TÉCNICO LISBOA

Mestrado em
Engenharia Eletrotécnica e de Computadores

Criptografia e Segurança das Comunicações

Relatório de Projeto

Group 21

José Vieira nº 90900
Pedro Guedes nº73637
Pedro Carmo nº 90989

Prof. Carlos Ribeiro

Lisboa, 21 de Janeiro de 2018

Código do servidor

Para a implementação do lado do servidor foi utilizado o *Apache*, juntamente com *Mysql*, utilizando php. Foram então utilizadas as instruções aconselhadas pelo *site*[1] fornecido pelo enunciado. Esta instalação permite a integração de várias componentes, formando uma *LAMP stack* que permite, ao servidor, alojar *websites* dinâmicos e aplicações *web*.

Relativamente ao protocolo *Oauth*, foram utilizadas as bibliotecas descritas no *site*[2] e acessíveis no *github*[3] onde também são fornecidos exemplos de aplicação para cada tipo de *Grant*.

Inicialmente foi criado um diretório "/home/user/Documents/proj-csc/project2". Este diretório é o *root* do projecto por parte do servidor. Neste constam a pasta das bibliotecas utilizadas, gerada pelo *composer* ("vendor/"), e a pasta com o código da aplicação *web* ("src/"). Dentro desta última encontram-se também duas pastas, uma com código a incluir nos ficheiros php ("include/") e a pasta com o código relativo ao log*in* e registo do utilizador ("public/") acedidos pela aplicação móvel. De forma a definir esta última como *root* do servidor apache, foi necessário configurar o ficheiro "/etc/apache2/sites-available/000-default.conf", substituindo:

```
DocumentRoot /var/www/html
```

Por:

```
DocumentRoot /home/user/Documents/proj-csc/project2/src/public
```

Note-se que esta configuração se encontra adaptada para a utilização deste diretório em específico com o apache.

Para permitir o acesso do utilizador aos ficheiros da pasta "public" é necessário, também, acrescentar ao ficheiro "/etc/apache2/apache2.conf":

```
<Directory /home/user/Documents/project2/src/public>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

Posto isto, é necessário realizar a instalação das bibliotecas do *Oauth* utilizando as funcionalidades do *composer*. Para isto, procedeu-se à execução dos seguintes comandos para instalação do *composer*, na diretoria /home/user/Documents/proj-csc/project2/:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088ffbc1f233e9b180f061') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Surge assim o ficheiro composer.json, onde deve ser colocado o seguinte texto:

---

[1] https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04#step-3-install-php

[2] https://oauth2.thephpleague.com/

[3] https://github.com/thephpleague/oauth2-server

```
{ "require": {
      "league/oauth2-server": "^6.1",
      "slim/slim": "^3.0”},
  "autoload": {"psr-4": {
                                    "OAuth2ServerExamples\\": "src/include/oauth/",
                                    "League\\OAuth2\\Server\\": "vendor/league/oauth2-server/src/"}}}
```

Realizando o comando seguinte, fica finalizado a instalação das bibliotecas:

```
php composer.phar install
```

Na diretoria “src/include/oauth” devem ser colocados as pastas contidas no seguinte link:

https://github.com/thephpleague/oauth2-server/tree/master/examples/src

Estas fornecem exemplos de implementação das interfaces dos repositórios e também das entidades. Adequando o código destas classes, a utilização das mesmas permite a interligação entre o *Oauth* e a base de dados.

Destes repositórios foram alterados o *ClientRepository*, para incluir o ficheiro que permite a utilização das bibliotecas e o ficheiro que permite interligar à base de dados. Antes da alteração, o repositório do cliente era composto por uma matriz com os respetivos valores associados ao cliente, sendo o repositório feito de forma programática. Após a modificação, o repositório passa a ser uma tabela da base de dados, onde as suas informações são verificadas caso o identificador do cliente seja encontrado na respetiva tabela. O código alterado apresenta-se do seguinte modo:

```
<?php
namespace OAuth2ServerExamples\Repositories;
include __DIR__."/../../../vendor/autoload.php";
include __DIR__."/../include/db_access.php";
use League\OAuth2\Server\Repositories\ClientRepositoryInterface;
use OAuth2ServerExamples\Entities\ClientEntity;

class ClientRepository implements ClientRepositoryInterface
{
   public function getClientEntity($clientIdentifier, $grantType, $clientSecret = null, $mustValidateSecret = true)
   {

