



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，使用Nginx作为反向代理服务器。

项目采用模块化设计，将业务逻辑拆分为多个模块，每个模块负责特定的业务功能。项目使用ER图进行数据库设计，确保数据结构的合理性和完整性。

项目采用“接口+实现”的模式，前端通过API接口与后端进行数据交互。后端通过接口定义和实现，确保前后端的数据交互清晰且易于维护。

## 1.2 数据库

- 数据库采用MySQL，使用MyBatis-Plus进行数据库操作。
- 数据库设计遵循ER图，确保数据结构的合理性和完整性。
- 项目使用ER图进行数据库设计，确保数据结构的合理性和完整性。

## 1.3 部署

项目部署采用Docker容器化技术，使用Nginx作为反向代理服务器。

---

## 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. 项目结构

- 项目结构采用模块化设计，将业务逻辑拆分为多个模块。
- 项目使用ER图进行数据库设计，确保数据结构的合理性和完整性。
- 项目采用“接口+实现”的模式，前端通过API接口与后端进行数据交互。
- 项目使用JWT进行身份认证，确保系统的安全性。

### 3.1 数据库设计

- 数据库采用MySQL，使用MyBatis-Plus进行数据库操作。
- 数据库设计遵循ER图，确保数据结构的合理性和完整性。
- 项目使用ER图进行数据库设计，确保数据结构的合理性和完整性。

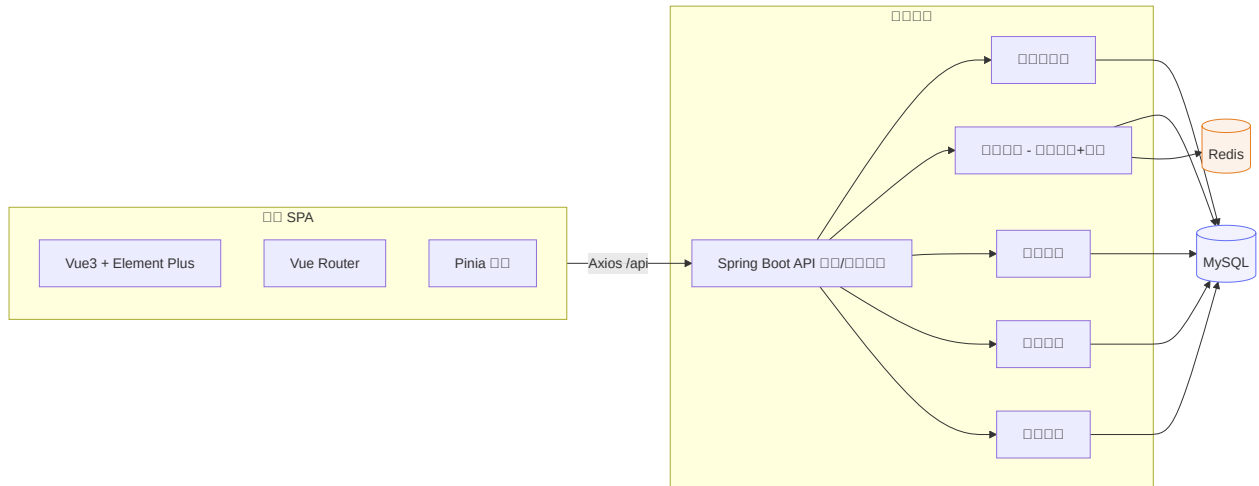
### 3.2 接口设计

- JWT身份认证Token
- 接口定义和实现，确保前后端的数据交互清晰且易于维护。
- 接口采用“接口+实现”的模式，前端通过API接口与后端进行数据交互。
- 接口采用“接口+实现”的模式，前端通过API接口与后端进行数据交互。

### 3.3 部署

- 前后端分离/前后端分离前后端分离前后端分离前后端分离前后端分离
- 前后端分离前后端分离前后端分离前后端分离前后端分离前后端分离
- 前后端分离前后端分离前后端分离前后端分离前后端分离前后端分离 token
- 前后端分离前后端分离 HTTPS 前后端Token 前后端分离前后端分离前后端分离

#### 4. 前后端



##### 4.1 前后端

flowchart LR

```

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

subgraph Backend[Backend]
    APIGW[Spring Boot API]
    Auth[Auth]
    Rec[Rec - 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 架构图

flowchart TB

