HBnB

SUMMARY

Table of Contents

INTRODUCTION

Presentation

HBnB Evolution is a simplified version of an AirBnB-like web application, developed to manage **users**, **places**, **reviews**, and **amenities**. The system is built using a clean **three-layer architecture**:

- **Presentation Layer**: Manages user interaction via services and APIs.
- **Business Logic Layer**: Handles the core logic and entity management.
- **Persistence Layer**: Ensures proper storage and retrieval of data.

This structure ensures **scalability**, **maintainability**, and a clear **separation of concerns** throughout the application.

The project is developed as a **group effort** by:

Daniel Ramirez, Moussa Elisoltanov and Flora Salanson.

This **first phase** focuses on building the **technical documentation** that will serve as the foundation for development. It includes **UML diagrams**, business logic design, and a clear representation of system interactions.

Key Requirements

1. User Entity

- Attributes: first name, last name, email, password, is_admin
- Functionalities: Register, update profile, delete user

2. Place Entity

- Attributes: title, description, price, latitude, longitude
- Linked to: One user (owner), list of amenities
- Functionalities: Create, update, delete, list

3. Review Entity

- Attributes: rating, comment
- Linked to: One place and one user
- Functionalities: Create, update, delete, list by place

4. Amenity Entity

- Attributes: name, description
- Functionalities: Create, update, delete, list

HIGH LEVEL ARCHITECTURE

Presentation

This class diagram represents a three-layer architecture commonly used in software applications. The **Presentation Layer** is responsible for the user interface and user interaction, as indicated by its method to display user information.

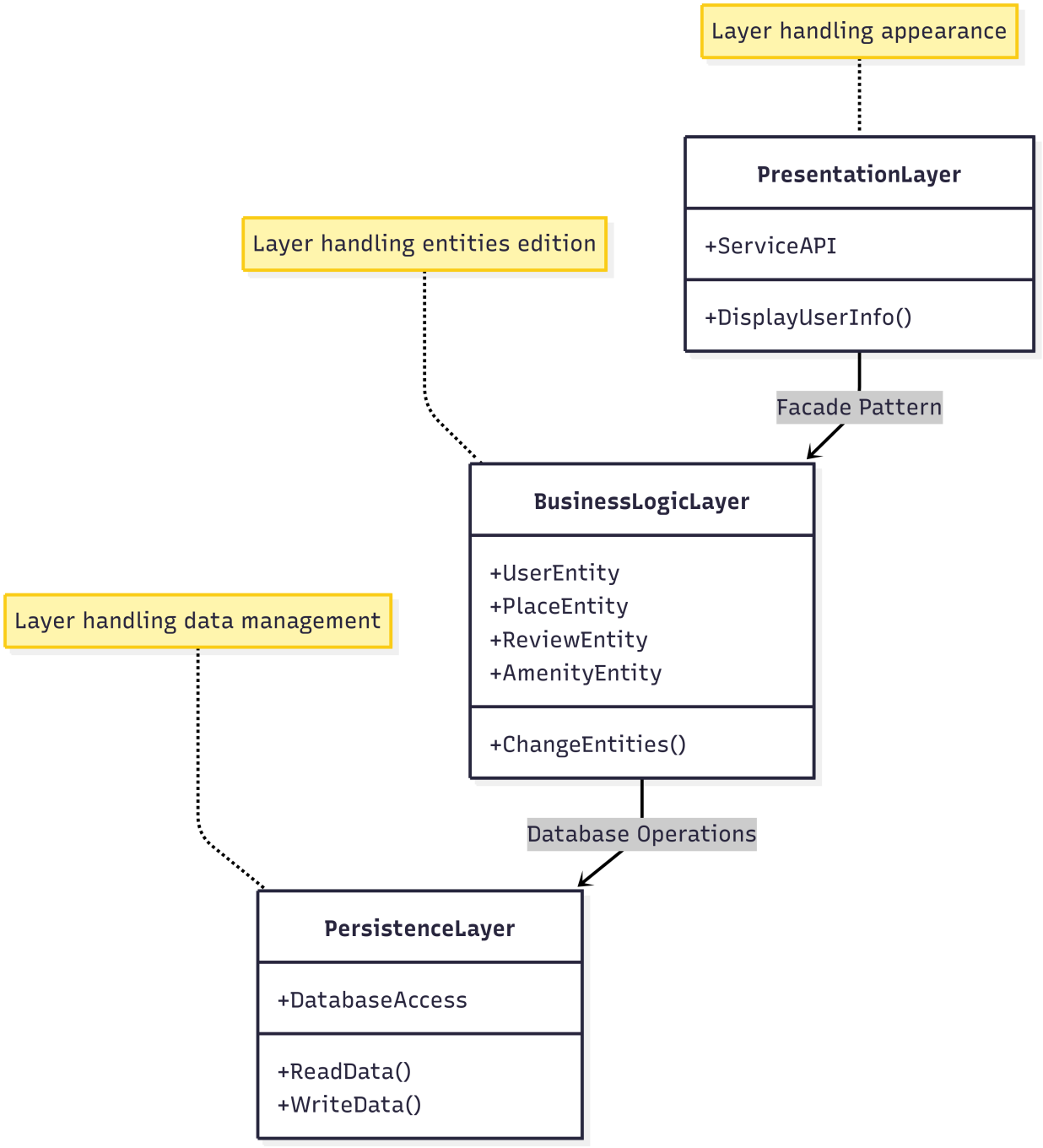
It communicates with the **Business Logic Layer** using the **Facade Pattern**, simplifying access to the application's core functionality.

