FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	OAuth 2 autorizace pro aplikaci OctoPrint
Student:	Jiří Hanuš
Vedoucí:	Ing. Miroslav Hrončok
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Prozkoumejte možnosti pluginů do aplikace OctoPrint. Navrhněte a implementujte plugin pro OctoPrint, který umožní přihlášení uživatelů a řízení práv přes OAuth 2. Funkčnost pluginu ověřte a předveďte na autorizačním serveru FIT ČVUT a na alespoň jednom dalším (např. GitHub). Využijte vhodnou existující OAuth 2 knihovnu pro jazyk Python. Připravte a nasaďte instanci aplikace OctoPrint používající FIT ČVUT OAuth 2 autentizaci pro nasazení v předmětu 3D tisk.

Kód musí být vhodně členěn, splňovat konvence jazyka Python, musí být psán, komentován a dokumentován v angličtině. Doplňte jej dostatečným množstvím jednotkových a integračních testů. Součástí dokumentace musí být návod na použití nějakého veřejného OAuth 2 poskytovatele (např. GitHub).

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 28. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

OAuth 2 autorizace pro aplikaci OctoPrint

Jiří Hanuš

Katedra Softwarového Inženýrství
Vedoucí práce: Ing. Miroslav Hrončok

14. května 2018

Poděkování

Rád bych poděkoval svému vedoucímu práce, Miru Hrončokovi, za všechny cenné rady. Děkuji mu za jeho ochotu odpovídat na moje občas hloupé otázky, za rady programování v Pythonu, za usměrňování mých zmatených myšlenek a také za vždy velkou dávku objektivity, rad, poznatků a připomínek na konzultacích.

Dále bych rád poděkoval všem mým kolegům, kamarádům, seznamovákům a známým a všem, kteří se mě alespoň jednou zeptali na to, jaké mám téma bakalářky nebo jak mi to jde. Moc si toho vážím.

Velké díky patří mojí přítelkyni Aničce, která se mnou vydržela období práce na bakalářce. Obdiv jí patří také proto, že v klidu snášela, že jsem se uprostřed noci zvedl z postele, protože mě zrovna napadlo, jak daný kus kódu naprogramovat.

Na závěr bych chtěl moc poděkovat svojí rodině za jejich obrovskou podporu nejen při psaní mé bakalářské práce ale i celkově ve studiu. Když toto píšu, ukápla mi slzička. Děkuji rodino.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2018 ……

Odkaz na tuto práci

Hanuš, Jiří. *OAuth 2 autorizace pro aplikaci OctoPrint.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem mé práce je navrhnout a naimplementovat plugin pro rozšíření aplikace OctoPrint. Díky němu se budou moci uživatelé přihlásit do uživatelského rozhraní pomocí serveru třetí strany využitím autorizačního protokolu OAuth 2.0. Aplikace OctoPrint se například používá na ovládání 3D tiskáren při výuce předmětu BI-3DT na Fakultě informačních technologií ČVUT v Praze.

V první části jsou zpracované možnosti vytváření pluginů do aplikace OctoPrint, dále je popsaná vnitřní struktura OctoPrintu. Následně je provedena rešerše autorizačního protokolu OAuth 2.0 a na základě těchto informací je vytvořen návrh samotného pluginu. Implementace je provedena pomocí autentizačních knihoven jazyka Python.

Dále je popsána samotná implementace softwaru, která je otestována, jak samostatně pomocí použití naprogramovaných testů, tak na Raspberry Pi ve 3D laboratoři na FIT ČVUT v Praze. Je také popsána konfigurace pluginu pro vlastní využití.

Výsledný plugin umožňuje přihlášení přes autentizaci OAuth 2.0 do aplikace OctoPrint. Toto rozšíření ulehčí výuku předmětu BI-3DT na FIT ČVUT v Praze. Dále je plugin dostupný i celé komunitě 3D tiskařů pro další možné úpravy či rozšíření.

Klíčová slova plugin OctoPrint, OAuth 2.0 framework, 3D Tisk, autorizace, usnadnění výuky, Python, KnockoutJS, Jinja2

Abstract

The aim of this bachelor thesis is to design and create plugin for an application OctoPrint. This extension will provide users the possibility to log into user interface using the authorization framework OAuth 2.0. OctoPrint application is used for example to control 3D printers at the Faculty of Information Technogogy at Czech Technical University in Prague.

The first part of the thesis describes capabilities of creating a plugin for the OctoPrint application. This part is followed by a research on an authorization protocol of OAuth 2.0 and, subsequently, a plugin design is created based on results of the research. The implementation is done using Python authentication libraries.

After that, the programming of the software is described. The software itself is afterwards tested separately using programmed tests, as well as on Raspberry Pis from a 3D laboratory at the Faculty of Information Technology. The plugins configuration for custom use is also described.

The resulting plugin enables the authorization via OAuth 2.0 into the OctoPrint application. This extension will facilitate the teaching of a subject 3D Printig at the FIT CTU in Prague. Additionally, the plugin is available to the community of people working with 3D printers for further editing or extensions.

Keywords plugin OctoPrint, OAuth 2.0 framework, 3D Printing, authorization, teaching facilitation, Python, KnockoutJS, Jinja2

Obsah

Seznam obrázků

Seznam ukázek kódu

Úvod

3D tisk je v dnešní době velice populární. Existuje několik druhů a mnoho typů 3D tiskáren, nekteré jsou komerční a jiné se svobodnou licencí. Samotnou tiskárnu je potřeba ale nějak ovládat. Za tímto účelem bylo vytvořeno několik aplikací na ovládání 3D tiskárny a jednou takovou je aplikace OctoPrint. OctoPrint je webové rozhraní pro ovládání 3D tiskárny. Je zcela open source a umožňuje přidávat i vlastní rozšíření, které pak mohou používat i ostatní z komunity tiskařů.

Výsledek mojí práce je určen hlavně pro studenty ČVUT v Praze, kteří studují předmět 3D tisk na Fakultě informačních technologií. Tato práce totiž umožňuje přihlášení do rozhraní OctoPrintu přes třetí stranu pomocí autentizačního protokolu OAuth 2.0. Tento protokol se využívá i na ČVUT. Studenti by se tedy mohli přihlásit k tiskárnám pomocí svých přihlašovacích údajů, a to by pomohlo právě během hodin 3D tisku ke kontrole zadaných úkolů. Samotný plugin je konfigurovatelný, a je tedy možné využít autorizaci i jiného autorizačního serveru. Například je tedy možné se přihlásit pomocí sociálních sítí nebo účtu na GitHubu.

Samotné téma jsem si vybral, protože jsem člen 3D laboratoře na FIT ČVUT v Praze a jako pomocný vyučující předmětu BI-3DT bych chtěl přispět něco fakultě zpět. Jsem také nadšenec do 3D tisku a líbí se mi komunita lidí kolem tohoto tématu.

Výsledek této práce má za cíl vytvořit takový plugin do aplikace OctoPrint, díky kterému by se do ní mohl uživatel přihlásit přes různé autorizační servery pomocí autentizace OAuth 2.0.

V této bakalářské práci se zabývám analýzou vytváření pluginů do aplikace OctoPrint, vnitřní strukturou OctoPrintu, dále pak rešerší autentizace pomocí OAuth 2.0 a dalších technologií, které jsou použité pro výsledný plugin. Na základě těchto analýz navrhnu možná řešení, ze kterých si jedno vyberu a naimplementuji.

Cíl práce

Cílem mé bakalářské práce je vytvořit takový plugin do aplikace OctoPrint, díky kterému se budou moci uživatelé přihlásit do OctoPrintu přes třetí stranu pomocí autentizačního protokolu OAuth 2.0.

Na úvod je potřeba prozkoumat možnosti pluginů do aplikace OctoPrint. Dále pak navrhnout řešení umožňující nastavit plugin pro aplikaci OctoPrint pro různé autentizační servery využívající protokol OAuth 2.0. Do návrhu je nutné začlenit použití vhodné autentizační knihovny jazyka Python.

Cílem praktické části je naimplementovat návrh aplikace pro autentizaci a samotné řešení integrovat ve formě pluginu do aplikace OctoPrint. Dílčím cílem práce je otestovat výsledný plugin pomocí jednotkových a integračních testů a dále pak otestovat nasazení a funkčnost na školních Raspberry Pi.

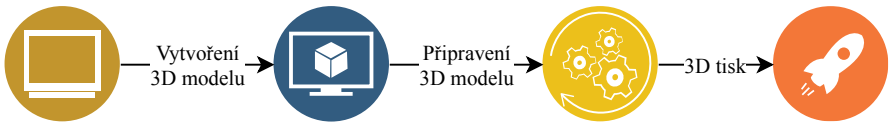
Kapitola 1

Rešerše

V této kapitole se budu věnovat obecně 3D tisku, jeho možnostmi, technologiemi a 3D tiskárnami. Jelikož je pro uživatele pohodlné 3D tiskárnu ovládat přes nějaké rozhraní, budu se z části práce zabývat také rešerší aplikace OctoPrint, do které je výsledný plugin určený. Budu také zkoumat možnosti vývoje pluginů do OctoPrintu. Dále popíšu základní myšlenku a principy autorizačního protokolu OAuth 2.0, který je potřeba prostudovat pro tuto bakalářkou práci.

1.1 3D tisk

3D tisk [2] je proces, při kterém se díky digitální předloze, 3D modelu, vytvoří třídimenzionální objekt. Tento objekt je vytvořen aditivně, tedy přidáváním nějakého materiálu, který se zatvrdí. Existuje několik možných technologií, většina ale funguje na principu přidávání vrstev materiálu. Nejznámější technologií je vrstvení termoplastu neboli FDM – Fused Deposition Modeling, během níž je nahřátý plast tryskou vytlačován ven do jednotlivých vrstev [3]. Druhou takovou technologií je SLA – Stereolitografie, při kterém se objekt postupně zatvrzuje z tekutého polymeru. Za zmínku ještě stojí technologie, která tiskne pomocí kovu. Jednotlivé vrstvy jsou tvořeny práškem kovu, jenž je pomocí laseru zatvrzen.



Obrázek 1.1: Proces 3D tisku

3D tisk přináší mnoho výhod, které je důležité zmínit. Nejvýznamnější výhodou názoru výroba prototypů [3]. Dříve, když člověk potřeboval nějaký prototyp, musel buď předmět vyřezat nebo vyrobit na soustruhu. Díky 3D tisku

ale člověk s vytvořeným 3D modelem snadno vytvoří reálný objekt k otestování. Potom už může přejít na větší výrobu například pomocí vstřikování.

1.1.1 3D tiskárny

Pro 3D tisk je nezbytná samozřejmě nějaká 3D tiskárna. Většina 3D tiskáren, se kterými jsem se setkal, je z projektu RepRap [4]. Jsou to tiskárny „open source hardware“, tedy hardwarové návrhy 3D tiskáren jsou volně dostupné, stejně tak jako zdrojové kódy a návody pro sestavení. To vše je pod licencí GNU General Public License. Tyto tiskárny mají tu výhodu, že velká část součástek k samotné tiskárně může být vytisklá na jiné 3D tiskárně. Proto i v názvu projektu RepRap se skrývá slovo replicating, tedy že se tiskárny mohou samy replikovat. Hlavním cílem projektu RepRap bylo vytvořit levnou a dostupnou 3D tiskárnu pro veřejnost.

Nejznámější výrobci jsou například Ultimaker, MakerBot, LulzBot nebo Josef Průša s jeho společností Prusa Research [5]. Průša je pravděpodobně největším průkopníkem 3D tisku v České Republice díky tiskárně Prusa i3, které se dostalo již několik vylepšení a byla i celosvětově oceněna nejlepší tiskárnou roku 2017 i 2018. Nejvíce se pyšní svojí cenovou dostupností a proto jsou také podle mého názoru u obyčejných uživatelů tak oblíbené [2].

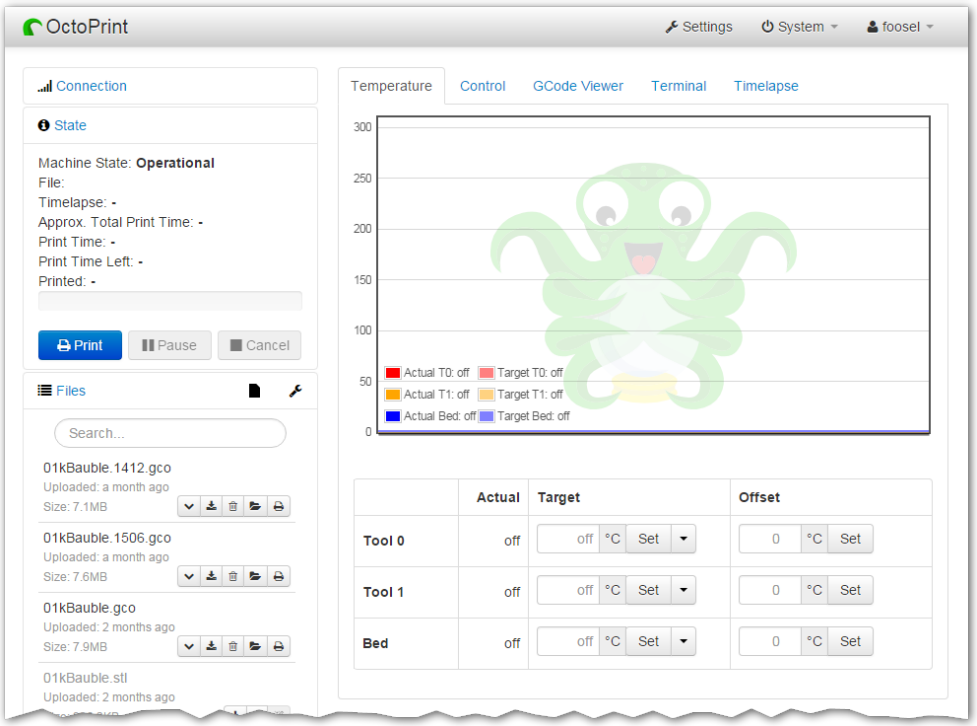
Tiskárna Prusa i3 Multi Material je vytvořena tak, že si během tisku dokáže vyměnit termoplast, a tak tisknout vícebarevně. Největší novinkou od Prusa Reseach je ale tiskárna Prusa i3 MK3 [6], která má pružnou odjímatelnou podložku, kterou je možné po tisku vyměnit za jinou a tisknout další model. Také obsahuje senzor, který pozná, jestli 3D tiskárně došel filament[1] a v takovém případě dokáže pozastavit tisk.

1.1.2 Materiály pro 3D tisk

Jednou z nejpoužívanějších technologií je SLA – Stereolitografie. Ta využívá tekutý polymer, který je pomocí světla na určité vlnové délce zatvrzován. Jeho výhodou je velká přesnost výtisků, kde výsledný objekt může pouhému oku připadat už jako hladký. Nevýhodou SLA je hlavně vysoká cena materiálu a pro běžného uživatele se nákup materiálu podle mého názoru nevyplatí.

Cenově výhodnější jsou například materiály z termoplastu, tedy plastu, který se v trysce nataví a je vytlačován do jednotlivých vrstev. Nejpoužívanější materiály jsou ABS – Akrylonitrilbutadienstyren, PLA – Kyselina polymléčná a PET-G – Polyethylentereftalát, kde G značí modifikovaný glykol, který se přidává během polymerace. Rozdíly mezi těmito plasty najdete v [7].

