Kumpulan jurnaL Ilmu Komputer (KLIK)
Volume 09 No.02 2022
ISSN: 2406-7857

A WEB SERVICE ARCHITECTURE FOR MOBILE ACADEMIC INFORMATION SYSTEM: A CASE STUDY OF LAMPUNG UNIVERSITY

Kurnia Muludi[1], Saiful Anwar[2], Dwi Sakethi[3], Akmal Junaidi[4], Wartariyus[5]
[1,2,3,4,5]Computer Science, Lampung University
Jl. Sumantri Brojonegoro 1, Bandar Lampung
kmuludi@fmipa.unila.ac.id[1], saiful.anwar1129@students.unila.ac.id[2],
dwijim@fmipa.unila.ac.id[3], akmal.junaidi@fmipa.unila.ac.id[4],
wartariyus@fmipa.unila.ac.id[5]

Abstract

Lampung University has many information systems with many different platforms and programming languages. This condition degraded the performance of each system. In this case study, we develop a web service that can increase collaboration between systems and between business units. Through a combination of Django REST Framework and OAuth 2.0 technology, it is shown that various platforms and applications can work together. While Django Rest Framework provides an excellent web service, OAuth strives to address the shortcomings of proprietary authentication protocols by creating universal and interoperable authorization mechanisms between service units. To prove the concepts, we implemented web service for an Android application client.

Keywords: *Web Service, Django REST Framework, OAuth 2.0, Android Application, Application Programming Interface*

1. INTRODUCTION

Web services use the standard XML format for message delivery. Web services are also not tied to a particular programming language or operating system [1]. According to [2] web service is a software system designed to support interaction that can operate machine-to-machine on the network. Web services have interfaces that are described in machine-processable format (specifically WSDL). Other systems are interconnected with the web service in the manner specified by the descriptions using SOAP messages, typically conveyed using HTTP with XML serialization, along with other standards related to the web.

Representational State Transfer (REST) is an architectural style that aims to minimize latency and network communication while simultaneously trying to maximize the independence and scalability of component implementations [3]. REST evolves with the development of web technology, so it is often used in conjunction with Hypertext Transfer Protocol (HTTP) technology. For this reason,

the RESTful implementation using HTTP technology is called RESTful HTTP. The REST (RESTful) implementation makes multiple apps able to communicate and work together. Through web service, every application on the web has the potential to reach other applications. When applications exchange information through standard web services, they can communicate separately and escape from the system's information, programming, processor, and internal protocols [4].

OAuth (Open Authorization) is an open standard authorization protocol that allows users to access applications without the need to share their passwords [5]. The app owner integrates the user's credentials with authentication technology originating from the API (Application Programming Interface) publisher. OAuth also allows the authorization of protected APIs from desktop or web applications via simple and standard methods. Manage data traffic between applications and use when API publishers want to know who is involved and communicating within the system [6]

Combining the Django REST Framework with OAuth 2.0 technology is very likely to be implemented, so it promises ease and improvement in supporting the integration of various system platforms and applications. OAuth has been widely adopted by the player industry and is on track to become the standard for authorization. OAuth strives to address the shortcomings of proprietary authentication protocols by creating universal and interoperable authorization mechanisms between service units. OAuth 2.0 work mechanism is formed from the active role of 4 (four) parts consisting of client, resource owner, authorization server, and resource server. This article will discuss the implementation of Django REST Framework architecture and OAuth 2.0 technology, which will be applied to the Academic Information System of Lampung University. The result of the implementation of the system architecture is the android application that provides student-related information (class schedules, grades, etc.), lecturers, and development of simple social media for communication between lecturers and students.

Lampung University is the oldest state university in Lampung province established since 1965. Each unit has different system platforms and different programming languages. This condition has a significant influence on the performance of each unit. The web service system is expected to increase collaboration between programmers and inter-business organizations, which allows other applications can use functionality in web services without the need to know the details of programming contained in it.

RESTful web service is suitable for solving problems in the old business concept system to integrated business systems so that one business concept model can be accessed and used by various applications and devices. Also, the old authentication method process makes new problems in the management of user identity. The new method for authenticating data access to the API raises a new name of OAuth. The problem of user authentication and authorization in the academic information system becomes a serious matter to note. The OAuth (Open Authorization) Protocol is an authentication protocol sourced from an API provider

service, which authorizes someone to gain their access rights by redeeming the credential (username and password) into an access token.

