Webové technológie 10

Aplikácie Webových technológií

Peter Bednár

Zabezpečenie prenosu dát

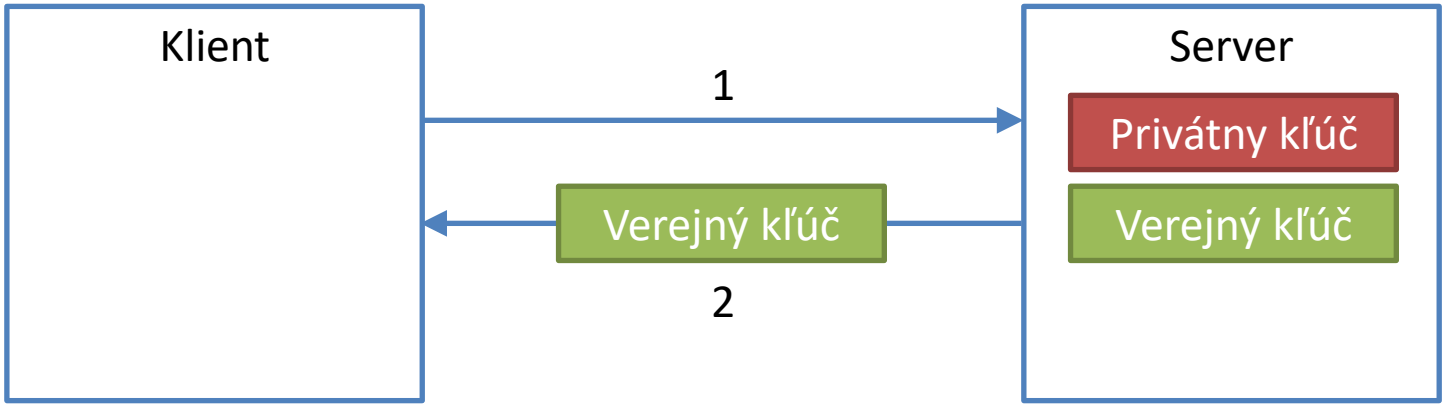
- Pri prenose dát je potrebné dáta zašifrovať do nečitateľnej podoby – dekódovanie je možné len pomocou utajeného kľúča

- Symetrické šifrovanie
  - Správa sa zakóduje a dekóduje pomocou rovnakého kľúča, ktorý musí byť zdieľaný medzi odosielateľom a prijímateľom
- Asymetrické šifrovanie
  - Správa sa zakóduje jedným kľúčom, ktorý pozná odosielateľ a dekóduje druhým, ktorý pozná iba prijímateľ
  - Nevýhoda – kľúče musia byť dostatočne dlhé aby bolo šifrovanie spoľahlivé (zložitejšie kódovanie/dekódovanie)

Protokol HTTPS (1)

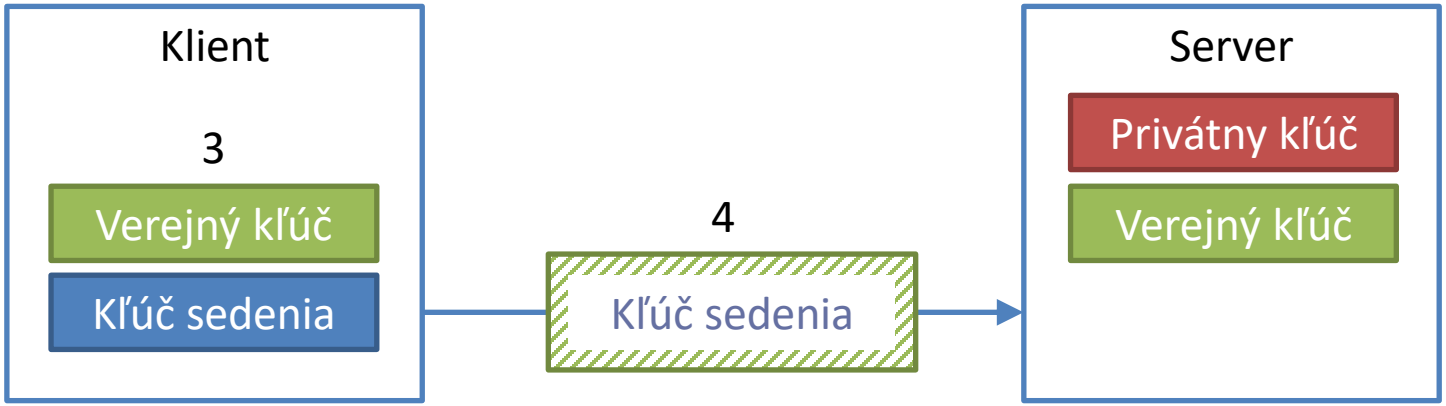
- Protokol SSL/TLS – komunikácia sa inicializuje pomocou asymetrického šifrovania a dáta sa potom prenášajú zakódované symetrickou šifrou
  - Prenos HTTP cez SSL/TLS = HTTPS

1. Klient (prehliadač) odošle nezakódovanú požiadavku na server
2. Server v nezakódovanej odpovedi odošle svoj verejný kľúč



Protokol HTTPS (2)

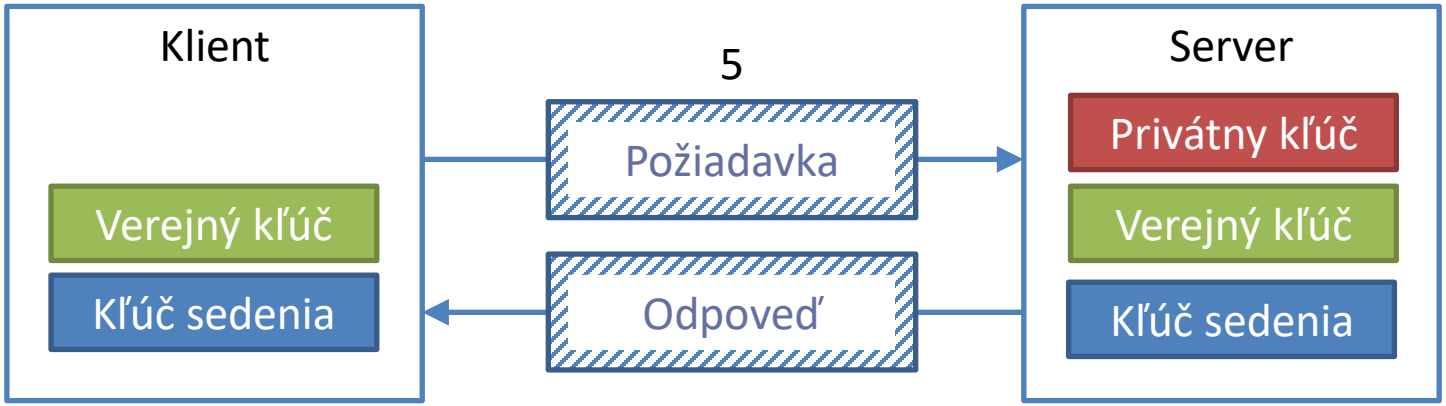
3. Klient vygeneruje náhodný kľúč platný pre jedno sedenie (tzn. pokiaľ používateľ neuzavrie okno prehliadača)
4. Klient zakóduje vygenerovaný kľúč pomocou verejného kľúča servera a odošle ho na server



Protokol HTTPS (3)

5. Klient a server zdieľajú kľúč sedenia na symetrické zakódovanie a dekódovanie dát pri výmene HTTP požiadaviek/odpovedí

- Pre používateľa a pre prenos dát v prostredí JavaScriptu je šifrovanie transparentné – jediný rozdiel je v použití `https` schémy



