Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

HbNb - UML

Introduction:

The HbNb project is a simplified version of an AirBnB-like application, where users can register, list properties, manage amenities, and submit reviews. This documentation outlines the system's architecture and design, serving as a guide during the implementation process. It provides detailed breakdown of the structure, focusing on the high-level architecture, business logic, and key API interactions.

The documents is organized as follows:

- **High-Level Architecture:** Provides an overview of the application's layered architecture, demonstrating how the presentation, business logic, and persistence layers interact.*
- **Business Logic Layer:** Offers a detailed breakdown of the entities within the business logic layer; including their attributes, relationships and roles within the system.
- **API interaction flows:** Explains how the different layers communicate through the system via key API calls, using sequence diagrams to illustrate the flow of information.

Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

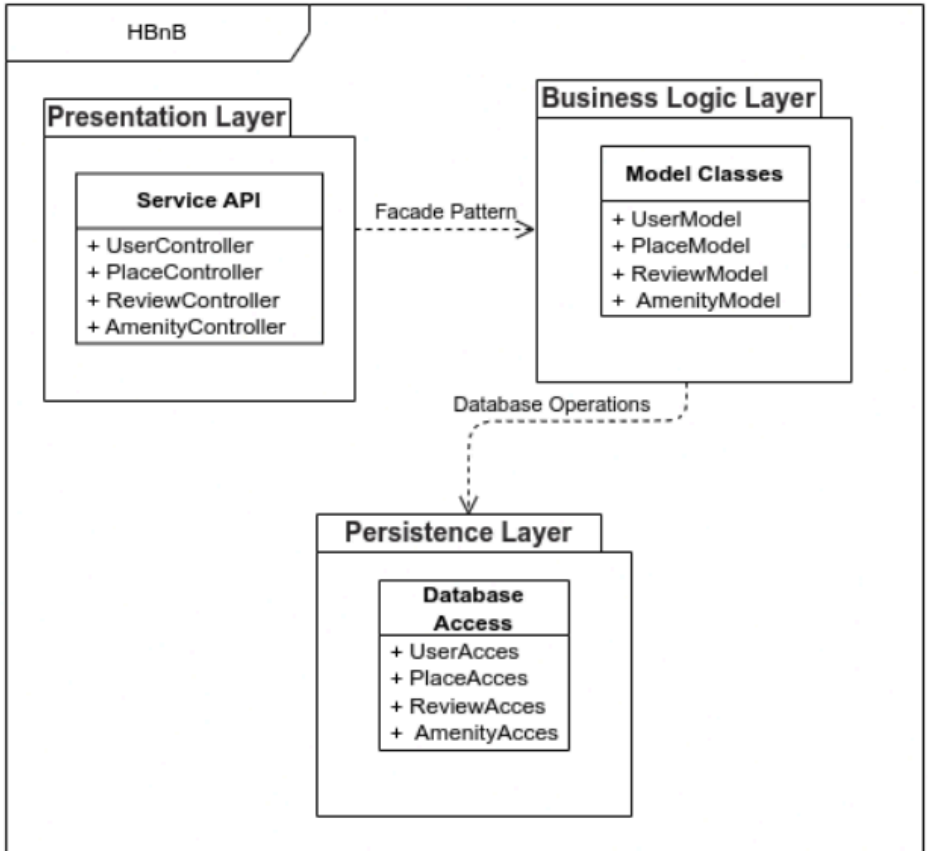
High-level Architecture:

For this part, we will be using a High-Level Package Diagram to demonstrate how the different layers are organized. The application is structured around a three-layer architecture:

- **Presentation Layer:** This layer handles user interactions through an API. it receives requests from users and forwards them to the business logic layer. The controllers act as intermediaries that validate inputs and manages responses

- **Business Logic Layer**: This is the core of the application, where all the business rules, validations and operations are handled. It includes services for managing users, places, reviews, and amenities. These services abstract the operations performed in the persistence layer and interact with the presentation layer through the Facade Pattern to simplify the communication.

