# HBNB

## INTRODUCTION

This documentation presents a detailed exploration of the **HBNB project architecture**, a system designed to evolve from a simple command-line interpreter into a fully dynamic web application.
To make the structure and interactions of the project clear, the documentation is organized around **three complementary diagrams**, each illustrating a different layer of the system.

Together, these diagrams reveal how the components depend on one another, how data flows through the application, and how the project progressively transitions from local storage to a scalable, database-backed web platform.

### PACKAGE DIAGRAM

- ○ The presentation layer:

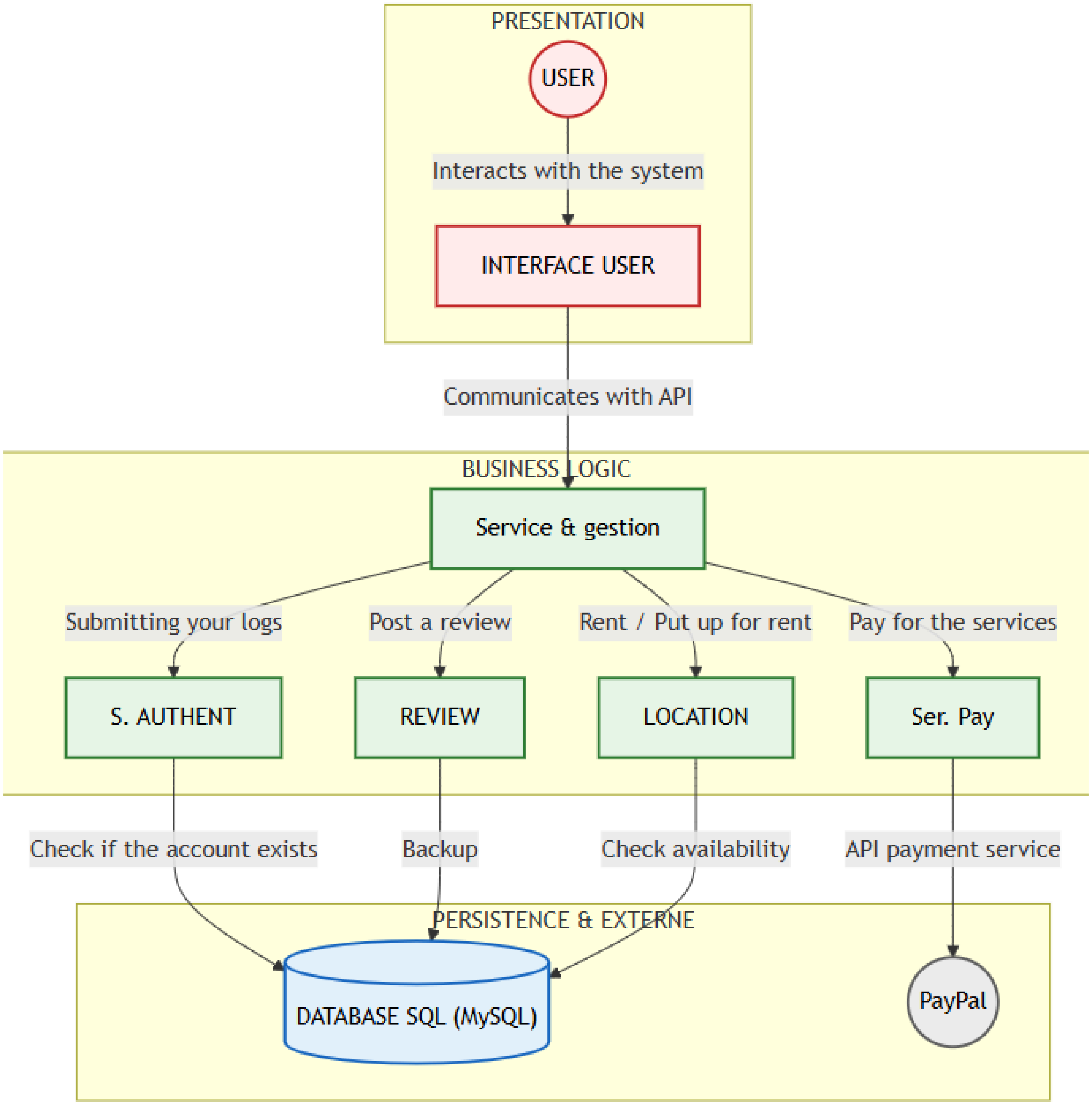
This layer consists of how users communicate with the application (via an interface).

