Diagramme du package

Diagramme d'architecture en trois couches HBnB

Ce diagramme représente **l'organisation technique de notre application HBnB** en trois grandes couches, chacune ayant un rôle précis. C'est ce qu'on appelle une **architecture en couches** (*three-layer architecture*), très utilisée pour structurer les applications web modernes.

1. Présentation (PresentationLayer)

- **Rôle** :
  C'est la couche qui reçoit les demandes des utilisateurs et gère tout ce qui concerne **ces interactions avec l'utilisateur** (le site web ou l'application mobile). Elle contient les contrôleurs qui envoient ces demandes à la partie principale du site (LogicLayer).
- **Composants techniques** - Contrôleurs :
  - `UserController` : gère les actions liées aux utilisateurs (inscription, connexion…)
  - `PlaceController` : gère les logements (création, affichage, modification…)
  - `ReviewController` : gère les avis laissés par les utilisateurs
  - `AmenityController` : gère les équipements (WiFi, piscine, etc.)
- **Concrètement** :
  Quand un utilisateur clique sur un bouton ou soumet un formulaire, c'est un de ces contrôleurs qui reçoit la demande et la transmet à la suite du système.

2. Logique métier (BusinessLogicLayer)

- **Rôle** :
  C'est le **coeur** du site, elle contient les modèles qui représentent les objets du site, et le HBnBFacade qui fait le lien entre les contrôleurs et les données. C'est ici que les règles métier et les actions sont gérées.

- **Composants techniques** - Modèles :
    - `User`, `Place`, `Review`, `Amenity` : ce sont les **modèles** (représentation des objets principaux du site, avec leurs attributs et méthodes).
    - `HBnBFacade` : c'est une **façade** (pattern Facade) qui sert d'interface unique entre la présentation et la logique métier. Elle simplifie l'accès aux fonctionnalités : les contrôleurs appellent la façade, qui se charge de coordonner les opérations sur les modèles.
- **Concrètement** :
  Quand un contrôleur veut, par exemple, enregistrer un utilisateur, il appelle une méthode de la façade (`register_user()`), qui va ensuite vérifier les données, créer l'utilisateur, etc.

3. Persistence (PersistenceLayer)

- **Rôle** :
  C'est la couche qui gère les données, qui s'occupe de **stocker et retrouver les données** dans la base (CRUD : Create, Read, Update, Delete). Elle contient les Repositories qui vont chercher ou enregistrer les infos dans la base de données. Elle ne parle qu'au cœur, jamais directement à l'utilisateur.
- **Composants techniques** :
    - `UserRepository`, `PlaceRepository`, `ReviewRepository`, `AmenityRepository` : ce sont les **repositories**, des classes spécialisées pour accéder à la base de données pour chaque entité.
    - `persistence calls` : ce sont les appels concrets à la base (requêtes SQL ou ORM).
- **Concrètement** :
  Quand la logique métier a besoin d'enregistrer ou de lire un objet, elle passe par le repository correspondant, qui s'occupe de la communication avec la base de données.

Flux de l'information

- **Du haut vers le bas** :
    a. L'utilisateur interagit avec le site → un contrôleur reçoit la demande.
    b. Le contrôleur transmet à la façade (`via HBnBFacade`).
    c. La façade orchestre les appels aux modèles métier.
    d. Les modèles utilisent les repositories pour lire/écrire en base (`repository interaction`).

- **Du bas vers le haut** :
  Les données remontent depuis la base, sont traitées par la logique métier, puis renvoyées au contrôleur qui prépare la réponse à l'utilisateur.

Pourquoi cette structure ?

- **Lisibilité et organisation** : chaque couche a un rôle bien défini.
- **Sécurité et robustesse** : la logique métier est centralisée, la persistence est bien séparée.
- **Évolutivité** : on peut faire évoluer une couche sans tout casser (par exemple, changer de base de données sans toucher à la logique métier).
- **Réutilisation** : les repositories et les modèles peuvent être réutilisés dans différents cas d'usage.

Résumé :

- **PresentationLayer** : reçoit les demandes des utilisateurs via des contrôleurs.
- **BusinessLogicLayer** : traite la logique métier, centralisée par la façade (`HBnBFacade`), manipule les objets principaux du site.
- **PersistenceLayer** : gère l'accès aux données via des repositories spécialisés.

Cette organisation permet à toute l'équipe de comprendre qui fait quoi dans le code, de faciliter la maintenance, et de garantir la qualité du projet HBnB.