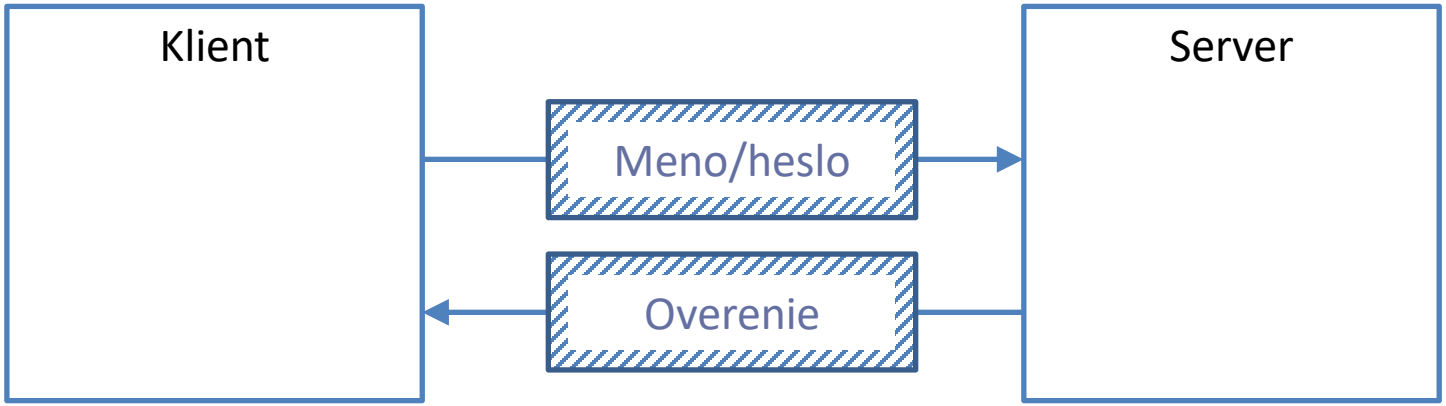
Autentifikácia a autorizácia používateľov

- Autentifikácia
  - Overenie totožnosti používateľa
  - Zvyčajne sa realizuje výmenou zdieľaného tajomstva – napr. používateľské meno/heslo
- Autorizácia
  - Kontrola prístupových práv – ku ktorým dátam má používateľ prístup a aké operácie nad nimi môže vykonávať
  - Zvyčajne sa overuje na strane servera

Autentifikácia na strane servera (1)

- Používateľské mená a heslá sú spravované na servery

1. Klient pre autentifikáciu zobrazí stránku s prihlasovacím formulárom do ktorej používateľ zadá svoje meno a heslo
2. Používateľské meno a heslo sa odošle cez HTTPS na server, ktorý overí totožnosť používateľa



Autentifikácia na strane servera (2)

- Nevýhody
  - Klient musí overiť totožnosť servera
    - Nestačí URL adresa – URL môže byť útočníkom presmerovaná na iný server, ktorý tak získa prístup k používateľským menám a heslám
  - Server udržiava zoznam používateľov a hesiel, musí byť dostatočne zabezpečený
  - Používateľ sa musí zaregistrovať na každom serveri zvlášť – veľa mien/hesiel – používanie jednoduchých hesiel, alebo toho istého hesla na viacerých stránkach znižuje bezpečnosť

Overenie totožnosti servera – SSL certifikáty

- Certifikačná autorita
  - Dôveryhodná organizácia, ktorá overí totožnosť poskytovateľa servera na danej adrese a vydá mu SSL certifikát
- SSL certifikát je súbor, ktorý obsahuje:
  - Informácie o organizácii poskytovateľa a o samotnom certifikáte (napr. dátum vydania a platnosti, URL servera, atď.)
  - Verejný kľúč servera
  - Digitálny podpis certifikačnej autority, ktorý sa používa na overenie platnosti a integrity certifikátu (tzn. nikto nemôže zmeniť informácie v certifikáte bez toho aby sa neporušil digitálny podpis)

Digitálny podpis (1)

- Digitálny podpis je založený na aplikácii hašovacej funkcie a asymetrického šifrovania
- Hašovacia funkcia – pre zadané dáta vypočíta kontrolné číslo – tzv. hašovací kód (*hash*)
  - Pre hašovaciu funkciu musí platiť:
    - Je veľmi zložité vypočítať vstupné dáta z výstupného kódu
    - Je veľmi nepravdepodobné, že sa vygenerujú rovnaké výstupné kódy pre rôzne vstupné dáta

Digitálny podpis (2)

- Postup pri digitálnom podpise:

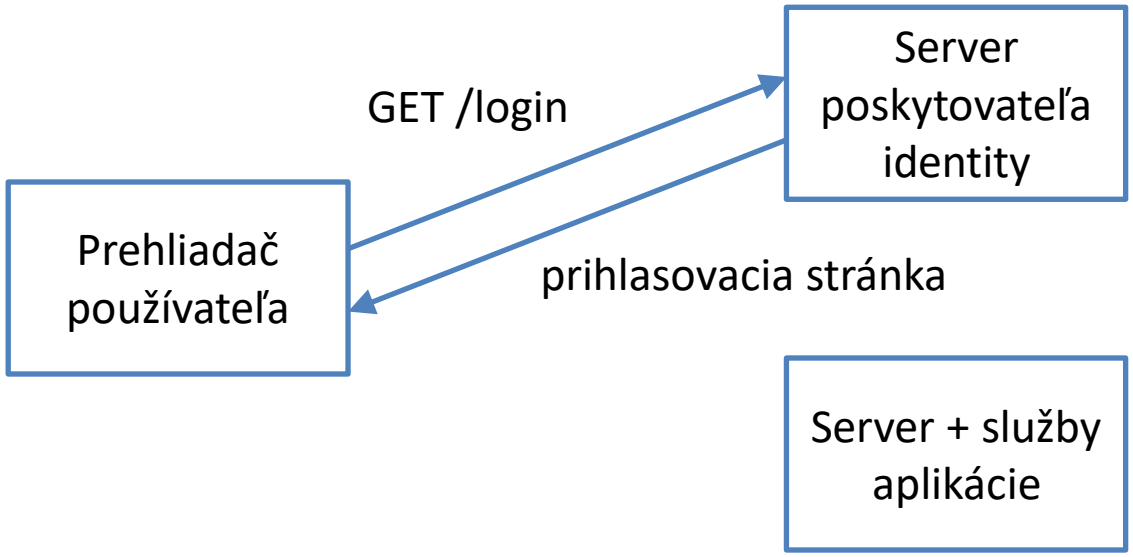
1. Odosielateľ vypočíta hašovací kód dokumentu a zašifruje ho svojim privátnym kľúčom
   - Zašifrovaný hašovací kód tvorí digitálny podpis dokumentu, ktorý sa odošle s dokumentom prijímateľovi
2. Prijímateľ vypočíta hašovací kód dokumentu
3. Prijímateľ dekóduje digitálny podpis dokumentu verejným kľúčom odosielateľa (tzn. získa pôvodný hašovací kód odosielateľa), ak sa hašovacie kódy zhodujú, platí:
   - Dokument bol skutočne podpísaný odosielateľom s daným verejným kľúčom (tzn. klient si môže overiť identitu odosielateľa)
   - Nikto iný nezmenil obsah dokumentu

Autentifikácia a autorizácia implementovaná poskytovateľom identity

- Väčšina používateľov je už zaregistrovaná na sociálnych sieťach:
  - Chceme aby sa používateľ mohol prihlásiť do našej aplikácie s používateľským menom a heslom zaregistrovaným u poskytovateľa identity
    - Google, Facebook, Microsoft, Twitter, GitHub, ...
- Protokol OpenID Connect/OAuth 2.0
  - Dokumentácia: https://oauth.net/2/
  - Podporuje autentifikáciu používateľa
  - Podporuje autorizáciu na servery pri prístupe k dátam