                        $found_client = false;
                        $connection = $_SESSION['conn'];
                        $result = $connection->query("SELECT * from clients;");
                        foreach($result as $row){
                                $client_id = $row['client_id'];
                                $client_secret = $row['client_secret'];
                                $client_confidential = $row['is_confidential'];
                                $client_name = $row['name'];
                                $client_uri = $row['redirect_uri'];
                                if ($client_id === $clientIdentifier){
                                        $found_client = true;
                                        if (
                                           $mustValidateSecret === true
                                           && $client_confidential === 'true'
                                           && password_verify($clientSecret, $client_secret) === false
                                        ) {
                                          return;
                                        }
                                }
                        }
                        if($found_client === false){
                                return;
                        }
      $client = new ClientEntity();
      $client->setIdentifier($client_id);
      $client->setName($client_name);
```

```
        $client->setRedirectUri($client_uri);
        return $client;
    }
}
```

Também o UserRepository sofre alterações, tendo-se também incluído o mesmo ficheiro para carregar as bibliotecas desejadas. À semelhança do caso anterior, também o repositório de utilizadores era composto por uma matriz, neste caso com utilizadores e os seus dados respetivos. As alterações implementadas realizam um query à base de dados com os utilizadores e verifica se existe algum utilizador com a password que foi introduzida. O código alterado resulta em:

```
<?php
namespace OAuth2ServerExamples\Repositories;
include __DIR__."/../../../../vendor/autoload.php";
use League\OAuth2\Server\Entities\ClientEntityInterface;
use League\OAuth2\Server\Repositories\UserRepositoryInterface;
use OAuth2ServerExamples\Entities\UserEntity;
class UserRepository implements UserRepositoryInterface
{
    public function getUserEntityByUserCredentials(
        $username,
        $password,
        $grantType,
        ClientEntityInterface $clientEntity
    ) {
                        $connection = $_SESSION['conn'];
                        //queries users from database resource
                        $result = $connection->query('select * from users;');
                        //verifies username and password
                        foreach($result as $row) {
                            $salt = $row['salt'];
                            $user = $row['name'];
                            $pass = $row['password'];
                            $id = $row['id'];
                             if ($username === $user && $pass === md5($salt . $password)) {
                                    $new_user = new UserEntity();
                                    $new_user->setIdentifier($id);
                                    return $new_user;
                              }
                    }
        return;
    }
}
```

Relativamente às entidades, foi alterada a UserEntity de forma a poder implementar as funções associadas à EntityInterface utilizadas para colocar o valor do identificador do objeto, de forma a poder identifica-lo na base de dados.

O ficheiro php db_access.php implementa uma simples conexão ao servidor sql e coloca a conexão numa variável global de sessão. O código surge então da seguinte forma:

```
<?php
    $host = "localhost";
    $user = "user";
    $pass = "inseguro2";
    $dsn = "mysql:host=$host; dbname=project";
    try {
          $connection = new PDO($dsn, $user, $pass);
    }
    catch (PDOException $exception) {
          echo("<p>Error: ");
          echo($exception->getMessage());
          echo("</p>");
          exit();
```

```
    }
    $_SESSION['conn'] = $connection;
?>
```

Por fim, na diretoria "src/public/" foi inserida uma cópia do exemplo fornecido no link do github[4]. A única alteração realizada foi na introdução da chave pública no servidor de autorização:

```
'lxZFUEsBCJ2Yb14IF2ygAHI5N4+ZAUXXaSeeJm6+twsUmIen' // encryption key
```

Por:

```
'file://' . __DIR__ . '/../public.key'      // encryption key
```

O nome do ficheiro foi alterado de *password.php* para *index.php*. Surge assim o código:

```
<?php
include __DIR__ . "/../../vendor/autoload.php";
include __DIR__ . "/../include/db_access.php";
use League\OAuth2\Server\AuthorizationServer;
use League\OAuth2\Server\Exception\OAuthServerException;
use League\OAuth2\Server\Grant\PasswordGrant;
use OAuth2ServerExamples\Repositories\AccessTokenRepository;
use OAuth2ServerExamples\Repositories\ClientRepository;
use OAuth2ServerExamples\Repositories\RefreshTokenRepository;
use OAuth2ServerExamples\Repositories\ScopeRepository;
use OAuth2ServerExamples\Repositories\UserRepository;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;
use Slim\App;
$app = new App([
    // Add the authorization server to the DI container
    AuthorizationServer::class => function () {
        // Setup the authorization server
        $server = new AuthorizationServer(
            new ClientRepository(),              // instance of ClientRepositoryInterface
            new AccessTokenRepository(),         // instance of AccessTokenRepositoryInterface
            new ScopeRepository(),               // instance of ScopeRepositoryInterface
            'file://' . __DIR__ . '/../private.key',   // path to private key
            'file://' . __DIR__ . '/../public.key'     // encryption key
        );
        $grant = new PasswordGrant(
            new UserRepository(),          // instance of UserRepositoryInterface
            new RefreshTokenRepository()   // instance of RefreshTokenRepositoryInterface
        );
        $grant->setRefreshTokenTTL(new \DateInterval('P1M')); // refresh tokens will expire after 1 month
        // Enable the password grant on the server with a token TTL of 1 hour
        $server->enableGrantType(
            $grant,
            new \DateInterval('PT1H') // access tokens will expire after 1 hour
        );
        return $server;
    },
]);
$app->post(
    '/access_token',
    function (ServerRequestInterface $request, ResponseInterface $response) use ($app) {
        /* @var \League\OAuth2\Server\AuthorizationServer $server */
        $server = $app->getContainer()->get(AuthorizationServer::class);
        try {
            // Try to respond to the access token request
            return $server->respondToAccessTokenRequest($request, $response);
        } catch (OAuthServerException $exception) {
            // All instances of OAuthServerException can be converted to a PSR-7 response
            return $exception->generateHttpResponse($response);
        } catch (\Exception $exception) {
            // Catch unexpected exceptions
            $body = $response->getBody();
```

[4] https://github.com/thephpleague/oauth2-server/blob/master/examples/public/password.php

```
        $body->write($exception->getMessage());
        return $response->withStatus(500)->withBody($body);
      }
    }
);
$app->run();
```

Este código é relativo à implementação de um servidor de autenticação utilizando o Password Grant Type, sugerido para a implementação de aplicações móveis quando a aplicação e o servidor são de confiança, sem terceiros envolvidos.
Como a instanciação deste AuthorizationServer requer a criação de chaves pública e privada, na diretoria "src/" é necessário correr o seguinte comando no terminal:

```
Openssl genrsa -out private.key 2048
```

Para gerar a chave pública é preciso também correr o seguinte comando no terminal:

```
Openssl rsa -in private.key -pubout -out public.key > public.key
```

Voltando à diretoria "src/public/" é necessário criar um ficheiro chamado ".htaccess" e introduzir neste:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule . index.php [L]
```

Por fim, é necessário correr na base de dados o ficheiro create_*table*.*sql*, responsável pela criação das tabelas *users* e *clientes*, tabelas essas que contém informação relevante dos utilizadores e dos clientes permitidos, respetivamente. Na tabela *clientes* é inserido uma linha com a informação da aplicação móvel, sendo que o cliente_*secret* corresponde à *hash* da *password* 'abc123'. O código que compõe o ficheiro é o seguinte:

```
DROP TABLE IF EXISTS users;

DROP TABLE IF EXISTS clients;

CREATE TABLE users (

  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,

  name VARCHAR(255) NOT NULL DEFAULT '',

  password VARCHAR(255) NOT NULL DEFAULT '',

  salt VARCHAR(255) NOT NULL DEFAULT '',

  created DATE NOT NULL DEFAULT '1111-11-11',

  PRIMARY KEY (id)

);

CREATE TABLE clients (

  client_id VARCHAR(255) NOT NULL DEFAULT '',

  client_secret VARCHAR(255) NOT NULL DEFAULT '',

  name     VARCHAR(255) NOT NULL DEFAULT '',

  redirect_uri VARCHAR(255) NOT NULL DEFAULT '',

  is_confidential VARCHAR(255) NOT NULL DEFAULT '',

  PRIMARY KEY (client_id)

);
```

INSERT INTO clients values("projetocsc", "$2y$10$BiXbl/B3IzpRroX3f07R1uoa7Wk649IjBbnyTgIHgtpBHsI2CJWa.", "Projecto CSC", "http://foo//bar", "true");

