Documentation

Erlando Dominico

1. Feature

1.1 Register

The register page is a crucial component of any system or website, enabling new users to create and manage their accounts. It involves a form requiring personal information, such as name, email, and password, and allows the system to learn about its users.

1.2 Login

The login page is a crucial security feature for systems or websites, allowing users to access it by verifying their identity through an API. It checks credentials against a database, granting access if the user has access or not. The page also collects user information, such as name, email address, and password, that they want to input. The result of this API is to generate a user access token (decrypted password) and save it in the database.

1.3 Show CoinMarketCap API

The CoinMarketCap API is an API to access and display CoinMarketCap data in applications using this API : https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest

2. environment

a. **API_CRYPTO :** Is to define Coin MarketCap API
b. API_KEY_CRYPTO: Is used to define API_KEY that got from Crypto Account
c. JWT_KEY: Is used to define JWT Secret Key
d. DB_URI: Is used to define DB Mongo’s string URL
e. DB_NAME :Is used to define database name
f. PORT: Is used to define port server
g. IS_AES: Is used to define whether the program is using an encrypted password with the AES 256 algorithm or not. This is to protect the front end not sending the original password

h. CIPHER_KEY: This is used to define a string for the secret key/ cipher key that is used to encrypt and decrypt the AES-256 algorithm. It must be 32 Characters

i. IV_KEY : Is used to define string for init vector that is used to encrypt/decrypt algorithm. It must be 16 Characters

j. CLIENT_KEY: Is used to define string for client key that already decided

```
k. export API_CRYPTO="https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest"
l. export API_KEY_CRYPTO="b54bcf4d-1bca-4e8e-9a24-22ff2c3d462c"
m. export JWT_KEY="vecktwfpkigdiqtpyhlqptihwxgqmyszxmldyscwigsc"
n. export DB_URI="mongodb+srv://user:admin123@cluster0.jvfu57r.mongodb.net/?retryWrites=true&w=majority"
o. export DB_NAME="Crypto"
p. export PORT=3000
q. export IS_AES=true
r. export CIPHER_KEY="kovgywjqwjlfxndwvxlscdfhhqyzyoab"
s. export IV_KEY="labncrpoqlyhxtqc"
t. export CLIENT_KEY="UTBOMmJBLVlRZ1dJODgwN2xNNTVHUGpOQm9Ycmh2c2FSRWNrLTZXTG4waw=="
```

3. API

All of the API Postman Collections can be seen here: https://api.postman.com/collections/23989743-f8193b9b-2df5-421c-89f7-0aef91395673?access_key=PMAT-01HDF0W73EA7Q6PXAWR0E938BX

2.1 API environment

a. {{client_key}}

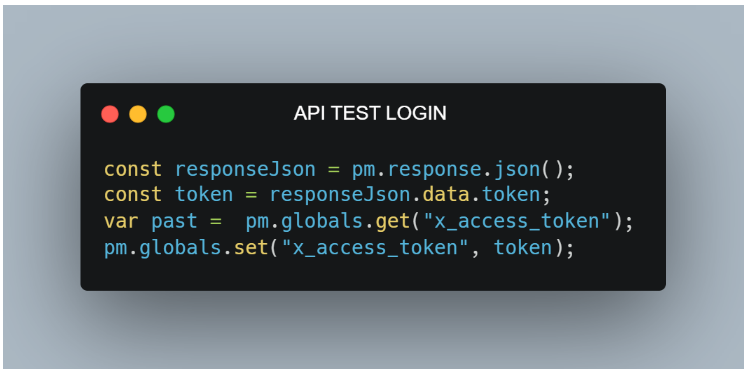
1. Function: Is used to validate the client key. Just only several key from FrontEnd that able to access the backend server. This is

        used to protect the application from unauthorized applications. This key was already defined before

    2. value: UTBOMmJBLVlRZ1dJODgwN2xNNTVHUGpOQm9Ycmh2c2FSRWNrLTZXTG4waw==

b. {{api_key}}
    1. Function: Is used to validate the application that run the front end. This key was already defined before
    2. value:
        a. Andoid: eyJpZCI6IjciLCJuYW1lIjoic2Nvb3Bfd2ViX2FwcHMifQ
        b. Apple:....
        c. Web:.....

c. {{x_access_token}}
    1. Function: Is used to validate token login
    2. The token set when FE hit login API
    3. Test Function:



```
const responseJson = pm.response.json();
const token = responseJson.data.token;
var past =  pm.globals.get("x_access_token");
pm.globals.set("x_access_token", token);
```

d. {{server}}
    1. Function: Is used to define the server port
    2. Value: localhost:3000/

2.2 API Collection

a. Register API
    1. Method: POST
    2. Header:

        a. Authorization
        b. api-key
    3. Body:
        a. email
        b. password (if IS_AES environment in back end server is on, then password is encrypted text, else just normal password)
        c. name

```
// if IS_AES environment is on
{
    "email": "erlandoDomin1ico251125@gmail.com",
    "password" :"09da634d49e534de0520a0841e06e8b2",
    "name":"Erlando12"
}

// if IS_AES environment is off
{
    "email": "erlandoDomin1ico251125@gmail.com",
    "password" :"ourSecret123",
    "name":"Erlando12"
}
```

    4. cURL:

```
        curl --location --globoff '{{server}}user/register' \
        --header                                                    'Authorization:
UTBOMmJBLVlRZ1dJODgwN2xNNTVHUGpOQm9Ycmh2c2FSRWNrLTZXTG4waw==' \
        --header 'api-key: eyJpZCI6IjciLCJuYW1lIjoic2Nvb3Bfd2ViX2FwcHMifQ' \
        --header 'Content-Type: application/json' \
        --data '{
                        "email": "erlandoDomin1ico251125@gmail.com",
                        "password" :"ourSecret123",
                        "name":"Erlando12"
        }'
```

b. Login API

    1. Method: POST
    2. Header:
        a. Authorization

   b. Api-key

3. Body
   a. email
   d. password (if IS_AES environment in back end server is on, then password is encrypted text, else just normal password)

```
// if IS_AES environment is on
{
    "email": "erlandoDomin1ico251125@gmail.com",
    "password" :"09da634d49e534de0520a0841e06e8b2",


}

// if IS_AES environment is off
{
    "email": "erlandoDomin1ico251125@gmail.com",
    "password" :"ourSecret123",
}
```

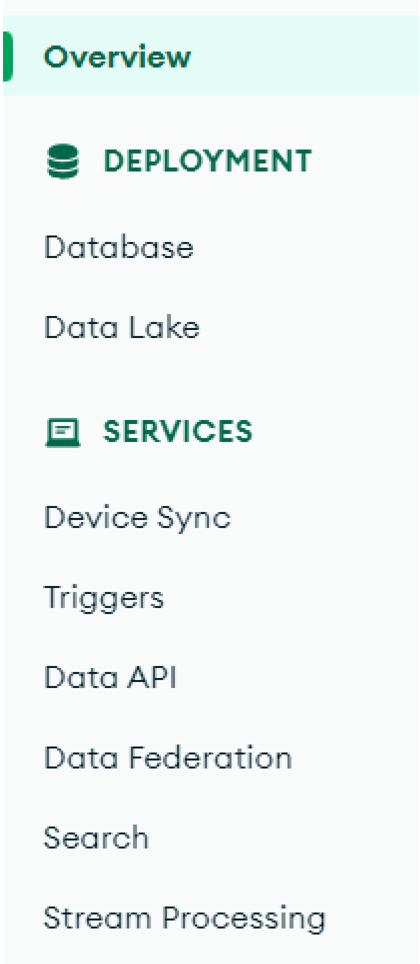
c. cURL:

```
        curl --location --globoff '{{server}}login/action' \
        --header                                                                                'Authorization:
UTBOMmJBLVlRZ1dJODgwN2xNNTVHUGpOQm9Ycmh2c2FSRWNrLTZXTG4waw==' \
        --header 'api-key: eyJpZCI6IjciLCJuYW1lIjoic2Nvb3Bfd2ViX2FwcHMifQ' \
        --header 'Content-Type: application/json' \
        --data '{
            "email": "erlandoDomin1ico251125@gmail.com",
            "password" :"09da634d49e534de0520a0841e06e8b2",
        }'
```

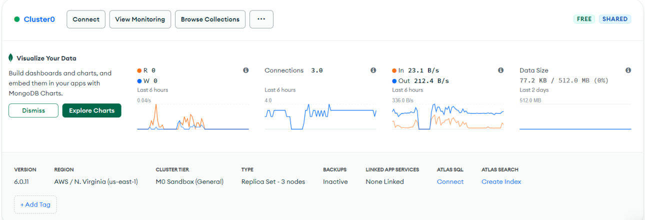
d. Show Dashboard Coin API
e. Method: GET
f. Header:
   i. Authorization
   ii. Api-key
   iii. x-access-token
g. End Point: {{server}}getData
h. cURL:

```
        curl --location --globoff '{{server}}getData' \
        --header                                                    'Authorization:
UTBOMmJBLVlRZ1dJODgwN2xNNTVHUGpOQm9Ycmh2c2FSRWNrLTZXTG4waw==' \
        --header 'api-key: eyJpZCI6IjciLCJuYW1lIjoic2Nvb3Bfd2ViX2FwcHMifQ' \
        --header                                                    'x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1MzZiNWMwYjgyMjVhZGIzNDM5ODE2ZiIsIm
VtYWlsIjoiZXJsYW5kb0RvbWluMWljbzI1MTEyNSIsIm5hbWUiOiJFcmxhbmRvMTIiLCJpYXQiOjE2OT
gwODQzMDgsImV4cCI6MTY5ODA4NzkwOH0.Dvk--VLmb0vem2HvcKnayCdkb7QzO6QyDFSMkgsTP
0M' \
        --data ''
```

