Hbnb Evolution

Author: Fernando L. Albert Cotto

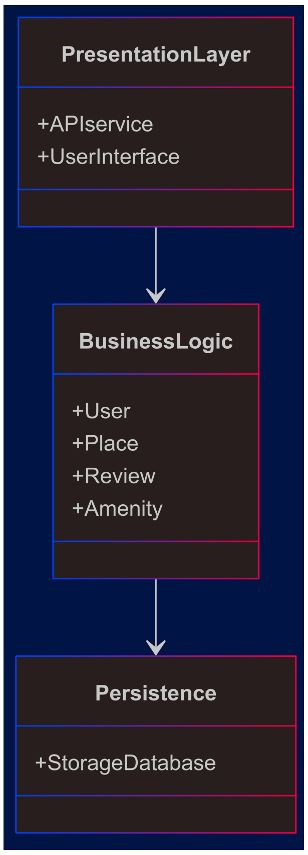
Content

Introduction

IntroductionThis is a full technical documentation about the HBnB Evolution application. This document will serve as a technical blueprint for the development of the application. In this document will find, the high level of the system architecture, business logic flows and major API calls to provide a solid design for the application deployment.

High-Level Architecture

High-Level Package Diagram

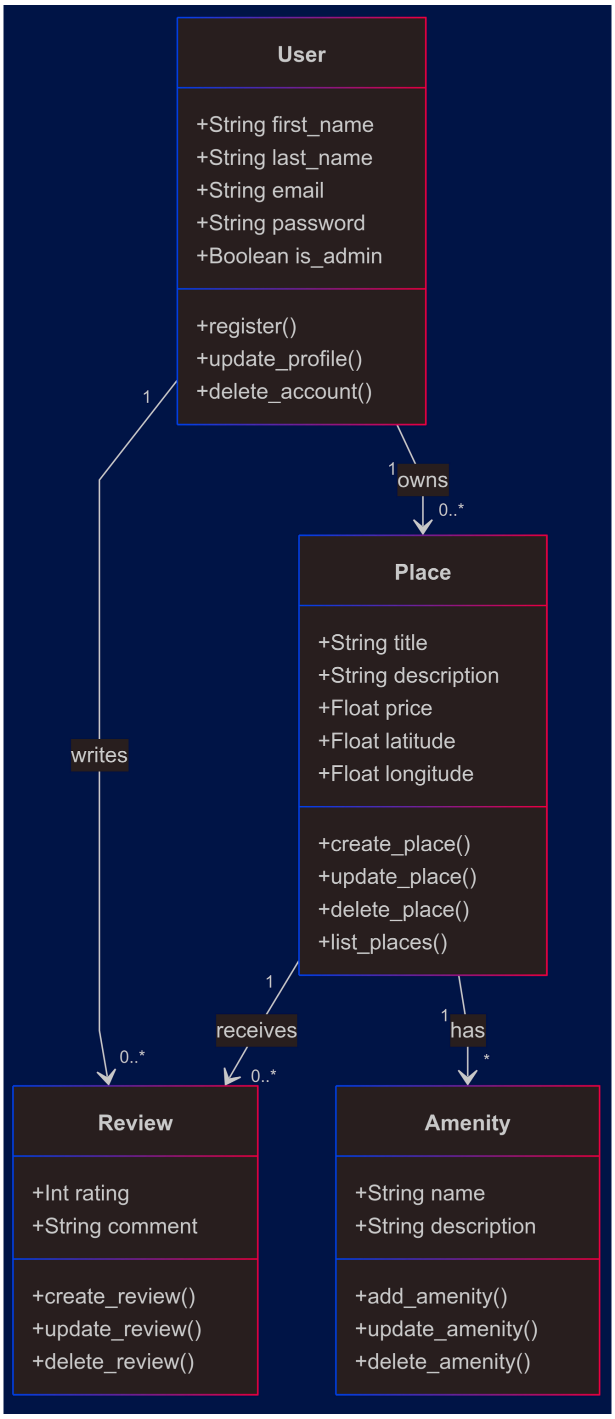


```
classDiagram
direction TB
  class PresentationLayer {
    +APIservice
    +UserInterface
  }
  class BusinessLogic {
    +User
    +Place
    +Review
    +Amenity
  }
  class Persistence {
    +StorageDatabase
  }
  PresentationLayer --> BusinessLogic
  BusinessLogic --> Persistence
```

Layered Architecture

- **Presentation Layer** - Handles user interaction and API request.
- **Business Logic Layer** - Implements business rules and entity
- **Persitence Layer** - Handle data storage and retrival.