[1] *Filament* – Náplň do 3D tiskárny, obvykle termoplast.



Obrázek 1.2: Uživatelské rozhraní aplikace OctoPrint [1]

1.2 OctoPrint

Existuje několik aplikací na ovládání 3D tiskáren jimiž jsou například Pronterface, Repetier, Cura, CraftWare, MatterControl nebo 3D Builder na Windows. Jednou z těchto aplikací je právě OctoPrint. Přehled najdete na [8].

OctoPrint [1] je webové rozhraní pro ovládání 3D tiskárny a je jednou z nejpoužívanějších variant pro ovládání 3D tiskáren obecně. OctoPrint je zcela pod svobodnou licencí AGPL, jeho zdrojové kódy jsou dostupné veřejnosti a vylepšovat je může celá komunita 3D tiskařů.

Za vytvořením OctoPrintu stojí Gina Häußge z Německa. Protože je to aplikace velkého rozsahu a práce na ní je na plný úvazek, je potřeba nějak tuto práci zaplatit. Poté co stvořitelce OctoPrintu vypršela podpora od komerčních partnerů, obrátila se na samotné uživatele. V tento moment zhruba 1300 uživatelů přispívá, nutno podotknout že dobrovolně, každý měsíc kolem pěti a půl tisíce dolarů. OctoPrint je možné podporovat pravidelně každý měsíc přes server Patreon nebo také jednorázově pomocí PayPal.

Díky OctoPrintu je možné na dálku ovládat a monitorovat 3D tiskárnu či nahrávat do něj 3D modely ve formě gcode[2]. Dále je možné pozorovat

[2] *Gcode* - Série instrukcí (nejen) pro 3D tiskárnu pro vytisknutí modelu.

vizualizaci gcode, ovládat teplotu tiskárny nebo pohybovat s tryskou po všech třech osách. Je také možné kdykoliv pozastavit nebo zrušit tisk.

OctoPrint se pyšní svou kompatabilností. Je možné ho spustit na většině operačních systémů. Nejčastěji se ale používá pro mikropočítače, jako je například Raspberry Pi, pro který je vytvořen speciální obraz na SD kartu pojmenován OctoPi [1]. Kompatabilní je i různými 3D tiskárnami. Jako příklad bych uvedl modely tiskáren od Makerbot, Ultimaker, Prusa Research a další tiskárny RepRap.

OctoPrint je možné také spojit s jinými aplikacemi, jako je například Pushbullet, přes který je možno například poslat upozornění, že tisk byl dokončen. OctoPrint je také v jistých ohledech dobře rozšiřitelný. Je možné zcela nahradit uživatelské rozhraní a přizpůsobit ho například pro mobilní zařízení. Existují dva hlavní způsoby, jak OctoPrint rozšířit. Jedním z nich je přímý vývoj aplikace a druhým, pro mě podstatnějším způsobem, je vytvoření pluginu do aplikace OctoPrint. Této variantě se budu věnovat později.

1.2.1 Vnitřní struktura OctoPrintu

Vniřtní strukturu OctoPrintu ovlivňuje několik faktorů. Vnitřní funkcionalita webového rozhraní je napsána v jazyce Python, uživatelské rozhraní je pak tvořeno pomocí Knouckout a šablonovacího jazyka Jinja2. Díky těmto dvěma utilitám OctoPrint kombinuje statické a dynamické prvky uživatelského rozhraní.

KnockoutJS [9] je javascriptová knihovna umožňující vytvořit responzivní uživatelské rozhraní. Knockout dokáže chytře měnit uživatelské rozhraní v případě, když se změní zdroje informací z jednotlivých modelů. Umí spojovat jednotlivé části uživatelského rozhraní s data modely pomocí „data-bind“. Díky tomu se dá jednoduše vytvořit komplexní a dynamické uživatelské rozhraní. Právě pomocí této knihovny je možné jednoduše naprogramovat nové „data-binds“ a definovat nové chování rozhraní.

Jinja2 [10] je šablonovací jazyk pro Python. Je navržen tak, aby byl co nejvíce flexibilní, dostatečně rychlý a bezpečný. Tento jazyk nabízí web designerům možnost pracovat s jednotlivými šablonami a automaticky tak vygenerovat širokou škálu internetových stránek. Jinja2, jako i ostatní šablonové jazyky, má tu možnost dědit od jednotlivých šablon a snadno je rozšiřovat. Jinja2 je velmi podobný šablonovacímu enginu Django.

OctoPrint je celkem snadno nastavitelný díky konfiguračním souborům ve formátu `YAML`. Tento formát má rekurzivní zkratku, která značí „`YAML` Ain't Markup Language“. Znamená to, že to není další značkovací jazyk jako je HTML nebo XML. `YAML` se používá pro serializaci strukturovaných dat. Je důležité, aby se zachovávalo správné odsazení pro jednotlivé sekce. V OctoPrintu je nejdůležitější `config.yaml`, díky kterému se dá nastavit celý OctoPrint. Dále se pak využívá `users.yaml`, který v sobě obsahuje informace

o jednotlivých uživatelích. `YAML` se vyznačuje tím, že je snadno pochopitelný i pro běžného uživatele [11].

1.2.2 Konfigurace OctoPrintu

Jak už bylo dříve zmíněno, OctoPrint je velmi škálovatelný. Veškeré nastavení lze nalézt v souboru `config.yaml`, které na Linuxu naleznete v `~/.octoprint`, na Windows v `%APPDATA%/OctoPrint` a na macOS `~/Library/Application Support/OctoPrint`. Velká část samotného nastavení se nemusí měnit ručním přepsáním konfiguračního souboru, ale to řeší samotný OctoPrint přes uživatelské rozhraní v nastavení OctoPrintu.

Pomocí tohoto konfiguračního souboru můžete jeho přepsáním nastavit přístupy do OctoPrintu, REST API a vzhled uživatelského prostředí[12].

U nastavení uživatelského rozhraní se na chvíli pozastavím. Je možné totiž nastavit nejen například barvu rozhraní, ale také i jednotlivé komponenty navigačního menu, záložek a také nastavení. Jednotlivé komponenty se mohou libovolně zpřeházet, smazat nebo nahradit vlastními, což bude potřeba právě pro vytvoření pluginu.

Dále je v konfiguračním souboru možné nastavit nebo přidat vlastní ovládací prvky pro 3D tiskárnu. Pokud se někdo věnuje vývoji OctoPrintu, bude se mu určitě hodit konfigurace pro vývoj. Pokud programátor nevlastní tiskárnu, ale potřeboval by vyzkoušet vývoj přímo na tiskárně, je možné si pomocí konfigurace vývoje vytvořit virtuální tiskárnu, a to s různými parametry. Například pokud potřebuje otestovat svůj vývoj nebo plugin na tiskárně s více prvky, dá se to nastavit v `config.yaml`.

Jelikož OctoPrint také manipuluje se souboury, jako jsou například 3D modely ve formě gcode, do kterých je možné pomocí OctoPrintu nahlížet, je třeba tyto soubory někam ukádat. Do nějaké složky je také potřeba ukládat logy z tiskárny. Některé existující pluginy dokáží nahrát model během tisku nebo dokonce streamovat online. I tyto časosběry se někam mohou ukládat. Cesty k těmto složkám se nastavují v `config.yaml`.

OctoPrint umožňuje pro každý samostatný plugin vyhradit místo v konfiguračním souboru. Jednotlivá rozšíření OctoPrintu si pak zde mohou ukládat svoje možnosti nastavení. I nastavení jednotlivých pluginů je možné měnit přes uživatelské rozhraní, pokud to programátor naimplementoval.

Další z důležitých věcí, které je možné nakonfigurovat, je nastavení samotného serveru OctoPrintu, tedy nastavení hostitele, portu a pokud se něco hodně pokazí, tak nastavení pro první spuštění OctoPrintu, kdy se zobrazí průvodce instalace.

Ostatní nastavitelné věci jsou například: odhad tisku, události, analýza gcode a možnost náhledu gcode, profily tiskáren, skripty pomocí nichž je možné ovládat tiskárnu, nastavení řezání modelu, ovládání teploty a nebo také nastavení webkamery.

1.2.3 Komponenty OctoPrintu

V OctoPrintu existuje pět základních komponent ovlivňujících vzhled uživatelského rozhraní a tyto prvky se mohou nastavovat v konfiguračním souboru `config.yaml`. Tyto komponenty je možno zaměňovat za vlastní nebo je odstranit a jsou to:

navbar – navigační komponenta obsahující nastavení, systémové menu a přihlašování uživatele.

sidebar – komponenta pro postranní menu obsahující stav připojení k tiskárně a manipulaci se soubory.

tab – komponenta umožňující manipulovat s jednotlivými záložkami v OctoPrintu jako například nastavování teploty nebo ovládání tiskárny.

settings – komplexní komponenta pro nastavení OctoPrintu.

usersettings – komponenta pro nastavení uživatele.

1.2.4 Uživatelské rozhraní OctoPrintu

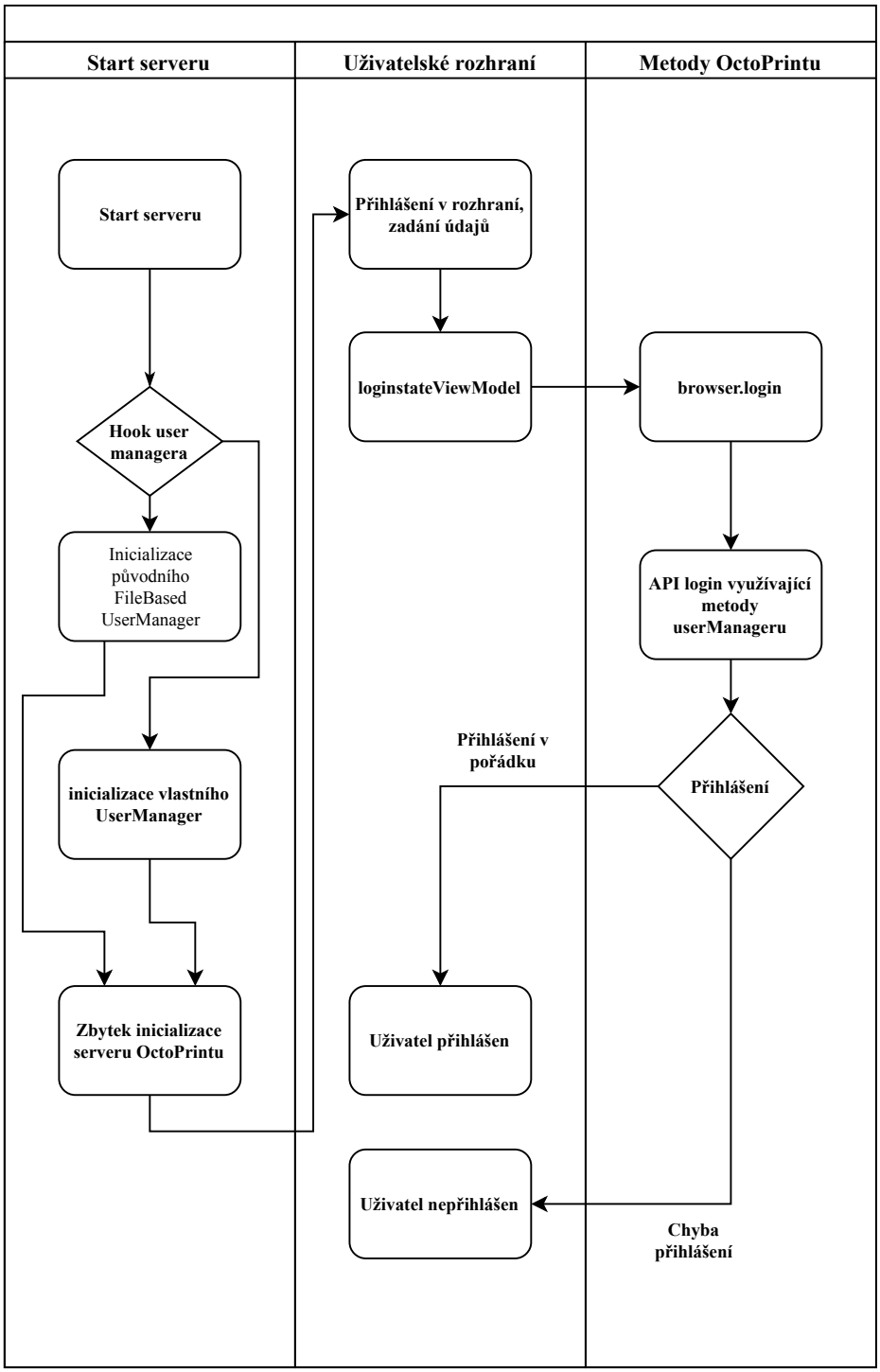
Webové rozhraní OctoPrintu se snaží být uživatelsky přívětivé. Maskot OctoPrintu je zelená chobotnice, po které samotná aplikace dostala název z anglického slova „Octopus“. Nad celým tělem aplikace je navigace, ve které je buď vidět možnost přihlášení, nebo přihlášený uživatel s nastavením OctoPrintu a nastavení pro samotného uživatele.

Nalevo v rozhraní je možno vidět stav, jestli je OctoPrint připojen k 3D tiskárně a stav procesu 3D tisku s odhadem času tisku s načítací lištou. Dokonce i pokud je uživatel odhlášený, může vidět tyto statistiky. Dále nalevo v liště je možno manipulovat se soubory určené pro tisk. Je možné je nahrávat, mazat nebo také vybrat, který bude zrovna určen pro tisk. Přihlášení uživatelé zde mohou zapnout, vypnout nebo pozastavit tisk.

V rozhraní může uživatel také přepínat mezi jednotlivými záložkami pro ovládání tiskárny, jako je nastavení teploty, jak podložky, tak trysky. V OctoPrintu je vytvořeno vykreslování grafu teplot, v další záložce najdeme ovládání motorů tiskárny. Hlavou tiskárny se dá pohybovat po všech třech osách, navíc je možné i ovládat extruzi – vytlačit nebo zasunou termoplast. V dalších záložkách nalezneme prohlížeč modelu pomocí vizualizaci gcode a také výstupy terminálu z 3D tiskárny.

1.2.5 Přihlášení uvnitř OctoPrintu

V rámci analýzy je nezbytné zmínit, jak funguje přihlášení do aplikace OctoPrint trochu detailněji.



Obrázek 1.3: Diagram přihlašování uživatele OctoPrintu

Při startu OctoPrintu se inicializuje instance třídy `userManager`, která definuje, jak se uživatelé budou přihlašovat. Pomocí hook, ke kterému se dostanu v 1.3.2, `octoprint.users.factory` může plugin nastavit svojí vlastní instanci `userManager` a nastavit svoje vlastní metody pro manipulování s uživateli. K popisu jednotlivých hooks se dostanu v kapitole 1.3 Tvorba pluginů pro Octoprint.

Pro následující příklad budeme předpokládat, že se při startu serveru OctoPrintu zvolil původní `FileBasedUserManager`, který informace o uživatelích ukládá do souboru `users.yaml`.

Po inicializaci serveru je možné se přihlásit do OctoPrintu přes webové rozhraní zadáním uživatelského jména a hesla. Tlačítkem pro přihlášení dáte vědět `loginStateViewModel`, že se má zahájit procedura přihlášení, a spustí se javascriptová metoda `browser.login`.

