VOTEROID SECURE API GATEWAY USING MICROSERVICES ARCHITECTURE

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted by:

Kulanshu Sharma
(2k21/SWE/12)

Under the supervision of

Mr RAHUL

Delhi Technological University, Delhi

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi - 110042

MAY, 2023

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi - 110042

CANDIDATE'S DECLARATION

I, Kulanshu Sharma, Roll No 2k21/SWE/12 student of M.Tech. Software Engineering, hereby declare that the project Dissertation titled "Voteroid Secure API Gateway Using Microservices Architecture" which is submitted by me to the Department of Software Engineering, Delhi Technological University , Delhi in partial fulfillment of the requirement for the award of the degree of the Master in Technology, is original and not copied from any source without citation. This work has not previously formed the basis for the award of any Degree , Fellowship or any other similar recognition and title.

Kulanshu Sharma
Digitally signed by Kulanshu Sharma
Date: 2023.06.01 12:41:45 +05'30'

Place : New Delhi

Date : 22/May/2023

KULANSHU SHARMA

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi - 110042

CERTIFICATE

I hereby certify that the Project Dissertation titled " Voteroid Secure API Gateway using Microservices Architecture " which is submitted by Kulanshu Sharma, Roll no 2K21/SWE/12, Software Engineering, Delhi Technological University , Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Rahul
Digitally signed by Rahul
Date: 2023.05.26 12:43:16 +05'30'

Place : New Delhi

Date : 22-May-2023

Mr. RAHUL

SUPERVISOR

ASSISTANT PROFESSOR

SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

ABSTRACT

In the current era of increasing web and software development based on Microservices Architecture , security concerns are also increasing to protect applications from malicious activities, attacks, unauthorized and unauthenticated accesses. An API Gateway resolves that problem to some extent. Various API Gateways are currently available to support various security challenges. This Paper Proposed a System named VSAG (Voteroid Secure API Gateway) which not just works on the cross cutting concerns but also provide advanced facilities like Secured Access key generation, Users subscription model, low degree of coupling, Tracking model of users, blocking/unblocking API's or users. Developers are free to invest time in designing business logics after mapped with VSAG System. At last, various API Gateways are compared on various parameters and functionalities with VSAG.

The gateways which are compared with VSAG are Spring Cloud gateway, Zuul & NGNIX on the parameters includes technology, routing, support of eureka naming server, support to web sockets, degree of loose couplings, user subscription model, user authentication with JWT Tokens, inbuilt API endpoint for license key generation for authorization purpose, inbuilt tracking model of API users.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my project guide **Mr. Rahul**, Assistant Professor, Department of Software Engineering, Delhi Technological University, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Prof. Ruchika Malhotra**, Head of the Department, Software Engineering, for providing us with the required facilities for the completion of the Dissertation.

We thank all the teaching **faculty** of the Department of Software Engineering, whose suggestions and reviews helped us in accomplishment of our project. We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

KULANSHU SHARMA

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In the Microservice architecture design [2], different components are independently connected to each other and also execute independently and communicated through message passing strategy [3]. Scaling is quite easy in microservice architecture where any number of instances of services can be increased according to the server load [4] and M. Song [11] also designed an auto scaling system using kubernetes. Various big companies like Google, Amazon and Netflix are already working on the microservice architecture. The design concept and strategy of microservice architecture are proposed [5,6] in this study and background. K. Bakshi [9] in his article compared the microservice architecture and monolithic architecture on categories like code and understandability.

Esposito et al. (2017) discussed the impact of privacy, security and also discussed the current or existing methods [7]. Securing microservices is a biggest challenge, Quy Nguyen [8] in his study uses OAuth2 and Spring Security over the spring framework to secure the microservices.

An API Gateway is the entry point to enter into any application deployment which majorly works on the throttling policies, authentication and authorizations [1]. Various types of API Gateways are currently available and discussed [10]. API Gateway simplifies the communication between end users and the application backend or application server. It also efficiently decreases the quantity of remote calls between the backend server and the application [11]. API Gateway acts as a shield to protect the application from unauthorized access, unauthenticated access, various web attacks etc. In Microservices architecture, multiple services are developed which run independently and API Gateway is the initial point for all these microservices to access.

This Paper proposes a system named Voteroid Secure API Gateway (VSAG) which is responsible for securing the microservices and helps developers to work on the business logic only. VSAG is a combination of various security mechanisms which make this system more powerful and secure. JSON Web token authentication is used to authenticate the users. Spring cloud config server is used for dynamic configuration of the system. Netflix Eureka Naming Server is used as a load balancer and as a naming server. VSAG is implemented using spring boot framework which uses hibernate for the database connectivity. Relational database Mysql is used to store the API and users data.

The API hits or accesses can be controlled by introducing Rate Limiter in Voteroid secure API Gateway system. Rate Limiter is responsible for stopping DDOS attacks on the applications. The configuration of rate limiter includes multiple parameters like no. of users per second is allowed/ no. of requests per second allowed etc.

VSAG is an architecture inspired from the web server and application server participation to separate application business logics from the client api's and their handling for authorization and authentication purposes. Load balancers must be placed in front of both the web server and the application server to distribute the load according to the capacity of the instances of the application. Application server layer will interact with different data servers or data centers where the data is stored in multiple forms like in relational database management system (RDMS) , Blob store (used to store large size files like videos and audio files), No SQL database where scalability will not be an any issue.

1.1 MICROSERVICES ARCHITECTURE

Microservices refers to the concept where a very large application is divided into the small services in logical fashion and these all services are connected to each other through standard protocols or standard procedures. Various big IT Companies are shifting their monolithic architecture to the microservices architecture. Microservices architecture will provide scalability to the application, availability to the application. Fig. 1 shows the breaking of monolithic

architecture into the micro services architecture. A single monolithic service (node.js API Service) is divided into the services like Users Service, Threads Service and Post Service. Users service will handle all the responsibilities related to the users, threads service is responsible for the management of all the application threads, posts service will be responsible for maintaining the posts of users, the content present in the posts etc.

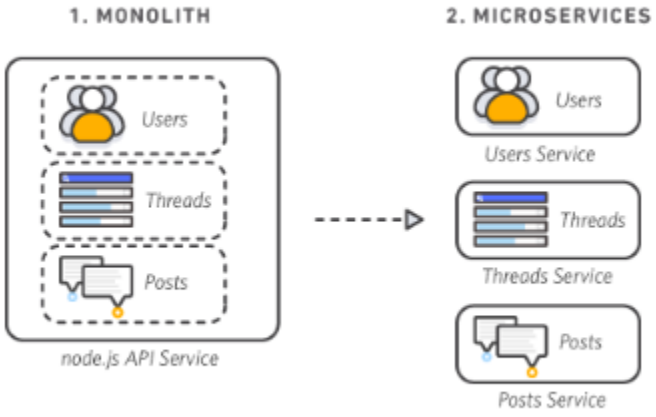

Fig 1.1.1 : Breaking of Monolithic to Microservice architecture

1.2 API GATEWAY

The API Gateway is the mechanism or system which provides the security, authentication, authorization, cross cutting concerns and makes the application loosely coupled with all these functionalities. There are various API Gateways which are present in the market which provide various functionalities like spring cloud api gateway, zuul, NGNIX etc.

