TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

ITV40LT

Ilja Krjutškov
112528

AUTHENTICATION WITH MICROSOFT OAUTH2 SERVICE, MICROSOFT OUTLOOK API AND IMPLEMENTED FRAMEWORK

Bachelor's thesis

Supervisor: Ago Luberg
MSc
Assistant

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

ITV40LT

Ilja Krjutškov
112528

AUTENTIMINE LÄBI MICROSOFT OAUTH2 TEENUSE, MICROSOFT OUTLOOK API JA REALISEERITUD RAAMISTIK

bakalaureusetöö

Juhendaja: Ago Luberg
MSc
Assistent

Tallinn 2016

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ilja Krjutškov

20.05.2016

Abstract

The main aim of this thesis is to develop Microsoft OAuth2 library that can be used on multiple platforms. This library should have a functionality of interacting with Outlook mail API. At present period of time there is a lack of such library in Java that is easy to use and not a part of some bigger library or project. None of the current products provide easy to use library of OAuth and Outlook API in Java. Library can be used to determine whether the user is a TUT student and to read his mail.

The library is using the Microsoft OAuth2 endpoint for logging in. It also uses Microsoft REST API for mail operations. Library can be used on Desktop or Android.

The result of the work is the library and Android Application that works with that library. there is also an example command line application for the desktop to show how to use the library.

This thesis is written in English language and is 34 pages long, including 9 chapters, 9 figures and 2 tables.

Annotatsioon

Töö eesmärgiks on implementeerida Microsoft OAuth2 teek, mida saab kasutada erinevatel platvormidel. Realiseeritud raamistik omab funktsionaalsust, mis võimaldab kasutada Microsoft Outlook API-t. Praegusel ajal Java keeles sellist implementatsiooni ei ole, mis oleks kergesti kasutatav ja ei ole osa suuremast projektist või teegist. Teeki saab kasutada TTÜ emaili lugemiseks ja see kontrollib, kas kasutaja on TTÜ tudeng.

Raamistik kasutab Microsoft OAuth2 teenust autentimiseks ja Outlook REST API teenust emaili lugemiseks ja saatmiseks. Raamistikku saab kasutada nii Androidi ja arvuti peal.

Täiendavalt on realiseeritud Androidi rakendus ja näidisrakendus käsureal, mis on näidiseks, kuidas teeki saab kasutada oma rakenduses.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 34 leheküljel, 9 peatükki, 9 joonist, 2 tabelit.

List of abbreviations and terms

DPI	Dots per inch
TUT	Tallinn University of Technology
API	Application Programming Interface
REST	Representational transfer state
HTTPS	Hypertext Transfer Protocol Secured
URL	Uniform Resource Locator
URI	Uniform resource Identifier
HMAC	Hash message authentication code
JSON	JavaScript Object Notation

Table of contents

List of figures

List of tables

1 Introduction

OAuth is used by a majority of large enterprises such as Microsoft, Google, Facebook, Twitter, etc. OAuth is also used in TUT for student email and other services. OAuth is a fast growing protocol. The first version was released in 2007, and an updated protocol OAuth 2.0 was published in 2012. OAuth provides secure access to user resources and user can revoke access at any time. The goal of this work is to analyze OAuth 2.0 framework and develop a library that uses Microsoft OAuth and can be used on desktop and mobile platforms. Library should have a functionality of interacting with Microsoft Mail API.

1.1 Goals

- Implement a library for Microsoft OAuth 2.0 authentication.
- Implement a library for using Microsoft REST API.
- Create an Android example application that uses the implemented library.
- Create a User manual that describes how to use the implemented library
- Summary of how OAuth 2.0 works.

1.2 Thesis Overview

The thesis has seven sections “Introduction” "What is OAuth", "Microsoft Outlook REST API", “Libraries”, “Microsoft Outlook OAuth and API library”, “Android Application” and “summary”.

Section 1 “Introduction” contains introduction of this thesis.

Section 2"What is OAuth" describes a summary of OAuth 2.0 framework.

Section 3 “Microsoft Rest API” describes operation and endpoints of Microsoft API

Section 4 “Libraries” describes libraries and tools used in creating library

Section 5 “Microsoft Outlook OAuth and API library” describes implemented library

Section 6 “Android Application” describes Android application.

Section 7 “Summary” contains summary of this thesis.