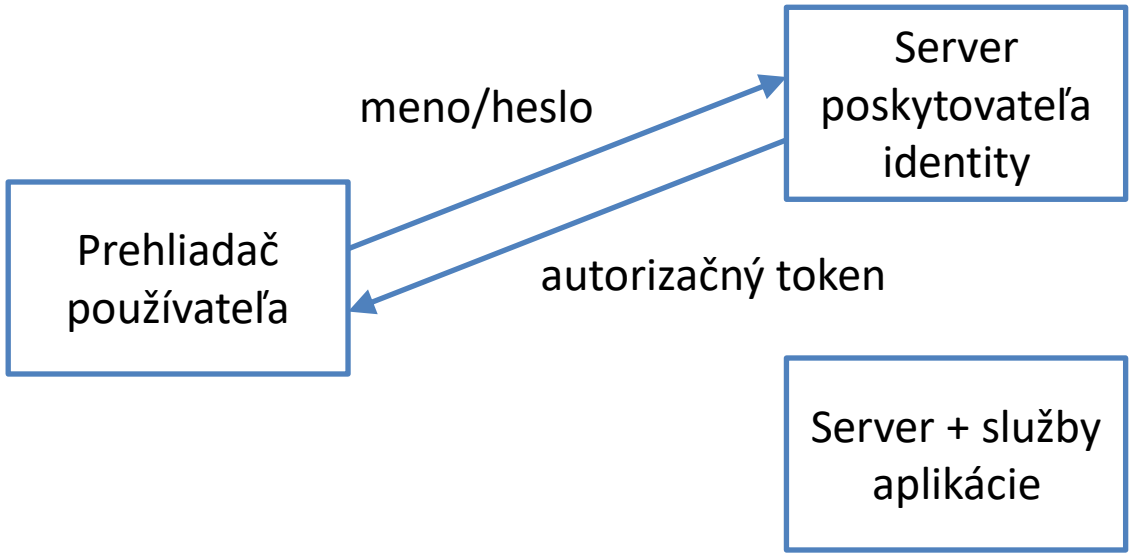
Autentifikácia pomocou OAuth 2.0 (1)

1. Pre prihlásenie používateľa sa zobrazí prihlasovacia stránka poskytovateľa identity



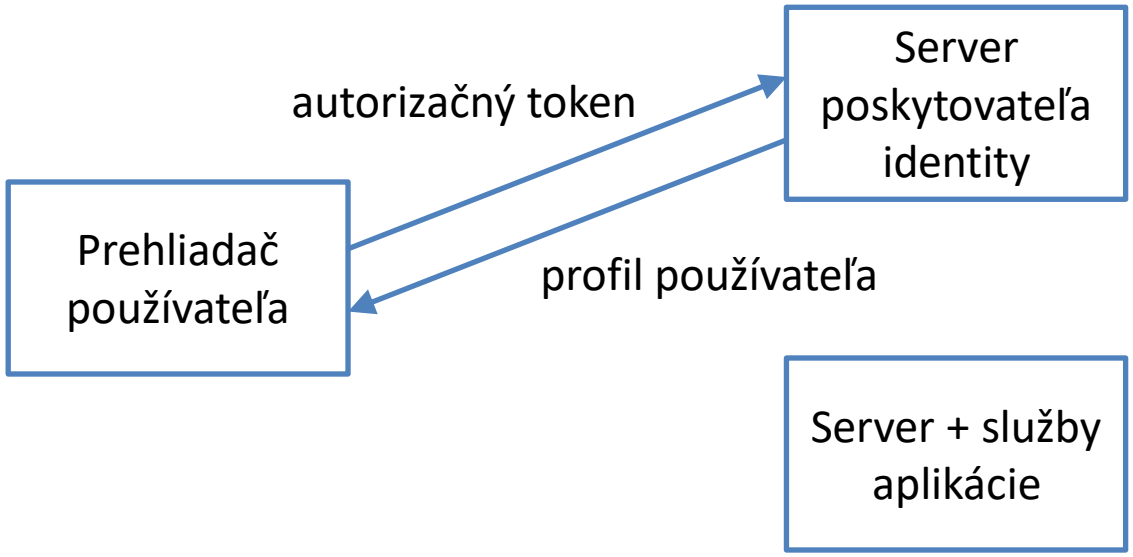
Autentifikácia pomocou OAuth 2.0 (2)

2. Používateľ sa prihlási svojim používateľským menom a heslom na stránke poskytovateľa identity. Server poskytovateľa identity overí jeho totožnosť a po úspešnom prihlásení presmeruje prehliadač na stránku aplikácie, ktorej predá vygenerovaný autorizačný token.



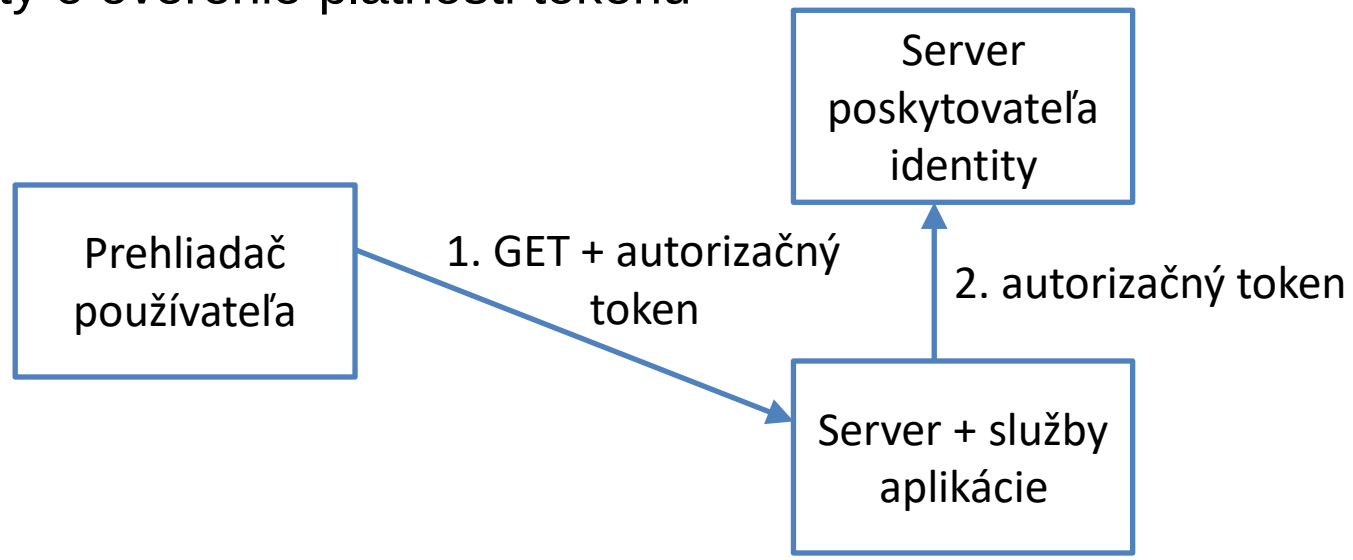
Autentifikácia pomocou OAuth 2.0 (3)

2. Aplikácia môže voliteľne požiadať server poskytovateľa identity o profil používateľa, ktorý obsahuje základné informácie o používateľovi. V požiadavke zašle autorizačný token platný pre dané prihlásenie.



