HBNB Evolution

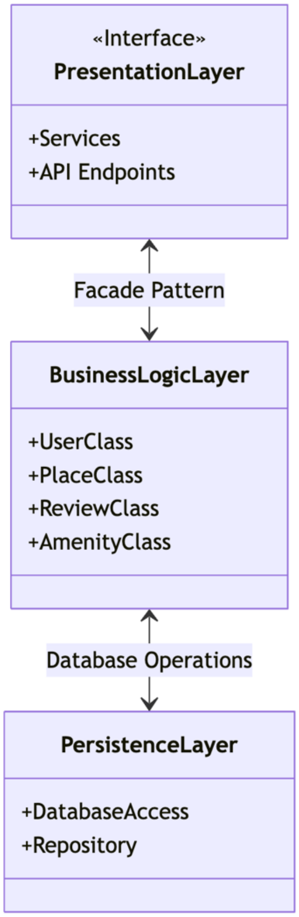
By Pablo Nudel, Julie Tolve, Gabriel Cohen, Ahmed Hmeyd

Introduction

HBnB Evolution is a personalized recreation of the online marketplace for short-and-long term homestays AirBnB. In this document we present the three-layer architecture for the making of this website, its entities (User, Place, Review, Amenities) and the flow of interactions, for some API Calls, across the different layers of the application.

1. High-Level Package Diagram

A high-level package diagram that illustrates the three-layer architecture of the HBnB application and the communication between these layers via the facade pattern. This diagram provides a conceptual overview of how the different components of the application are organized and how they interact with each other.



- **Presentation Layer**
  This includes the services and API through which users interact with the system.

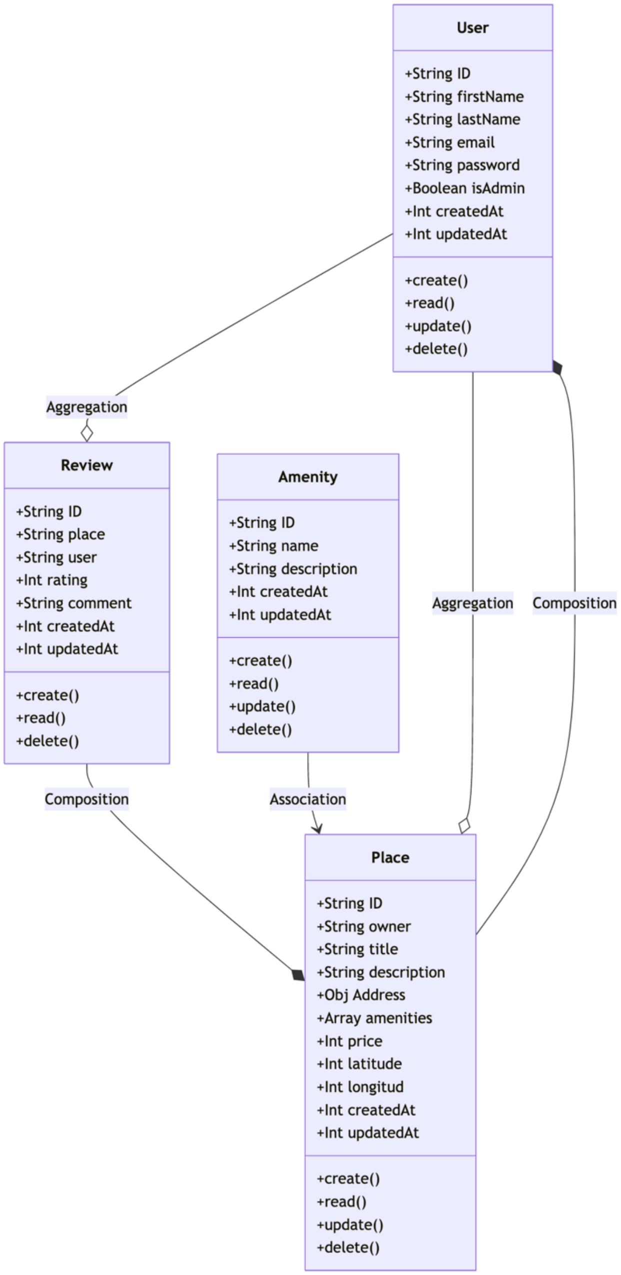
- **The Facade Pattern**
  The Facade pattern provides a simplified interface to a complex subsystem, abstracting its complexity and providing a higher-level interface for clients.

