Jurnal Inovatif : Inovasi Teknologi Informasi dan Informatika, 5 (2) (2022) 114-121
Journal homepage: http://ejournal.uika-bogor.ac.id/index.php/INOVA-TIF/index
E-ISSN : 2654-5519. P-ISSN : 2654-553 | DOI : 10.32832/inovatif
Published by: Universitas Ibn Khaldun

Implementation of Web Service Security Using Oauth2 in the Financial Facility Search System

Danu Maulana Zakaria[1], Fitrah Satrya Fajar.K[2]

[1,2] Program Studi Teknik Informatika, Fakultas Teknik dan Sains, Universitas Ibn Khaldun, indonesia

E-mail Coresponden Author: danu@gmail.com

Received : September 2022 Accepted : October 2022 Published : November 2022

Abstract

Implementation Of Web Service Security Using Oauth2 In Financial Facility Search System. The financial facility search system is an application that makes it easier for people to find the nearest financial facility. A similar application is Google maps which have been widely used to search for financial facilities, but for completeness of information and features, it is incomplete. So financial institutions need applications that can manage data on their financial facilities without other vendor intermediaries. This financial facility search system will be built on a mobile platform and website that is integrated with a web service. Financial facility data can be accessed publicly by the public so that the integrity of financial facility data needs to be maintained and secured from several threats that may occur. One such authentication process is the OAuth2 method. With the existence of OAuth2, this financial facility search system can overcome threats that may occur so that data integrity can be maintained.

Keywords : Financial Facilities , OAUTH2, Web Service

Introduction

A financial institution (LK) is a business entity that has main assets in the form of financial assets that function as an intermediary for the community with the interest of saving their funds and functioning as a channel for the collection of these funds to parties who need more funds in the form of loans and other assets in the form of bills in the form of shares. also bonds [1]. FIs have various forms of financial facilities consisting of categories such as ATMs, branch offices, head offices and so on. Financial facilities are devices owned by financial institutions to conduct transactions with users, for example branch offices, head offices, ATMs and so on. LK as the institution responsible for financial facility information wants to make it easier for the public to obtain valid financial facility information. As a form of realization of the dissemination of complete and valid information on financial facilities, FIs require an application to search for financial facilities.

A similar application to search for financial facilities can now be done using Google Maps, but for complete information and features regarding financial facilities it is still incomplete, such as the absence of information regarding weekend operations of the financial facility and what products it has. Financial facility data available on Google Maps is still incomplete, therefore this application is needed so that every financial institution can manage its financial facilities to this application without going through other vendors. This financial facility search application will be built on a mobile platform and website that is integrated with a web service. Financial institutions can manage financial facility data on the web admin while the front-end and mobile web will be accessed by the public to search for financial facilities.

Financial facility data can be accessed by the public publicly so that the integrity of the data must be maintained and only financial institutions are able to change the data. The data integrity of financial facilities needs to be maintained and secured from several threats that may occur. Threats that are likely to occur are modification or alternation, snooping, and usurpation [2]. In order to maintain the validity of financial facility information issued by FIs, it is necessary to implement a security protocol in the financial facility search application. One form of implementing security protocols is using tokens to access data sources, when a third-party application will access the data source, it will first ask the server authorization, whether the token sent is valid or not, if it is valid then the resource server will provide the source the required data and if it is invalid it will display that the application is not entitled to access the data source.

To be able to use the token, an authentication process is required on the user. There are many ways that can be done in the user authentication process. One of the authentication processes is the OAuth method. OAuth stands for Open Authorization, is an open and free authorization protocol sourced from a resource server, which gives access rights to third parties to be able to access data from server resources in the form of tokens [3]. OAuth was created on December 4, 2007 known as version 1, then on June 24, 2009 it was revised and in April 2010 was finalized. Then the next project of the OAuth protocol was continued in October 2012 namely OAuth version 2 (OAuth2) [4]. This OAuth2 authorization process will be carried out if you have authenticated in the form of id_app and user-agent, because authentication is a process to determine a person's identity.

Authorization and authentication are processes that will be performed on OAuth2 security. OAuth2 in the financial facility search system will first authenticate in the form of id_app and user-agent so that authorization is given in the form of access tokens and refresh tokens. Then it will re-authenticate to check whether the access token is valid or not, because the access token and refresh token provided have an expiration date. If the access token has expired, it is required to re-authenticate using a refresh token in order to get a new access token.