2 What is OAuth

OAuth [1] is a framework that allows third party apps such as web application, desktop application, and mobile application to authenticate and provide limited access to specific resources. It consists of a client and a server. The client generates URL that can be used by a user to authenticate, the client then is issued with a token that can be used to access specific resources on a server.

OAuth 2.0 has a lot of new features compared to OAuth1 [2]. OAuth 2.0 is entirely new protocol and is not backward compatible. Main differences are

- More authorization flows and better support for non-browser applications
- Provides a cryptography-free option for authorization. Instead of sending signed requests using HMAC and secret, the token itself used as a secret send over HTTPS.
- Short-lived tokens. Instead of a long lasting token, the client is issued with a short-lived token and a long-lived refresh token. It allows the client to receive new access token without user involvement into the process.
- Separates authorization server responsible for user authorization and authorization server responsible for issuing a token.

2.1 Roles

● End user - the person capable granting access to protected resource.

● Server – the server that stores resources, it can accept and response using secret token.

● Client – application that makes a request to protected resource on behalf of an end user.

● authorization server – server that gives out a secret token to the client after successful authorization process.

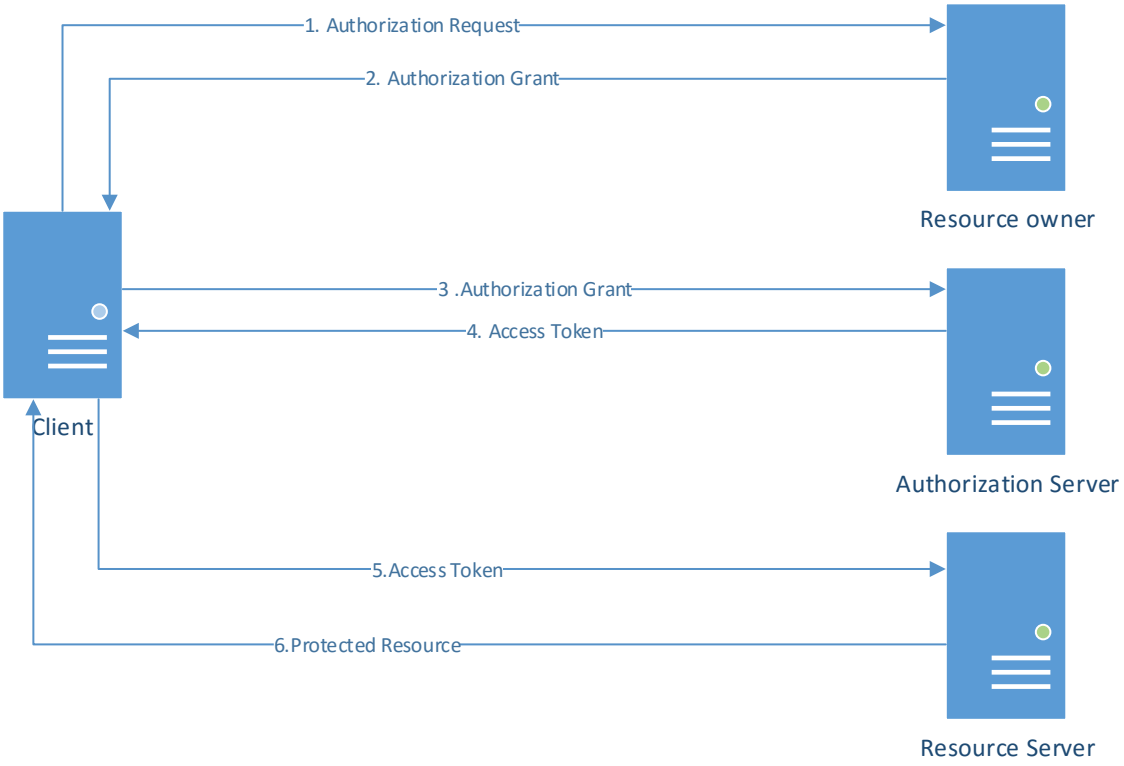

Figure 1. Abstract protocol flow

In Figure 1: Abstract OAuth 2.0 flow illustrates the interaction between roles. Contains following steps:

1.Client requests authorization from resource owner.

2.The client receives authorization grant, which can be represented by one of four grant types. Grant type depends on the method used by a client.

3.Client requests an access token from Authorization Server by presenting authorization grant.

4.The client receives Access token after authorization server controls authorization grant.

5.The client presents access token to authorization server and requests access to protected resource.

6.Resource server validates access token and grants access to protected resource.

