

□□□XX□□□□□□□□
 □□□XX□□ □□□XX □□□XXXXXXXXX
 □□□□□XX □□□XX
 □□□□□2026 □ X □ X □

本项目使用“前后端分离”模式，后端使用 Spring Boot + MyBatis-Plus，前端使用 Vue 3 + Vite + Element Plus。

This thesis presents the design and implementation of a homestay recommendation system. The frontend is built with Vue 3, Vite, and Element Plus, while the backend leverages Spring Boot and MyBatis-Plus, combining collaborative filtering and content-based recommendation to provide personalized listings, online booking, and host property management. The work covers background, requirement analysis, system architecture, key technologies, database and process design, implementation, and testing.

1. □□
2. □□□□□□□□
3. □□□□
4. □□□□
5. □□□□
6. □□□□□□ER □□
7. □□□□□□□□□□
8. □□□□□□□□
9. □□□□□□
10. □□□□□□
11. □□□□□□
12. □□

1.1 □□□□

项目整体架构采用前后端分离模式，前端采用Vue 3 + Vite，后端采用Spring Boot + MyBatis-Plus。数据库采用MySQL，缓存采用Redis。项目部署采用Docker容器化技术。

项目采用模块化设计，前端分为基础组件库、业务组件库和页面层。后端分为基础服务层、业务逻辑层和接口层。数据库采用ER模型设计，确保数据的一致性和完整性。

项目采用RESTful API设计风格，接口文档采用Swagger生成。项目采用Git进行版本控制，采用Jenkins进行自动化部署。

1.2 技术栈

- 前端技术栈：Vue 3、Vite、Element Plus、Pinia、Vue Router、Axios
- 后端技术栈：Spring Boot、MyBatis-Plus、JWT、Spring Validation、Lombok
- 数据库：MySQL、Redis

1.3 部署环境

项目部署环境要求如下：

2. 项目依赖

- 前端依赖：Vue 3、Vite、Element Plus、Pinia、Vue Router、Axios
- 后端依赖：Spring Boot、MyBatis-Plus、JWT、Spring Validation、Lombok
- 数据库依赖：MySQL、Redis

3. 项目结构

- 前端结构：src/目录下包含components、views、utils、api等子目录。
- 后端结构：src/main/java目录下包含controller、service、entity、dto等子目录。
- 数据库结构：src/main/resources目录下包含schema.sql、data.sql等文件。
- 配置文件：src/main/resources目录下包含application.yml、logback-spring.xml等文件。

3.1 前端结构

- src/目录下包含components、views、utils、api等子目录。
- components目录下包含基础组件和业务组件。
- views目录下包含页面组件。
- utils目录下包含工具类。