Authentication using OAuth regulates the Academic Information System of the University of Lampung (Siakad) in such a way that the user remains safe using the application service by entrusting the presence of a third party (third party). This third party has been trusted as a source server (resource server). The last has the power to discipline and manage all existing credentials. The resource server is also an authorization server that performs client authorization request API by redeeming authorization code in the form of an access token. Usually, an access token is published simultaneously with a token refresh.

2. METHOD

2.1 Design of System Architecture

Web services are built using a RESTful web service. In this research, RESTful is built using a framework in Django called DRF (Django REST Framework). In the process of requesting data (request), the client first has a valid token. First, the client registers to the authentication server to get client_id and client_scret. These two unique parameters will be used to perform the authentication process. Data obtained from the web service is sent in JSON format.

The user authentication process in the built system requires a grant from the resource server (unila.ac.id) by first performing the user authorization process. This process requires permission from the user to continue authentication. The Resource server and authorization server using the same domain that is unila.ac.id in this architecture. The design of web service architecture can be seen in Figure 1.

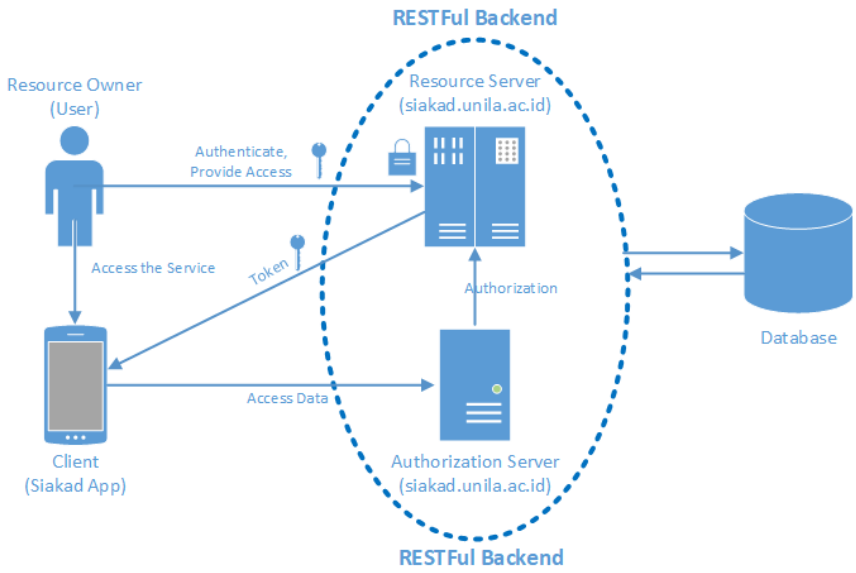

Figure 1. Design of web service architecture and OAuth 2.0

The technology used in system creation is a web service technology with the authentication process using OAuth 2.0. Web services are technologies that provide

process and data integration. In this research, the web service is built to connect Android mobile applications as clients by first performing the authorization process using OAuth 2.0 flow protocol, as shown in Figure 2.