API Gateway is the intermediate point between the client requests and the application layer of the system. All the client requests will pass through that gateway and based on certain protocols the request will get accepted or rejected. Fig 1.2.1 shows the high level architecture of the api gateway.

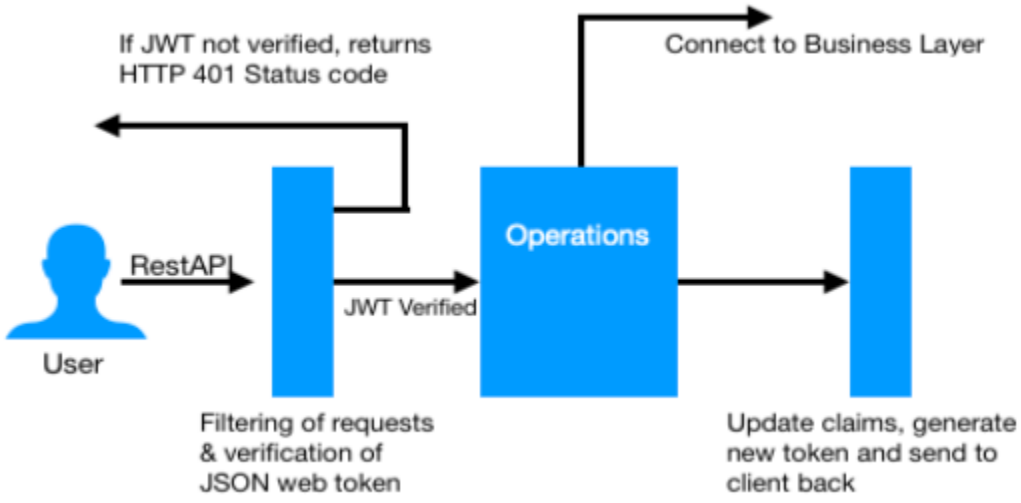

Fig 1.2.1 : High Level Architecture of API Gateways using JWT

1.3 CONTENT ORGANIZATION

The rest of the work is planned and organized as follows: Chapter II is VSAG Architecture where the high level architecture of voteroid secure API gateway is discussed. Chapter III focuses on the VSAG Authentication tree model where different authentication paths are analyzed based on time and space complexities. Chapter IV focuses on VSAG Implementation Model where different implementation models are discussed. Chapter V focuses on System Non functional requirement analysis and Chapter VI marked the Conclusion with comparison of multiple api's like spring cloud gateway, NGNIX, Zuul.

CHAPTER 2

VSAG ARCHITECTURE

The High level design (HLD) of Voteroid secure API Gateway is shown in Fig. 2.1. The basic components which are present in the design are Redis (A distributed cache), Eureka Naming Server (Load balancer), Spring Cloud Config Server (Configuration service), database (Data Storage) and a Git Repository.

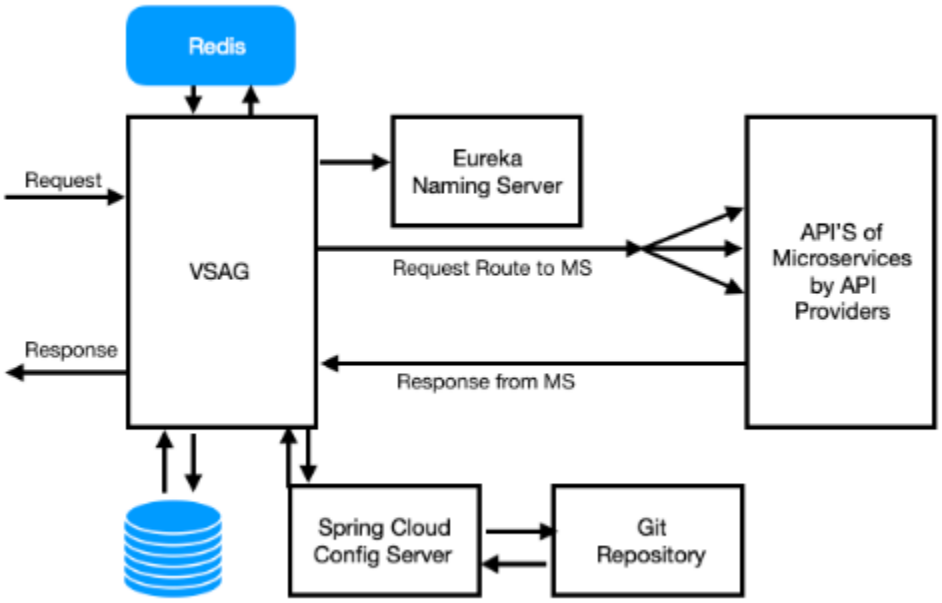

Fig. 2.1 : VSAG High level design

2.1 VSAG COMPONENTS

Various Components are using and inter connected to each other for the implementation of voteroid secure API gateway. These components are planned in spring boot microservices architecture.

2.1.1 REDIS

Redis is a Remote Dictionary Server which acts like a cache in key-value format. Redis made applications more scalable and reduced the user latency to a large extent. Redis is an open source framework and used widely in various IT softwares to reduce the user latency. Redis will also work very efficiently with distributed systems. Redis proved to be very efficient in accessing the session data, user data, and caching data with minimum latency. As per stack overflow [21], redis proved to be the most loveable and valuable database. Redis consists of a wide variety of data sets and data structures to work with. These data structures include lists, sets, objects, array, hashmap, hash table, json, strings etc. Redis plays major roles in various applications like caching, chat, messaging, queues, gaming leaderboards, session storage, rich media streaming, machine learning, geospatial, real-time analytics.

2.1.2 SPRING CLOUD CONFIG SERVER

Spring cloud config server plays a very vital role in the dynamic configuration of application properties and settings of the application. All the microservices settings and properties can be managed by spring cloud config server. All the properties are mapped with the properties files in the github repository. The application will first hit the request to spring cloud config server to fetch the application properties and then this server will hit the github repository for fetching the application properties. It consists of various properties like server port, application service name, expiry time of json web token, life of web sessions etc. It is an open source config server which is currently widely used in various worldwide applications.

