OAuth

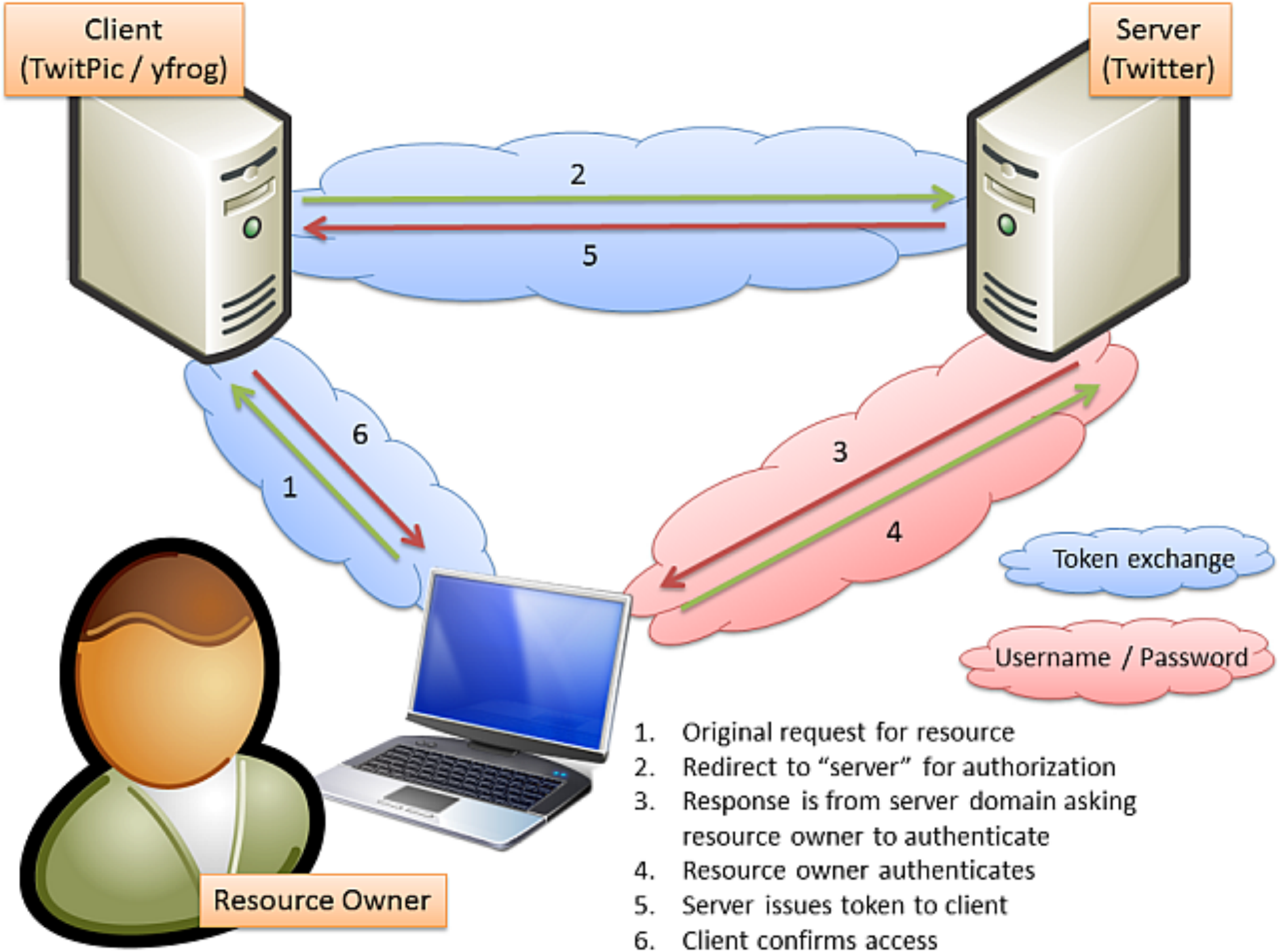
Martin Kuba, ÚVT MU

Co je OAuth

- otevřený standard specifikující protokol pro **autorizaci** přístupu k vyjmenovaným operacím nějakého API, ale lze jej využít i pro autentizaci v případě, že dané API má operace pro získání informací o uživateli
- umožňuje povolit pro konkrétního **poskytovatele služby** jen určité operace (ze všech operací) na API určitého poskytovatele API
- podporují API Google, Facebook, Twitter, ...
- seznam implementací - http://oauth.net/2/

