PROJET HBNB - DIAGRAMME : DOCUMENTATION

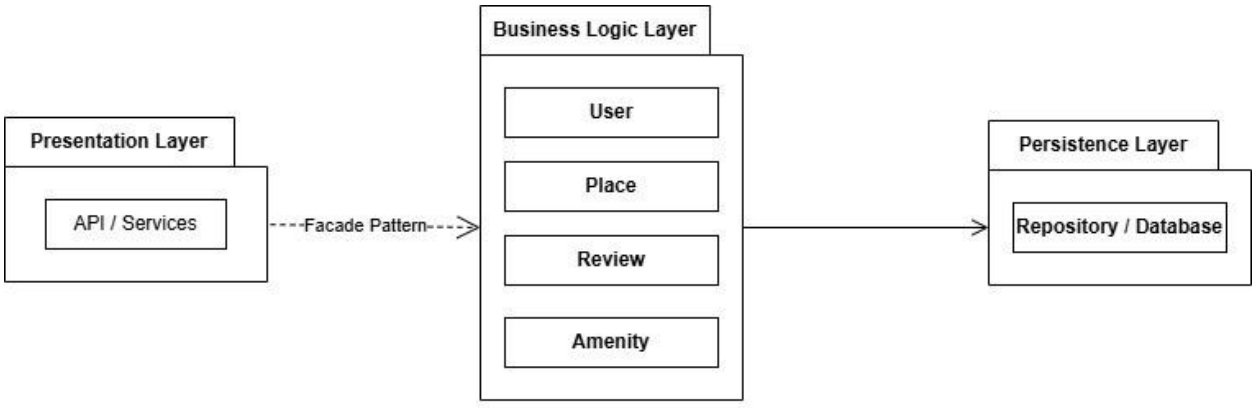
Introduction:

The HBNB project consists of designing an application that allows users to create and view housing listings, as well as add reviews and associated amenities. The goal is to develop a structured and organised system that facilitates the management of users, locations, reviews, and amenities.

To achieve this, the application is based on a layered architecture consisting of the presentation layer, the business logic layer and the persistence layer. This organisation allows for the separation of responsibilities, facilitates communication between components using the Facade pattern, and ensures reliable data storage in the database.

This project also uses UML diagrams to model the structure and interactions of the system, ensuring a clear, consistent and scalable design.

1 Package diagram



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

Business Logic Layer:
This layer contains the core business logic and models that represent the entities in the system:

1. User:
   This model allows the User to create an administrator account for User who proposes a place or create an User account for customers.
2. Place:
   This model allows the owner to create a place with a list of amenities.
3. Review:
   This model allows the user to create a review of the place.
4. Amenity:

By Pawnee & Mila

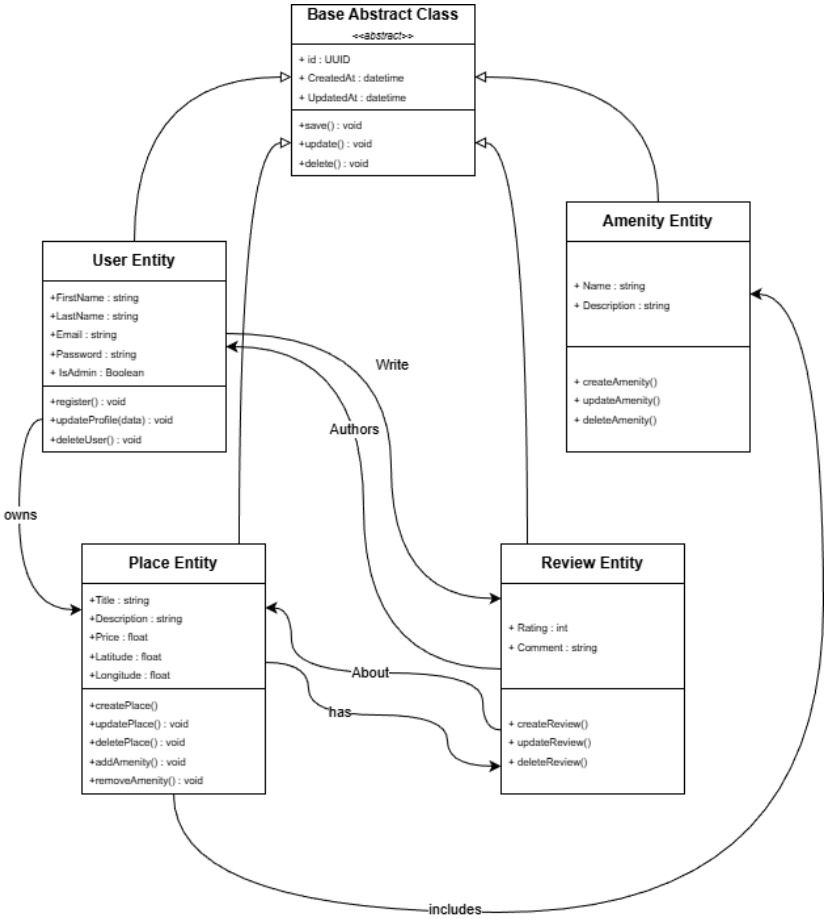
PROJET HBNB - DIAGRAMME : DOCUMENTATION

This model allows the administrator to create amenities for the place and the user can choose the amenities of the place.

Persistence Layer:
This layer is responsible for data storage and retrieval, interacting directly with the database.

Explanation:
Presentation Layer communicates thanks to facade Pattern facilitates communication between Presentation Layer and Business Logic Layer for directly interacting with multiple classes.
Presentation Layer communicates only with the Facade (Business Logic Layer).
Then the Business Logic Layer communicates with the Persistence Layer for the store database.

2 Class Diagram for Business Logic Layer



Base Abstract Class
Rôle : classe abstraite de base pour toutes les autres entités.

User Entity
Rôle : représente un utilisateur du système.

Place Entity
Rôle : représente un lieu, peut avoir des commodités et des reviews.

Amenity Entity
Rôle : représente une commodité que peut inclure un lieu.

Review Entity

Rôle : représente un avis sur un lieu.

PROJET HBNB - DIAGRAMME : DOCUMENTATION

1. Base Abstract Class

- **Attributes:** id (UUID), createdAt (datetime), updatedAt (datetime)
- **Methods:** save(), update(), delete()
- **Role:** Abstract base class for all other entities.

2. User Entity

- **Attributes:** firstName, lastName, email, password, isAdmin
- **Methods:** register(), updateProfile(data), deleteUser()
- **Role:** Represents a system user.

3. Place Entity

- **Attributes:** title, description, price, latitude, longitude
- **Methods:** createPlace(), updatePlace(), deletePlace(), addAmenity(), removeAmenity()
- **Role:** Represents a location; can have amenities and reviews.

4. Amenity Entity

- **Attributes:** name, description
- **Methods:** createAmenity(), updateAmenity(), deleteAmenity()
- **Role:** Represents an amenity that can be associated with a place.