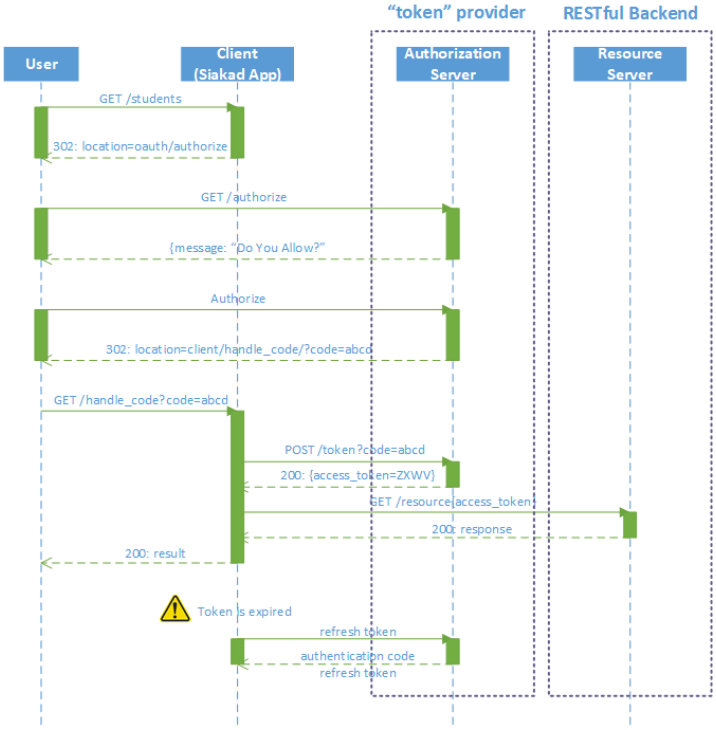

Figure 2. Designing client credentials flow web service with OAuth 2.0

The authorization code for the interacting client can accept incoming requests from the authorization server (may act as an HTTP server). Users want to get the response of student data, through the client user first perform the authorization process as follows:

a. A client is sending an in-app authorization URL. In this case, the user input username and password.
b. The authorization request is forwarded to the Authorization Server, then the response to the question *"Do you allow?"*.
c. The Authorization Server then sends the authorization code to the user.
d. The user sends authorization code to Authorization Server to get access_token.
e. Response received by Authorization Server is used to retrieve data through Resource Server.
f. Resource Server forwards the response to the user to obtain student data.

2.2. System Implementation

Implementation of the system is done after the analysis phase, system design, and interface creation is done. This implementation is done to resolve the existing system in the approved system design and interface documents.

On system implementation for Siakad application using web service technology. In this research, web service built to connect Android mobile application and Resource Server by using Web API to perform data call from a database server. Figure 3 below is the preparation of the built API project

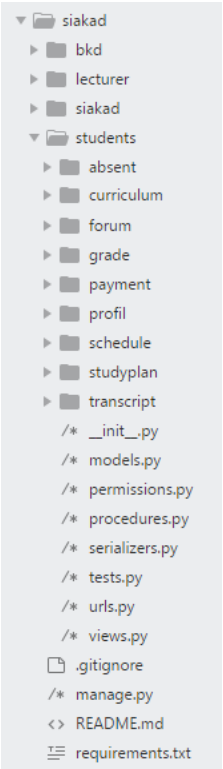

Figure **3**. Structure of API project Siakad

From the above DRF project structure, it looks very different from the standard file in Django. Some of the differences that try to derive from the Django framework are as follows.

Class View

Some views that are implemented in this research are as follows.

a. **APIView**

Class APIView is a subclass of Django view. The use of APIView is easy and neater compared to the Django View class. The reason is that there are methods such as *.post()* and *.get()* that allow for an override. Additionally, some attributes can be set in a class that controls various aspects of API policies.

b. **Generic views**

One of the significant benefits of class-based is how they make it possible to compose a small amount of reusable behavior. The REST framework takes advantage of this by providing several displays that provide commonly used patterns. In the DRF API (Django REST Framework), there is a class to manage the API process that runs a data call. The general view provided by the REST framework allows creating a quick API impression that closely maps the database model. The generic arrangement of this API allows deriving from super class built DRF.

c. **ViewSets**

A framework that allows combining a set of related views in a class. The methods contained in this class are slightly different from the methods in Generic views and APIView contained *.post()* and *.get()* but the methods in ViewSets have *.create()* and *.list()*.

```
class ProfilAPI(generics.ListAPIView):
    serializer_class = ProfilSerializer
    permission_classes = (permissions.IsAuthenticated,)

    def get_queryset(self):
        user = self.request.user
        id_user = user.username
        student_avail = Student.objects.filter(identity_student=id_user).exists()
        if srudent_avail == True:
            content = Student.objects.filter(identity_student =id_user)
            return content
        else:
            raise PermissionDenied
```

Code snippet 1. Class ProfilAPI which provides student details

1) Serializer

Serializer in REST framework works very similar to Django Form and ModelForm classes. The author provides a Serializer class that provides generic and powerful ways to control the output of responses, as well as the ModelSerializer class that provides handy shortcuts for creating serializer-related instances of models and queries.

