HBNB EVOLUTION

TECHNICAL DOCUMENTATION

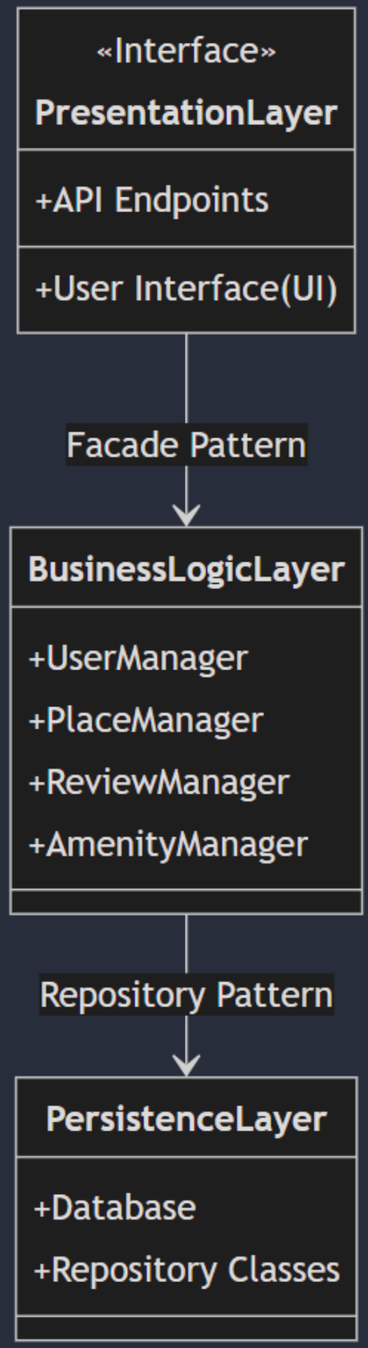
Authors:

Quentin Lepoutre / Jakob Stein / Lucas Niel / Raphaël Melnique

TABLE OF CONTENTS:

1. INTRODUCTION

The HBnB Evolution project aims to build a simplified version of an AirBnB-like application that allows users to register, list places, submit reviews, and manage amenities. The purpose of this document is to provide a clear and detailed technical design for the application, focusing on its architecture, business logic, and interactions through various API calls.
This document contains diagrams that illustrate the architecture, core entities, and the flow of data between different parts of the system. It is intended to guide developers through the implementation phase and ensure that the business requirements are accurately represented in the design.