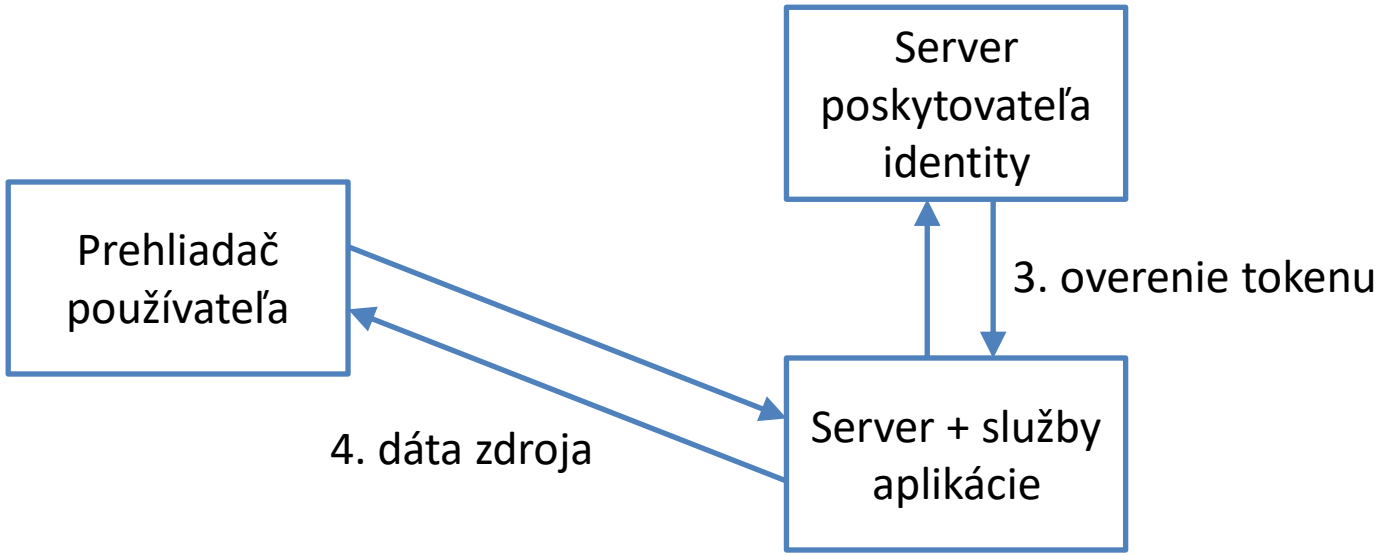
Autorizácia pomocou OAuth 2.0 (1)

1. Stránka aplikácie požiada o zdroj na servery aplikácie (súbor, alebo dáta služby). V požiadavke uvedie autorizačný token platný pre dané prihlásenie
2. Server, alebo služba aplikácie požiada server poskytovateľa identity o overenie platnosti tokenu



Autorizácia pomocou OAuth 2.0 (2)

3. Server poskytovateľa identity overí platnosť tokenu
4. Ak je token platný a prihlásený používateľ má prístupové práva k zdroju, server aplikácie vráti zdroj prehliadaču používateľa

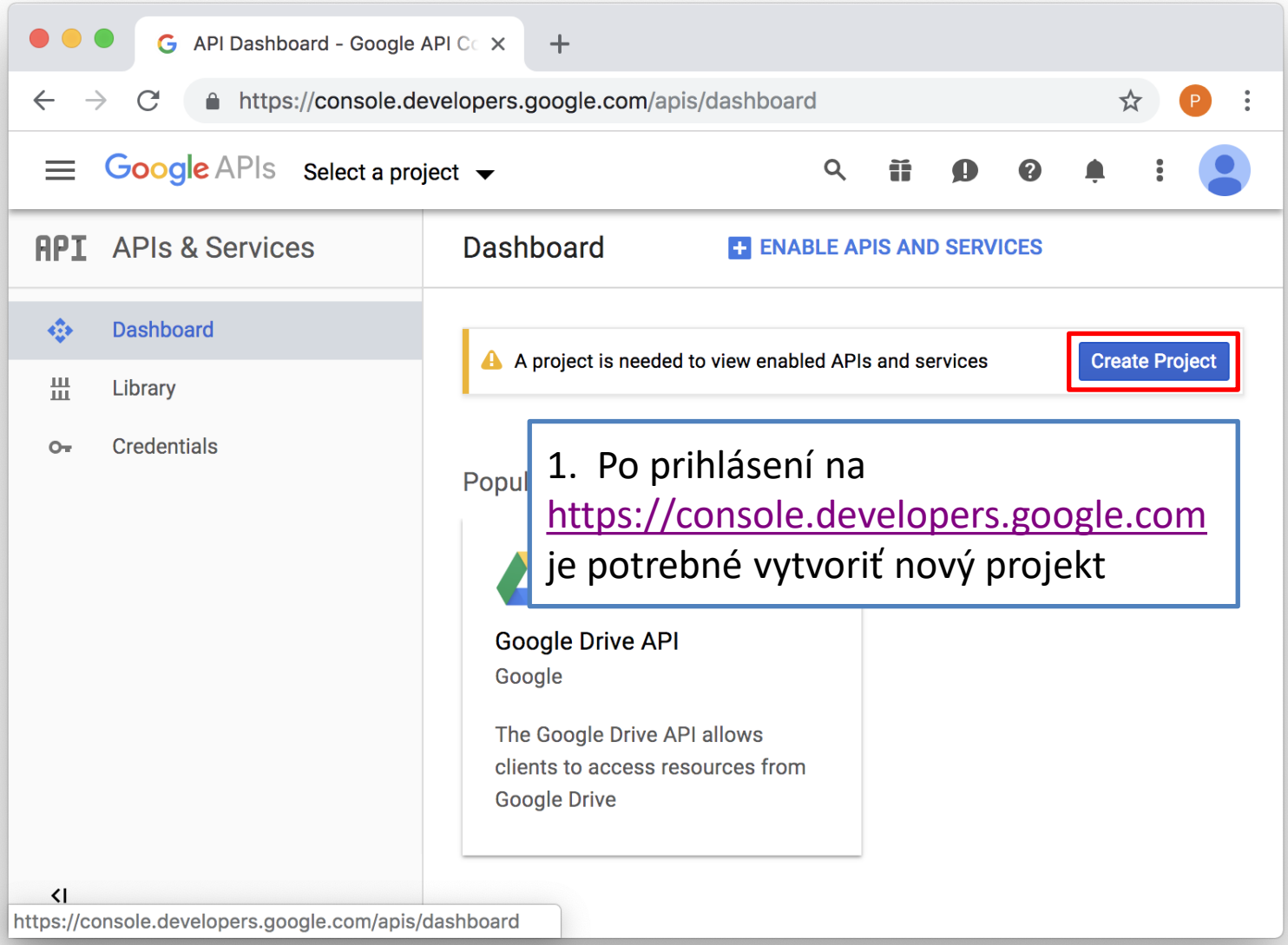


Autentifikácia s Google API

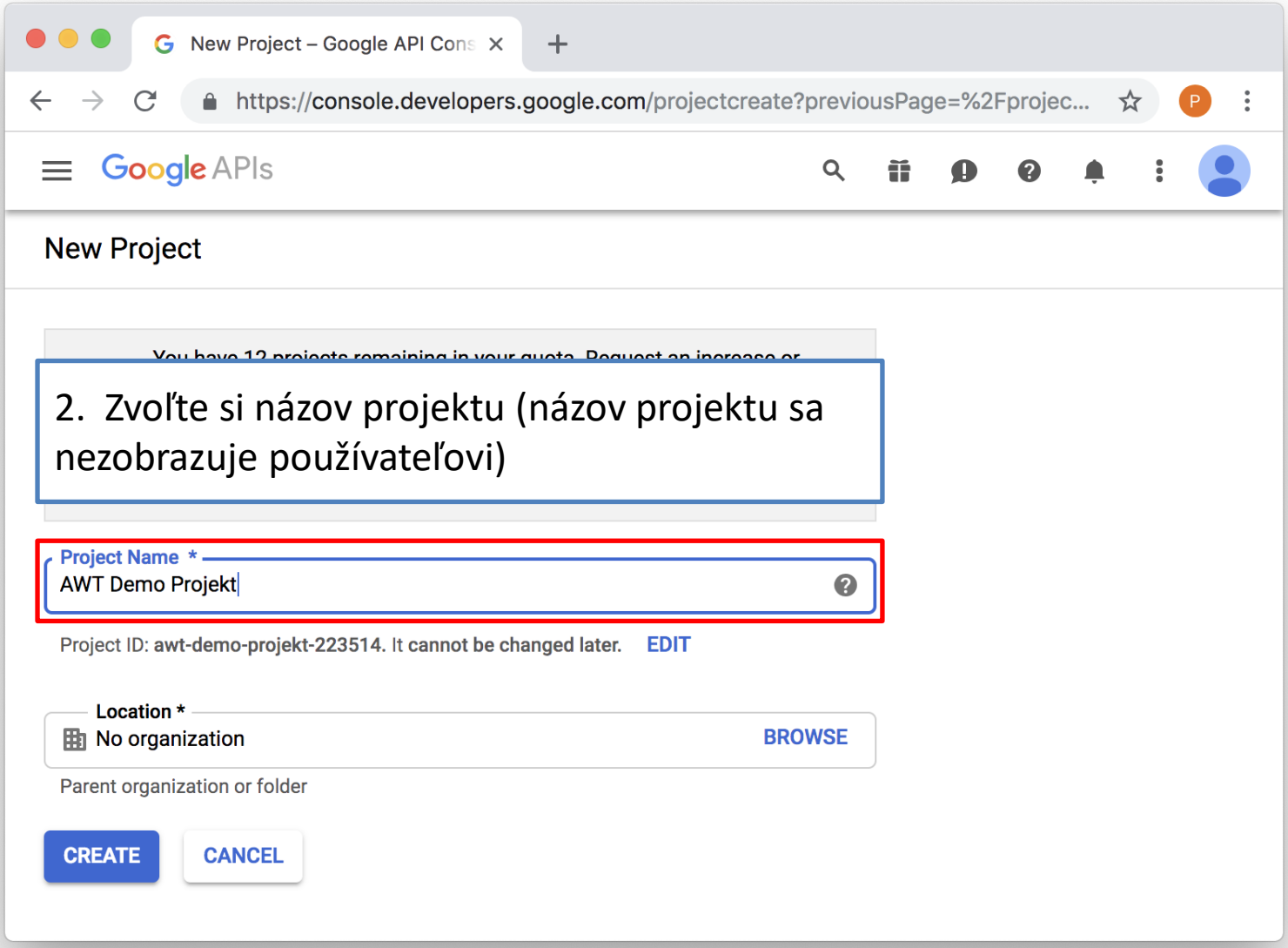
- Postup: https://developers.google.com/identity/sign-in/web/