Co umožňuje OAuth

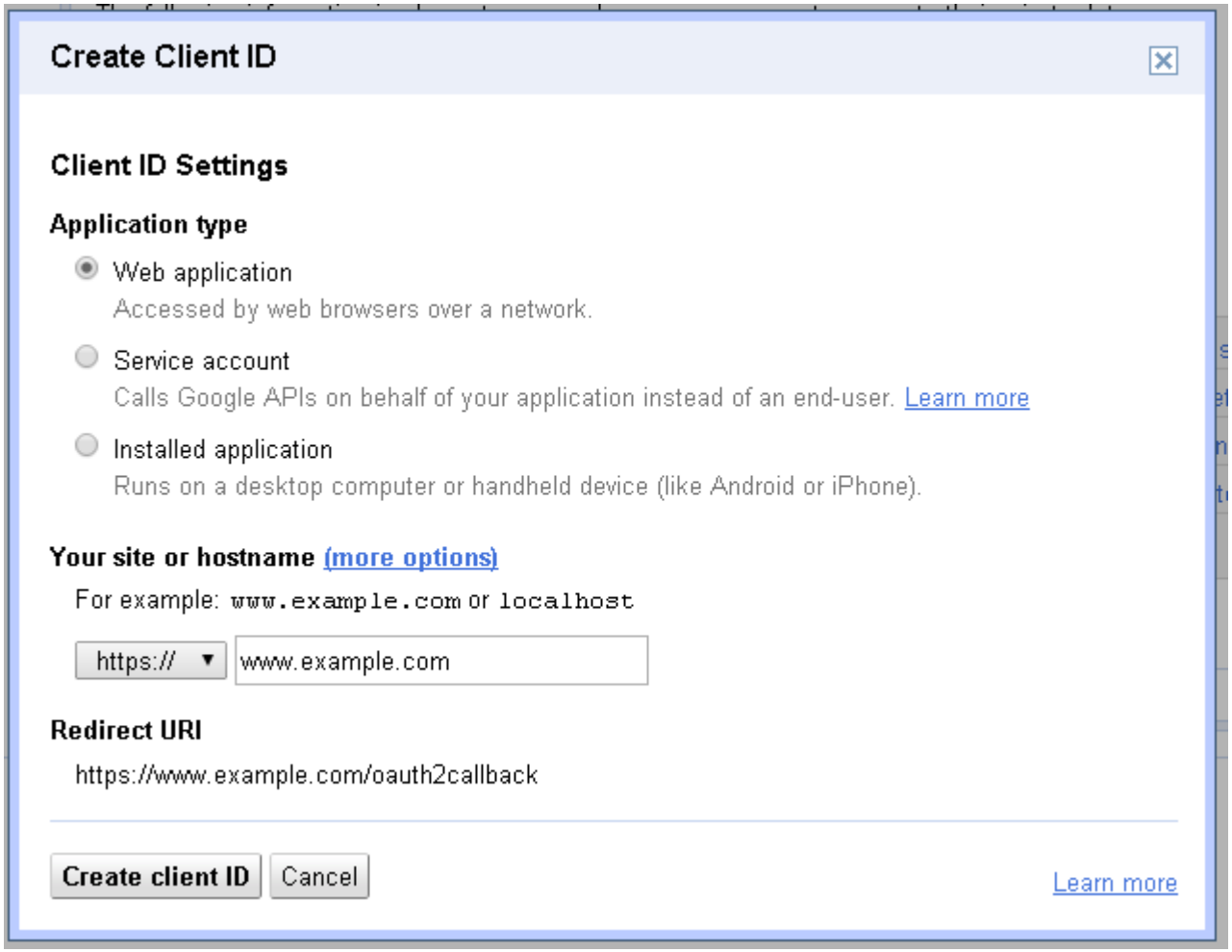
- není omezeno jen na web, lze i pro mobilní aplikace (Android, iOS), desktopové, SmartTV, embedded v set-top-boxech
- spolupráce dvou různých web serverů
- např. uživatel Google Picasa může povolit jinému webu od firmy X, případně jejich mobilní aplikaci, čtení obrázků a zápis jejich upravených verzí
- aplikace může přistupovat k API i bez uživatele


Client
(TwitPic / yfrog)
Server
(Twitter)
2
5
6
1
3
4
Token exchange
Username / Password
Resource Owner
1. Original request for resource
2. Redirect to "server" for authorization
3. Response is from server domain asking resource owner to authenticate
4. Resource owner authenticates
5. Server issues token to client
6. Client confirms access

Jak to funguje

1. vývojář aplikace se zaregistruje u poskytovatele API
   - Google API console https://code.google.com/apis/console/
   - Facebook developers https://developers.facebook.com/apps/
2. zaregistruje aplikaci, získá *client_secret*
3. při příchodu uživatele do aplikace přesměruje na OAuth server se žádostí o oprávnění k určitým operacím
4. aplikace získá od uživatele jednorázový kód
5. aplikace vymění kód a *client_secret* za token
6. aplikace volá API a prokazuje se tokenem

Registrace aplikace u Google



Zaregistrovaná aplikace u Google

Google apis

MUNI Photometric Arch... ▾

- Overview
- Services
- Team
- API Access

API Access

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key allows you to exceed anonymous limits by connecting requests back to your project.

Authorized API Access

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 20 client IDs. Learn more

Branding information

The following information is shown to users whenever you request access to their private data.

Product name:	MUNI Photometric Archive
Google account:	martinkuba@gmail.com
Home page URL:	https://www.cerit-sc.cz/login/

Edit branding information...