2.1.3 NETFLIX EUREKA NAMING SERVER

The Netflix Eureka Naming server was first discovered by Netflix to enhance their application on increasing the size of data and requests. After a few years, Netflix made this open source in the community. Netflix eureka naming server will act as a load balancer as well as service discovery for the services present or deployed in the system. It monitors the information regarding all the available instances of the running services of the application. It shows various information like the instance UP/DOWN status, server IP address of all instances etc.

2.2 PROPOSED ARCHITECTURE

VSAG (Voteroid Secure API Gateway) is connected to four major components which helps the gateway to work more dynamically, gives better response time, enhances security, allows runtime configuration changes etc.

Karl Pauls in his paper [12] describes a project, called Eureka, that Works to simplify the resource or instance management process by developing a resource discovery service to locate required service resources when deploying a service. Eureka Naming Server is used as a load balancer with multiple instances of VSAG in running states and it also provides the User Interface for the number of instances running or in dead state. Eureka Naming Server also provides the generalized configuration for the microservices without mentioning their instances port number and IP addresses.

Spring Cloud Config Server is responsible for the runtime or dynamic changes in the VSAG Configuration. Configuration includes user permissions, server configuration, JSON web token authentication key and much more. It is then connected to the Git Repository where all configurations can be defined or inserted without restarting the VSAG application.

Redis is used as an internal fast memory access tool and database is used to store the license key details with the user permissions. Redis is a very famous In-memory key- value storage system.

Latest Releases of Redis stored data in the distributed nodes which helps to support large data storage capacity but its performance is limited to the decentralized design as to serve a client request, client usually requires two connections. Shanshan Chen in his study [13] proposes a method which is based on client side key-to-node caching that can help to direct requests to the right service node and makes the system more scalable.

2.3 JSON WEB TOKENS AS VSAG LICENSE KEY

VSAG License Key structure as shown in Fig 2.2 is created as a JSON Web token. JSON Web token consists of three parts which are separated by a dot (.). First part is called a header, the second part is called a payload and the third part is called a signature [17,18].

2.3.1 : HEADER STRUCTURE

The Structure of Header contains information related to the -

1) Type of token : ‘typ’
2) Signed algorithm : ‘alg’

Header = Base64Encoded({
“typ” : “jwt”,
“alg” : “HMAC”
})

2.3.2 : PAYLOAD STRUCTURE

The Structure of Payload contains claims related information -

1) Registered Claims : ‘exp’ , ‘auth’ etc.
2) Private Claims : used to identify the user
3) Public Claims

Payload = Base64Encoded({
“exp” : “11/03/2023 04:24:10 ”,
“sub” : “JWT payload subject”,
“userId” : “Specific Id”
….
})

2.3.1 : SIGNATURE STRUCTURE

The Signature Part of JSON Web token plays a very vital role in License Key user authentication and verification -

Signature = Base64Encoded ((Base64Encoded(Header) +
Base64Encoded(payload)) + signed with secret key using algorithm ‘alg’)

The VSAG License Key Contains the basic information of the user to whom the license key belongs. This information is stored in the payload part of a JSON web token. Subject is a registered claim which signifies the unique id of an user. User Name is the License Key owner name. Other information which is stored in the license key is shown in Fig 2.2.

Various fields present in the License key of the Voteroid secure API gateway are Subject which is basically a User Id to whom that license key belongs, User name is the name of the owner of that license key , Created On is the date when the license key is generated, Expiry date is the

validity of the license key, hits limit per day is basically the rate limiter on the user requests that are allowed per day. API Provider is the owner party of the API whose access is given to the user via license key, token id is the unique id of the license key which is generated by the system.

Web tokens play a very vital role in the security, authentication and authorization in web service API. various types of web tokens are present like JSON (Javascript object Notation), SAML (Security Assertion Markup Language). JSON web token will be used in the proposed system as when SAML web token is encoded then it results in longer string length as compared to the JSON. Moreover JSON web tokens can be sent through HTTP Header, in URL, in POST request parameters because of the compact and small size..

Fig 2.2 : VSAG License Key Structure

Why is JSON Web token based authentication chosen over HTTP Session based authentication ?

HTTP Sessions are stored on the server side and the session key is maintained on the client side. Suppose 1 million users are logged in into the application but are not performing any activities and sitting idle. Their data is still present on the session (meaning in server memory) until the session has not expired,so it creates overhead on the server in respect of memory usage and will result in scaling issues if users are increasing continuously. whereas JSON web tokens are stored on client side only and if the user is not performing any activity after logged in then as data is present in token (means on client's browser and not on server) so it removes the memory overhead on server.

Why is Restful web service chosen over other web services ?

The Proposed system will use Restful Web Service rather than SOAP-based Web Service as Abhijit Bohra [28] in his study concludes that Restful web service architecture shows faster performance in respect of response time as compared to SOAP-based web service architecture. He also concludes that the devices that contain relatively lower hardware resources(such as hand held mobile devices) than server machines will perform better with Restful web services based architecture.

Why are JSON web tokens chosen over other web tokens ?

Web tokens play a very vital role in the security, authentication and authorization in web service API. various types of web tokens are present like JSON (Javascript object Notation), SAML (Security Assertion Markup Language). JSON web token will be used in the proposed system as when SAML web token is encoded then it results in longer string length as compared to the JSON. Moreover JSON web tokens can be sent through HTTP Header, in URL, in POST request parameters because of the compact and small size [30,31,39].

Web engineering in India is growing at a very fast pace because of the increasing number of web users due to growing internet connectivity. It also results in increasing competition among the software companies working on similar products with respect to the number of users accessing their website. According to [26], the best way to increase the users accessibility or attract customers on your website is through best user interface and user experience. UI/UX plays a very important role in designing web applications and due to huge competition user interface design is changing day by day on a large scale to meet the user friendly experience and to attract more users. So, the UI/UX part of the web application needs to change frequently as compared to the server (back-end) part of the application. The application must be loosely coupled with these two parts and according to Cesare Pautasso [27] loose coupling in the application can be achieved by introducing Web Service API's. The User Authentication using JSON Web token is secured and provides stateless web service [37,38].

CHAPTER 3

VSAG AUTHENTICATION TREE MODEL

In Fig 3.0 Different paths are possible for SAG Authentication for the Users to Grant or Reject Access to any API. It plays a vital role as a subscription model.

