HBnB Technical Documentation

Introduction :

This document provides a comprehensive technical overview of the HBnB application. It consolidates all architectural and design diagrams created in previous tasks into a single reference.

The purpose of this document is to guide the implementation phases, clarify system architecture, and provide detailed insight into the behavior and interactions of key components.

It includes:

- High-Level Package Diagram — overview of layered architecture
- Business Logic Layer Class Diagram — detailed entities, attributes, methods, and relationships
- API Sequence Diagrams — step-by-step flow for key API calls

1. High-Level Architecture

Overview :

HBnB follows a layered architecture with three main layers:

- Presentation Layer — handles client interactions (REST API, Application Services)
- Business Logic Layer — manages core domain entities, business rules, and Facade interfaces
- Persistence Layer — handles data storage and retrieval via repositories

This structure improves maintainability, scalability, and testability while enforcing separation of concerns.

Layer Responsibilities :

Layer	Responsibilities	Components
Presentation	Handle HTTP requests, validate inputs, format responses	REST API, Application Services
Business Logic	Validate business rules, manage domain entities	Facade, Models (User, Place, Review, Amenity)
Persistence	Store, retrieve, update, delete data	Repository, Database

The Facade provides a unified interface to the Business Logic Layer, reducing coupling between Presentation and domain models.

Benefits:

- Simplifies communication
- Improves maintainability and scalability
- Provides single entry point for business operations

