

XXXXXXXXXXXX

XXXXXXXXXXXX
XXX XXX XXXXXXXX
XXXXXXX XXXXX
XXXX2026 X X X

□□

“”Vue 3 + Vite + Element PlusSpring BootMyBatis-Plus /
Spring BootVue 3

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

□□

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

1. 项目背景

1.1 项目概述

本项目旨在开发一个基于微服务架构的分布式系统，用于处理大规模并发请求，并支持高可用性和可扩展性。系统采用Spring Cloud生态进行构建，并结合Redis进行缓存优化，以提升整体性能。

系统架构采用微服务化设计，通过Spring Cloud Gateway进行流量管理，并结合Eureka进行服务注册与发现。数据库采用MySQL，并结合Redis进行分布式缓存，以提高数据访问效率。

系统支持分布式部署，具备良好的横向扩展能力。通过Kubernetes进行容器化部署，并结合Docker进行环境管理，确保系统在不同环境下的稳定性和一致性。

1.2 技术选型

- 后端框架：Spring Boot 2.7.0
- 数据库：MySQL 8.0.28
- 缓存：Redis 6.0.11
- 消息队列：Kafka 2.8.0

1.3 部署环境

系统部署在CentOS 7.9.2009服务器上，使用Docker容器化技术进行部署。

2. 系统架构设计

- 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 + Kafka
- 部署：Docker + Kubernetes

3. 系统功能设计

- 用户认证与授权：支持OAuth2.0协议，实现用户登录、注册、密码重置等功能。
- 数据管理：提供RESTful API接口，支持数据的增删改查操作。
- 系统监控：集成Prometheus和Grafana，实现系统运行状态的实时监控。
- 日志管理：使用ELK（Elasticsearch, Logstash, Kibana）进行日志收集、分析和可视化。

3.1 数据库设计

- 用户表：存储用户基本信息，包括用户名、密码、手机号等。
- 权限表：存储系统权限信息，包括角色、权限名称、权限描述等。
- 操作日志表：记录用户操作日志，包括操作时间、操作人、操作内容等。

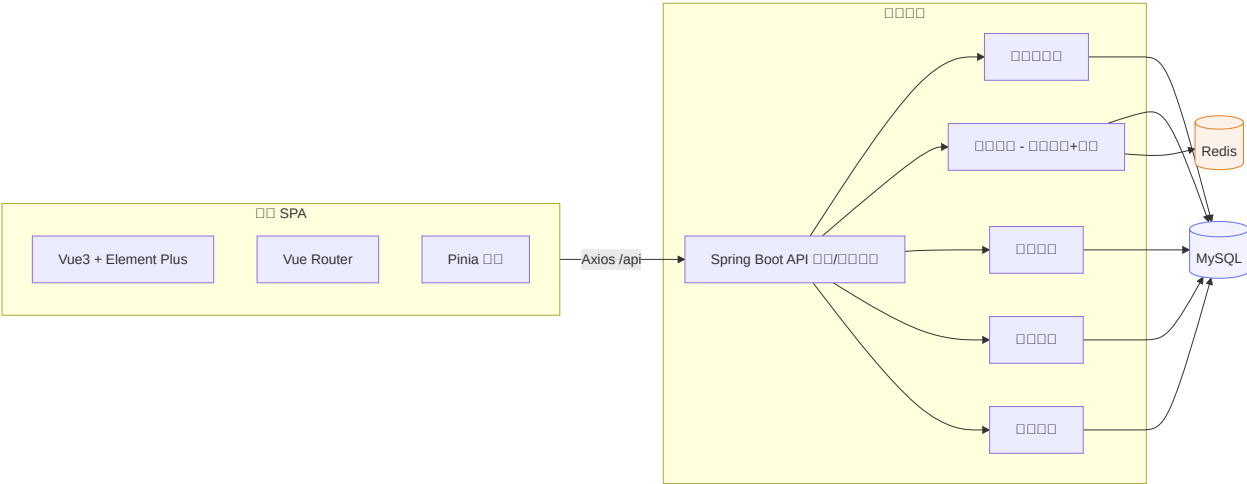
3.2 数据库

- JWT 令牌认证 Token
- 数据库设计/数据库表设计/数据库表结构
- 数据库设计/数据库表设计/数据库表结构
- 数据库设计/数据库表设计/数据库表结构

3.3 数据库

- 数据库设计/数据库表设计/数据库表结构
- 数据库设计/数据库表设计/数据库表结构
- 数据库设计/数据库表设计/数据库表结构 token
- 数据库设计/数据库表设计/数据库表结构 HTTPS Token 数据库设计/数据库表设计/数据库表结构