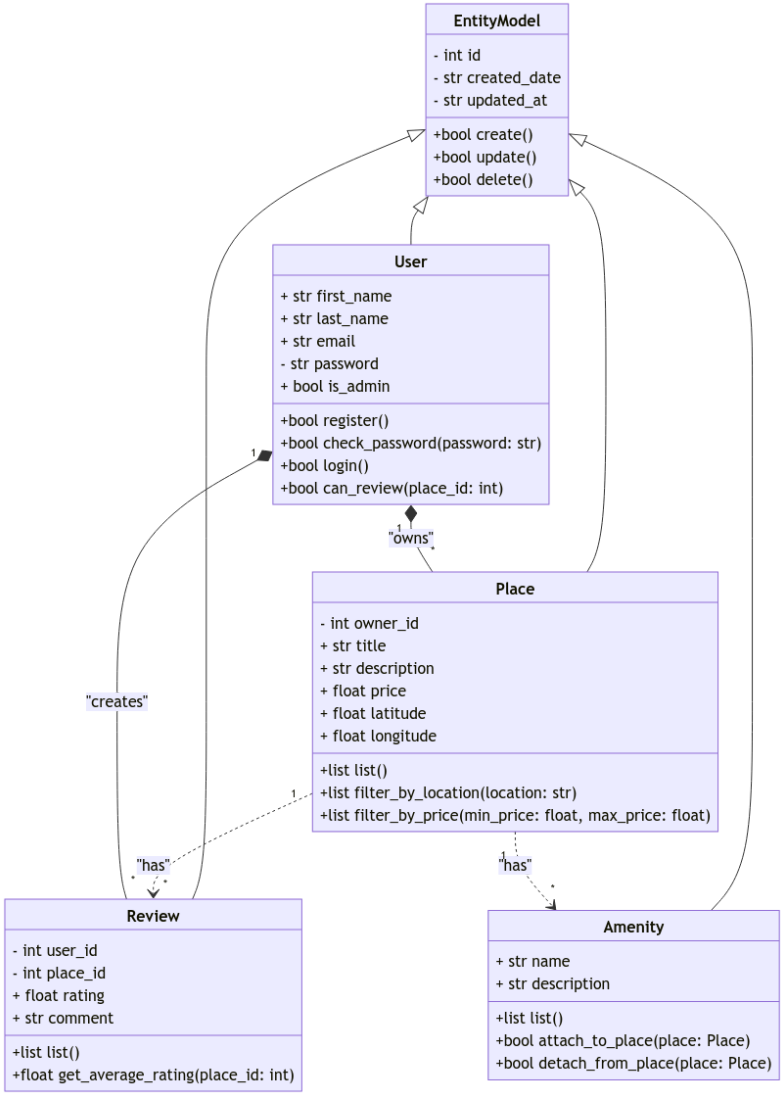
- **Persistence Layer:** The persistence layer deals with database operations. It is responsible for storing and retrieving data for users, places, reviews, and amenities. The Database Operations act as a bridge for the business logic into Database.



Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

Business Logic Layer:

The following diagram represents the Use, Place, Review, and Amenity entities. Each entity is a key part of the system, with attributes and methods that manage the business logic Operations.



- **Entity Model:** This base class includes common fields such as `id`, `created_date`, and `updated_at` to track the lifecycle of entities. It also includes generic methods such as `create()`, `update()`, and `delete()`.

- **User Entity:** Represents the users in the system, with attributes like `first_name`, `last_name`, `email`, and `password`. The `is_admin` boolean differentiates between regular users and administrators. Users can register, log in, and manage places and reviews. Key methods include:
  - `register()`: Registers a new user.
  - `check_password(password: str)`: Verifies the user's password.
  - `can_review(place_id: int)`: Determines if a user can leave a review for a specific place.

- **Place Entity**: Represents properties listed by users, with details like `title`, `description`, `price`, `latitude`, and `longitude`. Places are associated with users (owners) and can have multiple reviews and amenities. Key methods include:
  `list()`: Lists all places.
  `filter_by_location(location: str)`: Filters places by location.
  `filter_by_price(min_price: float, max_price: float)`: Filters places based on price range.

Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

- **Review Entity**: Represents feedback provided by users for places. Each review includes a `rating`, `comment`, and is linked to both a user and a place. Key methods include:
  `list()`: Lists all reviews.
  `get_average_rating(place_id: int)`: Gets the average rating for a place based on all user reviews.