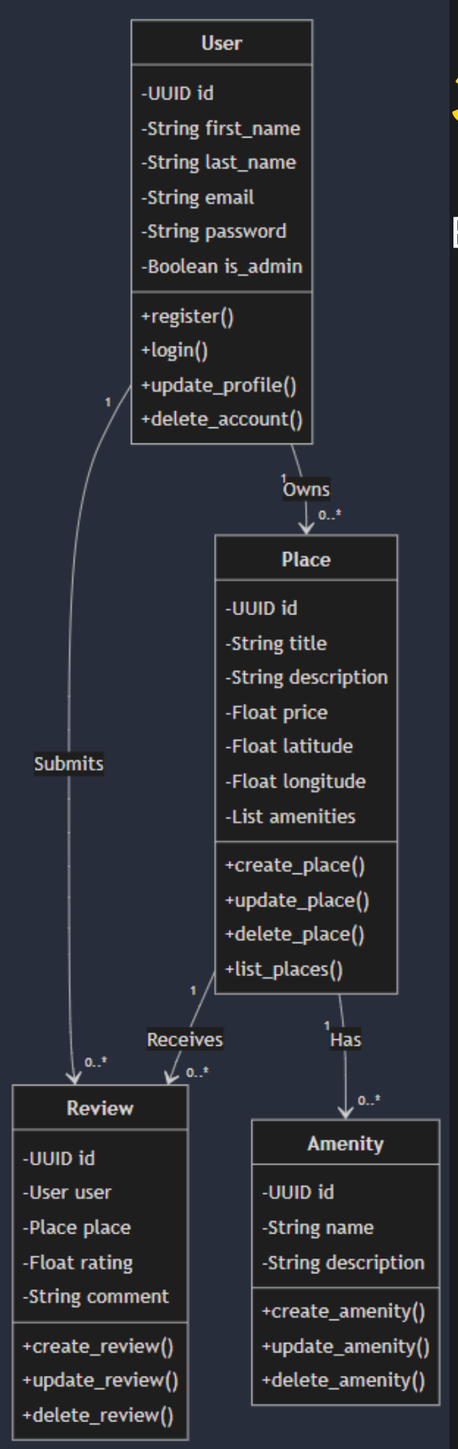
2. HIGH-LEVEL ARCHITECTURE

Explanation of Layers

- Presentation Layer:
  - This layer exposes the system's functionalities to the users through API endpoints. The user interface (UI) interacts with the API to perform actions like registering a user, creating places, submitting reviews, and fetching places. The API endpoints are the entry point to the system.

- Business Logic Layer:
  - This layer handles the core logic of the application, including managing users, places, reviews, and amenities. It uses managers like UserManager, PlaceManager, ReviewManager, and AmenityManager to encapsulate business rules and processes. The Facade Pattern is used to simplify the interface between the Presentation Layer and Business Logic Layer.

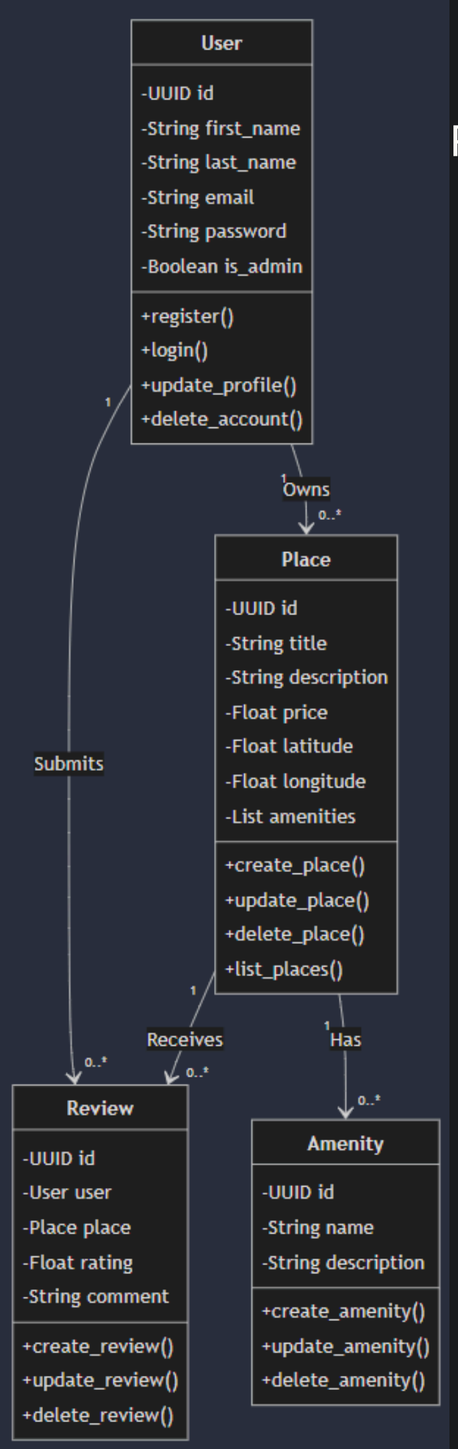
- Persistence Layer:
  - This layer is responsible for storing and retrieving data from the database. Each entity (e.g., User, Place, Review, Amenity) has a corresponding repository (e.g., UserRepository, PlaceRepository) that handles database operations. The Business Logic Layer communicates with the Persistence Layer via these repositories using the Repository Pattern.



