Architecture

Background

Presently on EJARA, every microservice gets to manage its API client credentials and all related security needs or constraints. This begins raising security issues and concerns when the microservices keep increasing with each microservice having its client to manage and the security team having to deal with each service individually on the same subject. Notwithstanding, management becomes difficult.

This microservice is meant to manage all EJARA-related API clients needed for all its backend microservices having security constraints in focus but yet flexible.

Implementation Summary

This microservice is to be used by EJARA system administrators with the administrator being able to alter any record or configuration related to an API client object and equally regenerate client secrets for client objects that were say expired or forgotten.

We will be using an asymmetric encryption algorithm for our client secrets.

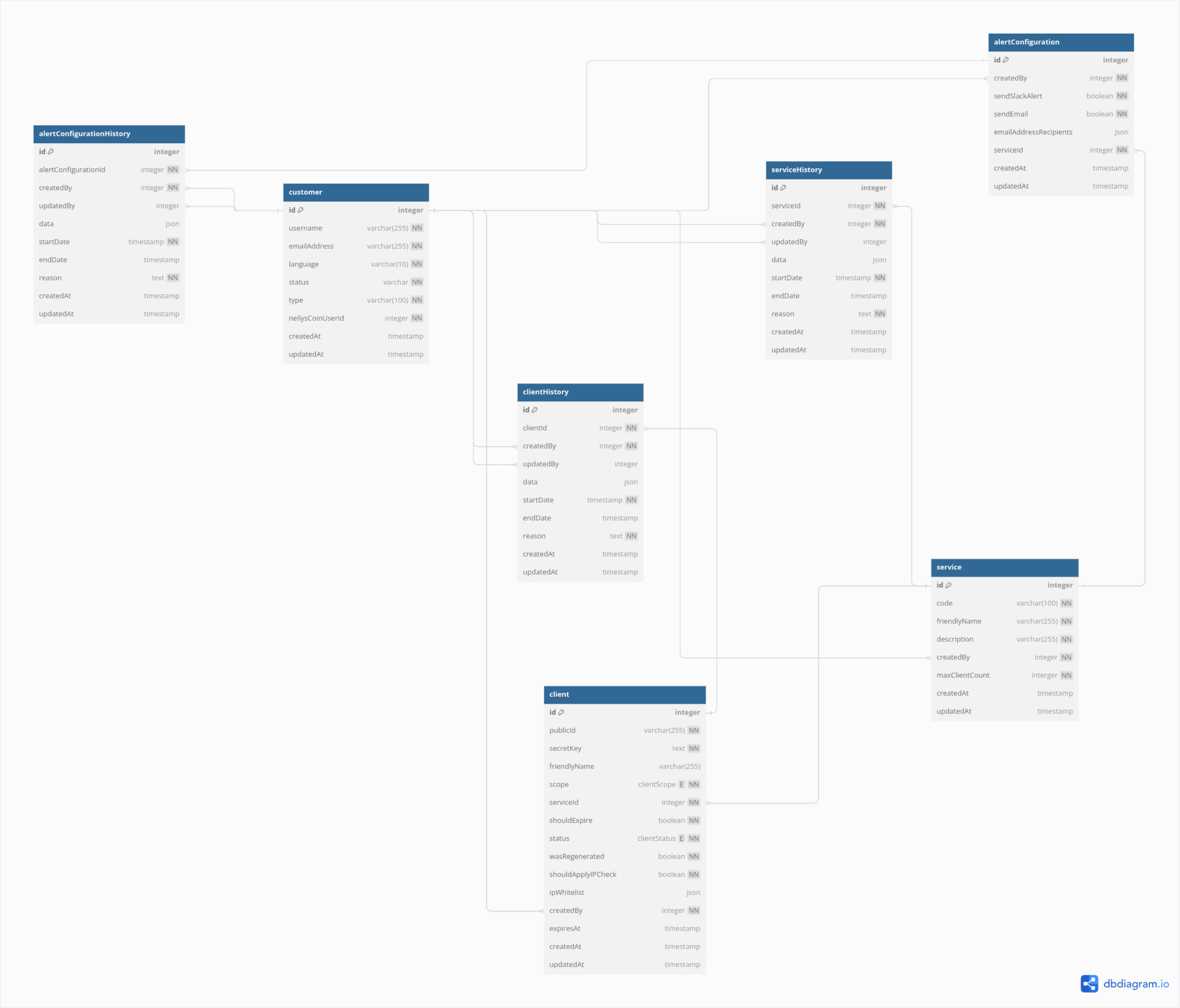
Administrators will be able to create API clients while providing the specific service that this client will be able to access.