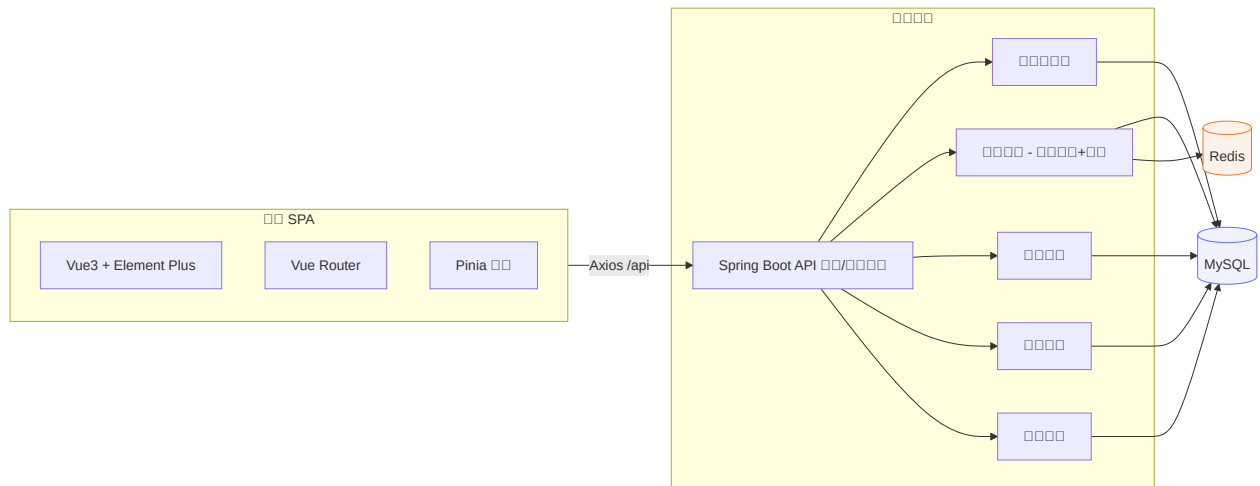
3.2 后端结构

- JWT Token生成和验证逻辑。
- 数据库连接池配置和事务管理。
- 日志配置和异常处理。

3.3 数据库结构

- 前后端分离/前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统
- 前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统
- 前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统 token
- 前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统 HTTPS Token 前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统/前后端分离系统

4. 前后端分离



4.1 前后端分离

flowchart LR

```

subgraph Client[Client SPA]
    UI[Vue3 + Element Plus]
    Router[Vue Router]
    Store[Pinia]
end

subgraph Backend[Backend]
    APIGW[Spring Boot API]
    Auth[Auth]
    Rec[Rec]
    Order[Order]
    Property[Property]
    UserSvc[UserSvc]
end

```

```

DB[(MySQL)]:::db
Cache[(Redis)]:::cache

```

```

Client -->|Axios /api| APIGW
APIGW --> Auth
APIGW --> Rec
APIGW --> Order
APIGW --> Property
APIGW --> UserSvc
Auth --> DB

```

```
Rec --> DB
Order --> DB
Property --> DB
UserSvc --> DB
Rec --> Cache
classDef db fill:#f2f2ff,stroke:#6370f4;
classDef cache fill:#fdf2e9,stroke:#e67e22;
```

4.2

flowchart TB

```
View[Vue3 + Element Plus] --> BFF[Axios + ]
BFF --> Ctrl[Controller]
Ctrl --> Service[Service]
Service --> Mapper[MyBatis-Plus]
Mapper --> DB[(MySQL)]
Service --> RecCore[Redis/]
RecCore --> Cache[(Redis/)]
subgraph Infra
    Security[JWT + Spring Security]
    Validation[ ]
    Logging[ ]
end
Ctrl --> Infra
```

5.