2.2 Access Token

The access token is a string that client uses to access protected resources on a server. Tokens have specific scope and duration time. The token may denote an identifier to retrieve information or token may contain some data and signature. Token provides a layer, replacing different authentication methods (login and password) and a single token is used to get access to protected resources on a server. Access tokens can have various formats and defined by the authorization server.

2.3 Refresh Token

The refresh token is credentials used to obtain Access Token when token expires, refresh tokens are given to the client by the authorization server. Issuing refresh token is optional.

2.4 Client Registration

Before a client can use authorization server, it must be registered. It involves filling web form on resource website.

You need to:

- specify the client name
- add redirection URL
- determine application type

Then the client is issued with client id and client secret that client uses to authenticate with the authorization server.

2.4.1 Client ID

Client id is issued to the client after the registration process. This value is used as a username. Value is encoded with the algorithm.

2.4.2 Client Secret

Client secret is used as client password. The client can skip this parameter is this is an empty string.

2.5 Protocol Endpoints

- Authorization endpoint - used to obtain authorization to server via redirection
- Token endpoint – used by client to receive access token
- Redirection endpoint – used by authorization server to return response containing credentials.

2.6 Response Type

Client informs server desired response type by using parameters

- Code - requesting authorization code
- Token – requesting authorization token

2.7 Access Token Scope

The server allows a client to specify the scope of the access request. Authorization Server informs the client of scope that token uses. The value of scope parameter consists of space delimited and case- sensitive strings. Authorization server defines the scope string. The order does not matter; each string adds additional access range.

2.8 Authorization with code

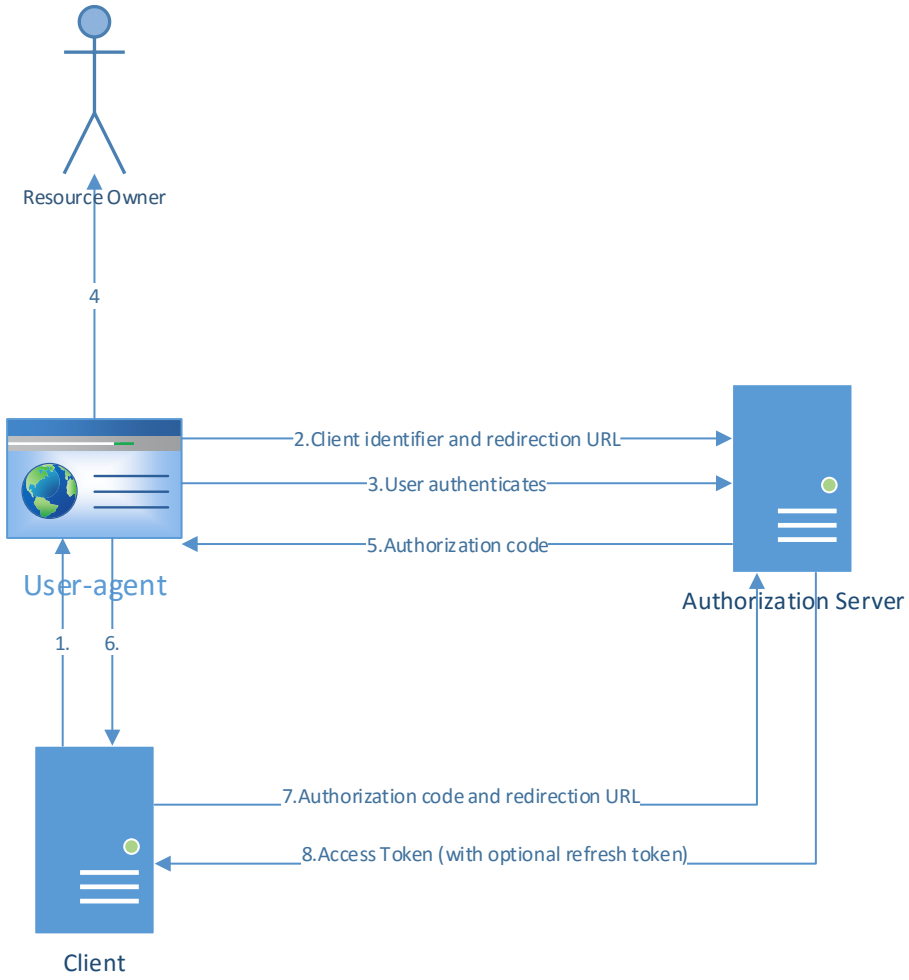

Figure 2. Authorization Code flow

Figure 2. Authorization Code flow: Illustrates Authorization flow with code. Authorization Code can be used to access both access and refresh tokens.