1. **High-Level Package Diagram :**

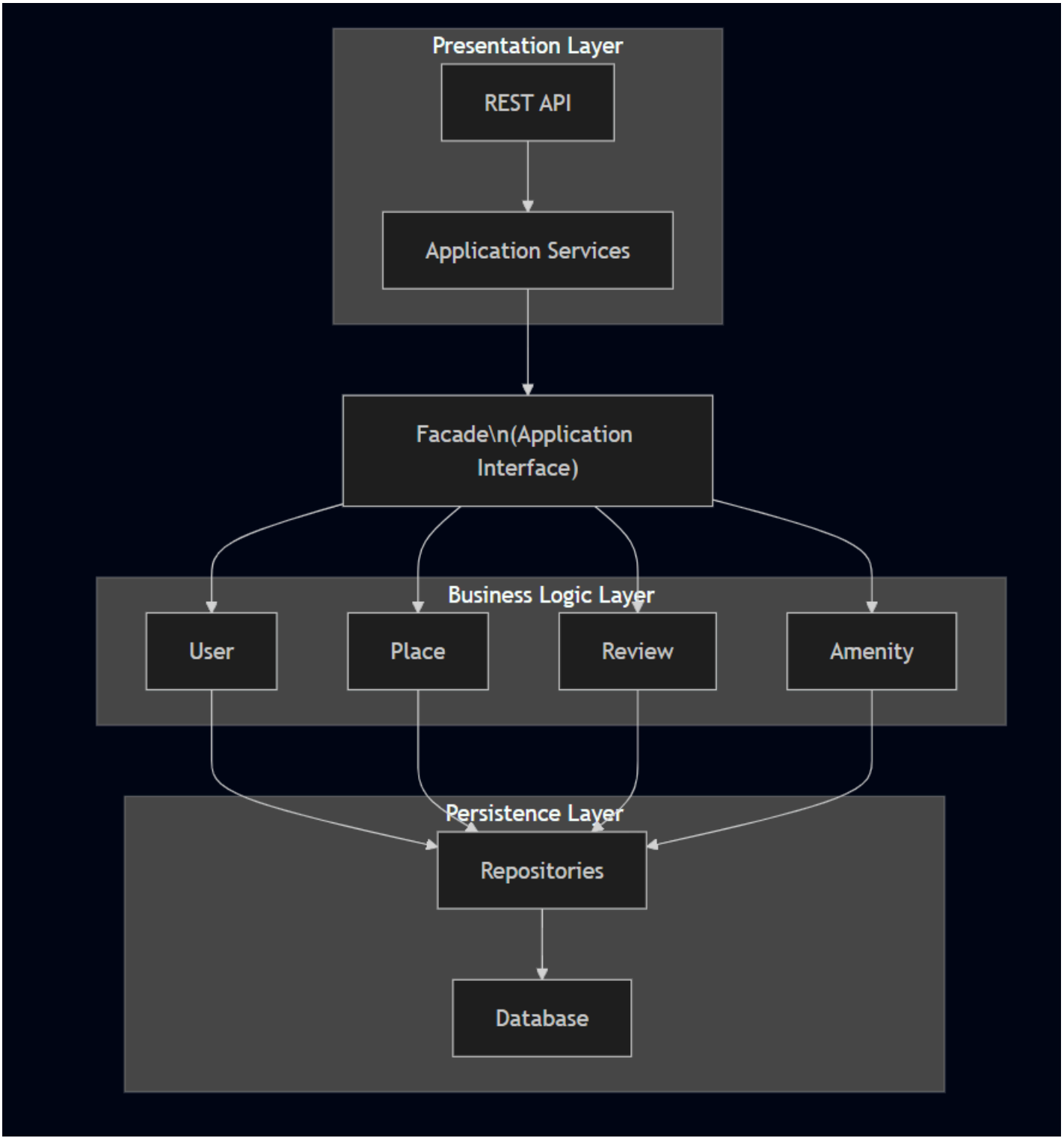


Diagram Explanation:

- Top-to-bottom flow: request moves downward, response moves upward
- Arrows: indicate method calls, communication, and delegation
- Facade: centralizes access to domain models
