Malik
Bouanani

Christophe
Barrere

Jeremy
Laurens

October 2025

# Holberton Project Hbnb

Documentation compilation

# Table of Contents

1. **Introduction**..........3
2. **High-Level Architecture**..........4
    a. Hight-level Diagram Package..........4
3. **Business Logic Layer**..........5
    a. Detailed class diagram..........6
4. **API Interaction Flow**..........7
    a. Sequence Diagram : User registration..........8
    b. Sequence Diagram : Place Creation..........9
    c. Sequence Diagram : Review Submission..........10
    d. Sequence Diagram : Place List..........11

# Introduction

## Technical Documentation of the HBnB Project!

This document aims to create a complete technical reference that gathers all diagrams and explanations to guide the implementation of the project.

The Project: as part of our Holberton curriculum, we must develop a simplified version of the Airbnb application, named HBnB.

The goal of this project is to apply the fundamental concepts of object-oriented programming, data management, and software architecture. We will be required to design and document the different stages of the application, from the database to the user interface, including the API.

This work will serve as a reference throughout development, ensuring a shared understanding among team members and guaranteeing consistency in the implementation. Writing this technical document is also an opportunity to acquire professional best practices in planning, design, and software documentation.

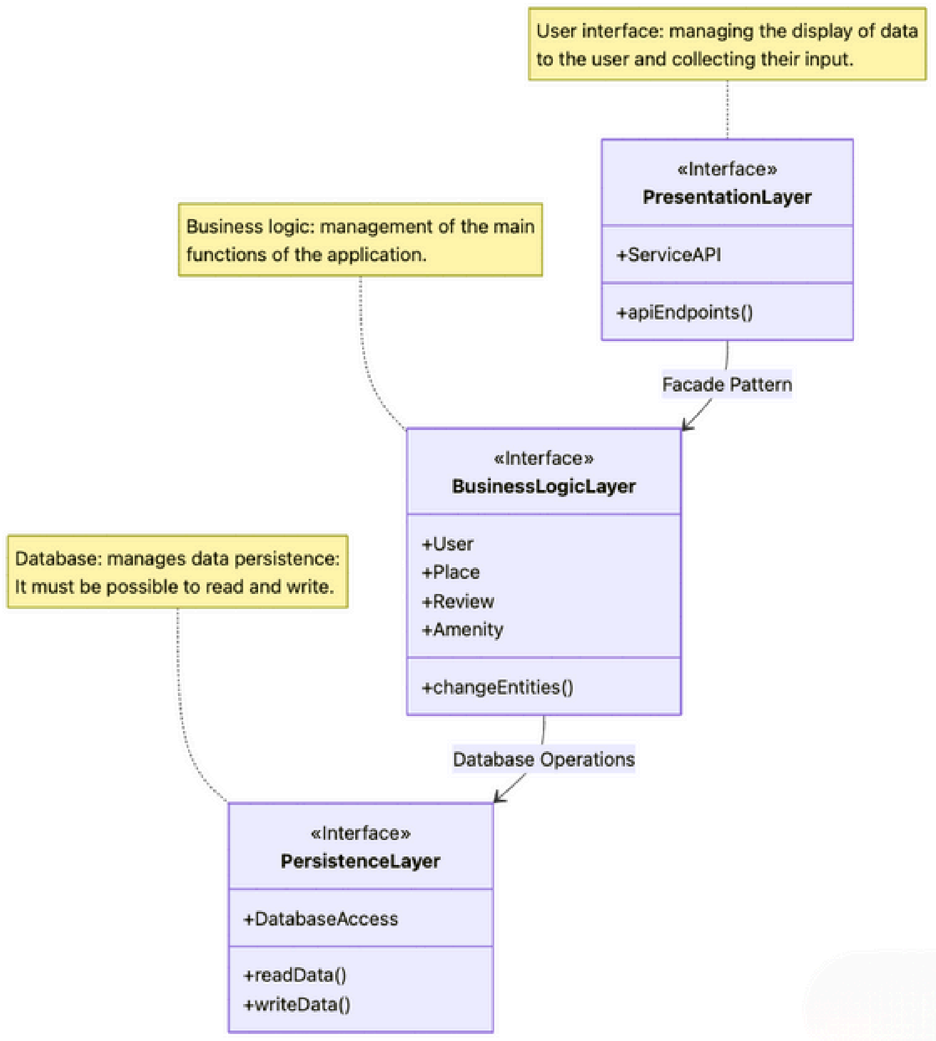
# High-Level Architecture

The application is based on a well-defined three-layer architecture:

- Presentation
- Business Logic
- Persistence

This organization ensures a clear separation of responsibilities and makes long-term maintenance easier.