- ○ The business logic layer:

The next layer represents the logic core of the application. It contains every entity and method needed for the system to work properly.

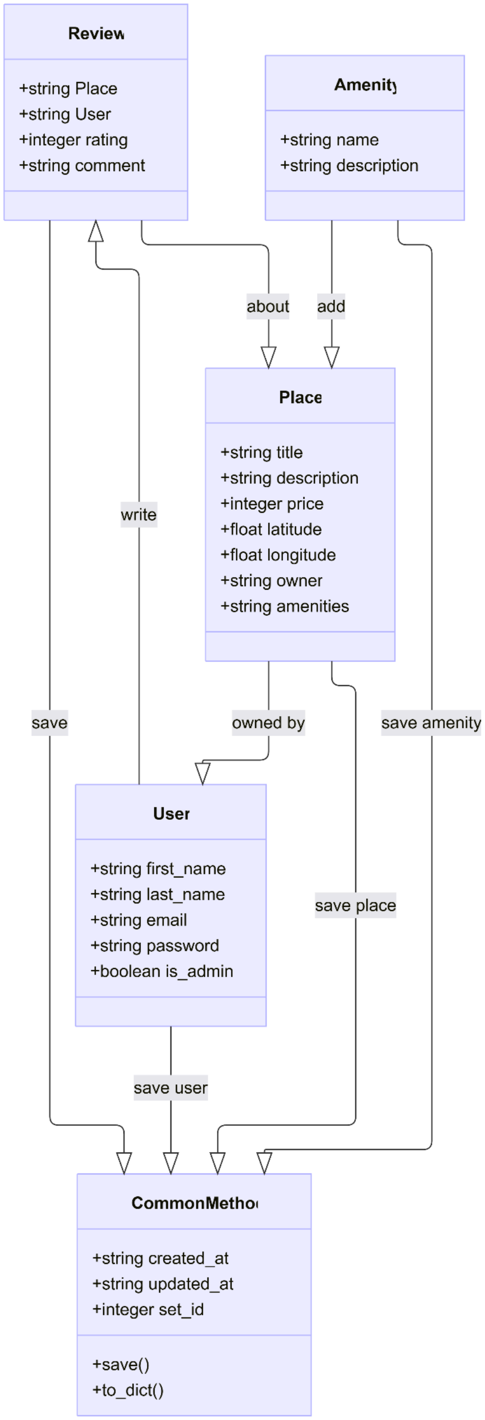
- ○ The persistence layer:

At last, this layer can save or load all data from the system. It shows how each layer communicates with the other one.


PRESENTATION
USER
Interacts with the system
INTERFACE USER
Communicates with API
BUSINESS LOGIC
Service & gestion
Submitting your logs
Post a review
Rent / Put up for rent
Pay for the services
S. AUTHENT
REVIEW
LOCATION
Ser. Pay
Check if the account exists
Backup
Check availability
API payment service
PERSISTENCE & EXTERNE
DATABASE SQL (MySQL)
PayPal

## CLASS DIAGRAM

- The Class diagram is composed of:



The User class: represent a user, he can add a review on a place.

The Place class: represents a place, it can have a review and some amenities.

The Review class: represents the review posted by a user about a place.

The Amenity class: represents the multiple possible amenities that a place can have.