- **Business Logic Layer**
  This layer contains the core business logic and the models that represent the entities in the system.

- **Persistence Layer**
  This layer is responsible for data storage and retrieval, interacting directly with the database.

2. Class Diagram for Business Logic Layer

A detailed class diagram for the Business Logic layer of the HBnB application. This diagram depicts the entities within this layer, their attributes, methods, and the relationships between them.



Entities

- **User Entity**

Each user has a first name, last name, email, and password Users should be able to register (user), update (user) their profile information, and be deleted (user/admin).

- **Place Entity**

Each place has a title, description, price, latitude, and longitude. Places are associated with the user who created them. Places can have a list of amenities Places can be created (user), updated (user), deleted (user/admin), and listed (user/admin).

- **Review Entity**

Each review is associated with a specific place and user, and includes a rating and comment. Reviews can be created (user), deleted (user/admin), and listed (user/admin) by place.

- **Amenity Entity**

Each amenity has a name, and description. Amenities can be created (admin), updated (admin), deleted (admin), and listed (user/admin).

Relationships

- **Association** ——→

Represents a bi-directional relationship between two classes. It establishes a connection between objects of the two classes. In this diagram each place could have a list of amenities.

- **Aggregation** ——◇

Represents a “whole-part” relationship. It’s a weaker form of association where one class (the whole) contains objects of another class (the part), but the part can exist independently of the whole. In this diagram each place should have an owner and each review should be written by a user.