4. **Database**
    a. **Using Atlas MongoDB**
    b. **Tutorial:**
        i. Create a free Mongo DB Atlas Account
        ii. Click on Database tab

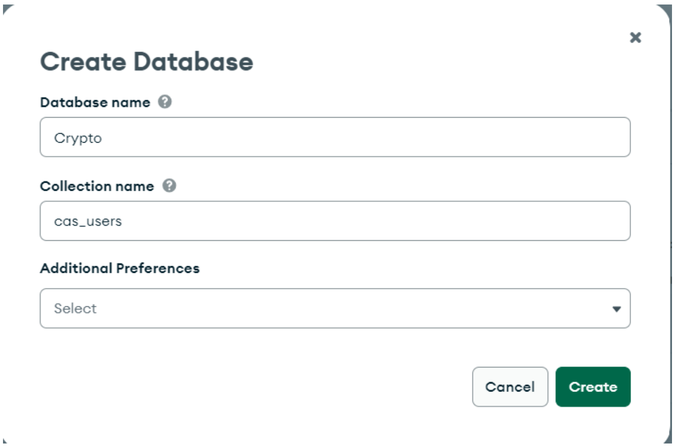


iii. Click on Browse Connection



VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL	ATLAS SEARCH
6.0.11	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Connect	Create Index

iv. Create Database named “**Crypto**” and collections “**cas_users**”



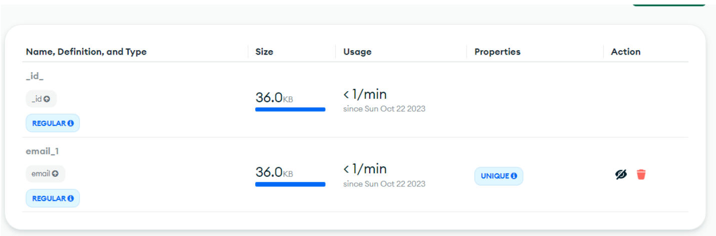
Create Database

Database name: Crypto

Collection name: cas_users

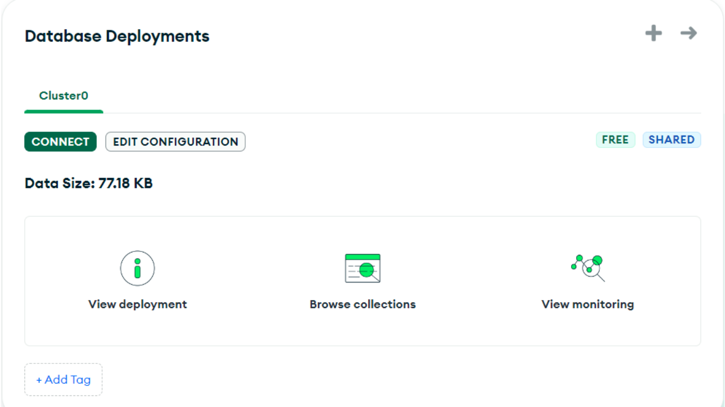
Additional Preferences: Select

v. Create indexes for email column (Primary Key)



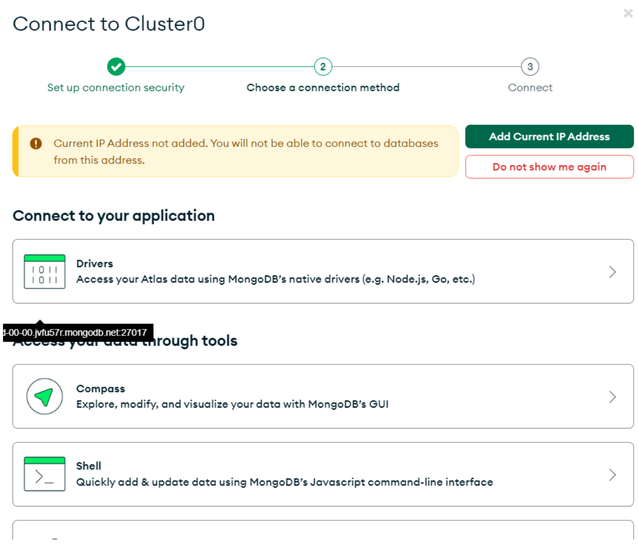
Name, Definition, and Type	Size	Usage	Properties	Action
_id_ _id REGULAR	36.0KB	< 1/min since Sun Oct 22 2023		
email_1 email REGULAR	36.0KB	< 1/min since Sun Oct 22 2023	UNIQUE	

vi. Back to homepage again, and click “connect”



Data Size: 77.18 KB

vii. Click Drivers



Connect to Cluster0

Current IP Address not added. You will not be able to connect to databases from this address.

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

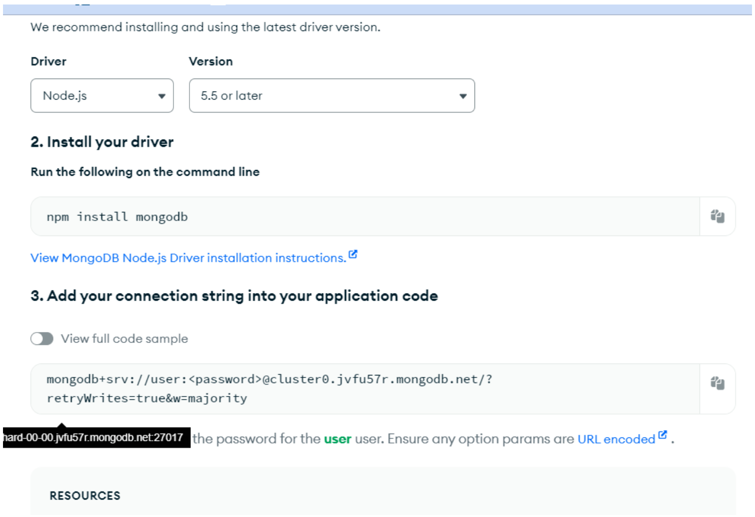
-00-00.jvfu57r.mongodb.net:27017

Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

viii. Choose Node JS Driver and copy the url connection string into DB_URI in environment



We recommend installing and using the latest driver version.

Driver: Node.js

Version: 5.5 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

3. Add your connection string into your application code

○ View full code sample

```
mongodb+srv://user:<password>@cluster0.jvfu57r.mongodb.net/?
retryWrites=true&w=majority
```

hard-00-00.jvfu57r.mongodb.net:27017 the password for the **user** user. Ensure any option params are URL encoded .

RESOURCES

5. Setting up Project

a. Clone Project from Github/ Bitbucket

i. Github Project (Open Access)

https://github.com/edhbs31/CryptoMarketAPI

ii. Bitbucket Project (Private Access)

https://bitbucket.org/cryptocoinerlando/cryptomarketapi

b. To start the server, for Linux users, just run:

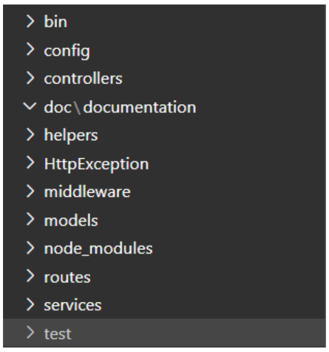
i. npm install

ii. source .env.local

iii. npm start

6. System Design

Using Factory Pattern Design (https://refactoring.guru/design-patterns/factory-method), our project divided into several folder



```
> bin
> config
> controllers
v doc\documentation
> helpers
> HttpException
> middleware
> models
> node_modules
> routes
> services
> test
```

a. bin

This folder is used for setting server configuration. including port and starting server (www.js)

b. config

This folder is used to setting configuration, including database configuration (config.js)

c. controllers

This folder is used to setting the controller inheritances (child end point)

d. doc

This folder is used to keep the documentation, including the uml process and pdf documentation

e. helpers

Our project is implementing SOLID (single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle). We are trying to make classes that can be used in every controllers (Single Responsibility)

f. HttpException

This folder is used to handle and make default return for each http exception process

g. middleware

This folder is used to handle middleware process

h. **models**

This folder is used to keep database model (class)

i. **node_modules**

This folder is used to keep node modules that already installed

j. **routes**

This folder is used to become main setting for routing endpoint

k. **service**

This folder is used to handle services for each controller

l. **test**

This folder is used to make the unit test program using JEST

7. **Database Model**

Is located on model folders in cas_users.js

Table Name: cas_users

Contains:

a. email: String (Must be unique and become composite PK)
b. name: String
c. password: String
d. salt: String
e. createdAt: date and default current date (Date.now)

8. **Logic Workflow**

a. **Limiter**

i. The limiter is to prevent users from sending requests multiple times each IP to 100 requests per windowMs for 1 second request

b. **MiddlewareClient (Client.js)**

i. MiddlewareClient is used to check if the client is valid or not
ii. It used to prevent another server inject our server
iii. MiddlewareClient is used in Login, Register and GetData.
iv. Logic:

1. Get authorization key from Headers
2. If it exists, then check whether the key is valid or not

3. The authorization key should be an encoded base64 string
4. If the encoded key equals with the client key env, then the user is authorized

c. **Middleware (Auth.js)**

i. Middleware is used to check if the user has already logged in or not

ii. Middleware is used to prevent unathorized user who haven't logged in.