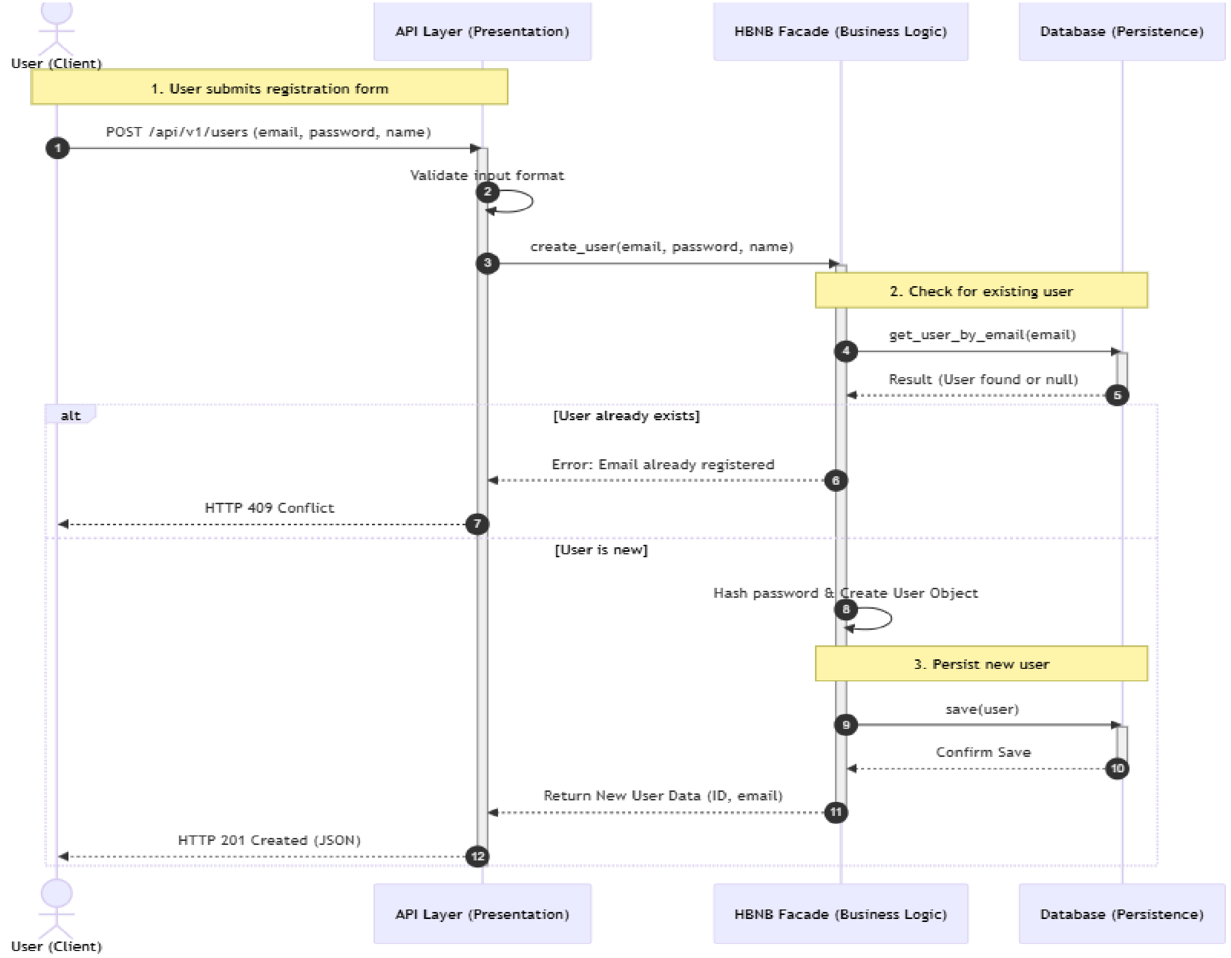
The Common Method class: represents common methods and entities used by all other classes.

# SEQUENCE DIAGRAM

The Sequence diagram represents the use case of the system at work.

It shows, in four specific cases, how each part interacts and communicates with each other.