We also use the Facade design pattern, which provides a simplified interface to the upper layers while hiding internal complexity.



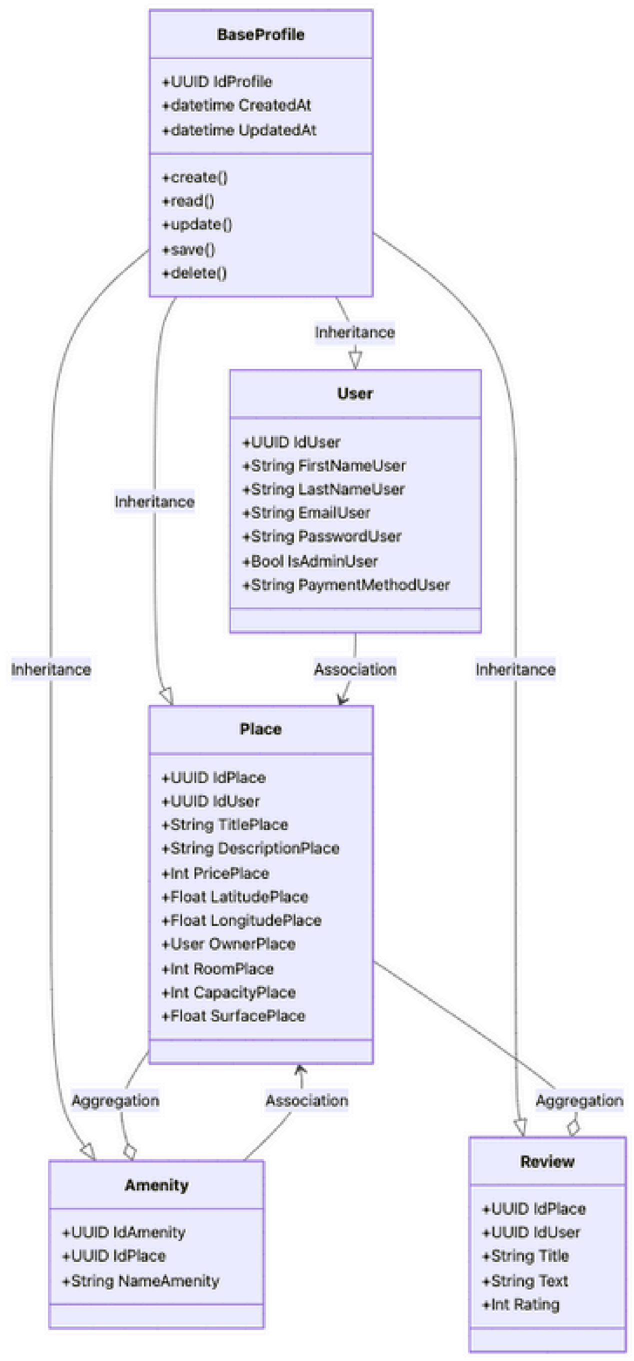
User interface: managing the display of data to the user and collecting their input.
«Interface»
PresentationLayer
+ServiceAPI
+apiEndpoints()
Facade Pattern
Business logic: management of the main functions of the application.
«Interface»
BusinessLogicLayer
+User
+Place
+Review
+Amenity
+changeEntities()
Database Operations
Database: manages data persistence: It must be possible to read and write.
«Interface»
PersistenceLayer
+DatabaseAccess
+readData()
+writeData()

Each layer plays a specific role:

- PresentationLayer: manages the interface and data collection.
- BusinessLogicLayer: contains the business logic (validation, rules, interactions).
- PersistenceLayer: handles saving to and reading from the database.

# Business Logic Layer

Business Logic. This is where we define our main entities (users, places, reviews, amenities) and their relationships.



The main design choices:

- BaseProfile serves as the parent class to factorize common fields.
- User represents our users with their personal information and role.
- Place manages the accommodations published and linked to users.
- Review allows users to leave feedback.
- Amenity describes the amenities associated with the places.

# Business Logic Layer

Explanatory Notes

Purpose of the diagram: describe the business entities and their shared lifecycle via BaseProfile.