With that, the author can divide the serializer into two primary uses:

a) Obtain model data from the database in JSON form
b) Like a form that is tasked to validate data and create new data instances of the model

```
class ProfilSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = ('gender_code', 'identity_student', 'depart_code',
                  'strata_code', 'faculty_code', 'study_program_code',
                  'fullname','email')
```

Code snippet 2. Serializer get Student data

2) DRF integration (Django REST Framework with OAuth 2.0)

Django REST framework is supported by OAuth 2.0 toolkit. The package is well documented, and well supported and is currently the recommended package for OAuth 2.0 support. Here are the steps to integrate DRF and OAuth 2.0.

Step 1: Install OAuth toolkit tool using pip

pip install django-oauth-toolkit

Step 2: Added the package to the application settings by adding the following line of code.

```
INSTALLED_APPS = (
    ...
    'oauth2_provider',
)

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication',
    )
}

OAUTH2_PROVIDER = {
    'ACCESS_TOKEN_EXPIRE_SECONDS': 200,
    'SCOPES': {'read': 'Read scope', 'write': 'Write scope', 'groups': 'Access
to your groups'}
}
```

Code snippet 3. OAuth code in settings.py file

From the settings above, we will get the authentication changes that are used in this project. This project has been overridden by default to OAuth2Authentication. This indicates when downgrading the class from *permissions.BasePermission* will require a standard rule process from OAuth that must be skipped.

Also, there are some simple configurations in the existing provider in OAuth, that is.

a) *ACCESS_TOKEN_EXPIRE_SECONDS*

It Represents the lifespan of the access token provided, and this can be configured to adjust to the business mode of the developed application system.

b) *SCOPES*

By default, the scope is set in the client set. However, the scope of access can be limited by digging some of the configurations in this rule.

3) Authentication Rule Process

Web apps running on the server-side are the most common types developed by the owner (resource owner) that are fully supported on OAuth Server. The process that occurs and applies when a visitor wants to access the academic information system to authenticate by clicking the "Log In" button is that the visitor will receive a link set at the browser address :

```
http://192.168.100.6:8000/oauth/authorize/?response_type=code&client_id=TeuUF1SxzFDN4VUunxQg7GTT
SRzHnApbSHdAJQYr&redirect_uri=http://example.com/
```

Some of the query parameters contained in the link series of browser addresses are as follows.

a) *client-id*:
   The value given when registering a web application owned by the resource owner.
b) *redirect_uri*
   Locations to which the user should return after approving access for the app.
c) *scope*
   Application data request access to and refer to OAuth web developer provider.
d) *response_type*
   This is a server-side web application flow code that identifies an authorization code that will be returned to the application once the user accepts and approves the authorization request.

On the side of the android app pages, visitors are published a grant permission page, to be approved by visitors to authorize by OAuth Server (authorization server). Figure X shows the visitor's approval request.

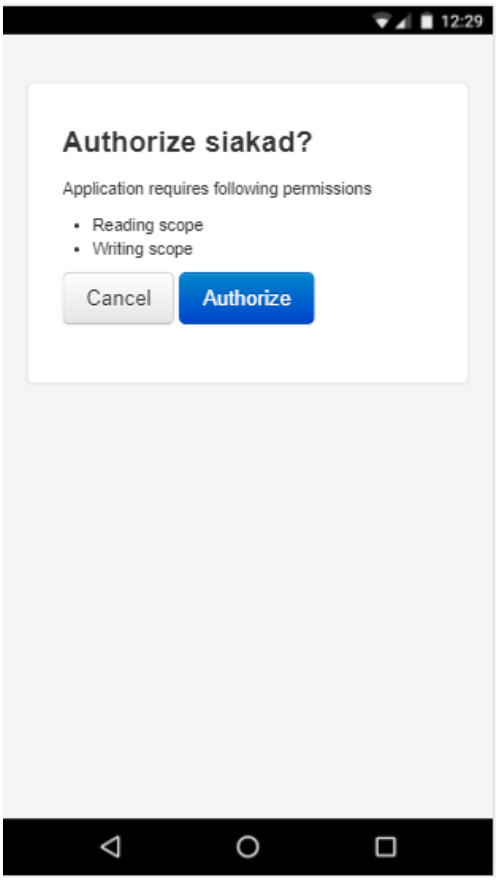

Figure 5. Display user approval request

If the visitor processes the "Allow" action, then the working system will redirect to the web application with an Auth Code. Auth Code for a web application has unique and different codes for every application. If it registered in one of OAuth Siakad, publishers would get a specific auth code. In Figure 18 shows a page with the auth code. In Figure 6 is the auth code obtained when registered on the OAuth service available on Siakad API.