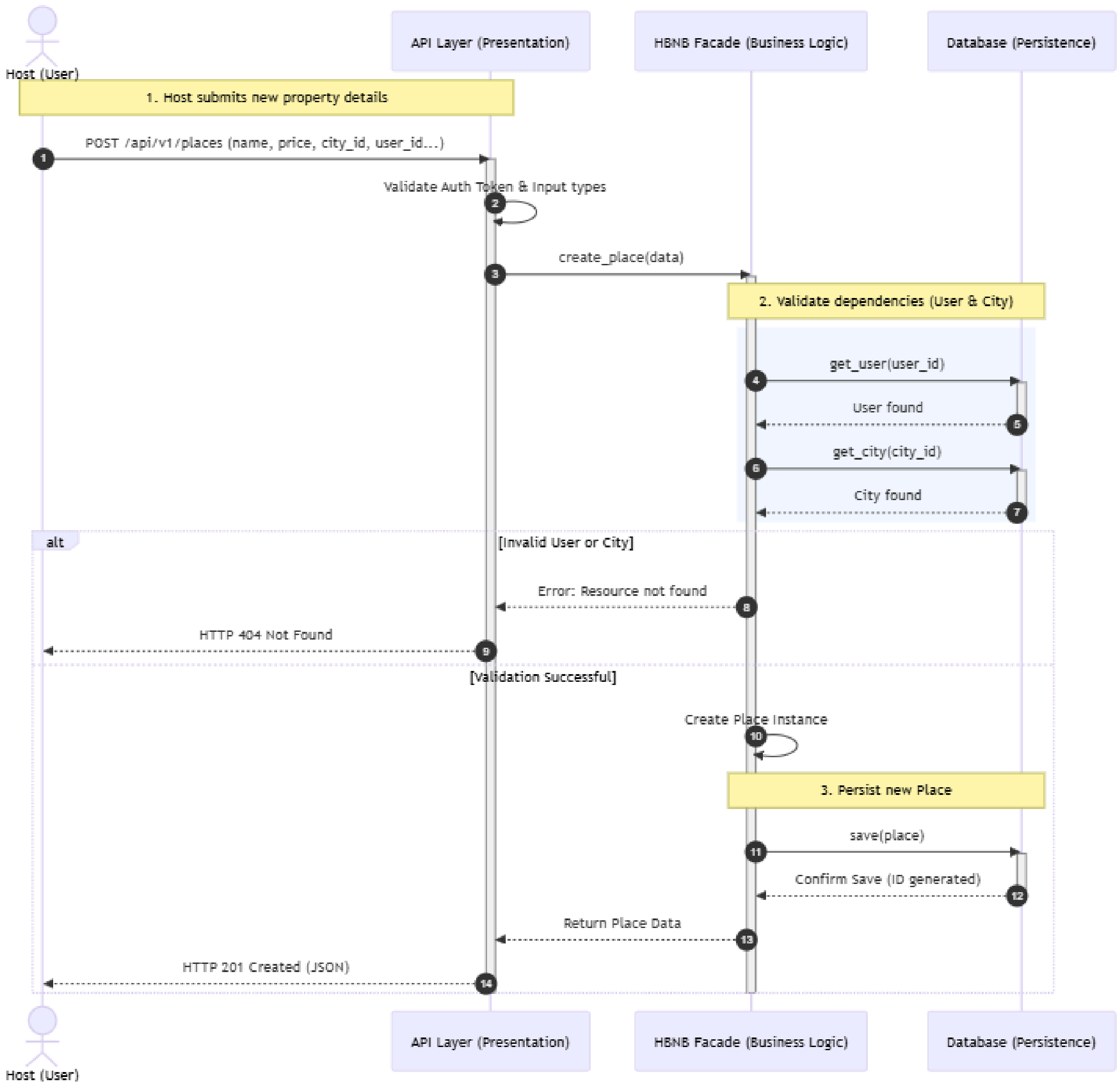
## API 1 : User Registration



User (Client)
API Layer (Presentation)
HBNB Facade (Business Logic)
Database (Persistence)
1. User submits registration form
POST /api/v1/users (email, password, name)
1
Validate input format
2
create_user(email, password, name)
3
2. Check for existing user
get_user_by_email(email)
4
Result (User found or null)
5
alt
[User already exists]
Error: Email already registered
6
HTTP 409 Conflict
7
[User is new]
Hash password & Create User Object
8
3. Persist new user
save(user)
9
Confirm Save
10
Return New User Data (ID, email)
11
HTTP 201 Created (JSON)
12
User (Client)
API Layer (Presentation)
HBNB Facade (Business Logic)
Database (Persistence)

- **User → API** : Sends the registration form.
- **API → HBNB Facade** : Requests user creation.
- **HBNB Facade → Database** : Checks if the email already exists.
- **HBNB Facade → Database** : Saves the new user if the email is free.
- **API → User** : Returns confirmation message with the new user info