The **Business Logic Layer** manages the application's core entities such as users, places, reviews, and amenities, and handles operations related to changing these entities.

The **Facade Pattern** is a design pattern used to provide a simplified interface to a more complex subsystem. In this diagram, the Presentation Layer uses a facade to interact with the Business Logic Layer.

Finally, the **Persistence Layer** deals with database operations, including reading and writing data. It is accessed by the Business Logic Layer to persist or retrieve information from the underlying data storage. Each layer has a clear responsibility, promoting separation of concerns and maintainability.

PACKAGE DIAGRAM



BUSINESS LOGIC LAYER

Presentation

This class diagram represents the core entities of the HBnB Evolution application and their relationships.

At the foundation, a **Base Model** class provides common attributes and methods such as unique IDs, creation and update datetimes, and basic CRUD operations.

The **User** class models the application's users, including attributes like name, email, password, admin status, and payment method.

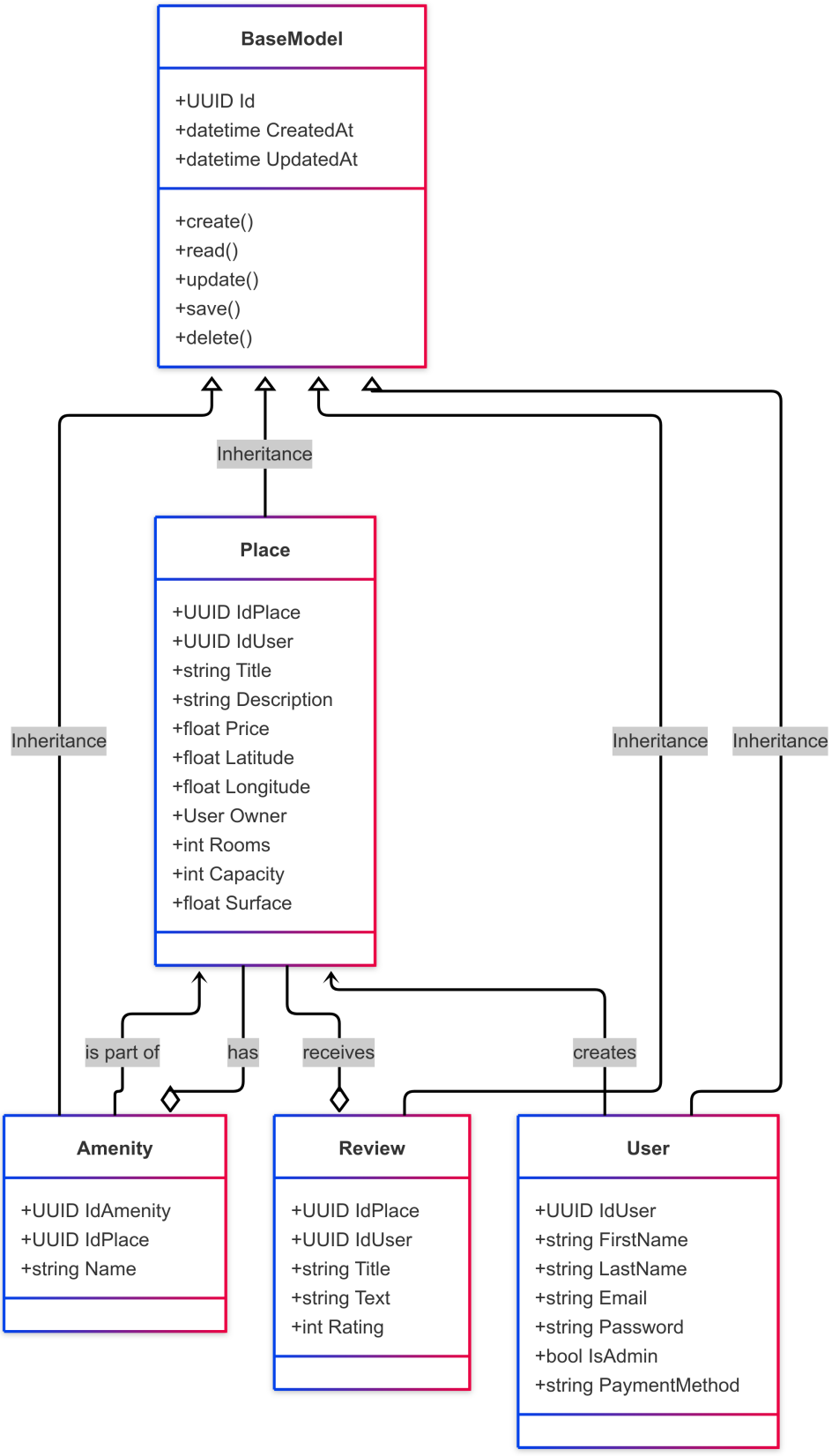
Users can create **Places**, which represent properties with details like title, description, price, location, rooms, capacity, and surface area.

Each **Place** can have multiple **Amenities**, describing features associated with the property, and can receive **Reviews** written by users.

Reviews include a title, text, and rating, and are linked to both the place and the user who created them.

Inheritance from **Base Model** ensures consistency across all entities with shared behaviors and properties.

CLASS DIAGRAM



API INTERACTION FLOW

Presentation

Sequence diagrams are a type of UML diagram used to represent how different parts of a system interact over time.

They show the flow of messages between actors (such as users) and system components (like the API, business logic, and database) during specific operations.

In the context of the HBnB Evolution project, sequence diagrams help illustrate the step-by-step processes behind key features — such as user registration, place creation, and review submission.

These diagrams are essential for understanding the system's behavior, ensuring a clear communication flow, and guiding the development of each interaction.

The following section presents four key sequence diagrams that illustrate the main interactions within the HBnB Evolution application. Each diagram demonstrates how the system handles a specific user action, showing the communication flow between the user, API, business logic, and database layers.

1. **User Registration**
   Describes the process of creating a new user account, including data validation and saving user information to the database.

2. **Place Creation**
   Shows how a registered user can create a new place, with the system validating input and storing the place details.

3. **Review Submission**
   Details the steps a user follows to submit a review for a place, including rating, comment, and linking the review to both the user and the place.

4. **Fetching List of Places**
   Illustrates how the system retrieves and returns a list of available places, including communication from the database to the user interface.

