HBnB Evolution application

Este documento técnico proporciona una descripción general completa del diseño arquitectónico del proyecto HBnB. Sirve como modelo para las fases de desarrollo e implementación, lo que garantiza una comprensión coherente de la estructura y las interacciones del sistema.

El documento incluye los siguientes componentes:

Diagrama de paquete de alto nivel: ilustra la arquitectura general de la aplicación, mostrando la separación de las preocupaciones en capas distintas.

Diagrama de clase detallado para la capa de lógica empresarial: representa las entidades, sus relaciones y la lógica empresarial central de la aplicación.

Diagramas de secuencia para llamadas de API: presenta el flujo de interacciones entre componentes para llamadas de API seleccionadas, lo que demuestra el comportamiento del sistema.

Arquitectura de alto nivel

El proyecto HBnB adopta una arquitectura en capas para promover la modularidad y la capacidad de mantenimiento. Las capas principales son:

Capa de presentación: maneja la interfaz de usuario y las interacciones del usuario.

Capa de lógica empresarial: encapsula las reglas y la lógica empresariales centrales de la aplicación.

Capa de persistencia: administra la interacción con el almacenamiento de datos subyacente.

En la capa de lógica empresarial se emplea un patrón de fachada para proporcionar una interfaz simplificada para interactuar con las funcionalidades del sistema.

Imagen que contiene Diagrama

Descripción generada automáticamente

Capa de lógica empresarial

La capa de lógica empresarial es responsable de implementar las funcionalidades básicas de la aplicación HBnB. Incluye las siguientes entidades claves:

User: representa a los usuarios de la plataforma, incluida su información personal y credenciales de autenticación, en esta clase se deben identificar 2 tipos de usuario

El dueño del lugar y el huésped

Place: representa listados de lugares disponibles para reservar.

Review: representa opiniones de los usuarios sobre los lugares.

Amenity: representa funciones o servicios adicionales ofrecidos por los lugares.

Texto

Descripción generada automáticamente con confianza media

En este diagrama especifique los vínculos que tieen clada clase, decidí crear una superclase la cual ayudara para heredar créate\_date, update\_date, IDD ya que son atributos que se necesitan en las demás clases,

relaciones entre las entidades:

User - Place: una relación de uno a muchos. Un usuario puede crear y postear varios lugares, pero un lugar solo puede pertenecer a un usuario.

User - Review: una relación de uno a muchos. Un usuario puede escribir varias opiniones, pero una opinión solo puede ser escrita por un usuario.

Place - Amenity: una relación de muchos a muchos. Un lugar puede tener varios servicios y un servicio puede estar asociado con varios lugares.

Place - Review: una relación de uno a muchos. Un lugar puede tener varias opiniones, pero una opinión solo puede estar asociada con un lugar.

**Flujo de interacción de la API**

Los siguientes diagramas de secuencia ilustran las interacciones entre los componentes para llamadas de API seleccionadas:

1. **User Registration:** A user signs up for a new account.

Diagrama

Descripción generada automáticamente

Para mayor entendimiento del diagrama brindare un ejemplo

* **El usuario inicia el registro:** El usuario ingresa sus datos de registro en la aplicación.
* **La aplicación envía la solicitud:** La aplicación envía la solicitud al API con los datos del usuario
* **El API procesa:** El API recibe la solicitud y la pasa al módulo de lógica de negocio para procesar los filtros.
* **La lógica de negocio consulta la base de datos:** El módulo de lógica de negocio consulta la base de datos si el usuario existe
* **Posible error de existencia de usuario:** si ocurre que el usuario ya existe la capa de lógica de negocio retorna un error a la api
* **Resultado exitoso:** si el usuario no existe, procede a crearlo si todo sale bien retorna exitoso si no retorna error
* **La API devuelve los resultados:** El API devuelve exitoso o fracaso el registro

1. **Place Creation:** A user creates a new place listing.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

Para mayor entendimiento del diagrama brindare un ejemplo

* **El usuario inicia la creación del lugar:** el usuario ingresa los datos del lugar
* **La aplicación envía la solicitud:** La aplicación envía la solicitud al API con los datos del lugar
* **El API procesa:** El API recibe la solicitud y la pasa al módulo de lógica de negocio para procesar los datos del lugar
* **La lógica de negocio consulta la base de datos:** El módulo de lógica de negocio consulta la base de datos si el lugar existe
* **Posible error de existencia de lugar:** si ocurre que el usuario ya existe la capa de lógica de negocio retorna un error a la api
* **Resultado exitoso:** si el lugar no existe, procede a crearlo si todo sale bien retorna exitoso si no retorna error
* **La API devuelve los resultados:** El API devuelve exitoso o fracaso la creación del lugar

1. **Review Submission:** A user submits a review for a place.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente con confianza media

Para mayor entendimiento del diagrama brindare un ejemplo

* **El usuario envía una reseña:** El usuario escribe una reseña sobre un lugar en la aplicación.
* **La aplicación envía la solicitud:** La aplicación envía la reseña al API.
* **El API valida la reseña:** El API pasa la reseña al módulo de lógica de negocio para validarla.
* **La lógica de negocio verifica la existencia del lugar:** El módulo de lógica de negocio consulta la base de datos para verificar si el lugar existe.
* **Si el lugar no existe:** Si el lugar no existe, el módulo de lógica de negocio devuelve un error al API.
* **Si el lugar existe:** Si el lugar existe, el módulo de lógica de negocio crea la reseña en la base de datos.
* **Resultado de creación de reseña:** La base de datos devuelve el resultado de la operación de creación (éxito o fracaso).
* **La lógica de negocio devuelve el resultado al API:** El módulo de lógica de negocio devuelve el resultado al API.
* **El API devuelve la respuesta al usuario:** El API devuelve una respuesta al usuario indicando si la reseña se creó correctamente o si ocurrió un error.

1. **Fetching a List of Places:** A user requests a list of places based on certain criteria.

Diagrama

Descripción generada automáticamente

Para mayor entendimiento del diagrama brindare un ejemplo

* **El usuario inicia la búsqueda:** El usuario ingresa sus filtros de búsqueda en la aplicación.
* **La aplicación envía la solicitud:** La aplicación envía la solicitud al API con los filtros del usuario.
* **El API procesa los filtros:** El API recibe la solicitud y la pasa al módulo de lógica de negocio para procesar los filtros.
* **La lógica de negocio consulta la base de datos:** El módulo de lógica de negocio consulta la base de datos con los filtros procesados para obtener los lugares relevantes.
* **Posible error de conexión:** Si ocurre un error de conexión con la base de datos, el módulo de lógica de negocio lo detecta y lo devuelve al API.
* **Resultado exitoso:** Si la consulta a la base de datos es exitosa, el módulo de lógica de negocio devuelve los resultados al API.
* **La API devuelve los resultados:** El API devuelve la lista de lugares al usuario, mostrándolos en la aplicación.