HBnB – UML Project: Technical Document

**Introduction**

This document serves as the comprehensive technical blueprint for the HBnB project. It compiles the High-Level Package Diagram, Detailed Class Diagram for the Business Logic Layer, and sequence diagrams for API calls into a cohesive document. The document is designed to provide a clear reference for the system’s architecture, design decisions, and component interactions, serving as the primary guide for the implementation phases.

**Scope of the Document**

This document includes:

1. A high-level package diagram illustrating the three-layer architecture of the application.
2. A detailed class diagram for the Business Logic Layer, showcasing its entities, relationships, and methods.
3. Sequence diagrams for critical API calls, visualizing the flow of information across layers.
4. Explanatory notes accompanying each diagram to clarify design decisions and interactions.

**1. High-Level Architecture**

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

The HBnB application utilizes a three-tier architectural pattern, promoting separation of concerns and maintainability:

**Presentation Layer (API):** This layer exposes the application's functionality to external clients (web, mobile, etc.). It handles incoming API requests, orchestrates processing via the Business Logic Layer, and returns responses. The Facade pattern simplifies interaction with the underlying layers.

**Business Logic Layer (Models):** This layer contains the core business logic and domain models (User, Place, Review, Amenity). It encapsulates business rules, data validation, and interactions with the Persistence Layer.

**Persistence Layer (Database):** This layer is responsible for data storage and retrieval. It handles all interactions with the database, implementing CRUD operations and ensuring data integrity.

**2. Business Logic Layer**

**Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.**

This section details the structure and relationships of the core entities within the Business Logic Layer. Each entity is represented as a class with defined attributes and methods.

* User: Represents a user of the HBnB platform.

**Attributes:**

* + - +UID: (string) Unique identifier for the user.
    - +FirstName: (string) User's first name.
    - +LastName: (string) User's last name.
    - +Email: (string) User's email address (unique).
    - +Password: (string) User's password (hashed).
    - +isAdmin: (boolean) Flag indicating administrator privileges.
    - +datetime creation: (int) Timestamp of user creation.
    - +datetimeupdate: (int) Timestamp of last user update.

**Methods:**

* + - +register (): Registers a new user.
    - +login (): Authenticates and logs in an existing user.
    - +logout (): Logs out the current user.
    - +updateProfile (Email(string), Password(string)): Updates user profile information.
    - +delete\_profile (): Deletes the user's profile.
    - +updatedatetime(int): Updates the datetimeupdate timestamp.
    - +createReview(place(string), user(string), rating(int), comment(string): Creates a new review for a place.
    - +createPlace(title(string), description(string), price(int), latitude(float), longitude(float)): Creates a new place listing.

**Place:** Represents a listing for a place available for rent.

* + - Attributes:
    - +UID: (string) Unique identifier for the place.
    - +title: (string) Title of the place.
    - +description: (string) Description of the place.
    - +price: (int) Price per night.
    - +latitude: (float) Latitude coordinate.
    - +longitude: (float) Longitude coordinate.
    - +datetimecreation: (int) Timestamp of place creation.
    - +datetimeupdate: (int) Timestamp of last place update.

**Methods:**

* + - +listReview (): Returns a list of reviews for the place.
    - +createAmenity(name(string), description(string)): Creates a new amenity for the place.
    - +updatedatetime(int): Updates the datetimeupdate timestamp.

**Review:** Represents a review submitted by a user for a place.

* + - Attributes:
    - +UID: (string) Unique identifier for the review.
    - +user: (string) User ID who created the review.
    - +rating: (int) Rating given by the user.
    - +Place: (string) Place ID the review is for.
    - +comment: (string) User's review text.
    - +datetime creation: (int) Timestamp of review creation.
    - +datetimeupdate: (int) Timestamp of last review update.
    - Methods:
    - +editComment (Comment(string)): Edits the review comment.
    - +delete(): Deletes the review.
    - +updatedatetime(int): Updates the datetimeupdate timestamp.

**Amenity:** Represents an amenity offered at a place.

* + - Attributes:
    - +UID: (string) Unique identifier for the amenity.
    - +name: (string) Name of the amenity.
    - +description: (string) Description of the amenity.
    - +datetime creation: (int) Timestamp of amenity creation.
    - +datetimeupdate: (int) Timestamp of last amenity update.
    - Methods:
    - +addToPlace (Place place): Adds amenity to a place.
    - +removeFromPlace (Place place): Removes the amenity from a place.
    - +updatedatetime(int): Updates the datetimeupdate timestamp.

**Relationships:**

* + - A User owns 0..\* Place(s).
    - A User writes 0..\* Review(s).
    - A Place has 1..\* Review(s).
    - A Place contains 1..\* Amenity(s).

**3. API Interaction Flow:**

The following sections detail specific API call workflows using sequence diagram

Une image contenant texte, capture d’écran, diagramme

Le contenu généré par l’IA peut être incorrect.

**Objective:** Illustrates the process of registering a new user through the API.

* + - Steps:
    - User: Initiates an API call (POST, Register User) to the API endpoint.
    - API: Receives the registration data.
    - Models: Validates the data with the database.
    - Database:
    - If the user doesn't exist, save the user's data and confirm.
    - If the user already exists, returns a "User already exists" message.
    - API: Returns a 201 (Created) success response or a 200 (OK) with user exists to user.
    - User: Receives suceess or failed response from the API.

**3.2 Place Creation**:

Place Creation Sequence Diagram

Une image contenant texte, capture d’écran, diagramme

Le contenu généré par l’IA peut être incorrect.

**Objective:** Illustrates the process of creating a new place listing through the API.

* + - Steps:
    - User: Initiates an API call (PUT, Place Creation) to the API endpoint.
    - API: Receives user data and new place data.
    - Models: Checks the validity and authorization of the user and the place details.
    - Database:
    - If creation of place is possible saves the new place data and comfirms the Save
    - If not possible returns the reason of failure.
    - API: Returns a 200 (ok) success respons, or a 404/205 respons for Creation Impossible.
    - User: Receives suceess or failed response from the API.