TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ Thông tin và Truyền thông



Báo cáo: Project II

Đề tài: Xây dựng ApiGateway trong kiến trúc microservice

|  |  |
| --- | --- |
| Sinh viên thực hiện: | Phạm Văn Hùng 20173154 |
| Giảng viên hướng dẫn: | TS. Ngô Lam Trung |

*Hà Nội, tháng 5 năm 2020*

**Mục lục**

[I.  Mô tả tổng quan   2](#_Toc2143451783)

[1. Giới thiệu chung   3](#_Toc48435644)

[2. Mô tả các chức năng   3](#_Toc2129631175)

[3. Mục tiêu mong muốn   3](#_Toc1454851766)

[4. Mô tả công nghệ   3](#_Toc1103419529)

[II.  Mô tả chi tiết 4](#_Toc639383950)

[1. Mô tả database  4](#_Toc991796152)

[2. Mô tả chi tiết chức năng   8](#_Toc526480811)

[2.1 Mô tả chức năng đăng ký    8](#_Toc1196658475)

[2.2 Mô tả chức năng đăng nhập  11](#_Toc804069622)

[2.3 Mô tả chức năng điều hướng api   14](#_Toc45992618)

[2.4 Mô tả chức năng đổi mật khẩu   18](#_Toc777320608)

[2.5 Mô tả chức năng ghi log hệ thống                21](#_Toc1824911378)

[2.6 Mô tả chức năng load balancing       21](#_Toc1007028800)

1. **Mô tả tổng quan**

**1. Giới thiệu chung**

- Ngày nay khi chúng ta xây dựng một sản phẩm theo kiến trúc microservice sẽ có rấ nhiều các service nhỏ tập hợp lại thành một hệ thống lớn.

- Khi đó sẽ phát sinh ra một vấn đề đó là các service sẽ giao tiếp với nhau như thế nào?

- Giả sử chúng ta sẽ để các service giao tiếp trức tiếp với nhau. Như vậy nếu coi mỗi service là một điểm trên đồ thì vô hướng thì việc các service giao tiếp trực tiếp với nhau sẽ tạo ra một đồ thị đầy đủ

- Điều này dẫn đến việc quản lý việc giao tiếp này sẽ vô cùng phức tạp và khó khăn.

- Từ những khó khăn đó mà hệ thống ApiGateway đã được ra đời đóng vai trò là cửa ngõ để giao tiếp với các service.

**2. Mô tả các chức năng**

- Làm cửa ngõ để giao tiếp giữa các service trong kiến trúc microservice

- Thực hiện quản lý các service của hệ thống

- Thực hiện xác thực và phân quyền user khi truy cập hệ thống

**3. Mục tiêu mong muốn**

- Xây dựng được các chức năng chính cho hệ thống

+) Đăng ký , đăng nhập cho user

+) Thực hiện phân quyền theo từng application và các service

+) Thực hiện điều hướng truy cập tới các service

- Thực hiện deploy ứng dụng, cấu hình ngnix, load balance

**4. Mô tả công nghệ**

- Ngôn ngữ: Java

- Framework: Sping MVC

- Database: PostgreSQL, Redis

- MessageQueue: Kafka

- Server: Tomcat

1. **Mô tả chi tiết**

**1. Mô tả database**

Bảng AG\_USER

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_USER\_ID | UUID | F |  |
| USERNAME | VARCHAR(100) | F |  |
| PASSWORD | VARCHAR(255) | F |  |
| SALT | VARCHAR(255) | F |  |
| EMAIL | VARCHAR(255) | F |  |
| STATUS | BOOL | F |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_APP

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_APP\_ID | UUID | F |  |
| APP\_NAME | VARCHAR(100) | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_SERVICE

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_SERVICE\_ID | UUID | F |  |
| SERVICE\_NAME | VARCHAR(100) | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_API

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_API\_ID | UUID | F |  |
| API\_NAME | VARCHAR(100) | F |  |
| API\_KEY | VARCHAR(255) | F |  |
| AG\_APP\_ID | UUID | F |  |
| AG\_SERVICE\_ID | UUID | F |  |
| ENDPOINT | VARCHAR(500) | F |  |
| RESOURCE | VARCHAR(500) | F |  |
| METHOD | INT4 | F |  |
| TIMEOUT | INT4 | F |  |
| VERSION | VARCHAR(50) | T |  |
| IS\_AUTH | BOOL | F |  |
| IS\_OTP | BOOL | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| TYPE | INT4 | T |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_ROLE

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_ROLE\_ID | UUID | F |  |
| ROLE\_NAME | VARCHAR(100) | F |  |
| ACCEPT\_METHOD | INT4 | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_THROTTLING

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_THROTTLING\_ID | UUID | F |  |
| THROTTLING\_NAME | VARCHAR(100) | F |  |
| NUM\_OF\_REQUEST | INT4 | F |  |
| TIME\_SECOND | INT4 | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_ROLE\_API

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_ROLE\_API\_ID | UUID | F |  |
| AG\_USER\_ID | UUID | F |  |
| AG\_APP\_ID | UUID | T |  |
| AG\_SERVICE\_ID | UUID | T |  |
| AG\_API\_ID | UUID | T |  |
| AG\_ROLE\_ID | UUID | F |  |
| AG\_THROTTLING\_ID | UUID | F |  |
| STATUS | BOOL | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