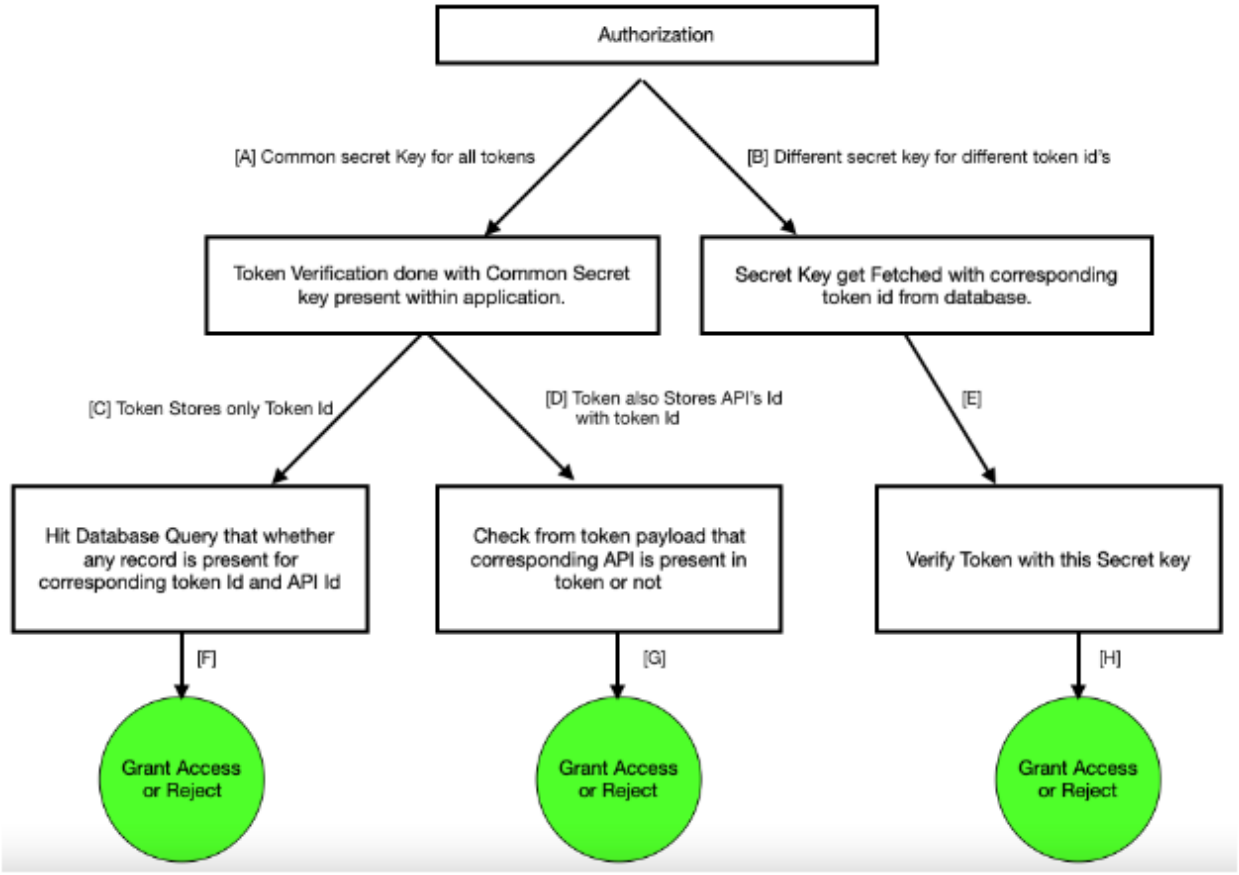

Fig 3.0 : VSAG Authentication Tree Model

Multiple paths can be possible from root node to the leaf node in above voteroid secure API gateway authentication tree model.

3.1 DIFFERENT PATHS OF AUTHENTICATION TREE MODEL

Different Authentication paths will be analyzed to find the most optimized path in terms of time complexity, code complexity and space complexity.

3.1.1 : PATH B ->E -> H

In BEH Path, all the user's License Key's (JWT Token) are signed with different and unique secret keys (one - to - one mapping) and mapping of license key and secret key is stored inside the database. The common attributes of that particular schema are shown in fig 2.2.

Composite Indexing on License Key attribute and API attribute is made to optimize the search query.

License Key	API	Secret Key

Fig 3.1.1 : Path BEH Schema Attributes

The License Key attribute will store only payload and signature part of JWT Token as Header part is almost similar for all the JWT tokens as they are generated with the same signed and encoded algorithm with 'JWT' as a type of token used. When a user wants the access of a particular API, a specific license key is generated with a unique secret key and the mapping gets stored in the database with attributes license key, API, Secret Key. Later, when the user demands the access of that particular API, the user passes the generated license key in the request header for Authentication and Authorization. Authorization is achieved by seeking the available record of that particular license key and API ID from the database. If the record is found then the user is successfully authorized and the derived secret key will be used for authentication purposes. This approach looks like a brute - force approach for authentication and authorization of a user via license key.

3.1.2 : PATH A -> C -> F

In ACF Path, all the user's License Key's (JWT Token) are signed with the same and only one secret key for the entire VSAG Gateway application. The mapping relationship between the secret key and license key is one - to - many mapping. A license key will be generated with the a common secret key and a random UUID token id is created for each unique license key and stored in the payload section of it (which is basically a JWT token).

This Common Secret key is used for the Authentication of all user's license key's and for authorization purposes, a separate database table will be created to store the mapping information of license key token ids with the API Id's that users have access to. A Database Search query is hit to know whether any particular record for given token id and api id is present in the database or not. If it is present, access is granted otherwise authorization is rejected with 401 Http Status Code.

3.1.3 : PATH A -> D -> G

In ADG Path, all the user's License Key's (JWT Token) are signed with the same and only one secret key for the entire VSAG Gateway application. The mapping relationship between the secret key and license key is the same as in ACF Path i.e. one - to - many mapping. A license key will be generated with the common secret key and a random UUID token id is created for each unique license key and stored in the payload section of it (which is basically a JWT token).

This Common Secret key is used for the Authentication of all user's license keys, same as done in ACF Path but for authorization purposes instead of creating and storing the token id and api id mapping relationship into the database tables, all valid and allowed api id's get stored inside the license key only. Once the license key is authenticated, its allowed api id's will be retrieved from the license key payload section and grant access if a particular required api id is present in it or not. If it is present, access is granted otherwise authorization is rejected with 401 Http Status Code.

Now, pros and cons of following paths is discussed -

3.2 PROS & CONS OF THE PATHS OF AUTHENTICATION TREE MODEL

Various pros and cons of the above discussed path of authentication tree model are possible based on their complexities, database calls etc.

B->E->H Path

Pros :
One License Key can store a large number of API's to access.

Cons :
1) Slower route then (ADG) path as Database Call is Required.
2) More Complex Handling then ACF path.

A->C->F Path

Pros :
One License Key can store a large number of API's to access.

Cons :
Slower route then (ADG) path as Database Call is Required.

A->D->G Path

Pros :
Fastest route as no database call is required.

Cons :
Very Limited API's can be accessed by one License key (as token size get increased in increasing api's count)

Our Proposed system will follow the hybrid approach as discussed in the next section where the basic algorithm is designed.