Ta následně předá informaci o tom, že se chce uživatel přihlásit rozhraní serveru OctoPrintu. API zahájí metodu pro přihlašování a pomocí metody POST zažádá o přihlašovací údaje uživatele. Jakmile je dostane, dočasně si je uloží. Rozhraní pomocí `userManager` vyhledá, jestli uživatel existuje a pokud ne, nastane chyba přihlášení. Pokud uživatel existuje, volají se další metody z `userManager` na kontrolu hesla. V případě správnosti vrátí zpátky rozhraní informace o uživateli a jeho nastaveních zpátky do uživatelského rozhraní a uživatel bude přihlášen.

1.3 Tvorba pluginů pro OctoPrint

V této sekci zmíním, jaké jsou možnosti vývoje pluginů do aplikace OctoPrintu. Na začátek uvedu několik obecných informací a dále v jednotlivých podsekcích vysvětlím detailněji jednotlivé možnosti.

Teď, když už jsem zmínil, jak OctoPrint pracuje uvnitř a jak vypadá jeho uživatelské rozhraní, přejdeme na analýzu tvorby pluginů pro aplikaci OctoPrint. Pluginy do OctoPrintu jsou balíčky pro jazyk Python, moduly, které jdou nainstalovat pomocí utility pip. V hlavním modulu jsou definovány základní informace, jako je jméno pluginu, verze, popis, licence nebo například jaké potřebné hooks jsou pluginem implementovány. Pro účely mé práce budu pravděpodobně potřebovat hooks, nebo kombinaci mixins a hooks. [12]

Existuje několik základních možností, jak plugin propojit se sytémem. První z možností jsou takzvané mixins, druhý ze způsobů je pomocí hooks. Na závěr bych ještě zmínil také helpers, díky kterým je možno plugin propojit s jiným už existujícím pluginem. Tyto tři prvky určují chování pluginu.

Přes ViewModels je pro tvorbu pluginu umožněno pozměnit nebo rozšířit grafickou stránku OctoPrintu. ViewModels jsou vytvořené pomocí javascriptové knihovny Knockout a jsou spojené přes datové návaznosti s jednotlivými jinja2 soubory, které tvoří samotnou stránku.

1.3.1 Mixins

Mixins jsou třídy, obsahující metody pro použití v jiných třídách. V OctoPrintu jsou to speciální pluginové třídy rozšiřující aplikaci. Pro OctoPrint jich je celkem dvanáct a každý z těchto mixins nabízí jiné možnosti rozšíření.

V rámci rešerše jsem si již vyzkoušel naimplementovat jednoduchý plugin, který používá několik mixins. První z nich byl `TemplatePlugin` mixin, který umožňuje přidávat tlačítka a záložky do rozhraní OctoPrintu. Další z nich je `SettingsPlugin` mixin, který, jak už vyplývá z názvu, pomáhá manipulovat s nastavením OctoPrintu.

Každý plugin může implementovat i více mixins, které se definují pomocí `__plugin_implementation__`. Jádro OctoPrintu tyto mixins vybere od jednotlivých pluginů a poskytne jejich metody aplikaci.

Pro tvorbu pluginů je důležité vědět, že některé druhy mixins, například `StartupPlugin` nebo `ShutdownPlugin`, podporují různé pořadí vykonání. Je tedy možné určit, který mixin se provede dříve než ostatní. Právě zmíněné dva mixins toto provádí. Dělá se to tak, že mixins, které potřebují mít dané pořadí, implementují `SortablePlugin` mixin. Pokud je nějaká metoda volána pluginem, tak subsystém OctoPrintu zajistí správné zařazení v pořadí.

Seznam jednotlivých pluginových mixins z [13]:

StartupPlugin – umožňuje provádět operace při startu OctoPrintu. Například zapnutí dalších služeb.

ShutdownPlugin – umožňuje provádět oprerace při vypínání OctoPrintu. Využívá se například, když je použit `StartupPlugin` mixin, na uzavření nějaké další služby.

SettingsPlugin – slouží pluginům, které potřebují manipulovat s vlastní konfigurací. Každé rozšíření OctoPrintu obsahující tento mixin bude obsahovat vlastní instanci `PluginSettingsManager`. Ta bude už správně nainicializovaná, aby ji mohl plugin použít. Tento mixin obsahuje metody, díky kterým může měnit položky v konfiguraci z rozhraní. Také je možné určit, které položky budou přístupné pouze správci, které uživateli a které nikomu.

AssetPlugin – tento mixin umožňuje pluginu nadefinovat si nové statické vylepšení. Tím se myslí nějaké rozšíření pomocí javascriptu nebo CSS. Díky `AssetPlugin` jsou pak tyto vylepšení zanesené přímo do uživatelského rozhraní OctoPrintu.

TemplatePlugin – umožňuje přidávat vlastní komponenty do webového rozhraní. Je možno rozšířit navigační pole, postranní menu, jednotlivé záložky, nastavení, průvodce a informace o OctoPrintu. Díky němu se mohou i původní šablony nahradit novými.

WizardPlugin – tento mixin umožňuje pluginu manipulovat s průvodci.

UiPlugin – umožňuje pluginu kompletně nahradit uživatelské rozhraní. Může se to hodit například při vytváření mobilní aplikace pro OctoPrint, protože je možné uvést na jakých platformách má mixin fungovat.

SimpleApiPlugin – pluginy využívající tento mixin mohou implementovat jednoduché API postavené HTTP metodách GET a POST.

BlueprintPlugin – tento mixin umožňuje pluginům vytvářet nové stránky, pro jakékoliv účely bude plugin potřebovat. Pro použití BlueprintPluginu potřebujete často i TemplatePlugin, abyste si nadefinovali novou šablonu.

EventHandlerPlugin – tento mixin umožňuje manipulovat s událostmi. Například pokud 3D tiskárna dotiskne, vznikne událost `PrintDone`. Plugin pak může toto využít a poslat majiteli tiskárny zprávu, že tisk byl hotov.

ProgressPlugin – mixin umožňující pluginu kontrolovat stav tisku nebo stav nařezání 3D objektu.

SlicerPlugin – umožňuje přidávat možnosti pro slicery[3], které jsou používány v OctoPrintu.

RestartNeedingPlugin – pro pluginy, které když se povolí/zakáží, tak potřebují restart celého OctoPrintu pro správnou funkčnost.

ReloadNeedingPlugin – jsou podobné jako `RestartNeedingPlugin`, ale stačí u nich obnovit uživatelské rozhraní OctoPrintu.

1.3.2 Hooks

Hooks jsou podobné mixins, ale narozdíl od nich jsou zaměřené na jednotlivé metody. Hooks jako takové umožní programu se za jistých podmínek chovat jinak, než jak je zadáno ve výchozích hodnotách. Dává to vytvářenému pluginu možnost přepsání některých metod, a tak poskytnout jistou flexibilitu. Tedy když potřebujeme udělat něco více specifického a mixin by byl příliš komplikovaný pro implementaci, použijeme hooks.

Hooks existuje v současnosti 22 a naleznete je na [14]:

- `octoprint.accesscontrol.appkey` – umožňuje registrovat další API klíče.
- `octoprint.accesscontrol.keyvalidator` – poskytuje možnost validovat vlastní API klíče.

[3] *Slicer* – Program pomocí něhož se 3D modely připraví pro tisk.

- `octoprint.cli.commands` – umožňuje registrovat nové příkazy pro příkazovou řádku OctoPrintu.
- `octoprint.comm.protocol.action` – díky tomuto hook je možné reagovat na libovolné zprávy od firmware.
- `octoprint.comm.protocol.atcommand.queuing` – hook manipulující s určitými speciálními příkazy v gcode.
- `octoprint.comm.protocol.atcommand.sending` – hook manipulující s určitými speciálními příkazy v gcode.

Následující 4 hooks manipulují se samotnými gcode instrukcemi. Ty se mohou nacházet v různých fázích. Zařazující se do fronty k tisku, být zařazené, být odesílané a odeslané k tisku. Tyto hooks mohou například kompletně potlačit různý příkaz pro gcode.

- `octoprint.comm.protocol.gcode.queing`.
- `octoprint.comm.protocol.gcode.queued`.
- `octoprint.comm.protocol.gcode.sending`.
- `octoprint.comm.protocol.gcode.send`.
- `octoprint.comm.protocol.gcode.received` – hook, který se dá použít, když tiskárna pošle, že byla instrukce pro tiskárnu přijata.
- `octoprint.comm.protocol.gcode.error` – hook, který pracuje s chybovými hláškami od 3D tiskárny.
- `octoprint.comm.protocol.scripts` – umožňuje pracovat se skripty.
- `octoprint.comm.protocol.temperatures.received` – používá se při zacházení s formátovanými daty ohledně teploty z 3D tiskárny. Umožňuje je modifikovat.
- `octoprint.comm.transport.serial.factory` – poskytuje možnost manipulovat s připojením k tiskárně.
- `octoprint.filemanager.extension_tree` – umožňuje manipulovat se soubory, které jsou nahrávané do rozhraní, například zakázat určitý typ souboru.
- `octoprint.filemanager.preprocessor` – umožňuje pracovat s vnitřní strukturou nahraných souborů.
- `octoprint.printer.factory` – hook, který vrací `PrinterInstance`. Umožňuje nahradit původní hodnoty OctoPrintu. Tento hook se volá pouze při zapínání serveru.

- `octoprint.server.http.bodysize` – umožňuje pracovat s maximální velikostí stránky a nahrávaných souborů.
- `octoprint.server.http.routes` – poskytuje rozšíření seznamu cest registrovaných na webovém serveru.
- `octoprint.ui.web.templatetypes` – umožňuje rozšířit množinu podporovaných šablon na webovém rozhraní.
- `octoprint.users.factory` – tento hook vrací instanci `userManager`, která manipuluje s přihlášením uživatelů. Volá se pouze při startu OctoPrintu. Pokud je pomocí tohoto hook vráceno `None`, bude přihlášení přizazeno globálnímu `userManager`. Současná implementace OctoPrintu využívá takzvaný `FileBasedUserManager`, který informace o uživatelích vyzvedává a ukládá pomocí `users.yaml`.

1.3.3 Helpers

Helpers jsou metody, které mohou být poskytnuty pro další pluginy, aby byla dostupná běžná funkčnost. Tedy můžeme naprogramovat plugin, který bude využívat jiný plugin, který je nainstalovaný. Tomuto pluginu můžeme už při instalaci říct závislosti, které musí být pro použití pluginu nainstalovány.

1.3.4 ViewModels

Pokud je potřeba při tvorbě pluginu implementovat nějaké frontendové komponenty, tak dříve či později budete potřebovat vlastní Knockout view models. Tyto vlastní modely se připojují k ostatním z OctoPrintu pojmenovaných `OCTOPRINT_VIEWMODELS`. Definují se pomocí `construct`, který funguje jako konstruktor pro daný model a je obvykle samotnou třídou. Dále se definuje jménem view modelu. Pokud už existuje model, který má stejné jméno, nastane chyba, která se vypíše do logu v terminálu.

Další vlastnosti view modelu, které je třeba zmínit, jsou závislosti, které jsou potřeba pro správný chod tohoto modelu. Závislosti jsou dělené do dvou kategorií – nezbytných a volitelných. Pokud některá z povinných závislostí bude chybět, tak inicializace daného view modelu skončí chybou. Pokud je nadefinovaná některá závislost, tak pomocí nového view modelu může být také rozšířena nebo změněna. U závislostí je potřeba si dát pozor, aby nevzniklo zacyklení mezi nimi. Například aby model A nezávisel na B, model B na C a C na A. Toto by způsobilo chybu.

V současnosti existuje celkem dvacet těchto view modelů. Za zmínku podle mého názoru stojí hlavně dva, které budu potřebovat pro tvorbu pluginu. Jsou to `loginStateViewModel`, `settingsViewModel`. `loginstateViewModel` je jedna z nejdůležitějších komponent. Je na ní závislá většina ostatních view modelů. Ovládá přihlášení do rozhraní OctoPrintu. `SettingsViewModel`, jak už vyplývá z názvu, je model pro správné zobrazení nastavení.

Je nutné podotknout, že tyto view modely jsou spojené se šablonami Jinja2 souborů přes data-bindy. Knockout efektivně zvládá spojovat data pro texty, HTML, CSS a různé atributy [9].

1.3.5 Jinja2

Jinja2 je moderní šablonovací jazyk pro Python. Je namodelovaný po vzoru frameworku Django [10]. Šablony OctoPrintu, jak už jsem zmínil, jsou snadno nahraditelné. Jednotlivě bych je rozdělil do několika kategorií – šablony pro navigační menu, záložky a boční menu. Tyto prvky jsou vidět pořád. Dále pak existují šablony pro nastavení, informace o OctoPrintu a nebo také šablona, která se používá při prvním spuštění OctoPrintu.

1.3.6 Existující pluginy do OctoPrintu

Existuje mnoho pluginů, ale žádný takový, který by se zaměřoval na autorizaci pomocí serveru třetí strany. První množina pluginů je zaměřená na vylepšení rozhraní aplikace OctoPrint a druhá množina je zaměřená na přímé ovládání 3D tiskárny. Například plugin, díky kterému je možné upravovat gcode přímo v aplikaci OctoPrint, nebo plugin, který v případě, že se tiskárna nahřála, ale dlouho se na ní netisklo, tak automaticky vypne nahřívání trysky i podložky.

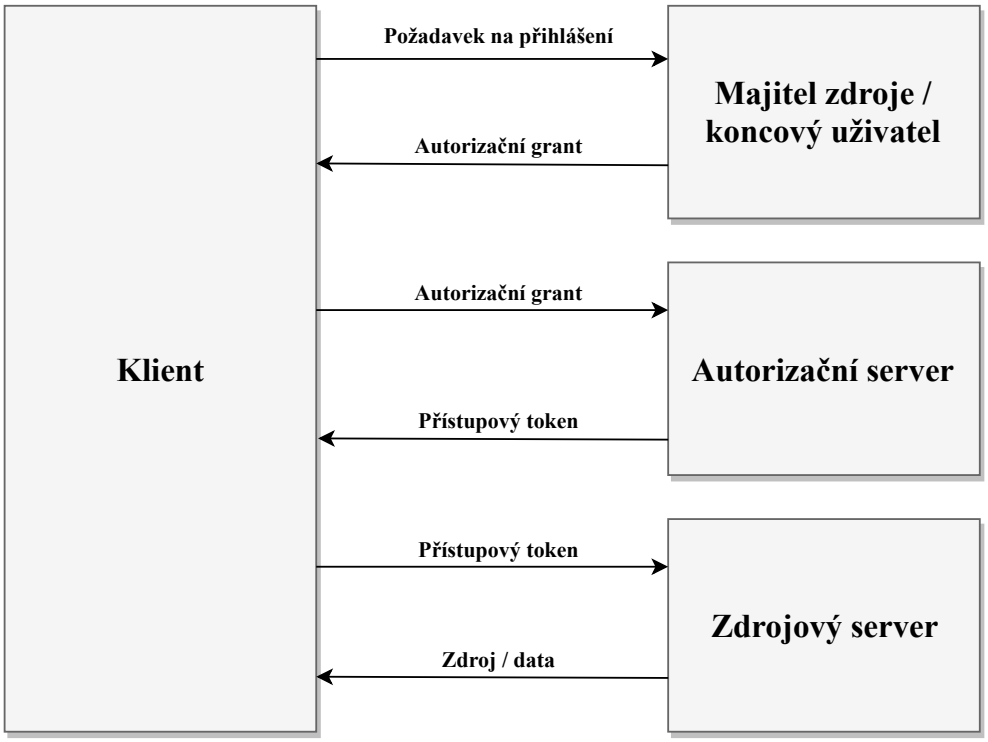
1.4 OAuth 2.0

OAuth 2.0 [15] je autorizační framework umožnující aplikacím třetích stran obdržet omezený přístup k nějaké webové službě. Tento framework (resp. protokol) pro autorizaci nahrazuje již zastaralý OAuth 1.0 [16] protokol z roku 2006. Hlavní výhoda OAuth 2.0 (dále jen OAuth2) je taková, že uživatel může poskytnout klientské aplikaci přístup ke svým datům z nějakého serveru, aniž by klientské aplikaci musel poskytnout své přihlašovací údaje.