1. Registrácia vývojára
   - Pre registráciu vašej aplikácie musíte mať Google účet
2. Registrácia aplikácie na stránke Google API Console
   - Po registrácii aplikácie získate client ID, ktoré sa používa na overovanie totožnosti vašej aplikácie pri prihlasovaní
3. Integrácia prihlasovania do stránky aplikácie

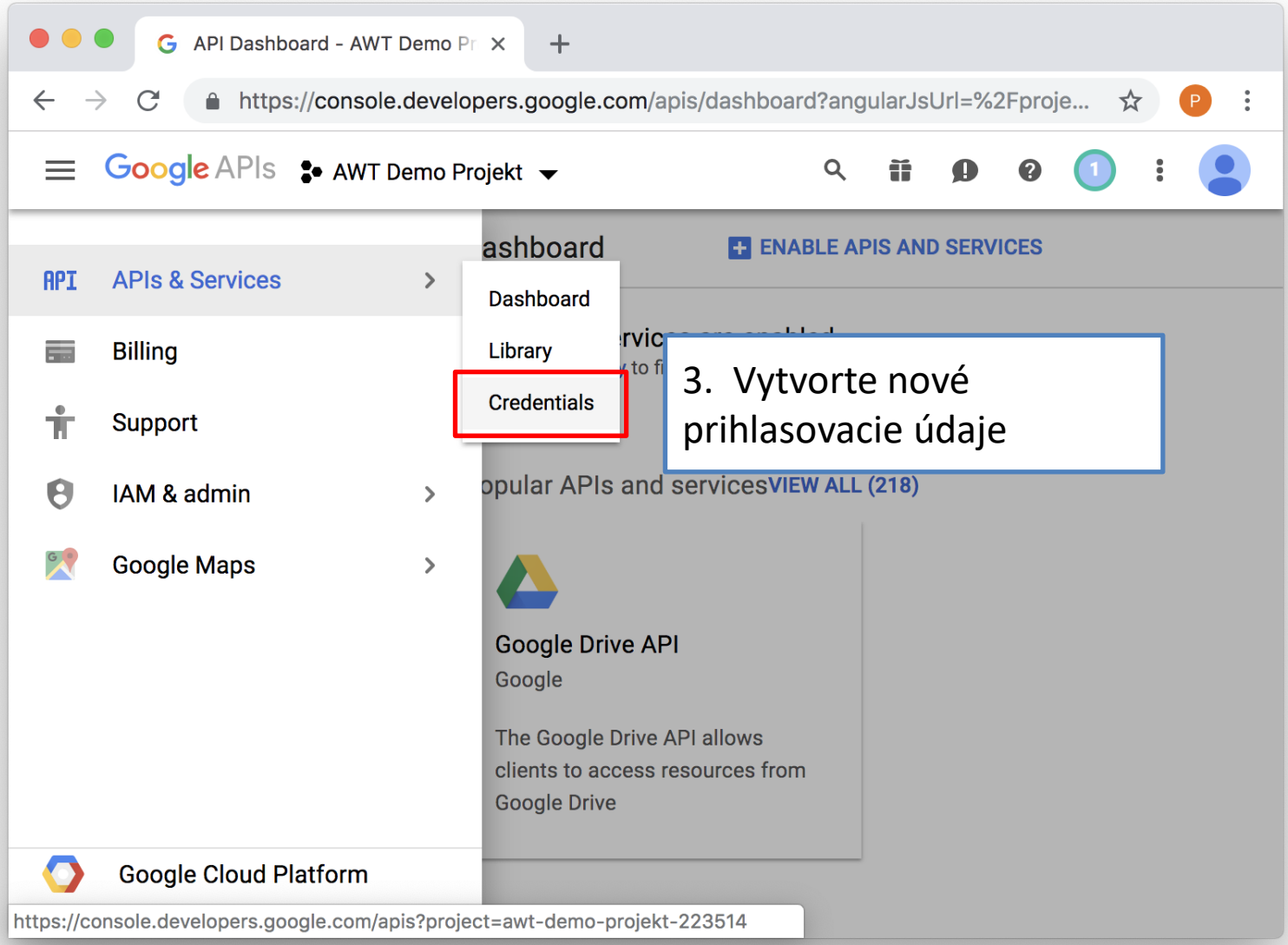
Registrácia aplikácie (1)