3.3 PROPOSED HYBRID APPROACH

Our Proposed System VSAG is using the hybrid approach of both ADG path and ACF path.

```
if(licenseKey contains <= 20 API's access)
        Path ADG is Used
Else if(licenseKey contains > 20 API's access)
        Path ACF is used
```

If the API's list in license key contains more than 20 api accesses then Path ADG will be followed where all the user's License Key's (JWT Token) are signed with the same and only one secret key for the entire VSAG Gateway application. The mapping relationship between the secret key and license key is the same as in ACF Path i.e. one - to - many mapping. A license key will be generated with the common secret key and a random UUID token id is created for each unique license key and stored in the payload section of it (which is basically a JWT token).

If the API's list of license keys exceeds 20 api accesses then the Path ACF is followed where all the user's License Key's (JWT Token) are signed with the same and only one secret key for the entire VSAG Gateway application. The mapping relationship between the secret key and license key is one - to - many mapping. A license key will be generated with them a common secret key and a random UUID token id is created for each unique license key and stored in the payload section of it (which is basically a JWT token).

CHAPTER 4

VSAG IMPLEMENTATION MODEL

Various systems were already designed which were based on micro services spring boot architecture. Hatma Suryotrisongko [14] designed and developed a backend application for public complaint systems by using spring boot microservice.Carlos M. Aderaldo [15] discussed the benchmark requirements for micro-services research.Satish Reddy Modugu [16] implements the Internet of things application based on REST Architecture and Spring Boot Microservices. VSAG implementation is done through Seven java spring boot based micro services. These Micro services are named as :

1) SAG Microservice
2) API Gateway Microservice
3) Client Microservice
4) User Microservice
5) API Cluster Node Microservice
6) SpringBoot Cloud Config Server Microservice
7) Netflix Eureka Naming Server Microservice

4.1 SAG MICROSERVICES

SAG (Secure API Gateway) is the heart of the VSAG Implementation. The functionalities that this microservice provides includes VSAG License Key Generation, Standard API Registration, Fetching Client's Registered API's, block/unblock new subscription on particular api, fetching all license keys for particular clients with the permissions specific to users. This microservice provides the main building blocks of the voteroid secure API gateway

architecture using microservices architecture. Multiple instances of this service need to be created as this service will handle millions of requests.

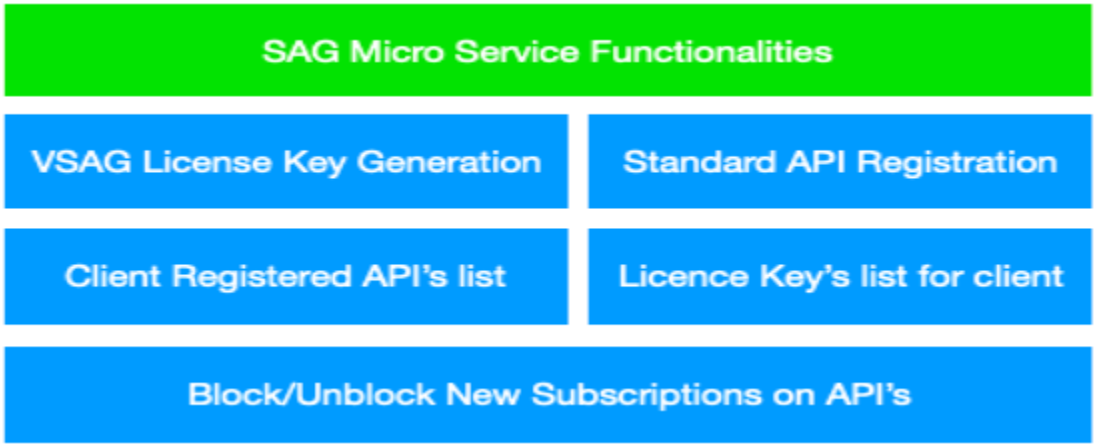

Fig 4.1.1 : SAG Functionalities Chart

4.2 API GATEWAY MICROSERVICE

API Gateway Microservice is the gateway for authentication and authorization of VSAG Users or Clients Only. It is also called the Internal Gateway of VSAG to authenticate all seven microservices. Fig 4.2 shows the architecture of VSAG Internal API Gateway.

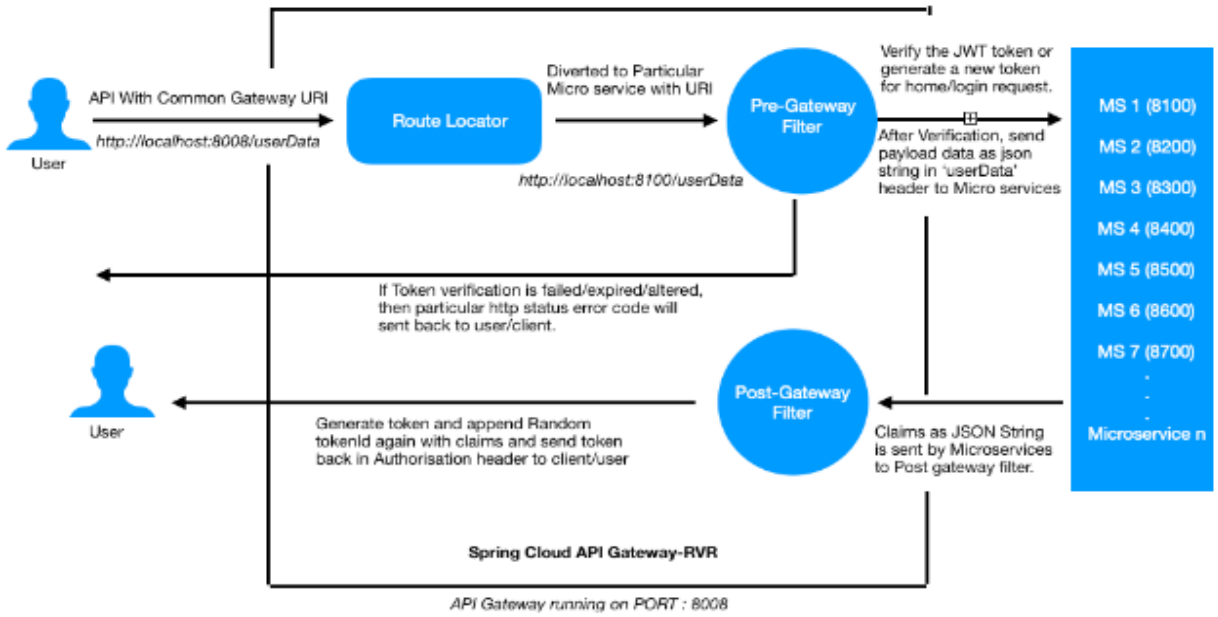

Fig 4.2.1 : VSAG Internal API Gateway

4.3 API CLUSTER NODE MICROSERVICES

API Cluster Node Microservice is responsible for storing the entire details of the registered API's by VSAG Clients. Details includes all Request/Response header details, path variables details, query parameters details, Request/Response body details etc. All details get stored in the database which can be retrieved back according to the query. Java Based Hibernate and JPA library is used with Mysql relational database to perform these operations.