Todo o código relativo à implementação do servidor pode ser encontrado em:

https://github.com/pcarmo/proj-csc/tree/master/project2

Código de Android

SplashActivity

Código de apresentação da página inicial.

```
package istecnico.csc;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.CountDownTimer;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.LinearLayout;

public class SplashActivity extends AppCompatActivity {

LinearLayout slinearLayout, elinearLayout;

@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_splash);

slinearLayout  = (LinearLayout) findViewById(R.id.init_layer);
elinearLayout = (LinearLayout) findViewById(R.id.end_layer);

elinearLayout.setVisibility(View.INVISIBLE);

new CountDownTimer(6000, 1000) {
public void onTick(long millisUntilFinished){}
public void onFinish() {
slinearLayout.setVisibility(View.INVISIBLE);
elinearLayout.setVisibility(View.VISIBLE);
}
}.start();
}

public void onClickPlay(View view) {
final Animation animAlpha = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
view.startAnimation(animAlpha);
Intent i = new Intent(SplashActivity.this, MainActivity.class);
startActivity(i);
}

@Override
public void onBackPressed() {
new AlertDialog.Builder(this)
.setIcon(android.R.drawable.ic_dialog_alert)
.setTitle("Closing APP")
.setMessage("Are you sure you want to close this app?")
.setPositiveButton("Yes...", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
finish();
}

})
.setNegativeButton("No!", null)
.show();
}
}
```

MainActivity

Interface que permite o registo, login de utilizadores e dispõe informação sobre a aplicação.

```
package istecnico.csc;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;

public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}

public void onClickRegister(View view){
final Animation animAlpha = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
view.startAnimation(animAlpha);
Intent i = new Intent(MainActivity.this, Register.class);
startActivity(i);
}

public void onClickSignIn(View view){
final Animation animAlpha = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
view.startAnimation(animAlpha);
Intent i = new Intent(MainActivity.this, LogIn.class);
startActivity(i);
}

public void onClickAbout(View view){
final Animation animAlpha = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
view.startAnimation(animAlpha);
Intent i = new Intent(MainActivity.this, About.class);
startActivity(i);
}
}
```

RegisterActivity

Interface que permite o registo de um utilizador no servidor de autenticação. Exitem três campos de entrada (edit text) onde é possível escolher o endreço do servidor (IP), username e password. Após o preenchimento do formulário pelo utilizador o pedido http (HTTP POST) de registo no servidor de autenticação é formulado usando a ferramenta volley.

```
package istecnico.csc;

import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkError;
import com.android.volley.ParseError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.ServerError;
import com.android.volley.TimeoutError;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.util.HashMap;
import java.util.Map;

public class Register extends AppCompatActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_register);
}

public void onClickRegister_r(View view) {
String ip_address = getTextString(R.id.serverIPInput_r);
final String username = getTextString(R.id.userNameInput_r);
final String userpass = getTextString(R.id.pwInput_r);
final TextView output = (TextView)findViewById(R.id.resultText_r);
output.setMovementMethod(new ScrollingMovementMethod());

RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://"+ip_address+"/register.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new
Response.Listener<String>() {
@Override
public void onResponse(String response) {

parse_str_register(response);
}

}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error){
if(error instanceof NetworkError)
output.setText("error: network!");
else if( error instanceof ServerError)
output.setText("error: server!");
else if( error instanceof AuthFailureError)
output.setText("error: auth failure!");
else if( error instanceof ParseError)
output.setText("error: parse!");
else if( error instanceof TimeoutError)
output.setText("error: time out!");
else
output.setText("Default: that didn't work!");
}
}){
```

```
@Override
protected Map<String, String> getParams() throws AuthFailureError {
Map<String,String> params=new HashMap<String,String>();
params.put("requester_name",username);
params.put("requester_pass",userpass);
return params;

}
};
queue.add(stringRequest);
}

private String getTextString(int id) {
EditText targetText = (EditText) findViewById(id);
return targetText.getText().toString();
}

public void parse_str_register(String str){
if(str.contains("1")) {
AlertDialog.Builder builder;
builder = new AlertDialog.Builder(Register.this, R.style.AlertDialogCustom);
builder.show();
builder.setTitle("Success!")
.setMessage("Successfully registered!")
.setIcon(R.drawable.success)
.show();
}else{
AlertDialog.Builder builder;
builder = new AlertDialog.Builder(Register.this, R.style.AlertDialogCustom);
builder.show();
builder.setTitle("Error!")
.setMessage("User name already in use!")
.setIcon(R.drawable.error)
.show();
}
}
}
```

LoginActivity

Tal como no registo, a aplicação pede para utilizador para preencher o IP, username e password. De seguida pelo método de obtenção do token (password grant), é formado e enviado ao servidor de autenticação o pedido de token em http. O pedido http tem o formato:

```
curl -X "POST" "http://localhost:4444/password.php/access_token" \
          -H "Content-Type: application/x-www-form-urlencoded" \
          -H "Accept: 1.0" \
          --data-urlencode "grant_type=password" \
          --data-urlencode "client_id=myawesomeapp" \
          --data-urlencode "client_secret=abc123" \
          --data-urlencode "username=alex" \
          --data-urlencode "password=whisky" \
          --data-urlencode "scope=basic email"
```