Business Logic Layer



```
classDiagram
 class User {
  +String first_name
  +String last_name
  +String email
  +String password
  +Boolean is_admin
  +register()
  +update_profile()
  +delete_account()
 }
 class Place {
  +String title
  +String description
  +Float price
  +Float latitude
  +Float longitude
  +create_place()
  +update_place()
  +delete_place()
  +list_places()
 }
 class Review {
  +Int rating
  +String comment
  +create_review()
  +update_review()
  +delete_review()
 }
 class Amenity {
  +String name
  +String description
  +add_amenity()
  +update_amenity()
  +delete_amenity()
 }
 User "1" --> "0..*" Place : owns
 User "1" --> "0..*" Review : writes
 Place "1" --> "0..*" Review : receives
 Place "1" --> "*" Amenity : has
```

User Entity

- **Attributes:** first_name, last_name, email, password, is_admin
- **Methods:** register(), update_profile(), delete_account()
- **Relationship:** Users own Places, write Reviews.

Place Entity

- **Attributes**: title, description, price, latitude, longitude
- **Methods:** create_place(), update_place(), delete_place(), list_places()
- **Relationship:** Each place belongs to a User, can have multiple Amenities, and receive Reviews.

Review Entity

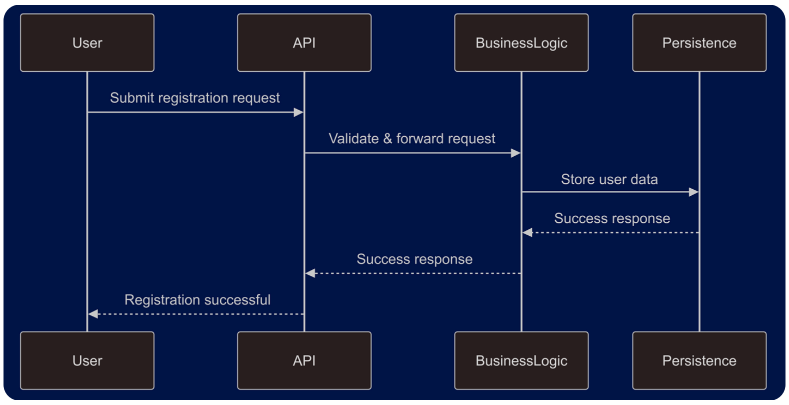
- **Attributes:** rating, comment
- **Methods:** create_review(), update_review(), delete_review()
- **Relationship:** Each Review is associated with a Place and a User.

Amenity Entity

- **Attributes:** name, description
- **Methods:** add_amenity(), update_amenity(), delete_amenity()
- **Relationship:** Amenities are associated with multiple Places.

User Registration

- User submits registration request.
- API validates input and forwards request to Business Logic Layer.
- Business Logic Layer stores user data in Persistence Layer.
- Persistence Layer returns success response.

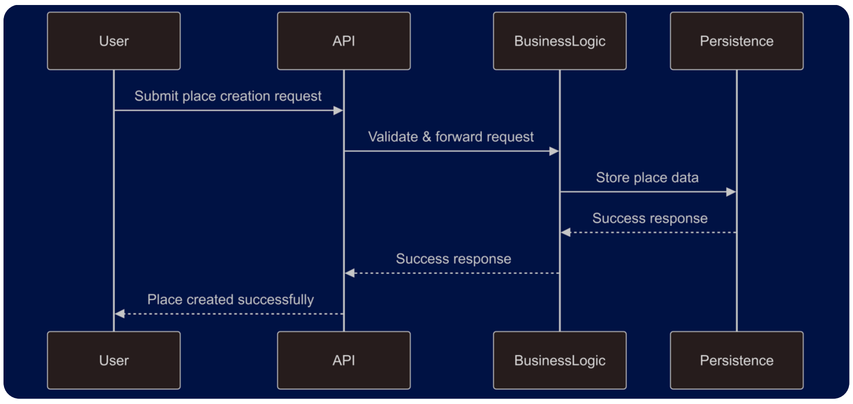


```
sequenceDiagram
  participant User
  participant API
  participant BusinessLogic
  participant Persistence
  User->>API: Submit registration request
  API->>BusinessLogic: Validate & forward request
  BusinessLogic->>Persistence: Store user data
  Persistence-->>BusinessLogic: Success response
  BusinessLogic-->>API: Success response
  API-->>User: Registration successful
```

Place Creation

- User submits place creation request.
- API validates input and forwards request to Business Logic Layer.