1.Client starts flow by directing to resource owner user agent to authorization endpoint. Example: Application opens browser and redirects to Authorization URL.

2.Browser opens Authorization page with application key include in URL as parameter.

3.Resource owner Authenticates on Athorization server via webbrowser. Example is entering login and password on Microsoft login page.

4.Resource Owner grants clients access request. Example: user presented with view of application can read your mail and needs to accept that.

5. Authorization server redirets to specific url with the authorization code included as parameter.

6.Client parses code from url.

7.Client request access token from authorization server by providing access code received in previouse step.

8. Authorization server validates authorization code and client credentials.If valid, server responds with an access token and optioanly refresh token.

2.9 Error response

If a request fails due to missing or invalid redirection URI, client id is missing or not valid, server should inform the client of the error and must not redirect the client to URI. If server denies request or request is missing or have invalid URI Server informs client by adding following parameters to redirection URI.

Error - required, single error code

- Invalid_request - request parameter is missing, invalid or included more than once.
- Unauthorized_client - the client is not authorized to access token.
- Access_denied - server denied the request.
- Unsupported_request_type - server does not support obtaining token using this method.
- Invalid_scope scope is invalid, unknown or malformed
- Server_error - server encountered an error, preventing fulfilling the request.
- Temporarily_unavailable - the server is currently unavailable finishing the request.

Error description - optional, text providing additional information

Error URI - optional, web page with additional information about the error.

State - required, if the state is present in the authorization request.

2.10 Accessing protected resource

The client can access protected resource by presenting Access Token to the resource server. The server must check presented Token, ensure that it is not expired and have the scope that covers requested resource.

2.11 Native Applications

Native applications are installed and executed on the user device. Authorization endpoint requires interaction between client and resource website. The application can invoke using external browser or render web page within the application.

External browser – native applications can capture the response from authorization server using redirection URI, with a scheme registered within the operating system to invoke client as a handler or manual copy-paste code from the website.

Embedded method – native application obtains credentials by directly communicating with resource server.

3 Microsoft Outlook REST API

Microsoft Outlook API [3] uses REST architecture and programming language can be used to interact with it. The user can get access to user email, calendar, and contacts on Office 365. REST API can be used on a desktop, Windows, IOS or Android device.

3.1 V2 Endpoints

- https://login.microsoftonline.com/common/oauth2/v2.0/authorize - is used to get Access code.

- https://login.microsoftonline.com/common/oauth2/v2.0/token is used to get Access Token with code that you got from the previous URL

To get info about users account Restful service needs to be called.

GET https://outlook.office.com/api/v2.0/me

Response is JSON with info about account


```
{
    "Id": "--ID HERE--",
    "EmailAddress": "Ilja.Krjutskov@ttu.ee",
    "DisplayName": "Ilja Krjutškov",
    "Alias": "Ilja.Krjutskov",
    "MailboxGuid": "e6aaeace-9215-434d-a124-f832a46d62a3"
}
```


Figure 3 illustrates JSON returned by Outlook API.

3.2 Outlook mail REST API

Outlook API [4] lets you read, create and send messages and attachments. It also provides the same functionality with Microsoft accounts (Hotmail, live, MSN, etc.).

3.2.1 Get Messages

Allows to get messages from INBOX. Required parameters:

- Folder_id – folder id or INBOX, Drafts, SentItems, DeletedItems.

Can be used with **$SELECT** to specify properties for better performance. Example:

```
GET
https://outlook.office.com/api/v2.0/me/MailFolders/sentitems/messages/?$select=Sender,Subject
```

The response in message collection in JSON format. The Minimal scope is **mail.read,** scope is string that defines what resources can be accessed on the server. Scope is set when access code is generated by authorization server.

3.2.2 Send and create messages

Allows you to send new messages or create a draft. Required parameters:

• Message – Message to send

• SaveToSentItems – Boolean, true to save to send items

Example:

```
POST https://outlook.office.com/api/v2.0/me/sendmail
{
  "Message": {
    "Subject": "Meet for lunch?",
    "Body": {
      "ContentType": "Text",
      "Content": "The new cafeteria is open."
    },
    "ToRecipients": [
      {
        "EmailAddress": {
          "Address": "garthf@a830edad9050849NDA1.onmicrosoft.com"
        }
      }
    ],
    "Attachments": [
      {
        "@odata.type": "#Microsoft.OutlookServices.FileAttachment",
        "Name": "menu.txt",
        "ContentBytes": "bWFjIGFuZCBjaGVlc2UgdG9kYXk="
      }
    ]
  },
  "SaveToSentItems": "false"
}
```

Figure 4 illustrates JSON that is send to Outlook API.