Entities & roles:

- BaseProfile: factors out Id*, CreatedAt, UpdatedAt, and CRUD operations.
- User: account info & role (IsAdminUser), payment method.
- Place: listing published by a User, with geolocation info and capacity.
- Review: review of a Place by a User (rating + text).
- Amenity: amenities (Wi-Fi, parking, etc.) attached to a Place.

Relationships:

- User owns Place (association).
- Place aggregates Amenity and Review (linked lifecycle).

Design decisions:

- Inheritance from BaseProfile to standardize traceability.
- Aggregation for Amenity/Review to preserve integrity constraints (logical cascade to be defined on the persistence side).
- OwnerPlace in Place: convenient shortcut to avoid an extra fetch (optional depending on the hydration strategy).

Global integration: these classes are the core objects manipulated by the Facade's use cases.

# API Interaction Flow

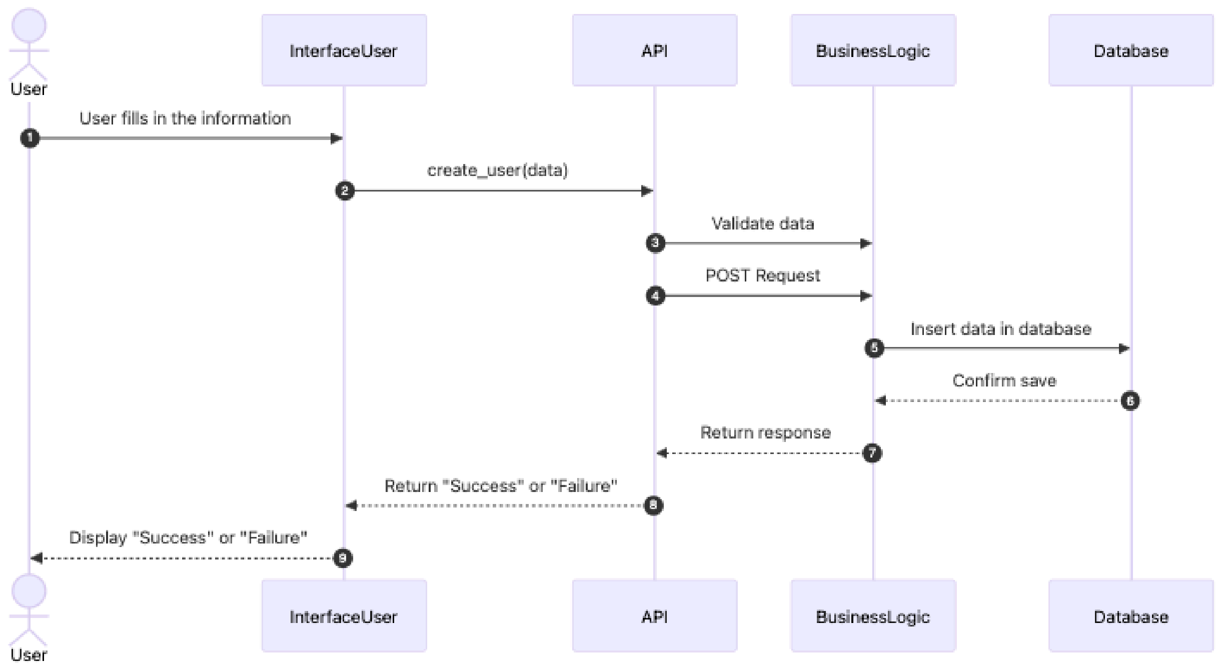
The API interaction flow describes how the different layers of our HBnB application collaborate when the user triggers an action via the interface (e.g., sign-up, creating a place, searching, or adding a review).

- The user acts from the interface (UI) — they fill out a form, click a button, or start a search.
- The interface sends the request to the API. The API serves as the entry point: it receives the request, translates it into a standard format, and checks surface-level parameters.
- The business logic takes over. It validates project-specific rules (uniqueness of an email, consistency of a place's coordinates, prohibition on posting duplicate reviews, etc.).
- The persistence layer steps in when data needs to be saved or read. The business logic calls the database through this specialized layer.
- Information returns: once the database is confirmed or the data is retrieved, the response propagates back up to the API, then to the interface, and finally to the user, who sees the result displayed.