Bảng AG\_ERROR\_CODE

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| NAME\_ERROR | VARCHAR(255) | F |  |
| ERROR\_CODE | VARCHAR(10) | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |

Bảng AG\_APP\_CONFIG

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| KEY | VARCHAR(255) | F |  |
| VALUE | VARCHAR(255) | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |

Bảng AG\_LOG

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_LOG\_ID | UUID | F |  |
| ERROR\_CODE | VARCHAR(10) | F |  |
| START\_TIME | TIMESTAMP | F |  |
| END\_TIME | TIMESTAMP | F |  |
| PROCESS\_TIME | INT4 | F |  |
| IP | VARCHAR(30) | F |  |
| API\_NAME | VARCHAR(255) | F |  |
| USERNAME | VARCHAR(255) | F |  |
| REQUEST | TEXT | F |  |
| RESPONSE | TEXT | F |  |

**2. Mô tả chi tiết chức năng**

**2.1 Mô tả chức năng đăng ký**

a) Mô tả thay đổi database

- Tạo function để tạo user mới và gán quyền mặc định

CREATE OR REPLACE FUNCTION create\_new\_user(in\_uname varchar, in\_pass varchar, in\_salt varchar, in\_email varchar)

RETURNS varchar

AS $$

DECRALE

    user\_id ag\_user.ag\_user\_id%type;

    app\_id ag\_app.ag\_app\_id%type;

    service\_id ag\_service.ag\_service\_id%type;

    role\_id ag\_role.ag\_role\_id%type;

    throttling\_id ag\_throttling.ag\_throttling\_id%type;

BEGIN

    insert into ag\_user(username , "password", salt, email, status,created\_date)

    values (in\_uname, in\_pass, in\_salt, in\_email, true, now())

    returning ag\_user\_id into user\_id;

    select a.ag\_app\_id into app\_id

    from ag\_app a where a.app\_name = 'TEST\_APP';

    select a.ag\_service\_id into service\_id

    from ag\_service a where a.service\_name = 'TEST\_SERVICE';

    select a.ag\_role\_id into role\_id

    from ag\_role a where a.role\_name = 'CALL\_API';

    select a.ag\_throttling\_id into throttling\_id

    from ag\_throttling a where a.throttling\_name = 'TEST\_THROTTLING';

    insert into ag\_role\_api (ag\_app\_id, ag\_service\_id , ag\_role\_id, ag\_throttling\_id, ag\_user\_id )

    values (app\_id, service\_id, role\_id, throttling\_id, user\_id);

    RETURN user\_id;