The method used in the security of the financial facility search system using OAuth2 is the security life cycle. There are 6 main stages implemented in the security life cycle, namely threats, policies, specifications, design, implementation, and operation and maintenance [2].

Web service security using OAuth2 has been done in previous studies, including: implementation of restful web services with OAuth2 authorization on parking payment systems [5]. the application of the django rest framework and Oauth 2 Authentication technology for the Academic Information System of the University of Lampung Android Version [6]. and Implementation of the Open Authorization Method (OAuth2) for Lecturer Data Management at the Nusantara Islamic University [7]. but from the many studies that apply OAuth2 it has not been tried to be applied to several platforms. Based on the description of the background above, this research will build an "Implementation of Web Service Security using OAuth2 on the Financial Facility Search System". In this study, it is hoped that OAuth2 security will enable the public to obtain valid financial facility information.

Methodology

This research was developed using the security life cycle method. The security life cycle method can be seen in Figure 1. This study focused on the threat stage to implementation due to time constraints, and was refined through the testing stage. Based on Bishop (2003) there are 6 main stages that are implemented in the security life cycle, namely: 1. Threats, 2 Policies, 3. Specifications, 4. Design, 5. Implementation. , 6. Operation and Maintenance (Operation and Maintenance).

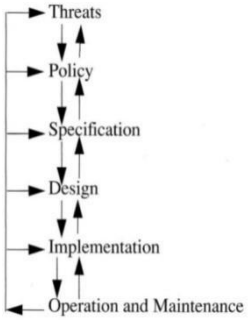

Figure 1. Security Life Cycle

Threat

In this study, threats are one of the stages of analyzing potential violence in a security system in LK. There are attacks that threaten the exchange of information sent and received by users or servers, including the following threats: 1. Snooping, when a module is accessed, there are parties who can capture the address (url) that is being accessed by the LK, thus allowing access to the module to be known by irresponsible parties. 2. Modification or alternation, namely the threat of modification that may occur is changing information in terms of description and location of financial facilities. Modifications can occur if the attacker has done snooping to get information in the form of usernames and passwords. 3. Usurpation, namely the arrangement

made by changing the program code in either the front-end web application or web service so that fraud occurs when displaying information on the financial facility search system.

Policy

Analysis of the policies used in research, the system can be computerized if there are protocols that guarantee such as individual privacy and prevention of threats from information changes by unauthorized parties. By complying with the following agreed policies: 1. Financial institutions can only access their financial facilities. 2. Mobile users can only make requests with the POST method on the rating, bookmark and search filter modules. 3. Admin cannot make requests with the POST method on the rating, bookmark and search filter modules.

Spesifikasi (Specification)

Arsitektur sistem Client-Server merupakan bagian dari gambaran spesifikasi yang mendukung keamanan agar dapat diterapkan sesuai kebijakan yang berlaku seperti pada Gambar 2. Dengan model Client-Server bagian server akan menyediakan perubahan data Client yang dikirim, dan menyediakan data client yang diminta.

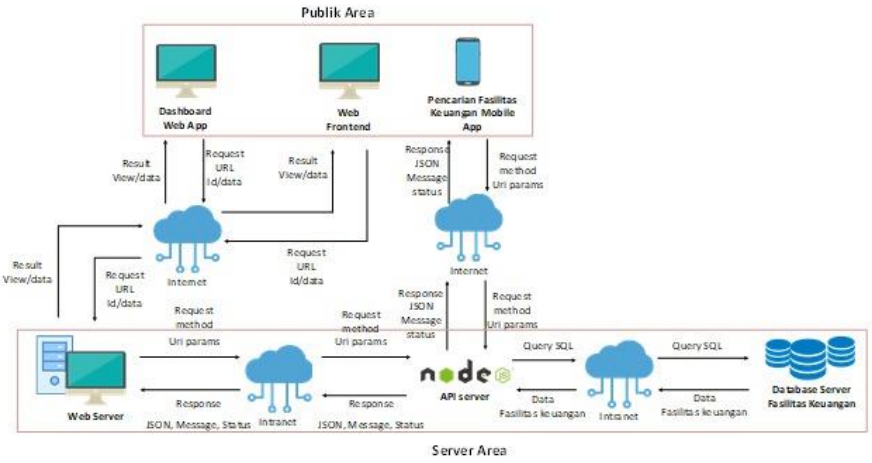

Figure 2. Application System Architecture

Design (Design)

The design described is a design that discusses the overall work of OAuth2 security which was built using DFD (Data Flow Diagram). DFD is a modeling tool used to describe how Oauth2 security works. The design diagram is described as how OAuth2 security works in the form of a data flow process. DFD level 0 is a diagram that consists of a process and describes the scope of OAuth2 security. in Figure 3, DFD Level 0 shows a client that can request a resource if it already has an access token, the client must request an access token first by sending the id_app and user_agent used. Id_app is the primary key that acts as the identity of the client, client is the device used to run the application, and user_agent is information that contains details of the OS, browser, and device used.