3. BUSINESS LOGIC LAYER

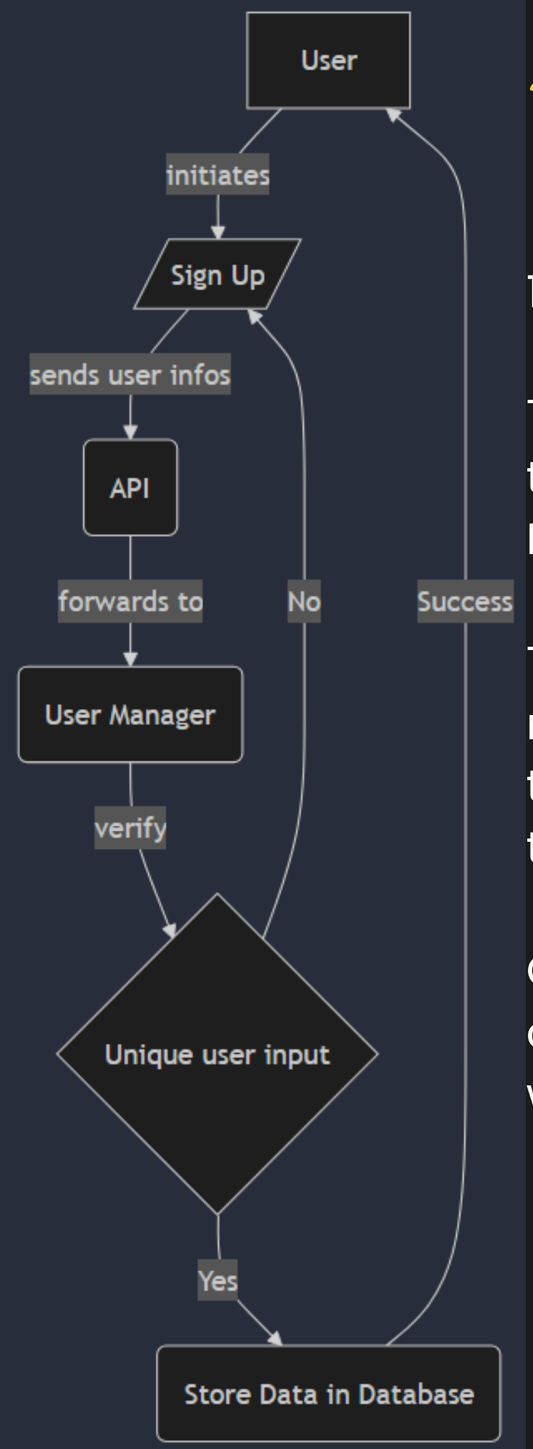
Explanation of Entities

- User:
  - Represents a user in the system. Attributes include first_name, last_name, email, and password, while methods include register(), login(), update_profile(), and delete_account(). The boolean attribute is_admin distinguishes between regular users and administrators.
- Place:
  - Represents a property listed by a user. Attributes include title, description, price, latitude, and longitude. Places can have a list of amenities and be managed by methods such as create_place(), update_place(), and delete_place().
- Review:
  - Represents a review that users submit for places. Each review is associated with a user and a place, and includes attributes like rating and comment. Methods for managing reviews include create_review(), update_review(), and delete_review().
- Amenity:
  - Represents additional features associated with places (e.g., Wi-Fi, Pool). Each amenity has attributes like name and description, and methods for managing amenities include create_amenity(), update_amenity(), and delete_amenity().



Relationships Between Entities

- User and Place:
  - A User can own multiple Places, represented by a one-to-many relationship (User --> Place).
- User and Review:
  - A User can submit multiple Reviews for different places (User --> Review).
- Place and Review:
  - A Place can have multiple reviews associated with it, represented by a one-to-many relationship (Place --> Review).
- Place and Amenity:
  - A Place can have multiple amenities, represented by a one-to-many relationship (Place --> Amenity).