- Repository: abstracts persistence layer to ensure loose coupling

2. Detailed Class Diagram – Business Logic Layer

Introduction

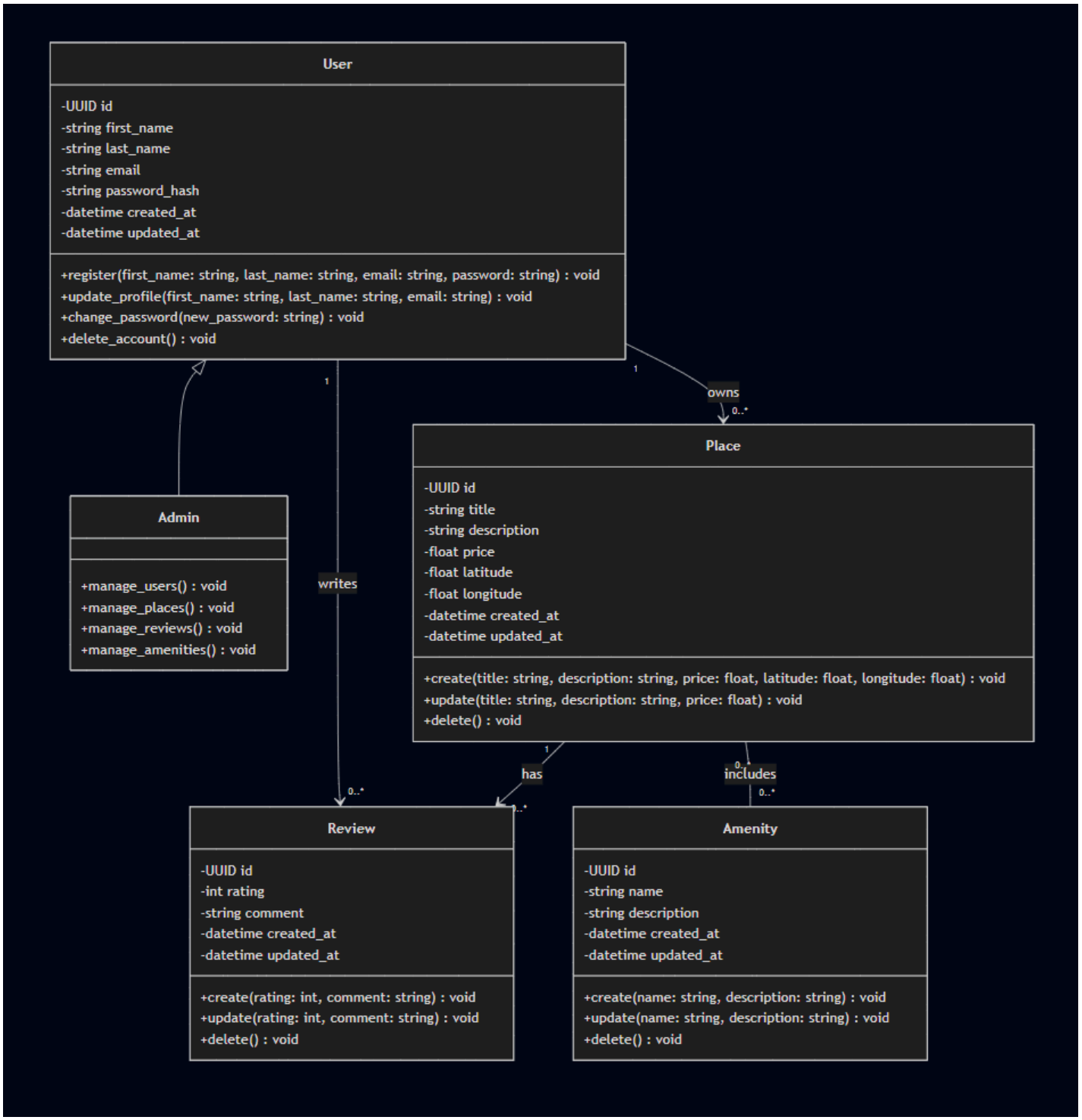
The Business Logic Layer manages core entities and enforces rules. This diagram represents:

