P2P Trading System – Backend

1. Mục tiêu dự án

Nền tảng backend hỗ trợ giao dịch tài sản số theo mô hình P2P. Hệ thống cung cấp API cho:

- Quản lý quảng cáo mua/bán (order) và khớp lệnh (trade) theo thời gian thực.
- Tra cứu thị trường P2P Binance để tham chiếu giá.
- Xử lý ví người dùng, chat giao dịch, tranh chấp và đồng bộ với hệ thống ngoài.
- Bảo vệ tài nguyên bằng JWT, phân quyền request và ghi nhận lịch sử hoạt động.

2. Công nghệ & thư viện chính

Nhóm	Công nghệ
Ngôn ngữ	Java 17
Framework	Spring Boot 3.1 (Web, Validation)
Persist	Spring Data JPA (Hibernate), Flyway
Bảo mật	Spring Security, JWT (jjwt 0.11.x)
CSDL	PostgreSQL
Tài liệu	springdoc-openapi 2.2 + Swagger UI
Build	Maven multi-module, JDK toolchain
Hỗ trợ khác	GZIP handling, RestTemplate, MapStruct/Lombok (có thể bổ sung)

3. Cấu trúc module Maven

```
├── pom.xml                 # BOM cấp cao, khai báo modules
├── p2p_common/             # Hằng số, exception, tiện ích dùng chung
├── p2p_repository/         # Entity, repository, migration (Flyway)
├── p2p_service/            # Domain service, use-case, Command/Result
├── p2p_security/           # Filter JWT, SecurityConfig, SecretKey bean
└── p2p_p2p/                # Ứng dụng Spring Boot expose REST API
```

Luồng logic cơ bản

1. Request tới `p2p_p2p` → controller nhận payload → mapper chuyển sang `*Command`.
2. `p2p_service` xử lý nghiệp vụ (gọi repository, tích hợp Binance, lock wallet...).
3. Service trả về `*Result` → controller map thành response DTO.
4. `p2p_security` kiểm tra JWT trước khi vào controller (ngoại trừ endpoint public).

4. Tích hợp nổi bật

- **Binance P2P**: `BinanceP2PMarketService` gọi API Binance, giải nén gzip, chuẩn hóa JSON.
- **Integration API**: `/api/integration/users/sync` đồng bộ user + wallet từ hệ thống ngoài và trả JWT.
- **Flyway**: bật mặc định (`spring.flyway.enabled=true`), migration đặt tại `classpath:db/migration`.
- **Swagger UI**: truy cập `http://localhost:8080/api/swagger-ui/index.html`.

5. Thiết lập môi trường

Yêu cầu

- JDK 17+
- Maven 3.8+
- PostgreSQL 14+ (tạo database `p2p_trading`)

Cấu hình môi trường

Ứng dụng dùng Spring Profiles và mặc định kích hoạt profile `local` (có thể override qua biến `SPRING_PROFILES_ACTIVE`).

- `application-local.properties`: phục vụ phát triển trên máy cá nhân, có giá trị mặc định an toàn để khởi động nhanh.
- `application-dev.properties`: dành cho môi trường dev/staging, đọc thông số từ biến môi trường (`DB_URL`, `DB_USERNAME`, `DB_PASSWORD`, `JWT_SECRET`...).
- `application-prod.properties`: cấu hình production, siết chặt Flyway và tắt log SQL.

Ví dụ export biến cho local:

```
export SPRING_PROFILES_ACTIVE=local
export DB_URL=jdbc:postgresql://localhost:5432/p2p_trading_dev
export DB_USERNAME=postgres
export DB_PASSWORD=123
export JWT_SECRET=mysupersecuresecretkey_which_is_at_least_32_chars
```

> **Lưu ý:** giữ `spring.jpa.hibernate.ddl-auto=none` để tránh xung đột với Flyway.

6. Cách chạy & luồng hoạt động

```
# Đóng gói toàn bộ modules
mvn clean install

# Khởi động ứng dụng (module p2p_p2p)
mvn -pl p2p_p2p -am spring-boot:run

# Hoặc chạy từ jar đã build
java -jar p2p_p2p/target/p2p_p2p-1.0-SNAPSHOT.jar
```

Ứng dụng lắng nghe tại `http://localhost:8080/api`.

Luồng khởi động nội bộ

1. Spring Boot nạp cấu hình datasource, JWT, servlet path `/api`.
2. `p2p_security` khởi tạo `SecretKey`, cấu hình `SecurityFilterChain`, `JwtAuthenticationFilter`.
3. `p2p_repository` quét entity, chạy Flyway migration, dựng `EntityManagerFactory`.
4. `p2p_service` khởi tạo service/use-case, bao gồm luồng giao dịch, ví, tranh chấp, Binance.
5. `p2p_p2p` publish REST controller; springdoc sinh OpenAPI → truy cập `/api/swagger-ui/index.html`.

Luồng xử lý request tiêu chuẩn

1. Client gửi request (ví dụ `POST /api/p2p/orders`) với JWT.
2. Filter JWT xác thực, inject principal vào `SecurityContext`.
3. Controller nhận payload → mapper chuyển thành `*Command`.
4. Service thực thi transaction: lock/bỏ lock wallet, cập nhật order, gọi repository hoặc Binance.
5. Service trả `*Result` → controller map sang response DTO và trả về `ResponseEntity`.

Endpoint nổi bật

- `/market/price` – giá tham chiếu Binance (BinanceP2PMarketService).
- `/p2p/orders` – quản lý order; `/p2p/orders/{orderId}/trades` – danh sách trade.
- `/p2p/trades` – tạo trade; `/p2p/trades/{id}/confirm-payment`, `/confirm-received`, `/cancel` – workflow thanh toán.
- `/p2p/trades/{id}/chat` – chat buyer/seller.
- `/integration/users/sync` – đồng bộ user & wallet từ hệ thống ngoài và phát hành JWT.

7. Quy ước code & kiến trúc

- **Command / Result / Response**: phân tầng rõ ràng, tránh lẫn lộn DTO giữa controller và service.
- **Transaction boundary**: `@Transactional` tại service đảm bảo atomicity (create trade, dispute...).
- **Locking**: dùng `PESSIMISTIC_WRITE` khi lấy order, `SellerFundsManager` lock/unlock `availableBalance` của ví.
- **Validation**: `jakarta.validation` trên payload, custom exception (`ApplicationException`) với `ErrorCode`.
- **Logging & Monitoring**: nên bật Spring Boot logging, xem xét bổ sung Actuator cho prod.

8. Kiểm thử & mở rộng

- Viết test tại `src/test/java` cho từng module; có thể dùng Testcontainers cho PostgreSQL.
- Container hóa bằng Docker (viết Dockerfile cho module `p2p_p2p`, mount file cấu hình).
- Theo dõi rate-limit Binance, cân nhắc caching/queue để giảm số lần gọi.

9. Lộ trình phát triển

- Hoàn thiện workflow tranh chấp (notify, phân quyền xử lý).
- Tích hợp notification service (email/websocket) cho trạng thái trade.

- Bổ sung audit log / lịch sử giao dịch.
- Tối ưu hiệu năng truy vấn (pagination, caching order/trade).

Liên hệ: đội ngũ Akabazan Backend.