iii. Logic:

1. Get the JWT token in ‘x-access-token’ from request header.
2. Verify if the jwt token is expired or not

d. **Register (Register Controllers)**

i. Get Data email, password and name from request body

1. Notes: If env IS_AES is true, then password must be sent encrypted; if env IS_AES is false, then you can send the original password

ii. Checking the email to see if the string is in a valid format

iii. Check the database if email not registered

iv. Declare and initiate CryptoHelpers

v. Check if env IS_AES is true or false

1. Notes: If IS_AES equals true, then password must be decrypted into original password

vi. Generate random salt password

vii. Encrypt original password with salt data using SHA 256 method (1 way encryption)

viii. Generate a salt using the bcrypt.genSaltSync() function with 10 rounds of hashing.

ix. hash the synchronous encrypted data generated using the bcrypt.hashSync() function and the gensalt salt.

x. create user into database using UserServices

xi. Return http success with status code 200

e. **Login (LoginControllers)**

i. Get email and password from request body
    1. Notes: If env IS_AES is true, then password must be sent encrypted; if env IS_AES is false, then you can send the original password

ii. Find email if exist in database or not

iii. Declare CryptoHelpers

iv. Check if env IS_AES is true or false
    1. Notes: If IS_AES equals true, then password from input must be decrypted into original password

v. Concate original password and salt string from database

vi. Encrypt the combination with SHA-256 algorithm (1- way decrypt)

vii. Compare passwords using asynchronous bcrypt. CompareSync()

viii. If the result is true, then Declare Token Helpers

ix. Set the Token Constructor with data object from database

x. Build token using JWT with data ID, email and name, and default expiration time is 1 hour

xi. Build a refresh token using JWT with data token, user id and email. In any case, if the token already expired, you can use refresh token for authentication

xii. if all processes are successful, return Http Success (status code:200) with data username, token and refresh token

f. Show Coin Market API (Dashboard Controllers)

i. Using Get Method

ii. Is used to get and find by name from the coin market API

iii. Logic:
    1. Call API CoinMarket using Axios
        a. Notes: The API Link and header CMC_PRO_API_KEY is using env
    2. Define data from API result
    3. Check if there is query that imputed
        a. Notes: If the query exists, then
            i. loop every single data

ii. then check which name which includes the query string

iii. If the query includes a string name, push into new array

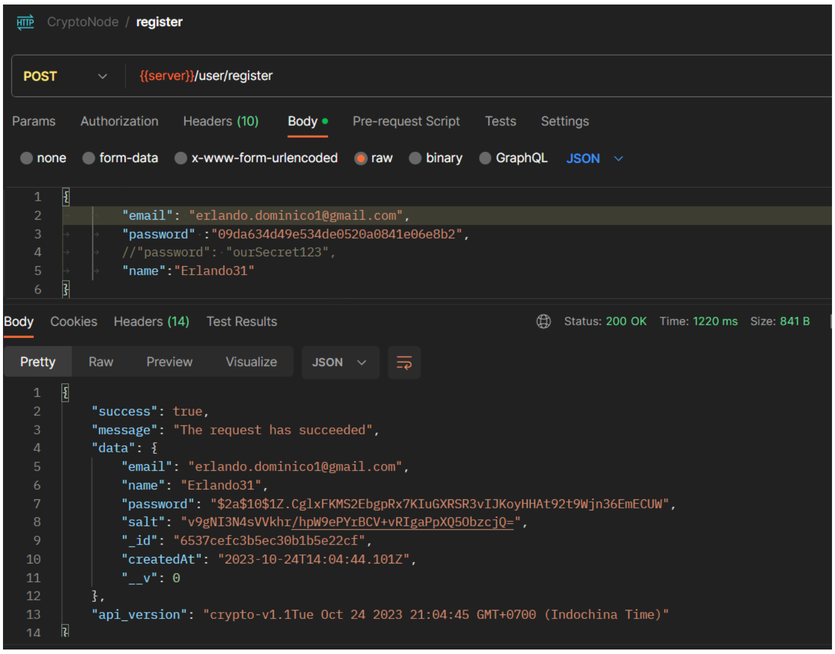
4. If success, return http success (status code 200)

9. Result API

a. Register API

i. Success Case

Login with IS_AES = true



CryptoNode / register

POST {{server}}/user/register

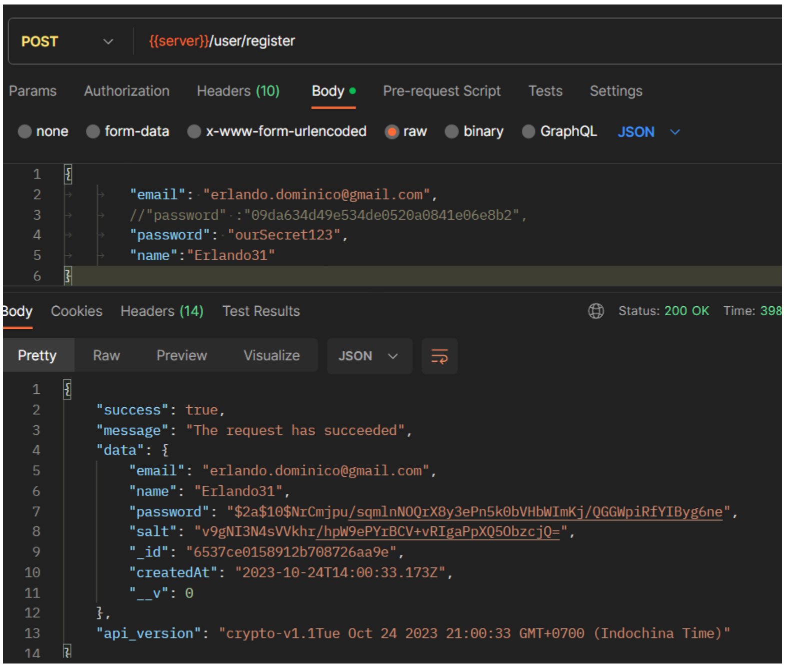
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "email": "erlando.dominico1@gmail.com",
    "password" :"09da634d49e534de0520a0841e06e8b2",
    //"password": "ourSecret123",
    "name":"Erlando31"
}
```

Status: 200 OK Time: 1220 ms Size: 841 B

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": {
        "email": "erlando.dominico1@gmail.com",
        "name": "Erlando31",
        "password": "$2a$10$1Z.CglxFKMS2EbgpRx7KIuGXRSR3vIJKoyHHAt92t9Wjn36EmECUW",
        "salt": "v9gNI3N4sVVkhr/hpW9ePYrBCV+vRIgaPpXQ5ObzcjQ=",
        "_id": "6537cefc3b5ec30b1b5e22cf",
        "createdAt": "2023-10-24T14:04:44.101Z",
        "__v": 0
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 21:04:45 GMT+0700 (Indochina Time)"
}
```

Login with IS_AES = false

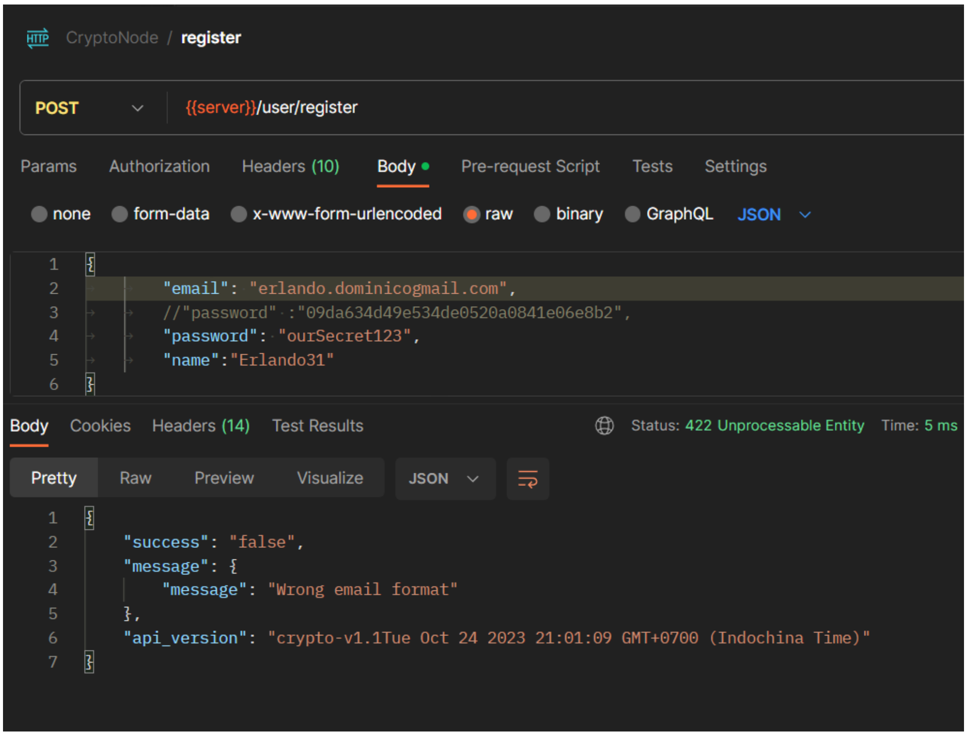


```
{
        "email": "erlando.dominico@gmail.com",
        //"password" :"09da634d49e534de0520a0841e06e8b2",
        "password": "ourSecret123",
        "name":"Erlando31"
}
```

