ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Hromadná správa aplikací ovládajících 3D tiskárny
Student:	Jiří Makarius
Vedoucí:	Ing. Miroslav Hronček
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Implementujte webovou aplikaci umožňující hromadnou správu několika instancí aplikace OctoPrint ovládajících 3D tiskárny. Instance mohou být dostupné v lokální síti nebo v síti internet. K řešení použijte API poskytované aplikací OctoPrint, které v případě potřeby rozšiřte. Aplikace mimo jiné umožní:
- Přidávání a odebírání instancí aplikace OctoPrint (jednotlivě i hromadně).
- Podporu skupin instancí, přepínání mezi nimi, správu příslušnosti ke skupinám.
- Hromadné (prováděné nad podmnožinou spravovaných instancí) i jednotlivé akce:
  - nastavování teplot,
  - spouštění tisku,
  - nahrávání souborů,
  - změna nastavení.
- Monitoring.
- Přenášení GCODE souborů mezi jednotlivými instancemi.
- Přihlašování pomocí univerzitních účtů s možností specifikace oprávněných osob.

Seznam odborné literatury

[1] HÄUSSGE, Gina. *OctoPrint 1.2.18+0.g1d0657a.dirty documentation: REST API* [online]. 2016 [cit. 2016-12-06]. Dostupné z: http://docs.octoprint.org/en/1.2.18/api/

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 18. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Hromadná správa aplikací ovládajících 3D tiskárny

Jiří Makarius

Vedoucí práce: Ing. Miroslav Hrončok

26. června 2017

Poděkování

Rád bych poděkoval Ing. Miroslavovi Hrončokovi za cenné rady, vedení této práce a korekturu nedokonalostí jak obsahových tak gramatických. Chtěl bych také poděkovat všem, kteří se podíleli na testování této aplikace a pomohli mi ji vytvořit uživatelsky přístupnější.

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.

V Praze dne 26. června 2017 .....

Odkaz na tuto práci

Makarius, Jiří. *Hromadná správa aplikací ovládajících 3D tiskárny.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem této bakalářské práce je vytvořit webovou aplikaci pro ovládání více 3D tiskáren používajících OctoPrint. Zvolené řešení je server napsaný v jazyce Python ve webovém frameworku Flask a uživatelské rozhraní napsané ve frameworku AngularJS. Výsledkem je aplikace, která propojuje přihlášené uživatele a tiskárny s možností omezení přístupu administrátory. Součástí práce je automatická detekce instancí OctoPrint v lokální síti pomocí mDNS. V práci je vysvětleno, proč je použité REST API nevhodné pro monitorování tiskáren.

Klíčová slova OctoPrint, 3D tisk, Python, webová aplikace, uživatelské rozhraní, Angular, Flask, OAuth, zeroconf

Abstract

The purpose of this bachelor thesis is to create a web application to control multiple 3D printers using OctoPrint. The chosen solution is server written in Python Flask framework and user interface in Angular framework. The result is an application that connects users and printers with access control for administrators. Aplication is also able to automatically detect OctoPrint instances in local network by mDNS. Part of this thesis explains why is REST API inappropriate for realtime monitoring.

Keywords OctoPrint, 3D printing, Python, web application, user interface, Angular, Flask, OAuth, zeroconf

Obsah

Seznam obrázků

Seznam ukázek kódu

Úvod

OctoPrint je software pro ovládání 3D tiskáren [1]. Nejedná se o řídící ovladače pro tiskárnu, ale o uživatelské rozhraní. Tento software je stavěný, aby byl v provozu na samostatném Raspberry Pi, což je malý nevýkonný počítač, připojený k jedné tiskárně. Tento software poskytuje přístup do uživatelského rozhraní po síti či internetu. Skrze tento systém je možné dávat tiskárně příkazy, ať se jedná o pohyby tiskárny nebo nahrání G-code pro vytisknutí modelu.

Nevýhoda tohoto systému je, že se předpokládá pro každou tiskárnu v provozu samostatně běžící OctoPrint, což velice může komplikovat práci na více tiskárnách najednou, neboť je potřeba pro každou tiskárnu pracovat s jiným oknem v prohlížeči a to může práci znepřehlednit. [2]

OctoPrint se sám o sobě neumí připojovat do databáze a neexistuje možnost přihlášení univerzitním systémem ČVUT, což také zamezuje přidávání a odebírání práv pro ovládání tiskáren jednotlivými studenty [3]. Proto je cílem této bakalářské práce vytvořit systém, který bude jednotlivé tiskárny spojovat a zároveň do něj bude přístup školním přihlašovacím systémem.

Systém bude umožňovat správu tiskáren jednotlivé i hromadně. Každou tiskárnu bude možné v reálném čase monitorovat a přenášet G-code modelů na jiné tiskárny. Do systému se budou moci jednotlivé instance OctoPrint přidávat i odebírat. Administrátor bude moci jednotlivým uživatelům přiřazovat nebo odebírat práva na ovládání jednotlivých tiskáren či skupin tiskáren.