Entities: User, Admin, Place, Review, Amenity
Attributes and methods with visibility (private/protected/public)

Relationships and multiplicity rules

Inheritance (Admin extends User)

UML Class Diagram :



Relationships Summary:

User → Place: 1-to-many, solid line, represents ownership

User → Review: 1-to-many, solid line, represents authorship

Place → Review: 1-to-many, solid line, represents feedback on a place

Place ↔ Amenity: many-to-many, dashed line, represents included amenities

3. API Sequence Diagrams

Overview

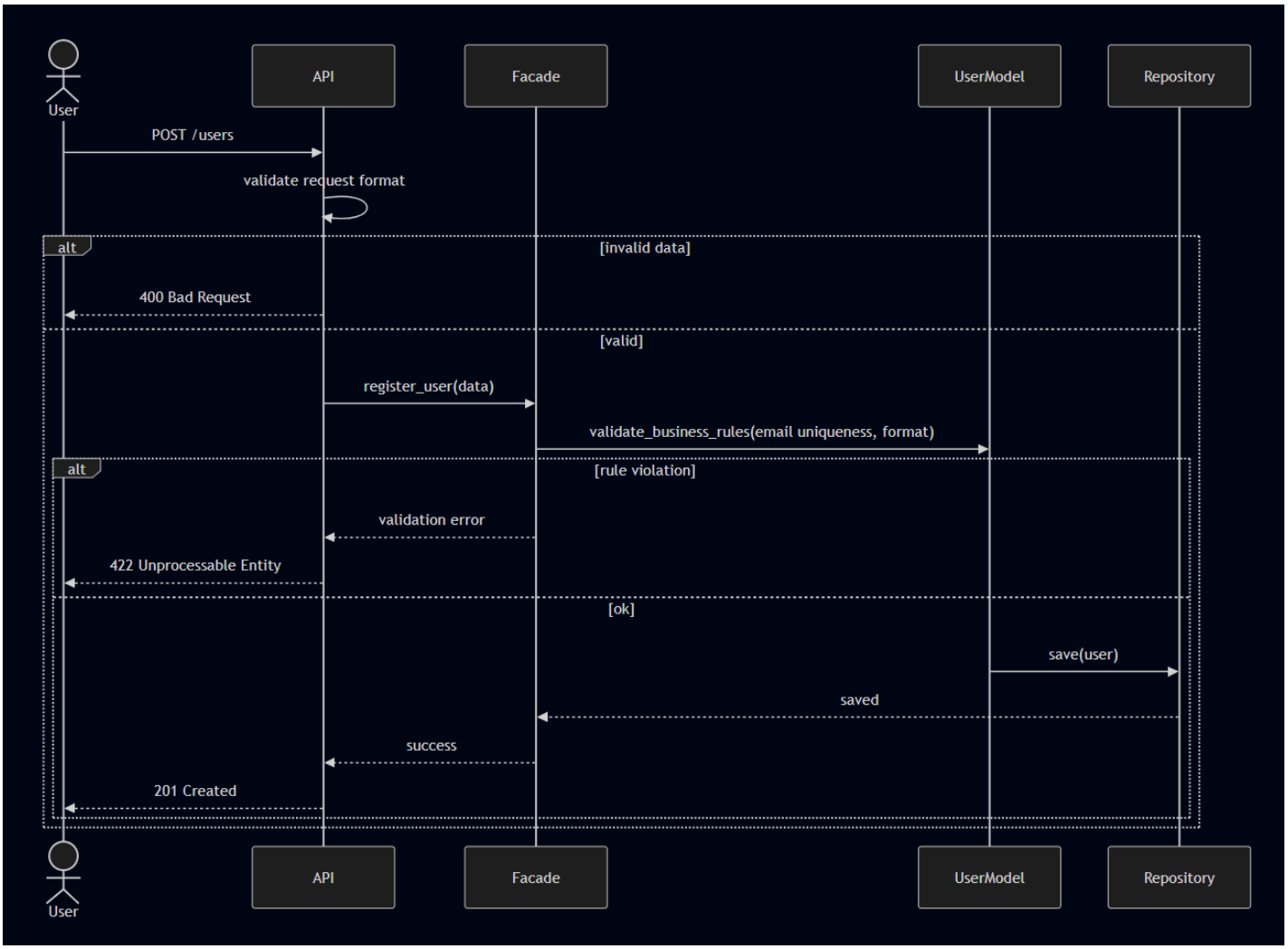
Sequence diagrams illustrate how the layers interact for key API calls. They highlight:

- Input validation and business rule enforcement
- Step-by-step message flow between Presentation, Business Logic, and Persistence
- Success and error responses with HTTP status codes

