

A horizontal row of twelve empty rectangular boxes, intended for children to draw or write in.

□□□XXX□□□□□□□

□□□XXX□□ □□□XXX □□□XXXXXXXXX

□□□□□XXX □□□XXX

□□□□□ 2026 □ X □ X □

1

“” Vue 3 + Vite + Element Plus Spring Boot MyBatis-Plus /

Abstract

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.

Keywords: Homestay recommendation; Personalized recommendation; Spring Boot; Vue 3; Hybrid recommender

2

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

1.

1.1 □□□□

“**前后端分离**”是近年来最火的架构模式之一，它将传统的“**前后端一体**”拆分为前后端两个独立的模块。

前后端分离的实现方式多种多样，其中最常见的是通过**前后端分离**（**前后端分离**）来实现。这种方式的优点在于前后端分离，可以分别进行开发和部署，从而提高了系统的可维护性和可扩展性。

“**前后端分离**”的实现方式多种多样，其中最常见的是通过**前后端分离**（**前后端分离**）来实现。这种方式的优点在于前后端分离，可以分别进行开发和部署，从而提高了系统的可维护性和可扩展性。

1.2 前端

- 前端框架：React、Vue、Angular
- 前端库：Bootstrap、Ant Design、Element UI
- 前端工具：Webpack、Vite、Babel

1.3 后端

后端框架：Spring Boot、Django、Rails、Node.js、MongoDB、MySQL、Redis

2. 架构设计

- 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

3. 安全性

- 前端反向代理：Nginx、Apache
- 后端反向代理：Apache、Nginx
- 前端静态资源压缩：Gzip、Brotli
- 后端身份验证：JWT

3.1 身份验证

- 前端登录表单：用户名、密码
- 后端API接口：登录、注册
- 前端会话管理：Cookie、LocalStorage

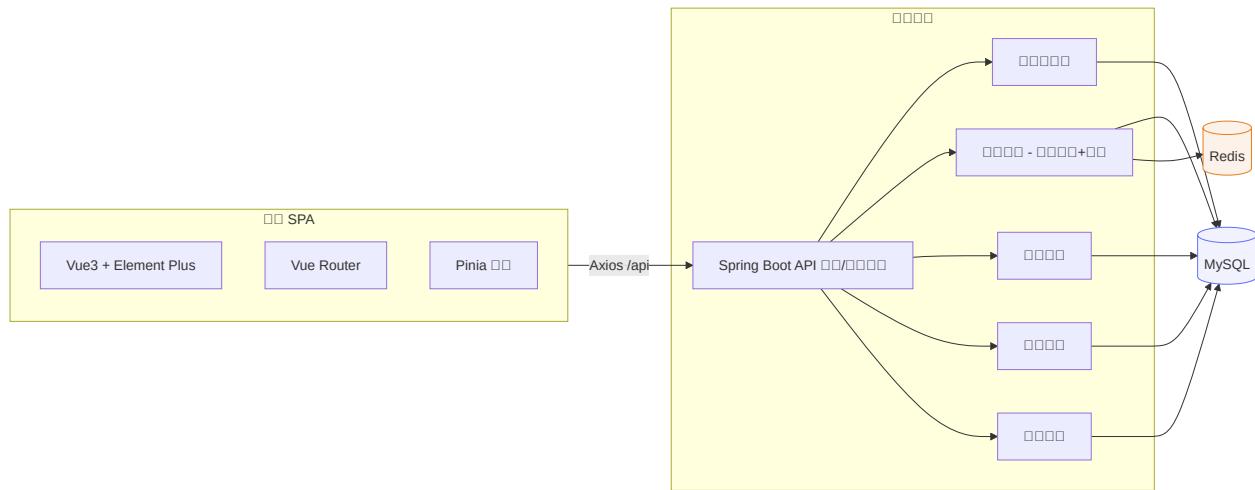
3.2 权限控制

- JWT Token
- 前端路由拦截：只有登录成功才能访问某些路由
- 后端API权限校验：根据用户角色分配不同的权限
- 前端权限管理：根据用户角色显示不同的菜单

3.3 日志记录

- **HTTP** / **HTTPS** **Cookie**
 - **HTTP** / **HTTPS** **Header** **token**
 - **HTTP** / **HTTPS** **Token** **JWT**
-

4. **架构图**



4.1 **架构图**

```

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

    subgraph Backend [Backend]
        APIGW[Spring Boot API GW]
        Auth[Auth]
        Rec[Rec - Rec+Order]
        Order[Order]
        Property[Property]
        UserSvc[User Svc]
    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 算法实现

