*Tìm hiểu, viết tài liệu phần Kiến trúc, luồng đi của Request từ app => transaction=>epay=>csb,osb => core, outgoing.*

*Note: chỉnh lại template file để gắn các mục 1,2,3,4 theo navigation*

*bổ sung thêm => thao tác app/postman =>  phần trace log trên ELK để ra đường đi API*

# **I. Mô hình kiến trúc :**

A diagram of a company

Description automatically generated

Hệ thống có 1 domain chính, triển khai trên nền Openshift: Epay domain.

Epay domain bao gồm nhiều service nghiệp vụ khác nhau, gọi là các Epay Services. Ngoài các service nghiệp vụ, Epay domain cũng bao gồm các Adapter tích hợp để tích hợp với các hệ thống sẵn có như HSM, CSB và hệ thống API Microservice.

Epay domain giao tiếp với hệ thống CSB thông qua ActiveMQ.

Redis cũng được sử dụng để làm hệ thống cache trung tâm cho hệ thống.

Hệ thống Epay có các lớp adapter, dùng để kết nối tới các hệ thống khác:

* **HSM Adapter**: kết nối tới hệ thống HSM để xác thực các thông tin bảo mật của người dùng cũng như của hệ thống
* **HTTP Adapter**: kết nối tới các hệ thống khác theo giao thức HTTP như OSB, hoặc các hệ thống microservice khác
* **CSB Adapter**: Nơi trung gian thực hiện việc quản lý message.

Hệ thống Epay cũng cần kết nối tới các service lớp Atomic như Customer Management Service và Card Management Service. Các kết nối này đều thông qua giao thức GRPC. Mỗi service lớp Atomic đều được kiến trúc một cách độc lập, có các thành phần cơ bản bao gồm:

* **Orchestration**: là thành phần nhận các request từ client và điều phối các service nghiệp vụ
* **Services**: là các service nghiệp vụ đơn lẻ
* **Adapters**: là các thành phần tích hợp hệ thống, chịu trách nhiệm kết nối tới các hệ thống khác

# **II: Luồng đi chung xử lý request**

A diagram of a software company

Description automatically generated

**B1**: Request sẽ được xác thực user/password (basic Authentication), nếu false thì kết thúc

**B2**: Request đi qua doFilter() – EwalletReplayAttackFilter.java . Tại đây sẽ kiểm tra requestID, requestDateTime, lưu cache Request, nếu yêu cầu false thì kết thúc.

**B3:** Request đi qua doFilter() – EwalletRequestFilter.java. Tại đây kiểm tra quyền “FunctionName” , kiểm tra “IsSession”, giải mã data, nếu false thì kết thúc.

**B4:** Request đi vào Controller, tại đây gọi hàm “RouteServices. routeFunction()” để điều hướng tới class service nghiệp vụ bằng “FunctionName”.

**B5:** Service gọi các service : repository => DB, service Utils, Adapter thực hiện chức năng nghiệp vụ.

**B6:** Sau khi hàm nghiệp vụ xử lý xong, dữ liệu trả ra đi qua ProcessBodyAdvice.java, lớp này sẽ thực hiện xử lý dữ liệu trước khi trả ra client, cụ thể thay đổi message, mã hóa data. => kết thúc trả data cho client.

# **III: Luồng đi từ epay tới CSB**

Epay domain giao tiếp với CSB qua tầng Atomic

* **Ochestration:** là service sẽ định nghĩa các đầu API, khởi tạo workflow cho từng api.
* **Validator:** là service sẽ thực hiện chức năng validate dữ liệu đầu vào theo nghiệp vụ khi service **Ochestration** tiếp nhận request từ phía client
* **Service:** là service sẽ nhận dữ liệu được truyền từ service **Ochestration** thông qua **GRPC**. Chức năng chính là gửi dữ liệu lên **Queue(ActiveMQ)** và xử lý một số nghiệp vụ cơ bản.
* **Adapter:** là service thực thi nghiệp vụ chính trong tầng Atomic.

A screenshot of a computer

Description automatically generated

**B1**: Client call vào **Ochestration -** service(nhận dữ liệu từ client, khởi tạo workflow, điều hướng đến các service liên quan)

**B2:** Sau khi nhận dữ liệu từ client, **Ochestration-** servicesẽ gửi dữ liệu tới **validate service** thông quan **GRPC**.

**B3:** Sau khi nhận dữ liệu và thực thi validate sau đó validate-service sẽ trả dữ liệu về cho service **Ochestration**.

