OnRamp Integration System

Hệ thống tích hợp dịch vụ On-ramp được thiết kế để dễ dàng mở rộng và bảo trì, sử dụng các design patterns hiện đại và Spring Boot framework.

Tổng quan

OnRamp Integration System là một giải pháp toàn diện cho việc tích hợp các dịch vụ fiat-to-crypto (On-ramp) vào sàn giao dịch tiền điện tử. Hệ thống được thiết kế với kiến trúc modular, cho phép dễ dàng thêm mới các nhà cung cấp dịch vụ mà không cần thay đổi code hiện có.

Đặc điểm chính

- **Kiến trúc Abstract**: Sử dụng Factory Pattern và Strategy Pattern để tách biệt logic nghiệp vụ khỏi implementation cụ thể
- **Dễ mở rộng**: Thêm nhà cung cấp mới chỉ cần implement interface có sẵn
- **Type Safety**: Sử dụng Java với Lombok để giảm boilerplate code
- **Async Processing**: Hỗ trợ xử lý bất đồng bộ với CompletableFuture
- **Comprehensive Testing**: Unit tests đầy đủ với Mockito
- **Spring Boot Integration**: Tận dụng dependency injection và auto-configuration

Nhà cung cấp được hỗ trợ

- **Onramper**: Aggregator hàng đầu với 20+ nhà cung cấp on-ramp, 130+ phương thức thanh toán
- **Extensible**: Dễ dàng thêm MoonPay, Banxa, Ramp Network, và các nhà cung cấp khác

Kiến trúc hệ thống

Design Patterns được sử dụng

1. **Factory Pattern**: `OnRampServiceFactory` để tạo instances của các service
2. **Strategy Pattern**: Mỗi nhà cung cấp implement `OnRampService` interface
3. **Adapter Pattern**: Chuyển đổi API responses thành domain models
4. **Builder Pattern**: Sử dụng Lombok `@Builder` cho data models
5. **Dependency Injection**: Spring Boot để quản lý dependencies

Cấu trúc thư mục

Plain Text

```
src/
├── main/java/com/onramp/integration/
│   ├── core/                    # Core interfaces và abstractions
│   ├── models/                  # Domain models và DTOs
│   ├── factories/               # Factory implementations
│   ├── providers/               # Provider-specific implementations
│   │   └── onramper/           # Onramper provider
│   ├── exceptions/              # Custom exceptions
│   └── config/                  # Spring configurations
└── test/                        # Unit tests
```

Cài đặt và sử dụng

Yêu cầu hệ thống

- Java 17+
- Maven 3.6+
- Spring Boot 3.x

Cài đặt dependencies

Bash

```
mvn clean install
```

Cấu hình

Thêm cấu hình vào `application.yml`:

YAML

```
onramp:
  providers:
    onramper:
      api-key: ${ONRAMPER_API_KEY}
      api-secret: ${ONRAMPER_API_SECRET}
      base-url: https://api.onramper.com
      is-sandbox: false
```

```
      timeout: 30
      retry-attempts: 3
```

Sử dụng cơ bản

Java

```
@Autowired
private OnRampServiceFactory serviceFactory;

// Tạo service cho Onramper
OnRampConfig config = OnRampConfig.builder()
    .providerName("onramper")
    .apiKey("your-api-key")
    .baseUrl("https://api.onramper.com")
    .build();

OnRampService service = serviceFactory.createService("onramper", config);

// Lấy báo giá
CompletableFuture<Quote> quoteFuture = service.getQuote("USD", "BTC", 100.0,
null);
Quote quote = quoteFuture.get();

// Tạo đơn hàng
CompletableFuture<Order> orderFuture = service.createOrder(
    "USD", "BTC", 100.0, null,
    "1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa",
    "https://yoursite.com/callback"
);
Order order = orderFuture.get();
```

Mở rộng hệ thống

Thêm nhà cung cấp mới

1. **Tạo package mới**: `src/main/java/com/onramp/integration/providers/newprovider/`
2. **Implement OnRampService**:

Java

```
@Service
public class NewProviderService implements OnRampService,
ConfigurableOnRampService {
```

```
    // Implementation details
}
```

1. **Đăng ký trong Factory**:

Java

```
// Trong DefaultOnRampServiceFactory.initializeProviderRegistry()
providerRegistry.put("newprovider", NewProviderService.class);
```

1. **Tạo response DTOs** cho API của nhà cung cấp mới
2. **Viết unit tests** cho implementation mới