- JWT USER/LANDLORD/ADMIN
- / /
- / /
-
-
- 60% + 40%

“ / / ” frontend

5.1

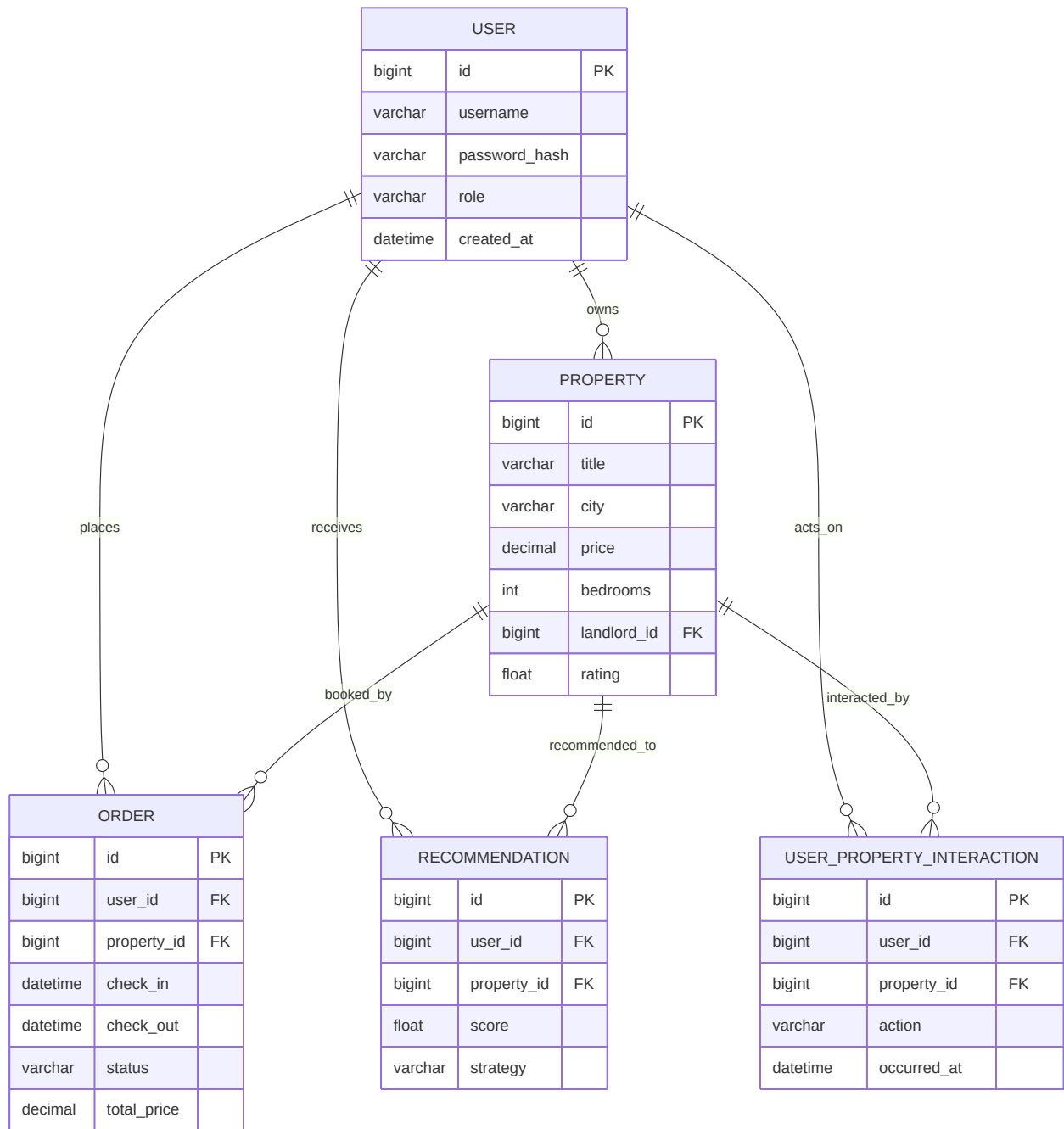
- USER /
- LANDLORD / /
- ADMIN

5.2

- Top-N
-
- 0.6 + 0.4 A/B

- 00000000000000000000/0000000000000000“000000 XX/00 YY”0

6. 000000ER 00



```

erDiagram
    USER {
        bigint id PK
        varchar username
        varchar password_hash
        varchar role
        datetime created_at
    }
    PROPERTY {
        bigint id PK
        varchar title
        varchar city
        decimal price
        int bedrooms
        bigint landlord_id FK
        float rating
    }
    ORDER {
        bigint id PK
        bigint user_id FK
        bigint property_id FK
        datetime check_in
        datetime check_out
        varchar status
        decimal total_price
    }
    RECOMMENDATION {
        bigint id PK
        bigint user_id FK
        bigint property_id FK
        float score
        varchar strategy
    }
    USER_PROPERTY_INTERACTION {
        bigint id PK
        bigint user_id FK
        bigint property_id FK
        varchar action
        datetime occurred_at
    }

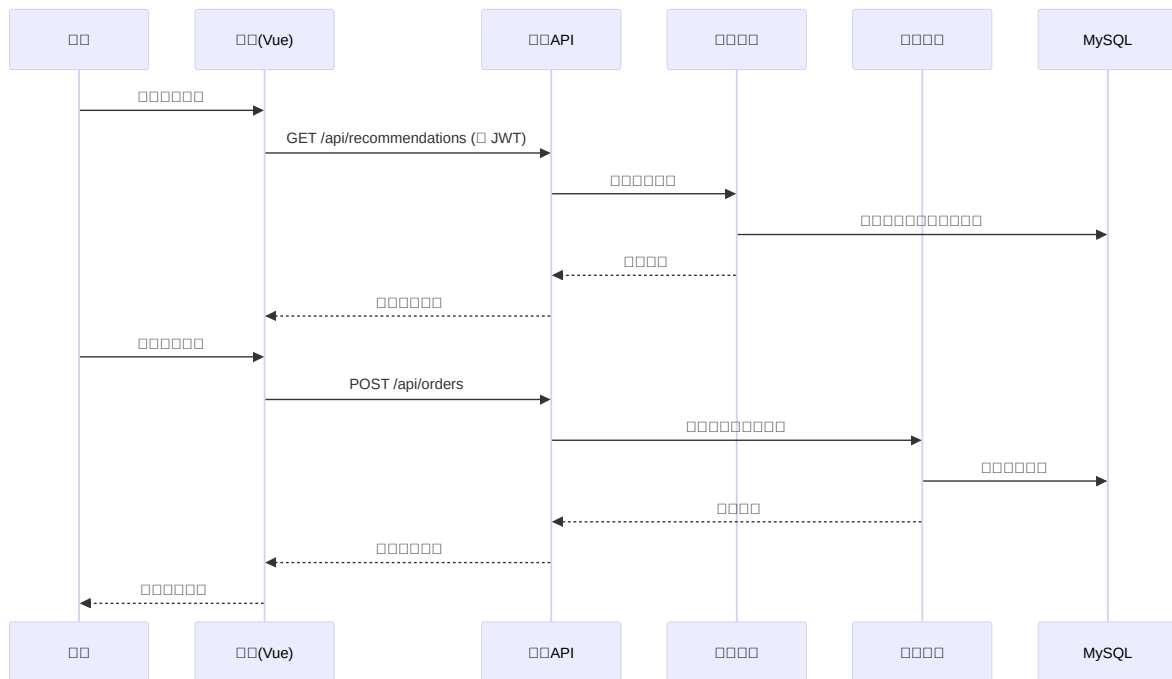
```

```

        bigint id PK
        varchar title
        varchar city
        decimal price
        int bedrooms
        bigint landlord_id FK
        float rating
    }
ORDER {
    bigint id PK
    bigint user_id FK
    bigint property_id FK
    datetime check_in
    datetime check_out
    varchar status
    decimal total_price
}
RECOMMENDATION {
    bigint id PK