Technology

- Database ➝ Postgres
- Backend ➝ NestJs, Prisma, Redis

Architecture Summary

https://dbdiagram.io/d/64d1e3ef02bd1c4a5e68933c



alertConfiguration		
id	integer	
createdBy	integer	NN
sendSlackAlert	boolean	NN
sendEmail	boolean	NN
emailAddressRecipients	json	
serviceId	integer	NN
createdAt	timestamp	
updatedAt	timestamp	

alertConfigurationHistory		
id	integer	
alertConfigurationId	integer	NN
createdBy	integer	NN
updatedBy	integer	
data	json	
startDate	timestamp	NN
endDate	timestamp	
reason	text	NN
createdAt	timestamp	
updatedAt	timestamp	

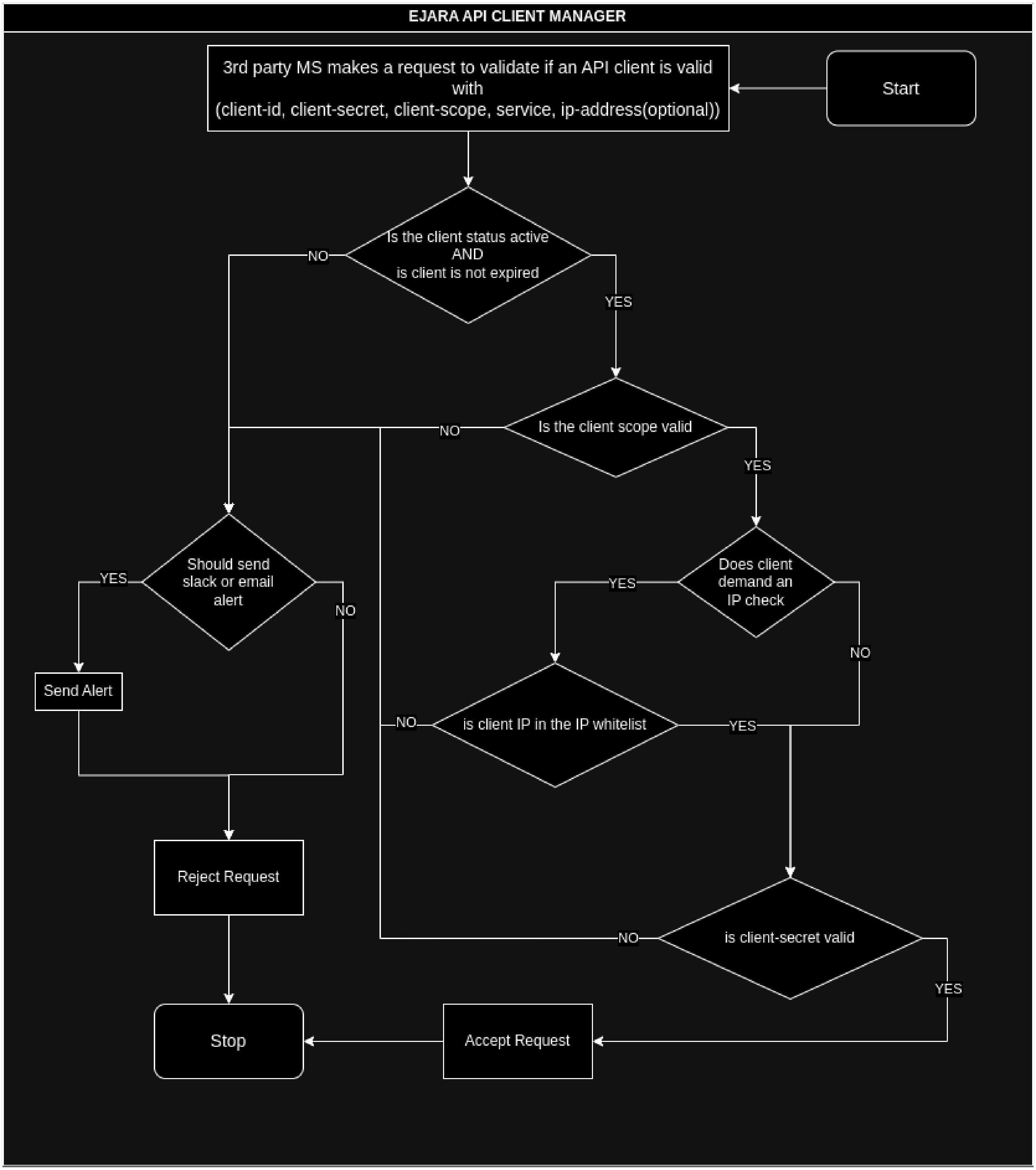
serviceHistory		
id	integer	
serviceId	integer	NN
createdBy	integer	NN
updatedBy	integer	
data	json	
startDate	timestamp	NN
endDate	timestamp	
reason	text	NN
createdAt	timestamp	
updatedAt	timestamp	

