



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。ER图如下所示：

项目采用“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。

## 1.2 数据库

- 数据库采用MySQL 8.0版本
- 数据库采用InnoDB引擎
- 项目ER图如下所示

## 1.3 部署

项目部署采用Docker容器化技术，部署环境如下：

## 2. 技术栈

- 前端：Vue 3、Vite、Element Plus、Pinia、Vue Router、Axios
- 后端：Spring Boot、MyBatis-Plus、JWT、Spring Validation、Lombok
- 数据库：MySQL、Redis
- 部署：Docker、Nginx

## 3. 部署

- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：

### 3.1 部署环境

- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：

### 3.2 部署步骤

- JWT令牌生成与验证
- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：
- 项目部署采用Docker容器化技术，部署环境如下：

### 3.3 部署命令



```
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

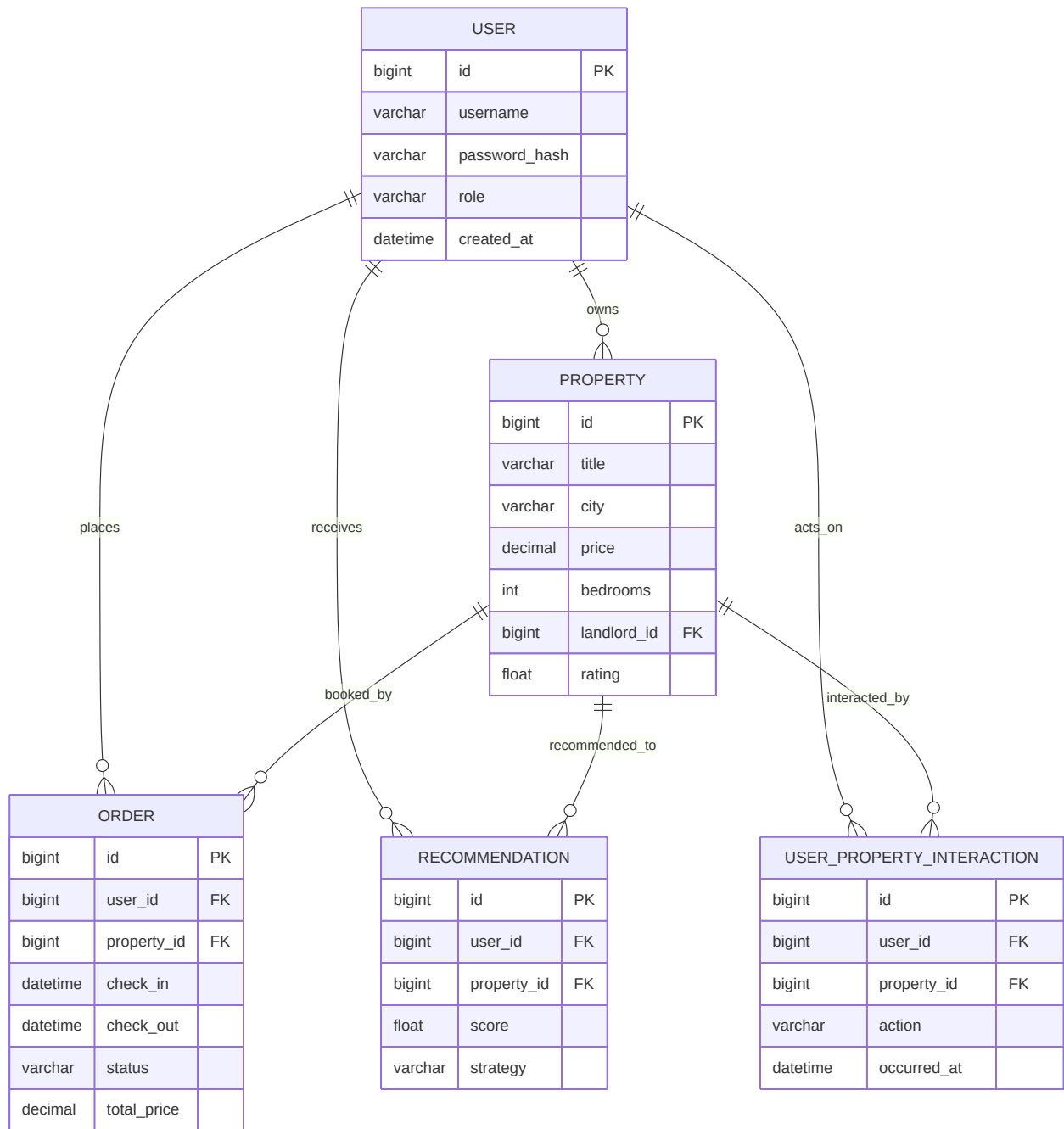
- USER /
- LANDLORD / /
- ADMIN

### 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 {