Response is code „Status code: 202”. Minimal scope is **mail.send**.

3.2.3 Forward Messages

The message can be forwarded; a comment can be optionally specified. The message is saved to sent folder after that.

```
POST https://outlook.office.com/api/v2.0/me/messages/{message_id}/forward
```

Required parameters:

- Message-id – id of message you want to forward
- Comment – Optional comment message
- ToRecipients – List of recipients

The response is status code. The Minimal scope is mail.readwrite.

3.2.4 Reply to sender

Reply to a sender of a message by adding a comment using reply method. The message is saved after that.
Example:

```
POST https://outlook.office.com/api/v2.0/me/messages/{message_id}=/reply
Content-Type: application/json

{
  "Comment": "reply text here!"
}
```

Figure 5 illustrates JSON for replying to sender.

Response is status code. Minimal scope is **mail.read**

3.2.5 Delete a message

Allows user to delete their messages. Deleted messages are not recoverable.

Example:

```
DELETE https://outlook.office.com/api/v2.0/me/messages/{message_id}
```

The response is status code. The Minimal scope is **mail.readwrite**

3.2.6 Move or copy message

Move or copy a message to another folder.

```
POST https://outlook.office.com/api/v2.0/me/messages/{message_id}/move
```

Or for copying

```
POST https://outlook.office.com/api/v2.0/me/messages/{message_id}/move
```

Parameters required:

- Messageid – id of message you want to move
- DestinationId – Id of destination folder, insert this parameter inside the body.

Destenationid is added to request body.

```
{
  "DestinationId": "inbox"
}
```

Figure 6 illustrates JSON which is added to request body.

The response is status code. The Minimal scope is mail.readwrite.

3.2.7 Get Attachment

You can get attachment collection or single attachment, attached to the message.

```
GET https://outlook.office.com/api/v2.0/me/messages/{message_id}/attachments
```

Message id is required parameter. Response is collection of attachments type can be FileAttachment or ItemAttachment.

```
GET
https://outlook.office.com/api/v2.0/me/messages/{message_id}/attachments/{attachment_
id}
```

Returns single attachment by specifying attachment id.

3.2.8 Delete Attachment

Deletes specific attachment by providing message id and attachment id.

```
DELETE
https://outlook.office.com/api/v2.0/me/messages/{message_id}/attachments/{attachment_
id}
```

The Response is status code. The Minimal scope is mail.readwrite.

3.3 Outlook Other REST API

Calendar API and Contacts API are described in the Appendix.

Notifications API allows the application to get changes in user email, calendar or contact data. API uses webhooks to deliver notifications to client apps. Webhooks are HTTP callbacks configured by third-party backend service. When triggered event occurs Office 356 pushes to a callback URL, then app turns and handles it accordingly to apps logic. Right now this api is available in Preview version.

User Photo API lets you download or set user mailbox photo. This api is in Preview version

4 Libraries

In this section Java libraries [5] that are used in Android app for connecting with OAuth 2.0 are described. At first, I was using Apache Oltu for authenticating with OAuth 2.0. It was easy to use, and Microsoft endpoints could be easily added, but it did not work with Android because there were issues with Apache libraries compatibility on Android. I moved to ScribeJava. It was lightweight, easy to use and allowed connection to Microsoft endpoint after a small modification. Other libraries that were available on OAuth 2.0 web page supported OAuth 2.0 as side functionality, and I only needed OAuth 2.0 connection. Unirest allows easy connection to Outlook REST functions and is easy to use. It also supported Android.

4.1 ScribeJava

ScribeJava [6] is framework written in Java that is aimed to build OAUTH 2.0 applications. It supports OAUTH 2.0 and OAUTH 2.0 protocol. ScribeJava is very easy to use OAuth/Oauth2.0 client. It supports async client out-the-box. And it is also Android compatible.

ScribeJava supports many OAuth APIs out of the box such as:

- Twitter
- Facebook
- Windows Live
- GitHub
- And much more

ScribeJava does not support Microsoft Outlook API right away, and it needs to be modified. I had to write JAVA class that ScribeJava is using to build URL for API connection.