customer		
id	integer	
username	varchar(255)	NN
emailAddress	varchar(255)	NN
language	varchar(10)	NN
status	varchar	NN
type	varchar(100)	NN
nellysCoinUserId	integer	NN
createdAt	timestamp	
updatedAt	timestamp	

clientHistory		
id	integer	
clientId	integer	NN
createdBy	integer	NN
updatedBy	integer	
data	json	
startDate	timestamp	NN
endDate	timestamp	
reason	text	NN
createdAt	timestamp	
updatedAt	timestamp	

service		
id	integer	
code	varchar(100)	NN
friendlyName	varchar(255)	NN
description	varchar(255)	NN
createdBy	integer	NN
maxClientCount	interger	NN
createdAt	timestamp	
updatedAt	timestamp	

client		
id	integer	
publicId	varchar(255)	NN
secretKey	text	NN
friendlyName	varchar(255)	
scope	clientScope E	NN
serviceId	integer	NN
shouldExpire	boolean	NN
status	clientStatus E	NN
wasRegenerated	boolean	NN
shouldApplyIPCheck	boolean	NN
ipWhitelist	json	
createdBy	integer	NN
expiresAt	timestamp	
createdAt	timestamp	
updatedAt	timestamp	


EJARA API CLIENT MANAGER
Start
3rd party MS makes a request to validate if an API client is valid with (client-id, client-secret, client-scope, service, ip-address(optional))
Is the client status active AND is client is not expired
NO
YES
Is the client scope valid
NO
YES
Should send slack or email alert
YES
NO
Send Alert
Does client demand an IP check
YES
NO
is client IP in the IP whitelist
NO
YES
Reject Request
is client-secret valid
NO
YES
Stop
Accept Request

Backend Specs

TECHNOLOGY STACK

Backend Framework: NestJS
Database: Postgresql
ORM: Prisma

SCHEMA ➝ https://dbdiagram.io/d/64d1e3ef02bd1c4a5e68933c

ClientScope [ENUM]

```
WEB
MOBILE
MICROSERVICE
```

ClientStatus [ENUM]

```
ACTIVE
BLOCKED
EXPIRED
```

Customer

```
id: int autoincrement unique
username: string, not null unique
language: string, not null
emailAddress: string, not null
nellysCoinUserId: int, not null unique
status: string, not null
type: string, not null
createdAt: date, not null
updatedAt: date, not null
```

Service

```
id: int autoincrement unique
code: string, not null unique
friendlyName: string, not null
description: string, not null
maxClientCount: int, null
createdBy: int, not null
```

```
createdAt: date, not null
updatedAt: date, not null
```