example.com/?code=YaeNo4WBcPjQpWkhLdQghBe9Vi4WRT

Figure 6. Auth code for authentication requests

```
curl --request POST \                                          Copy to Clipboard
  --url http:////oauth/token/ \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data 'client_id=TeuUF1SxzFDN4VUunxQg7GTTSRzHnApbSHdAJQYr&client_secret=qktxPT7m1qo2R4QEm
```

Figure 7. Request data authentication process

The description of each parameter.

a) client_id : The value of the id given when registering an application.
b) *client_secret*: Confidential trust is given when applying.
c) *grant_type*: The value of an authorization_code identifies the redemption of an authorization code on token access.
d) *code*: Authorization code sent to the application.
e) *redirect_uri*: Locations that have been registered and used in the initial request to authorize.

The authentication function is used to gain access token from the registered application, and in case of error, it will display *invalid_grant* (See Response 401, Figure X). The authentication response view is shown in Figure 8.

Response: 200

```
{
  "access_token": "YCXwFq2dQjJdqfylwzcMd4yMMBeQqx",
  "token_type": "Bearer",
  "expires_in": 200,
  "refresh_token": "obJr8a4v5Cgsi5LaOJX5vR6IkiJ0zb",
  "scope": "read write"
}
```

Response: 401

```
{
  "error": "invalid_grant"
}
```

Figure 8. Data request authentication function

As for some error conditions that occur on the URL address (uniform resource locator) and displayed on a browser page such as:

a) *invalid_request*: This is an invalid request, the parameter is missing, or the value of the parameter is not appropriate or not supported.
b) *unauthorized_client*: The client has no authority to request authorization code using .get token method.

c) *unsupported_response_type*: The server authorization does not support obtaining the authorization code using one of the token tokens methods.
d) *invalid_scope*: The requested scope is invalid, unknown, and not appropriate (disabled).
e) *server_error*: The server authorization has a problem, so it cannot prevent the problem and satisfy the token customization request.
f) *temporarily_unavailable*: The existence and condition of the authorization server (server authorization) that cannot handle the request due to server overloading or maintenance.

4) Authorization Process

After authentication to gain an access token. The next process is a data transaction that involves access to the token. Here is an example of some authorization process.

a) Student Profile Data Authorization Process

The student profile details function is used to get the student details by the access token brought. The result of this function is to obtain profile details or fail to get profile details successfully. The process function display obtains the student profile details presented in Figure 9, and the student profile details are shown in Figure 10.

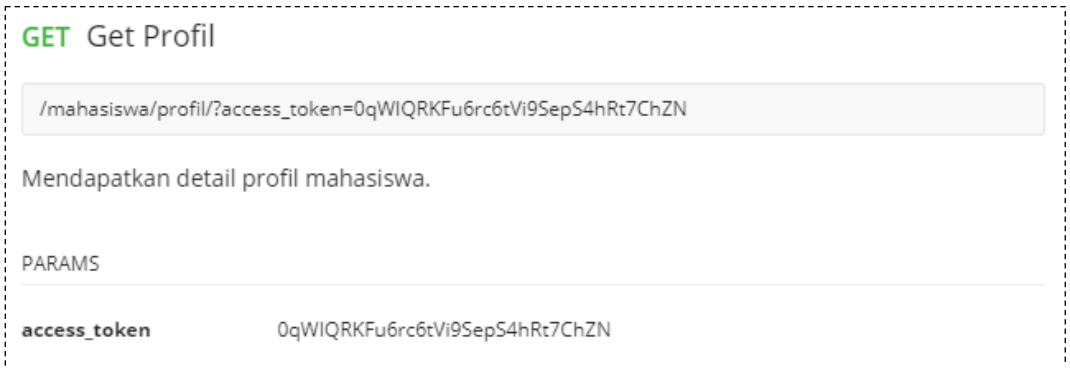

Figure 9. The function of authorization of student detail data

Response: 200

Figure 10. Response authorization of student detail data

b) Data Refresh Authorization Process Token

The token refresh function is used to update token access that has expired. The function display of the token refresh process is shown in Figure 11, and the token refresh response is presented in Figure 12.