The class must extend ScribeJava interface `DeafultApi20`, which is default interface for OAuth2.It must have two methods:

- getAccessTokenEndpoint - returns URL that is used for getting the access token using the access code that we got from URL provided by getAuthorizationUrl method.

- getAuthorizationUrl - returns URL that is used for authorization, URL contains all parameters that are needed for authorization server. The class has all parameters as input as Java object.

4.1.1 How to use

To use ScribeJava, you need to call OauthService in Java.

```
OAuthService service = new ServiceBuilder()
         .apiKey(YOUR_API_KEY)  //Here you put your API key
         .apiSecret(YOUR_API_SECRET) // And APIsecret here
         .build(LinkedInApi20.instance()); // Here you can
chose form OAuth service classes that ScribeJava provides
or use your own class.
```

Next you need can get authorization URL using following method:

```
String authorizationUrl = service.getAuthorizationUrl();
```

Then you can use this URL to get the authorization code. You need to open link in a browser for that. After that code is used to get the access token.

```
OAuth2AccessToken access Token =
service.getAccessToken(code);
```

I did not use ScribeJava for getting the access token. I used Unirest library for that, because ScribeJava did not have this functionality for Microsoft account.

4.2 UNIREST

Unirest [7] is a lightweight HTTP library that is available for multiple languages. Unirest is built and maintained by Mashape (https://github.com/Mashape). It makes GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS requests synchronous and asynchronous.

Unfortunately, UNIREST does not support Android out of the box, and you have to build it with all dependency files into one jar file to prevent Android dependency errors.

Here is example of how to use Unirest in android application:

```
HttpResponse<JsonNode> tokenResponse =
Unirest.post("https://login.microsoftonline.com/common/oaut
h2/v2.0/token")
      .header("Content-Type", "application/x-www-form-
urlencoded")
      .field("grant_type", "authorization_code")
      .field("client_id", "e12965b4-a703-4440-83eb-
8661d56b7b2d") // put client id here
      .field("redirect_uri", "urn:ietf:wg:oauth:2.0:oob")
// redirection url need to use this value for android
applicatiion
      .field("code", code) //put code here that you got
from ScribeJava
      .asJson(); //returns response as json object
```

Figure 7 illustrates token request using unirest library

You can use following code to get the access token using access code.

5 Microsoft Outlook OAuth and API library

I implemented a library that uses Microsoft Outlook OAuth 2.0 for authenticating and uses Microsoft API for using mail services. Library is written in Java and can be used on both Android and Desktop. I did Android Example Application and Desktop command line application.

5.1 How to use

First, jar file needs to be placed in local maven repository or directly in Java project. Jar file can be compiled with Maven command

```
Mvn install
```

Installs file in local maven repository on your computer.

```
Mvn package
```

Packages jar file to target folder of the project. Additional parameter `-DskipTests` can be used to skip tests.

If jar file was installed to local repository it can be used in you maven project with

```
<dependency>
      <groupId>loputoo</groupId>
      <artifactId>ttu.OAuth.Outlook</artifactId>
      <version>0.0.4-SNAPSHOT</version>
</dependency>
```

5.2 OAuth authentication

Microsoft OAuth 2.0 is used to authenticate. To get URL for Microsoft login page. First application needs to be registered on Microsoft page. Then application key and secret will be issued. To get URL you need to call method in application

```
OauthServiceURL OauthServiceURL = new OauthServiceBuilder()
            .apiKey("API-KEY")
            .scope("Scope")
            .callback("URL")
            .apiSecret("API-SECRET")
            .build();
```

Correct API-key and API-secret. Callback URL is needs to be mached with URL specified when registrating application on Microsoft page.

```
OauthServiceURL.getURL();
```

Returns String containing URL. After entering login and password on a Microsoft page, it redirects to URL specified earlier with code. That code is used to get Access Token.

```
OauthServiceURL.getToken(Code);
```

Returns Token object that contains Token and additional information.

Token_type – contains token type
Scope – contains token scope
Expires_in – time for how long token is valid
Access_token – String with access token

Access token can be used with Mail API to get your mail.

5.3 Mail API

API for Microsft Outlook mail operations.

```
MailAPI mailapi = new MailAPI();
```

Mail api needs to be called in application code.

5.3.1 getMail

Returns mail object containing list all of your mail. A mail folder can be specified. Throws MailErrorException if there was an error getting mail, Mail Exception error contains more specific information about the error. Input is token and optionally String with the folder name.