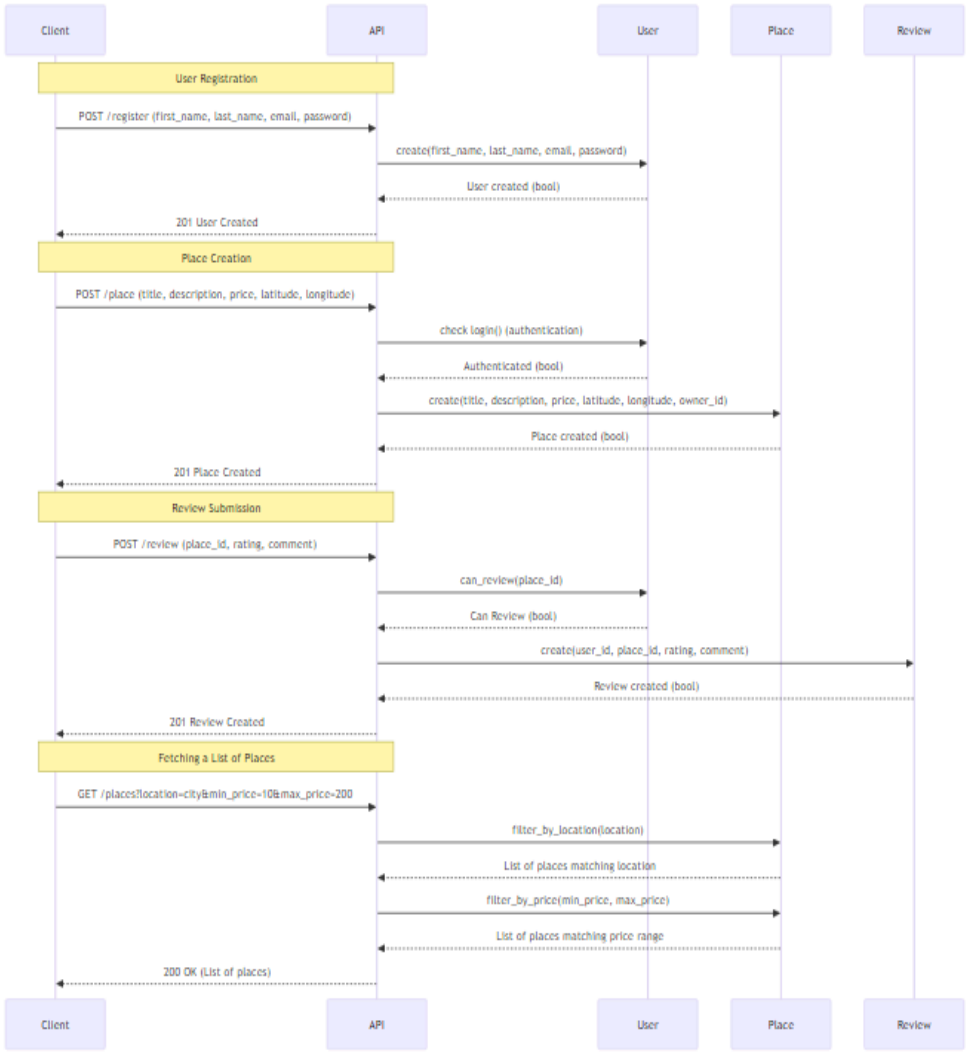
- **Amenity Entity**: Represents the amenities available at places. Each amenity has a `name` and `description`. Key methods include:
  `attach_to_place(place: Place)`: Attaches an amenity to a specific place.
  `detach_from_place(place: Place)`: Detaches an amenity from a place.

Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

API Interaction Flow:

The API interaction flow outlines the flow of information between layers for four key API interaction using an sequence diagram.it shows how the application processes requests, ensures business rules are applied, and interacts with the persistence layer to retrieve or store data.

1. **User Registration Sequence:**
   a. Client sends a `POST` request to the API at the `/register` endpoint with the user's first name, last name, email, and password.
   b. The API forwards this data to the User service, calling the `create()` method with the user details.
   c. The User service creates the new user in the persistence layer.
   d. Once the user is created, a `201 User Created` response is sent back to the client, confirming the registration process.
2. **Place Creation Sequence:**
   a. Client sends a `POST` request to the API at the `/place` endpoint with the place details such as title, description, price, latitude, and longitude.
   b. The API authenticates the request by calling `check_login()` on the user service to verify the user's credentials
   c. Upon successful authentication, the API passes the place details to the place service, calling `create()` to save the place.
   d. Once the place is successfully created, the API returns a `201 Place Create` response to the client.



Melvin Redondo–Tanis C24 Fréjus
Zakaria Aattache C24 Lille

3. **Review submission Sequence:**
    a. Client sends a `POST` request to the API at the `/review` endpoint with the place ID, rating and comment.
    b. The API checks if the user is eligible to review the place by calling `can_review()` on the user service.
    c. If the user is eligible, the API forwards the review details to the Review service to create the review for the specified place.
    d. Upon successful creation of the review, the API sends a `201 ReviewCreate` response to the client.
4. **Fetching a list of places sequence:**
    a. Client sends a `GET` request to the API at the `/places` endpoint with the filters such as location and price range
    b. The API forwards the request to the Place service which filters the list of places by calling `filter_by_location()` and `filter_by_price()`.
    c. The filtered list of places matching the specified criteria is retrieved from the persistence layer.
    d. Finally the API returns a `200 OK` response to the client, providing the filtered list of places.