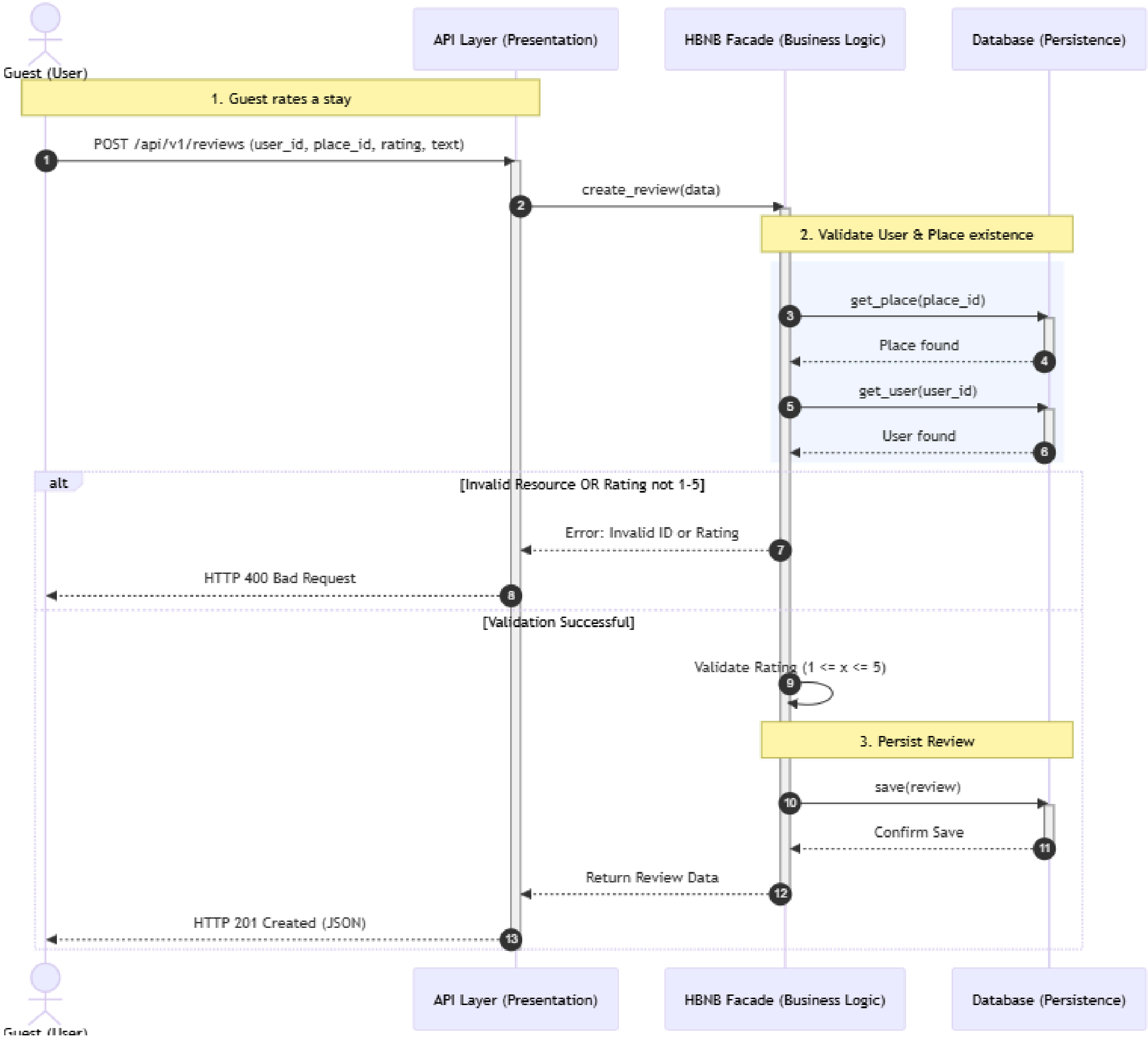
## API 2 : Place Creation



- **User → API :** Requests the list of places (with optional filters).
- **API → HBNB Facade :** Asks for all places matching the filters.
- **HBNB Facade** → Database : Retrieves all Place objects.