- Business Logic Layer associates place with user and stores data.
- Persistence Layer returns success response.

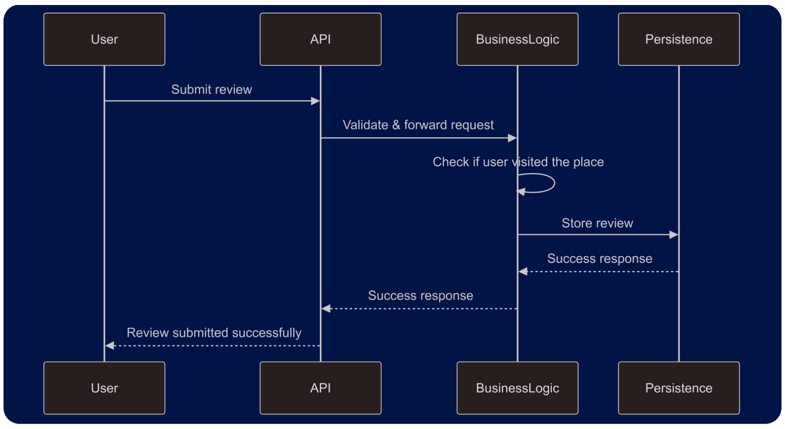


```
sequenceDiagram
 participant User
  participant API
  participant BusinessLogic
  participant Persistence
  User->>API: Submit place creation request
  API->>BusinessLogic: Validate & forward request
  BusinessLogic->>Persistence: Store place data
  Persistence-->>BusinessLogic: Success response
  BusinessLogic-->>API: Success response

  API-->>User: Place created successfully
```

Review Submission

- User submits a review for a place.
- API validates input and forwards request to Business Logic Layer.
- Business Logic Layer ensures user has visited the place.
- Persistence Layer stores review and returns success response.



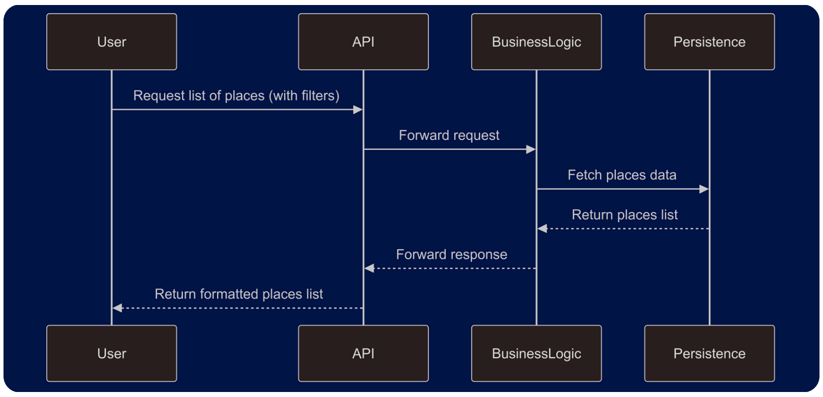
```
sequenceDiagram
  participant User
  participant API
  participant BusinessLogic
  participant Persistence
  User->>API: Submit review
  API->>BusinessLogic: Validate & forward request
  BusinessLogic->>BusinessLogic: Check if user visited the
place
  BusinessLogic->>Persistence: Store review
  Persistence-->>BusinessLogic: Success response
  BusinessLogic-->>API: Success response

  API-->>User: Review submitted successfully
```

Fetching a List of Places

- User requests a list of places with filters.
- API forwards request to Business Logic Layer.
- Business Logic Layer fetches data from Persistence Layer.

- Persistence Layer returns list of places.
- API returns formatted response to user.



```
sequenceDiagram
  participant User
  participant API
  participant BusinessLogic
  participant Persistence
  User->>API: Request list of places (with filters)
  API->>BusinessLogic: Forward request
  BusinessLogic->>Persistence: Fetch places data
  Persistence-->>BusinessLogic: Return places list
  BusinessLogic-->>API: Forward response
  API-->>User: Return formatted places list
```

Conclusion

This technical documentation provides a structured to understand and guide the HBnB Evolution architecture, business logic, and API interactions. By using this blueprint, I can ensure a well-organized implementation of the application.

Contact me:

Name	Github	Discord	Email
@Fernando Albert	@Falbert19	fernandoalbert19	9727@holbertonstudents.com