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": {
        "email": "erlando.dominico@gmail.com",
        "name": "Erlando31",
        "password": "$2a$10$NrCmjpu/sqmlnNOQrX8y3ePn5k0bVHbWImKj/QGGWpiRfYIByg6ne",
        "salt": "v9gNI3N4sVVkhr/hpW9ePYrBCV+vRIgaPpXQ5ObzcjQ=",
        "_id": "6537ce0158912b708726aa9e",
        "createdAt": "2023-10-24T14:00:33.173Z",
        "__v": 0
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 21:00:33 GMT+0700 (Indochina Time)"
}
```

ii. Failed Case

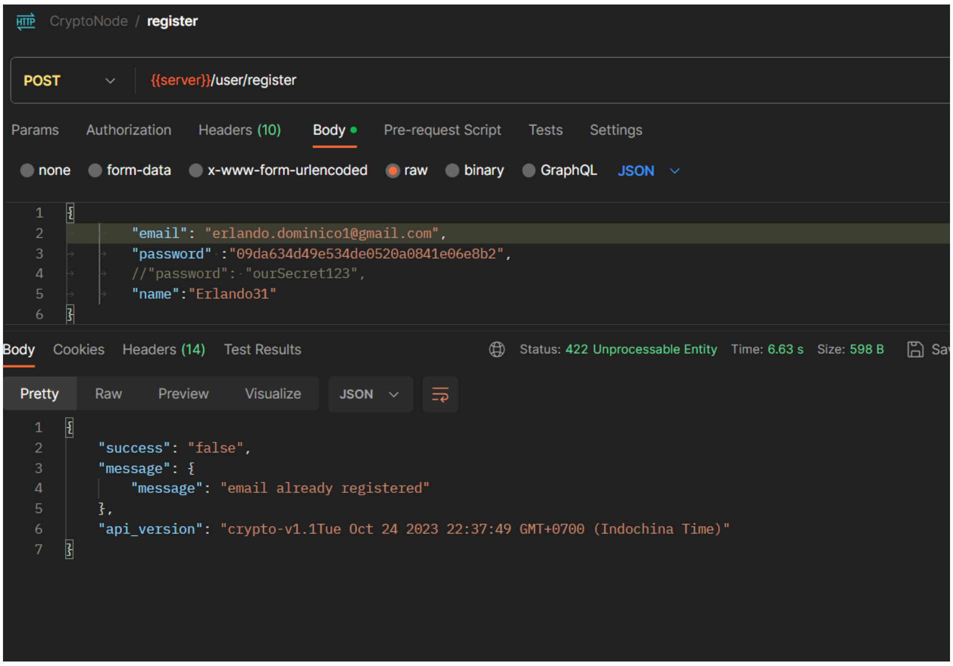
Wrong email format



```
{
        "email": "erlando.dominicogmail.com",
        //"password" :"09da634d49e534de0520a0841e06e8b2",
        "password": "ourSecret123",
        "name":"Erlando31"
}
```

```
{
    "success": "false",
    "message": {
        "message": "Wrong email format"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 21:01:09 GMT+0700 (Indochina Time)"
}
```

Email Already Registered



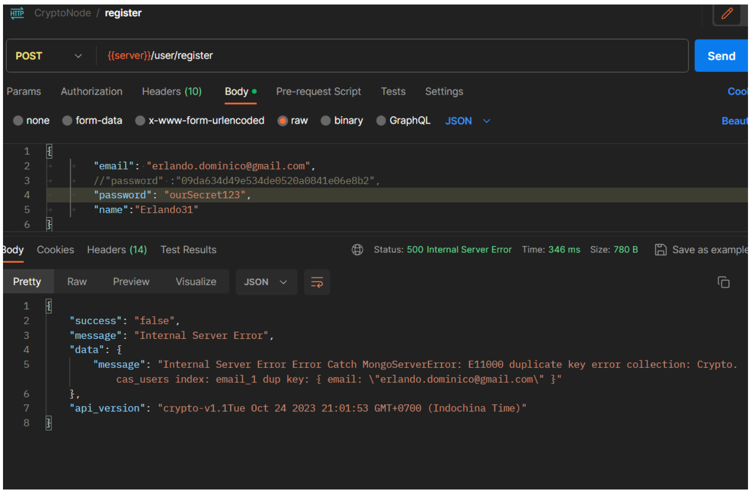
{{server}}/user/register

```
{
    "email": "erlando.dominico1@gmail.com",
    "password" :"09da634d49e534de0520a0841e06e8b2",
    //"password": "ourSecret123",
    "name":"Erlando31"
}
```

Status: 422 Unprocessable Entity Time: 6.63 s Size: 598 B

```
{
  "success": "false",
  "message": {
      "message": "email already registered"
  },
  "api_version": "crypto-v1.1Tue Oct 24 2023 22:37:49 GMT+0700 (Indochina Time)"
}
```

Duplicate Primary Key



{{server}}/user/register

```
{
    "email": "erlando.dominico@gmail.com",
    //"password" :"09da634d49e534de0520a0841e06e8b2",
    "password": "ourSecret123",
    "name":"Erlando31"
}
```

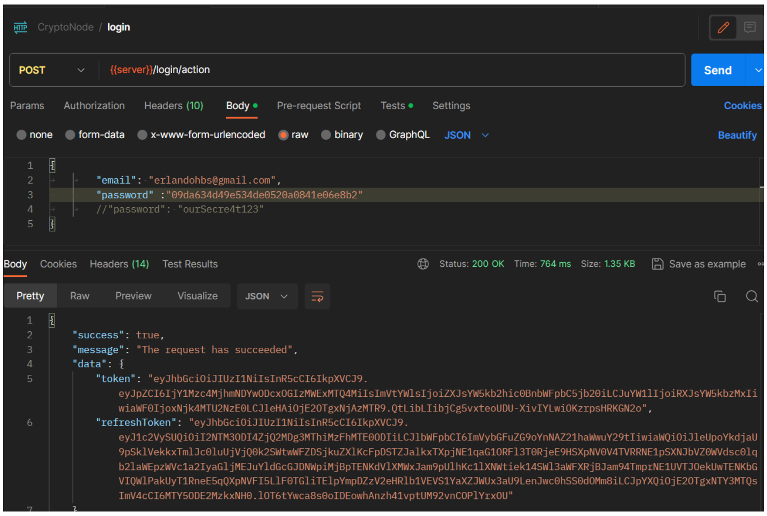
Status: 500 Internal Server Error Time: 346 ms Size: 780 B

```
{
  "success": "false",
  "message": "Internal Server Error",
  "data": {
      "message": "Internal Server Error Error Catch MongoServerError: E11000 duplicate key error collection: Crypto.
          cas_users index: email_1 dup key: { email: \"erlando.dominico@gmail.com\" }"
  },
  "api_version": "crypto-v1.1Tue Oct 24 2023 21:01:53 GMT+0700 (Indochina Time)"
}
```

b. Login API

i. Success Case

Login With IS_AES = true



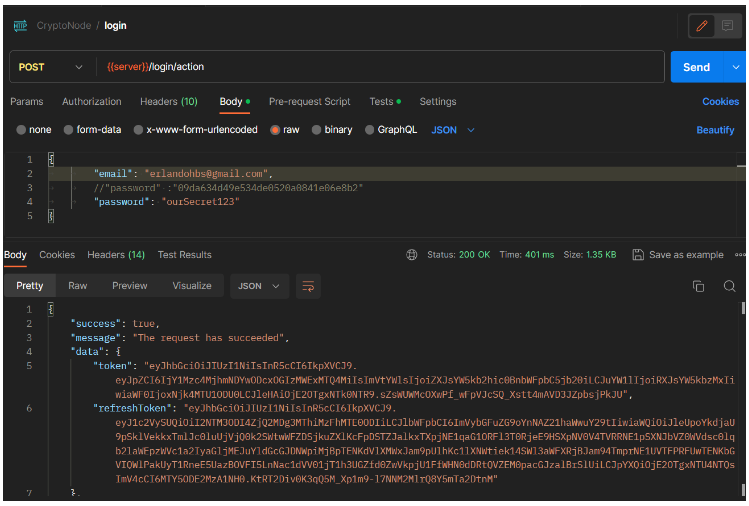
POST {{server}}/login/action

```
{
    "email": "erlandohbs@gmail.com",
    "password" :"09da634d49e534de0520a0841e06e8b2"
    //"password": "ourSecre4t123"
}
```

Status: 200 OK Time: 764 ms Size: 1.35 KB

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJpZCI6IjY1Mzc4MjhmNDYwODcxOGIzMWExMTQ4MiIsImVtYWlsIjoiZXJsYW5kb2hic0BnbWFpbC5jb20iLCJuYW1lIjoiRXJsYW5kbzMxIi
            wiaWF0IjoxNjk4MTU2NzE0LCJleHAiOjE2OTgxNjAzMTR9.QtLibLIibjCg5vxteoUDU-XivIYLwiOKzrpsHRKGN2o",
        "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJ1c2VySUQiOiI2NTM3ODI4ZjQ2MDg3MThiMzFhMTE0ODIiLCJlbWFpbCI6ImVybGFuZG9oYnNAZ21haWwuY29tIiwiaWQiOiJleUpoYkdjaU
            9pSklVekkxTmlJc0luUjVjQ0k2SWtwWFZDSjkuZXlKcFpDSTZJalkxTXpjNE1qaG1ORFl3T0RjeE9HSXpNV0V4TVRRNE1pSXNJbVZ0WVdsc0lq
            b2laWEpzWVc1a2IyaGljMEJuYldGcGJDNWpiMjBpTENKdVlXMWxJam9pUlhKc1lXNWtiek14SWl3aWFXRjBJam94TmprNE1UVTJOekUwTENKbG
            VIQWlPakUyT1RneE5qQXpNVFI5LlF0TGliTElpYmpDZzV2eHRlb1VEVS1YaXZJWUx3aU9LenJwc0hSS0dOMm8iLCJpYXQiOjE2OTgxNTY3MTQs
            ImV4cCI6MTY5ODE2MzkxNH0.lOT6tYwca8s0oIDEowhAnzh41vptUM92vnCOPlYrxOU"
    }
```