USER REGISTRATION DIAGRAM

This diagram shows the flow of interactions when a user registers on the HBnB Evolution platform:

1. **User → API**: The user submits a registration request.

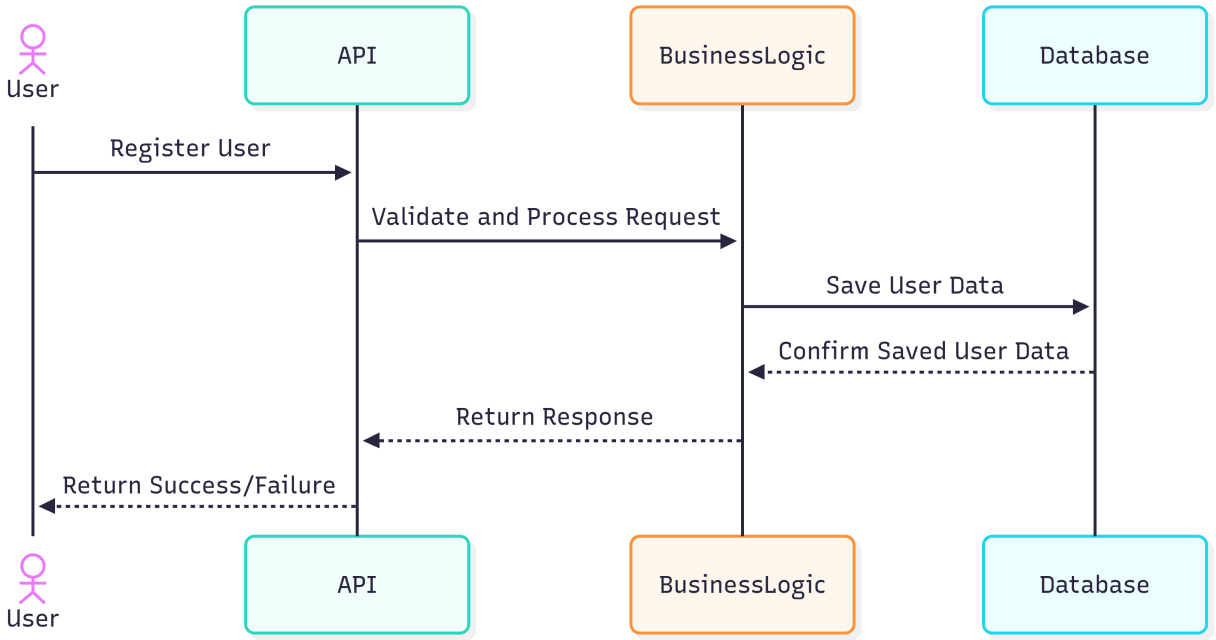
2. **API → BusinessLogic**: The API forwards the request to the business logic layer for validation and processing.

3. **BusinessLogic → Database**: Once validated, the business logic layer sends the user data to be saved in the database.

4. **Database → BusinessLogic**: The database confirms that the user data has been successfully saved.

5. **BusinessLogic → API**: The business logic sends a confirmation or error back to the API.

6. **API → User**: The API returns a success or failure message to the user based on the outcome.



PLACE CREATION DIAGRAM

This sequence diagram outlines the process of creating a new place in the HBnB Evolution application:

1. **User → API**: The user sends a request to create a new place.