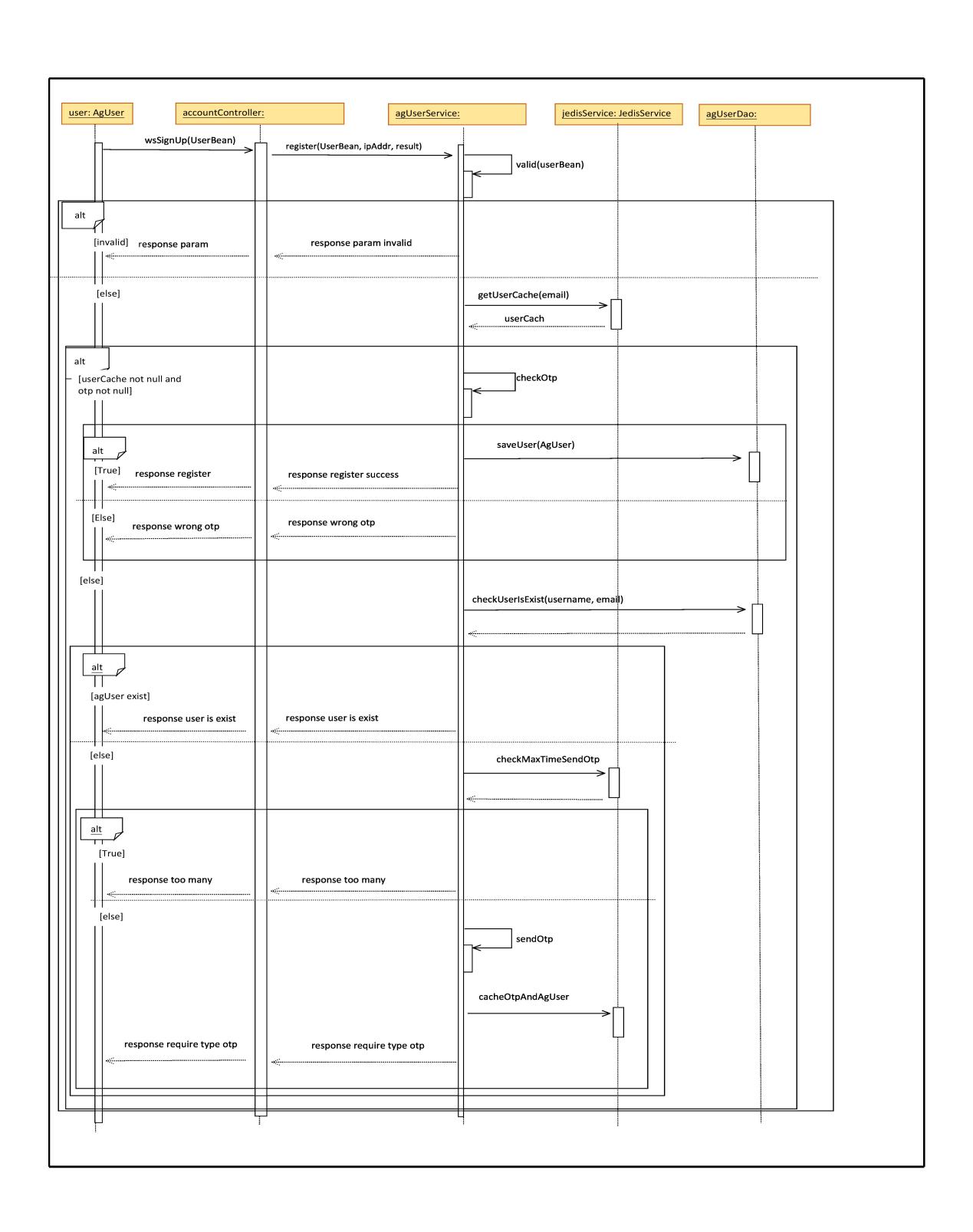
END;$$

LANGUAGE plpgsql;

b) Mô tả api

|  |  |  |
| --- | --- | --- |
| Tên API | wsSignUp |  |
| Tham số | username  password  validPassword  email |  |
| Kết quả trả về | Mã lỗi:  "0": Thành công,  "1": Thất bại |  |
| Bản tin request | {  "username":"tuhalang",  "password":"123456",  "validPassword":"123456",  "[email":"vanhung101299@gmail.com](mailto:email":"vanhung101299@gmail.com" \t "/home/tuhalang/Documents\\x/_blank)",  "otp":""  } | {  "username":"tuhalang",  "password":"123456",  "validPassword":"123456",  "[email":"vanhung101299@gmail.com](mailto:email":"vanhung101299@gmail.com" \t "/home/tuhalang/Documents\\x/_blank)",  "otp":"1234"  } |
| Bản tin response | {  "errorCode": "0",  "message": "Register successfully !",  "description": null,  "wsResponse": null,  "userMessage": null  } |  |

c) Sơ đồ tuần tự



d) Mô tả luồng xử lý

B1: Thực hiện kiểm tra các trường bắt buộc

|  |  |
| --- | --- |
| username | not null |
| password | not null |
| validPassword | not null && equals password |
| email | not null |
| otp | not require |

- Nếu có trường không hợp lệ trả về errorCode:'1" cùng message "input invalid !"

B2: Lấy thông tin user từ redis cache

- Nếu user null hoặc otp gửi lên null thì chuyển đến bước 2

- Nếu user khác null thì thực hiện :

    + Lấy otp từ redis cache so sánh với otp gửi lên

    + Nếu đúng otp thì thực hiện đăng kí tài khoản

         - Thực hiện call function create\_new\_user để insert vào database

         - Trả về thông báo đăng kí thành công

    + Nếu sai otp thì trả về thông báo sai otp

B3: Kiểm tra email và username đã tồn tại chưa

- Thực hiện truy vấn cơ sở dữ liệu ở bảng APIGW.AG\_USER:

    SELECT 1 FROM APIGW a WHERE a.USERNAME= :username or a.EMAIL= :email

- Nếu đã tồn tại trả về errorCode:"1" cùng message "Username or email is exist !"

B4: Gửi OTP và cache user

- Lấy thông tin otp từ redis về

- Kiểm tra só lần đã gửi và thời gian gửi gần nhất

- Nếu số lần gửi vượt quá tối đa 5 lần gửi thì trả về thông báo đã gửi otp quá số lần

- Nếu thời gian lần gửi gần nhất chưa quá 1 phút thì không gửi tiếp trả về thông báo yêu cầu nhập otp