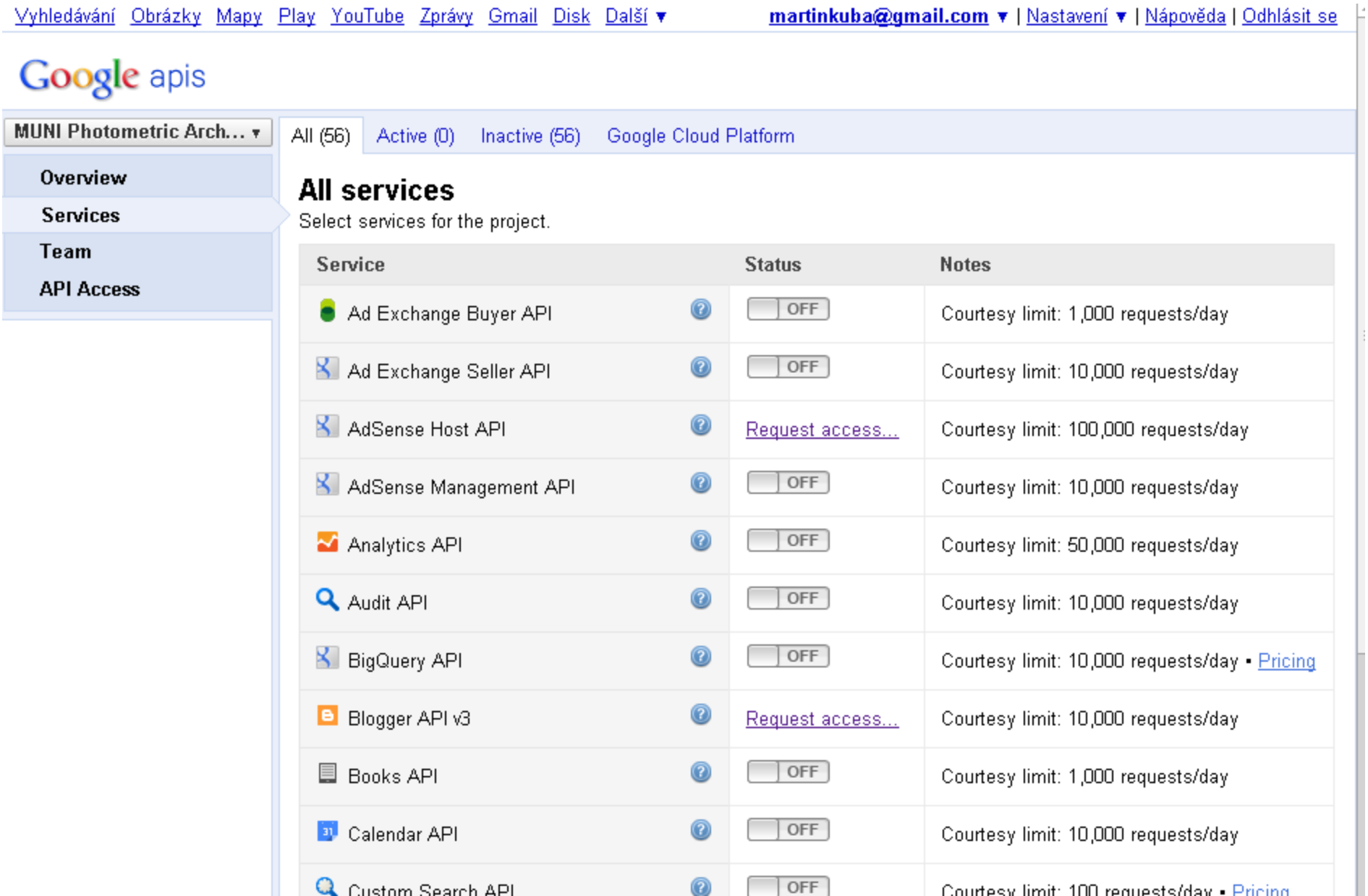
Client ID for web applications

Client ID:	558708443072.apps.googleusercontent.com
Email address:	558708443072@developer.gserviceaccount.com
Client secret:	6_woe[redacted]-LxMvG
Redirect URIs:	https://www.cerit-sc.cz/login/google/auth
JavaScript origins:	https://www.cerit-sc.cz