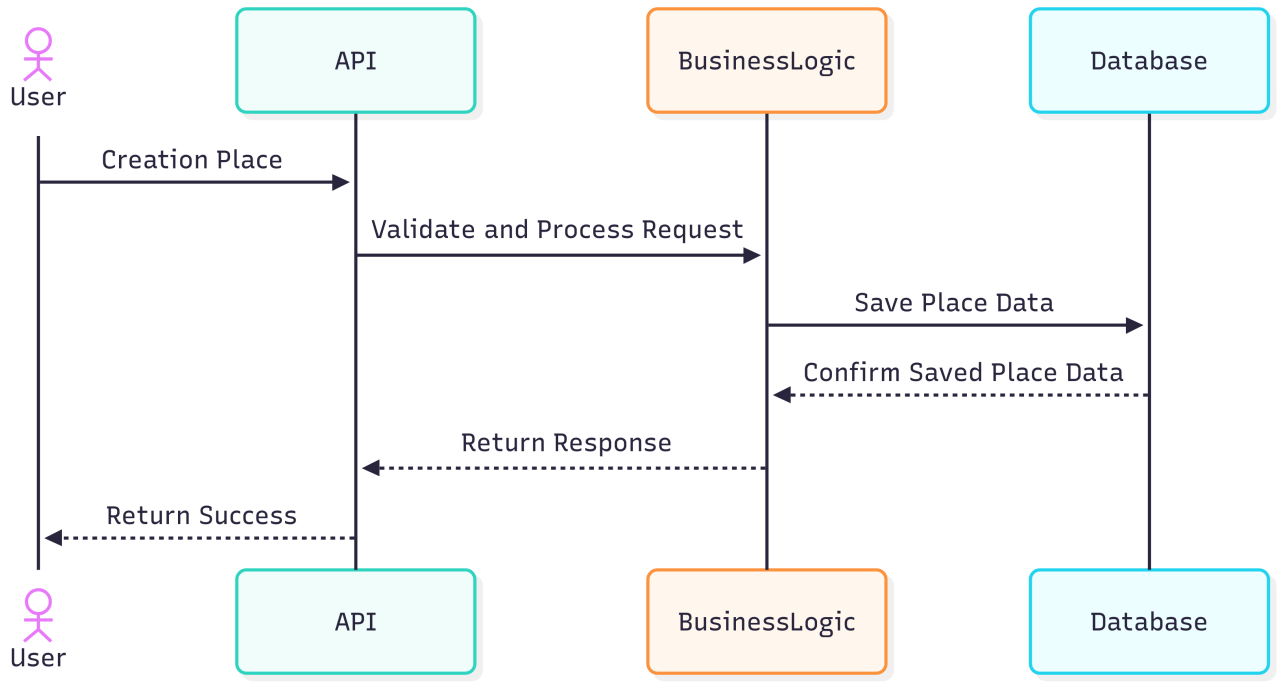
2. **API → BusinessLogic**: The API forwards the request to the business logic layer, which validates the data and processes the creation request.

3. **BusinessLogic → Database**: The business logic layer sends the new place data to be stored in the database.

4. **Database → BusinessLogic**: The database confirms that the place data has been successfully saved.

5. **BusinessLogic → API**: A response indicating success or failure is returned to the API.

6. **API → User**: The API sends the final response to the user confirming whether the place was created successfully or not.



REVIEW SUBMISSION DIAGRAM

This diagram outlines the steps involved when a user submits a review for a place in the HBnB Evolution application:

1. **User → API**: The user submits a review, including details like rating and comment.