ServiceHistory

```
id: int autoincrement unique
serviceId: int, not null
createdBy: int, not null
updatedBy: int, null
reason: text, not null
data: json, null
startDate: date, not null
endDate: date, null
createdAt: date, not null
updatedAt: date, not null
```

Client

```
id: int autoincrement unique
publicId: string, not null unique
friendlyName: string, not null
secretKey: string, not null
scope: ClientScope, not null
serviceId: int, not null
shouldExpire: boolean, not null
status: ClientStatus, not null
wasRegenerated: boolean, not null
shouldApplyIPCheck: boolean, not null
ipWhitelist: json, null
createdBy: int, not null
expiresAt: date, null
createdAt: date, not null
updatedAt: date, not null
```

ClientHistory

```
id: int autoincrement unique
clientId: int, not null
createdBy: int, not null
updatedBy: int, null
reason: text, not null
data: json, null
startDate: date, not null
endDate: date, null
```

```
createdAt: date, not null
updatedAt: date, not null
```

AlertConfiguration

```
id: int autoincrement unique
sendSlackAlert: boolean, not null @default(true)
createdBy: int, not null
serviceId: int, null
sendEmail: boolean, not null
emailAddressRecipients: json, null // list of email addresses
createdAt: date, not null
updatedAt: date, not null
```

AlertConfigurationHistory

```
id: int autoincrement unique
alertConfigurationId: int, not null
createdBy: int, not null
updatedBy: int, null
reason: text, not null
data: json, null
startDate: date, not null
endDate: date, null
createdAt: date, not null
updatedAt: date, not null
```

ENDPOINTS SECURITY

```
headers = {
  * client-id: string
  * client-secret: string
  * client-scope: string
  * service: string
  * authorization: <Bearer xxxxxxxx>
    ip-address: string
}
```

Language Header

```
headers = {
   language: string // The language code of the client
}
```

MIDDLEWARES

isAdmin

It is to be used to check if the user making the request is a valid EJARA admin

isUser

It is to be used to check if the user making the request is a valid EJARA customer client

isValidClient

It is to be used to check if all needed request headers for client validations are provided while appending the client IP address to the client request object. The required headers are `client-id`, `client-secret` , `client-scope` , `service` , and `ip-address [optional]`

The addition params to be appended to the request object are:

- client ➝ The API client object if found
- ipAddress ➝ The API client IP address

changeLanguage

To be used to set the client language code to be used. The default language code is `en` for English. We currently support two languages, which are `en` for English and `fr` for French

- You will have to supply a custom header with key `language` to set the client language code which can either be `en` or `fr`

ENDPOINTS

NOTE: `All request body properties with (*) are required`

POST /api/v1/services

- To be used in creating new services
- Can be accessed only by an admin with client-scope [web]

Request Body

```
{
    * friendlyName: string
    * description: string
```

```
      maxClientCount: number
}
```

Actions

Validations

- Check if a service with friendly name does not exist

Saving Process

- Create the service object

Response

```
On Success (201)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/services

- To be used in retrieving all registered services
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
    code: string
}
```

Response

```
On Success (200)
{
     message: string // the success message
}
```

```
On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/services/:serviceId

- To be used in retrieving a registered service
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{ }
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/services/history

- To be used in retrieving service history
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
  serviceId: number
  createdBy: number
  limit: number
  offset: number
}
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

PUT /api/v1/services/:serviceId

- To be used in updating a registered service
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
  * reason: string
    friendlyName: string
    description: string
    maxClientCount: number
}
```

Actions

Validations

- Ensure that no record exists with the given code or friendly name if any was provided

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

POST /api/v1/alert-configurations

- To be used in creating new configurations
- Can be accessed only by an admin with client-scope [web]

Request Body

```
{
    * sendSlackAlert: boolean
      sendEmail: boolean
      emailAddressRecipients: string[]
      serviceId: number
}
```

Actions

Validations

- Ensure the `emailRecipientAddresses` is required if **`sendEmail`** is true
- Validate service ID to ensure it exists