Se o pedido for aceite pelo servidor de autenticação (credenciais correctas) este irá devolver uma mensagem que contém o access token, o tipo, e a data de validade do token. Esse access token é enviado para obter acesso ao resource server pretendido.

```
package istecnico.csc;

import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkError;
import com.android.volley.ParseError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.ServerError;
import com.android.volley.TimeoutError;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class LogIn extends AppCompatActivity {

String str_type, str_token;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_log_in);
}

public void onClickLogInButton(View view) throws IOException {
final String ip_address = getTextString(R.id.serverIPInput);
final String username = getTextString(R.id.userNameInput);
final String userpass = getTextString(R.id.pwInput);
final TextView output = (TextView)findViewById(R.id.resultText);
output.setMovementMethod(new ScrollingMovementMethod());

final RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://"+ip_address+"/access_token";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new
Response.Listener<String>() {
@Override
public void onResponse(String response) {
parse_str_token(response);

String url ="http://"+ip_address+"/resource.php/users";
StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new
```

```
Response.Listener<String>() {

@Override
public void onResponse(String response) {
String[] str_split = response.split("\"");

AlertDialog.Builder builder;
builder = new AlertDialog.Builder(LogIn.this, R.style.AlertDialogCustom);
builder.show();
builder.setTitle("Success!")
.setMessage("Wellcome " + str_split[1] + "!")
.setIcon(R.drawable.success)
.show();
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error){
String message = "";

if(error instanceof NetworkError)
message="Message: network!";
else if( error instanceof ServerError)
message="Message: server!";
else if( error instanceof AuthFailureError) {
message="Message: authentication failure!";
}else if( error instanceof ParseError)
message="Message: parse!";
else if( error instanceof TimeoutError)
message="Message: time out!";
else
message="Default message: Something went wrong.";

AlertDialog.Builder builder;
builder = new AlertDialog.Builder(LogIn.this, R.style.AlertDialogCustom);
builder.show();
builder.setTitle("Error - resource server :(")
.setMessage(message)
.setIcon(R.drawable.error)
.show();
}
}){
@Override
protected Map<String, String> getParams() throws AuthFailureError {
Map<String,String> params=new HashMap<String,String>();
return params;
}

@Override
public Map<String, String> getHeaders() throws AuthFailureError {
Map<String,String> headers=new HashMap<String,String>();
headers.put("Authorization",str_type + " " + str_token);
return headers;
}
};
queue.add(stringRequest);
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error){
String message = "";

if(error instanceof NetworkError)
message="Message: network!";
else if( error instanceof ServerError)
message="Message: server!";
else if( error instanceof AuthFailureError)
message="Message: authentication failure!";
else if( error instanceof ParseError)
message="Message: parse!";
else if( error instanceof TimeoutError)
message="Message: time out!";
else
message="Default message: Something went wrong.";

AlertDialog.Builder builder;
builder = new AlertDialog.Builder(LogIn.this, R.style.AlertDialogCustom);
```

```
builder.show();
builder.setTitle("Error authentication server :(")
.setMessage(message)
.setIcon(R.drawable.error)
.show();
}
}){
@Override
protected Map<String, String> getParams() throws AuthFailureError {
Map<String,String> params=new HashMap<String,String>();
params.put("grant_type","password");
params.put("client_id","myawesomeapp");
params.put("client_secret", "abc123");
params.put("username",username);      //alex
params.put("password",userpass);      //whisky
params.put("scope","basic email");
return params;

}

@Override
public Map<String, String> getHeaders() throws AuthFailureError {
Map<String,String> headers=new HashMap<String,String>();
headers.put("Accept","1.0");
headers.put("Content-Type","application/x-www-form-urlencoded");
return headers;
}
};
queue.add(stringRequest);
}

private String getTextString(int id) {
EditText targetText = (EditText) findViewById(id);
return targetText.getText().toString();
}

public void parse_str_token(String str){
String[] str_split = str.split("\"");
str_type = str_split[3];
str_token = str_split[9];
}
}
```

Todo o Código relativo à implementação da aplicação móvel pode ser encontrado em:

https://github.com/pcarmo/proj-csc/tree/master/CSC