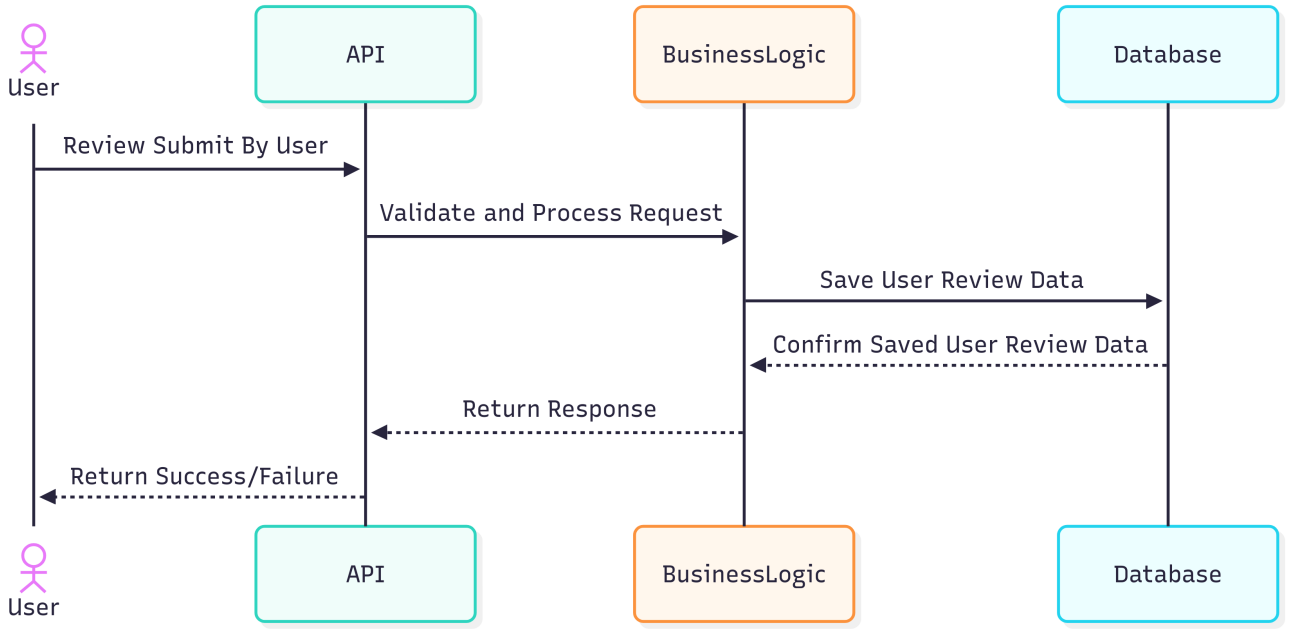
2. **API → BusinessLogic**: The API sends the request to the business logic layer, which validates the data and processes the review.

3. **BusinessLogic → Database**: The validated review data is sent to the database to be saved.

4. **Database → BusinessLogic**: The database confirms that the review has been successfully stored.

5. **BusinessLogic → API**: A response is returned to the API indicating the success or failure of the operation.

6. **API → User**: The API sends a final confirmation back to the user.



PLACE CRITERIA DIAGRAM

This diagram shows the process when a user requests a list of places filtered by specific criteria:

1. **User → API**: The user sends a request with criteria to search for places.

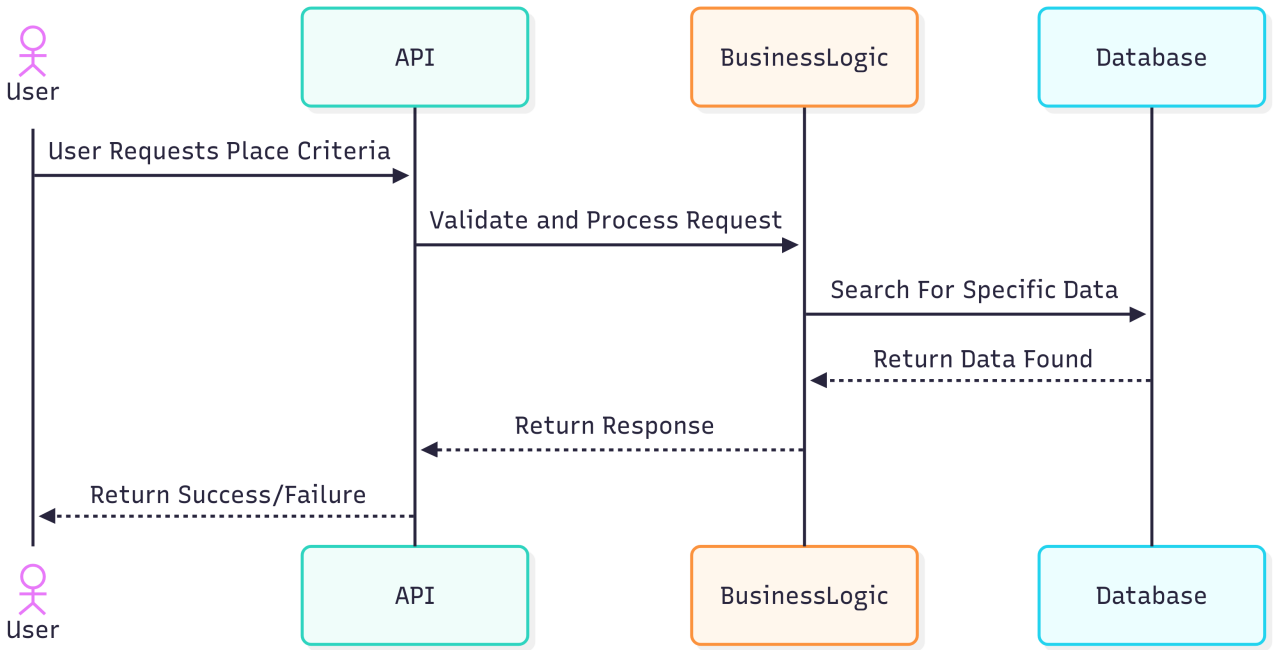
2. **API → BusinessLogic**: The API forwards the request to the business logic layer for validation and processing.

