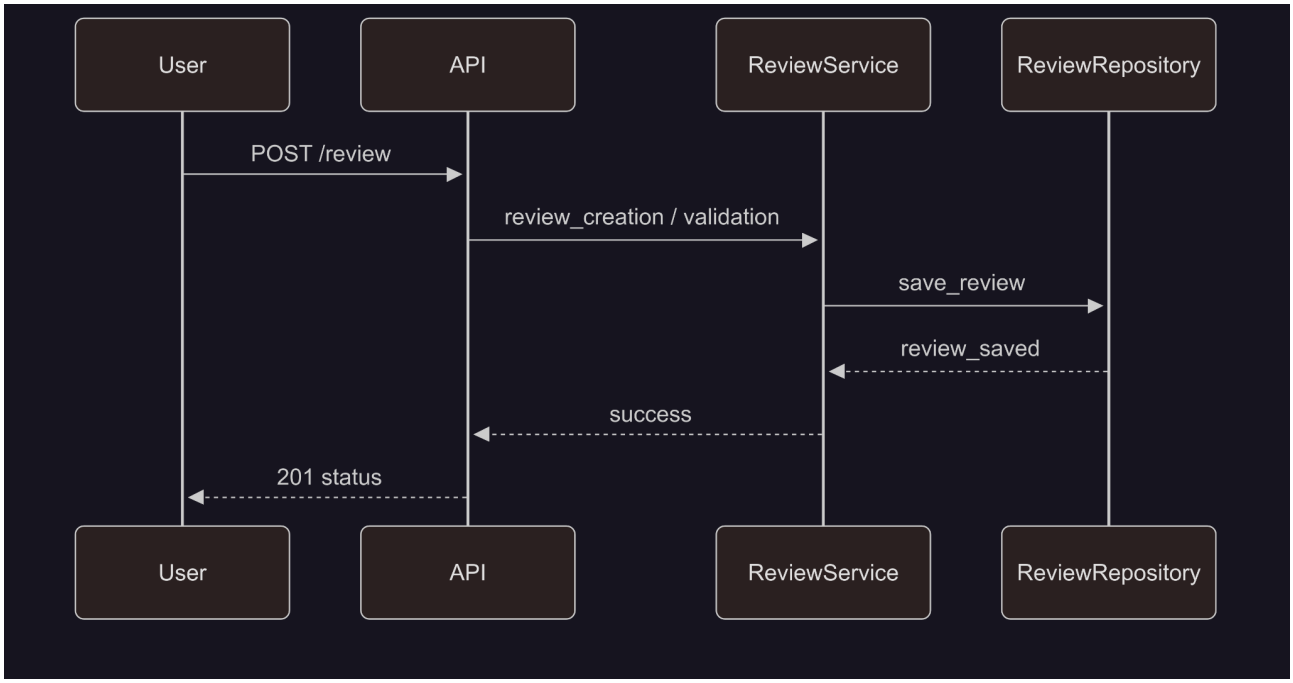
This sequence diagram shows the step-by-step flow of a user registration request — from when a user submits the form to when their information is saved in the database and a success response is returned.



Place Creation

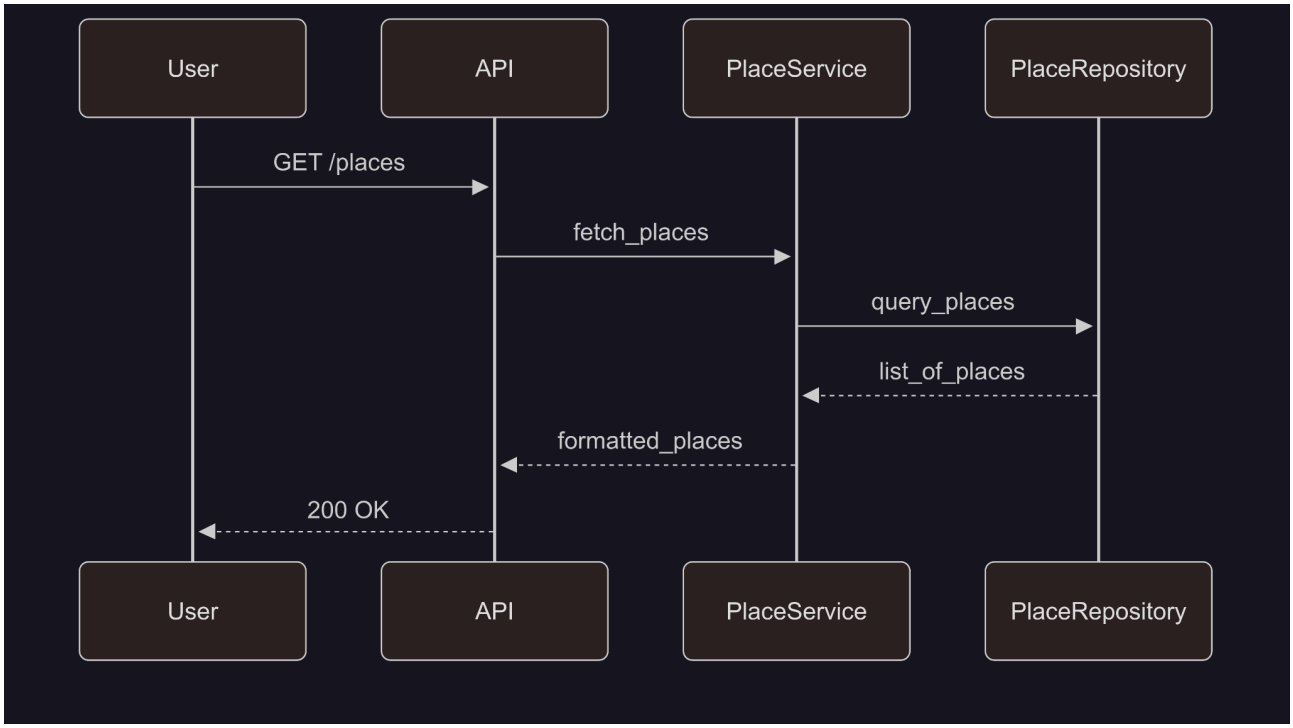
This diagram describes the process that occurs when a user submits a request to create a new place listing. It tracks the request flow across all layers of the system, ensuring proper separation of responsibilities.

API Interaction Flow - Sequence Diagrams - continued



Review Submission

This diagram demonstrates how a user submits a review for a place, and how the system processes, validates, and stores that review. It illustrates the proper flow of data from the Presentation Layer to the Persistence Layer.



Fetching a List of Places

This diagram shows how a user requests a list of available places and how the system retrieves and returns the data. It represents a read-only operation that traverses all three layers of the architecture.