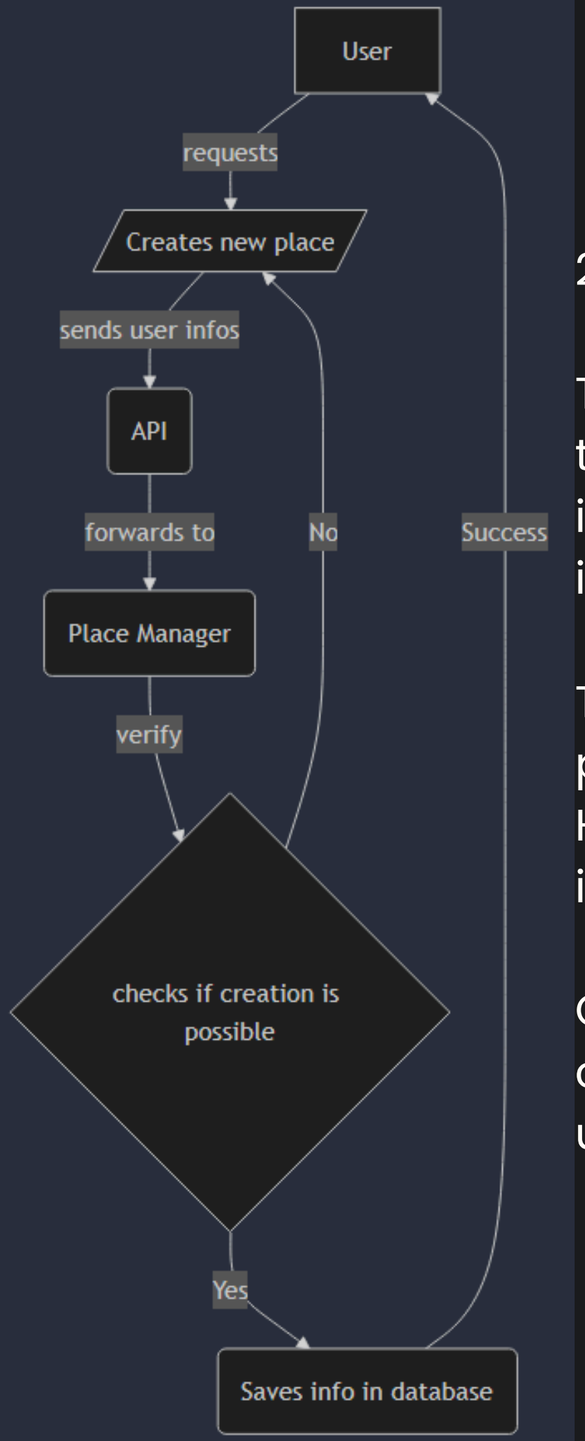
4. API INTERACTION FLOW

1. User Registration:

The process begins when a user initiates the sign-up process. The system then collects the user's information and sends it to an API. The API forwards the user data to the User Manager, which is responsible for verifying the user information.

The User Manager checks if the user input (e.g., username, email) is unique. If the input is not unique (for instance, if the username or email already exists), the process ends, and the sign-up is unsuccessful. However, if the input is unique, the system proceeds to store the user's data in the database.

Once the data is stored successfully, the user is informed of the successful sign-up, completing the process. If the data is not stored due to duplicate input, the process stops without success.