1.4.1 Základní principy

V klasickém modelu klient-server klient žádá o data ze serveru, kde je omezený přístup a je potřeba autorizace. To přináší několik problémů, jako kde si bude klientská aplikace ukládat hesla uživatelů? Jak servery, kde jsou informace uloženy, zjistí, že informace dává správnému klientovi?

OAuth2 tohle řeší tak, že se nepřihlašuje přímo ke zdroji informací, ale nejdříve k autorizačnímu serveru, kde se koncový uživatel přihlásí. Tím dá vědět serveru, že může poskytnout jeho informace. Autorizační server ale nemá k informacím o uživateli přístup. Tento server pošle zpátky klientské aplikaci přístupový token, díky kterému uživatel může skrz klientskou aplikaci přistoupit ke svým datům na zdrojovém serveru. Na úvod je potřeba vysvětlit několik základních rolí.



Obrázek 1.4: Proces autorizace použitím protokolu OAuth 2.0

- První z nich je takzvaný majitel zdroje. Tento majitel (dále jako koncový uživatel) má někde uchovaná svá data, která by chtěl poskytnout klientské aplikaci. Majitel zdroje musí dát klientské aplikaci povolení (grant), který pak může klientská aplikace použít. Existuje několik typů grantů, které popíšu v podsekci 1.4.2 – Autorizační granty.

- Další z důležitých rolí je server, na kterém jsou data o uživateli uchována (dále jako zdrojový server). Ke zdrojovému serveru přes OAuth2 nelze přistoupit přímo. Aby klient mohl přistoupit k datům, potřebuje přístupový token z autorizačního serveru.

- Třetí z rolí je klient. Uživatel se skrze klienta (resp. klientskou aplikaci) pomocí autorizačního povolení přihlašuje k autorizačnímu serveru, aby obdržel přístupový token. Díky tomuto tokenu si klientská aplikace může vyzvednout data ze zdrojového serveru.

- Poslední z rolí, které si nadefinujeme, je autorizační server, na kterém se uživatel přihlásí, dá vědět, jaká data má zdrojový server poskytnout. Autorizační server pak vytvoří přístupový token.

1.4.2 Autorizační granty

Autorizační grant je pověření od koncového uživatele pro klientskou aplikaci. Tento grant se využívá pro získání přístupového tokenu od autorizačního serveru. Existují čtyři druhy autorizačních grantů. Autorizační kód, implicitní grant, přístupové údaje majitele zdroje, klientské pověření.

- Autorizační kód je typ pověření od koncového uživatele. Tento je obdržen od serveru služby po přihlášení uživatele. Klientská aplikace poté může tento kód použít na autorizačním serveru, aby obdržela přístupový token. Server, na kterém se uživatel přihlásil, předá tuto informaci autorizačnímu serveru. Díky tomu klientská aplikace vůbec nepotřebuje znát přihlašovací údaje koncového uživatele.

- Implicitní grant je zjednodušení verze autorizačního kódu. Místo toho, aby ale klient dostal pouze kód, obdrží přímo přístupový token. V tomto případě se už není potřeba identifikovat na autorizačním serveru. Implicitní granty sice mohou přinést větší efektivitu pro nějaké klientské aplikace, ale velmi to snižuje bezpečnost dat.

- Třetí z možnosti je přímé získání přístupových údajů uživatele jako je přihlašovací jméno a heslo. Tento způsob pověření by měl být použit pouze v případě velké důvěry mezi klientskou aplikací a koncovým uživatelem.

- Klientské pověření je typ grantu, který se používá, když je určen nějaký rámec poskytnutých dat ze zdrojového serveru. V případě tohoto pověření má klientská aplikace stejné možnosti jako samotný koncový uživatel. Tedy aplikace může přistupovat ke zdroji informací.

1.4.3 Klient a servery

Před používáním protokolu OAuth2 je potřeba si vytvořit klientskou aplikaci na autorizačním serveru. Každá klientská aplikace se vyznačuje třemi základními prvky. Jsou to `client_id`, `client_secret` a `redirect_uri`. Indetifikátor klientské aplikace `client_id` je unikátní řetězec pro každý klient na serveru služby. Druhý prvek je `client_secret`, které slouží jako heslo klientské aplikace. Třetí hodnota je přesměrovací adresa zpět do aplikace, ze které se autorizace provádí.

Pro OAuth2 se používají dva hlavní servery. Jedním z nich je autorizační server, na který se klientská aplikace přihlásí pomocí pověření od koncového uživatele. Ten poté předá zpátky aplikaci přístupový token, nebo v případě špatné autorizace předá chybový kód a informaci o chybě. Druhý je zdrojový server, na kterém jsou uloženy informace o koncovém uživateli, které mohou být důvěrné.

1.4.4 Typy tokenů

Existují dva tokeny, které se využívají při autorizaci pomocí OAuth2 – přístupový token a obnovovací token.

Přístupové tokeny jsou způsob, jakým jsou uloženy přístupové údaje, aby se aplikace dostala k chráněným informacím. Obvykle je to náhodný řetězec vygenerovaný autorizačním serverem. Tento řetězec je pro klientskou aplikaci nesrozumitelný. Přístupový token v sobě může obsahovat informaci, jak hodně omezené informace má zdrojový server poskytnout. Přístupový token nelze používat do nekonečna. Váže se k němu i volitelná informace o tom, kdy daný token už nebude platit.

Obnovovací tokeny se používají k tomu, aby klientská aplikace mohla obdržet od autorizačního serveru nový přístupový token, když ten starý přestane platit. I tento obnovovací token je řetězec, který je pro klientskou aplikaci nečitelný. Obnovovací token, narozdíl od přístupového, se ale nedostane do styku se zdrojovým serverem.

1.4.5 Přístupový token

V případě, že naše aplikace potřebuje přístupový token, pošle na autorizační server grant od koncového uživatele. Jsou dvě odpovědi, které aplikace od autorizačního serveru dostat – úspěch či chyba.

V případě úspěchu dostane také od serveru přístupový token. Dále obdrží jeden z typů tokenu, v současné době existují dva: bearer [17] a mac [18]. Dále může aplikace získat dobu, za jakou platnost tokenu vyprší. Tato hodnota je doporučovaná.

V případě neúspěchu odezvy můžeme obdržet několik možností napovídající chybě. Chyba `invalid_request` může značit chybějící požadovaný parametr, nebo například že se jeden parametr objevil dvakrát. Další z možností je `invalid_client`, který značí to, že bylo špatně zadáno `client_id` nebo `client_secret` během autorizace. Další eventualita je `invalid_grant` uvádějící špatně použitý grant. Tato chyba například nastane, pokud se použije špatný autorizační kód [15].

1.4.6 Aplikace využívající OAuth 2.0

Na závěr této sekce bych rád zmínil několik aplikací, pomocí nichž je možné použít protokol OAuth 2.0. Jsou to často sociální sítě jako Facebook, Instagram, Twitter nebo LinkedIn [19]. Dále se dá využít tento způsob autorizace přes Google. Další možností je využít OAuth2 přes GitHub, kde je poměrně jednoduché si vytvořit vlastní klientskou aplikaci. Jako poslední bych zmínil ČVUT v Praze, kde je možné si také vytvořit vlastní klientské aplikace a využívat je s autorizačním serverem na FIT ČVUT v Praze [20]. V kombinaci s KOSapi [21] se dají pak dělat webové aplikace jako je například 730ne.cz [22].

Kapitola 2

Návrh

V této části se budu zabývat návrhem vytvářeného rozšíření pro aplikaci OctoPrint [1]. V první části jsou vyjmenovány jednotlivé prvky OctoPrintu použité na vývoj pluginu. Dále je popsán způsob použití autorizace pomocí OAuth 2.0 [15] a vybrán autorizační grant. Následně je detailně uveden návrh pluginu OctoPrintu a vztahy mezi jednotlivými komponenty. Poté je navržen způsob ukládání potřebných dat pro autorizaci pomocí protokolu OAuth 2.0 a možnost konfigurace pluginu. Na závěr jsou popsány funkční a nefunkční požadavky na toto rozšíření pro aplikaci OctoPrint.

Součástí této kapitoly jsou také diagramy popisující vztahy mezi jednotlivými operacemi v aplikaci a kusy kódu, které pomáhají ujasnit současné fungování aplikace OctoPrint.

2.1 Potřebné komponenty pluginu OctoPrintu

Na úvod je nutné si určit, které ze zmíněných možností pro tvorbu pluginu bude aplikace vyžadovat. Je potřeba nahradit stávající přihlášení OctoPrintu využívající `FileBasedUserManager`. Dále přímo nahradit tlačítko pro přihlášení. Vytvořit si vlastní view model, který bude rozšiřovat jiné dva modely `loginStateViewModel` a `settingsViewModel`.

2.1.1 Vlastní userManager

Pro tvorbu pluginu je nutné naimplementovat si vlastní třídu `userManager`. To dokážeme pomocí hook z `octoprint.users.factory`, který se použije při startu serveru a vrátí OctoPrintu novou instanci `userManager`.

Tato instance rozšiřuje nebo nahrazuje základní `userManager` a implementuje jeho metody. Rozhodl jsem se, že pro účely pluginy vytvořím vlastní třídu `OAuthBasedUserManager`, která bude dědit od a pracovat tak s nějakými metodami od již využívaného `FileBasedUserManager`. Některé metody je třeba nahradit a další doplnit.

```
user = octoprint.server.userManager.findUser(username)
if user is not None:
  if octoprint.server.userManager.checkPassword(username,
                                                password):
    if not user.is_active():
      return make_response(("Your account is deactivated",
                            403, []))

    if octoprint.server.userManager.enabled:
      user = octoprint.server.userManager.login_user(user)
      session["usersession.id"] = user.session
      g.user = user
    login_user(user, remember=remember)
    identity_changed.send(current_app._get_current_object(),
                          identity=Identity(user.get_id()))
    return jsonify(user.asDict())
return make_response(("User unknown or password incorrect",
                      401, []))
```

Ukázka kódu 2.1: Část kódu současného přihlašování uživatelů v OctoPrint

2.1.1.1 Metoda pro nalezení uživatele

Ve `FileBasedUserManager` existuje metoda `findUser`, která vyhledává uživatele podle identifikátoru zadaného jako parametr. Kvůli současné implementaci OctoPrintu je nutné tuto metodu přepsat. Metoda `findUser` se na rozhraní serveru volala, aby ověřila, zda daný uživatel existuje v textové podobě v konfiguračním souboru pro uživatele `users.yaml`. Pro tento plugin ale potřebuji, aby se do OctoPrintu přihlásil kdokoliv s omezenými právy. Takový uživatel ale nemusí ještě existovat.

Nová metoda musí také najít uživatele, který je již zaznamenaný vkonfiguračním souboru pro uživatele `users.yaml`. Navíc musí také ale pro nové uživatele vytvořit dočasného uživatele, aby dále mohlo dojít ke kontrole náležitostí protokolu OAuth2. Ty se podle mého návrhu budou provádět v metodě `loginUser`.

2.1.1.2 Metoda pro kontrolu hesla

Ve stávající implementaci OctoPrintu existuje metoda `checkPassword`, která kontroluje hesla a tu je potřeba nahradit. V současné podobě se volá na serveru v API při přihlašování. Metoda již nebude potřeba, neboť přihlašování bude probíhat pomocí protokolu OAuth 2.0. I přesto je ji potřeba nahradit, aby se nevolala její současná podoba.

2.1.1.3 Metoda pro přihlášení uživatele

Nejdůležitější metodou pro autentizaci je samozřejmě metoda `login_user`. Ta vrací samotného objekt reprezentující uživatele, jehož přihlašovací jméno se později zobrazí v uživatelském rozhraní OctoPrintu. I tuto metodu OctoPrintu je třeba nahradit. Protože je možné, aby k 3D tiskárně mohlo být přihlášeno skrze rozhraní více uživatelů současně, vytvoří OctoPrint pro každého z nich samostatnou relaci.

Tuto možnost OctoPrintu jsem také převzal do návrhu rozšíření pro aplikaci OctoPrint. Dále bude tato metoda potřebovat informace ke správnému pracování autorizace pomocí OAuth 2.0. Z uživatelského rozhraní se předají informace o autorizačním kódu a přesměrovací adresa.

Pro každou přesměrovací adresu budou existovat dva údaje. Klientský identifikátor `client_id` a klientův tajný klíč `client_secret`. Důvod je ten, že může existovat více přesměrovacích adres, které se vážou k různým klientským aplikacím. Pro každou přesměrovací adresu existuje jedno `client_id` a jedno `client_secret`. Následně metoda provede autorizaci pomocí nových metod `get_token` a `get_username`, které jsou dále popsány.

Autorizace je prováděna pomocí autentizační knihovny jazyka Python `requests_oauthlib` [23]. Metoda `login_user` pomoci ní vytvoří relaci, ve které si plugin s autentizačním serverem předávají data. Jak je tato knihovna využita detailněji a jaký autorizační grant byl vybrán, je popsáno v sekci 2.2.

Každý nový uživatel, který se přihlásí přes protokol OAuth2, bude mít práva uživatele. Tedy nebude moci konfigurovat OctoPrint, ale může provádět operace, jako je spuštění tisku. Informace o tomto uživateli se uloží do konfigurační složky uživatelů `users.yaml`. Pokud by chtěl tento uživatel větší oprávnění, musí požádat správce, aby mu je přidal buď přes webové uživatelské rozhraní a nebo přímou změnou konfiguračního souboru.

Metoda také musí provést patřičné kontroly, zda jsou informace správné, a popřípadě vypsat chybovou hlášku. Přihlášení kontroluje, zda jsou, nebo nejsou přítomné patřičné hlavičky v nastavení. Pokud chybí, tak se hlavičky pro autorizaci neposílají. Dále hlídá, jestli přístupový token, který dorazil, není prázdný. Na závěr kontroluje, zda uživatelské jméno, které bylo obdrženo, není prázdné.

2.1.1.4 Metoda pro získání přístupového tokenu

Nová metoda pluginu pro autorizaci přes OAuth2 vezme grant v podobě autorizačního kódu jako parametr, dále `client_id` a `client_secret`. Poslední parametr je potom relace vytvořená autentizační knihovnou Pythonu `requests_oauthlib` [23]. Pomocí této relace, vytvořené v metodě `login_user`, je umožněno zaslat kód na autorizační server. V případě obdržení správného kódu vrací pluginu přístupový token.