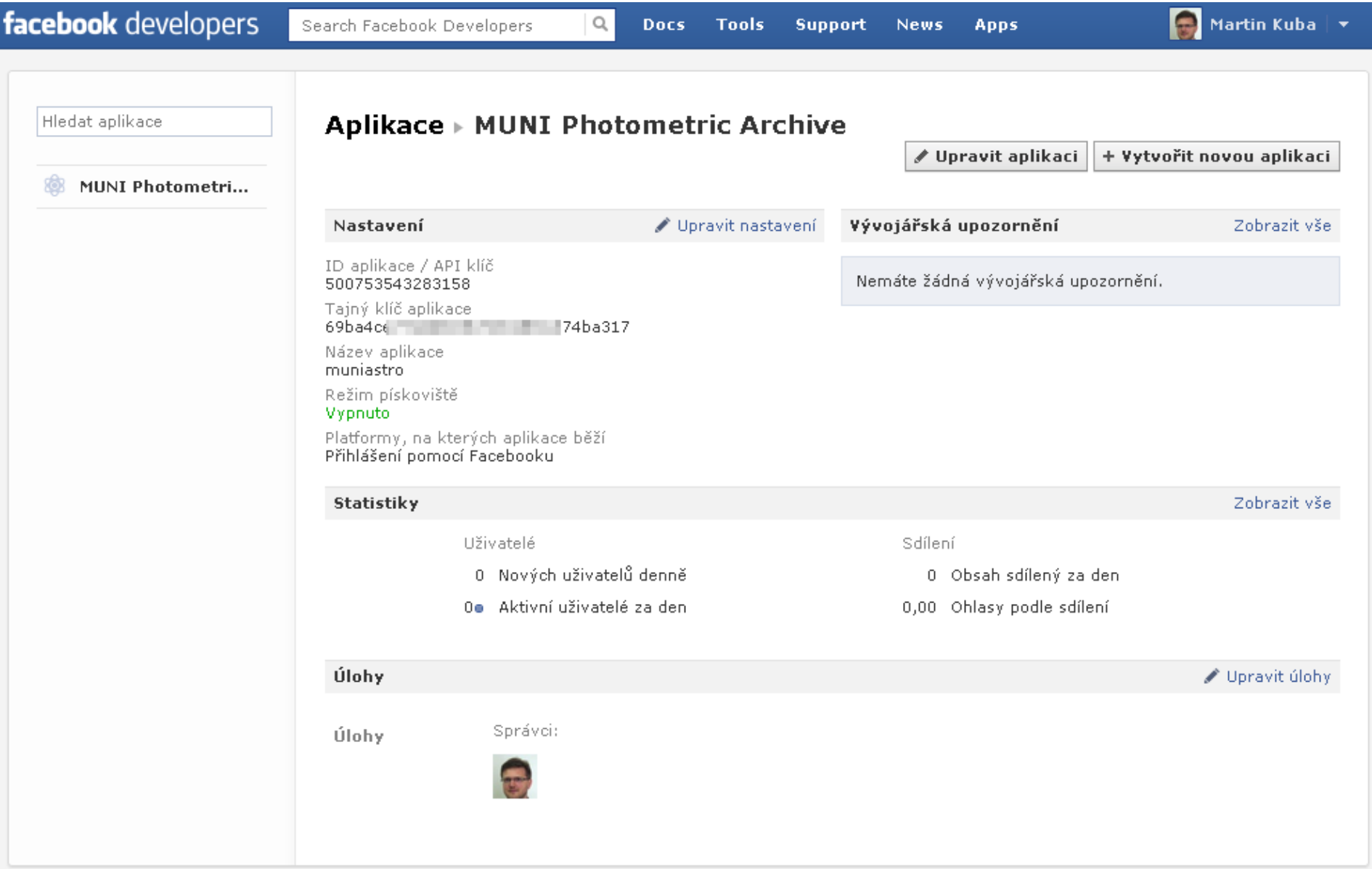
Edit settings...
Reset client secret...
Download JSON
Delete...

Create another client ID...

Povolená API u Google

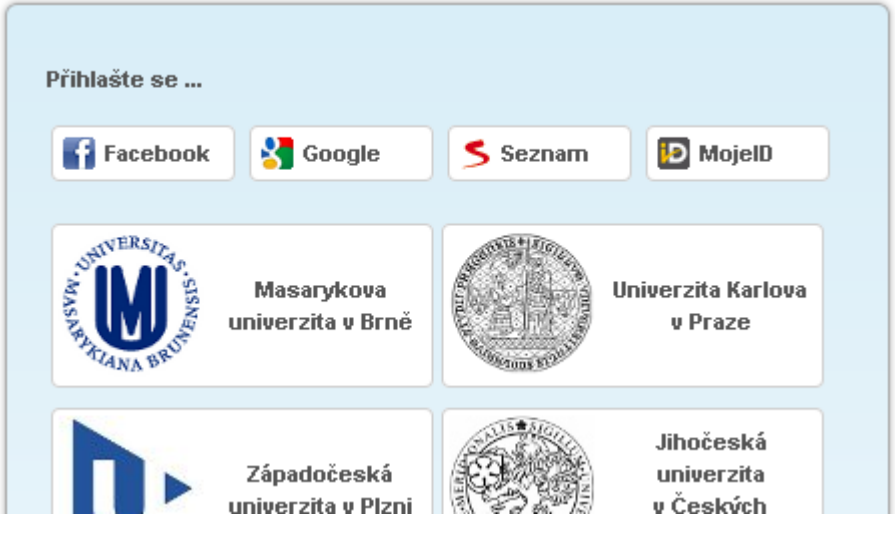


Zaregistrovaná aplikace u Facebooku



Příchod uživatele do aplikace

- uživatel si vybere poskytovatele účtu
- na screenshotu
  - Facebook a Google - OAuth
  - Seznam.cz a MojeID - OpenID
  - MU, UK, ZČU, JU - Shibboleth