5.3.2 sendMail

Sends mail and returns String with response status text. Returns "Accepted" if mail send was successful. Input is SendMail object. SendMail object can be constructed manually, or MailBuilder can be used. Example

```
SendMail mail = new MailBuilder()
                    .content("—Content--")
                    .toRecipient("i.krjutskov@gmail.com")
                    .subject("—Message subject--")
                    .build();
```

Creates mail object that can be sent to the recipient. Attachment can be added, attachment name and content in bytes need to be put in mail builder.

5.3.3 replyToMessage

Quick reply can be created. Input is token, message id that you need to reply to and comment that needs to be sent. String with status is returned.

5.3.4 ForwardMail

Mail can be forwarded to another email. Input parameters are Token, message id and forwardMail object that contains an array of emails that email is forwarded to.

5.3.5 Delete, copy, move

Mail can be deleted, copied or moved to another folder. Input is Token, mail id and mail name for moving or copying. Returns string with status text.

5.3.6 getAttachments, getAttachment

Returns list of files that are attached to the specific mail. Specify attachment id to receive the single attachment. Input is token, mail id and optionally attachment id. Returns Java object containing an attachment. Throws mailErrorException if there was error getting the attachment.

5.3.7 deleteAttachment

Deletes specified attachment of the message. Input is message id and attachment id. Returns String with status text.

5.4 User Info API

Can be used to get user info, also can be checked if a user is TUT student.

5.4.1 getUserInfo

Return Object with user info. Input is Token. User object contains

- Alias – user alias, for TUT student it is first name and last name divided by dot.
- DisplayName – users real name.

- Id – unique user id.
- EmailAddress – users email address.

5.4.2 isTUTstudent

Checks if a user is TUT student. Checks that users email address is ending with "@ttu.ee". Return true if student is TUT student and false if he is not.

6 Android Application

Android application allows a user to read mailbox content from TUT email using native application. The application uses Microsoft Outlook OAuth and API library.

6.1 Log in

Application using Microsoft OAuth 2.0 for logging in. Created library is used to generate URL. Callback URL is set to "urn:ietf:wg:oauth:2.0:oob" for Android and application secret is not required. WebView [8] is used on android to open sign in page and redirects to new activity after the code is present in URL as the parameter.

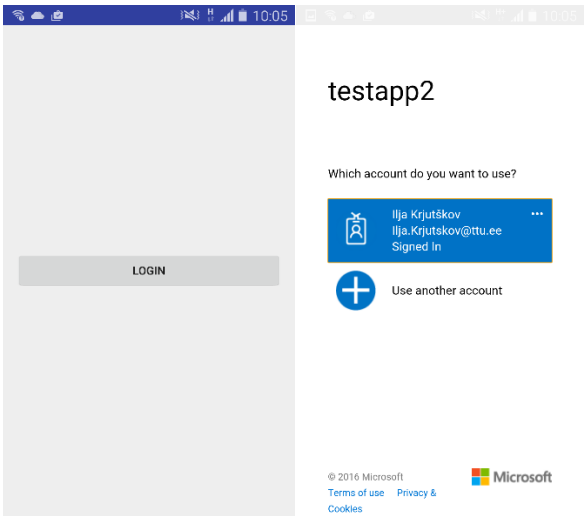

Figure 8.Log in using OAuth

6.2 Reading Mail

The application allows reading mail by pressing “read mail” button after logging in. The application reads mail from the server every time you press read mail button. Mail can be refreshed by pressing refresh in options. Mail can be filtered by specifying email addresses in settings after that email only from that addresses is displayed. If there are no email addresses specified application displays all mail.