Metoda také musí obsahovat ošetření, zda je token správný, aby nenarušila správný chod aplikace OctoPrint. V případě neúspěchu musí do konzole vypsat chybovou hlášku, aby o tom byl uživatel informován.

V některých případech je také potřebné posílat na autorizační servery správné hlavičky požadavků. I takové případy musí metoda ošetřovat.

2.1.1.5 Metoda pro získání uživatelského jména

Tato nová metoda dostane parametr relaci pro autentizaci pomocí protokolu OAuth2. Skrz ní využije přístupový token k získání informací o uživateli. Z informací je potřeba získat uživatelské jméno přihlašované osoby.

Pro získání dat ze zdrojového serveru je potřebné poslat požadavek, většinou metodou GET, se správným přístupovým tokenem. Protože se zdrojové servery mezi sebou liší, může nastat problém, pod jakým klíčem je potřeba předat přístupový token a který klíč je třeba použít pro vyzvednutí uživatelského jména přihlašovací osoby.

Tady přijdou na řadu správná data a možnost konfigurace, ke které se dostanu později. Uživatel zadá do konfiguračního souboru klíč pro přístupový token. Na serveru GitHub je to například `access_token` a na serveru FIT ČVUT v Praze pouze `token`.

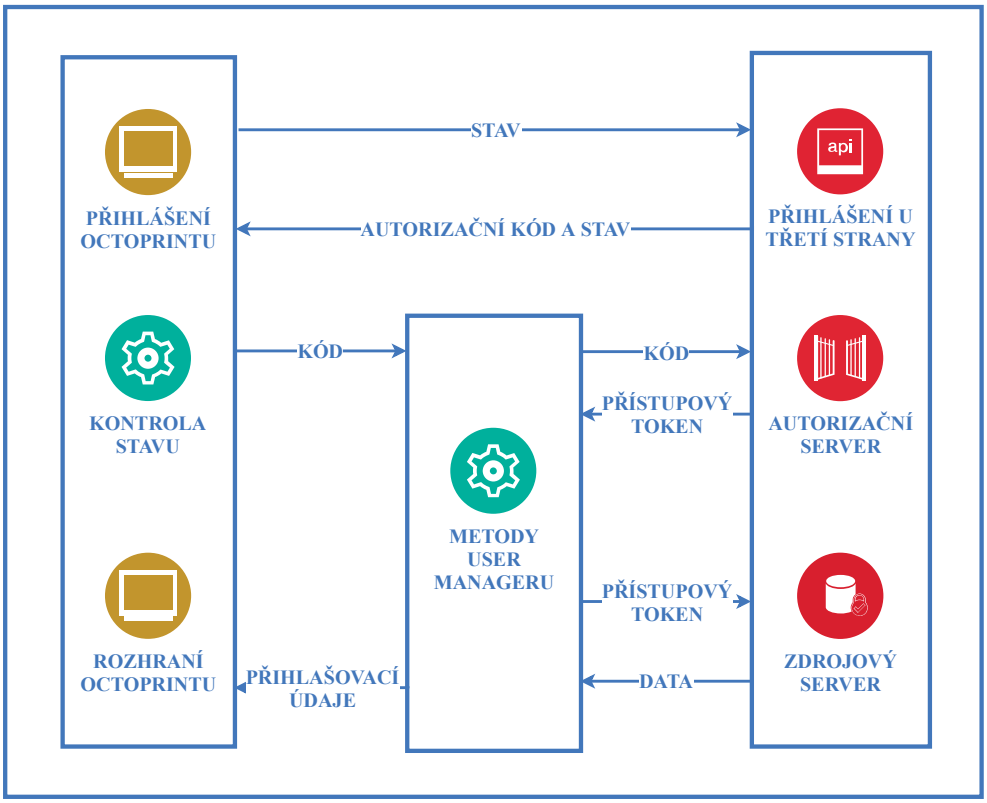
Podobný problém nastává při získání uživatelského jména. Na jednom zdrojovém serveru se identifikátor uživatele může skrývat pod názvem `login`, na jiném serveru `user_id` a na třetím jako `username`. Konfigurace musí poskytovat i možnost nastavit klíč, pod kterým má být uživatelské jméno. Pokud je tohle jediná špatně nastavená položka, můžeme uživateli vypsat všechna jím poskytnutá data, využili jsme totiž jeho přístupový token. Přihlašovaná osoba pak z konzole, na které běží aplikace OctoPrint, může zjistit správný klíč.

2.1.2 ViewModel

Pro ovládání OctoPrintu, jak už bylo uvedeno v analýze, se používají takzvané view modely. Pro plugin tedy bude potřeba vytvořit vlastní. Tento model musí nahradit původní přihlašování i odhlašování.

Plugin je navržen tak, že autorizace pomocí protokolu OAuth 2.0 se spustí právě ve view modelu, přesměruje uživatele OctoPrintu na přihlašovací server třetí strany a jakmile se uživatel přihlásí, je přesměrován zpět do webového rozhraní OctoPrintu. Na server je také poslán náhodný vygenerovaný řetězec pojmenovaný `state` a aby tento stav nebyl ztracen, je uložen do lokálního úložiště prohlížeče.

ViewModel následně odchytne URL parametry metody GET, které jsou poslány po přihlášení a přesměrování zpět. Jeden z nich je autorizační kód a druhý dříve vygenerovaný řetězec. Parametr `state` je z důvodu bezpečnosti potřeba zkontrolovat. Pokud se tedy stav neshoduje s tím, který jsme obdrželi od serveru a stav uložení v lokálním úložišti, tak uživatel nebude přihlášen.



Obrázek 2.1: Způsob použití OAuth 2.0 v aplikaci OctoPrint

Následně ViewModel využije tyto údaje a přesměruje je na přihlášení do serverového rozhraní OctoPrintu, které už používá `userManager` pluginu.

Aby se uživatel odhlásil kompletně, je potřeba se také odhlásit na serveru, na kterém se uživatel přihlásil pomocí OAuth2.

2.1.3 Šablony

Šablona, kterou je potřeba nahradit, je ta, pomoci níž se uživatel přihlásí. Tedy nahradit tlačítko na přihlášení tlačítkem pluginu. To je pro tvorbu pluginů umožněno díky `SettingsPlugin` mixinu, který obsahuje metodu nastavení šablon – `get_template_configs`. Tuto metodu je možné použít a určit, zda komponenta OctoPrintu bude nahrazena.

Dále by bylo dobré, aby po přihlášení správce mohl nastavovat jednotlivé proměnné z konfigurace. Tuto funkcionalitu zatím do návrhu nepřidávám, ale pro další rozšíření by to bylo možné.

2.2 Způsob použití OAuth 2.0

Pro účely pluginu jsem se rozhodl, že plugin bude využívat OAuth2 grant autorizační kód. Ten se nejčastěji používá pro webové aplikace [24]. Je to hlavně proto, že tento typ povolení je jeden z bezpečnějších.

Pro tyto účely jsem se rozhodl použít autentizační knihovnu Pythonu `requests_oauthlib`, která rozšiřuje další knihovnu jménem `requests`. Pomocí této knihovny vytvořím v user manageru relaci, během níž si plugin může s autorizačním serverem bezpečně předávat informace. OctoPrint sám o sobě však nevyužívá SSL[4].

Plugin po přihlášení uživatele dostane grant ve formě přístupového kódu. Tento kód se poté využije na autorizačním serveru třetí strany. Ze serveru obdržíme přístupový token, který je samotnému pluginu nečitelný, využijeme ho ale na zdrojovém serveru pro obdržení uživatelských údajů.

2.3 Návrh ukládání dat

Pro správné fungování pluginu je potřeba ukládat několik položek. OctoPrint pluginům umožňuje ukládat si své konfigurace taktéž do souboru `config.yaml`. Jak už jsem dříve uvedl, OctoPrint umožňuje zamezit přístup k určitým datům konfiguračního souboru díky `SettingsPlugin` mixinu. Samozřejmě to nezamezí například fyzickému přístupu k SD kartě na Raspberry Pi [25], na které je OctoPrint nahraný.

2.3.1 Potřebná data pro plugin

Potřebná data pro správné fungování pluginu jsou tato:

- URL adresa, kam se má uživatel přihlásit u třetí strany.
- URL adresa autorizačního serveru, kde plugin vyzvedne přístupový token.
- URL adresa zdrojového serveru, kde plugin vyzvedne data o uživateli.
- Název klíče, pod kterým se uloženo přihlašovací jméno. Data ze zdrojového serveru jsou v `JSON` formátu. Uvnitř dat jsou záznamy ve formátu klíč:hodnota. Na různých zdrojových serverech se totiž identifikátor může skrývat pod jiným klíčem. Například pro server GitHub je to `login` a pro server FIT ČVUT v Praze je tato hodnota `user_id`.
- Název klíče přístupového tokenu, u kterého je podobný problém. Na jednom zdrojovém serveru se využívá přístupový token jako parametr URL `token` a na druhém `access_token`.

[4] *Secure Sockets Layer* – Protokol využívající se pro bezpečnou komunikaci s webovými servery

- Hlavičky, které se posílají na autorizační server. To je z důvodu. že na některých serverech je potřeba specifikovat v jakém formátu aplikace přijme přístupový token. Plugin využije pouze `JSON`, a je tedy potřebné nakonfigurovat hlavičky, které se posílají na autorizační server. Uživatel dále může nakonfigurovat i další hlavičky.

Pro správné fungování pluginu jsou potřeba ještě tři klíčové hodnoty. Jedna z nich je přesměrovací adresa. Tady nastává jeden problém. Při vytváření klientských aplikací na serverech, kde bude vygenerováno `client_id` a `client_secret`, je možné většinou zadat pouze jednu přesměrovací adresu.

Praktický příklad je u 3D tiskáren na FIT ČVUT v Praze. V učebnách se studenti přihlašují na adresu `printerX.ucebny`, kde X je číslo tiskárny, ale v 3D laboratoři je to `printerX.local`. Pokud by se tiskárna přenesla, musel by se změnit konfigurační soubor.

Můj návrh ale podporuje takovou možnost, že bude vytvořeno více klientských aplikací. Pro každou přesměrovací adresu pak budou uloženy dva prvky – klientův identifikátor a klientův tajný klíč. Těch pak může být v konfiguračním souboru klidně více, protože návrh je vymyšlen tak, aby přesměroval na adresu, odkud byla spuštěna autorizace.

2.3.2 Návrh a struktura konfigurace

Jak už bylo zmíněno v analýze, OctoPrint pro jednotlivé pluginy nabízí možnost ukládání vlastní konfigurace [12]. Ty jsou ukládány ve formátu `YAML` v konfiguračním souboru pod klíčem `plugins` a pod jménem pluginu Octoprintu – v mém případě `oauth2`.

Zmíněné položky v konfiguraci jsem pojmenoval následovně.

- `login_path` – cesta pro přihlášení,
- `token_path` – cesta k autorizačnímu serveru,
- `user_info_path` – cesta ke zrojovému serveru,
- `username_key` – klíč pro vyhledání hodnoty uživatele,
- `access_token_query_key` – klíč přístupového tokenu,
- `token_headers` – hlavičky pro autorizační server,
  - `nazev_hlavicky`: `hodnota`,
- URI:
  - `client_id` – identifikátor klientské aplikace,
  - `client_secret` – tajná hodnota klientské aplikace.

Hlaviček může být samozřejmě více. Taktéž i IP adres, na kterých by OctoPrint mohl fungovat. Pod každou IP adresou se pak nachází vlastní klientské údaje. Návrh konfigurace si můžete prohlédnout na v ukázce kódu 2.2.

```
plugins:
  oauth2:
    login_path: path_to_login_user
    token_path: paht_to_authorization_server
    user_info_path: path_to_resource_server
    username_key: username_key
    access_token_query_key: access_token_key
    token_headers:
      header1: your_header
      header2: your_second_header
    first_redirect_uri:
      client_id: first_client_id
      client_secret: first_client_secret
    second_redirect_uri:
      client_id: second_client_id
      client_secret: second_client_secret
```

Ukázka kódu 2.2: Návrh konfigurace pluginu v YAML formátu

```
plugins:
  oauth2:
    login_path: https://github.com/login/oauth/authorize
    token_path: https://github.com/login/oauth/access_token
    user_info_path: https://api.github.com/user
    username_key: login
    access_token_query_key: access_token
    token_headers:
      Accept: application/json
    http://0.0.0.0:5000/:
      client_id: first_client_id
      client_secret: first_client_secret
    http://localhost:5000/:
      client_id: second_client_id
      client_secret: second_client_secret
```

Ukázka kódu 2.3: Příklad konfigurace pluginu v YAML formátu

2.3.3 Funkční požadavky

Nyní když už je jasný návrh fungování a návrh konfigurace pluginu pro aplikaci OctoPrint, je třeba ještě uvést požadavky na aplikaci. Funkční požadavky popisují, jaké funkcionality musí plugin mít.

Přihlášení uživatele Uživatel musí mít možnost se přihlásit do webového rozhraní OctoPrintu.

Přesměrování na přihlášení Plugin přesměruje uživatele na přihlášení serveru třetí strany. Jakmile se tam uživatel přihlásí, přesměruje ho server zpět do uživatelského rozhraní OctoPrintu.

Zachování správné funkcionality OctoPrintu Aplikace musí zochovávat normální fungování OctoPrintu a nesmí ho omezovat.

Snadná integrace do OctoPrintu Rozšíření se musí dát snadno nainstalovat do aplikace OctoPrint. To by mělo jít přes utilitu pip a nebo přímo ve webovém rozhraní.

Možnost konfigurace Uživatel OctoPrintu [1] musí mít možnost chování nastavit konfigurační soubory. To by mělo být uživatelsky srozumitelné.

Odhlášení uživatele Plugin musí zajišťovat i správné odhlášení uživatele z webového rozhraní aplikace OctoPrint.

2.3.4 Nefunkční požadavky

Tento druh požadavků popisuje vlastnosti pluginu OctoPrintu, které nesouvisí s tím, co by mohl uživatel ve webovém rozhraní ovlivnit.

Zajištění zabezpečení Pro autorizaci přes protokol OAuth 2.0 je nezbytné aby aplikace poskytovala dostatečnou míru zabezpečení. Případný útočník se nesmí dostat k důvěrným informacím, jako je například klientův tajný klíč.

Intuitivní uživatelské rozhraní Rozšíření pro OctoPrint by mělo být pro uživatele intuitivní a neomezovat je v dalších aktivitách prováděných na OctoPrintu.

Rozšiřitelnost Plugin by měl být naimplementovaný tak, aby se i po dokončení této bakalářské práce mohly vytvářet další rozšíření a vylepšení.

Testování Implementace by měla být doplněná o dostatečné množstvím jednotkových testů kontrolující metody výsledného pluginu a také o integrační testy, které zkontrolují správné zasazení do aplikace OctoPrint.

Členění kódu Kód by měl být řádně členěn a splňovat určitou kvalitu podle konvencích použitých jazyků. Třídy a metody musí být zdokumentované a okomentované.

Licence Celý plugin by měl být volně dostupný a být pod svobodnou licencí.

Kapitola 3

Realizace

V této kapitole se budu zabývat implementací pluginu pro aplikaci OctoPrint. Na začátek uvedu použité nástroje a knihovny, dále je popsána adresářová struktura, ve které se nachází zdrojové kódy rozšíření pro aplikaci OctoPrint. Následně je popsána realizace uvedených funkčních požadavků na plugin. Na závěr jsou zde uvedeny ukázky kódu vnitřního fungování OctoPrintu a realizace protokolu OAuth 2.0.