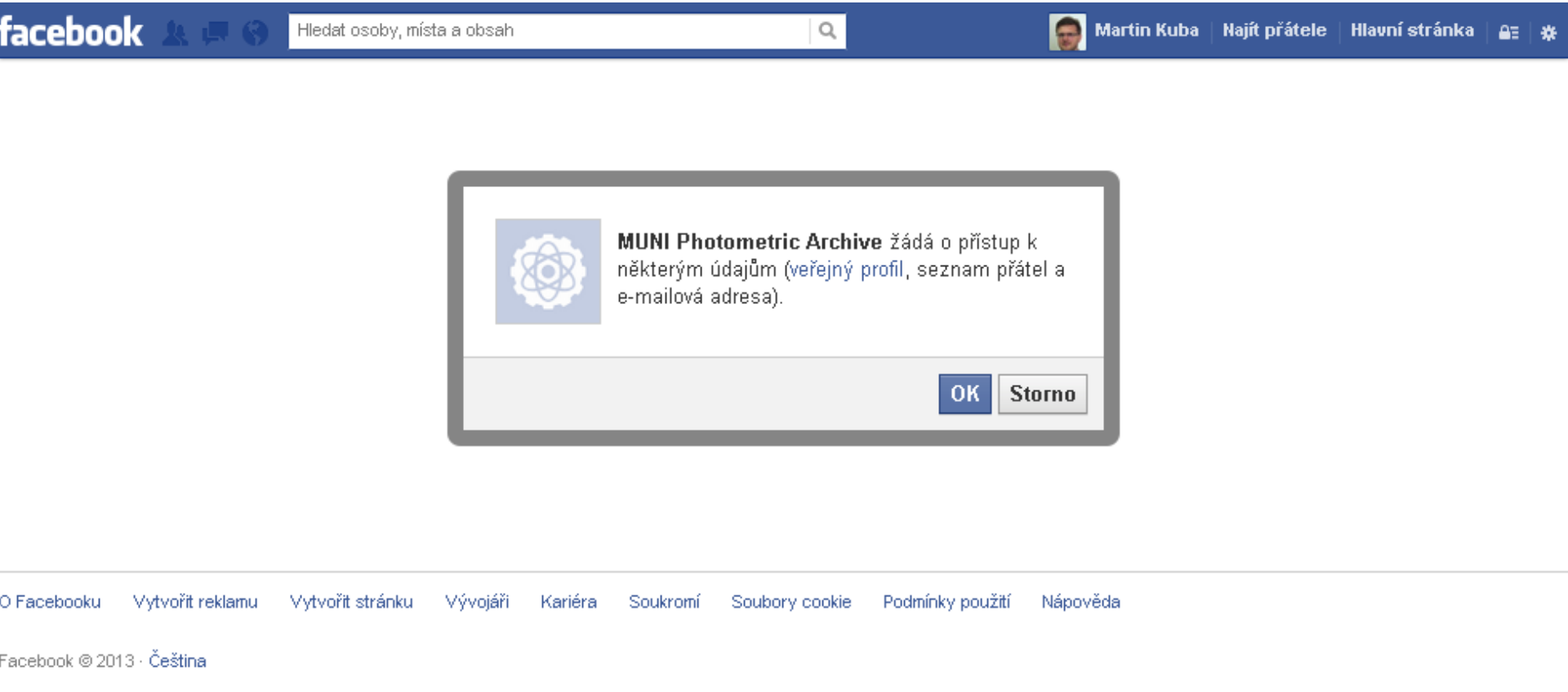


Odeslání uživatele na OAuth server

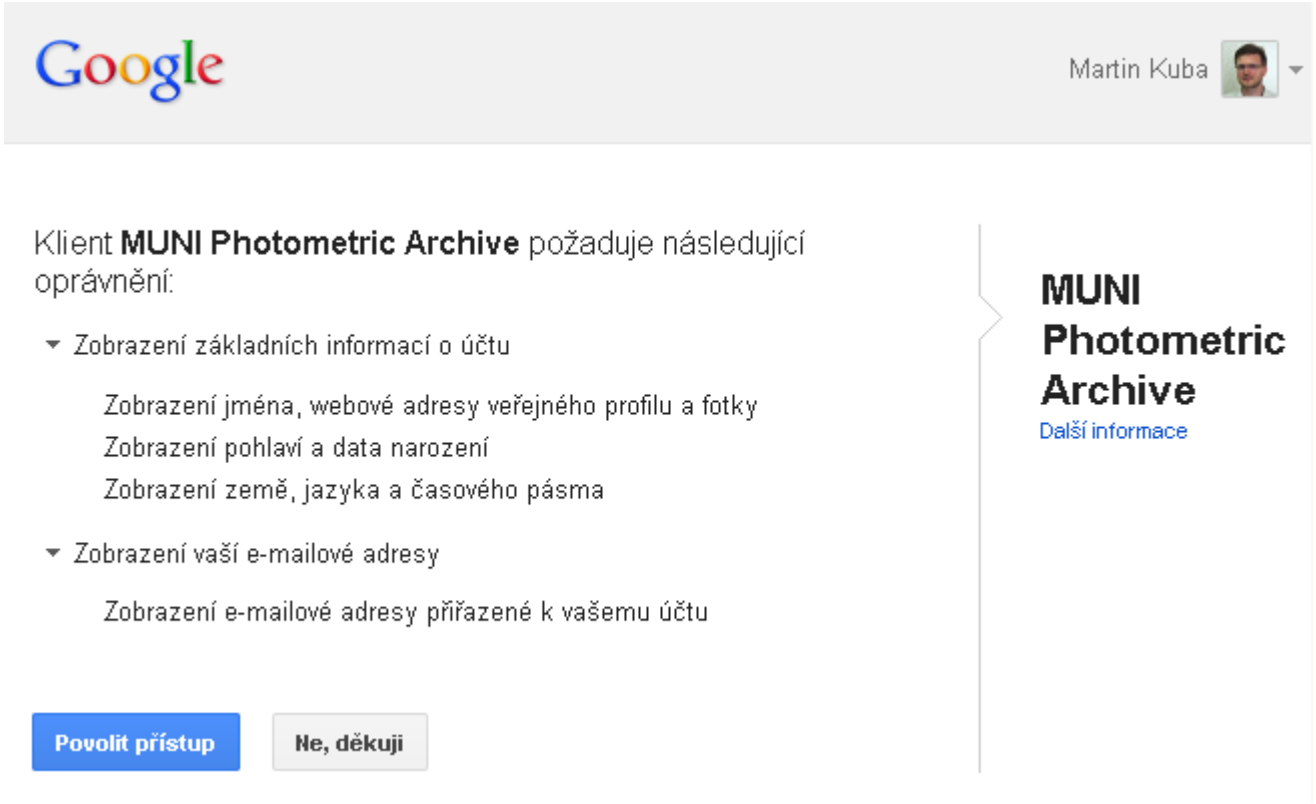
```
public class FacebookServlet extends HttpServlet {

    private static final String client_id = "500753543283158";
    private static final String client_secret = "69ba4c[redacted]ba317";
    private static final String redirect_uri = "https://www.cerit-sc.cz/login/facebook/auth";
    private static final String LOGIN_URL = "https://www.facebook.com/dialog/oauth";
    private static final String TOKEN_URL = "https://graph.facebook.com/oauth/access_token";
    private static final String SCOPE = "email";

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // Step 1
        if ("/login".equals(req.getPathInfo())) {
            //initiate facebook authorization
            String state = Integer.toString(random.nextInt(Integer.MAX_VALUE)); //random value protecting against XSRF
            req.getSession(true).setAttribute("state", state);
            //redirect to Facebook to ask for permission on email
            String loginRedirectURL = LOGIN_URL
                    + "?client_id=" + urlEncode(client_id)
                    + "&redirect_uri=" + urlEncode(redirect_uri)
                    + "&state=" + urlEncode(state)
                    + "&scope=" + urlEncode(SCOPE);
            resp.sendRedirect(loginRedirectURL);
        } else if ("/auth".equals(req.getPathInfo())) {
```

Uživatel se přihlásí k účtu ...