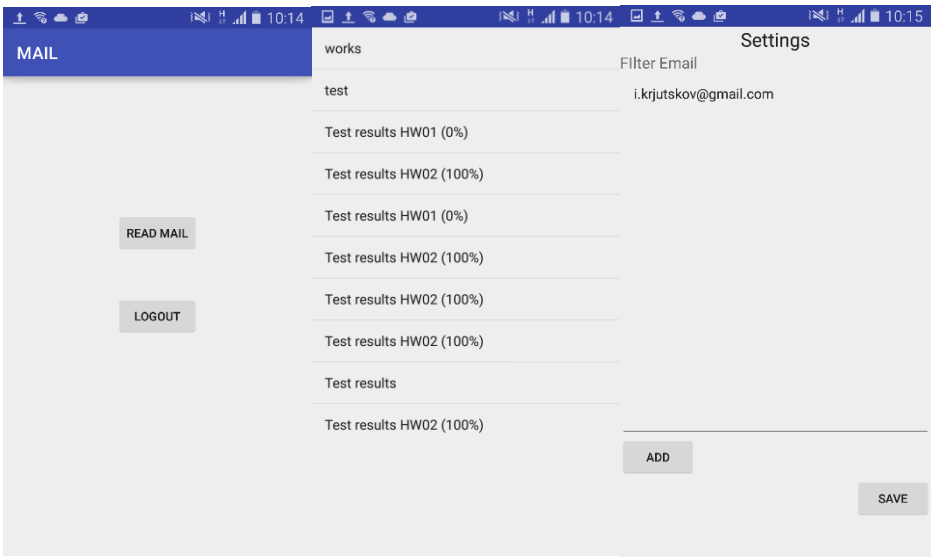

Figure 9. Reading mail

7 Summary

The primary goal of this thesis was to implement the library, which can be easily used in another Java application. Library can use OAuth 2.0 for authentication and Outlook API services. Library can be used with Microsoft account, for authentication and read mail from mail.ttu.ee. The result of the work is also an Android application that uses the implemented library. Android application can authenticate using Outlook account and read mail. Android application can filter mail so that you can display mail form only specific senders. Outlook API implementation returns a java object instead of JSON and throws Java exceptions when an error occurs.

There are several ways to improve the implementation. For example, not every Outlook operations are added. Calendar, Contacts, and Notification API can be added. Android application can be modified to use notification API and display notifications on a phone when new mail is received. Authentication implementation can be improved by adding another support of another service like Facebook or Google.

8 References

[1] E. D. Hardt, "The OAuth 2.0 Authorization Framework," [Online]. Available: http://tools.ietf.org/html/rfc6749. [Accessed 01 04 2016].

[2] "Introducing OAuth 2.0," [Online]. Available: https://hueniverse.com/2010/05/15/introducing-oauth-2-0/. [Accessed 19 05 2016].

[3] "Outlook REST API," [Online]. Available: https://dev.outlook.com/restapi. [Accessed 19 05 2016].

[4] "Outlook Mail REST API reference," [Online]. Available: https://msdn.microsoft.com/office/office365/APi/mail-rest-operations. [Accessed 19 05 2016].

[5] "OAuth 2.0," [Online]. Available: http://oauth.net/2/. [Accessed 01 04 2016].

[6] "ScribeJava," [Online]. Available: https://github.com/scribejava/scribejava. [Accessed 19 05 2016].

[7] Unirest. [Online]. Available: http://unirest.io/java.html. [Accessed 01 05].

[8] "Android performing OAuth2 Authorization," [Online]. Available: https://www.learn2crack.com/2014/01/android-oauth2-webview.html. [Accessed 19 05 2016].

Appendix 1 – Git links

https://bitbucket.org/ilja2/ttu.outlook.lib - implemented library.

https://bitbucket.org/ilja2/androidttu - Android application.

Appendix 2 – Outlook Calendar and Contacts REST API

Calendar API provides Access to events and calendar group data and similar data on Microsoft accounts.

Table 1. Calendar REST API

Operation	Description	Scope
Get events	Returns calendar events in JSON format. Specific start and end date can be specified. Single event by specifying event id.	Calendars.read
Sync events	Synchronize to get new. Updated or Deleted events in specific time period.	Calendars.read
Create events	Create events in specific calendar	Calendars.readwrite
Update events	Change an event.Only properties that are specified are changed.	Calendars.readwrite
Accept event	Accept specific event	Calendars.readwrite
Delete events	Move event to deleted items folder	Calendars.readwrite
Get Attachments	Get collection or single attachment	Calendars.read
Create Attachments	Add file or multiple files to calendar event	Calendars.readwrite
Get Reminders	Get list of events between two dates and times	Calendars.read
Snooze Reminders	Postpone until new date	Calendars.read
Dismiss Reminders	Dismiss a reminder that was triggered.	Calendars.readwrite

Contacts API proovides Access to the users contacts in Office 365 and similar data in Microsoft accounts

Table 2. Contacts REST API

Operation	Description	Scope
Get Contacts	Get all contacts from default or specific folder. Contacts are returned as JSON	Contacts.read
Create Contacts	Create contact in specific folder	Contacts.readwrite
Update Contacts	Change contact properties	Contacts.readwrite
Delete Contact	Remove contact. May not be recoverable	Contacts.readwrite