3. **BusinessLogic → Database**: The business logic queries the database to find matching places.

4. **Database → BusinessLogic**: The database returns the list of places that meet the criteria.

5. **BusinessLogic → API**: The business logic sends the results back to the API.

6. **API → User**: The API returns the search results or an error message to the user.



GLOSSARY

- **Presentation Layer**: The top layer of the architecture responsible for interacting with the user. It handles the display and user input.
- **Business Logic Layer**: The core layer that contains the main application logic. It manages entities and processes user requests coming from the presentation layer.
- Sure! Here's a shorter and clear definition in English:
- **API :** is a set of rules that allows different software systems to **communicate** with each other. It lets one program **request or send data** to another, often used to connect the frontend and backend in web applications.
- **Persistence Layer**: The bottom layer that deals with storing and retrieving data from a database or other storage systems.
- **Entity**: An object that represents a specific concept in the application, such as a User, Place, Review, or Amenity.
- **Facade Pattern**: A design pattern used to provide a simplified interface to a more complex subsystem. In this diagram, the Presentation Layer uses a facade to interact with the Business Logic Layer.
- **Database Operations**: Actions such as reading and writing data, typically handled by the Persistence Layer to manage information in a database.
- **ChangeEntities()**: A method in the Business Logic Layer used to update or modify the application's main entities.
- **ReadData() / WriteData()**: Methods in the Persistence Layer that handle reading from and writing to the database.
- **BaseModel**: A base class that provides common attributes and methods like unique identifier (Id), creation and update timestamps (CreatedAt, UpdatedAt), and standard operations (create(), read(), update(), save(), delete()) shared by all entities.
- **User**: Represents a person using the application, with personal details such as first and last name, email, password, admin status, and payment method. Users can create places.
- **Place**: Represents a property listed by a user. It includes information like title, description, price, geographic location (latitude and longitude), number of rooms, capacity, and surface area. Each place is owned by a user.
- **Amenity**: Features or facilities associated with a place, such as Wi-Fi or parking. Amenities are linked to places.
- **Review**: Feedback provided by users on places they have visited. Reviews include a title, comment text, and a rating score, and are associated with both the user and the place.
- **UUID**: A universally unique identifier used to uniquely identify entities such as users, places, reviews, and amenities.
- **Inheritance**: A programming concept where classes like User, Place, Review, and Amenity inherit common attributes and methods from the BaseModel class, promoting code reuse and consistency.

CONCLUSION

In this first phase of the HBnB Evolution project, we have established a clear and detailed technical foundation by documenting the system's architecture, core business entities, and key interactions using UML diagrams.

This comprehensive documentation will guide the development process, ensuring consistency and clarity across all layers of the application.

By clearly defining user, place, review, and amenity management, the project is well-prepared to move into the implementation stage.