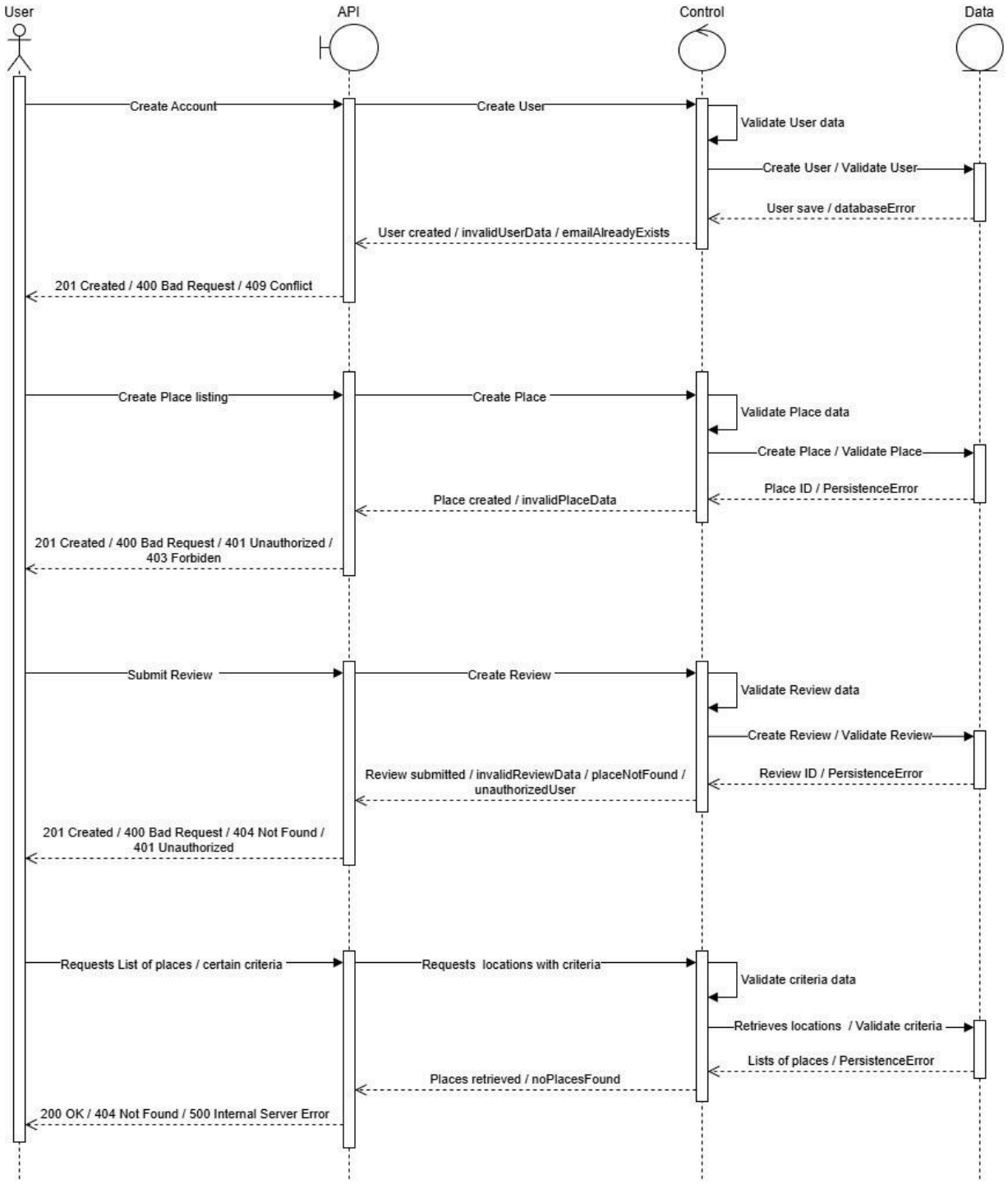
5. Review Entity

- **Attributes:** rating, comment
- **Methods:** createReview(), updateReview(), deleteReview()
- **Role:** Represents a review for a place.

By Pawnee & Mila

PROJET HBNB - DIAGRAMME : DOCUMENTATION

3 Sequence Diagrams for API Calls



By Pawnee & Mila

PROJET HBNB - DIAGRAMME : DOCUMENTATION

Create Account:

When the User wants to create an account, the User sends a request to the API.
The API forwards this message to the Control layer to handle the business logic.
The Control validates the user data (password, email, e.g.).
Then, check whether an account with the same email already exists in the database.
If the account already exists, the Control sends an error message back to the API , if not the Control sends a message to the Data layer to save the new user.
The Data layer tries to persist the account and return a success response or a failure message.
The Control returns the message of success or failure to the API and the API sends the message corresponding to the user.

Create Place listing:

When the User asks to create a new place listing, the request is sent to the API then forwards to the Control layer.
The Control validates the place data and then checks whether the listing already exists.
If the place listing already exists, an error response is sent back, if not already exists and the data is valid, the Control sends a message to the Database to store the new place listing.
The Database saves the listing in the database and returns either a success or a failure.
The Control forwards the result to the API, and the API sends the message corresponding to the user.