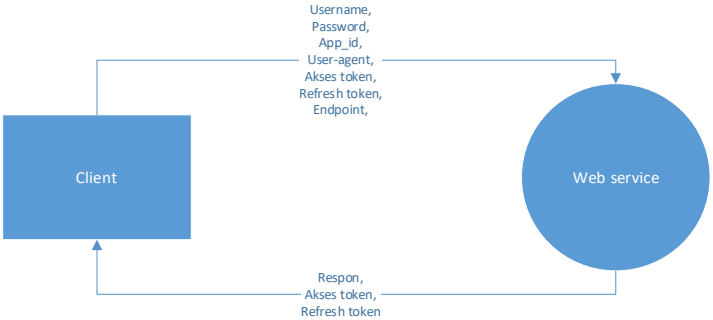

Figure 3. DFD

Result

Implementation

After doing the design, the next step is the implementation of the system to be made. There are 3 steps in creating OAuth2 security, namely:

auth_api Tabel Table Creation

Creating a table is needed in order to store access tokens and to be able to find out who can access this web service. The following is the structure and description of the auth_api table which can be seen in Table 1.

Table Name : Auth_api

Primary Key : id_app

Description: The auth_api table is a table used to store user data and tokens

Table 1. Auth_api

Name Field	Type Data	Size	Null
id_app	varchar	36	No
platform	enum(“mobile”,”website”)		No
name_app	varchar	40	No
name_author	varchar	40	No
email_author	varchar	50	Yes
aksestoken	varchar	50	No
refreshtoken	varchar	50	No
lifetime_token	char	2	No
token_lifetime _refresh	char	2	No
refresh_token _last_regen	datetime		No
created_at	timestamp		No
updated_at	datetime		No
deleted_at	datetime		No

Pembuatan model auth_api

Model dibuat sebagai penghubung tabel yang akan berisi data auth_api agar memudahkan dalam proses update data. Proses update data akan mengubah isi pada kolom ”aksestoken”, ”refreshtoken”, dan ”updated_at”. Berikut merupakan code untuk membuat model auth_api.

```
Auth_api.init({
  id_app: {
    type: Sequelize.CHAR,
    primaryKey: true
  },
  platform: {
    type: Sequelize.ENUM(
'mobile',
'desktop'
)
  }
  , app_name: {
    type: Sequelize.CHAR
  }
  , author_name: {
    type: Sequelize.CHAR
  }
  , author_email: {
    type: Sequelize.CHAR
  }, access_token: {
    type: Sequelize.CHAR
, refresh_token_lifetime: {
  type: Sequelize.INTEGER
 }, created_at: {
  type: Sequelize.DATE,
 }, updated_at: {
  type: Sequelize.DATE
  }
 , deleted_at: {
  type: Sequelize.DATE
 }
 ,last_regen_refresh_token: {
  type: Sequelize.DATE
   }
 }, {
   defaultScope: {
 where:
 Sequelize.literal(
 'deleted_at is null'
 ) },
   deletedAt: 'deleted_at',
   createdAt: 'created_at',
```

}
, refresh_token: {
type: Sequelize.CHAR
}, token_lifetime: {
type: Sequelize.INTEGER
}
	updatedAt: 'updated_at',
	tableName: 'auth_api',
	sequelize,
	modelName: 'auth_api'
	});
	module.exports = Auth_api;

Endpoint creation, authorization and authentication

The endpoint is the url/address of the request that will be accessed to make a token request. The request method that will be used to request tokens is the POST method. id_app and user-agent will be sent via request header. Here is the code to create a url address and perform authentication and authorization. Here is the code to create a url address:

```
'use strict';
const appConst = require('../config/constant');
module.exports = (app) => {
 app.use('/v' + appConst.apiVersion + '/oauth2', require(
'../controllers/authController'));
}
```

The url address that is accessed to request authentication and authorization to get the token is http://localhost:3000/v1/oauth2 which will point to the controller for authentication and authorization of OAuth2 and the url address used to refresh the token by adding an endpoint to the url previously became http://localhost:3000/v1/oauth2/refresh_token.