Login With IS_AES = false



POST {{server}}/login/action

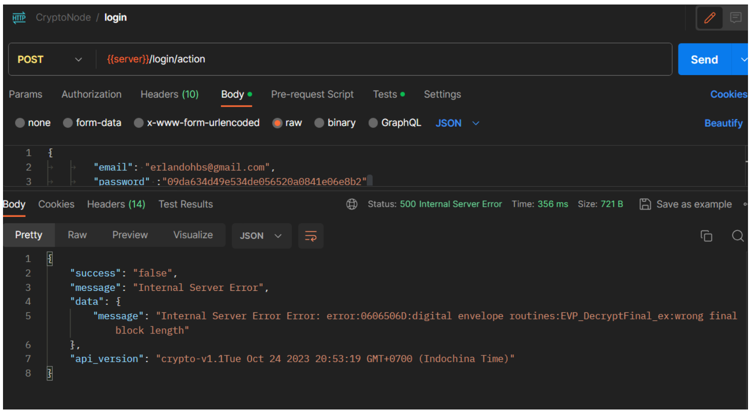
```
{
    "email": "erlandohbs@gmail.com",
    //"password" :"09da634d49e534de0520a0841e06e8b2"
    "password": "ourSecret123"
}
```

Status: 200 OK Time: 401 ms Size: 1.35 KB

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJpZCI6IjY1Mzc4MjhmNDYwODcxOGIzMWExMTQ4MiIsImVtYWlsIjoiZXJsYW5kb2hic0BnbWFpbC5jb20iLCJuYW1lIjoiRXJsYW5kbzMxIi
            wiaWF0IjoxNjk4MTU1ODU0LCJleHAiOjE2OTgxNTk0NTR9.sZsWUWMcOXwPf_wFpVJcSQ_Xstt4mAVD3JZpbsjPkJU",
        "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJ1c2VySUQiOiI2NTM3ODI4ZjQ2MDg3MThiMzFhMTE0ODIiLCJlbWFpbCI6ImVybGFuZG9oYnNAZ21haWwuY29tIiwiaWQiOiJleUpoYkdjaU
            9pSklVekkxTmlJc0luUjVjQ0k2SWtwWFZDSjkuZXlKcFpDSTZJalkxTXpjNE1qaG1ORFl3T0RjeE9HSXpNV0V4TVRRNE1pSXNJbVZ0WVdsc0lq
            b2laWEpzWVc1a2IyaGljMEJuYldGcGJDNWpiMjBpTENKdVlXMWxJam9pUlhKc1lXNWtiek14SWl3aWFXRjBJam94TmprNE1UVTFPRFUwTENKbG
            VIQWlPakUyT1RneE5UazBOVFI5LnNac1dVV01jT1h3UGZfd0ZwVkpjU1FfWHN0dDRtQVZEM0pacGJzalBrSlUiLCJpYXQiOjE2OTgxNTU4NTQs
            ImV4cCI6MTY5ODE2MzA1NH0.KtRT2Div0K3qQ5M_Xp1m9-l7NNM2MlrQ8Y5mTa2DtnM"
    },
```

ii. Failed Case

Login failed decrypt password



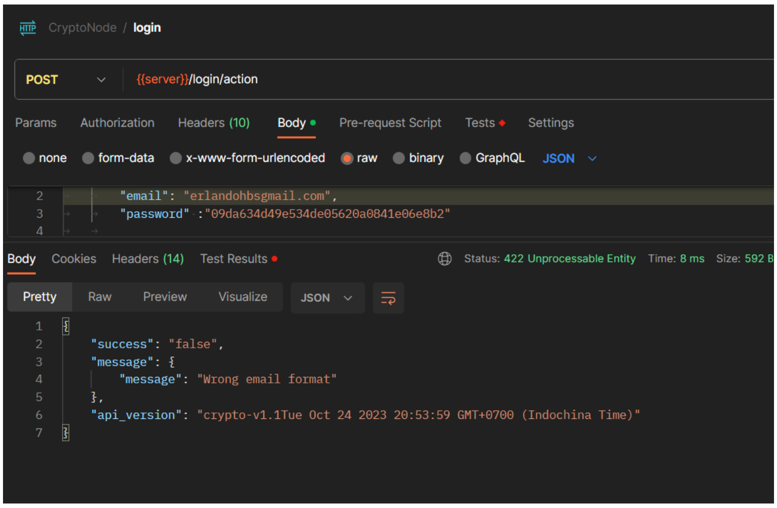
{{server}}/login/action

```
{
    "email": "erlandohbs@gmail.com",
    "password" :"09da634d49e534de056520a0841e06e8b2"
```

Status: 500 Internal Server Error Time: 356 ms Size: 721 B

```
{
    "success": "false",
    "message": "Internal Server Error",
    "data": {
        "message": "Internal Server Error Error: error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final
            block length"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 20:53:19 GMT+0700 (Indochina Time)"
}
```

2. Wrong email format



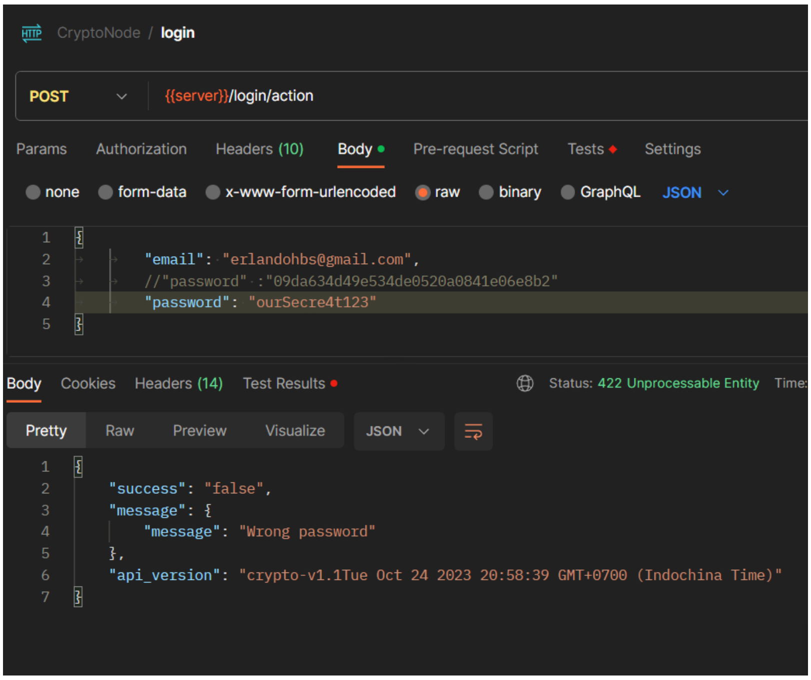
{{server}}/login/action

```
    "email": "erlandohbsgmail.com",
    "password" :"09da634d49e534de05620a0841e06e8b2"
```

Status: 422 Unprocessable Entity Time: 8 ms Size: 592 B

```
{
    "success": "false",
    "message": {
        "message": "Wrong email format"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 20:53:59 GMT+0700 (Indochina Time)"
}
```

3. Wrong password



POST {{server}}/login/action

```
{
    "email": "erlandohbs@gmail.com",
    //"password" :"09da634d49e534de0520a0841e06e8b2"
    "password": "ourSecre4t123"
}
```

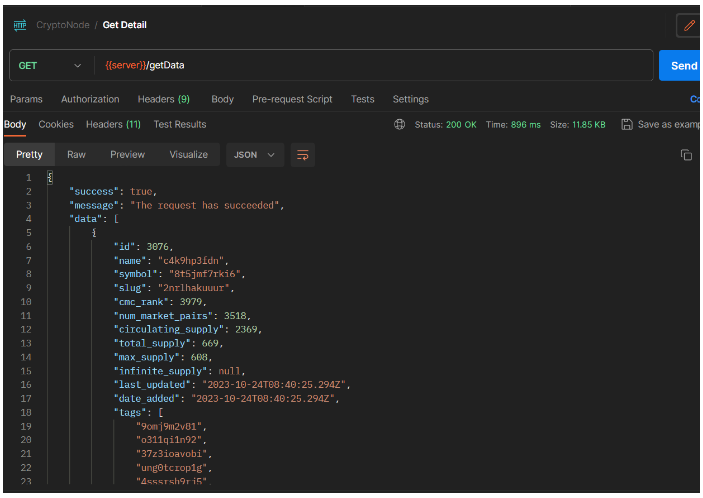
Status: 422 Unprocessable Entity