4. 数据库



4.1 数据库

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 - Redis]
    Order[Order]
    Property[Property]
    UserSvc[UserSvc]
end
```

```

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[Redis/]
RecCore --> Cache[(Redis/)]
subgraph Infra
    Security[JWT + Spring Security]
    Validation[ ]
    Logging[ ]
end
Ctrl --> Infra

```

5. 接口

- JWT 用户/LANDLORD/ADMIN
- / /
- / /
-

- [illegible]

□□□□□□□□□□□□□□“□□□□/□□□□/□□□□”□□□□ frontend □□□□□□□□□□□□□□□□
□□□□□□

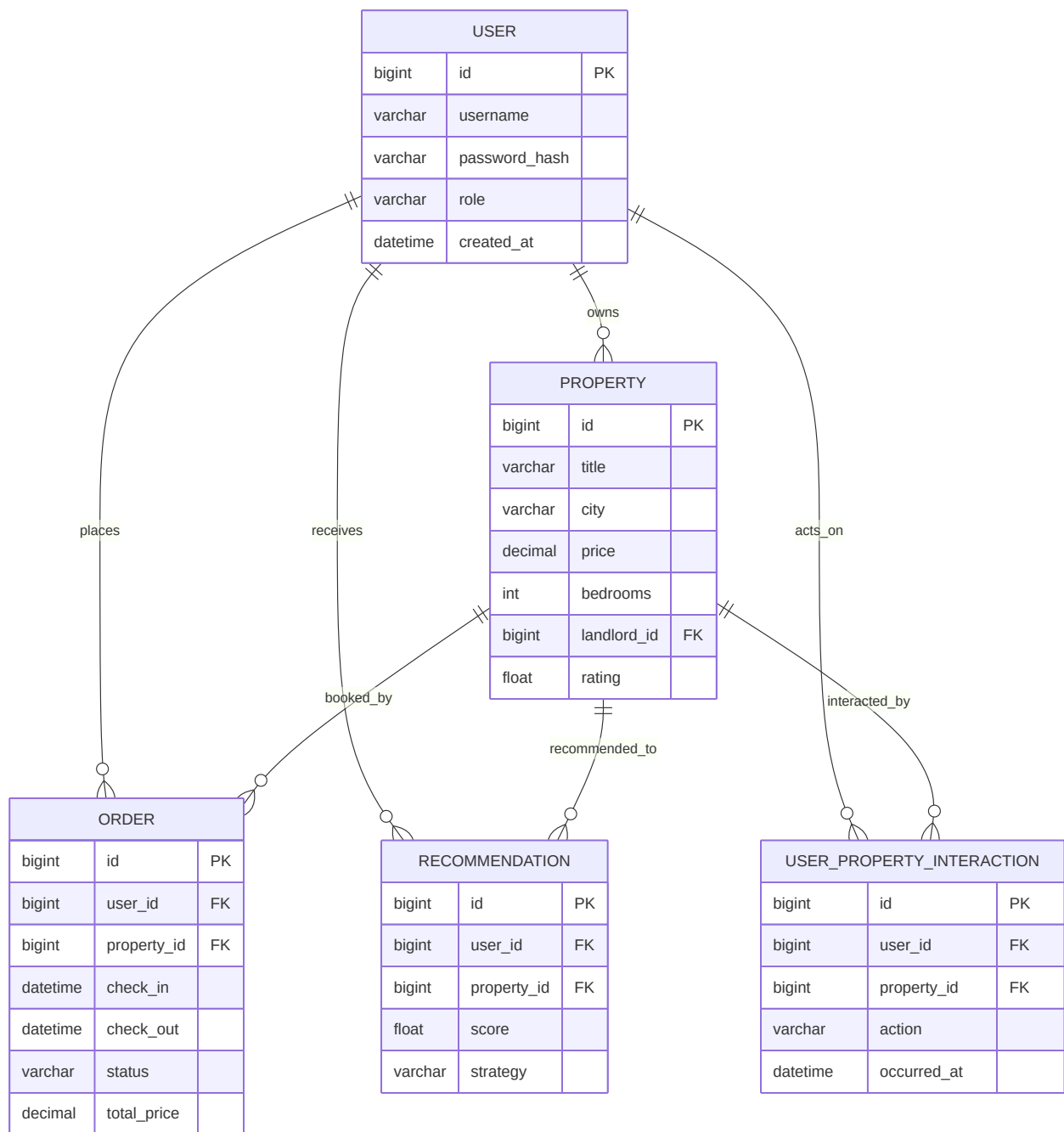
5.1

- **USER** /
- **LANDLORD** / /
- **ADMIN**

5.2 5.2.2.2.2.2.2

- [illegible]

6. **ER**