3.1 Použité nástroje

V této sekci jsou vyjemnovány nástroje, které byly použity při tvorbě mé bakalářské práce.

3.2 Použité frameworky, knihovny a moduly

Tato sekce je věnovaná neobvyklým knihovnám, které byly použité pro vývoj nebo testování pluginu do aplikace OctoPrint.

Requests-OAuthlib [23] Tato knihovna využívá dvě jiné knihovny Pythonu – Requests a OAuthlib. Díky nim poskytuje snadno použitelné rozhraní pro autorizační protokoly OAuth 1.0 i po OAuth 2.0. Instalace je možná pomocí utility pip.

Requests [26] Dále je použitá také knihovna requests pro některé úkoly, které neposkytovalo rozhraní předešlé knihovny.

SocketServer [27] Pro testovací účely je potřeba vytvořit si vlastní falešný server pro OAuth2.0. K tomu posloužil právě modul SocketServer, který pochází ze standardní knihovny Pythonu. Z té samé knihovny byl také použit modul BaseHTTPServer [28], který umožňuje zacházet s požadavky `GET` a `POST`.

3.3 Adresářová struktura pluginu

V této sekci se budu zabývat adresářovou strukturou rozšíření pro aplikaci OctoPrint. Plugin je balíček Pythonu, který se dá nainstalovat pomocí utility pip. Pro dodržení správných konvencí je potřeba dodržet i adresářovou strukturu, která je následující:

```
setup.py ..... vygenerovaný instalační soubor
octoprint_oauth2
    static
        js ..... adresář pro view modely
    templates ..... adresář pro šablony
    tests ..... adresář pro testy a testovací server
    __init__.py ..... inicializační soubor pro plugin
    oauth_user_manager.py ..... soubor obsahující user manager
```

Uvnitř adresáře jsou soubory potřebné pro správný chod pluginu. Ve složce `static` je pouze adresář `js`, uvnitř kterého je soubor obsahující view model pluginu pro OctoPrint.

Adresář pro šablony obsahuje soubor `oauth2_login.jinja2`, díky kterému je možné přihlásit se přes protokol OAuth2.

Složka `tests` obsahuje testy, kterým se věnuji v kapitole 4.

3.4 Ukázky implementace

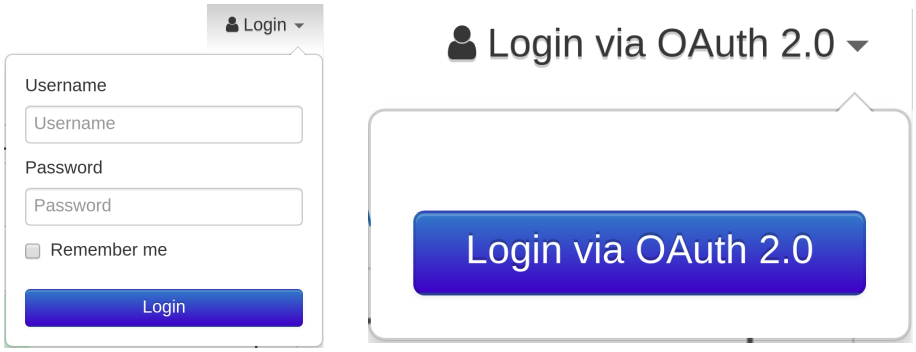
V této sekci budou předvedeny ukázky implementace pluginu pro aplikaci OctoPrint. Na začátek představím šablonu pro přihlášení `oauth_login.jinja2`. Poté vysvětlím implementaci view modelu. K čemuž se váže objasnění různých částí naimplementovaného kódu. Na závěr se budu věnovat popsání implementace vlastního `user_managera`, ve kterém probíhá většina autorizace přes protokol OAuth 2.0.

3.4.1 Šablona přihlášení

První část pluginu je přepsaná šablony na přihlášení. V inicializaci pluginu je uvedeno, že šablona `oauth2_login.jinja2` nahrazuje původní možnost přihlášení. Zadávání hesla nadále není potřeba, protože se uživatel přihlásí na serveru třetí strany.

3.4.2 View model

V této podčásti se budu věnovat view modelu napsaném v `KnockoutJS`. Vyvíjené rozšíření pro aplikaci OctoPrintu nahrazuje původní modely pojmenované



(a) Přihlášení přes klasický OctoPrint (b) Přihlášení přes výsledný pluginu

Obrázek 3.1: Přihlašování v OctoPrintu

`LoginStateViewModel` a `SettingsViewModel`. Nový model díky tomu může manipulovat s komponenty modelu přihlašování a nastavení.

Při kliknutí na tlačítko přihlášení přes OAuth 2.0 si plugin načte správná nastavení z konfiguračního souboru `config.yaml`. Pomocí vlastní URL adresy vybere správný klientský identifikátor a url přihlášení. Tajný klientský klíč se zde nenačítá. Dále si plugin vytvoří `state`, tedy náhodně vygenerovaný řetězec pro další kontrolu. Tento stav je uložen do perzistentní paměti prohlížeče. Z konfigurace poté plugin vytvoří parametry do URL adresy a uživatele na danou adresu přesměruje. Adresa obsahuje i parametr `state`.

Po správném přihlášení na serveru třetí strany by měl být uživatel přesměrován zpět na adresu OctoPrintu. To pracuje správně, i když tato aplikace je spuštěna lokálně. View model při zapnutí webového rozhraní zkontroluje, zda URL adresa obsahuje parametry `code` a `state`. Pokud ano, tak view model zkontroluje, jestli se vrácený stav po autorizace shoduje se stavem uloženým v perzistentní paměti prohlížeče. Pokud je stav jiný, vypíše se chybová hláška a autorizace se neprovede.

Po úspěšné kontrole náhodného řetězce se parametry `code` a současná url adresa OctoPrintu předají do `OctoPrint.browser.login`. Současná URL adresa se posílá proto, aby se vybrala správná konfigurace klientského identifikátoru a tajného klíče. `OctoPrint.browser.login` dále přesměruje tyto parametry na rozhraní serveru OctoPrintu, který využívá metody `userManager` pro přihlášení uživatele.

3.4.3 User manager

Největší část algoritmu je implementována v `OAuthBasedUserManager`. Tento user manager je nainicializovaný při startu serveru OctoPrintu pomocí hook `users.factory`, o kterém jsem psal v 1.3.2. Na začátku je uživatel vyhledán pomocí metody `findUser`, dále, pokud je uživatel nalezen, tak se přestoupí na kontrolu hesla. To je ale drobný zádrhel, protože uživatelé v souboru ještě

```
const code = getParameterByName("code", window.location.href);
const stateFromOAuth = getParameterByName("state",
    window.location.href);

if(!!stateFromOAuth && !!code){
    const state = localStorage.getItem("state");
    localStorage.removeItem("state");
    if (stateFromOAuth != state) {
        alert("State sent to oauth server is not same." +
             " Possible attack");
        return;
    }

    const url = parseUrl(window.location.href).origin + "/";
    const parameters = {"code": code,
                        "redirect_uri": url};

    OctoPrint.browser.login(parameters, "unused", false)
```

Ukázka kódu 3.4: Část kódu kontroly správného stavu a předání parametrů do API

nemusí existovat. Naštěstí jsem již v návrhu zmínil, že metodu pro nalezení uživatele také musíme přepsat a vytvořit dočasného uživatele.

```
def findUser(self, userid=None, apikey=None, session=None):
    user = FilebasedUserManager.findUser(self, userid,
                                         apikey, session)
    if user is not None:
        return user

    user = User(userid, "", 1, ["user"])
    return user
```

Ukázka kódu 3.5: Metoda `findUser`

Dále se v rozhraní serveru provede metoda user manageru `checkPassword`. Sama metoda pro kontrolu hesla nemusí nic kontrolovat, protože využíváme protokol OAuth 2.0 a uživatel se tedy nepřihlašuje lokálně. Na API serveru se ještě provede kontrola, jestli je uživatelský účet aktivován. Metoda pro kontrolu hesla vrací tedy vždy `True`.

```
try:
    redirect_uri = user.get_id()["redirect_uri"]
    code = user.get_id()["code"]
except KeyError:
    OAuthbasedUserManager.logger.error(
        "Code or redirect_uri not found")
    return None

client_id = self.oauth2[redirect_uri]["client_id"]
client_secret = self.oauth2[redirect_uri]["client_secret"]
oauth2_session = OAuth2Session(client_id,
                               redirect_uri=redirect_uri)
access_token = self.get_token(oauth2_session, code,
                              client_id, client_secret)

if access_token is None:
    return None

username = self.get_username(oauth2_session)
if username is None:
    OAuthbasedUserManager.logger.error("Username none")
    return None
user = FilebasedUserManager.findUser(self, username)

if user is None:
    # addUser(username, password, active, roles)
    # password not needed, user will be active, role is user
    self.addUser(username, "", True, ["user"])
    user = self.findUser(username)
```

Ukázka kódu 3.6: Část metody login_user

3.4.3.1 Přihlášení v user manageru

Přihlášení se v user manageru provádí pomocí metody `login_user`, která bere jako parametr uživatele. Na úvod se provádějí kontroly, které jsem převzal od přihlášení v OctoPrintu. Nejprve se v této metodě testuje, zda daný uživatel není prázdný. Dále se kontroluje, zda uživatel již v OctoPrintu existuje, případně je rovnou přihlášen.

Ukázka kódu 3.6 popisuje, jak nová metoda z uživatelova `get_id` vybere potřebnou přesměrovací adresu a kód, který je dále využíván na získání potřebného `client_id` a `client_secret`. Pro správné získání přístupového tokenu je potřeba vytvořit relaci. Ta se vytváří právě pomocí knihovny

requests_oauthlib.

```
def get_token(self, oauth2_session, code, client_id,
              client_secret):
    try:
        token_json = oauth2_session.fetch_token(
            self.path_for_token,
            authorization_response="authorization_code",
            code=code,
            client_id=client_id,
            client_secret=client_secret,
            headers=self.token_headers)

        try:
            # token is OK
            access_token = token_json["access_token"]
            return access_token
        except KeyError:
            try:
                error = token_json["error"]
                OAuthbasedUserManager.logger.error(
                    "Error of access token: %s", error)
            except KeyError:
                OAuthbasedUserManager.logger.error(
                    "Error of access token, "
                    "error message not found")

    except OAuth2Error:
        OAuthbasedUserManager.logger.error(
            "Bad authorization_code")

    return None
```

Ukázka kódu 3.7: Metoda pro získání přístupového tokenu

Přihlášení v `userManager` využívá dvě nové metody. Jsou to `get_token` a `get_username`. Metoda pro vyzvednutí přístupového tokenu bere za parametry relaci, vytvořenou v `login_user`, dále kód, klientský identifikátor a klientův tajný klíč. Pomocí relace metoda vyzvedne přístupový token z autorizačního serveru. Následně provede patřičné kontroly a když je vše v pořádku, tak navrátí přistupový token do `login_user`.

Přihlašovací metoda využije přístupový token a předá ho jako parametr metodě `get_username`. Z konfiguračního souboru je zde použito pojmenování přístupového klíče, na různých serverech se tato hodnota totiž může lišit.

Následně je pomocí knihovny `requests` využito metody `GET` k obdržení dat ze zdrojového serveru pomocí přístupového tokenu. Přihlašovací údaj je pak z dat vybrán, dle hodnoty v konfiguračním souboru `username_key`. Pokud vše dopadne dobře, předá se uživatelské jméno zpět metodě `login_user`.

Teď, když už plugin zná přihlašovací jméno, aplikace zkontroluje, zda už uživatel existuje v konfiguračním souboru pro uživatele `users.yaml`. V případě, že uživatel neexistuje, současná implementace pluginu přihlašovaného zaregistruje s právy uživatele. V případě, že by potřeboval uživatel práva správce, může mu je současný správce přidat přes uživatelské rozhraní OctoPrintu.

Přihlášený uživatel má možnost manipulovat s ostatními prvky OctoPrintu. Správci mohou nastavovat vše, co je v původním OctoPrintu možné, a to včetně práv uživatelů. Plugin má na starost pouze přihlašování uživatelů. Přihlášených uživatelů k jednomu OctoPrintu může být i více.

Kapitola 4

Testování

V této kapitole se budu věnovat testování výsledného pluginu do aplikace OctoPrint. Pro správné testování bylo potřeba vytvořit si vlastní falešný autorizační server a falešný zdrojový server. Dále se budu věnovat jednotkovým testům zaměřujících se na správnou funkčnost metod `OAuthBasedUserManager`. Následně jsou popsány integrační testy, které kontrolují správné začlenění do aplikace OctoPrint.

4.1 Vlastní poskytovatel OAuth 2.0

Pro správné otestování funkčnosti pluginu do aplikace OctoPrint na autorizaci pomocí protokolu OAuth 2.0 bylo potřeba si vytvořit falešné servery, které simulují poskytovatele OAuth 2.0. Falešné autorizační servery jsou dva – jeden umožňuje provést jednotkové testy pro metody user manageru a slouží i jako zdrojový server pro integrační testy, druhý z nich je právě určen pro integrační testy.

4.1.1 Falešný server pro jednotkové testy

Server, který určený pro jednotkové testy, využívá `SocketServer` ze standardní knihovny pro jazyk Python a byl inspirován podle testovacího serveru pro ÚTVS ČVUT [29]. Odposlouchává metody `GET` a `POST` a podle URL adresy vyhodnotí, které metodě předat další funkcionalitu. Pro server bylo vytvořeno několik metod simulující poskytovatele OAuth 2.0.

- Metoda `authorize()` simuluje přihlášení uživatele na server třetí strany. Pokud obdrží správné `CLIENT_ID` ze souboru `constants.py` určených pro testování, vrátí metoda jinou konstantu jménem `GOOD_CODE`.
- Metoda `fake_access_token()` navrátí konstantu `GOOD_ACCESS_TOKEN` v případě obdržení správného kódu. V jiných případech zašle metoda chybovou hlášku.

- Metoda `fake_user_info()` zkontroluje, zda dostane konstantu v podobě správného přístupového tokenu. V takovém případě navrátí konstantu zpět data `GOOD_USERNAME_KEY` a `GOOD_USERNAME`. Klíč uživatele se posílá proto, že testy kontrolují i jestli zdrojový server obsahuje správný klíč uživatele.
- Metoda `fake_resource_server()` simuluje zdrojový server pro integrační testy. Počítají se přístupy do dané metody a podle nich se vyhodnotí, zda falešný zdrojový server vrátí uživatele nebo admina.

4.1.2 Falešný server pro integrační testy

Pro integrační testy bylo zvolené řešení OAuth 2.0 poskytovatele z balíčku `python-oauth2` [30]. Tento modul je pod svobodnou licencí a obsahuje příklady serverů s různými autorizačními granty. Použil jsem tedy server využívající grant autorizační kód.

Protože tento poskytovatel OAuth2 neposkytuje zdrojový server, rozhodl jsem se, že zabuduji falešný zdrojový server do stávajícího serveru pro jednotkové testy. Ten odchytává požadavky se začínající cestou URL `/api/login`. Dále počítá počet přihlášených uživatelů a podle čísla přihlášení rozhodne, zda daný přihlašující bude mít v OctoPrintu roli uživatele nebo admina.