```
{
    "success": "false",
    "message": {
        "message": "Wrong password"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 20:58:39 GMT+0700 (Indochina Time)"
}
```

c. Show Coin Market API

i. Success Case

a. Without Query

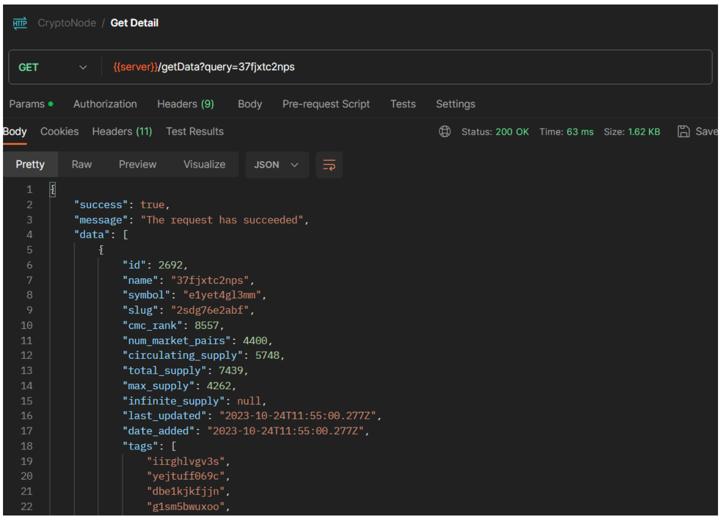


GET {{server}}/getData

Status: 200 OK Time: 896 ms Size: 11.85 KB

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": [
        {
            "id": 3076,
            "name": "c4k9hp3fdn",
            "symbol": "8t5jmf7rki6",
            "slug": "2nrlhakuuur",
            "cmc_rank": 3979,
            "num_market_pairs": 3518,
            "circulating_supply": 2369,
            "total_supply": 669,
            "max_supply": 608,
            "infinite_supply": null,
            "last_updated": "2023-10-24T08:40:25.294Z",
            "date_added": "2023-10-24T08:40:25.294Z",
            "tags": [
                "9omj9m2v81",
                "o311qi1n92",
                "37z3ioavobi",
                "ung0tcrop1g",
                "4sssrsh9ri5",
```

b. With Query



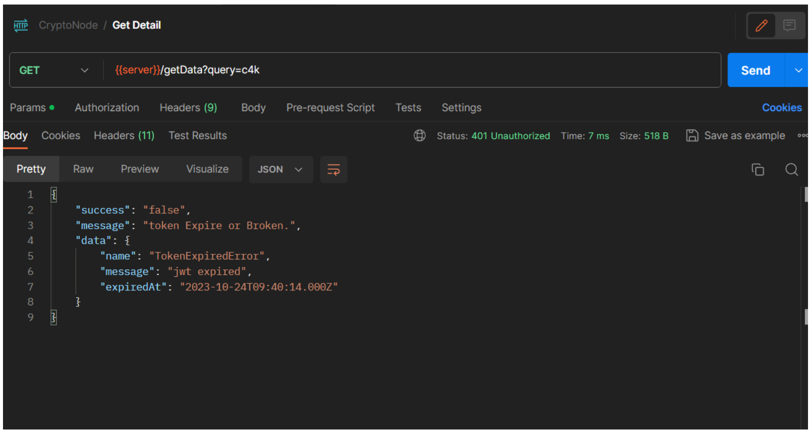
GET {{server}}/getData?query=37fjxtc2nps

Status: 200 OK Time: 63 ms Size: 1.62 KB

```
{
    "success": true,
    "message": "The request has succeeded",
    "data": [
        {
            "id": 2692,
            "name": "37fjxtc2nps",
            "symbol": "e1yet4gl3mm",
            "slug": "2sdg76e2abf",
            "cmc_rank": 8557,
            "num_market_pairs": 4400,
            "circulating_supply": 5748,
            "total_supply": 7439,
            "max_supply": 4262,
            "infinite_supply": null,
            "last_updated": "2023-10-24T11:55:00.277Z",
            "date_added": "2023-10-24T11:55:00.277Z",
            "tags": [
                "iirghlvgv3s",
                "yejtuff069c",
                "dbe1kjkfjjn",
                "g1sm5bwuxoo",
```

ii. Failed Case

1. Expired Token Access

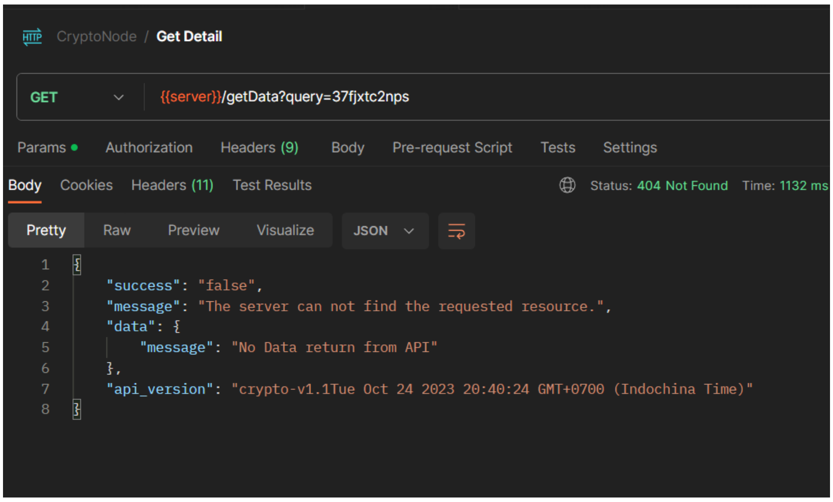


GET {{server}}/getData?query=c4k

Status: 401 Unauthorized Time: 7 ms Size: 518 B

```
{
    "success": "false",
    "message": "token Expire or Broken.",
    "data": {
        "name": "TokenExpiredError",
        "message": "jwt expired",
        "expiredAt": "2023-10-24T09:40:14.000Z"
    }
}
```

2. No data return

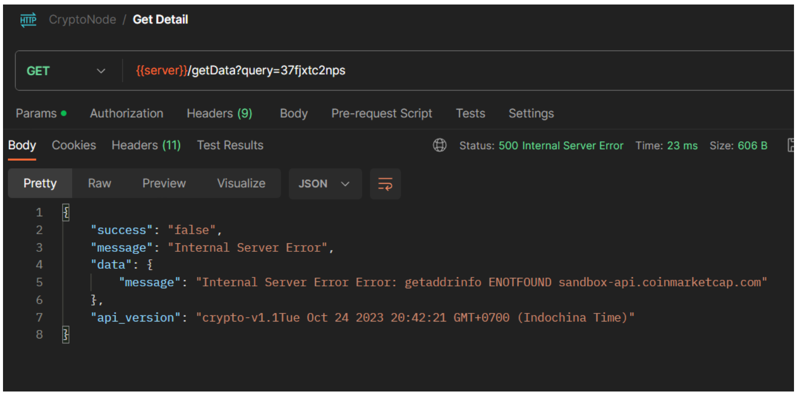


GET {{server}}/getData?query=37fjxtc2nps

Status: 404 Not Found Time: 1132 ms

```
{
    "success": "false",
    "message": "The server can not find the requested resource.",
    "data": {
        "message": "No Data return from API"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 20:40:24 GMT+0700 (Indochina Time)"
}
```

3. Internet Off



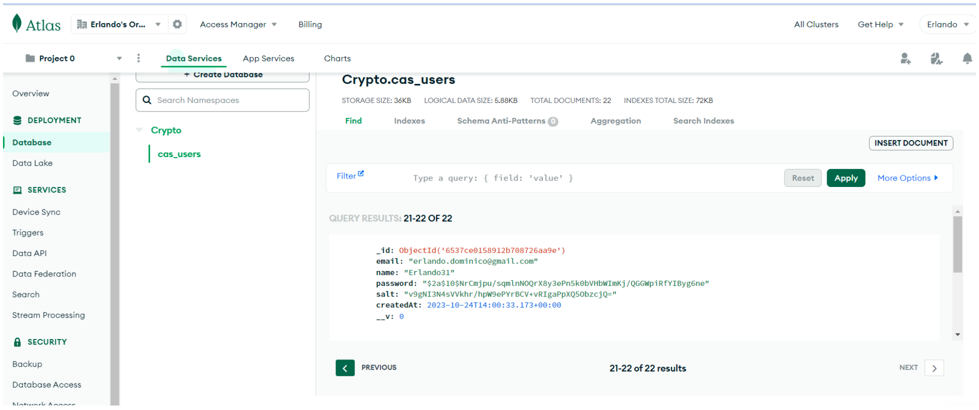
GET {{server}}/getData?query=37fjxtc2nps

Status: 500 Internal Server Error Time: 23 ms Size: 606 B

```
{
    "success": "false",
    "message": "Internal Server Error",
    "data": {
        "message": "Internal Server Error Error: getaddrinfo ENOTFOUND sandbox-api.coinmarketcap.com"
    },
    "api_version": "crypto-v1.1Tue Oct 24 2023 20:42:21 GMT+0700 (Indochina Time)"
}
```

d. Database

Success Created



Crypto.cas_users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 5.88KB TOTAL DOCUMENTS: 22 INDEXES TOTAL SIZE: 72KB

