HBnB Project

Introduction

HBnB (AirBnB Clone) is about creating, step by step, a complete web application inspired by Airbnb, with a well-structured and scalable architecture.
This document will serve as a detailed blueprint for the project, guiding the implementation phases and providing a clear reference for the system's architecture and design.

High-Level Architecture

Create 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 will provide a conceptual overview of how the different components of the application are organized and how they interact with each other.

```
classDiagram
class PresentationLayer {
    <>
    +ServiceAPI
}
class FacadePattern {
    +handleRequest
}
class BusinessLogicLayer {
    +ModelClasses
}
class PersistenceLayer {
    +DatabaseAccess
}
PresentationLayer --> FacadePattern : Calls API
FacadePattern --> BusinessLogicLayer : Calls Business Methods
BusinessLogicLayer --> PersistenceLayer : Database Operations

%% Custom CSS for Mermaid
style PresentationLayer fill:#808080,stroke:#00ffff,stroke-
width:2px,color:white;
style FacadePattern fill:#808080,stroke:#00ffff,stroke-
width:2px,color:white;
style BusinessLogicLayer fill:#808080,stroke:#00ffff,stroke-
width:2px,color:white;
style PersistenceLayer fill:#808080,stroke:#00ffff,stroke-
width:2px,color:white;
```

- PresentationLayer : This layer handles the interaction between the user and the application. It includes all the services and APIs that are exposed to the users.

- FacadePattern : This layer acts as an intermediary, grouping together calls to differents components of a subsystem and providing a simpler interface for the user.
- BusinessLogicLayer : This layer contains the core business logic and the models that represent the entities in the system (e.g., User, Place, Review, Amenity).
- PersistenceLayer : This layer is responsible for data storage and retrieval, interacting directly with the database.

Business Logic Layer

Design a detailed class diagram for the Business Logic layer of the HBnB application. This diagram will depict the entities within this layer, their attributes, methods, and the relationships between them. The primary goal is to provide a clear and detailed visual representation of the core business logic, focusing on the key entities: User, Place, Review, and Amenity.

```
classDiagram
direction RL
class BaseModel:::className {
    <>
    int id
    string datetime
    add()
    update()
    del()
}
class User {
    string Email
    string First Name
    string Last Name
    string Password
    boolean Administrator
    list places
    change_permission()
    list_places()
}
class Place {
    string Title
    string Description
    float Price
    float Latitude
    float Longitude
    list Amenities
    list_amenities()
    list_review()
}
class Review {
    float Rating
    string Comment
}
class Amenity {
    string Name
```

```
        string Description
    }
    User <|-- BaseModel
    Place <|-- BaseModel
    Review <|-- BaseModel
    Amenity <|-- BaseModel
    User "1.*" <|--|>"1" Place
    User "0.n" <|--|> "1" Review
    Place "0.n" <|--|> "1" Review
    Place "0.n" <|--|> "1" Amenity

    %% Custom CSS for Mermaid
    style User fill:#808080,stroke:#00ffff,stroke-width:2px,color:white;
    style Place fill:#808080,stroke:#00ffff,stroke-width:2px,color:white;
    style Review fill:#808080,stroke:#00ffff,stroke-width:2px,color:white;
    style Amenity fill:#808080,stroke:#00ffff,stroke-width:2px,color:white;
    style BaseModel fill:#3a8eba,stroke:#00ffff,stroke-width:2px,color:white;
```

Class Description

- User : class containing user information like names, email, password... It can list places created.
- Place : class containing places information like description, price, location... It can list associated amenities and reviews.
- Review : class containing reviews information like ratings and comments.
- Amenity : class containing amenities information like name and description.
- BaseModel : Abstract Base Class containing information common to all classes of the BusinessLogicLayer

Class Relationships

All classes inherits attributes and methods from ABC BaseModel

- Users : can creates multiple reviews and owns multiples places
- Places : owned by a User, can have list of amenities and can be reviewed multiple times.
- Amenities : associated with a Place.
- Reviews : can be created by User about Place.