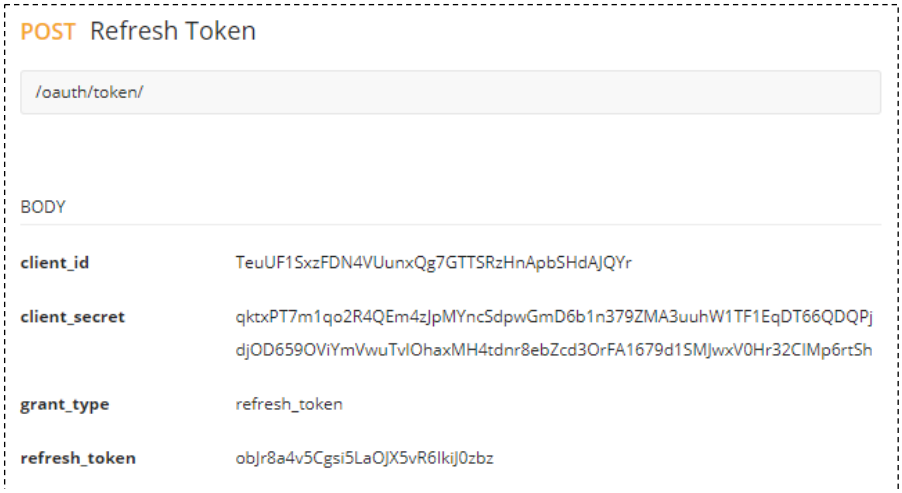

Figure 12. The token refresh authorization function

Response: 200


```
{
    "access_token": "0cVBm0o14oKMif4eaFhzIfQigZGgv4",
    "token_type": "Bearer",
    "expires_in": 200,
    "refresh_token": "YNYf7MOQd9flfqeqvEjCdWIQTsXGZ4",
    "scope": "read write"
}
```


Response: 401


```
{
  "error": "invalid_grant"
}
```


Figure 13. Response authorization refresh token

5) Implementation on Android App

Here are some of the viewing functions available on Android apps.

a) Display function Login (Authentication)

The login page is the starting page of the Siakad application. Users are required to login first before entering the main page of the application, but the login is only done once the first-time users use this application, so no need to log in again. Username and user password obtained from the Academic Information System University of Lampung. However, the data used in this study is dummy data. The login page is an authentication method using the Authorization Code from OAuth 2.0 technology. The look of the login page can be seen in Figure 14.

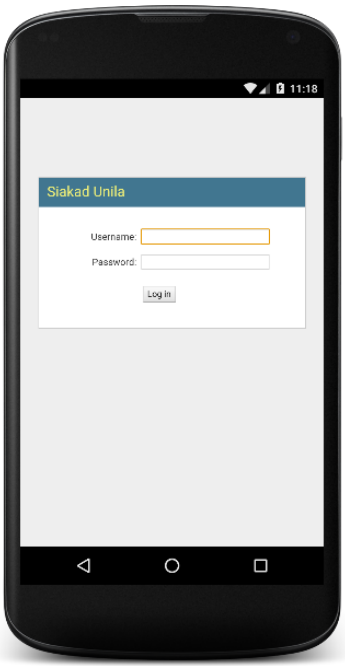

Figure 14. Display the login page

b) Display Home Page

The home page will be displayed after the user logs in the first time using the app or displayed after the splash screen page if the user has used the second time and so on. The home page contains menus, which consist of the Schedule menu, Presentation, Academic Calendar, and Graduation. There is an addition to the top bar of the Language Selection menu, Notification Bell, User Manual, FAQ (Frequently Asked Question), and Contact List. Also, there is the latest banner announcement. The home page view is shown in Figure 15.