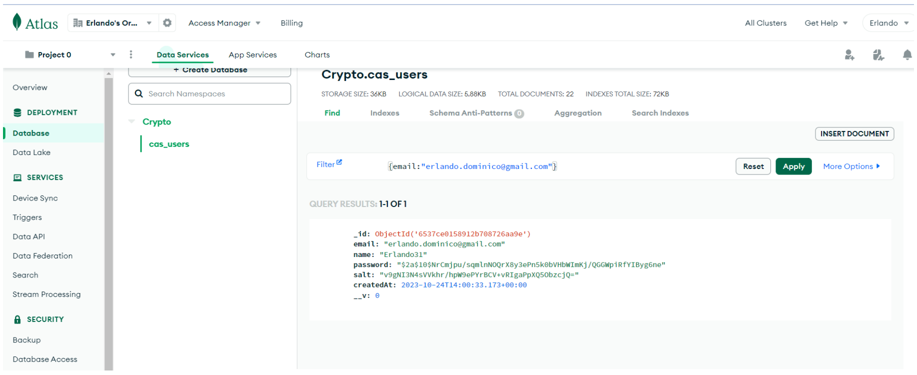
Type a query: { field: 'value' }

QUERY RESULTS: 21-22 OF 22

```
_id: ObjectId('6537ce0158912b708726aa9e')
email: "erlando.dominico@gmail.com"
name: "Erlando31"
password: "$2a$10$NrCmjpu/sqmlnNOQrX8y3ePn5k0bVHbWImKj/QGGWpiRfYIByg6ne"
salt: "v9gNI3N4sVVkhr/hpW9ePYrBCV+vRIgaPpXQ5ObzcjQ="
createdAt: 2023-10-24T14:00:33.173+00:00
__v: 0
```

21-22 of 22 results

Filter by email



Crypto.cas_users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 5.88KB TOTAL DOCUMENTS: 22 INDEXES TOTAL SIZE: 72KB

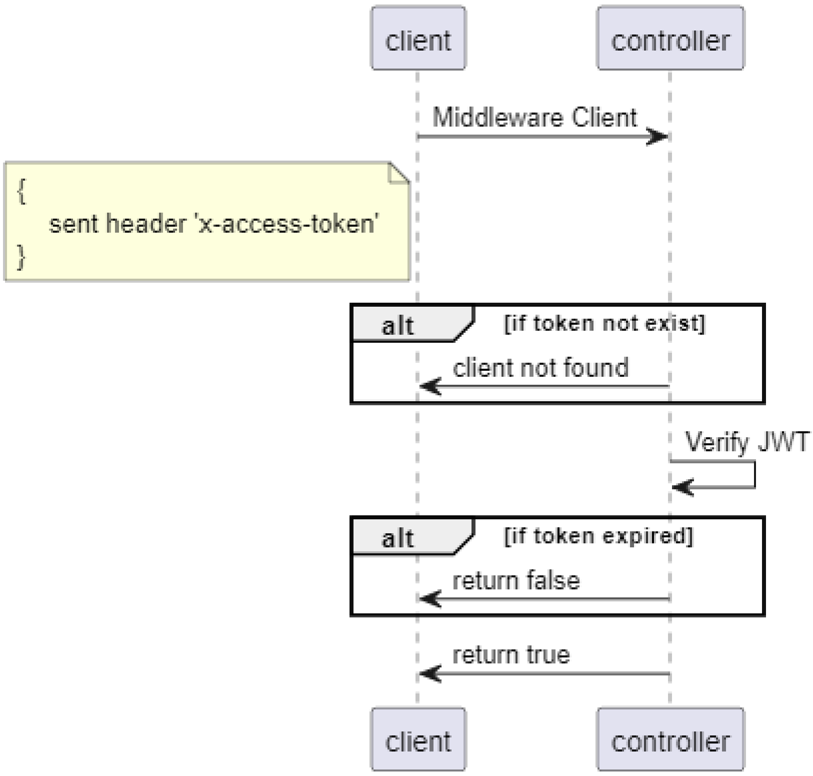
{email:"erlando.dominico@gmail.com"}

QUERY RESULTS: 1-1 OF 1

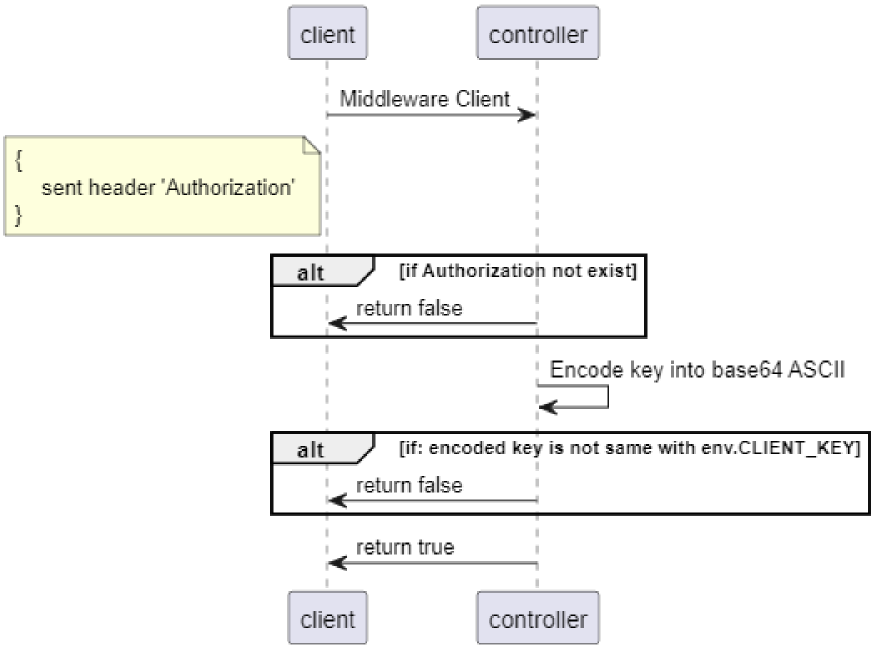
```
_id: ObjectId('6537ce0158912b708726aa9e')
email: "erlando.dominico@gmail.com"
name: "Erlando31"
password: "$2a$10$NrCmjpu/sqmlnNOQrX8y3ePn5k0bVHbWImKj/QGGWpiRfYIByg6ne"
salt: "v9gNI3N4sVVkhr/hpW9ePYrBCV+vRIgaPpXQ5ObzcjQ="
createdAt: 2023-10-24T14:00:33.173+00:00
__v: 0
```