3.1 User Registration (POST /users) :

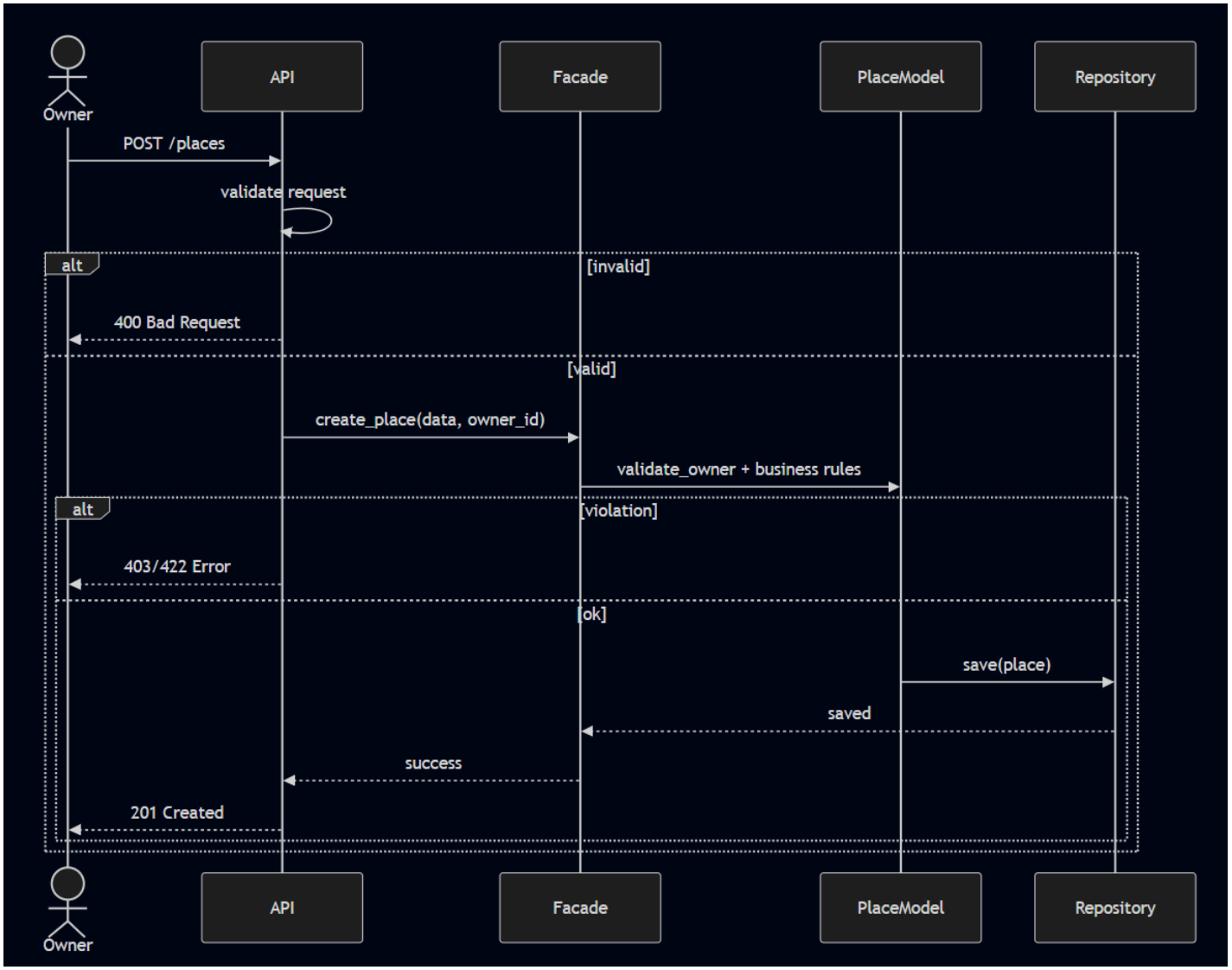
Description:

This diagram shows the full lifecycle of a user registration request. It illustrates how the API validates the request format, how the Business Logic Layer (Facade and UserModel) enforces business rules (e.g., unique email), and how the user is persisted to the database. Error scenarios such as invalid data or rule violations are also represented with appropriate HTTP status codes.