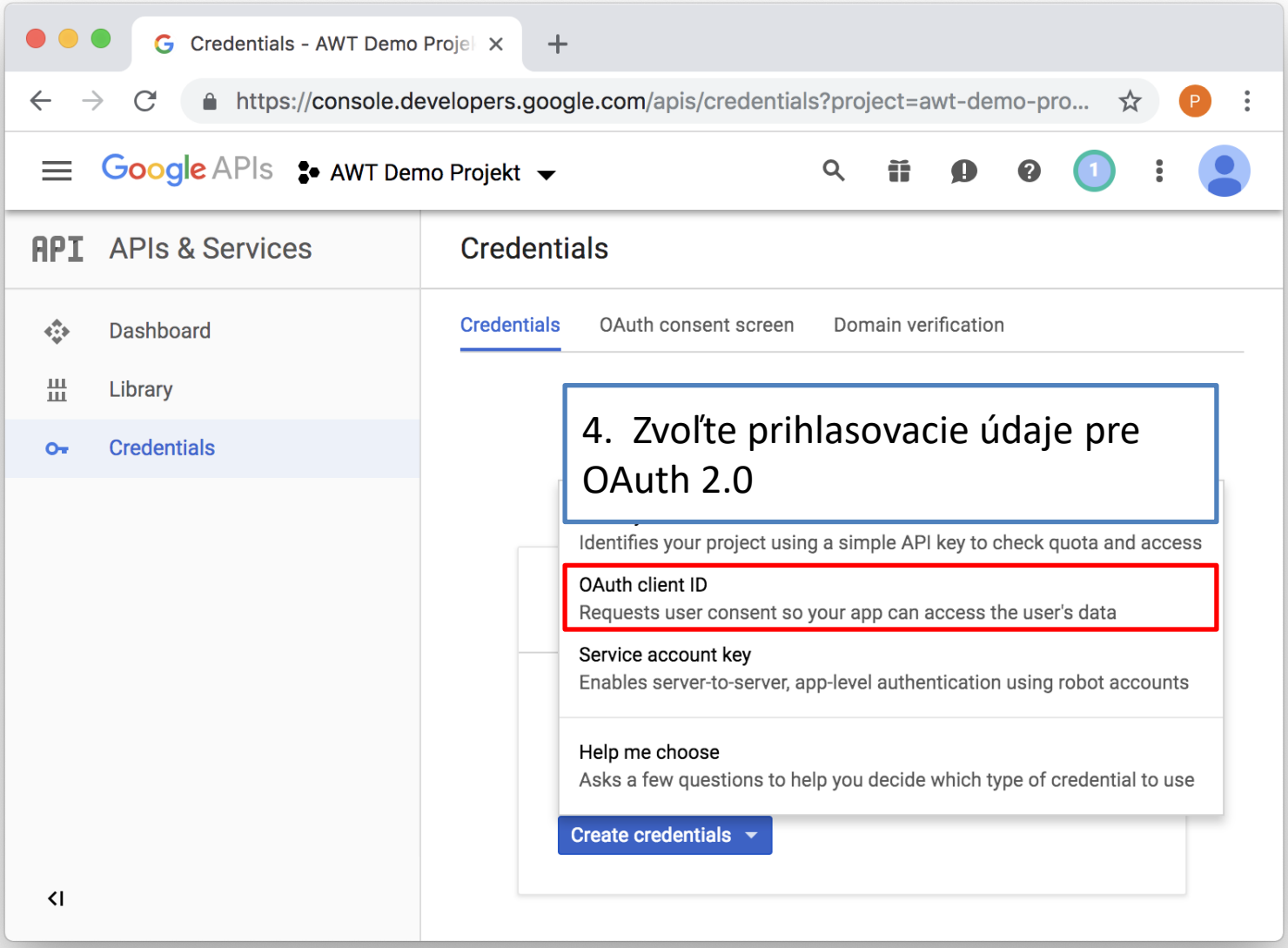
Registrácia aplikácie (2)



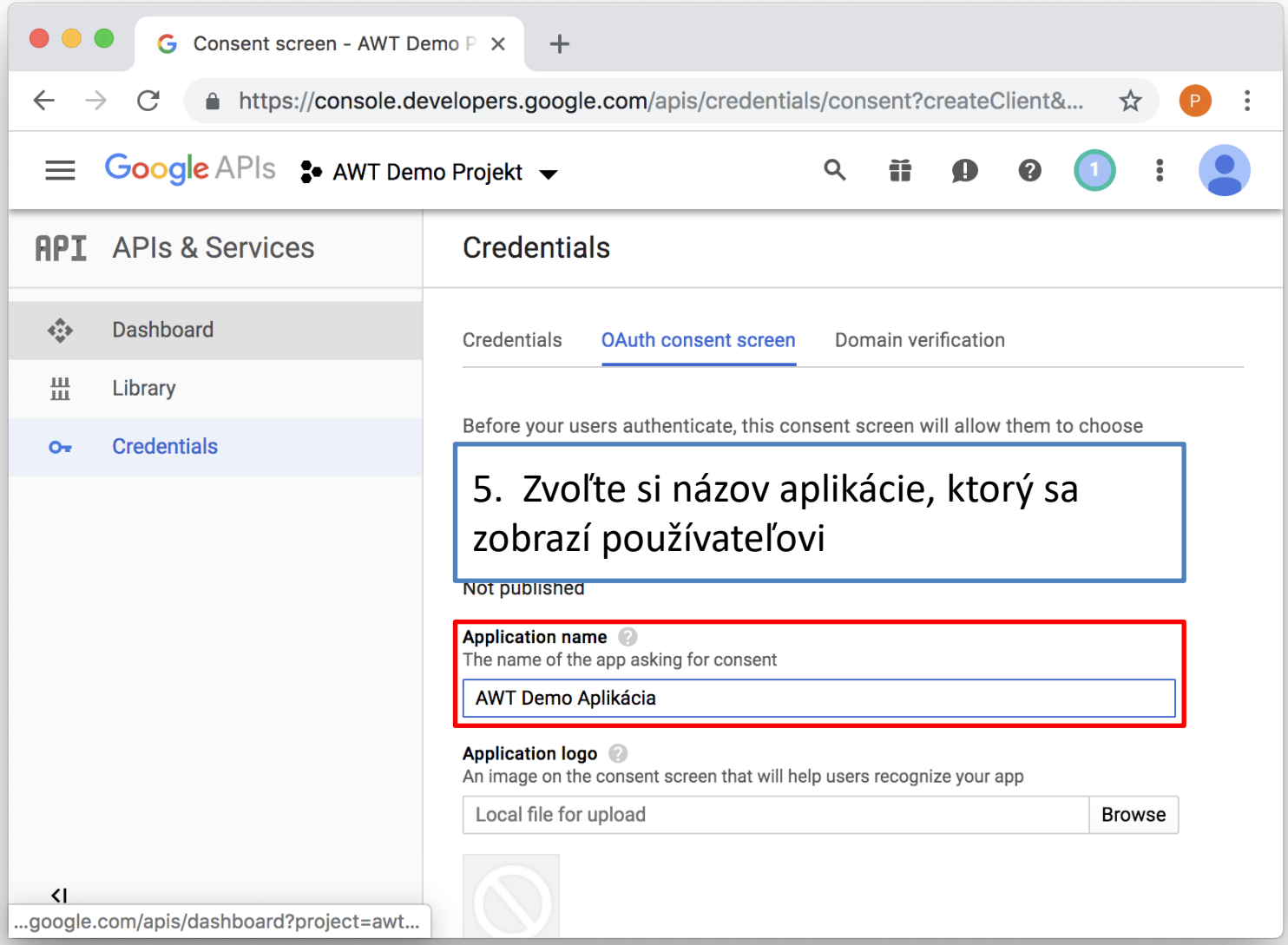
Registrácia aplikácie (3)



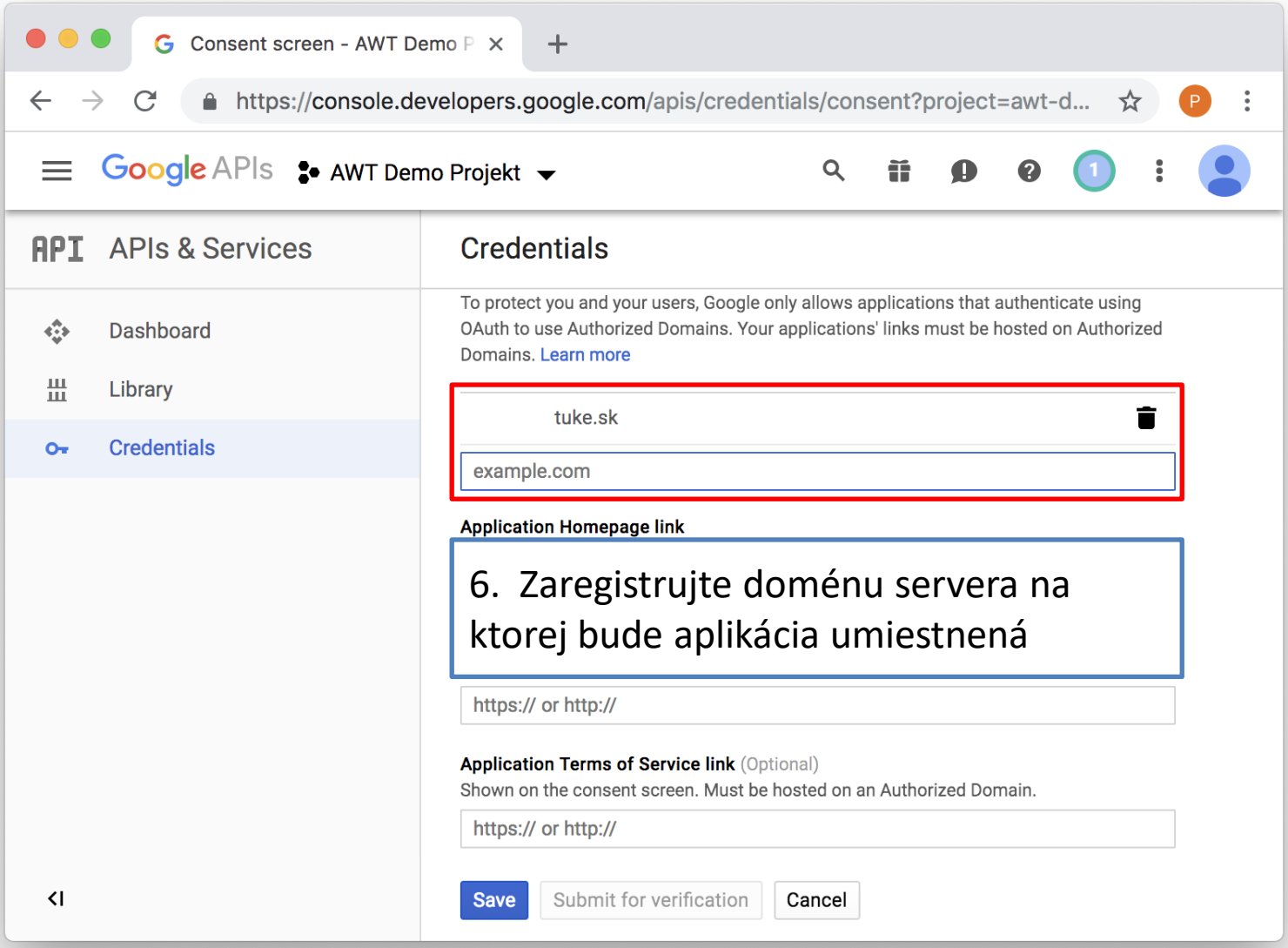
Registrácia aplikácie (4)