```

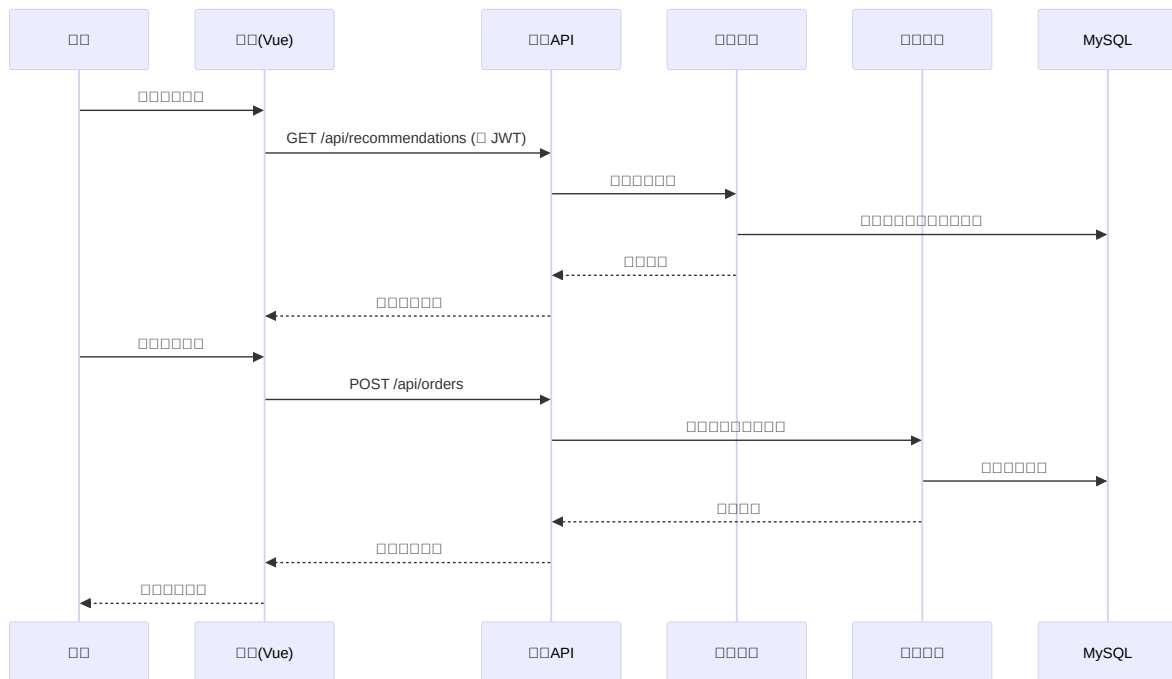
```

        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
}
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

- 在 application.yml 中配置 JWT 在 frontend/vite.config.js 中 API 接口

## 8.1 部署

- 部署 JWT + 在 Axios 中配置 Token 在 401 中
- 部署在 Docker/Compose 中
- 部署在 Kubernetes 中
- 部署在 AWS 中

## 8.2 数据库

- 数据库表结构 user\_id, property\_id, city, created\_at
- 数据库索引
- 数据库 ID/UUID
- 数据库连接池

## 8.3 部署

- 部署在 Docker/Compose 中
- 部署在 ELK/Prometheus+Grafana 中 QPS DB 部署
- 部署 MySQL 在 CDN 中
- 部署 HTTPS/JWT 在 API 中

## 9. 部署

- 部署在 Docker/Compose 中
- 部署在 Postman/Rest Client 中 API 部署
- 部署在 JMeter 中

## 9.1 部署

- 部署 Token 在 401 中
- 部署在 Docker/Compose 中
- 部署在 Kubernetes 中
- 部署在 AWS 中
- 部署 Token 在 401 和 403 中

## 9.2 部署

- 部署 JUnit/MockMvc 在 Vitest 中
- 部署 Postman/Newman 在 Rest Client 中
- 部署 JMeter/Locust 在 95/99 中
- 部署在 Docker/Compose 中

## 10. 部署



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] 张明. 深度学习. 机械工业出版社, 2016.
- [5] Kraska T. ML-based DBMS Design. SIGMOD, 2018.