10. UML PLANT (Please kindly look on doc/uml file)

a. Middleware

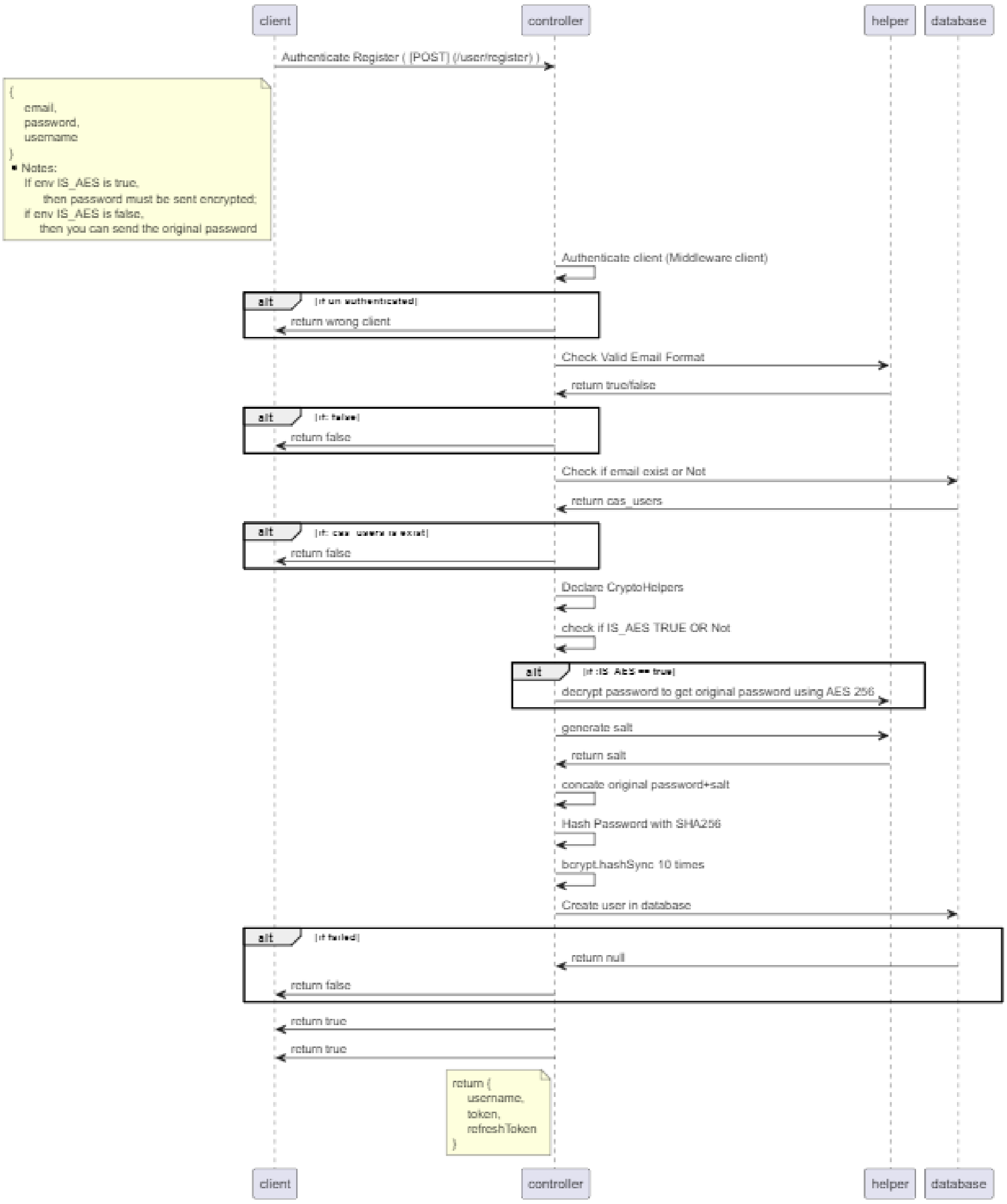


b. Middleware client

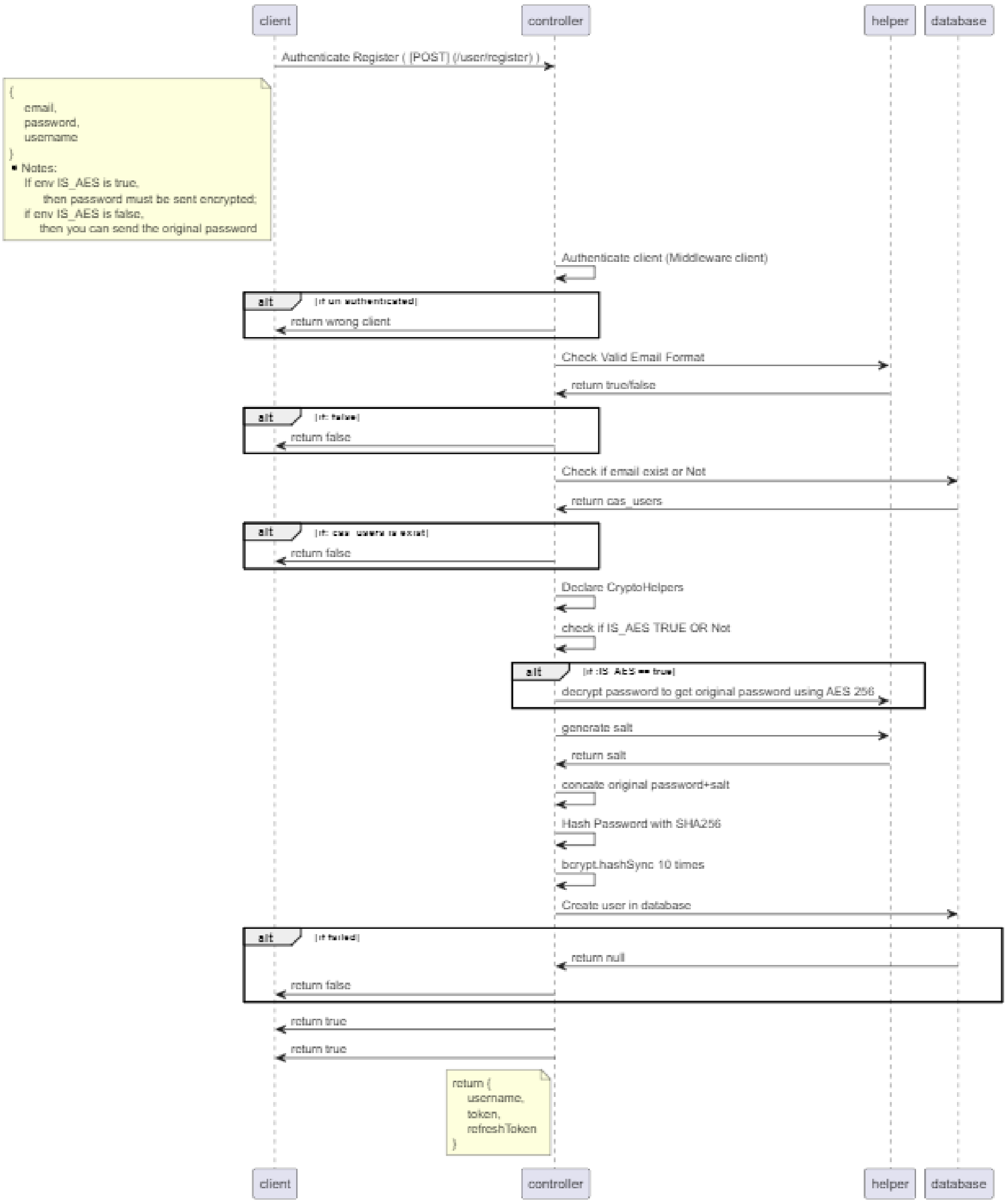


client
controller
Middleware Client
{
sent header 'Authorization'
}
alt
[if Authorization not exist]
return false
Encode key into base64 ASCII
alt
[if: encoded key is not same with env.CLIENT_KEY]
return false
return true
client
controller

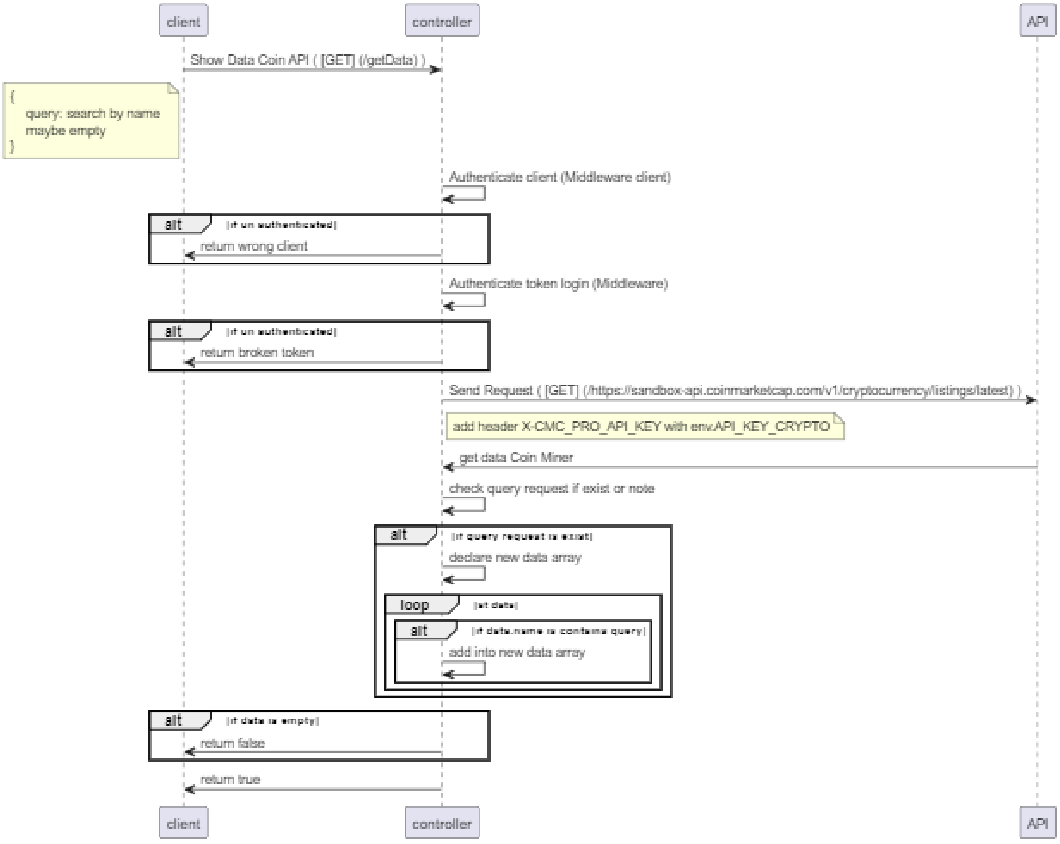
c. Login


client
controller
helper
database
Authenticate Register ([POST] (/user/register))
{
email,
password,
username
}
■ Notes:
If env IS_AES is true,
then password must be sent encrypted;
if env IS_AES is false,
then you can send the original password
Authenticate client (Middleware client)
alt
[if un authenticated]
return wrong client
Check Valid Email Format
return true/false
alt
[if: false]
return false
Check if email exist or Not
return cas_users
alt
[if: cas_users is exist]
return false
Declare CryptoHelpers
check if IS_AES TRUE OR Not
alt
[if :IS_AES == true]
decrypt password to get original password using AES 256
generate salt
return salt
concate original password+salt
Hash Password with SHA256
bcrypt.hashSync 10 times
Create user in database
alt
[if failed]
return null
return false
return true
return true
return (
username,
token,
refreshToken
)
client
controller
helper
database

d. Register


client
controller
helper
database
Authenticate Register ([POST] (/user/register))
{
email,
password,
username
}
• Notes:
If env IS_AES is true,
then password must be sent encrypted;
if env IS_AES is false,
then you can send the original password
Authenticate client (Middleware client)
alt
[if un authenticated]
return wrong client
Check Valid Email Format
return true/false
alt
[if: false]
return false
Check if email exist or Not
return cas_users
alt
[if: cas_users is exist]
return false
Declare CryptoHelpers
check if IS_AES TRUE OR Not
alt
[if :IS_AES == true]
decrypt password to get original password using AES 256
generate salt
return salt
concate original password+salt
Hash Password with SHA256
bcrypt.hashSync 10 times
Create user in database
alt
[if failed]
return null
return false
return true
return true
return {
username,
token,
refreshToken
}
client
controller
helper
database

e. Show Data Coin API


client
controller
API
Show Data Coin API ([GET] (/getData))
{
query: search by name
maybe empty
}
Authenticate client (Middleware client)
alt
[if un authenticated]
return wrong client
Authenticate token login (Middleware)
alt
[if un authenticated]
return broken token
Send Request ([GET] (/https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest))
add header X-CMC_PRO_API_KEY with env.API_KEY_CRYPTO
get data Coin Miner
check query request if exist or note
alt
[if query request is exist]
declare new data array
loop
[at data]
alt
[if data.name is contains query]
add into new data array
alt
[if data is empty]
return false
return true
client
controller
API