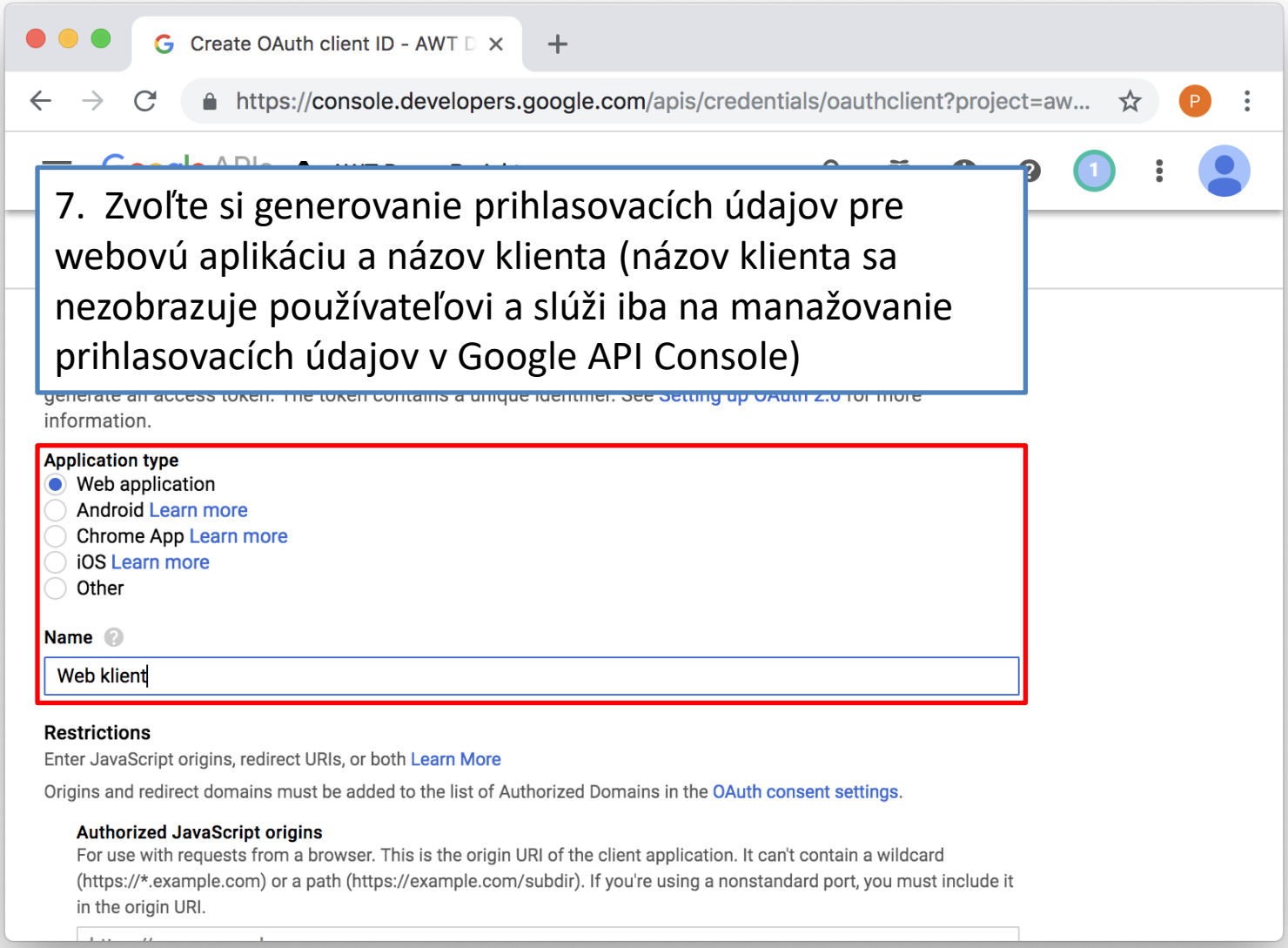
Registrácia aplikácie (5)



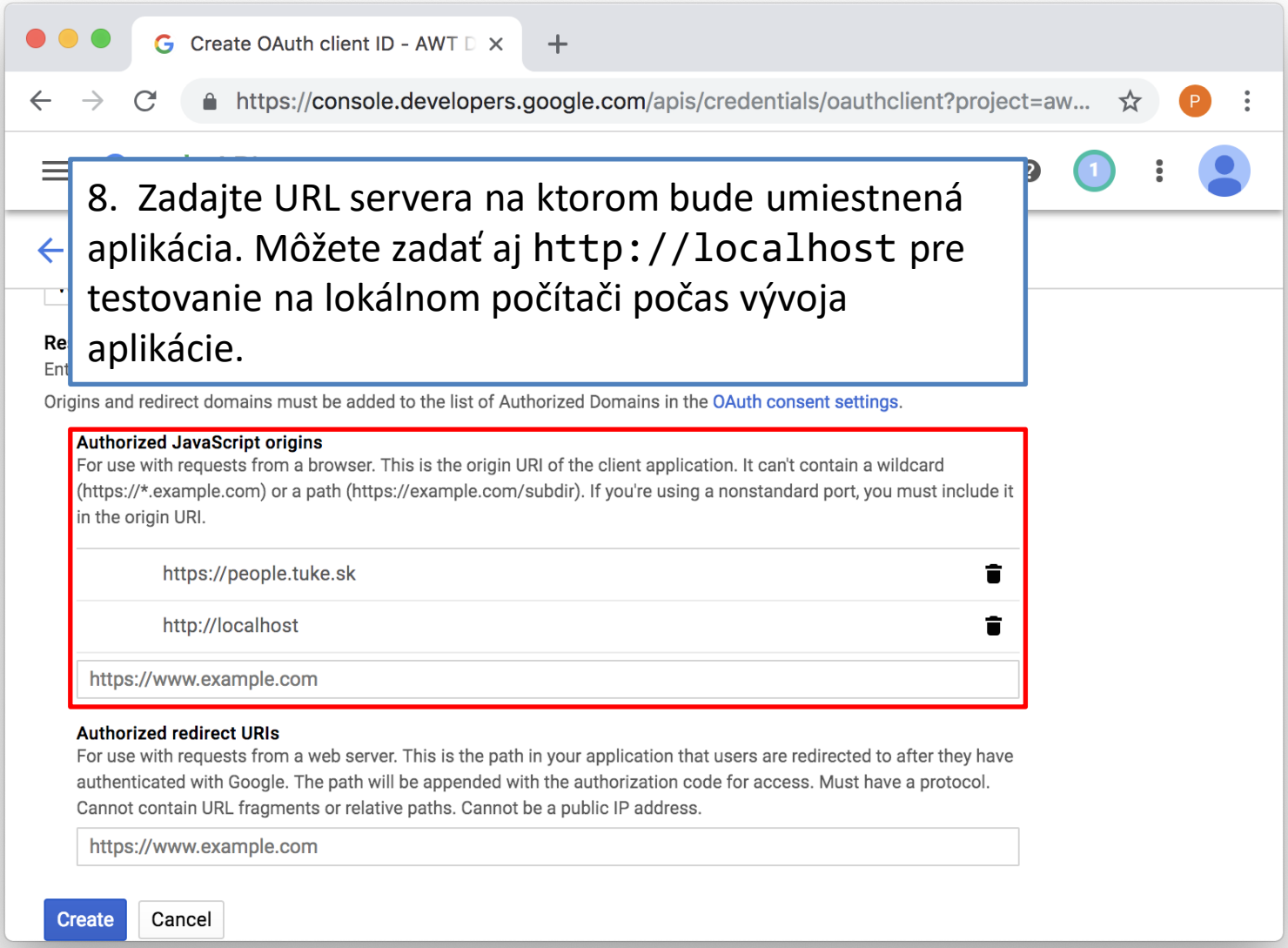
Registrácia aplikácie (6)