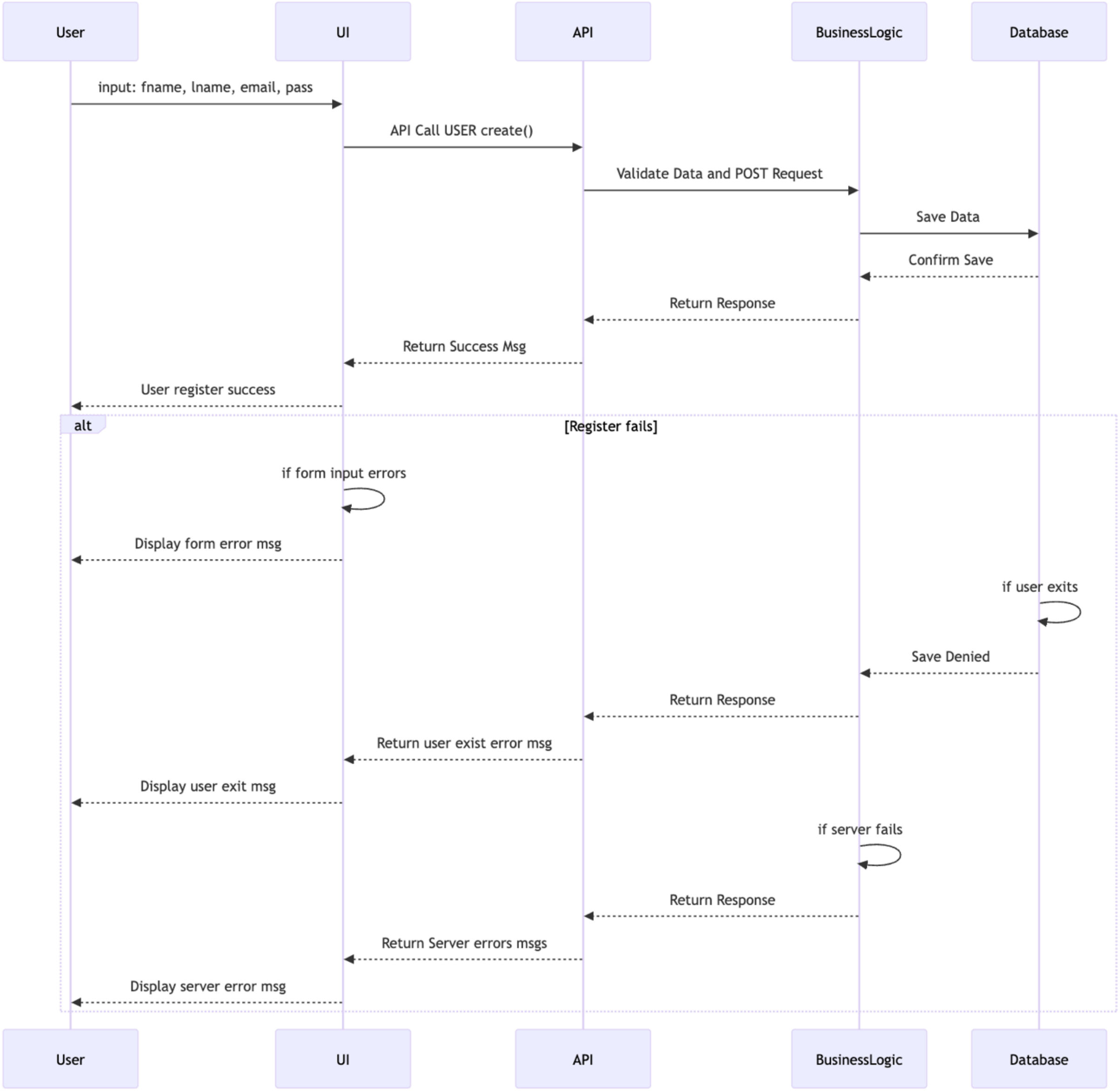
- **Composition** ——◆

A stronger form of aggregation. Represents a “whole-part” relationship where the part cannot exist without the whole. If the whole is destroyed, the part is destroyed as well. In this diagram if a user is deleted the place that he/she owns is deleted and if a place is deleted all its reviews are deleted.