Testing

Testing of authentication and authorization on the security of the OAuth2 financial facility search system was carried out using blackbox testing with the help of insomnia tools. This test will check the entire request header using the POST method.

Authentication and Authorization Testing

Before testing authentication and authorization, it is necessary to prepare several things, such as creating dummy data in the auth_api table. Then the test is continued by requesting headers in the form of "X-ACCESS-TOKEN" or id_app and user-agent to get access tokens. Next, the request process will authenticate id_app against the data in the auth_api table. The process will generate a row of data that will be processed to authorize/grant access tokens. The authorization results will be sent by the server in the form of JSON and the test results can be viewed using the Insomnia application. In Figure 4 is an example of the contents of the request header on the access token request followed by a preview of the results of the request in the form of JSON data.

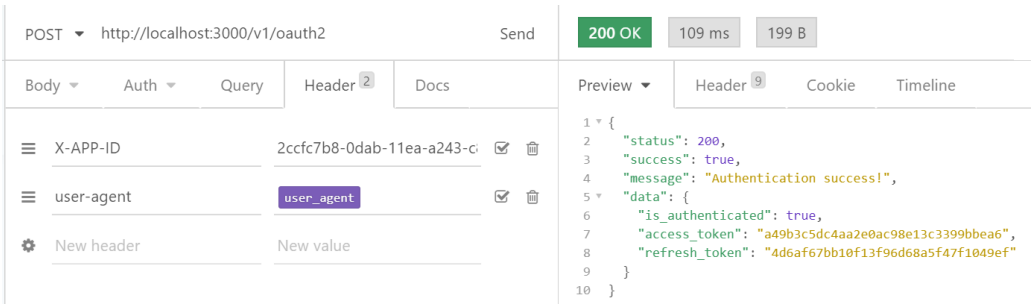

Figure 4. Successful Response To Token Request with App id

The response status from the server gives a code of 200 which means the request has been successfully carried out and is followed by a successful authentication message and an access token code and refresh

token. Access token requests using id_app can only be made once a day. When the request is repeated on the same day, the results will be issued as shown in Figure 5.

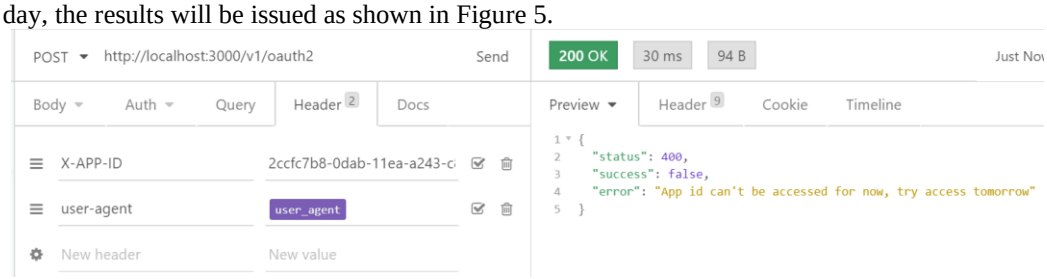

Figure 5. Error Request Token Response with App id

The response status from the server gives a code of 400 which means the request has failed and is followed by an error message. The access token is active for 5 minutes. When the access token's active period has expired, it is necessary to request an access token again using the refresh token obtained from the results of the previous request in Figure 4.6. Figure 4.6 is an example of an access token request using a refresh token followed by a preview of the results of the request in the form of JSON data.

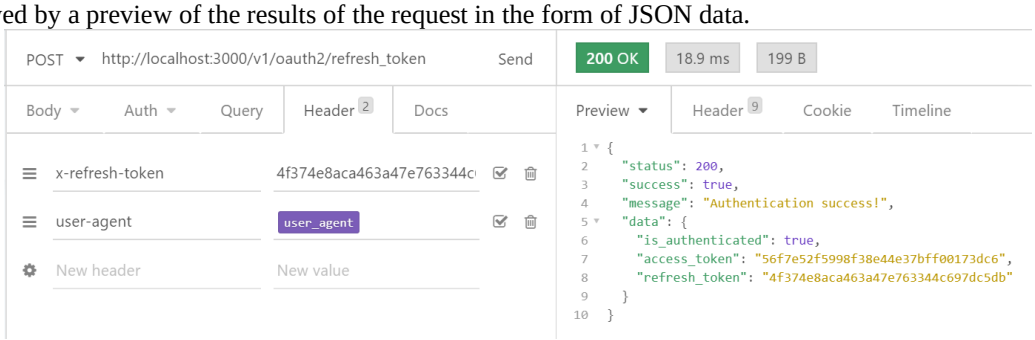

Figure 6. Successful Token Request Response with Refresh Token

Test Request Resource

After getting the access token, then testing the resources owned by the token, the results will be issued in the form of a response from the server if the data is available and the results can be seen in Figure 7.