    bigint user_id FK
    bigint property_id FK
    float score
    varchar strategy
}
USER_PROPERTY_INTERACTION {
    bigint id PK
    bigint user_id FK
    bigint property_id FK
    varchar action
    datetime occurred_at
}
USER ||--o{ ORDER : places
USER ||--o{ RECOMMENDATION : receives
USER ||--o{ USER_PROPERTY_INTERACTION : acts_on
PROPERTY ||--o{ ORDER : booked_by
PROPERTY ||--o{ RECOMMENDATION : recommended_to
PROPERTY ||--o{ USER_PROPERTY_INTERACTION : interacted_by
USER ||--o{ PROPERTY : owns

```

7. □□□□□□□□□□



“”

sequenceDiagram

```

participant U as 用户
participant FE as 前端(Vue)
participant API as 后端API
participant REC as 推荐服务
participant ORD as 订单服务
participant DB as MySQL
  
```

```

U->>FE: 消息
FE->>API: GET /api/recommendations (JWT)
API->>REC: 消息
REC->>DB: 消息
DB-->>REC: 消息
REC-->>API: 消息
API-->>FE: 消息
U->>FE: 消息
FE->>API: POST /api/orders
API->>ORD: 消息
ORD->>DB: 消息
DB-->>ORD: 消息
ORD-->>API: 消息
API-->>FE: 消息
FE-->>U: 消息
  
```

8. 部署

- `cd frontend && npm install && npm run build` 前端构建
- `mvn spring-boot:run` 后端启动
- MySQL 8.x 数据库配置 sql/ 脚本
- Redis 缓存配置

1. □□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□□□
3. □□ *A/B* □□□□□□□□□□□□□□
4. □□□□□□□□□□□□□□□□□□□□

- [1] Resnick P, Varian H R. Recommender systems. Communications of the ACM, 1997.
- [2] He X, et al. Neural Collaborative Filtering. WWW, 2017.
- [3] Sarwar B, et al. Item-based Collaborative Filtering Recommendation Algorithms. WWW, 2001.
- [4] 张俊明, 王云, 王云, 2016.
- [5] Kraska T. ML-based DBMS Design. SIGMOD, 2018.