H’BNB

Documentation technique – Présentation de HBnB

H’BNB est une site web inspiré d’Airbnb, développée pour permettre aux utilisateurs de gérer la location de biens immobiliers en ligne.

Cette documentation technique décrit en détail l’architecture, la conception et les principaux éléments du système.

Objectif du document

Ce document a pour but d’offrir une compréhension approfondie du fonctionnement et de la structure de H’BNB.

Il s’adresse principalement :

- Aux développeurs impliqués dans l’implémentation du système
- Aux architectes souhaitant analyser les choix de conception
- Aux équipes de maintenance qui assureront les futures évolutions

Portée du projet

H’BNB repose sur quatre fonctionnalités essentielles :

- **Gestion des utilisateurs** : Inscription, mise à jour du profil et attribution des rôles (utilisateurs classiques et administrateurs)
- **Gestion des propriétés** : Création, modification et suppression des annonces de location
- **Système de notation** : Ajout et gestion des avis sur les logements
- **Gestion des équipements** : Administration des services et commodités associés aux propriétés

Architecture du système

Le système suit une architecture en trois niveaux :

- **Couche de présentation** : Interface utilisateur et services API
- **Couche métier** : Logique applicative et modèles de données
- **Couche de persistance** : Stockage et gestion des données

Organisation de la documentation

Cette documentation est structurée en plusieurs parties :

- **Diagramme de packages** : Vue globale de l’architecture
- **Diagramme de classes** : Détails sur la structure et la logique métier

- **Diagrammes de séquence** : Représentation des interactions pour les principales opérations

L'architecture mise en place garantit :

- Une répartition claire des responsabilités
- Une maintenance facilitée
- Une évolutivité optimale
- Une gestion des données fiable et robuste

Documentation du Schéma d'Architecture API

Qu'est-ce qu'une API ?

Une API (Application Programming Interface) est un ensemble de règles et de protocoles permettant à différentes applications de communiquer entre elles. Elle définit la manière dont les logiciels interagissent en facilitant l'échange de données et l'exécution de fonctionnalités spécifiques.

1. Présentation Générale

Ce schéma représente l'architecture d'une API REST qui permet aux utilisateurs de :

- Créer un compte
- Créer des lieux
- Ajouter des avis sur des lieux
- Rechercher des lieux avec des filtres

Le diagramme montre comment les requêtes du client sont traitées à travers différentes couches avant d'atteindre la base de données.

2. Explication des Différentes Couches

L'API est divisée en plusieurs couches :

- **Client** : L'utilisateur ou l'application qui envoie des requêtes à l'API.
- **API Layer** : Reçoit les requêtes, les valide et envoie les réponses.
- **Facade** : Gère la logique métier (vérification des permissions, transformation des données, etc.).
- **Models** : Contient les objets métiers (utilisateur, lieu, avis).
- **Database** : Stocke les données de manière permanente.

3. Description des Fonctionnalités

1. Création d'un utilisateur (POST /api/users)

- Le client envoie un email, un mot de passe et un nom.
- L'API crée un utilisateur et l'enregistre dans la base de données.
- Retourne les informations de l'utilisateur.

2. Création d'un lieu (POST /api/places)

- Vérification de l'authentification.
- Création et enregistrement du lieu en base de données.
- Retourne les détails du lieu créé.

3. Ajout d'un avis sur un lieu (POST /api/places/{id}/reviews)

- Vérification de l'authentification.
- Récupération des informations du lieu.
- Création et sauvegarde de l'avis en base de données.
- Retourne les détails de l'avis ajouté.

4. Recherche de lieux avec filtres (GET /api/places/filters)

- Le client envoie des critères de recherche.
- L'API récupère les lieux correspondants dans la base de données.
- Retourne la liste des lieux filtrés.

Documentation du Diagramme de Classes UML

1. Qu'est-ce qu'un diagramme de classes UML ?

Un **diagramme de classes UML** est une représentation graphique des classes, de leurs attributs, méthodes et des relations entre elles dans un système. Il est utilisé pour modéliser la structure d'un programme avant son développement.

Chaque **classe** définit un ensemble d'objets partageant les mêmes caractéristiques (attributs) et comportements (méthodes). Les **relations** entre les classes montrent comment elles interagissent entre elles.

2. Analyse du Schéma

Ce diagramme représente les entités principales et leurs relations dans un système de gestion de lieux et d'avis.

2.1. Les Classes et leurs Attributs

1. User (Utilisateur)

- **Attributs** : prénom, nom, email, mot de passe, rôle admin
- **Méthodes** : s'enregistrer, mettre à jour le profil, supprimer un compte

2. Place (Lieu)

- **Attributs** : titre, description, prix, latitude, longitude
- **Méthodes** : créer, mettre à jour, supprimer, lister

3. Review (Avis)

- **Attributs** : note, commentaire
- **Méthodes** : créer, mettre à jour, supprimer, lister les avis d'un lieu

4. Amenity (Commodité/Équipement)

- **Attributs** : nom, description
- **Méthodes** : créer, mettre à jour, supprimer, lister

3. Les Relations Entre les Classes

- **Un User possède (owns) un ou plusieurs Places.**
- **Un User écrit (writes) des Reviews sur des Places.**
- **Une Place reçoit (receives) des Reviews.**
- **Une Place a (has) des Amenities.**
-

Documentation de l'Architecture en Trois Couches (Three-Tier Architecture)

1. Qu'est-ce que l'Architecture en Trois Couches ?

L'**architecture en trois couches** (*Three-Tier Architecture*) est un modèle de conception logicielle qui divise une application en trois niveaux distincts. Cette séparation permet une meilleure organisation du code, une maintenance facilitée et une évolutivité optimale.

Chaque couche a un rôle bien défini et communique avec les autres pour assurer le bon fonctionnement du système.

2. Les Trois Couches de l'Architecture

1. Presentation Layer (Couche Présentation)

- Contient l'interface utilisateur (*User Interface*).
- C'est la partie visible de l'application avec laquelle l'utilisateur interagit.
- Elle envoie les actions de l'utilisateur à la couche métier.

2. Business Layer (Couche Métier)

- Gère la logique métier.
- Contient les entités principales : **User, Place, Review, Amenity**.
- C'est ici que les règles métiers sont appliquées (exemple : vérification des permissions, calculs, validation des données).

3. Persistence Layer (Couche de Persistance)

- Gère l'accès aux données et aux services externes.
- Contient **Data Access** (accès aux bases de données) et **Service Agents** (communication avec des API ou services externes).
- Stocke et récupère les informations demandées par la couche métier.

Cette structure en trois couches assure une séparation claire des responsabilités, facilitant ainsi le développement, la maintenance et l'évolution du système. 🚀