```

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

```

## 5. 需求

- 系统支持JWT鉴权，支持USER/LANDLORD/ADMIN三种角色
  - 系统支持多租户，租户信息存储在数据库中
  - 系统支持多语言，支持中文/英文/日语
  - 系统支持多平台，支持Web/Android/iOS
  - 系统支持多设备，支持PC/手机/平板
  - 系统支持多版本，支持60% 旧版 + 40% 新版
- 系统支持多语言，支持中文/英文/日语，支持多平台，支持Web/Android/iOS，支持多设备，支持PC/手机/平板，支持多版本，支持60% 旧版 + 40% 新版。

### 5.1 用户管理

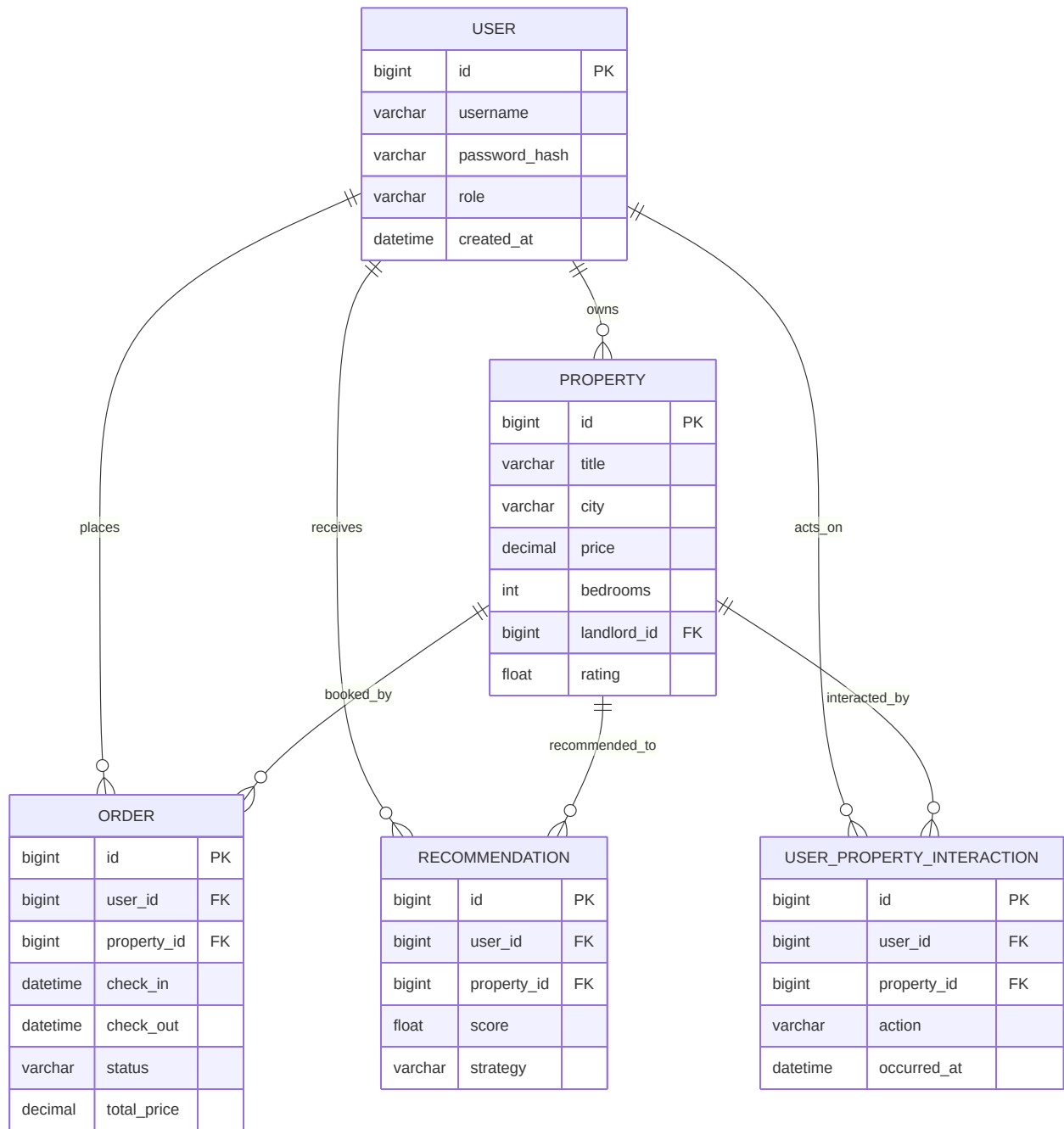
- 支持USER角色，支持多语言，支持多平台，支持多设备，支持多版本
- 支持LANDLORD角色，支持多语言，支持多平台，支持多设备，支持多版本
- 支持ADMIN角色，支持多语言，支持多平台，支持多设备，支持多版本

### 5.2 系统管理

- 支持多语言，支持多平台，支持多设备，支持多版本，支持Top-N
- 支持多语言，支持多平台，支持多设备，支持多版本，支持A/B
- 支持多语言，支持多平台，支持多设备，支持多版本，支持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 {