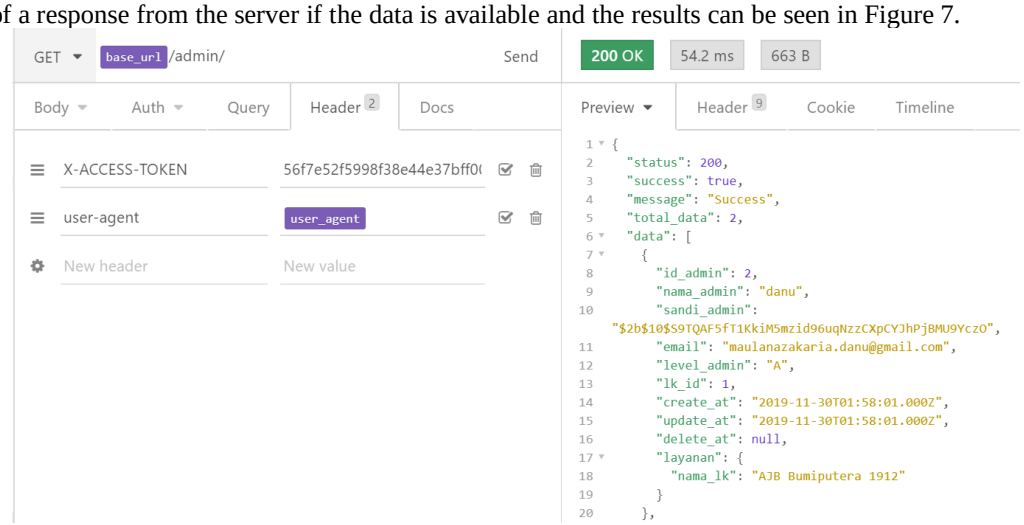

Figure 7. Request Resource

Testing by sending wrong data

The last experiment is done by sending data or combinations that do not match. The following are data transmissions and data combinations that do not match. The following is a test of the appropriate and inappropriate id_app and user-agent combinations:

The combination of id_app and user_agent will be tested with appropriate and non-matching values. The following are the appropriate id_app values and inappropriate user-agents and the results can be seen in Figure 8.

Id_app : 377a1606-09e1-11ea-822d-c85b76597a8b
User-agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko) Version/9.0.2 Safari/601.3.9

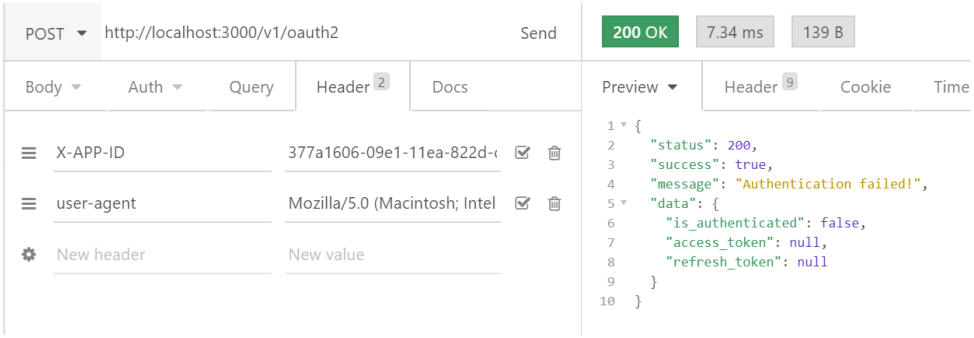

Figure 8.Id_app is appropriate and User-agent is not appropriate

The following is an invalid id_app by changing its value and the appropriate user-agent and the results can be seen in Figure 9.