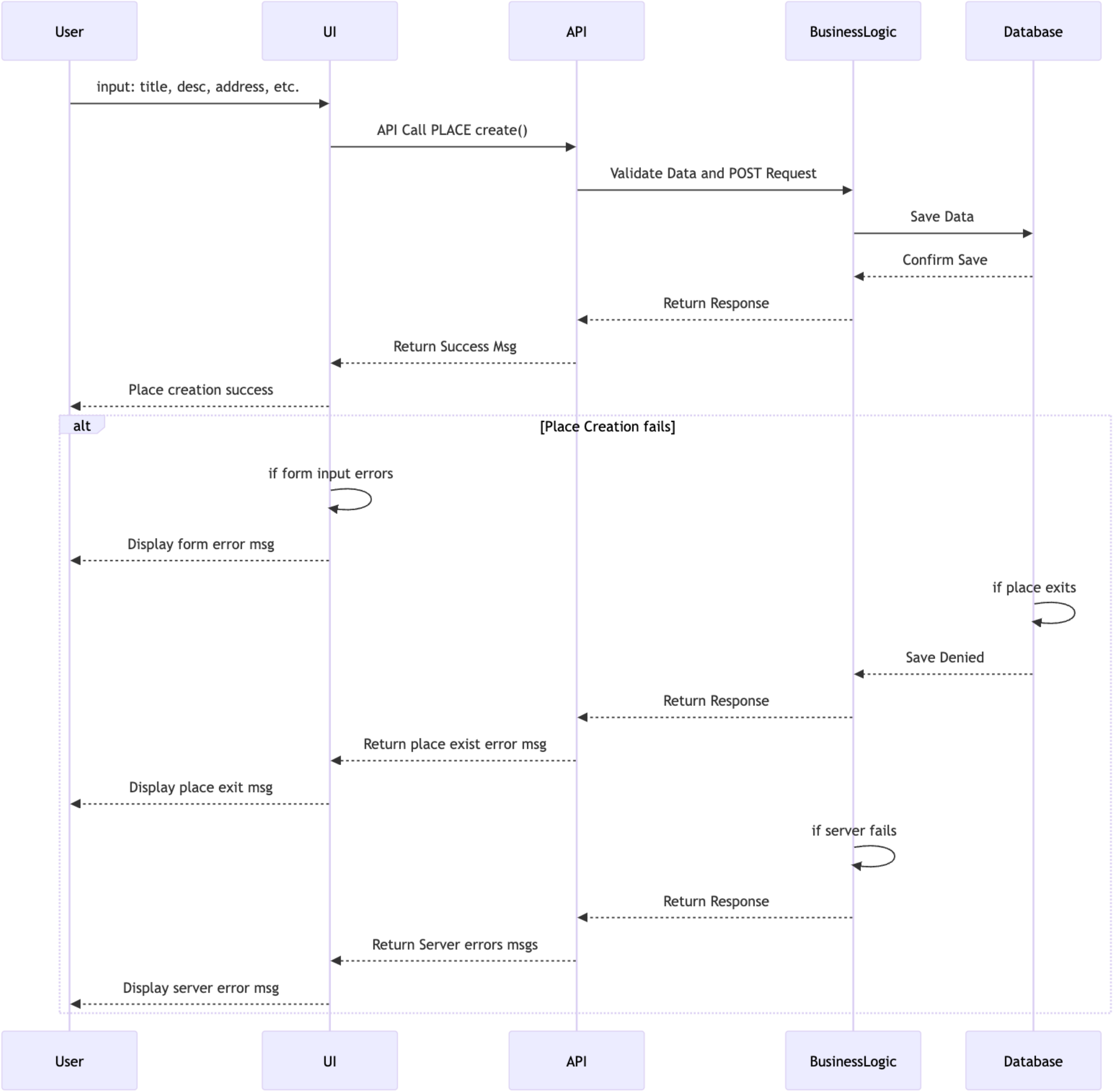
3. Sequence Diagrams for API Calls

Sequence diagrams for some API calls to illustrate the interaction between the layers (Presentation, Business Logic, Persistence) and the flow of information within the HBnB application.