Ví dụ: Thêm MoonPay

Java

```
@Service
public class MoonPayService implements OnRampService,
ConfigurableOnRampService {

    @Override
    public CompletableFuture<Quote> getQuote(String fiatCurrency, String
cryptoCurrency,
                                             Double fiatAmount, Double
cryptoAmount) {
        // MoonPay specific implementation
        return webClient.get()
            .uri("/v3/currencies/{crypto}/quote", cryptoCurrency)
            .retrieve()
            .bodyToMono(MoonPayQuoteResponse.class)
            .map(this::convertToQuote)
            .toFuture();
    }

    // Other methods...
}
```

Testing

Chạy tests

Bash

```
# Chạy tất cả tests
mvn test

# Chạy tests cho một provider cụ thể
mvn test -Dtest=OnramperServiceTest

# Chạy tests với coverage
mvn test jacoco:report
```

Test Coverage

Hệ thống có test coverage > 90% với các test cases:

- Unit tests cho tất cả service methods
- Integration tests cho factory pattern
- Error handling tests
- Configuration validation tests

API Reference

Core Interfaces

OnRampService

Java

```
public interface OnRampService {
    CompletableFuture<List<Asset>> getSupportedAssets();
    CompletableFuture<Quote> getQuote(String fiatCurrency, String
cryptoCurrency,
                                      Double fiatAmount, Double cryptoAmount);
    CompletableFuture<Order> createOrder(String fiatCurrency, String
cryptoCurrency,
                                         Double fiatAmount, Double
cryptoAmount,
                                         String walletAddress, String
redirectUrl);
    CompletableFuture<Order> getOrderStatus(String orderId);
    CompletableFuture<List<PaymentMethod>> getPaymentMethods(String
fiatCurrency, String cryptoCurrency);
    CompletableFuture<List<Transaction>> getTransactionHistory(String
userId);
    String getProviderName();
```

```
    CompletableFuture<Boolean> isServiceAvailable();
    CompletableFuture<Boolean> validateConfiguration();
}
```

OnRampServiceFactory

Java

```
public abstract class OnRampServiceFactory {
    public abstract OnRampService createService(String providerName,
OnRampConfig config);
    public abstract boolean isProviderSupported(String providerName);
    public abstract String[] getSupportedProviders();
    public abstract OnRampService createServiceWithDefaultConfig(String
providerName);
}
```

Domain Models

Quote

Java

```
@Data
@Builder
public class Quote {
    private String fiatCurrency;
    private String cryptoCurrency;
    private Double fiatAmount;
    private Double cryptoAmount;
    private Double exchangeRate;
    private Double fee;
    private Double totalFiatAmount;
    private String providerName;
    private LocalDateTime expiresAt;
    private Map<String, Object> metadata;
}
```

Order

Java

```
@Entity
@Data
```

```
@Builder
public class Order {
    @Id
    private String orderId;
    private String externalOrderId;
    private String providerName;
    private String fiatCurrency;
    private String cryptoCurrency;
    private Double fiatAmount;
    private Double cryptoAmount;
    private String walletAddress;
    private String redirectUrl;
    private OrderStatus status;
    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
    private Map<String, Object> metadata;
}
```

Troubleshooting

Lỗi thường gặp

1. **InvalidConfigurationException**: Kiểm tra API keys và cấu hình
2. **ProviderNotSupportedException**: Đảm bảo provider đã được đăng ký trong factory
3. **WebClient timeout**: Tăng timeout trong cấu hình
4. **API rate limiting**: Implement retry logic với exponential backoff

Logging

Hệ thống sử dụng SLF4J với Logback. Cấu hình logging level:

YAML

```
logging:
  level:
    com.onramp.integration: DEBUG
    org.springframework.web.reactive: DEBUG
```

Contributing

1. Fork repository
2. Tạo feature branch: `git checkout -b feature/new-provider`

3. Commit changes: `git commit -am 'Add new provider'`
4. Push branch: `git push origin feature/new-provider`
5. Tạo Pull Request

Code Style

- Sử dụng Google Java Style Guide
- Lombok annotations để giảm boilerplate
- Comprehensive JavaDoc cho public APIs
- Unit tests cho tất cả new features

License

MIT License - xem file LICENSE để biết chi tiết.

Support

- GitHub Issues: [Link to issues]
- Documentation: [Link to docs]
- Email: support@yourcompany.com

Tác giả: Manus AI

Phiên bản: 1.0.0

Cập nhật lần cuối: 2025-09-04