**B4**: Sau khi đã tiếp nhận kết quả từ validate-service

1. Success: Tiếp tục luồng xử lý.
2. Fail: Kết thúc luồng xử lý và trả ra mã lỗi tương ứng.

**B5**: service **Ochestration** truyền dữ liệu qua **Service** thông qua GRPC.

**B6**: **Service** tiếp nhận dữ liệu và gửi lên **queue.**

**B7**: **Adapter-serivce** sẽ tiếp nhận và xử lý **queue.**

**B8**: Sau khi **Adapter-service** xử lý xong sẽ gửi lại lên **queue.**

**B9**: **Service** tiếp nhận và trả về dữ liệu cho **Ochestration.**

**B10**: **Ochestration** trả về cho client

# **IV: Thao tác app/postman**

## **4.1: Thao tác call Postman với api**

Thao tác call Postman: dự án ePay mục đích chuyển đổi từ trục C# sang Java nên khi chuyển đổi sẽ lấy response của C# là kết quả đối chiếu.

**VD:** Hình ảnh của 1 thao tác trên Postman

A screenshot of a computer

Description automatically generated

Đặc tả của từng phần, mỗi request tới hệ thống bao gồm các thành phần:

* Dịch vụ: RESTful service
* Phương thức: POST
* Giao thức TLS1.2
* Cấu trúc thông điệp: JSON

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thành phần** | **Cấp độ** | **Tham số** | **Ghi chú** |
| 1 | **Header** | L1 |  |  |
| 2 | L2 | Authorization | Tương ứng với mỗi request, đối tác cần gửi thông tin Username và Password đã được cung cấp để xác thực.  **Phương thức xác thực:** Basic authentication |
| 3 | L2 | Accept | Thông số chấp nhận định dạng   * application/json |
| 4 | L2 | Content-Type | Định dạng loại dữ liệu truyền nhận   * application/json |
| 5 |  | L2 | Method | POST |
| 6 | **Body** | L1 |  | Thông tin trao đổi giữa host LVT và host Sacombank  Thông tin body của từng API sẽ được mã hóa theo chuẩn 3DES với key là password do Sacombank cung cấp. |

**B1: Bấm chọn button + (hoặc Ctrl +T ) để new 1 Post request.**

A screenshot of a computer

Description automatically generated

**B2: Bổ sung đường dẫn URL để gọi tới các môi trường khác nhau**

**VD: Đường dẫn C#:** (đây là đường dẫn để lấy kết quả đối chiếu với code java, theo dự án ePay)

**A screenshot of a computer

Description automatically generated**

Tuỳ theo dự án mà sẽ có các đường dẫn tới môi trường khác nhau (UAT, SIT, Staging, ..)

VD: SIT java (Các đường dẫn này thường sẽ lưu ở file thông tin dự án)

<https://ewallet-ingress-gateway-istio-system.apps.uat-cloud.sacombank.local/ewallet-gateway-openshift/>

**B3: Nhập các thông tin ở phần Header**

Mục header cung cấp các thông tin bổ sung giữa client và server, tại đây thường chứa các yêu cầu xác thực như Authorization

**A screenshot of a computer

Description automatically generated**

**B4: Nhập các thông tin request của Body**

Phần body của 1 request mục đích đóng gói dữ liệu được gửi từ client đến server. Ở đây, dự án ePay sẽ là định dạng JSON ( khi call sang hệ thống khác thì sẽ đề cập ở mục sau) **A screenshot of a computer

Description automatically generated**

**Thông tin trong body:**

**A screenshot of a computer code

Description automatically generatedData:** Chứa thông tin chính của request để gửi tới server

**FunctionName:** Tên của api

**RequestDateTime:** Thời gian gọi api

**RequestID:** Mã định danh của request là 1 chuỗi chứa 36 ký tự

**B5: Đóng gói dữ liệu được gửi từ client đến server**

Khi bấm button Send thì yêu cầu sẽ được gửi tới server.

**A screenshot of a computer

Description automatically generated**

Phần response trả về có các danh mục sau:

**Data** : Kết quả của dữ liệu sau khi được api xử lý trả về

**Function:** Tên của api khi gửi request đi

**RequestID:** Là mã định danh của request khi gửi, là mục **RequestID** trong phần body với giá trị là ***{{$guid}}***

**RequestDateTime:** Thời gian gọi api ở mục **RequestDateTime** trong phần body với giá trị là ***{{today}}***

**Description :** Mô tả kết quả khi gọi api, đi kèm với **RespCode** ở bên dưới

**RespCode:** Mã của kết quả khi gọi api.

VD: Một số mã thường gặp

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **note / Error Code** | **Description** | **Additional Information** |
| 1 | 00 | Financial transaction has been approved |  |
| 2 | 01 | Action fail |  |
| 3 | 02 | Invalid data |  |
| 4 | 03 | Invalid input |  |
| 5 | 05 | Do not honor | Chờ đối soát |
| 6 | 06 | Error | Chờ đối soát |
| 7 | 12 | Invalid transaction |  |
| 8 | 13 | Invalid Currency/Transaction Amount |  |
| 9 | 14 | Invalid account number |  |
|  | 17 |  |  |
| 10 | 18 | Card not support |  |
| 11 | 20 | Invalid response |  |
| 12 | 22 |  |  |
| 13 | 26 | Data exists |  |
| 14 | 30 | Format Error |  |
| 15 | 36 | Restricted card |  |
| 16 | 40 | Requested function not supported |  |
| 17 | 51 | Insufficient funds |  |
| 18 | 54 | Not sufficient funds |  |
| 19 | 55 | verify fail PIN/mPass/OTP/CVV |  |
| 20 | 57 | Transaction not permitted |  |
| 21 | 61 | Withdrawal amount limit exceeded |  |
| 22 | 68 | Issuer timed out | Chờ đối soát |
| 23 | 91 | Issuer inoperative | Chờ đối soát |
| 25 | 94 | Duplicate transmission. | This error is returned if a duplicate message is received and the transaction amount is different. |
| 26 | 96 | System malfunction | Chờ đối soát |
| 27 | 98 |  |  |

## **4.2: Thao tác call Postman với hệ thống khác**

### **4.2.1 Call RestAPI HTTP**

Thao tác call Postman với hệ thống khác: nhằm kiểm tra, đối chiếu khi gọi trực tiếp tới hệ thống khác mà không thông qua api service.

**VD:** Hình ảnh của 1 thao tác trên Postman

A screenshot of a computer

Description automatically generated

**B1: Bấm chọn button + (hoặc Ctrl +T ) để new 1 Post request.**

**B2: Bổ sung đường dẫn URL để gọi tới hệ thống cần gọi**

**VD:** <http://192.168.181.63/T24Live/Card/WalletService> (**gọi sang OSB Deposit)**

A screenshot of a computer

Description automatically generated

**B3: Nhập các thông tin ở phần Header**

Mục header cung cấp các thông tin bổ sung giữa client và server, tại đây thường chứa các yêu cầu xác thực như Authorization

**A screenshot of a computer

Description automatically generated**

**B4: Nhập các thông tin request của Body**

Phần body của 1 request mục đích đóng gói dữ liệu được gửi từ client đến server. Ở đây, khi gọi sang OSB sẽ là định dạng XML **A screenshot of a computer

Description automatically generated**

**B5: Đóng gói dữ liệu được gửi từ client đến server**

Khi bấm button Send thì yêu cầu sẽ được gửi tới server và nhận response.

**A screenshot of a computer

Description automatically generated**

### **4.2.2 Call RestAPI gRPC**

Postman trong máy remote chưa có chức năg call gRPC nên sẽ cập nhật sau

# **V: Thao tác trace log trên ELK**

Khi trace log của 1 api trước tiên cần RequestID của api đó

**VD:** Khi call 1 api phần response sẽ trả ra như sau

**A screenshot of a computer code

Description automatically generated**

**B1:** Truy cập vào trang Kibana với user/password đã được cấp:

A screenshot of a computer

Description automatically generated

**B2:** Khi vào trang chủ, mở menu bên trái, **B3:** Tại màn hình Discover, chọn vào

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedchọn vào mục Discover để vào phần xem log phần danh sách các project.

**B4:** Tại màn hình này có danh mục tìm kiếm các project, có thể nhập để tìm kiếm theo tên

VD: **ALL-EPAY, HSM, CSB**

A screenshot of a computer

Description automatically generated

**B5:** Ở màn chi tiết log của project, nhập RequestID của api để xem log của api đó.

A screenshot of a computer

Description automatically generated

Tại đây sẽ hiển thị log tổng quan của api, để xem chi tiết log của api khi call thì cần LMID của api đó.

**\*Note:** Có thể tìm kiếm theo các tuỳ chọn **AND, OR** để kết hợp các điều kiện tìm kiếm /

A screenshot of a computer

Description automatically generated

**B6:** Coppy LMID của api => nhập vào mục tìm kiếm.

Tại màn hình này sẽ hiển thị toàn bộ các logs khi call api.

A screenshot of a computer

Description automatically generated

Tại màn hình này sẽ hiển thị toàn bộ các logs khi call api. Các thông tin trong log ở hình sau, ngoài ra khi gọi tới hệ thống khác thì sẽ có thêm các thông tin liên quan tới hệ thống được gọi tới.

A screenshot of a computer

Description automatically generated

Như ví dụ trên, luồng đi của api **eWallerCustInfoUpdate** sẽ là :

Nhận request từ client, sau đó gọi sang **eWallet** rồi trả ra response. Đây là ví dụ khi call happycase, nếu call api gặp lỗi sẽ có log liên quan đến lỗi đó

**Ví dụ :** Đây là 1 case lỗi khi call api và gọi sang hệ thống khác.

A screenshot of a computer

Description automatically generated