- Nếu 2 điều kiện trên đều thỏa mãn thì thực hiện gửi otp

- Cache otp và user info trên redis với thời gian 2 phút

- Trả về thông báo yêu cầu nhập otp

**2.2 Mô tả chức năng đăng nhập**

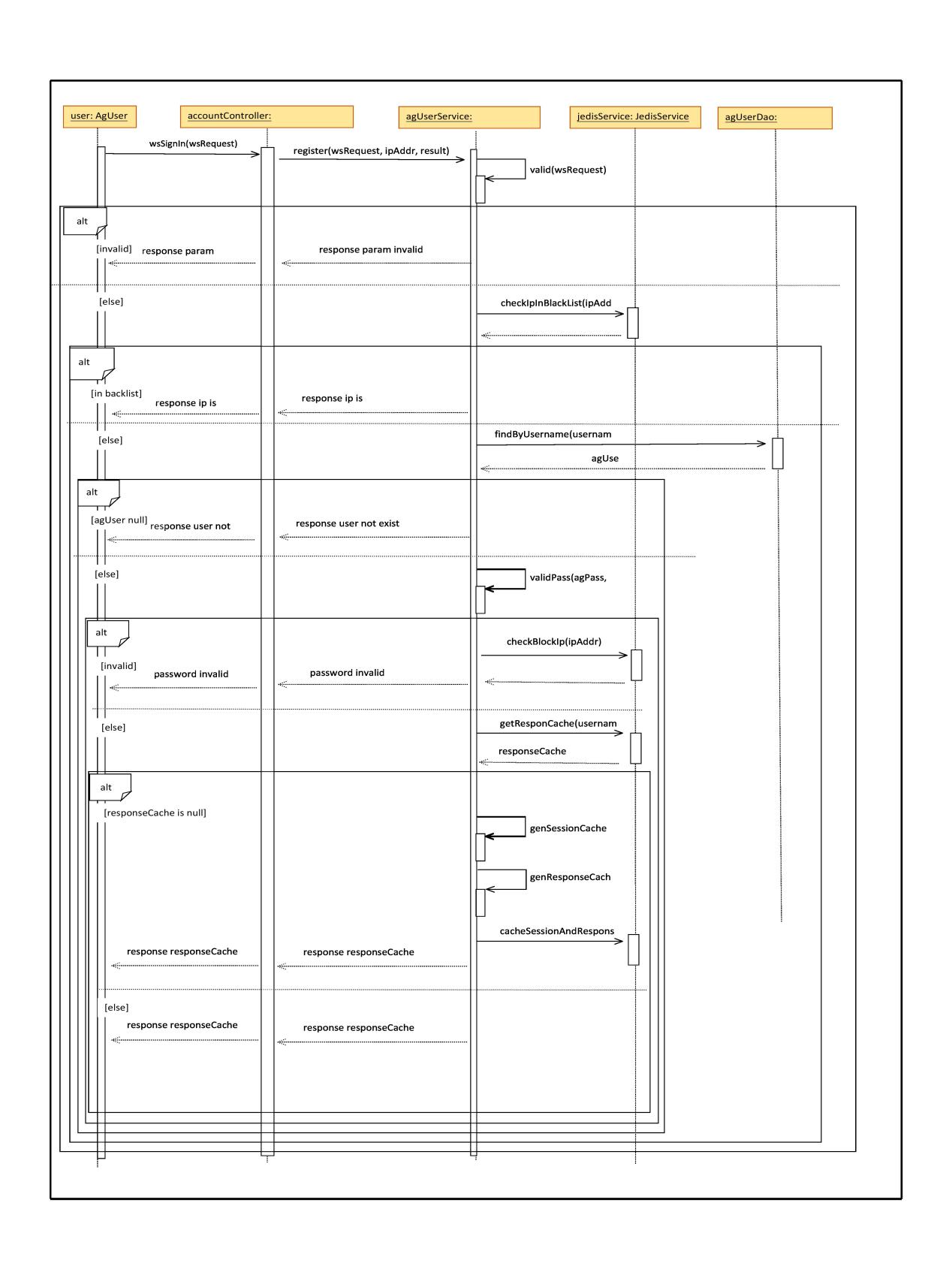
a) Mô tả thay đổi database

- N/A

b) Mô tả api

|  |  |
| --- | --- |
| Tên API | wsSignIn |
| Tham số | username  password  appName  serviceName |
| Kết quả trả về | Mã lỗi:  "0": Thành công,  "1": Thất bại |
| Bản tin request | {  "username" : "tuhalang",  "password" : "123456",  "appName" : "TEST\_APP",  "serviceName" : "SERVICE\_APP"  } |
| Bản tin response | {  "errorCode": "0",  "message": "Login Successfully !",  "description": null,  "wsResponse": {  "tokenKey": "FzeLt2WngYxHuuHlnS0aVg==",  "sessionId": "55c9399c-806f-4242-b73e-3a7bfbd43991"  },  "userMessage": null  } |

c) Sơ đồ tuần tự



d) Mô tả luồng xử lý

B1: Thực hiện valid các trường gửi lên

|  |  |
| --- | --- |
| username | not null |
| password | not null |

- Nếu có trường không hợp lệ trả về errorCode:'1" cùng message "input invalid !"

B2: Kiểm tra backlist

- Thực hiện kiểm tra trên redis cache số lần sai mật khẩu đã vượt quá gới hạn chưa

- Nếu chưa vượt quá chuyển đến bước 3

- Nếu đã vượt quá trả về erroCode 1

B3: Xác thực user

- Thực hiện kiểm tra username và password của user có đúng không

- Nếu đúng chuyển đến bược 4

- Nếu sai thì tăng số lần sai mật khẩu của user trên redis thêm 1 và trả về errorCode 1