4.2 Jednotkové testy

Tato skupina testů se zaměřuje na funkčnost metody `oauth_user_manager`. Testy jsou prováděny pomocí frameworku pytest [31]. Před spuštěním jednotlivých testů se pomocí fixture [32] vytvoří zmíněný falešný OAuth2 poskytovatel. Dále se pro každý jednotlivý test vytvoří testovací user manager. Tomu se nastaví hodnoty používané v testech a dále nějaké hodnoty potřebné OctoPrintem.

Je důležité tyto hodnoty takto explicitně nastavit, jinak bychom museli nastartovat celý OctoPrint. Abychom otestovali pouze funkčnost metod user managera, tak to není potřeba. Dále pro některé z testů existuje fixture zajišťující OAuth2 relaci pomocí knihovny `requests_oauthlib` [23].

4.2.1 Testy

Jednotkové testy kontrolují:

- Obdržení správného přístupového tokenu při převzetí správného autorizačního kódu,
- kontrolu, že přístupový kód neexistuje, když byl zaslán špatný kód,
- obdržení správného uživatelského jména při převzetí správného přístupového tokenu,

- zda uživatelské jméno nebylo poskytnuto, když byl zaslán špatný přístupový token,
- správnost klíče pro autorizační token,
- přihlášení uživatele, pokud už pro něj existuje relace s OctoPrintem,
- přihlášení uživatele pomocí metody `login_user`.

4.2.2 Průběh testů

Všechny testy probíhají podobně. Před jejich spuštěním se nastartuje pomocný server a nainicializují se potřebné proměnné. Každý test otestuje metodu nového user manageru a zkontroluje, jestli jsou navrácené hodnoty správné nebo špatné – to záleží na typu testu. Některé testy jsou založené na pozitivní odezvě od user manageru. Další testy kontrolují odchycení výjimek při zadání špatných hodnot.

Po vývoji testů byla ještě upravena struktura user managera. To samozřejmě ovlivnilo i testy a ty musely být taky upravené. Jednotkové testy mi několikrát během mé implementace pluginu pro autorizaci pomocí protokolu OAuth 2.0 usnadnily práci při objevování chyb.

Jednotkové testy odhalily chybu, kdy user manager odchytával špatný typ výjimky. Díky tomu jsem pak mohl chybu jednoduše opravit.

4.3 Integrační testy

Integrační testy jsou vytvořené pomocí Selenium [33], což je nástroj pro automatické testování webových aplikací. Pomocí Selenia se vyhledávají elementy na webové stránce. Pokud je nalezne, může s nimi dále manipulovat a tím simulovat uživatele, který je na webové stránce a provádí určité operace. Testy jsou stejně jako jednotkové spouštěny pomocí frameworku pytest.

4.3.1 Před spuštěním testů

Pro testy využívající Selenium je potřeba v terminálu, před spuštěním OctoPrintu, dočasně vyexportovat proměnnou z knihovny oauthlib pomocí příkazu v ukázce kódu 4.8.

```
export OAUTHLIB_INSECURE_TRANSPORT=1
```

Ukázka kódu 4.8: Nastavení proměnné z knihovny oauthlib

Protokol OAuth 2.0 využívá totiž zabezpečené připojení. Následně je potřeba pro spuštění integračních testů spustit samotnou aplikaci OctoPrint na adrese `http://0.0.0.0:5000/` se správným konfigurační souborem v sekci

plugins, jak je uvedeno v ukázce kódu 4.9. Dále je také potřeba nastavit konfigurační soubor pro uživatele `users.yaml`, který je uveden v ukázce kódu 4.10.

```
plugins:
  oauth2:
    login_path: http://localhost:8282/authorize
    token_path: http://localhost:8282/token
    user_info_path: http://localhost:8080/api/login
    username_key: username
    access_token_query_key: access_token
    token_headers:
      Accept: application/json
    http://0.0.0.0:5000/:
      client_id: abc
      client_secret: xyz
```

Ukázka kódu 4.9: Konfigurace pro integrační testy

Pokud jsou v konfiguračním souboru pro uživatele ještě nějací další uživatelé, tak se tam mohou ponechat. Hodnota, která tam ale nesmí být, je uživatel `test_user` s právy admina. Pokud ale konfiguračním souboru je, tak musí mít pouze práva uživatele.

```
test_admin:
  active: true
  apikey: null
  password: ''
  roles:
  - user
  - admin
  settings: {}
```

Ukázka kódu 4.10: Konfigurace uživatelů pro integrační testy

4.3.2 Testy

V současné době existují čtyři testy vytvořené v Selenium:

- První test se přihlásí přes autorizační server a ze zdrojového serveru získá údaje pro uživatele `test_admin`. Test pak zkontroluje, jestli je uživatel opravdu přihlášen do webového rozhraní aplikace OctoPrint.

- Druhý test nejprve přihlásí uživatele `test_admin` a následně ho zkusí z webového rozhraní OctoPrintu odhlásit.

- Třetí test je nejkomplikovanější. Pomocí Selenium se otevřou dvě okna, na kterých se otevře stránka swebovým rozhraním OctoPrintu. První z nich je přihlášení s rolí admina a druhý jako pouhý uživatel.

  Test provede to, že uživatel `test_admin` přidá roli admina uživateli `test_user`. Ten následně obnoví stránku s OctoPrintem a zkontroluje, jestli má opravdu práva pro admina.

  Na závěr test_admin znovu provede změnu práv pro tohoto uživatele a `test_user` ověří, jestli má opět pouze uživatelská práva.

- Čtvrtý test zkontroluje, pokud uživatel na autorizačním serveru zaškrtne, že se nechce autorizovat do aplikace OctoPrint pomocí protokolu OAuth 2.0, tak nebude přihlášen ani ve webovém rozhraní OctoPrintu.

4.3.3 Průběh testů

Přihlášení uživatele proběhlo v pořádku. U druhého testu jsem narazil na komplikaci – OctoPrint totiž po přihlášení uživatele dává na stránku notifikace o OctoPrintu. Například tedy aby uživatel používal novou verzi, nebo že existuje nový plugin do aplikace. Kvůli těmto notifikacím bylo zastíněno tlačítko pro odhlášení a Selenium ohlásilo chybu, že daný prvek na stránce nemůže najít.

Tuto záležitost jsem vyřešil tak, že testy vyhledají elementy pro notifikace a zaškrtne na nich, aby byly ignorovány. Tímto notifikace zmizí a test odhlášení uživatele proběhne v pořádku.

Největší výzvou ale bylo přidávání a odebírání práv pro jistého uživatele. První se přihlásí `test_admin`, druhý se přihlásí testovací uživatel. V tento moment, pokud jeho záznam neexistoval v konfiguračním souboru, tak se pro něj vytvoří záznam. Testovací admin musí tedy znovu načíst stránku, aby tato data od OctoPrintu obdržel.

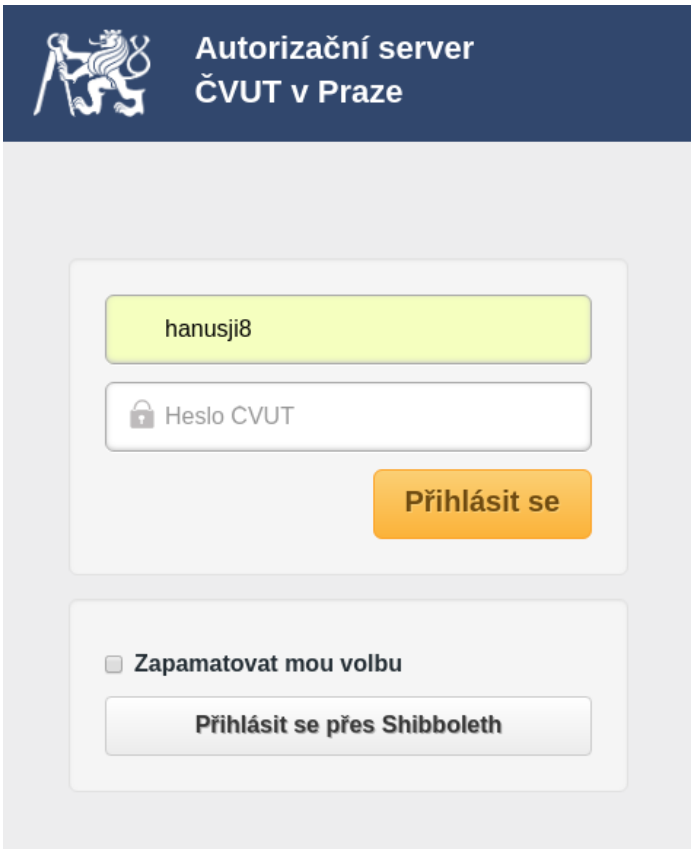
Následuje kliknutí na nastavení obecné, poté pro uživatele. Test vyhledá uživatele se jménem `test_user` a najde pro něho prvek s upravením práv. nalezení těchto hodnot pomocí Selenium je celkem sofistikované. Jakmile je možnost uživateli změnit práva, tak se změní a nastavení se uloží.

Může se to zdát jako trivialita, ale nalezení tlačítka pro uložený nastavení je velmi obtížné. Takových tlačítek na stránce OctoPrintu existuje několik. Tlačítko, které je potřeba zaškrtnout, musí být viditelné a to se v Selenium nedá jen tak jednoznačně udělat.

Všechny testy probíhaí v pořádku a to všetně posledního, kdy uživatel zadal, že nechce být přihlášen.

4.4 Otestování u třetí strany

Součástí mé práce bylo také otestovat práci na více než jednom autorizačním serveru třetí strany. Rozhodl jsem se tedy, že využiji autorizační server na ČVUT v Praze a GitHub. Pro každý z těchto serverů jsem tedy vytvořil konfigurační soubory `config.yaml` a `users.yaml`. Dále bylo potřeba zaregistrovat si klientské aplikace u těchto serverů a využít údaje z nich pro autorizaci.



Obrázek 4.1: Přihlášení na autorizačním serveru FIT ČVUT

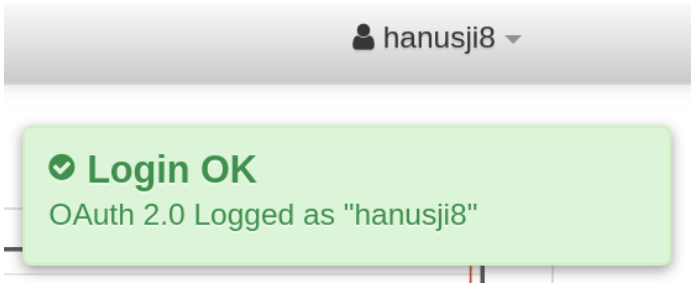
4.4.1 FIT ČVUT v Praze

První jsem vyzkoušel autentizační server na ČVUT v Praze. Využil jsem informací sepsaných kolegou Jakubem Jirůtkou ohledně frameworku OAuth2 [20]. Také jsou zde přehledně vypsané adresy, kde autorizovat uživatele a kde získat přístupový kód.

Následně jsem si zaregistroval klientskou aplikaci na serveru FIT ČVUT pro správu OAuth2 aplikací. Tam mi byly poskytnuty údaje jako je klientský identifikátor a klientský tajný klíč. Na oplátku jsem já uložil na server svoji

přesměrovací adresu, na které lokálně pouštím aplikaci OctoPrint pro testovací účely.

Po startu OctoPrintu s instancí pluginu pro autorizaci přes OAuth2 jsem se tedy přihlásil přes autorizační server ČVUT v Praze, to můžete vidět na obrázku 4.1. Server se ještě zeptal, jaké informace může ze zdrojového serveru poskytnout. Poté jsem byl přesměrován zpět do OctoPrintu, kde na pozadí proběhla správná autorizace pomocí protokolu OAuth 2.0 a následně jsem byl přihlášen do aplikace OctoPrint přes své uživatelské údaje, jak je vidět na obrázku 4.2, kde mám pouze práva uživatele.



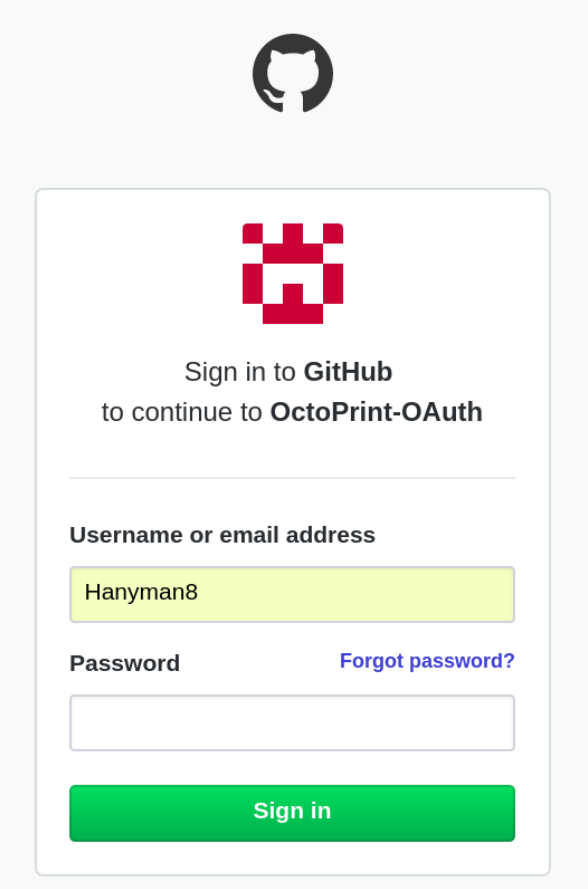
Obrázek 4.2: Notifikace po přihlášení přes autorizační server FIT ČVUT

4.4.2 GitHub

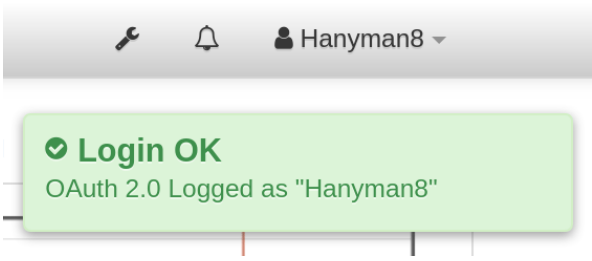
Druhým autorizačním serverem byl GitHub. Postup byl velmi podobný jako u autorizace na ČVUT serveru. Konfigurace se však lišila. Pro získání uživatelského jména se na zdrojovém serveru GitHubu používá klíč `username` narozdíl od `user_id` na ČVUT v Praze. Další zajímavou změnou bylo získání tokenu. Pro GitHub se používá pojmenování `access_token` a pro server na FIT ČVUT pouze `token`.

Autorizace však probíhala podobně. Byl jsem přesměrován na možnost přihlášení jak je vidět na obrázku 4.3. Poté jsem byl přesměrován zpět se správným autorizačním kódem a stavem. Ty se využili pro získání přístupového tokenu a díky němu bylo zjištěno uživatelské jméno, které bylo použito pro přihlášení. Na žádné větší problémy jsem nenarazil.

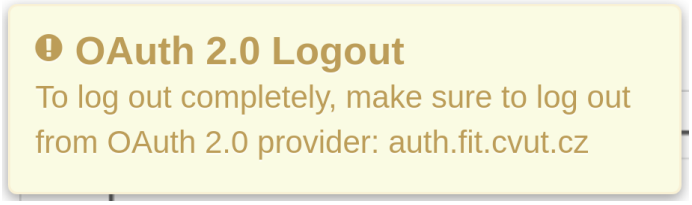
Na obrázcích 4.5 můžete vidět, že plugin do aplikace OctoPrint zjistí přes jaký autorizační server se uživatel přihlásil a podle toho po odhlášení vypíše náležité upozornění.



