H'BNB

Technical Documentation – Overview of HBnB

H'BNB is a web application inspired by Airbnb, designed to allow users to manage property rentals online. This technical documentation details the system's architecture, design, and key components.

Objective of the Document

This document aims to provide an in-depth understanding of the functionality and structure of H'BNB.

It is primarily intended for:

- Developers involved in system implementation
- Architects analyzing design choices
- Maintenance teams ensuring future updates

Project Scope

H'BNB is built around four core functionalities:

- **User Management:** Registration, profile updates, and role assignments (regular users and administrators)
- **Property Management:** Creation, modification, and deletion of rental listings
- **Review System:** Adding and managing reviews for properties
- **Amenity Management:** Administration of services and amenities associated with properties

System Architecture

The system follows a three-tier architecture:

- **Presentation Layer:** User interface and API services
- **Business Layer:** Application logic and data models
- **Persistence Layer:** Data storage and management

Documentation Organization

This documentation is structured into several sections:

- **Package Diagram:** Overall view of the architecture
- **Class Diagram:** Details on structure and business logic
- **Sequence Diagrams:** Representation of interactions for key operations

The established architecture ensures:

- A clear division of responsibilities
- Simplified maintenance

- Optimal scalability
- Reliable and robust data management

API Architecture Documentation

What is an API?

An API (*Application Programming Interface*) is a set of rules and protocols that allows different applications to communicate with each other. It defines how software components interact by facilitating data exchange and the execution of specific functions.

1. General Overview

This diagram represents the architecture of a REST API that allows users to:

- Create an account
- Create places
- Add reviews to places
- Search for places with filters

The diagram illustrates how client requests are processed through different layers before reaching the database.

2. Explanation of Different Layers

The API is divided into several layers:

- **Client:** The user or application that sends requests to the API.
- **API Layer:** Receives requests, validates them, and sends responses.
- **Facade:** Manages business logic (permission verification, data transformation, etc.).
- **Models:** Contains business objects (user, place, review).
- **Database:** Permanently stores data.

3. Functionalities Description

1. User Creation (POST /api/users)

- The client sends an email, password, and name.
- The API creates a user and stores it in the database.
- Returns the user information.

2. Place Creation (POST /api/places)

- Authentication verification.
- Creation and storage of the place in the database.
- Returns the details of the created place.

3. Adding a Review to a Place (POST /api/places/{id}/reviews)

- Authentication verification.
- Retrieval of place information.
- Creation and storage of the review in the database.
- Returns the details of the added review.

4. Searching for Places with Filters (GET /api/places/filters)

- The client sends search criteria.
- The API retrieves matching places from the database.
- Returns the list of filtered places.

UML Class Diagram Documentation

1. What is a UML Class Diagram?

A UML class diagram is a graphical representation of classes, their attributes, methods, and relationships within a system. It is used to model a program's structure before development.

Each class defines a set of objects sharing the same characteristics (attributes) and behaviors (methods). The relationships between classes illustrate how they interact with each other.

2. Diagram Analysis

This diagram represents the main entities and their relationships in a place and review management system.

2.1. Classes and Their Attributes

1. User

- **Attributes:** First name, last name, email, password, admin role
- **Methods:** Register, update profile, delete account

2. Place

- **Attributes:** Title, description, price, latitude, longitude
- **Methods:** Create, update, delete, list

3. Review

- **Attributes:** Rating, comment
- **Methods:** Create, update, delete, list reviews for a place

4. Amenity

- **Attributes:** Name, description
- **Methods:** Create, update, delete, list

3. Relationships Between Classes

- A **User** owns one or more **Places**.
- A **User** writes **Reviews** on **Places**.
- A **Place** receives **Reviews**.
- A **Place** has **Amenities**.

Three-Tier Architecture Documentation

1. What is Three-Tier Architecture?

The **Three-Tier Architecture** is a software design model that divides an application into three distinct levels. This separation allows for better code organization, easier maintenance, and optimal scalability.

Each layer has a defined role and communicates with the others to ensure proper system functionality.

2. The Three Layers of the Architecture

1. Presentation Layer

- Contains the user interface (*User Interface*).
- It is the visible part of the application with which the user interacts.
- It sends user actions to the business layer.

2. Business Layer

- Manages business logic.
- Contains the main entities: **User, Place, Review, Amenity**.
- This is where business rules are applied (e.g., permission checks, calculations, data validation).

3. Persistence Layer

- Manages data access and external services.
- Contains **Data Access** (database access) and **Service Agents** (communication with APIs or external services).
- Stores and retrieves information requested by the business layer.

This three-tier structure ensures a clear separation of responsibilities, making development, maintenance, and system evolution easier. 🚀