B4: Lấy thông tin user từ redis cache

- Nếu user khác null thì gửi response trả về cho user

- Nếu user null thì chuyển đến bước 2

B5: Sinh token && sessionId

- Thực hiện sinh token và sessionId

- Cache thông tin đăng nhập của user trên redis

- Response trả về token và sessionId cho user

**2.3 Mô tả chức năng điều hướng api**

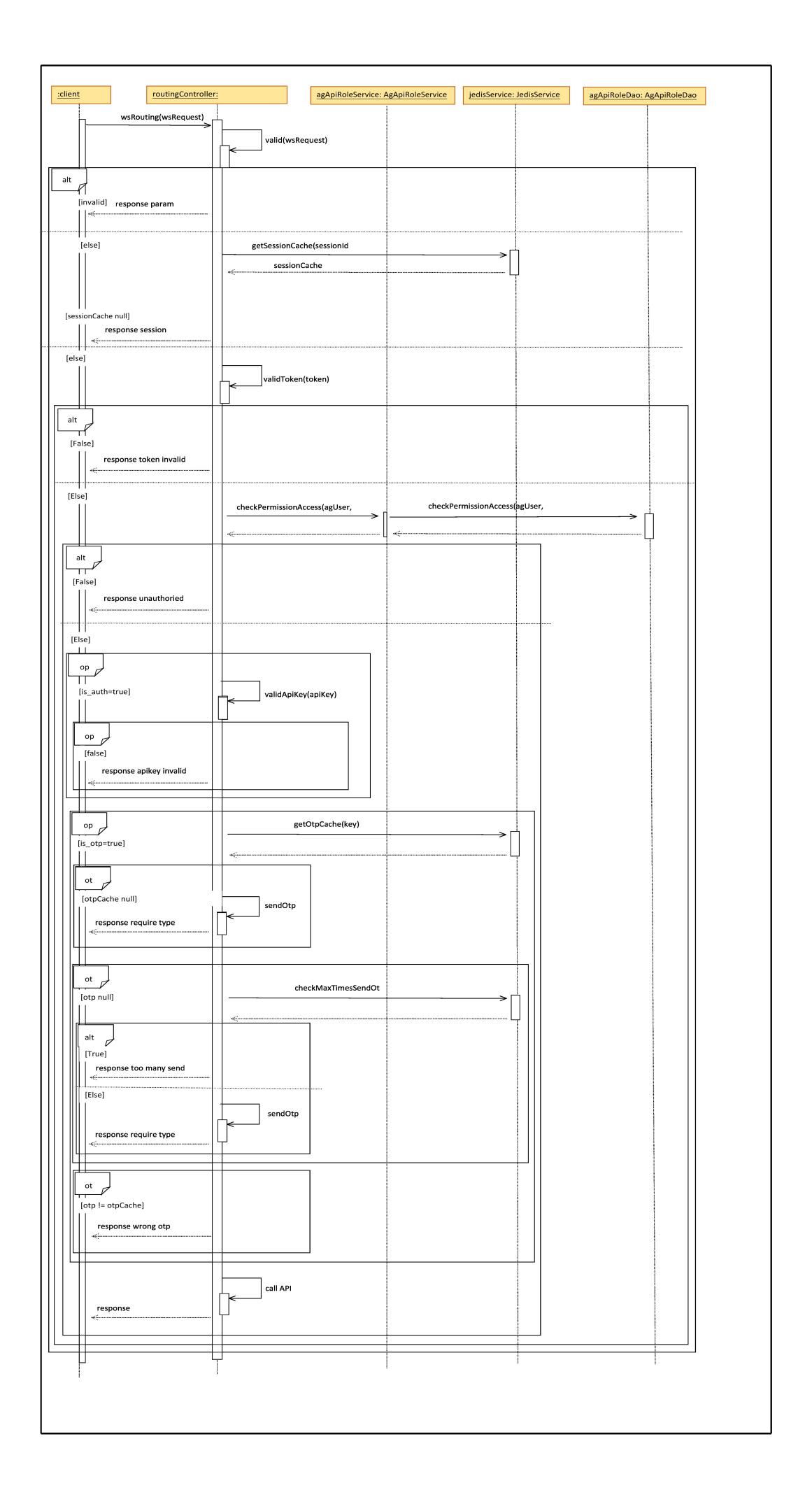
a) Mô tả thay đổi database

- N/A

b) Mô tả api

|  |  |
| --- | --- |
| Tên API | wsRouting |
| Tham số | sessionId  tokenKey  username  apiName  method  wsRequest |
| Kết quả trả về | Mã lỗi:  "0": Thành công,  "1": Thất bại |
| Bản tin request | {  "sessionId":"3741db86-b78e-4aac-b081-4a81864036a6",  "tokenKey":"FzeLt2WngYxHuuHlnS0aVg==",  "username":"tuhalang",  "apiName":"wsTestGet",  "method": "1"  "wsRequest":{    }  } |
| Bản tin response |  |

c) Sơ đồ tuần tự



d) Mô tả luồng xử lý

B1: valid dữ liệu

|  |  |
| --- | --- |
| sessionId | not null |
| tokenKey | not null |
| username | not null |
| apiName | not null |
| method | not null |
| wsRequest | not null |

B2: Get session từ redis cache

- Nếu session không tồn tại trả về errorCode 1 và thông báo không tồn tại session

- Nếu có session chuyển đến bước 3

B3: Valid token & check permission

- Valid token lưu trên redis bằng private key trong request của user

- Nếu token không hợp lệ trả về errorCode 1 và kết thúc xử lys

- Tiếp tục kiểm tra quyền truy cập API trong bảng ag\_role\_api

- Nếu user không có quyền truy cập trả về error code 1 và kết thúc xử lý

- Nếu API bắt xác thực apiKey thì chuyển đến bước B4

- Nếu API bắt xác thực bằng otp thì chuyển đến bước B5

- Chuyển đến bước B7

B4: Xác thực apiKey (nếu is\_auth=true)

- So sánh apiKey của api và apikey gửi lên từ client

      + Nếu sai trả về errorCode 1

B5: Xác thực otp (nếu is\_otp=true)

- Lấy otp của user từ redis cache

- Nếu otp không tồn tại chuyển đến bược 6

- Nếu otp tồn tại và otp trong request gửi lên khác null thì thực hiện so sánh với otp gửi lên trong request