Id_app : xx377a1606-09e1-11ea-822d-c85b76597a8b
User-agent : Mozilla/5.0 (Linux; U; Android 4.4.2; en-us; SCH-I535 Build/KOT49H) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30

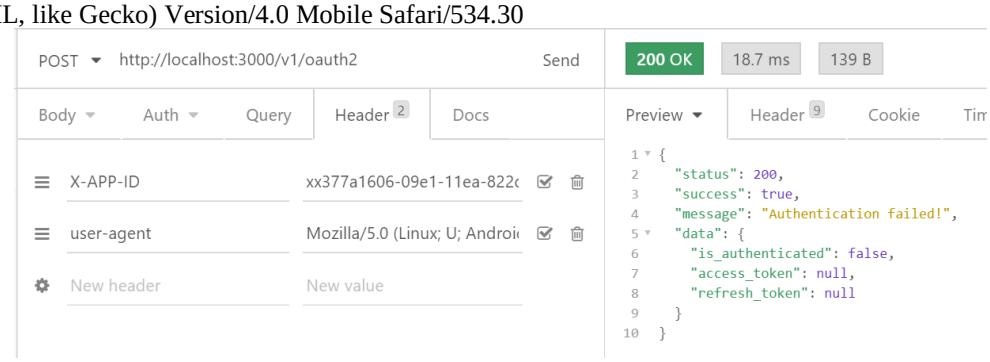

Figure 9. Id_app does not match and User-agent matches

Table 2 is the result of the OAuth2 test using insomnia.

Table 2. OAuth2 . Test Results

No	Test Module	Scenario	Expected output	Test Output	Information
1	*Request token with id_app*	*id_app* and *user-agent match*	Success	Success	
2	*Request token* with *refresh token*	*Refresh token* and *user-agent* match	Success	Success	
3	*Request resource*	Access tokens match and user-agents don't match	Success	Sukses	
4	*Request token* with *id_app*	*id_app matches and user-agent does not match*	Fail	Success	*Id_app mobile* and *user-agent* website
5	*Request token* with *refresh token*	*Refresh tokens match and user-agent doesn't match*	Fail	Succes s	*Id_app mobile* and *user-agent* website
6	*Request resource*		Fail	Succes	*Id_app mobile* and

		Access tokens match and user-agents don't match		s	*user-agent* website
7	*Request token* with *id_app*	*Id_app does not match and user-agent matches*	Fail	Success	Add character encoded id_app
8	*Request token* with *refresh token*	*Invalid refresh token and appropriate user-agent*	Fail	Success	Added refresh token encoded character
9	*Request resource*	Incorrect access token and appropriate user-agent	Fail	Success	Add access token encoded characters

Conclusion

Based on the results and discussion, it can be concluded that OAuth2 authentication has been successfully applied to the financial facility search system web service. This system is able to interact with different platforms, both web-based and mobile-based. Furthermore, using OAuth2 technology is able to help to find out who can access, what access rights are granted and what resources can be accessed. If the id_app and user-agent do not match then the server will not grant access to grant tokens. Then if the token given is changed or created, then the server will respond that the token is invalid/unauthorized. So with the implementation of OAuth2 technology with specifications that support the security of this financial facility search system, it is able to overcome threats that may occur so that data integrity can be maintained.

Refrence

[1] E. H. Peni Sawitri, "Bank dan Lembaga Keuangan Lain," Universitas Gunadarma, 2007.
[2] M. Bishop, Computer Security Art and Science, Boston: Pearson Education Inc, 2003.
[3] S. R. Greg Brail, OAuth The Big Picture, Apigee, 2012.
[4] D. Hardt, "The OAuth 2.0 Authorization Framework," 31 juli 2012. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-v2-31. [Accessed 15 juli 2020].
[5] A. S. I. U. W. M. Indra kusuma, "Implementasi Restful Web Services Dengan Otorisasi Oauth 2.0 Pada Sistem Pembayaran Parkir," Jurnal Simetris, vol. 10, 2019.
[6] S. anwar, Penerapan Django Rest Framework Dan Teknologi Otentikasi Oauth 2.0 Untuk Sistem Informasi Akademik Universitas Lampung Versi Android, 2018.
[7] R. O. Yenni Fatman, "Implementasi Metode Open Authorization (OAuth2) Untuk Pengelolaan Data Dosen di Universitas Islam Nusantara," AINET Jurnal Informatika, vol. 2, pp. 10-18, 2020.
[8] D. Hardt, "The OAuth 2.0 Authorization Framework," 31 juli 2012. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-v2-31. [Accessed 15 juli 2020].