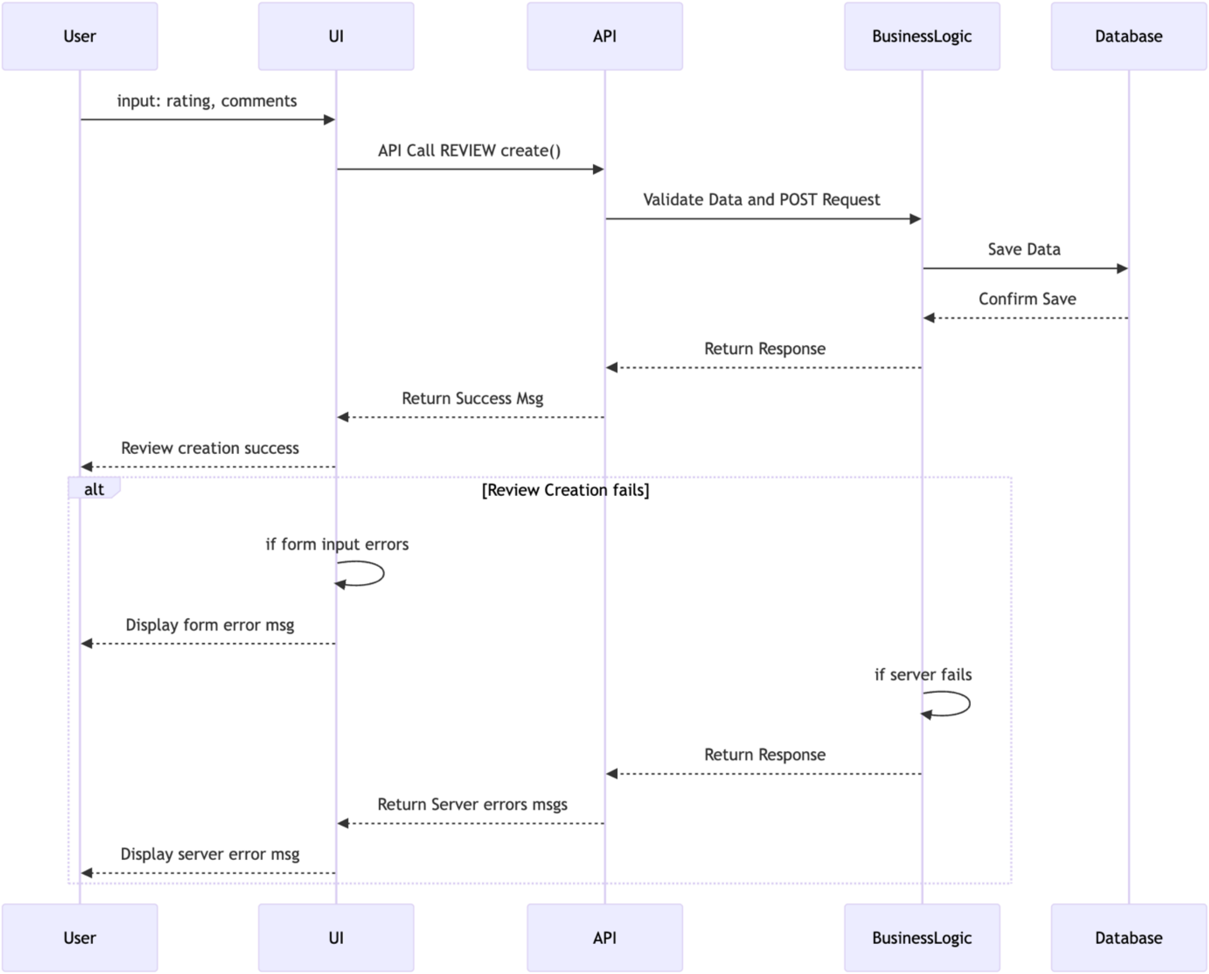
User Registration



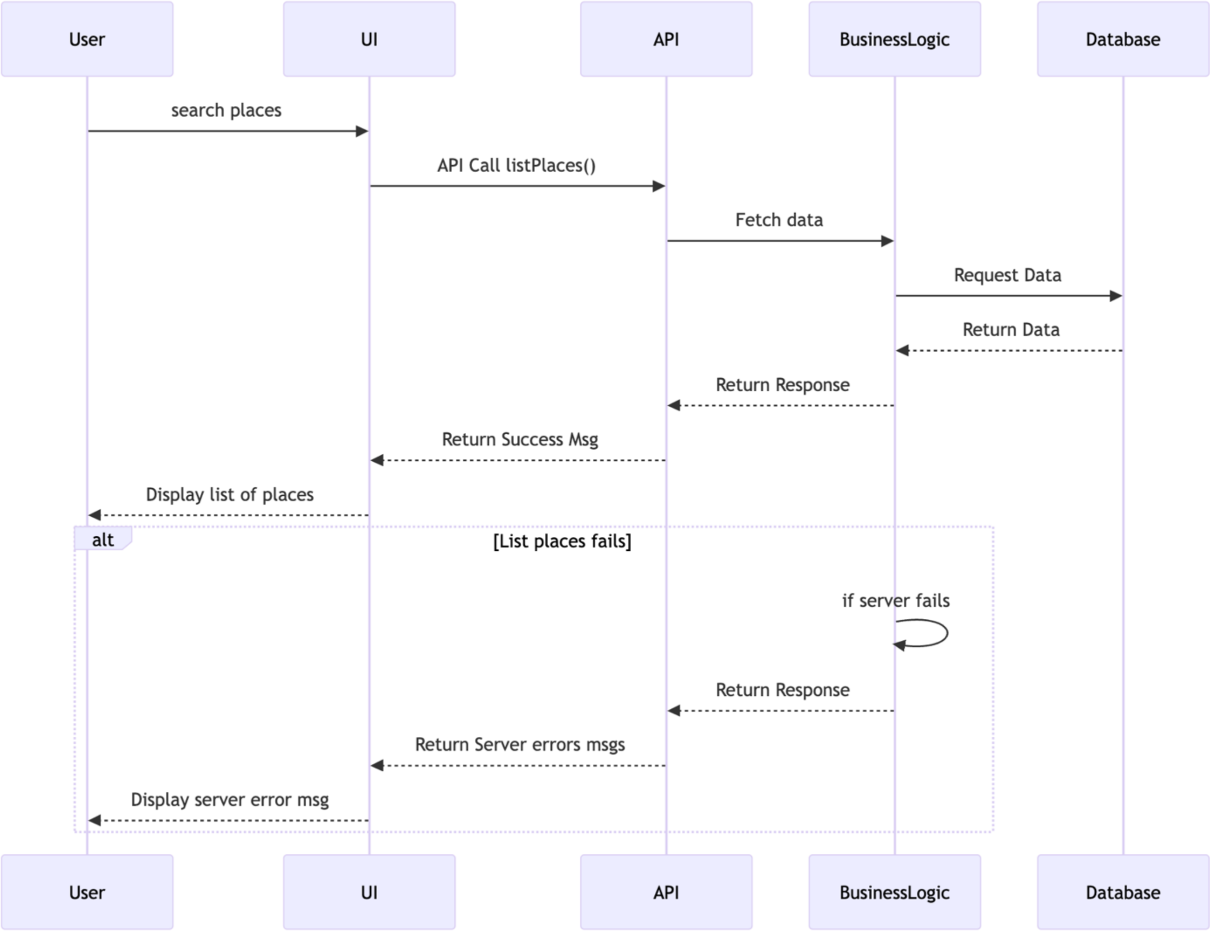
Place Creation



Review Submission



Fetching a List of Places


User
UI
API
BusinessLogic
Database
search places
API Call listPlaces()
Fetch data
Request Data
Return Data
Return Response
Return Success Msg
Display list of places
alt
[List places fails]
if server fails
Return Response
Return Server errors msgs
Display server error msg
User
UI
API
BusinessLogic
Database