    + Nếu đúng chuyển đến B7

    + Nếu sai trả về errorCode 1

B6: Thực hiện gửi otp và cache otp

- Lấy thông tin otp từ redis về

- Kiểm tra só lần đã gửi và thời gian gửi gần nhất

- Nếu số lần gửi vượt quá tối đa 5 lần gửi thì trả về thông báo đã gửi otp quá số lần

- Nếu thời gian lần gửi gần nhất chưa quá 1 phút thì không gửi tiếp trả về thông báo yêu cầu nhập otp

- Nếu 2 điều kiện trên đều thỏa mãn thì thực hiện gửi otp

- Cache otp và user info trên redis với thời gian 2 phút

- Trả về thông báo yêu cầu nhập otp

B7: Call API

- Thực hiện gọi api và trả về response cho client

2.4 Mô tả chức năng đổi mật khẩu

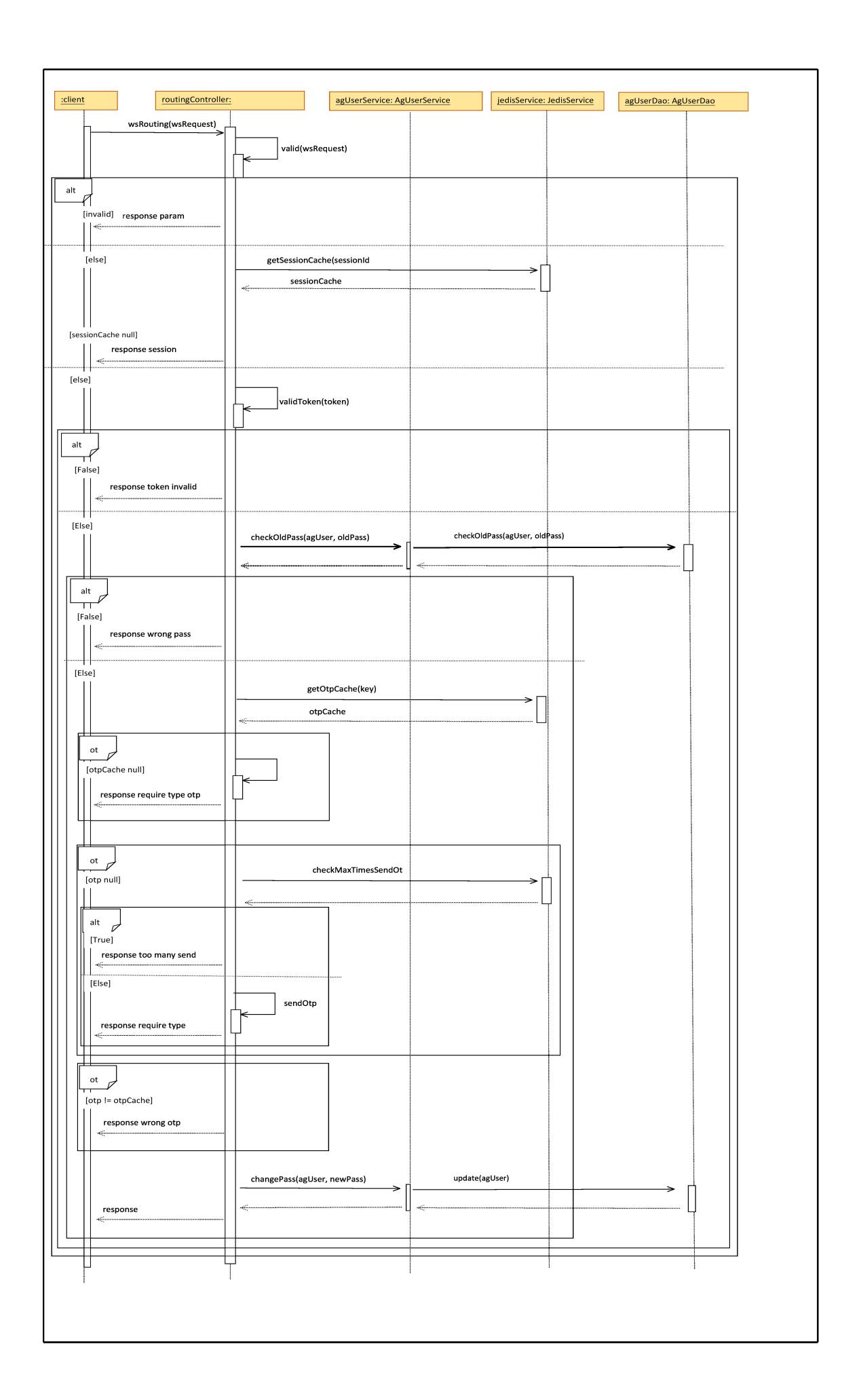
a) Mô tả thay đổi database

- N/A

b) Mô tả api

|  |  |
| --- | --- |
| Tên API | wsRouting |
| Tham số | sessionId  tokenKey  username  apiName  method  wsRequest |
| Kết quả trả về | Mã lỗi:  "0": Thành công,  "1": Thất bại |
| Bản tin request | {  "sessionId":"3741db86-b78e-4aac-b081-4a81864036a6",  "tokenKey":"FzeLt2WngYxHuuHlnS0aVg==",  "username":"tuhalang",  "apiName":"wsChangePass",  "wsRequest":{  "oldPass":"",  "newPass":""  }  } |
| Bản tin response |  |

c) Sơ đồ tuần tự



d) Mô tả luồng xử lý

B1: valid dữ liệu

|  |  |
| --- | --- |
| sessionId | not null |
| tokenKey | not null |
| username | not null |
| apiName | not null |
| method | not null |
| wsRequest | not null |
| oldPass | not null |
| newPass | not null |