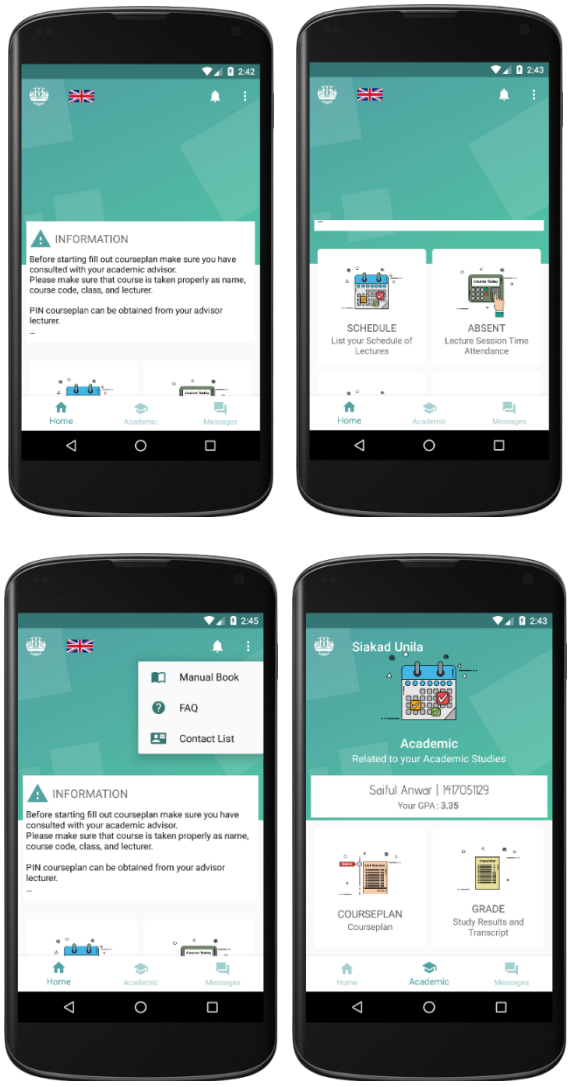

Figure 15 Display Home Page

3. RESULT AND DISCUSSION

The Siakad Android version is an application that implements a web service by using authentication and authorization using OAuth 2.0 technology. The services presented differ from the web version, including daily schedule services, student profiles, and student and lecturer communication forums. Also supported by notification in giving notice in real-time. This application is built using Java and XML programming language for Android, Python for web service, and using PostgreSQL as application database server management.

The Siakad Android version app uses a web service that offers users to perform data transactions that can be minimized — supported by the OAuth 2.0 authentication process that reduces complexity and reduces client services

provided by the web service itself. The application must first be registered as a client to get access permissions from the server. In performing data transactions, the Owner performs the authentication process through the Authorization Server by using the owned resource to obtain the access token. The process of database processing using data parsing techniques, with first built web API, then accessed through the Android operating system that connects directly with the server in real-time. Then the credential data will be stored in Android as a sign that the app successfully obtains permission to access data from Resource Server. The credentials are temporary, so the app will continually update the access token.

This application uses a user-oriented approach to get the information needed for this application by the needs of its users. The Implementation of the user interface shows the hallmark of the University of Lampung. Also, user experience is built and implemented according to the perspective and user interaction on the application, where most users are students.

4. CONCLUSION

It has been successfully built application Siakad Android version made to produce web service and authentication technology by applying technology Django REST Framework and OAuth 2.0. This app has successfully displayed lecture schedules, displays academic calendars, online graduation pages, grades and study plans, curriculum based on year of class, displays tuition fee payment records, profile summaries, and displays various types of forums and posts on student users.

REFERENCES

[1]. Cerami, E. **Web Services Essential**. California: O'Reilly Media. 2002.
[2]. Booth, D., Haas, H., McCabe, F., Newcomer, E., Michael, C., Ferris, C., & Orchard D. **Web Services Architecture - W3C Working Group Note 11 February 2004.** Url: https://www.w3.org/. 2004.
[3]. Fielding R. (2000). **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation. University of California. Irvine.
[4]. Breitman K, Casanova MA & Truszkowski W. **Semantic Web: Concepts, Technologies and Applications**. London, UK: Springer. 2007.
[5]. Chiragsh. **OAuth** 2. Url: https://code.google.com/p/google-api-php-client/wiki/OAuth2. 2012.
[6]. Google Developers. *Using OAuth 2.0 to Access Google APIs*. Url: https://developers.google.com/accounts/docs/OAuth2. 2017.