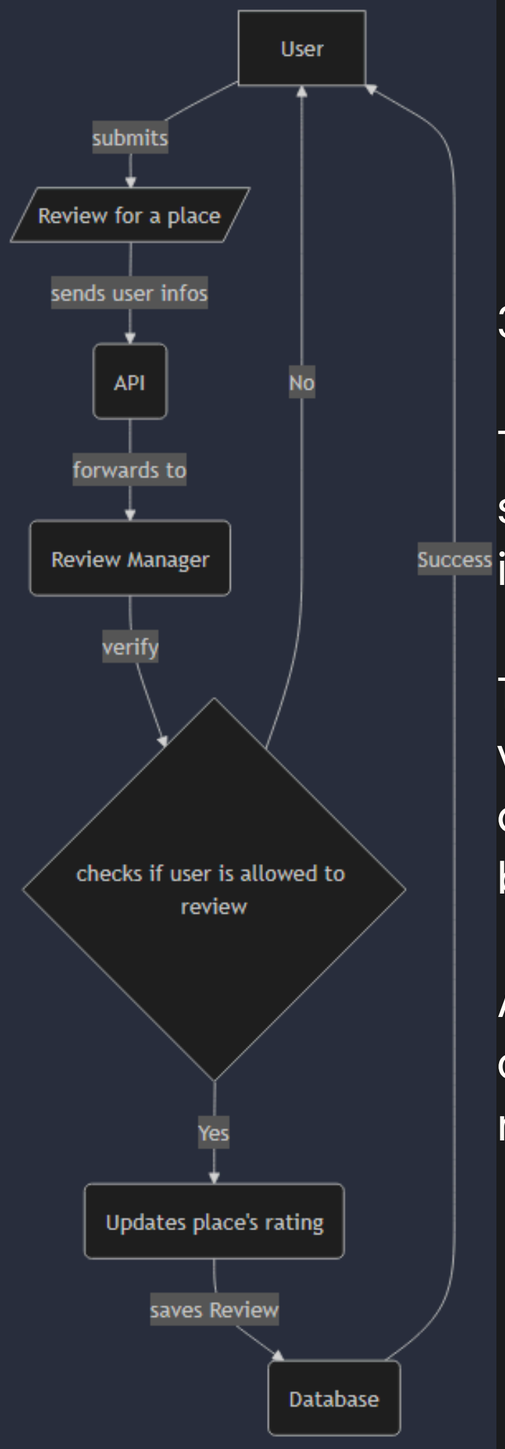


2. Place Creation:

The process starts when a user makes a request to create a new place. Upon receiving this request, the system initiates the creation of the place by sending the user's information to an API. The API then forwards the information to a Place Manager, which is responsible for handling the next steps.

The Place Manager verifies the details and checks whether creating the place is possible. If the creation is not possible, the process stops, and no action is taken. However, if the creation is deemed possible, the system proceeds to save the relevant information in the database.

Once the data is successfully stored, the system confirms that the new place has been created and informs the user that the operation was successful. If the creation fails, the user does not receive a success message.