... a schválí povolení k operacím



Obdobně u Google



Aplikace vymění kód od uživatele a vlastní client_secret za token

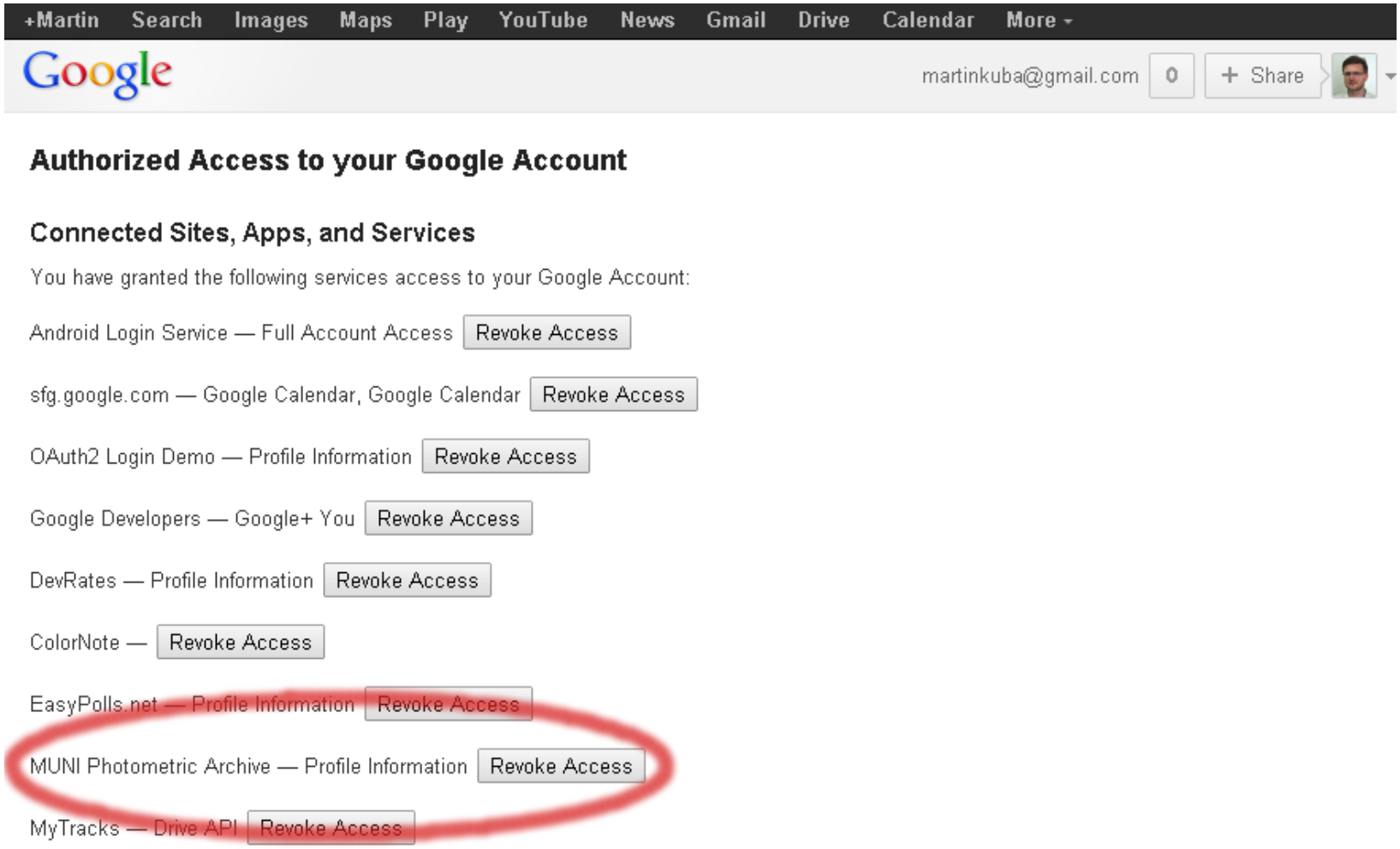
```
} else if ("/auth".equals(req.getPathInfo())) {
//process google authorization
    //check state for XSRF attack
    String state = req.getParameter("state");
    String state1 = (String) req.getSession(true).getAttribute("state");
    if (state == null || !state.equals(state1)) {
        resp.sendError(HttpServletResponse.SC_FORBIDDEN, "state does not match, probably a XSRF attack");
        return;
    }
    //get code
    String code = req.getParameter("code");
    if (code == null) {
        resp.sendError(HttpServletResponse.SC_FORBIDDEN, "code not present");
        return;
    }
    //exchange code for token
    RestTemplate restTemplate = new RestTemplate();
    MultiValueMap<String, String> map = new LinkedMultiValueMap<>();
    map.add("client_id", client_id);
    map.add("client_secret", client_secret);
    map.add("redirect_uri", redirect_uri);
    map.add("code", code);
    map.add("grant_type", "authorization_code");
    JsonNode jsonNode = restTemplate.postForObject(TOKEN_URL, map, JsonNode.class);
    String accessToken = jsonNode.path("access_token").asText();
    String expires = jsonNode.path("expires_in").asText();
    log.debug("accessToken={} expires={}", accessToken, expires);

    /*
```

Aplikace volá API

- aplikace volá určitá URL
- prokazuje se tokenem
- odpověd je obvykle JSON

```
//use token for getting user data
JsonNode userData = restTemplate.getForObject(USER_INFO_URL+"?access_token={access_token}",JsonNode.class,accessToken);
String userId = userData.path("id").asText();
String userEmail = userData.path("email").asText();
String userName = userData.path("name").asText();
```

Uživatel může odebrat oprávnění



Nastavení aplikací

Vaše jméno, profilová fotka, úvodní fotka, pohlaví, sítě, uživatelské jméno a vaše ID číslo jsou vždy dostupné všem uživatelům Facebooku, a to včetně aplikací (důvody). Aplikace navíc mohou přistupovat k vašemu seznamu přátel a ostatním údajům, které jste na svém profilu nastavili jako veřejné.

Aplikace, které používáte

Chcete na Facebooku i jinde používat aplikace, plug-iny, hry a weby? **Zapnuto** Upravit

CiteULikeAuth	Přátelé	Upravit
Cities I've Visited	Přátelé	Upravit
Vimeo	Přátelé	Upravit
Heureka.cz	Přátelé	Upravit
Uplay	Přátelé	Upravit
Geocaching.com	Přátelé	Upravit

MUNI Photometric Archive — Poslední přihlášení: 21 únor — Zavřít

Viditelnost aplikace: Přátelé

Tato aplikace vyžaduje:
- Vaše základní informace [?]
- Vaši e-mailovou adresu (makub@ics.muni.cz)

Poslední přístup k údajům: Základní informace — Dnes
Zobrazit podrobnosti · Další informace