B2: Get session từ redis cache

- Nếu session không tồn tại trả về errorCode 1 và thông báo không tồn tại session

- Nếu có session chuyển đến bước 3

B3: Valid token

- Valid token lưu trên redis bằng private key trong request của user

- Nếu token không hợp lệ trả về errorCode 1 và kết thúc xử lý

B4: Valid old pass

- Kiểm tra oldPass có trùng với password của user không

- Nếu trùng thì chuyển đến bước 5

- Nếu không trùng thì trả về errorCode 1

B5: Xác thực otp

- Lấy otp của user từ redis cache

- Nếu otp không tồn tại chuyển đến bược 6

- Nếu otp tồn tại và otp trong request gửi lên khác null thì thực hiện so sánh với otp gửi lên trong request

    + Nếu đúng chuyển đến B7

    + Nếu sai trả về errorCode 1

B6: Thực hiện gửi otp và cache otp

- Lấy thông tin otp từ redis về

- Kiểm tra só lần đã gửi và thời gian gửi gần nhất

- Nếu số lần gửi vượt quá tối đa 5 lần gửi thì trả về thông báo đã gửi otp quá số lần

- Nếu thời gian lần gửi gần nhất chưa quá 1 phút thì không gửi tiếp trả về thông báo yêu cầu nhập otp

- Nếu 2 điều kiện trên đều thỏa mãn thì thực hiện gửi otp

- Cache otp và user info trên redis với thời gian 2 phút

- Trả về thông báo yêu cầu nhập otp

B7: Thực hiện cập nhật password mới cho user

**2.5 Mô tả chức năng ghi log hệ thống**

a) Hiện trạng

- Hiện tại đang sử dụng ghi log vào file sử dụng thư viện log4j

b) Vấnđề găp phải

- AppiGateway là cửa ngõ của hệ thống nên cần ghi lại tất cả mọi hoạt động truy suất để phục vụ tra cứu, tracking trong tương lai

- Việc ghi log vào file hiện tại không tốt khi có nhiều lượt truy cập sẽ gậy chậm cho hệ thống

- Đồng thời việc ghi vào file cũng gây khó khăn cho việc tra cưu

c) Giảipháp

- Thực hiện ghi log vào database để có thể dễ ràng tra cứu

- Sử dụng message broker để đảm bảo hiệu suất cho việc ghi log

**2.6 Mô tả chức năng load balancing**

a) Mô tả thay đổi database

- Thay đổi bảng AG\_API xóa trường ENDPOINT

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_API\_ID | UUID | F |  |
| API\_NAME | VARCHAR(100) | F |  |
| API\_KEY | VARCHAR(255) | F |  |
| AG\_APP\_ID | UUID | F |  |
| AG\_SERVICE\_ID | UUID | F |  |
| RESOURCE | VARCHAR(500) | F |  |
| METHOD | INT4 | F |  |
| TIMEOUT | INT4 | F |  |
| VERSION | VARCHAR(50) | T |  |
| IS\_AUTH | BOOL | F |  |
| IS\_OTP | BOOL | F |  |
| DESCRIPTION | VARCHAR(500) | T |  |
| STATUS | BOOL | F |  |
| TYPE | INT4 | T |  |
| CREATED\_DATE | TIMESTAMP | T |  |
| CREATED\_BY | VARCHAR(255) | T |  |
| UPDATED\_DATE | TIMESTAMP | T |  |
| UPDATED\_BY | VARCHAR(255) | T |  |

- Tạo mới 02 bảng AG\_ENDPOINT và AG\_API\_ENDPOINT

AG\_ENDPOINT

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_ENDPOINT\_ID | UUID | F |  |
| ENDPOINT | VARCHAR(500) | F |  |
| URL\_HEALTHY\_CHECK | VARCHAR(500) | F |  |
| LAST\_USED | TIMESTAMP | T |  |
| STATUS | VARCHAR(10) | T |  |

AG\_API\_ENDPOINT

|  |  |  |  |
| --- | --- | --- | --- |
| **TÊN CỘT** | **LOẠI GIÁ TRỊ** | **NULLABLE** | **MÔ TẢ** |
| AG\_API\_ENDPOINT\_ID | UUID | F |  |
| AG\_ENDPOINT\_ID | VARCHAR(500) | F |  |
| AG\_API\_ID | VARCHAR(500) | F |  |

b) Sơ đồ tuần tự

- N/A

c) Mô tả luồng xử lý

- Thay đổi câu truy vấn để lấy endpoint cho api khi thực hiện routing

SELECT distinct a FROM AgApi a JOIN FETCH a.endpoints e WHERE a.apiName = :apiName and e.status=:status ORDER BY e.lastUsed