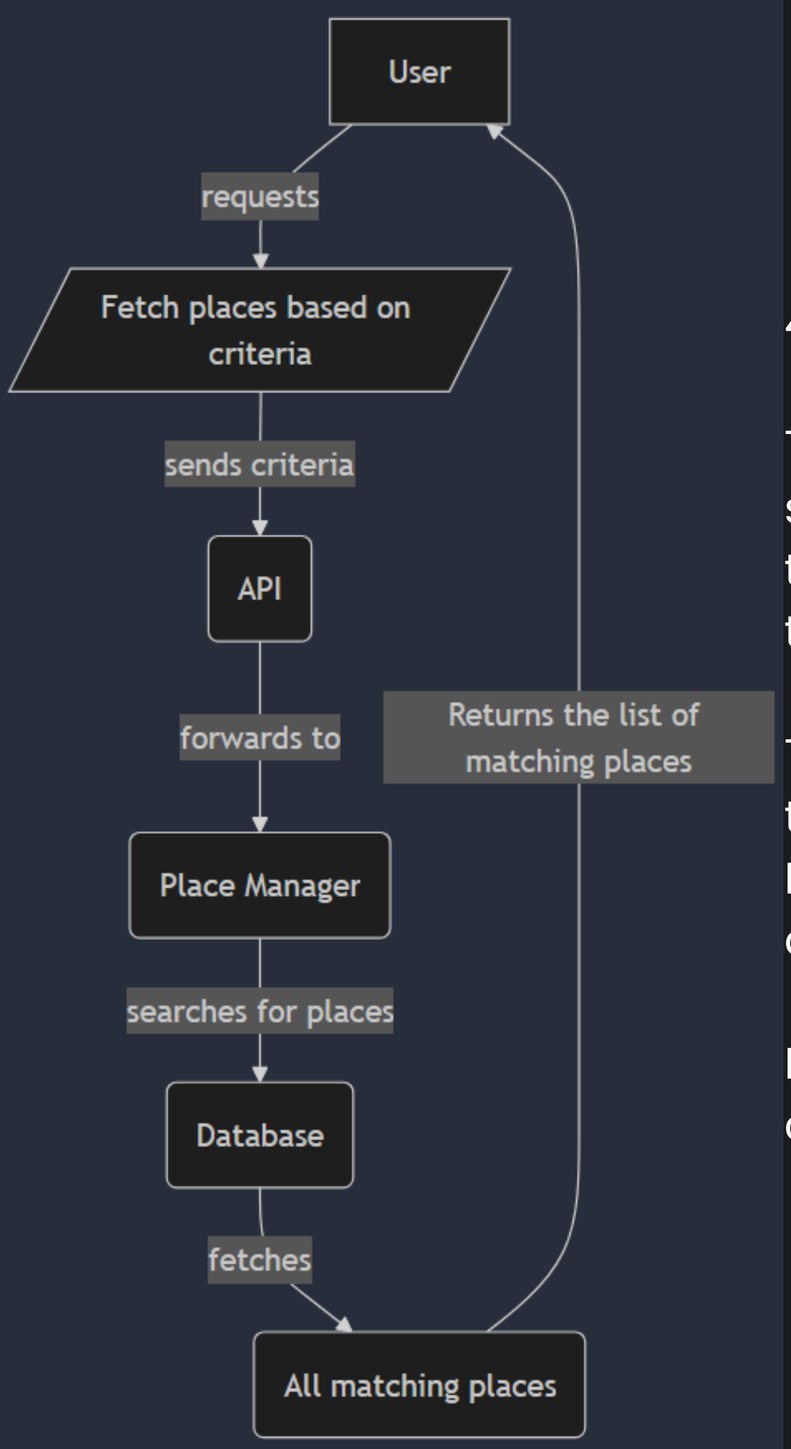


3. Review Submission:

The process begins when a user submits a review for a place. Once the review is submitted, the system sends the user's information to an API. The API forwards this information to the Review Manager, which is responsible for handling the review process.

The Review Manager then verifies whether the user is allowed to submit a review. If the verification fails (i.e., the user is not allowed to review), the process ends without further action. However, if the user is allowed to review, the system updates the place's rating based on the new review.

After the rating is updated, the system saves the review in the database. Once this is completed, the process ends successfully, and the user is informed of the success. If the review submission fails, the user does not receive a success message.



4. Fetching List of Places :

The process begins when a user makes a request to fetch places that match specific criteria. The system then gathers the user's search criteria and sends this information to an API. The API forwards the request, along with the criteria, to the Place Manager.

The Place Manager is responsible for handling the search for places. It queries the database to find all the places that match the specified criteria. Once the Place Manager retrieves the matching places from the database, the system compiles the list and sends it back to the user.

Finally, the user receives a list of all the places that meet the search criteria, completing the process.

CONCLUSION

This technical document serves as a detailed guide for the design and architecture of the HBnB Evolution project. It includes high-level and detailed class diagrams, sequence diagrams for key API calls, and thorough explanations of each component's role in the system. This document will act as a blueprint during the implementation phase to ensure that the system is developed in alignment with the specified business rules and design principles.

NEXT STEPS

Developers should now proceed with the implementation phase, using this document as a reference for coding the core entities and their interactions, as well as ensuring the API calls follow the designed flow.