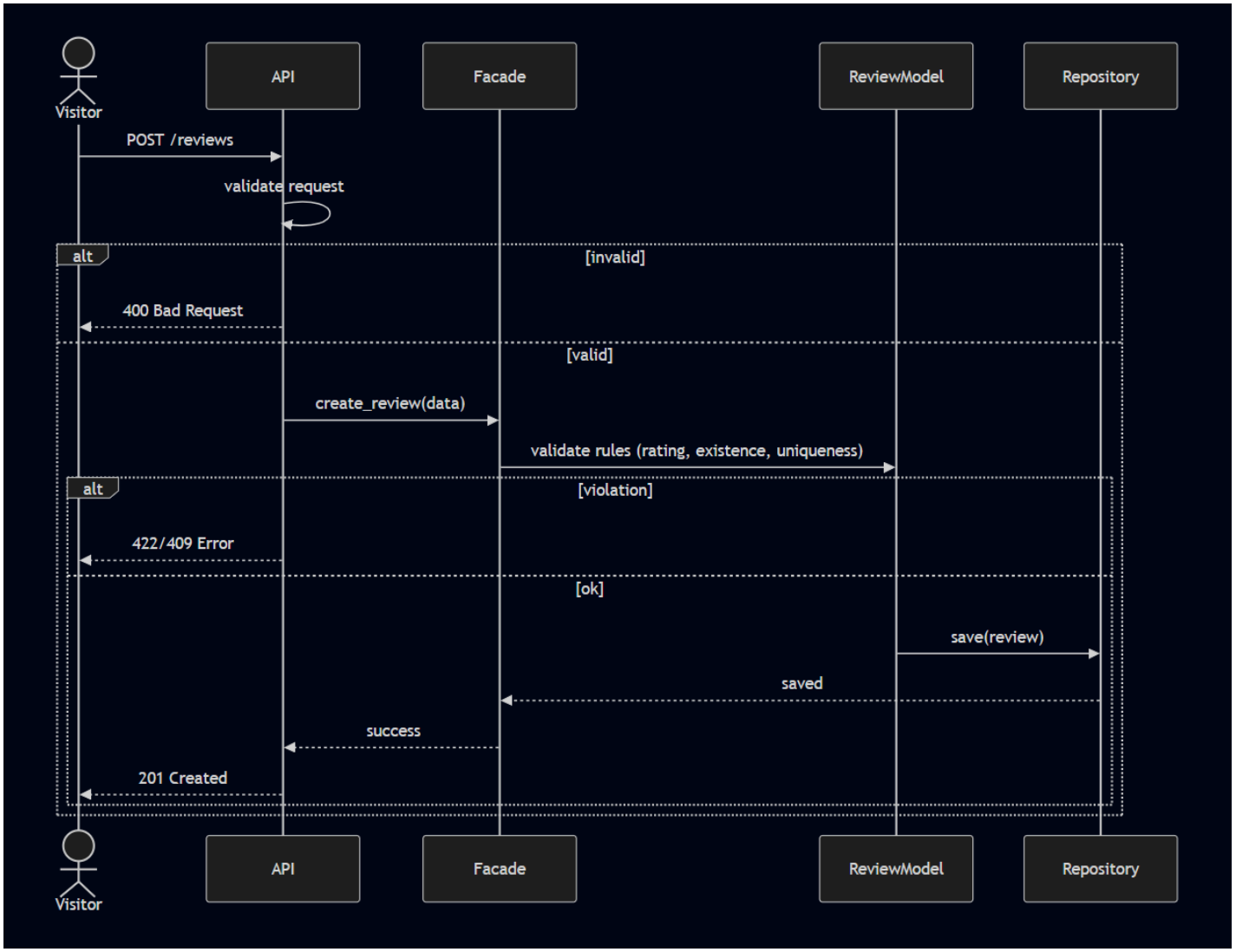
3.2 Place Creation (POST /places) :

Description: This diagram demonstrates how an Owner creates a new Place. It includes request validation, authorization checks, and business rule enforcement. It also shows the persistence step via the Repository and possible error responses such as 400, 403, or 422, depending on the type of violation.