# API Interaction Flow

## User Registration

Goal: create an account.
Decisions: business validation in the BusinessLogic layer (e.g., email uniqueness); password hashing handled in the domain (business) layer or via a persistence adapter

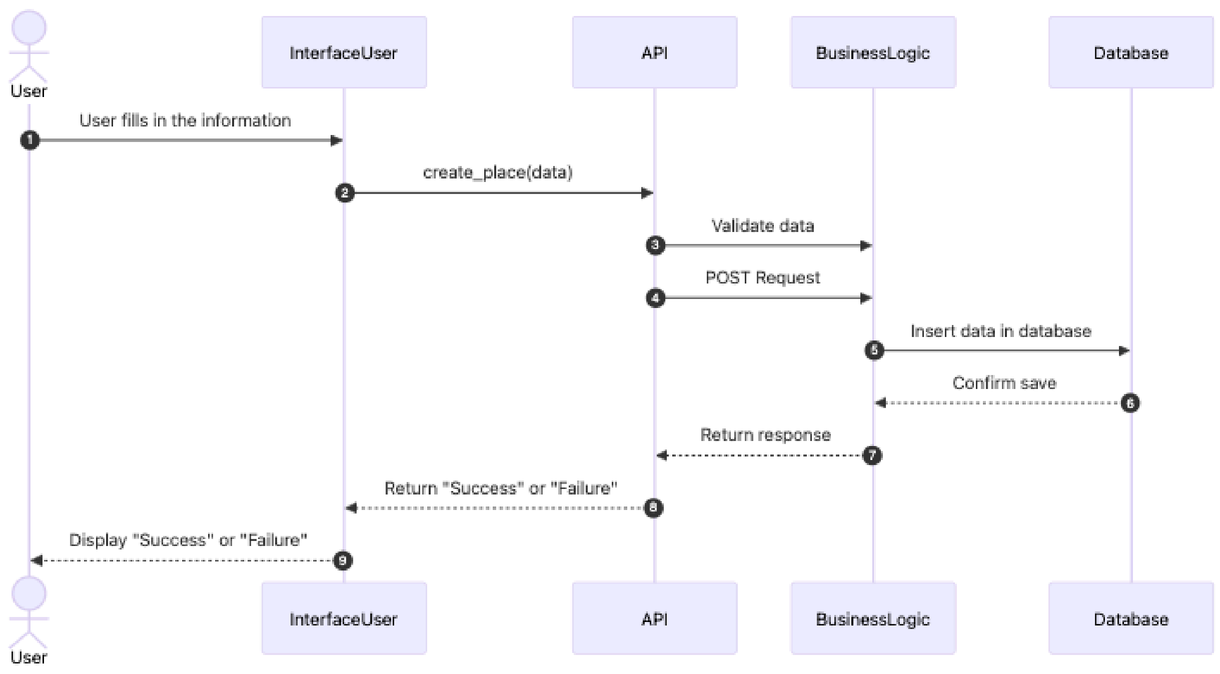


This diagram illustrates the scenario:
user registration. It shows the central role of the API as an intermediary between the user interface, the business logic, and the database.

# API Interaction Flow

## Place Creation

Goal: publish a Place.
Decisions: authorization check (the current User becomes OwnerPlace), GPS coordinates normalization.



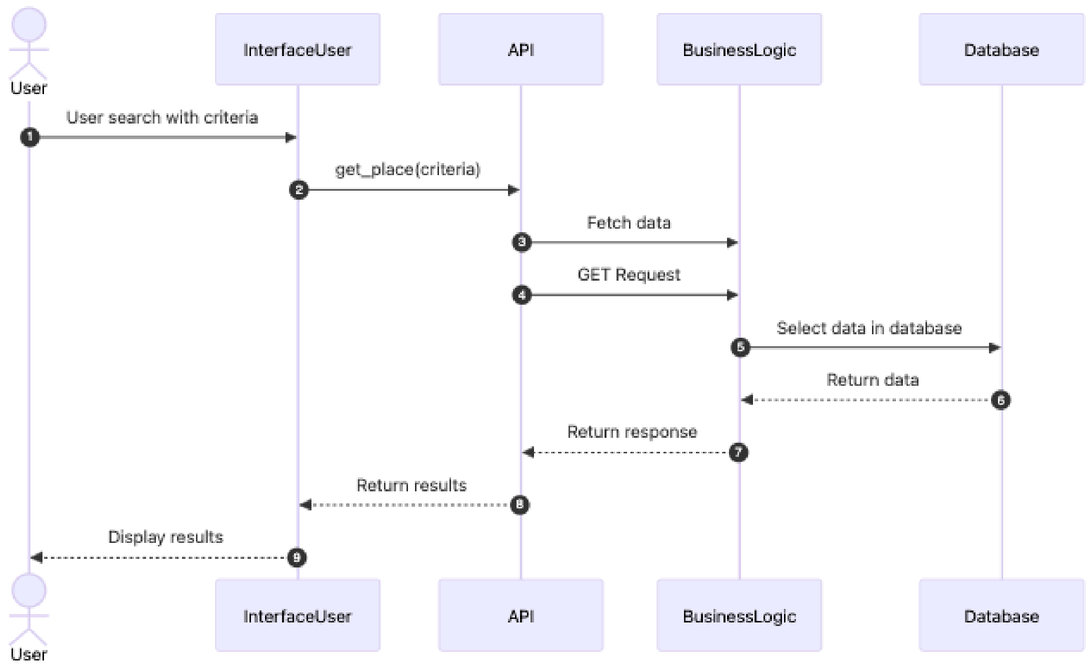
This diagram illustrates the scenario:
creation of a place. It highlights the central role of the API as an intermediary between the user interface, the business logic, and the database.