Submit Review:

When the user submits a review, he sends a request to the API and forwarded to the Control layer.
The Control validates the review data and verifies whether the referenced place exists and if the review doesn't already exist.
If validation fails or the place doesn't exist, the Control returns an error to the API, otherwise everything is valid, the Control sends a request to the Data layer to save the review.
The Data layer stores the review and returns a success or a failure.

By Pawnee & Mila

PROJET HBNB - DIAGRAMME : DOCUMENTATION

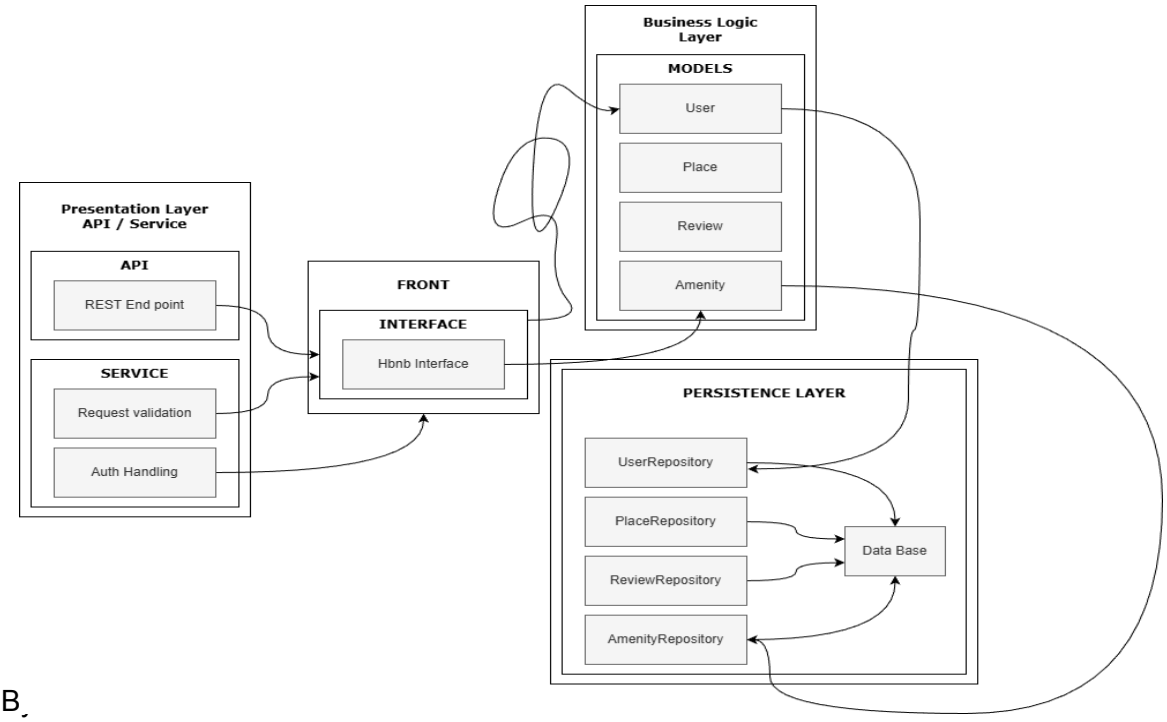
The Control sends the result to the API, and the API responds to the user.

Requests List of places / certain criteria:
When a user requests a list of places based on specific criteria (e.g., location, category, rating), the request is sent to the API and forwarded to the Control layer.
The Control validates the search criteria (e.g., correct format, allowed values).
Once validated, it sends a request to the Data layer to retrieve the matching places.
The Data layer queries the database and returns a list of matching places, or an error message if the retrieval fails.
The Control processes the result and sends it back to the API, which then returns the list or error message to the user.

Message success or failure:

201 Created : Used when a resource has been successfully created.
200 OK : Used when a read request works.
400 Bad Request : The client sent invalid data.
401 Unauthorized : The client is not authenticated.
409 Conflict : There is a conflict with the current state of the server.
404 Not Found : The requested resource does not exist.
403 Forbidden : The client is authenticated, but does not have permission to perform the action.
500 Internal Server Error : A server-side issue.

4 Conclusion Diagram



B