Saving Process

- Create the alert object

Response

```
On Success (201)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/alert-configurations

- To be used in retrieving all alert configs
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
  sendSlackAlert: boolean
  sendEmail: boolean
  serviceId: number
  limit: number
  offset: number
}
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/alert-configurations/history

- To be used in retrieving alert configuration history
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
  serviceId: number
  createdBy: number
  limit: number
  offset: number
}
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

PUT /api/v1/alert-configurations/:alertConfigurationId

- To be used in updating a registered service alert configuration
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
    * reason: string
      sendSlackAlert: boolean
      serviceId: number
      sendEmail: boolean
      emailAddressRecipients: string[]
}
```

Actions

Validations

- Ensure the `emailRecipientAddresses` is required if **`sendEmail`** is true

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/customers

- To be used in retrieving customers
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
    username: string
    nellysCoinUserId: int
    limit: number
    offset: number
}
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

PUT /api/v1/customers/:nellysCoinId

- To be used in updating a registered service alert configuration
- Can only be accessed by a client with client-scope [microservice]
- Prevent any customer object update or creation in the **isUser** middleware

Query Params

```
{
    * oldUsername: string
    * newUsername: string
      emailAddress: string
      status: string
      type: string
}
```

Actions

Validations

- Ensure that no record exists with the new username provided

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

POST /api/v1/clients

- To be used in creating new API client object
- Can be accessed only by an admin with client-scope [web]

Request Body

```
{
    * friendlyName: string
    * scope: string
    * serviceId: int
      shouldExpire: boolean
      shouldApplyIPCheck: boolean
      ipWhitelist: string[]
      expireAt: date
}
```

Actions

Validations

- Check if the service with ID exists
- Ensure that the max client object count for service is not exceeded
- Ensure that no record exists with the friendly name
- Ensure the ipWhitelist is required if the shouldApplyIPCheck is true
- Ensure the expireAt is required if the shouldExpire is true

Saving Process

- Create the client object and generate a secret

Response

```
On Success (201)
{
     message: string // the success message
}
```

```
On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/clients

- To be used in retrieving API clients
- Can be accessed only by an admin with client-scope [web]

Query Params

```
{
    publicId: string
    scope: string
    friendlyName: string
    serviceId: int
}
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

GET /api/v1/clients/:clientId/history

- To be used in retrieving API client's history

- Can be accessed only by an admin with client-scope [web]

Query Params

```
{ }
```

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

PUT /api/v1/clients/:clientId

- To be used in updating a registered client object
- Can be accessed only by an admin or manager linked to this client with client-scope [web]
- Prevent any customer object update or creation in the **isUser** middleware

Query Params

```
{
    scope: string
    serviceId: int
    shouldExpire: boolean
    expireAt: date
    shouldApplyIPCheck: boolean
    ipWhitelist: string[]
    status: string
    friendlyName: string
}
```

Actions

Validations

- Ensure that no record exists with the friendly name
- Ensure that if service ID is provided we check if the max client object count for the service is not exceeded
- Ensure the IP whitelist is required if the shouldApplyIPCheck is true
- Ensure the expireAt is required if the shouldExpire is true

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
     message: string // the error message
}


On Server Error (500)
{
     message: string // the error message
}
```

POST /api/v1/clients/:clientId/regenerate-secret

- To be used in regenerating the client's secret
- Can be accessed only by an admin or manager linked to this client with client-scope [web]

Query Params

```
{
    * reason: string
}
```

Actions

TODO

- Ensure that the client column of **wasRegenerated** is updated to true

Response

```
On Success (200)
{
     message: string // the success message
}


On Validation Error (400-451)
{
    message: string // the error message
}


On Server Error (500)
{
    message: string // the error message
}
```

POST /api/v1/clients/validate

- To be used in validating API client objects
- This is a bearer token free request
- Can be accessed by any client with the required request headers

Request Body

```
{ }
```

Actions

Validations

- Check if the client shouldApplyIPCheck and validate against the client IP whitelist
- Check if the client shouldExpire and fail the validation if the client object is already expired

TODO

- If every validation was a success, then return a successful response

Response

```
On Success (200)
{
    message: string // the success message
}


On Validation Error (400-451)
{
    message: string // the error message
}
```

```
On Server Error (500)
{
     message: string // the error message
}
```