```
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

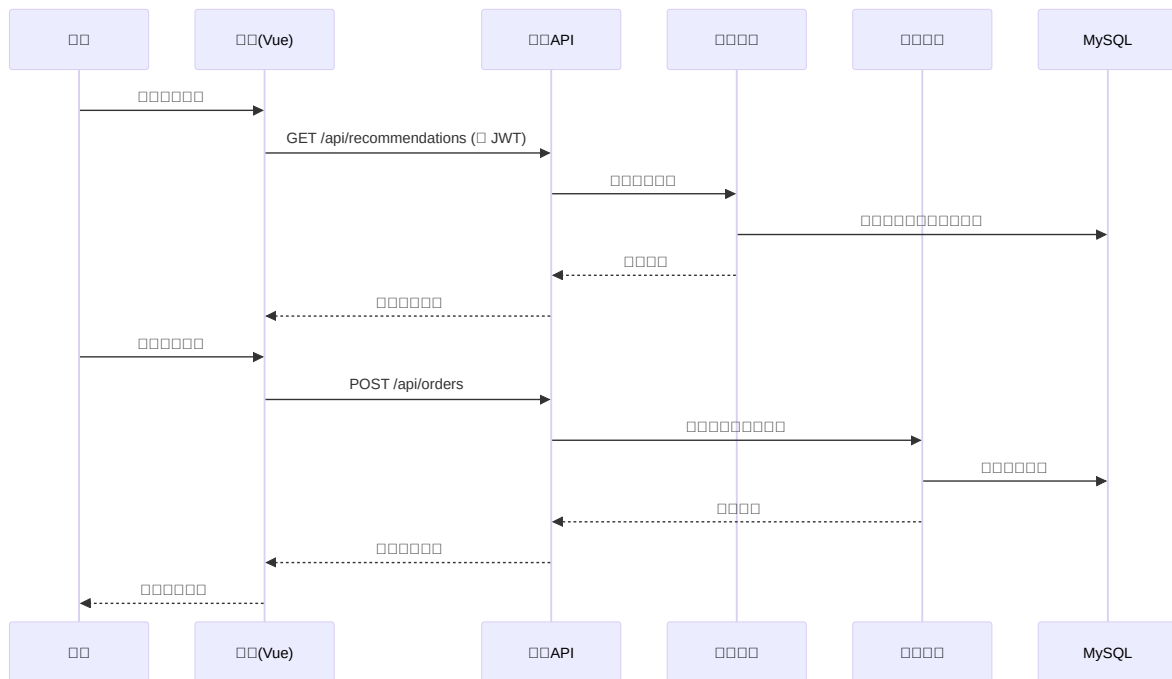
```

```

        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 □□
 participant FE as □□(Vue)
 participant API as □□API
 participant REC as □□□□
 participant ORD as □□□□
 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. □□□□□□□□

- □□□cd frontend && npm install && npm run build□□□□□ frontend/dist□

- `mvn spring-boot:run` `java -jar target/*.jar`
- MySQL 8.x `sql/` Redis
- `application.yml` JWT `frontend/vite.config.js` API

8.1

- JWT + axios/Axios Token 401
- /
-
-
-

8.2 □□□□□□

- user_id, property_id, city, created_at
- ID/UUID
- ID/UUID
- /

8.3 □□□□□□

- 负载均衡/CDN/数据库/缓存/消息队列/容器/云原生
- 监控系统 ELK/Prometheus+Grafana/日志系统 QPS/DB 性能优化
- 数据库 MySQL 优化+分布式数据库/缓存 CDN/CDN
- 安全 HTTPS/JWT 鉴权/身份认证/权限控制

[illegible]

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

9.1 ☐☐☐☐☐☐

- `00/0000000000000000Token 000000000`
- `0000000000000000/0000000000000000`
- `000000000000000000000000000000000000`
- `0000000000000000000000000000/0000000000000000`
- `00000 Token 000000000 4010000000 4030000000000000`

9.2 □□□□□□

- JUnit/MockMvc Vitest

- 使用 Postman/Newman 或 Rest Client 进行测试
- 使用 JMeter/Locust 进行测试 95/99 通过率
- 使用其他测试工具进行测试

10. 测试

测试用例设计

1. 测试用例设计
2. 测试用例设计
3. 测试 A/B 测试
4. 测试用例设计

11. 参考文献

- [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] 张三. 测试. 测试, 2016.
- [5] Kraska T. ML-based DBMS Design. SIGMOD, 2018.

12. 附录

附录 A

附录 B