4.4 CLIENT MICROSERVICE

VSAG has multiple clients that want to secure their api's via VSAG. Client Microservice provides the platform and user interface for the client to register themselves, choose appropriate VSAG security plans, register api's and maintain their user access to these client API's. They can also monitor and control the user traffic on particular API's registered by them through the Client User interface Panel. They can block/unblock users as well.

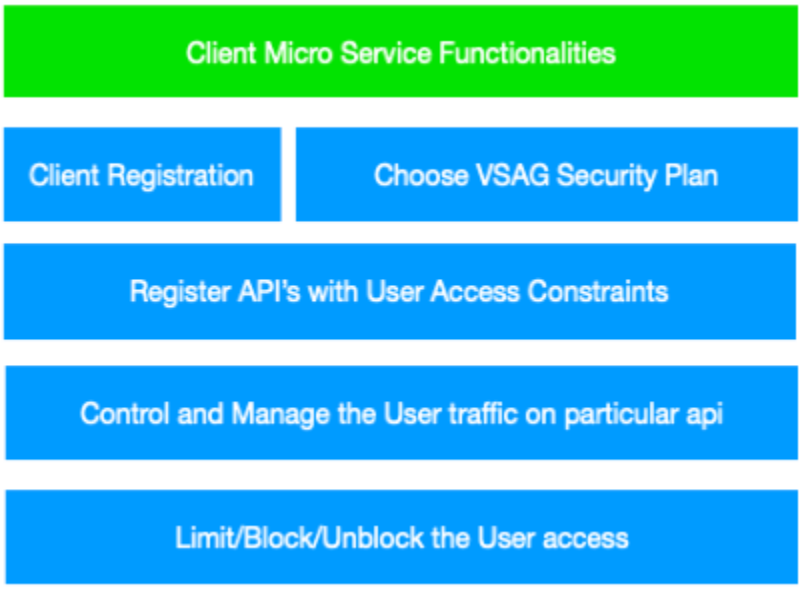

Fig 4.3.1 : Client Microservice functionality chart

4.5 USER MICROSERVICE

VSAG can have multiple clients and multiple clients can have their own multiple users who can access their registered api's. These Users are managed by the User Microservice. It provides various functionalities like user registration, visibility of clients available api's, choose subscription plan managed by clients for particular api's, manage different license keys available for that particular user etc.

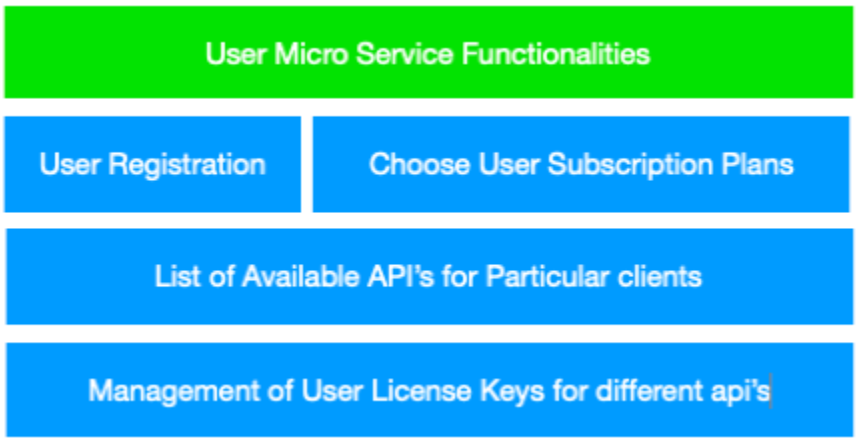

Fig 4.4.1 : User Microservice functionality chart

SpringBoot Cloud Config Server Microservice and Eureka Netflix Naming server are used for dynamic managing VSAG Configurations and load balancing of multiple instances of VSAG respectively.

4.5 BLOCKCHAIN TECHNOLOGY

Blockchain nowadays is a leading technology which is based on the distributed ledger mechanism **.** There are various applications of blockchain technology which enhance the security system of the application, making the system immutable so that no one can change the content of data after the data gets uploaded on blockchain. Many industries are currently using blockchain for multiple purposes in the field of energy consumption, in the field of health care where patients data and medical history get stored on the blockchain distributed ledger.

Blockchain is currently playing a major role in the field of energy sector as it uplifts the applications based on sharing economy and also benefits small renewable generators and potential consumers to play a vital role in the market of energy and renewable resources [20]. Blockchain also had a great impact on the field of Supply Chain Management where it successfully reduced the role of middleman and brokers from the process [21]. The Healthcare industry is also improved a lot by introducing blockchain technology for storing the patients related medical data and managing electronic health records [22,24]. Blockchain also had a great

impact on Post Covid Management like contact tracing, vaccine management, disaster relief management etc. and also proposes models to handle Covid pandemic. [23,25].

In cryptocurrency, blockchain proved to be a more secure and scalable technology and is gaining huge importance in recent years because of its distributed ledger nature and immutability nature. Blockchain is a chain of multiple blocks in which the calculated hash of one block is stored in the previous hash section of the next block. In this way, if anyone wants to alter any block of blockchain then the hash value verification will fail and all the succeeding blocks will get invalid.

Blockchain consists of the blocks connected through each other by cryptographic hash functions. Next block contains the hash of the previous block and goes on. In our proposed system the major data structures used to develop the blockchain is Hash map and double linked list. The *values* part of hashmap (*also called as block*) is connected through a double linked list with the next block containing the hash of the previous block to develop the blockchain as shown in the fig 4.5.1.

Hash map is a data structure based on key value pairs and the search operation normally takes *O(1)* time complexity if the key is known to us. No specific order of data values are maintained in the hash map. To make data values/nodes in specific order, a double linked list data structure is used to connect values/nodes of the hashmap.