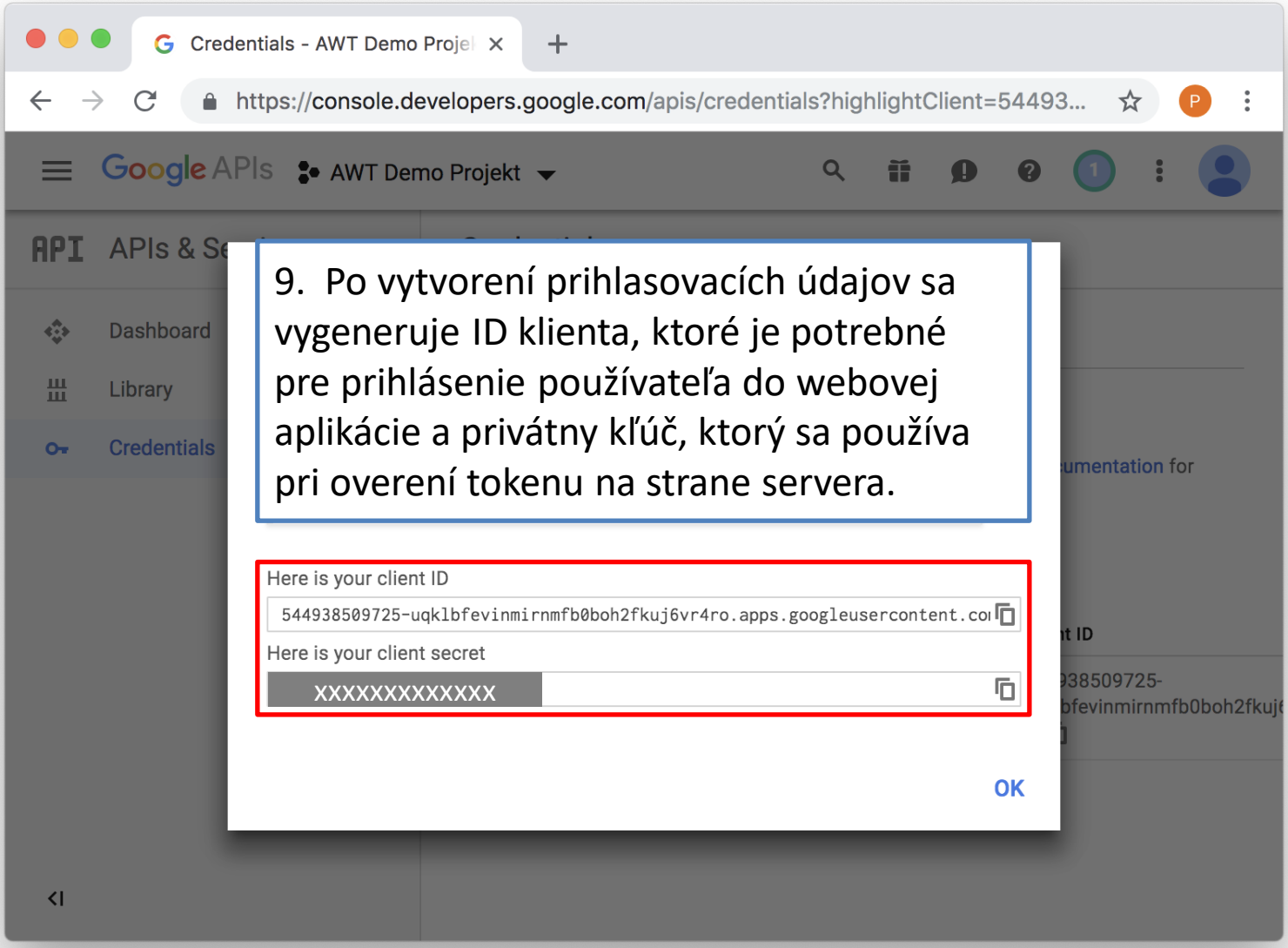
Registrácia aplikácie (7)



Registrácia aplikácie (8)



Registrácia aplikácie (9)



Integrácia prihlasovania do stránky aplikácie (1)

1. Do hlavičky stránky je potrebné pridať `<meta>` element s client ID aplikácie a skript Google API

client ID aplikácie

```
<head>
  <meta name="google-signin-client_id"
    content="XXX-xxx...xxx.googleusercontent.com" />
  <script src="https://apis.google.com/js/platform.js"
    async defer></script>
</head>
```

skript Google API

Integrácia prihlasovania do stránky aplikácie (2)

2. Na stránke vytvoríme prihlasovacie tlačidlo
   – Tlačidlo je vytvorené ako prázdny element zaradený do CSS triedy `g-signin2`. Obsah elementu a obsluhu kliknutia nastaví skript Google API

```
<div class="g-signin2" data-onsuccess="onSignIn"></div>
```

- Atribút `data-onsuccess` tlačidla nastavíme na funkciu, ktorá sa zavolá po úspešnom prihlásení na stránke Google

Integrácia prihlasovania do stránky aplikácie (3)

3. Naprogramujeme funkciu volanú pri úspešnom prihlásení. Parametrom funkcie je objekt, ktorý reprezentuje prihláseného používateľa

Autentifikačný token sa používa pre overenie používateľa na servery

```
function onSignIn(googleUser) {
  // autorizačný token
  var id_token = googleUser.getAuthResponse().id_token;
  // profil používateľa
  var profile = googleUser.getBasicProfile();
  console.log(profile.getEmail());
  ...
}
```

Profil používateľa obsahuje základné informácie ako napr. meno a emailovú adresu

Odhlásenie používateľa

- Vytvoríme napr. odkaz na odhlásenie:

```
<a href="#" onclick="signOut();">Odhlásiť sa</a>
```

- Funkcia pre odhlásenie:

Získame objekt rozhrania pre prácu s Google API

```
function signOut() {
  var auth2 = gapi.auth2.getAuthInstance();
  auth2.signOut().then(function () {
    alert("Boli ste odhlásený...");
    ...
  });
}
```

Odhlásime používateľa volaním metódy `signOut()` a zaregistrujeme funkciu, ktorá sa zavolá po úspešnom odhlásení

Príklady

- Prihlasovanie pomocou Google API