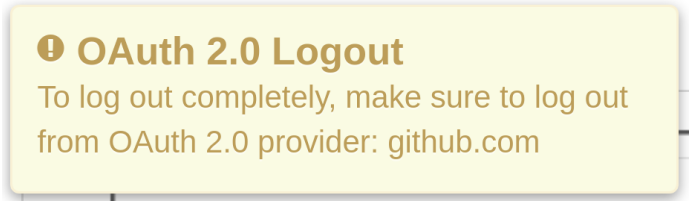
Obrázek 4.3: Přihlášení na autorizačním serveru GitHub



Obrázek 4.4: Notifikace po přihlášení přes GitHub



(a) Notifikace po odhlášení po autorizaci na FIT ČVUT



(b) Notifikace po odhlášení po autorizaci na GitHub

Obrázek 4.5: Ukázky notifikace po odhlášení z OctoPrintu

Kapitola 5

Nasazení

V rámci mé bakalářské práce bylo mým úkolem nasadit instanci mého pluginu na Raspberry Pi z laboratoře 3D tisku na FIT ČVUT v Praze. Tato instance pluginu musí používat autorizační server ČVUT. S vedoucím mé bakalářské práce jsem se domluvil, že nasadím instanci pluginu na čtyři Raspberry Pi.

Nejprve jsem se to rozhodl nasadit pouze na jedno testovací Raspberry Pi. Zde jsem narazil na problém, že toto Raspberry Pi využívalo starou verzi OctoPrintu a také starou verzi OctoPi. OctoPi je operační systém založení na Raspianu.[1] Tyto verze ještě nepodporovaly správnou instalaci pluginů do OctoPrintu a proto jsem dané Raspberry Pi přeinstalovat na správnou verzi OctoPrintu.

5.1 Instalace

Po přeinstalování na správnou verzi OctoPrintu byla instalace pluginu jednoduchá. Ve webovém rozhraní OctoPrintu jsem přidal uživatele s mým přihlašovacím jménem. Při prvním nasazením jsem použil konfiguraci pro autorizační server GitHubu. Na GitHubu jsem si ještě upravil přesměrovací adresu na dané Raspberry Pi – `printer15.local`.

Do OctoPrintu jsem poté přes uživatelské rozhraní přidal uživatele s mým přihlašovacím jménem na GitHub a právy admina. To jsem udělal proto, abych nemusel upravovat konfiguraci uživatelů v `users.yaml` ručně. Dále jsem ručně přes čtečku SD karet přidal do konfiguračního souboru správné nastavení pro plugin. Ukázku konfigurace pro GitHub naleznete na ukázce kódu 2.3.

Po novém startu OctoPrintu jsem přes uživatelské rozhraní nainstaloval plugin. Instalace pluginů v OctoPrintu není obtížná, protože stačí zkopírovat adresu balíčku pro plugin a ten se poté v uživatelském rozhraní nainstaluje pomocí utility pip.

Po úspěšné instalaci je znovu vyžadován restart aplikace OctoPrint, a jakmile se tak stalo, bylo v uživatelském rozhraní možno vidět, že plugin

byl nainstalován správně. Zbylo tedy akorát plugin vyzkoušet přihlášením. Přihlásil jsem se tedy přes autorizační server GitHub a byl jsem do rozhraní OctoPrintu přihlášen svým uživatelským jménem.

Vyzkoušet plugin měl možnost i bývalý vedoucí 3D laboratoře Ing. Marek Žehra, který si plugin chválí.

5.2 Nasazení na Raspberry Pi

S vedoucím mé bakalářské práce jsme se dohodli, že nasadím plugin na čtyři 3D tiskárny. Některá Raspberry Pi v sobě opět obsahovala starší verze OctoPrintu, dvě z nich obsahovali verzi 1.3.2, která by už měla přidávání pluginů podporovat.

Pluginy byly otestované na verzi 1.3.8, tak jsem se rozhodl, že OctoPrint v Raspberry Pi přeinstaluji na novější verzi. U dvou tiskáren to šlo přímo z uživatelského rozhraní, ale u dalších dvou se musela nová verze OctoPrintu přidat ručně. Pomocí SSH jsem se tedy přihlásil k jednotlivým tiskárnám a manuálně tiskárny převedl na novější verzi.

Nejprve jsem připravil konfigurační soubory pro jednotlivé 3D tiskárny. Pro každou z tiskáren v tento moment existují konfigurace dvě – pro přihlašování v 3D laboratoři a pro učebnu 3D tisku. Konfigurační soubory upravoval pomocí SSH. Příklad konfigurace pluginu pro autorizaci přes OAuth 2.0 naleznete v ukázce kódu 5.11.

```
plugins:
login_path: https://auth.fit.cvut.cz/oauth/authorize
    token_path: https://auth.fit.cvut.cz/oauth/token
    user_info_path: https://auth.fit.cvut.cz/oauth/api/v1/tokeninfo
    username_key: user_id
    access_token_query_key: token
    token_headers:
      Accept: application/json
    http://0.0.0.0:5000/:
      client_id: 12345678-1234-5665-4321-87654321098
      client_secret: MHIASNTYRBHYALNZUDSE
```

Ukázka kódu 5.11: Příklad konfiguračního souboru pro FIT ČVUT

Na první dvě tiskárny byla instalace bezproblémová. U druhých dvou zabralo hodně času vylepšování OctoPrintu na novější verzi. Žádné větší problémy při nasazování pluginu nebyly a proto považuji plugin do aplikace OctoPrint, jehož pomocí se uživatel do webového rozhraní aplikace může přihlásit přes protokol OAuth 2.0, za hotový.

Závěr

Cílem mé bakalářské práce bylo vytvořit rozšíření pro aplikaci OctoPrint, díky kterému se uživatelé budou moci přihlašovat do webového rozhraní OctoPrintu pomocí protokolu OAuth 2.0.

V rámci práce byla vytvořena analýza přidávání pluginů do aplikace OctoPrint, dále je prozkoumán autorizační protokol OAuth 2.0. Následně je na základě této analýzy vytvořen návrh, jak má výsledné rozšíření pracovat a s jakými komponentami OctoPrintu bude manipulovat. Rozšíření do aplikace OctoPrint má být snadno nastavitelné pro různé autorizační servery, proto je také vytvořen návrh a forma ukládání těchto dat.

Po vytvoření návrhu byl tento plugin pro aplikaci OctoPrint implementován a výsledek byl zdokumentován. Pomocí jednotkových a integračních testů bylo ověřeno, že výsledný plugin splňuje vytyčené požadavky. Pro vytvořené rozšíření byl vytvořen manuál v anglickém jazyce, který popisuje instalaci a konfiguraci pro autorizační servery. Manuál, který je taktéž obsažen na přiloženém CD, obsahuje příklad konfigurace pro autorizační server GitHub.

Součástí mé bakalářské práce bylo také potřebné nasadit instanci pluginu na Raspberry Pi u 3D tiskáren využívajících se na předmět BI-3DT na FIT ČVUT v Praze. To bylo uskutečněno a tím došlo ke splněním všech cílů této bakalářské práce.

Možná rozšíření

Další rozšíření pluginu jsou možná, protože je to práce vytvořená pod svobodnou licencí AGPLv3 a já věřím, že tato práce bude využívaná nejen pro předmět 3D tisk na Fakultě informačních technologií, ale i v komunitě využívající aplikaci OctoPrint. Jednotlivá rozšíření jsou:

- Zkombinovat původní přihlašování, které využívá ukládání uživatelů do souboru `users.yaml` s přihlašováním přes protokol OAuth 2.0. To by poskytlo možnost přihlásit se do rozhraní OctoPrintu, kdyby nastal výpadek internetového připojení.
- Poskytnout uživatelům možnost nastavení konfiguračního souboru přímo z webového rozhraní apliakce OctoPrint. Konfigurační soubor se pro plugin musí nastavovat ručně.
- Vylepšit přidávání práv nově zaregistrovaným uživatelům. Šlo by to udělat například pomocí OAuth 2.0 scopes.
- Zaznamenávat činnosti přihlášených uživatelů do logů a efektivně v nich vyhledávat, co který uživatel dělal za operace.

Literatura

[1] HÄUßGE, G. *OctoPrint*. [online], 2013, [cit. 2018-05-12]. Available from: https://octoprint.org/

[2] Vydavatelství Nová média, s. r. o. *3D-tisk*. [online], 2012, [cit. 2018-05-13]. Available from: https://www.3d-tisk.cz/

[3] 3D Hubs. *What is 3D Printing?* [online], 2018, [cit. 2018-05-13]. Available from: https://www.3dhubs.com/what-is-3d-printing/

[4] BOWYER, A. *RepRap*. [online], 2004, [cit. 2018-05-12]. Available from: http://reprap.org/

[5] HANSON, M. *The 10 best 3D printers of 2018*. [online], 2018, [cit. 2018-05-12]. Available from: https://www.techradar.com/news/best-3d-printer

[6] Vydavatelství Nová média. *Nejpopulárnější česká 3D tiskárna má ambiciózního nástupce – Prusa i3 MK3*. [online], 2017, [cit. 2018-05-13]. Available from: https://www.3d-tisk.cz/nejpopularnejsi-ceska-3d-tiskarna-ma-ambicioziho-nastupce-prusa-i3-mk3//

[7] Materialpro3d.cz. *ROZDÍL MEZI ABS, PLA, PETG*. [online], 2017, [cit. 2018-05-12]. Available from: https://www.materialpro3d.cz/blog/rozdily-abs-pla-petg/

[8] All3DP. *20 Best 3D Printing Software Tools*. [online], 2018, [cit. 2018-05-12].

[9] knockoutjs.com. *Knockout*. [online], [cit. 2018-05-12]. Available from: http://knockoutjs.com/documentation/introduction.html

[10] RONACHER, A. *Jinja2*. [online], 2008, [cit. 2018-05-12]. Available from: http://jinja.pocoo.org/docs/2.10/

[11] EVANS, C. *YAML: YAML Ain't Markup Language*. [online], 2001, [cit. 2018-05-13]. Available from: http://yaml.org/

[12] HÄUßGE, G. *OctoPrint documentation*. [online], 2013, [cit. 2018-05-12]. Available from: http://docs.octoprint.org/en/master/

[13] HÄUßGE, G. *Mixins*. [online], 2013, [cit. 2018-05-13]. Available from: http://docs.octoprint.org/en/master/plugins/mixins.html

[14] HÄUßGE, G. *Hooks*. [online], 2013, [cit. 2018-05-13]. Available from: http://docs.octoprint.org/en/master/plugins/hooks.html

[15] HARDT, D. a. *The OAuth 2.0 Authorization Framework*. [online], 2012, [cit. 2018-05-12]. Available from: https://tools.ietf.org/html/rfc6749

[16] HAMMER-LAHAV, E. *The OAuth 1.0 Protocol*. [online], 2010, [cit. 2018-05-12]. Available from: https://tools.ietf.org/html/rfc5849

[17] *JONES, M.; HARDT, D. The OAuth 2.0 Authorization Framework: Bearer Token Usage*. [online], 2012, [cit. 2018-05-12]. Available from: https://tools.ietf.org/html/rfc6750

[18] HAMMER-LAHAV, E. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. [online], 2012, [cit. 2018-05-12]. Available from: https://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-01

[19] REITZ, K. *Requests-OAuthlib OAuth Examples*. [online], 2014, [cit. 2018-05-12]. Available from: http://requests-oauthlib.readthedocs.io/en/latest/examples/examples.html

[20] JIRŮTKA, J. *OAuth 2.0*. [online], 2014, [cit. 2018-05-12]. Available from: https://rozvoj.fit.cvut.cz/Main/oauth2

[21] JIRŮTKA, J.; a další. *KOSapi*. [online], 2011, [cit. 2018-05-12]. Available from: https://kosapi.fit.cvut.cz/projects/kosapi/wiki

[22] HRONČOK, M. *730ne*. [online], 2015, [cit. 2018-05-12]. Available from: https://github.com/hroncok/730ne

[23] REITZ, K. *Requests-OAuthlib: OAuth for Humans*. [online], 2014, [cit. 2018-05-12]. Available from: http://requests-oauthlib.readthedocs.io/en/latest/index.html

[24] REITZ, K. *Requests-OAuthlib OAuth 2 Workflow*. [online], 2014, [cit. 2018-05-12]. Available from: http://requests-oauthlib.readthedocs.io/en/latest/oauth2_workflow.html

[25] Raspberry Pi Foundation. *Raspberry Pi*. [online], 2012, [cit. 2018-05-12]. Available from: https://www.raspberrypi.org/

[26] REITZ, K.; a další. *Requests: HTTP for Humans*. [online], 2011, [cit. 2018-05-12]. Available from: http://requests-oauthlib.readthedocs.io/en/latest/index.html

[27] Python Software Foundation. *SocketServer — A framework for network servers*. [online], 1990, [cit. 2018-05-12]. Available from: https://docs.python.org/2/library/socketserver.html

[28] Python Software Foundation. *BaseHTTPServer — Basic HTTP server*. [online], 1990, [cit. 2018-05-12]. Available from: https://docs.python.org/2/library/basehttpserver.html

[29] HRONČOK, M. *utvsapitoken – fakeserver*. [online], 2016, [cit. 2018-05-12]. Available from: https://github.com/hroncok/utvsapitoken/blob/master/utvsapitoken/fakeserver.py

[30] MEYER, M. *python-oauth2*. [online], 2015, [cit. 2018-05-12]. Available from: https://github.com/wndhydrnt/python-oauth2

[31] KREKEL, H.; pytest-dev team. *pytest*. [online], 2004, [cit. 2018-05-12]. Available from: https://github.com/wndhydrnt/python-oauth2

[32] KREKEL, H.; pytest-dev team. *pytest fixtures: explicit, modular, scalable*. [online], 2015, [cit. 2018-05-14]. Available from: https://docs.pytest.org/en/latest/fixture.html

[33] HUGGINS, J. *Selenium*. [online], 2004, [cit. 2018-05-12]. Available from: https://www.seleniumhq.org/

Příloha A

Seznam použitých zkratek

ABS Akrylonitrilbutadienstyren

AGPL Affero General Public License

API Application Programming Interface

CSS Cascading Style Sheets

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

PET-G Polyethylentereftalát glykol

PLA Kyselina polymléčná

RepRap Replicating rapid prototyper

REST Representational State Transfer

SLA Stereolitografie

SSH Secure Shell

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

YAML YAML Ain't Markup Language

Příloha B

Obsah přiloženého CD

- `README.txt` ..... stručný popis obsahu CD
- `src` ..... adresář se zdrojovými kódy pluginu
  - `README.md` ..... anglická dokumentace pluginu
  - `octoprint_oauth2` ..... adresář s implementací pluginu
    - `tests` ..... adresář se zdrojovými soubory testů
- `text`
  - `thesis` ..... adresář se zdrojovou formou ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF

Obsah práce je také dostupný z:

- `https://github.com/Hanyman8/OctoPrint-OAuth2`
- `https://gitlab.fit.cvut.cz/hanusji8/BP_Hanus_Jiri_2018`