Kdy si přejete být upozorněni? Když vám aplikace pošle upozornění

Odebrat aplikaci · Nahlásit aplikaci

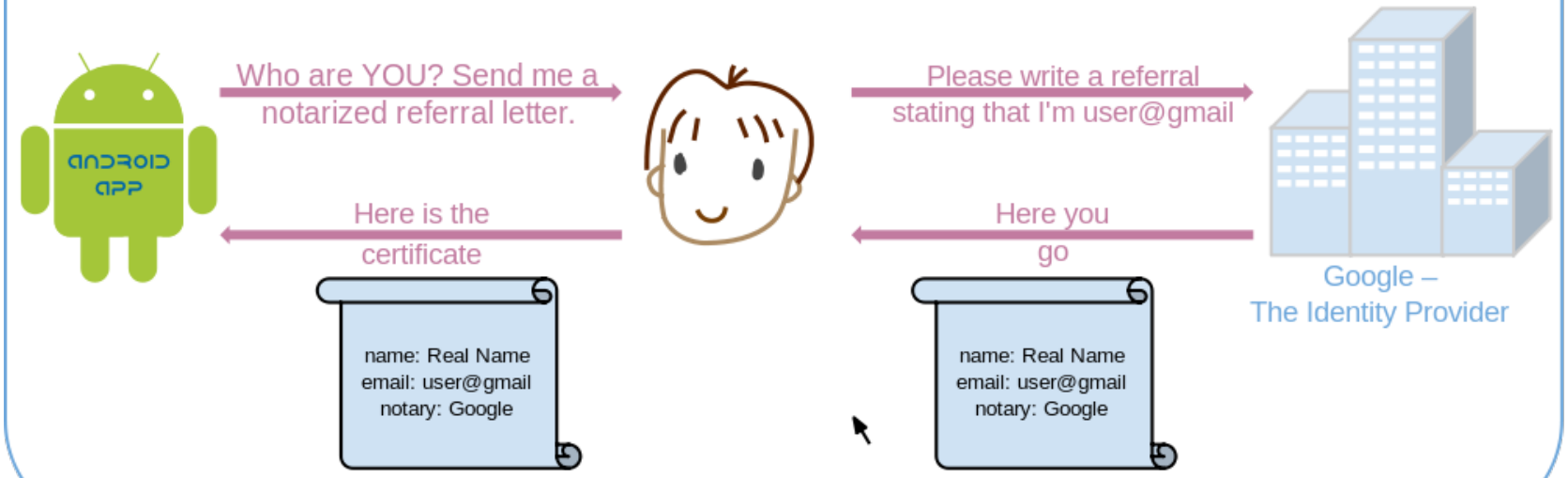
OAuth shrnutí

- uživatel i aplikace jsou zaregistrovány
- oba mají své heslo (client_secret u app)
- poskytovatel API v dokumentaci uvádí možná oprávnění (seznam operací)
- aplikace žádá uživatele o konkrétní oprávnění (povolení k určité množině operací)
- pokud uživatel schválí, aplikace získá token
- uživatel může kdykoliv token revokovat

Srovnání s OpenID

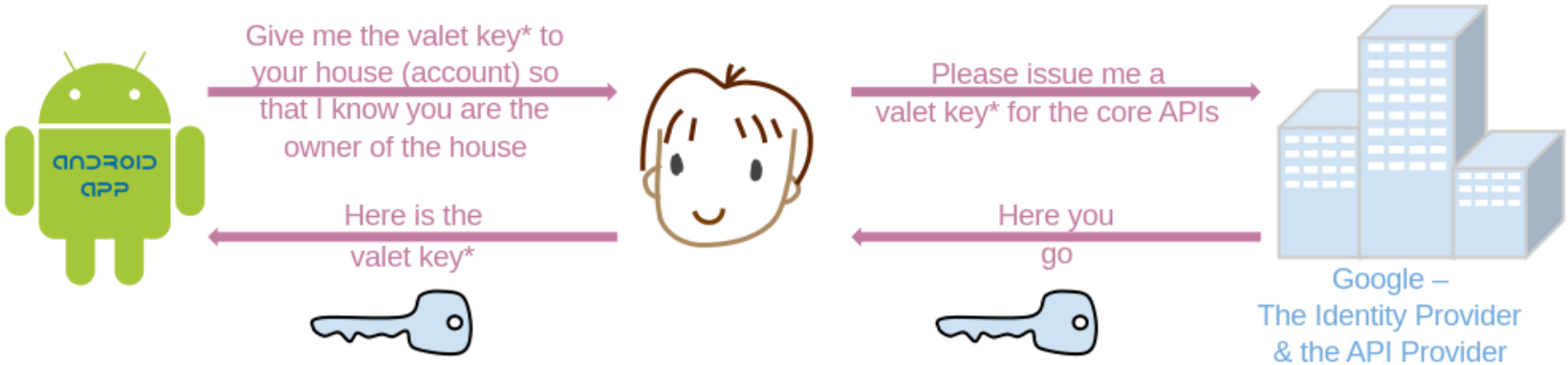
- otevřený standard
  - OpenID - pro autentizaci
  - OAuth - pro autorizaci
- aplikace se registrovat
  - OpenID - nemusí
  - OAuth - musí
- poskytovatel totožnosti
  - OpenID - kdokoliv, ale jak mu věřit ?
  - OAuth - konkrétní, API-specific

OpenID Authentication



vs.

Pseudo-Authentication using OAuth



***valet key = limited scope OAuth Token**

adapted from a drawing by @_nat_en

Srovnání se Shibboleth

- Shibboleth
  - implementace SAML
  - autentizace, jako OpenID
  - potřebuje Discovery Service/WAYF
  - uživatel nemá kontrolu nad vydávanými údaji
  - IdP a SP se musí vzájemně dohodnout
  - SP nemůže požádat uživatele o více informací
  - uApprove - uživatel může schválit všechny nebo nic
- OAuth
  - autorizace
  - autentizace jako nulová autorizace
  - uživatel má kontrolu nad vydávanými oprávněními
  - může je i zpětně revokovat

Konec

Děkuji za pozornost