```

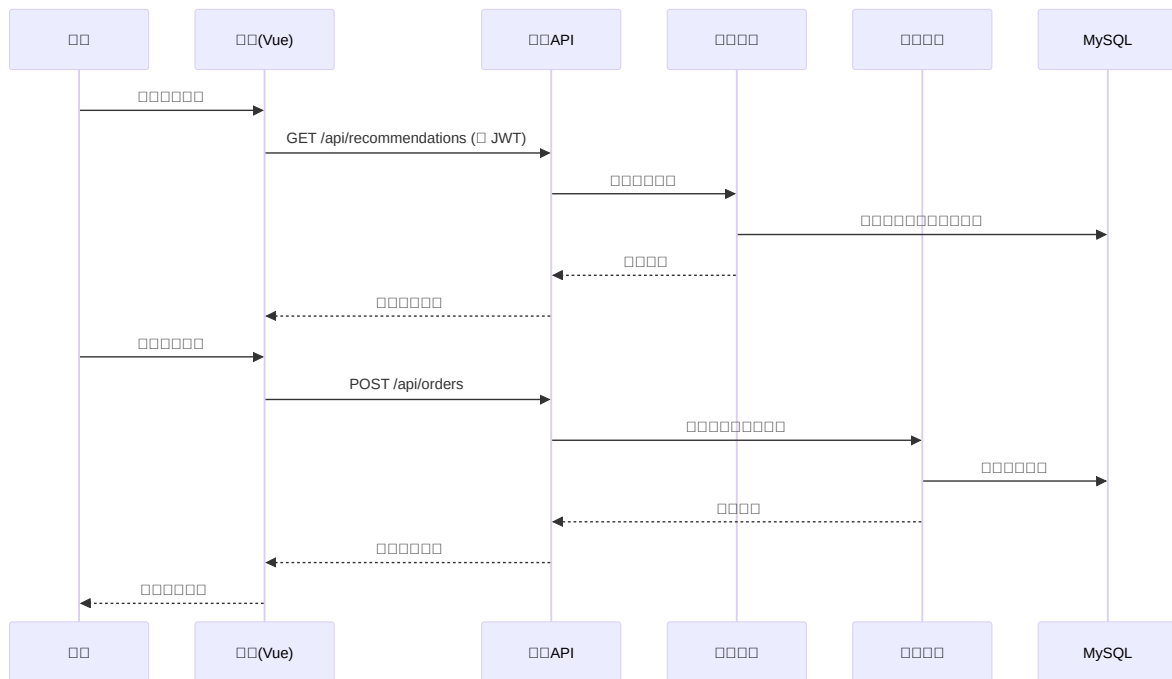
```

        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
- Redis 缓存

- 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
- 在 /api/ 中配置 Token 在 401 中
- 在 /api/ 中配置 Token 在 401 中
- 在 /api/ 中配置 Token 在 401 中

## 8.2 部署

- 在 user\_id, property\_id, city, created\_at 中
- 在 ID/UUID 中
- 在 ID/UUID 中
- 在 ID/UUID 中

## 8.3 部署

- 在 ELK, Prometheus+Grafana, QPS, DB 中
- 在 MySQL, CDN, HTTPS, JWT 中

## 9. 部署

- 在 Postman/Rest Client 中 API
- 在 JMeter 中

## 9.1 部署

- 在 Token 中 401, 403

## 9.2 部署

- 在 JUnit/MockMvc, Vitest 中
- 在 Postman/Newman, Rest Client 中
- 在 JMeter/Locust 中 95/99
- 在 95/99 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.