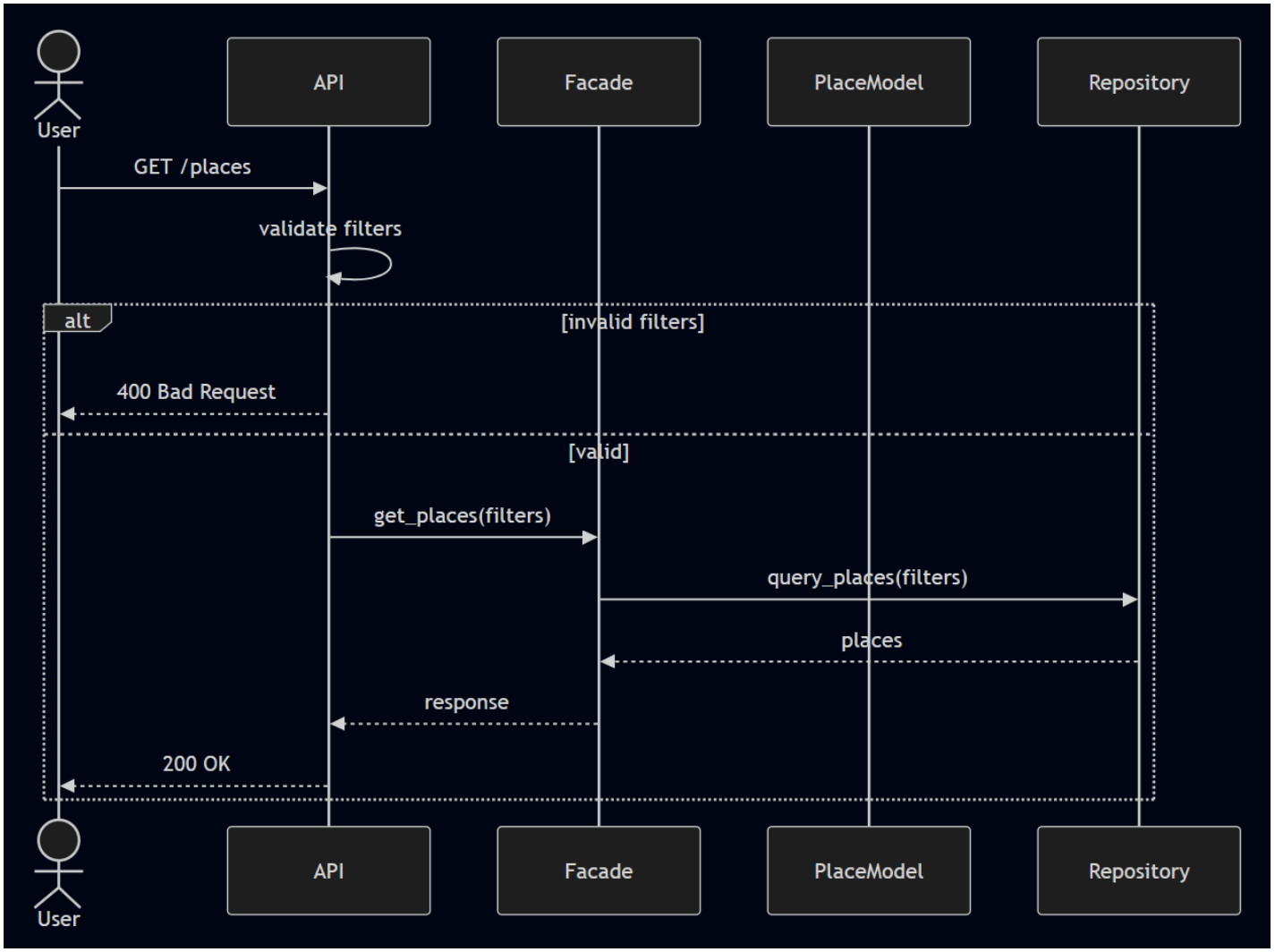
3.3 Review Submission (POST /reviews) :

Description: This diagram illustrates how a Visitor submits a review. It highlights API-level validation, business rule checks (rating limits, uniqueness, and place existence), and persistence. Different error codes (400, 422, 409) are returned depending on the type of failure, while a successful submission returns 201 Created.



3.4 Fetching Places (GET /places) :

Description: This diagram shows how users retrieve a list of Places. It demonstrates API-level query validation, interaction with the Facade and Repository layers, and successful retrieval or error handling (400 Bad Request) for invalid filters. It also emphasizes the distinction between read and write operations.



Conclusion

This comprehensive documentation consolidates the HBnB architecture and system behavior.

Key takeaways:

- Clear separation of layers ensures maintainable and scalable design
- Business Logic Layer enforces domain rules with structured entities and relationships
- Sequence diagrams clarify API call handling, validation, and persistence interactions
- UML conventions and Facade pattern support clean architecture principles

This document serves as a single source of truth for developers, ensuring consistency and clarity throughout implementation and future maintenance.