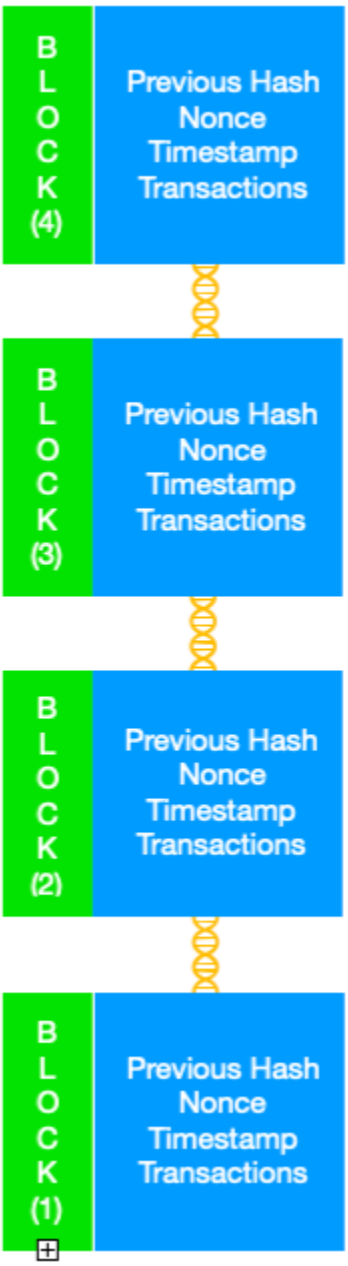

Fig 4.5.1 : Blockchain Implementation Using Hash Map

CHAPTER 5

SYSTEM NON-FUNCTIONAL REQUIREMENT ANALYSIS

In VSAG, New API Registration count will be much lesser as compared to the old API's access/usage and redirection request. This read-write ratio will be estimated as 1000:1 (Means in a particular interval of time, 1 new API is registered and 1000 API access requests are received).

5.1 TRAFFIC ESTIMATE CALCULATION

The VSAG system is planned to control a maximum 10 thousand new API registrations per day. So, total read operations (API accesses) will be 1 Million per day.

No. of read operations per day = 10K *1K => 1M

Traffic of API Registration per minute = 10K/24*60 => **7 API's/Min**

Traffic of API Access per minute = 1M/24*60 => **7000 API's/Min**

So, the traffic for New API Registration is 7 API's/Min and for API's Access is 7000 API's/Min.

5.2 STORAGE ESTIMATE CALCULATION

VSAG system is using a relational database mysql which stores the details of API's into a relational table. 2048 bytes are consumed as storage for new entries of the API into the database table. System is expected to store API's for 10 years.

Total API's created in 10 years = 10K * 30 days * 12 months * 10 years = 3,6000K API's

Total Storage needed for 10 years = 3,6000K * 2048 bytes = **68.66 GB**

Approximately, 68.66 GB Storage space is required to store API details inside the database. Adding other storage requirements for user and client details, approx. 80 GB of storage is required.

5.3 BANDWIDTH ESTIMATE CALCULATION

For the New API Registration (write request) system is expecting 7 New APIs per minute.

Total incoming data bandwidth required = (7/60 seconds)*2048 bytes => **0.23 KB/sec**

Total outgoing data bandwidth required =(7000/60 seconds)*2048 bytes =>**0.23 MB/sec**

TABLE 1 Estimated Calculated Values on Parameters

Estimate Parameter	Estimate Calculated
New API Registration Traffic	7 API's/Min
API Access & Redirect Traffic	7000 API's/Min
Storage needed for 10 years	80 GB
Incoming data bandwidth	0.23 KB/sec
Outgoing data bandwidth	0.23 MB/sec

CHAPTER 6

CONCLUSION AND FUTURE WORK

API Gateway is the initial point of contact between user and business layer(back-end layer) and it is responsible for various security and cross cutting concerns. Various API Gateways are Compared on different parameters that are shown in Table 2. Voteroid Secure API Gateway provides advanced facilities which will help developers to work on their business logic rather than caring for other security concerns. VSAG System design worked on Java Based Technology which supports Eureka Naming Server, supports long lived connections, provides low degree of coupling, provides user subscription model with effective user interface, provides user authentication with JWT tokens and also provides inbuilt tracking model of API's.

Various API gateways are compared based on different parameters like technology, routing, support of eureka naming server, support to web sockets (long lived connections), degree of loose coupling, user subscription model, User authentication with jwt tokens, inbuilt tracking model of API Users etc.

In Future our aim is to integrate VSAG with Kubernetes which will manage service instances more efficiently and effectively. Blockchain as a distributed ledger[19] can be introduced to verify the user authentication data and enhance the security of the system to a larger extent.

TABLE 2 Estimated Calculated Values on Parameters

Operations	Zuul	NGNIX	Spring Cloud Gateway	VSAG
Technology	Java Based	C/C++ Based	Java Based	Java Based
Routing	Yes	Yes	Yes	Yes
Support Eureka Naming Server	Yes	Need Huge Configurations	Yes	Yes
Support to Web Sockets (long lived connections)	No	No	Yes	Yes
Degree of loose coupling	Low	Relatively High	Low	Low
User Subscription Model	No	No	No	Yes
User Authentication with JWT Tokens	Need Developer Support	Need Developer Support	Need Developer Support	In-built
Inbuilt API Endpoint for License Key Generation for authorization Purpose	No	No	No	Yes
Inbuilt Tracking Model of API Users	No	No	No	Yes

APPENDICES - 1

1.1 ECLIPSE

Eclipse is a developing tool for java and all is framework. Eclipse is an open source framework which is owned and managed by Oracle. Various functionalities are provided by eclipse for the smooth development of java based applications. Various frameworks that are supported by eclipse includes :

- Spring boot
- Hibernate
- Java Server Pages (jsp's)
- Servlet's
- EJB
- Swings
- GWT

Eclipse workspace also supports a variety of web servers and application servers. Web server includes Apache Tomcat which is a very famous and mostly used web server. Application server includes wildfly, jboss and glassfish.

1.2 APACHE TOMCAT

Apache tomcat is an open source web server which is currently widely used by various corporations or organizations. Web applications can easily deploy in the web server apache tomcat and can also deploy all dependencies. Various major directories apache tomcat folder contains :

- lib
- conf
- bin

- logs
- temps
- webapps
- work

Webapps is the directory where applications can be deployed with all its dependencies.

1.3 JDK (JAVA DEVELOPMENT KIT)

JDK is the heart and brain of the JAVA Programming language which is responsible for both java application development and java application deployment. Various versions of jdk are currently available in the market from java 6 to java 17. JDK is maintained and owned by Oracle and it is an open source tool for JAVA development. Java can support a wide range of applications from small enterprises to large enterprises and can handle a large number of users. JDK provides various libraries through which the application can integrate with other multiple frameworks and tools. Java is very rich in the set of libraries which includes mailing libraries, sending SMS libraries, database connectivity libraries through database drivers which includes JDBC (JAVA Database connectivity).

REFERENCES

[1]. Siriwardena, P. (2020). Edge Security with an API Gateway. In: Advanced API Security. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2050-4_5

[2]. M Fowler and J Lewis. Microservices. ThoughtWorks, 2014.

[3]. Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch-Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. CoRR, abs/1606.04036, 2016.