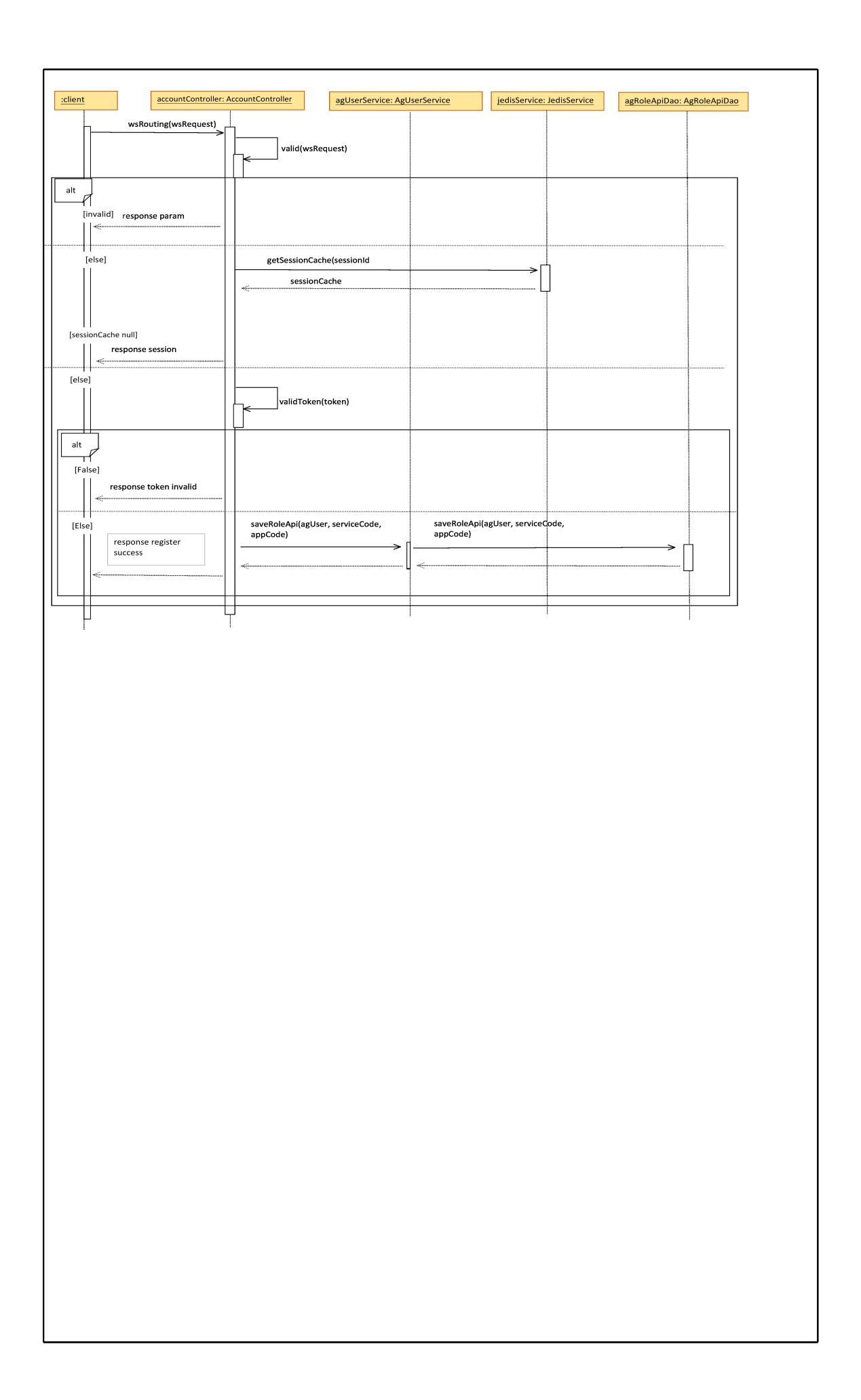
- Sau khi thực hiện gọi api thì cập nhật lại thời gian truy cập cho api là thời điểm gọi api.

**2.7 Mô tả chức năng đăng ký service**

a) Mô tả thay đổi database

- N/A

b) Biểu đồ tuần tự



c) Mô tả luồng xử lý

B1: valid dữ liệu

|  |  |
| --- | --- |
| sessionId | not null |
| tokenKey | not null |
| username | not null |
| apiName | not null |
| method | not null |
| wsRequest | not null |
| appCode | not null |
| serviceCode | not null |

B2: Get session từ redis cache

- Nếu session không tồn tại trả về errorCode 1 và thông báo không tồn tại session

- Nếu có session chuyển đến bước 3

B3: Valid token

- Valid token lưu trên redis bằng private key trong request của user

- Nếu token không hợp lệ trả về errorCode 1 và kết thúc xử lys

B4: Cập nhật quyền truy cập các api trong service cho user và thông báo về cho người dùng

**2.8 Mô tả cấu hình https cho server tomcat**

- Cấu hình thực hiện trên hệ điều hành Ubuntu-18.04

B1: Mở terminal và chạy câu lệnh để sinh ra key được lưu ở file ~/.keystore

keytool -genkey -alias tomcat -keyalg RSA

B2: Cấu hình tomcat

- Mở file conf/server.xml và sửa lại đoạn cấu hình như sau:

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true"

scheme="https" secure="true"

keystoreFile="/home/tuhalang/.keystore" keystorePass="tuhalang"

clientAuth="false" sslProtocol="TLS" >

<!--

<SSLHostConfig>

<Certificate certificateKeystoreFile="conf/localhost-rsa.jks"

type="RSA" />

</SSLHostConfig>

-->

</Connector>

B3: Khởi động lại server và truy cập theo port 8443 để truy cập ứng dụng.