# API Interaction Flow

## Place List

Notes

Goal: list Places by criteria.
Decisions: mandatory pagination, sorting (price, rating, distance), lightweight DTO (no sensitive fields).



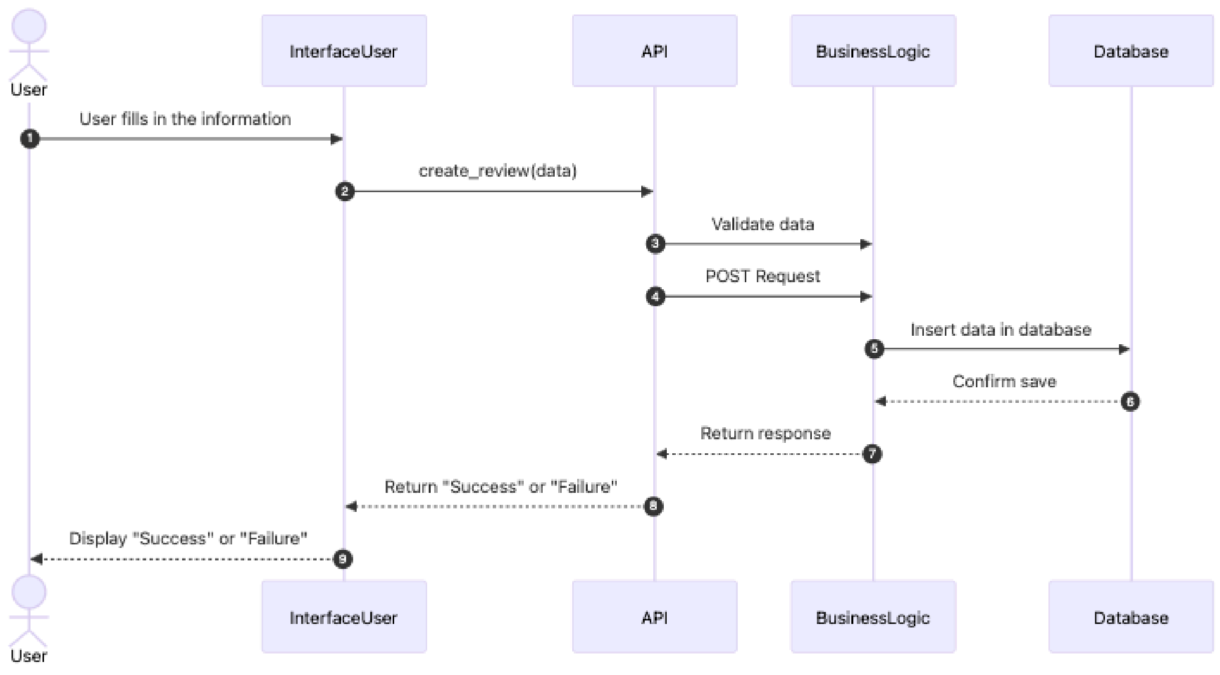
This diagram illustrates the scenario:
searching for places. It highlights the central role of the API as an intermediary between the user interface, the business logic, and the database.

## API Interaction Flow

### Review Submission

Notes

Goal: create a Review.
Decisions: business checks (a user ≠ multiple reviews for the same Place?), recalculation of the aggregated rating on the Place side.



This diagram illustrates the scenario:
submitting a review. It highlights the central role of the API as an intermediary between the user interface, the business logic, and the database.