[4]. Maurizio Gabbrielli, Saverio Giallorenzo, Claudio Guidi, Jacopo Mauro, and Fabrizio Montesi. Self-reconfiguring microservices. In Theory and Practice of Formal Methods, pages 194–210. Springer, 2016.

[5]. Pan Xiaoyang, Huang Xiaofang. Design [J].] of security control mechanisms for micro-service frameworks Journal of Southwest University of Science and Technology.

[6]. Han Daoqi. Design of Micro-Service Architecture and Security System [J].] and Electronic Technology and Software Engineering ,2019,000(002):199-201.

[7]. Esposito C, Castiglione A, Tudorica C A, et al.Security and privacy for cloud-based data management in the health network service chain: a microservice approach[J]. IEEE Communications Magazine, 2017,55(9): 102-108.

[8]. Nguyen Q, Baker O F. Applying Spring Security Framework and OAuth2To Protect Microservice Architecture API[J]. JSW,2019,14(6): 257-264

[9]. K. Bakshi, “Microservices-based software architecture and approaches,” IEEE Aerosp. Conf. Proc., 2017

[10]. Zhao, J & Jing, S & Jiang, L. (2018). Management of API Gateway Based on Micro-service Architecture. Journal of Physics: Conference Series. 1087. 032032. 10.1088/1742-6596/1087/3/032032.

[11]. M. Song, C. Zhang and E. Haihong, "An Auto Scaling System for API Gateway Based on Kubernetes," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018, pp. 109-112, doi: 10.1109/ICSESS.2018.8663784.

[12]. Pauls, K., Hall, R.S. (2004). Eureka – A Resource Discovery Service for Component Deployment. In: Emmerich, W., Wolf, A.L. (eds) Component Deployment. CD 2004. Lecture Notes in Computer Science, vol 3083. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24848-4_11

[13]. S. Chen, X. Tang, H. Wang, H. Zhao and M. Guo, "Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis," 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 2016, pp. 1660-1667, doi: 10.1109/TrustCom.2016.0255.

[14]. Hatma Suryotrisongko, Dedy Puji Jayanto, Aris Tjahyanto, “Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot”, Procedia Computer Science, Volume 124, 2017, Pages 736-743, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.12.212.

[15]. C. M. Aderaldo, N. C. Mendonça, C. Pahl and P. Jamshidi, "Benchmark Requirements for Microservices Architecture Research," 2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE), Buenos Aires, Argentina, 2017, pp. 8-13, doi: 10.1109/ECASE.2017.4.

[16]. Modugu, S.R., Farhat, H. (2020). Implementation of the Internet of Things Application Based on Spring Boot Microservices and REST Architecture. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds) Software Engineering Perspectives in Intelligent Systems. CoMeSySo 2020. Advances in Intelligent Systems and Computing, vol 1294. Springer, Cham. https://doi.org/10.1007/978-3-030-63322-6_3

[17]. Introduction to JSON web tokens. https://jwt.io/introduction/

[18]. JSON web token (JWT). https://tools.ietf.org/html/rfc7519.

[19]. Gomber, P. ”Hinz-O. Nofer M. Schiereck D.,‘Blockchain’.” Springer 59.3 (2017): 183-187.

[20] Andoni, Merlinda, et al. "Blockchain technology in the energy sector: A systematic review of challenges and opportunities." *Renewable and Sustainable Energy Reviews* 100 (2019): 143-174.

[21] Saberi, Sara, et al. "Blockchain technology and its relationships to sustainable supply chain management." *International Journal of Production Research* 57.7 (2019): 2117-2135.

[22] Tandon, Anushree, et al. "Blockchain in healthcare: A systematic literature review, synthesizing framework and future research agenda." *Computers in Industry* 122 (2020): 103290.

[23] Kalla, Anshuman, et al. "The role of blockchain to fight against COVID-19." *IEEE Engineering Management Review* 48.3 (2020): 85-96.

[24] Al Mamun, Abdullah, Sami Azam, and Clementine Gritti. "Blockchain-based Electronic Health Records Management: A Comprehensive Review and Future Research Direction." *IEEE Access* (2022).

[25] Sharma, Abhishek, et al. "Blockchain technology and its applications to combat COVID-19 pandemic." *Research on Biomedical Engineering* (2020): 1-8.

[26] Eugeniu Cozac, ”The importance of UI/UX in attracting customers” April 1 2021, URL:https://eugeniucozac.medium.com/the-importance-of-ui-ux-in-attracting-customers-f6db052243ba

[27] Pautasso, Cesare & Wilde, Erik. (2009). Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design. 18th International World Wide Web Conference. 911-920. 10.1145/1526709.1526832.

[28] Bora, Abhijit, and Tulshi Bezboruah. "A comparative investigation on implementation of RESTful versus SOAP based web services." International Journal of Database Theory and Application 8.3 (2015): 297-312.

[29] Giessler, Pascal, et al. "Best practices for the design of restful web services." International Conferences of Software Advances (ICSEA). 2015.

[30] Calles M.A. (2020) Authentication and Authorization. In: Serverless Security. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6100-2_9

[31] JSON web tokens, URL: https://auth0.com/docs/secure/tokens/json-web-tokens

[32] Cosmina I. (2020) Aspect-Oriented Programming with Spring. In: Pivotal Certified Professional Core Spring 5 Developer Exam. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5136-2_4

[33] Prasad Reddy K.S. (2017) Web Applications with Spring Boot. In: Beginning Spring Boot 2. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2931-6_10

[34] M. P. Hossain, M. Khaled, S. A. Saju, S. Roy, M. Biswas and M. A. Rahaman, "Vehicle Registration and Information Management using Blockchain based Distributed Ledger from Bangladesh Perspective," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 900-903, doi: 10.1109/TENSYMP50017.2020.9230781.

[35] Benarous, Leila, et al. "Blockchain‐based forgery resilient vehicle registration system." *Transactions on Emerging Telecommunications Technologies* (2021): e4237.

[36] Nofer, M., Gomber, P., Hinz, O. *et al.* Blockchain. *Bus Inf Syst Eng* 59, 183–187 (2017). https://doi.org/10.1007/s12599-017-0467-3

[37] M. Haekal and Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," 2016 International Conference on Informatics and Computing (ICIC), 2016, pp. 175-179, doi: 10.1109/IAC.2016.7905711.

[38] S. I. Adam, J. H. Moedjahedy and J. Maramis, "RESTful Web Service Implementation on Unklab Information System Using JSON Web Token (JWT)," 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), 2020, pp. 1-6, doi: 10.1109/ICORIS50180.2020.9320801.

[39] Lakshmiraghavan, B. (2013). Web Tokens. In: Pro ASP.NET Web API Security. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-5783-7_10

[40] A Rahmatulloh *et al* 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* 550 012023