```

flowchart TB
    View[前端<br/>Vue3 + Element Plus] --> BFF[后端<br/>Axios + Express]
    BFF --> Ctrl[Controller]
    Ctrl --> Service[Service]
    Service --> Mapper[Mapper MyBatis-Plus]
    Mapper --> DB[(MySQL)]
    Service --> RecCore[核心<br/>逻辑/数据库]
    RecCore --> Cache[(Redis/Memcached)]
    subgraph Infra[基础设施]
        Security[JWT + Spring Security]
        Validation[Validation]
        Logging[Logging]
    end
    Ctrl --> Infra

```

5. 算法设计

- 使用 JWT 进行身份验证，支持 USER/LANDLORD/ADMIN 角色
 - 支持租户隔离，不同租户的数据不会混杂
 - 支持多租户模式 / 租户 / 多租户管理
 - 支持租户隔离，不同租户的数据不会混杂
 - 支持租户隔离，不同租户的数据不会混杂
 - 支持租户隔离，不同租户的数据不会混杂
- 同时支持租户隔离，“租户/租户/租户”级别的 **frontend** 支持租户隔离，同时

5.1 用户角色

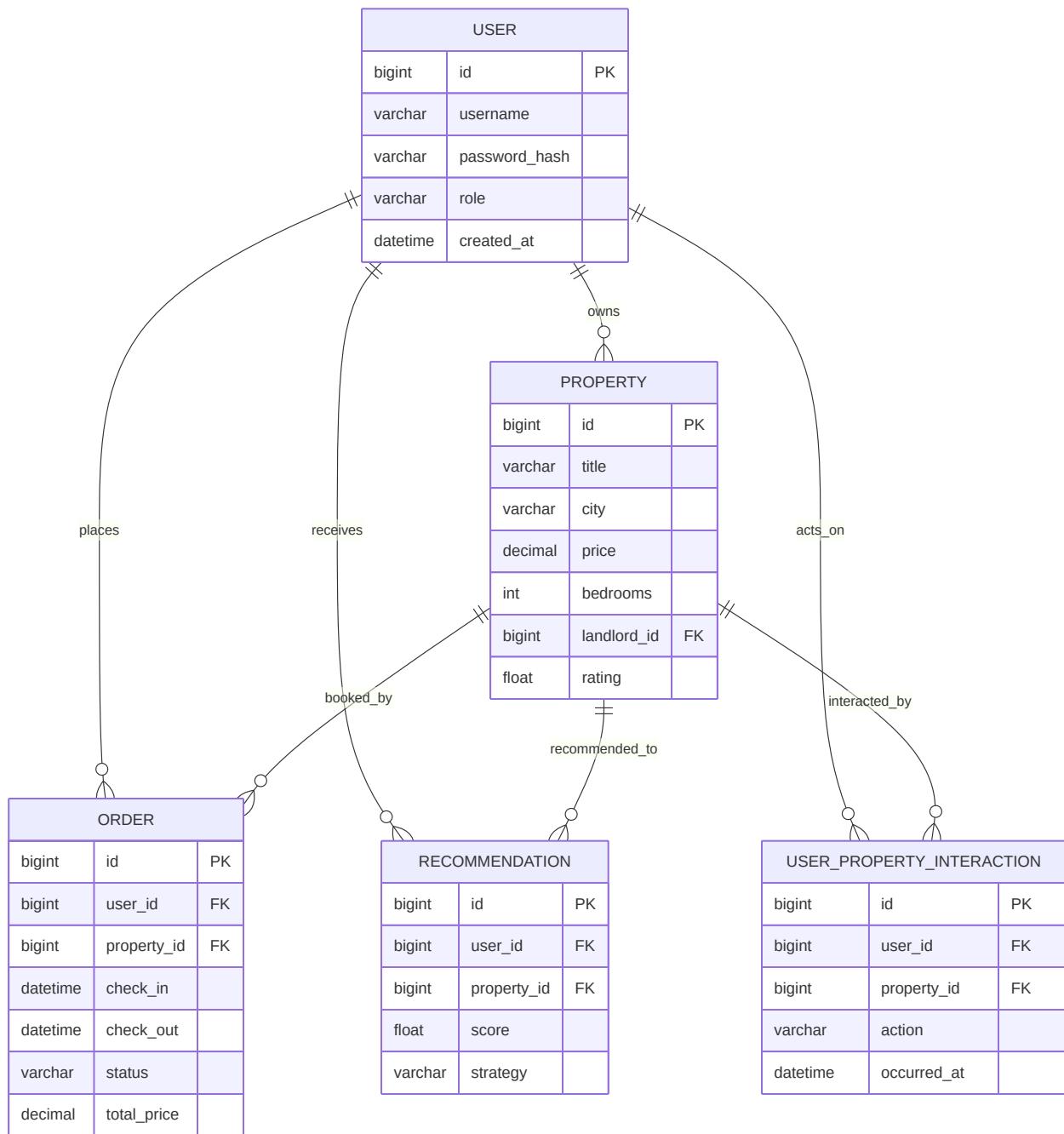
- 支持 **USER** 角色 / 租户 / 多租户管理
- 支持 **LANDLORD** 角色 / 租户 / 多租户管理
- 支持 **ADMIN** 角色 / 租户 / 多租户管理

5.2 算法设计

- 支持租户隔离，同时支持租户隔离 - 租户隔离的 Top-N 算法
- 支持租户隔离，同时支持租户隔离 - 租户隔离的 Top-N 算法
- 支持租户隔离，同时支持租户隔离 - 租户隔离的 Top-N 算法

- 『XX年YY月XX日XX时XX分XX秒』“XX/XX YY”

6. ER 图



erDiagram