- **HBNB Facade** → API : Returns an empty list or a filtered/serialized list.
- **API** → **User :** Sends confirmation message OK with the JSON array

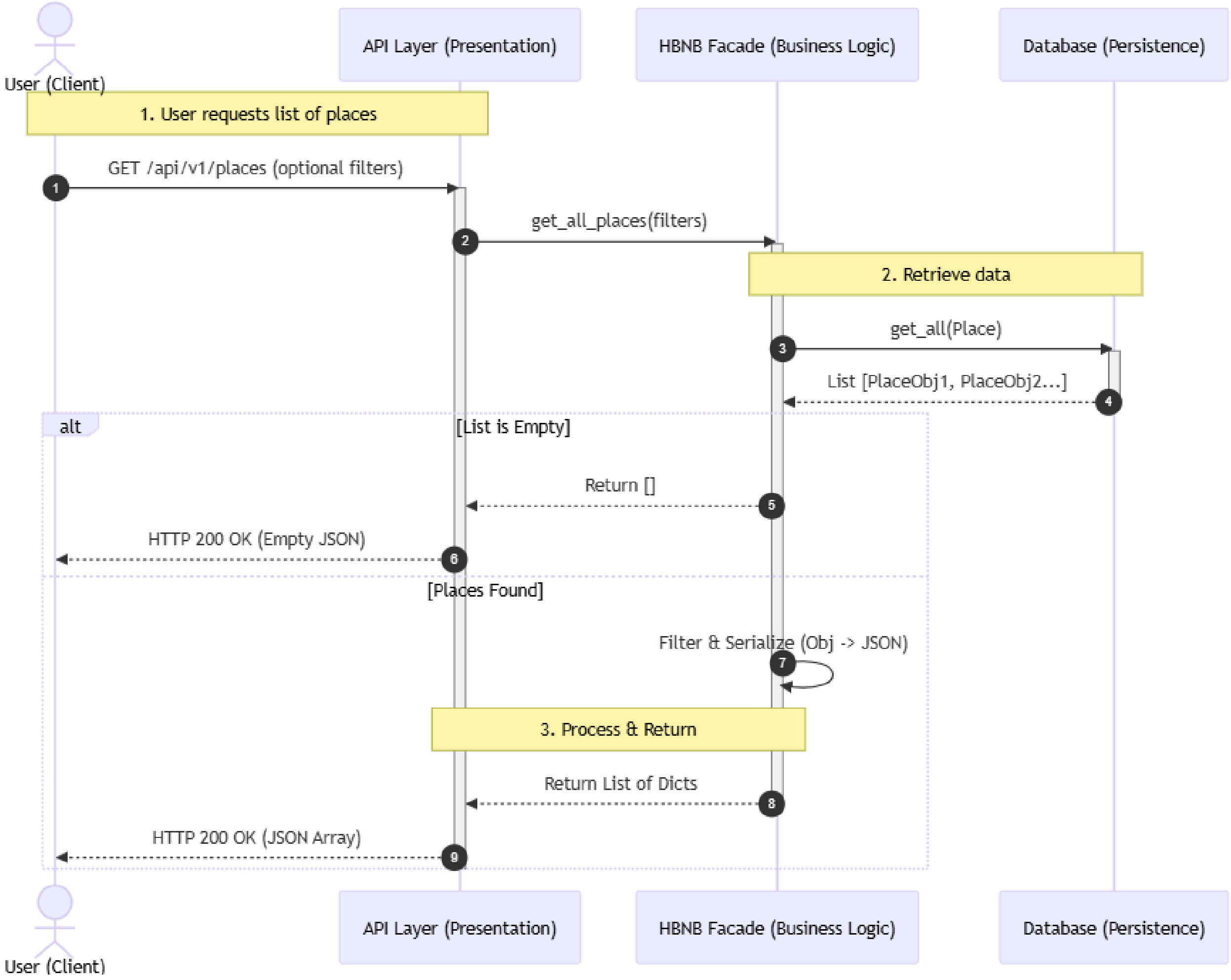
### API 3 : Review Submission



- **Guest** → **API :** Submits a review (user_id, place_id, rating, text).
- **API** → HBNB Facade : Requests review creation.
- **HBNB Facade** → Database : Validates place and user existence.

- **HBNB Facade** → API : Returns error message or saves the review.
- **API** → **Guest :** Sends confirmation message *Created* with the review data.

### API 4 : Fetch Place List



- **Host** → **API :** Sends new property details.
- **API** → **HBNB Facade :** Requests place creation after validation.
- **HBNB Facade** → Database : Checks user and city existence.

- **HBNB Facade** → Database : Saves the new place if validation succeeds.
- **API** → **Host :** Returns confirmation message *Created* with the place data.