API Interactions

Develop sequence diagrams for at least four different API calls to illustrate the interaction between the layers (Presentation, Business Logic, Persistence) and the flow of information within the HBnB application. The sequence diagrams will help visualize how different components of the system interact to fulfill specific use cases, showing the step-by-step process of handling API requests.

User Registration

A user signs up for a new account.

```
sequenceDiagram
participant User
participant Presentation Layer(API)
participant Business Logic Layer(Models)
participant Persistence Layer(Database)

User->>Presentation Layer(API): Register User(email, password...)
Presentation Layer(API)->>Business Logic Layer(Models): Validate and
Process Request
Business Logic Layer(Models)->>Persistence Layer(Database): Save Data
Persistence Layer(Database)-->>Business Logic Layer(Models): Confirm Save
Business Logic Layer(Models)-->>Presentation Layer(API): Return Response
Presentation Layer(API)-->>User: Return Success/Failure
```

This API call uses input data to create a new User.

1. User gives datas to the Presentation Layer.
2. The Presentation Layer sends the datas to the Business Logic Layer to create the new User.
3. Then the Business Logic Layer sends the new User to Persistence Layer to save it in the database.
4. Finally, each layer send success or failure response to previous layer up to the user.

Place Creation

A user creates a new place listing.

```
sequenceDiagram
participant User
participant Presentation Layer(API)
participant Business Logic Layer(Models)
participant Persistence Layer(Database)

User->>Presentation Layer(API): Create New Place(title, description...)
Presentation Layer(API)->>Business Logic Layer(Models): Validate and
Process Request
Business Logic Layer(Models)->>Persistence Layer(Database): Save Data
Persistence Layer(Database)-->>Business Logic Layer(Models): Confirm Save
Business Logic Layer(Models)-->>Presentation Layer(API): Return Response
Presentation Layer(API)-->>User: Return Success/Failure
```

This API call uses input data to create a new Place.

1. User gives datas to the Presentation Layer.
2. The Presentation Layer sends the datas to the Business Logic Layer to create the new Place.
3. Then the Business Logic Layer sends the new Place to Persistence Layer to save it in the database.
4. Finally, each layer send success or failure response to previous layer up to the user.

Review Submission

A user submits a review for a place.

```
sequenceDiagram
participant User
participant Presentation Layer(API)
participant Business Logic Layer(Models)
participant Persistence Layer(Database)

User->>Presentation Layer(API): Create New Review(rating, comment...)
Presentation Layer(API)->>Business Logic Layer(Models): Validate and
Process Request
Business Logic Layer(Models)->>Persistence Layer(Database): Save Data
Persistence Layer(Database)-->>Business Logic Layer(Models): Confirm Save
Business Logic Layer(Models)-->>Presentation Layer(API): Return Response
Presentation Layer(API)-->>User: Return Success/Failure
```

This API call uses input data to create a new Review.

1. User gives datas to the Presentation Layer.
2. The Presentation Layer sends the datas to the Business Logic Layer to create the new Review.
3. Then the Business Logic Layer sends the new Review to Persistence Layer to save it in the database.
4. Finally, each layer send success or failure response to previous layer up to the user.

Fetching a List of Places

A user requests a list of places based on certain criteria.

```
sequenceDiagram
participant User
participant Presentation Layer(API)
participant Business Logic Layer(Models)
participant Persistence Layer(Database)

User->>Presentation Layer(API): Request list of places(price, loction...)
Presentation Layer(API)->>Business Logic Layer(Models): Process Request
with filters
Business Logic Layer(Models)->>Persistence Layer(Database): Fetch
corresponding places
Persistence Layer(Database)-->>Business Logic Layer(Models): Send places
Business Logic Layer(Models)-->>Presentation Layer(API): Return Places Data
Presentation Layer(API)-->>User: Display places
```

This API call uses input data to retrieve Places.

1. User gives datas to the Presentation Layer.
2. The Presentation Layer sends the datas to the Business Logic Layer to apply filters.
3. Then the Business Logic Layer fetch the corresponding places from Persistence Layer.
4. Persistence Layer send Places to Business Logic Layer.
5. Business Logic Layer return these Places datas to Presentation Layer.
6. Presentation Layer display the list of Places.