```

USER {
    bigint id PK
    varchar username
    varchar password_hash
    varchar role
    datetime created_at
}

PROPERTY {

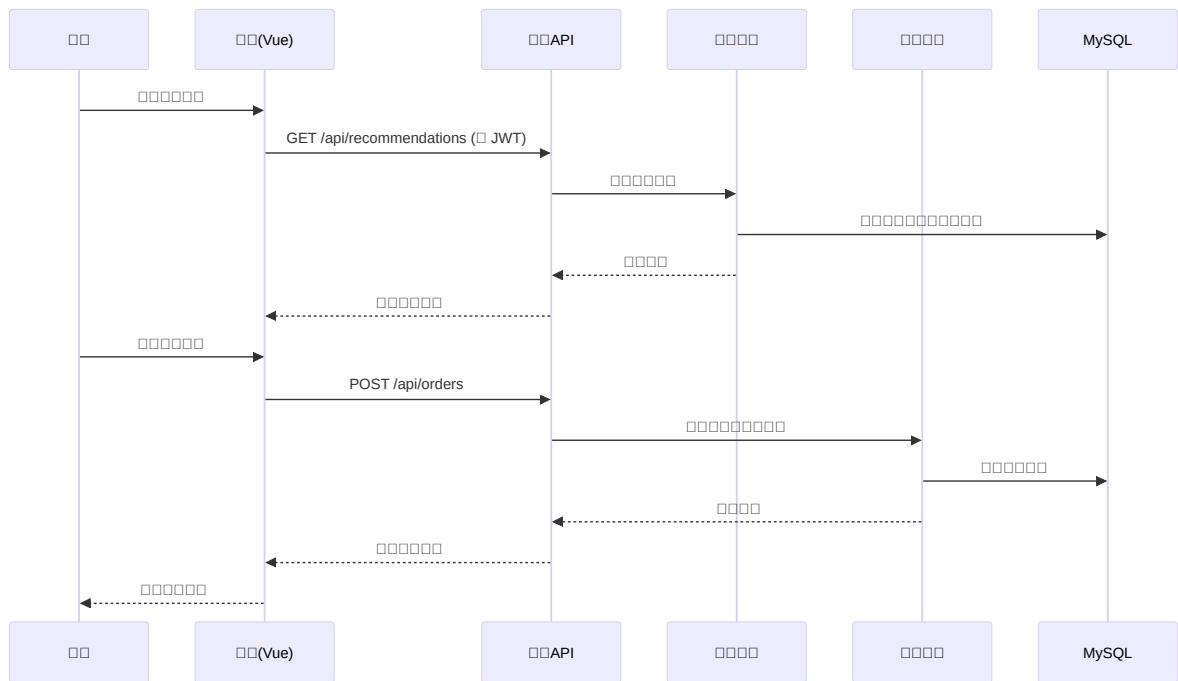
```

```

        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

```

sequenceDiagram
    participant U as 用户
    participant FE as 用户(Vue)
    participant API as API
    participant REC as 推荐
    participant ORD as 订单
    participant DB as MySQL
    
```

```

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

8. 前端部署

- cd frontend && npm install && npm run build
- mvn spring-boot:run
- MySQL 8.x sql/ Redis

- `application.yml` `JWT` `frontend/vite.config.js` `API`

8.1

- **JWT + Axios Token 401**
 - **JWT + Axios Token 401**
 - **JWT + Axios Token 401**
 - **JWT + Axios Token 401**

8.2

- `user_id`/`property_id`/`city`/`created_at`
 - `user_id`/`property_id`/`city`
 - `user_id`/`property_id` **ID/UUID**
 - `user_id`/`property_id`

8.3

9.

- [Postman](#) / [API](#)
 - [Postman/Rest Client](#) [API](#)
 - [JMeter](#) [API](#)

9.1

- **Token**
 - **Token**
 - **Token**
 - **Token**
 - **Token**

9.2

- JUnit/MockMvc Vitest
 - Postman/Newman Rest Client
 - JMeter/Locust 95/99
 -

10. □ □ □ □ □

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

11. □ □ □ □

- [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] 田晓峰. 机器学习. 清华大学出版社, 2016.
 - [5] Kraska T. ML-based DBMS Design. SIGMOD, 2018.

12. □□