

# Diagramme de classe

## BaseEntity

Classe abstraite contenant les attributs de base communs à toutes les entités du système.

### Attributs :

- id : identifiant unique au format UUID4.
- created\_at : date de création.
- updated\_at : date de dernière modification.

Toutes les entités héritent de cette classe pour garantir une structure uniforme et simplifier la sérialisation ou la traçabilité des objets.

## HbnBFacade

La classe HBNBFacade est une **façade métier** qui centralise les appels aux entités User, Place, Review et Amenity.

Elle est utilisée par la couche de présentation pour **simplifier l'accès** à la logique métier tout en masquant la complexité interne.

En regroupant les opérations clés dans une seule interface, elle améliore la **lisibilité**, la **modularité** et permet de **limiter les dépendances directes** à plusieurs classes métier.

### Méthodes exposées :

- register\_user() : crée un nouveau compte utilisateur
- create\_place() : ajoute une nouvelle annonce
- add\_review() : permet à un utilisateur de laisser un avis
- list\_amenities() : affiche les commodités disponibles
- get\_user\_profile() : récupère les informations d'un utilisateur

## User

Représente un utilisateur inscrit sur la plateforme HBNB.

Il peut créer des annonces, laisser des avis, et selon son statut (is\_admin), accéder à des fonctions avancées.

### Attributs :

- first\_name, last\_name, username, email, password : informations personnelles et d'identification.

- is\_admin : indique si l'utilisateur est un administrateur (booléen).

**Méthodes :**

- register() : enregistrement de l'utilisateur.
- update\_profile() / del\_profile() : gestion du compte.
- is\_admin() : vérification du statut d'administrateur.

**Place**

Représente un logement mis en ligne sur la plateforme.

Cette entité contient à la fois des informations descriptives, géographiques et économiques sur le bien proposé.

**Attributs :**

- country, city, address : localisation administrative du logement.
- housing\_type, room\_count : caractéristiques physiques du bien.
- description : présentation libre du logement.
- price : prix de location (corrigé en float pour plus de précision).
- latitude, longitude : coordonnées géographiques exactes du logement.

**Méthodes :**

- create\_place(), update\_place(), delete\_place() : gestion CRUD du logement.
- list\_places() : affichage ou filtrage des biens disponibles.

**Review**

Représente un avis laissé par un utilisateur sur un logement.

**Attributs :**

- place\_id : identifiant du logement concerné.
- user\_id : identifiant de l'auteur de l'avis.
- rating : note attribuée.
- commentary : commentaire textuel.

**Méthodes :**

- create\_review(), update\_review(), delete\_review() : gestion des avis.
- list\_reviews() : affichage des avis liés à un logement.

**Amenity**

Représente un équipement ou service proposé avec un logement.

**Attributs :**

- type : nom/type de l'équipement (ex. Wi-Fi, Piscine).
- description : détail de l'équipement.

**Méthodes :**

- create\_amenity(), update\_amenity(), delete\_amenity() : gestion des équipements.
- list\_amenity() : affichage des équipements disponibles.

**Relations entre les entités**

Les entités User, Place, Review et Amenity héritent toutes de la classe parente BaseEntity, qui définit une structure commune (identifiant unique, date de création, date de mise à jour).

Les relations entre ces entités reflètent les règles métier fondamentales du système HBNB.

**1. BaseEntity → Tous les modèles (héritage)**

**Relation :** Généralisation (<|--)

**Description :** Toutes les entités du système héritent de la classe abstraite BaseEntity, qui fournit les attributs communs id, created\_at et updated\_at. Cette relation structure l'ensemble du modèle métier en garantissant une base uniforme.

**2. User → Place**

**Relation :** Agrégation (◇)

**Multiplicité :** Un User peut posséder 1 ou plusieurs Place (1..\*), chaque Place est associé à un seul User (1).

**Justification :** Les utilisateurs créent les logements. Un logement ne peut exister sans auteur, mais sa suppression ne nécessite pas la suppression de son créateur.

**3. User → Review**

**Relation :** Agrégation (◇)

**Multiplicité :** Un User peut créer plusieurs Review (1..\*), chaque Review appartient à un seul User (1).

**Justification :** Un utilisateur peut laisser plusieurs avis. Si un utilisateur est supprimé, les avis peuvent être conservés (système de traçabilité, anonymisation).

**4. Place → Review**

**Relation :** Composition (●)

**Multiplicité** : Un Place peut contenir plusieurs Review (1..\*), chaque Review concerne un seul Place (1).

**Justification** : Les avis n'ont de sens que s'ils sont liés à un logement existant. Si un logement est supprimé, ses avis sont également supprimés.

#### 5. Place → Amenity

**Relation** : Agrégation (◇)

**Multiplicité** : Un Place peut avoir plusieurs Amenity (0..\*), chaque Amenity peut être partagé par plusieurs Place (\*).

**Justification** : Un logement peut être équipé de plusieurs commodités. Ces équipements sont réutilisables d'un logement à l'autre (ex. Wi-Fi, Jardin), sans être dépendants d'un logement spécifique.

#### 6. User → Amenity

**Relation** : Agrégation (◇)

**Multiplicité** : Un User peut créer plusieurs Amenity (0..\*), chaque Amenity a un seul créateur (1).

**Justification** : Certaines commodités peuvent être ajoutées manuellement par les utilisateurs (par exemple des équipements personnalisés). Leur durée de vie n'est pas liée à celle du créateur, car elles peuvent être réutilisées ailleurs.

#### 7. HBnBFacade → User, Place, Review, Amenity

**Relation** : Association simple (-->)

**Multiplicité** : HBnBFacade peut interagir avec un ou plusieurs objets des entités User, Place, Review, et Amenity (\*).

Chaque entité peut également être utilisée dans plusieurs opérations de la façade.

**Justification** : La classe HBnBFacade joue un rôle d'interface entre la couche de présentation et la logique métier.

Elle centralise les appels vers les entités principales pour simplifier l'usage du système côté contrôleurs.

Ce n'est pas une entité métier, mais une façade technique, ce qui justifie l'usage d'associations simples plutôt qu'une agrégation ou une composition.