Motivace k výběru tohoto tématu byla převážně možnost vytvořit webovou aplikaci, která bude použita pro výuku 3D tisku na FIT ČVUT a pro potřeby laboratoře 3D tisku na FIT ČVUT. Díky tomu by také mohla být dále vyvíjena nejen pro univerzitní potřeby, ale taky s lehkými úpravami pro jakéhokoliv příznivce OctoPrintu.

Nejdříve je v práci uveden úvod do problematiky 3D tisku a OctoPrintu, další kapitolou jsou cíle práce, kde je stanoveno, čeho by měla práce dosáhnout. V kapitole Analýza a návrh je uveden průzkum potřebných metod a technologií pro pochopení a vývoj aplikace. Realizace obsahuje popis struktury, metod a problémů, které byly v aplikaci nalezeny během vývoje.

Kapitola 1

Úvod do problematiky

Jelikož výsledný software má být nadstavbou pro již existující software, nejprve je třeba objasnit základní pojmy týkající se 3D tisku a OctoPrintu.

1.1 3D tisk

3D tisk je způsob výroby, který využívá takzvané aditivní metody. Tato metoda spočívá v nanášení vrstvy po vrstvě a výsledkem je cílový model. Výrobky vytvořené touto metodou jsou geometricky přesné předloze, díky počítačem přepočítaným pohybům. Na rozdíl od tradičních metod výroby, kde jsou výrobky vytvářeny odstraňováním materiálu a kde dochází k výrobě až 95 % odpadu počátečního materiálu, při aditivní výrobě je přebytečný materiál minimalizován a tedy je minimalizována i cena na výrobu produktu. [4]

1.2 Fused Deposition Modeling

Nejpoužívanější metodou 3D tisku je Fused Deposition Modeling (FDM) [5]. FDM tiskárny používají termoplastický materiál, který zahřejí na tavný bod a poté vytvářejí objekt vrstvu po vrstvě od základu. Materiál je v podobě vlákna zaveden do trysky, kde se taví a je vytlačován na desku tiskárny. Tryska i deska tiskárny jsou ovládané počítačem pomoci X, Y, Z souřadnic. V mnoha případech je potřeba, aby objekt měl podpory. Jedná se o dočasné objekty připojené k výrobku, aby mohly být vytištěny části výrobku, které nejsou stabilní nebo jsou umístěny nad zemí v prostoru.

Tento druh tisku je používán pro výrobu prototypů, jelikož je možné vytisknout i malé a detailní části s přesností. Nevýhodou pro masovou výrobu je délka tisku, oproti tomu je výhodou redukce odpadního materiálu při tisku.

1.3 RepRap

RepRap je dle [6] projekt pro vytváření, replikaci a distribuci 3D tiskáren používajících FDM metodu tisku. Hlavní specifikací této tiskárny je schopnost replikovat některé své části a zbytek jejích částí, je navržen tak, aby byl jednoduše k dostání. Jelikož RepRap je volně šiřitelný projekt, může kdokoliv udělat jakýkoliv počet kopií tiskárny za použití jiné 3D tiskárny.

1.4 Proces 3D tisku

Dříve než je možné něco vytisknout, je třeba vytvořit počítačový 3D model. Tento model se nejčastěji ukládá ve formátu Stereolithography (STL), kde je uložen jako soustava koordinátů trojúhelníků, které tvoří dohromady model. Tento model je poté za pomoci specializovaného softwaru převeden na G-code. G-code je textový formát, kde jednotlivé řádky jsou příkazy pro tiskárnu, jedná se o procedurální programovací jazyk [7]. Jednotlivé příkazy jsou specifické pro firmware tiskárny, jedná se například o pohyby, vytlačování materiálu, rychlost. Do tiskárny je nahrán tento G-code soubor, který tiskárně dodá příkazy pro zhotovení objektu.

1.5 OctoPrint

OctoPrint je software s webovým uživatelským rozhraním pro ovládání 3D tiskárny. Je vydaný jako volně šiřitelný program pod licencí GNU Affero General Public License (AGPL). OctoPrint je kompatibilní s většinou uživatelům přístupných tiskáren. [1]

OctoPrint je vyroben převážně pro Raspberry Pi, což je minimalizovaný nevýkonný počítač. Na oficiální stránce [1] je připraveno vydání přímo jako součást operačního systému nazývané OctoPi. Toto vydání je na bázi Raspbianu, což je operační systém pro Raspberry Pi a obsahuje službu OctoPrint [8]. Toto Raspberry Pi je připojené k tiskárně a k síti, ne které na portu 80 hostuje webové uživatelské rozhraní. OctoPrint lze také použít na jiném zařízení než Raspberry Pi, ale každý uživatel si ho musí stáhnout, nainstalovat a provést všechna nastavení sám.

Tento software umožňuje skrze své uživatelské rozhraní:

- Spouštět tisk z G-code souboru a pracovat se soubory v uložišti.
- Monitorovat stav tiskárny, zda-li tiskne, je pozastavena, je připojena.
- Sledovat průběh tisku zabudovanou webkamerou, je-li přítomna.
- Sledovat teploty tiskárny a nastavovat je pro předehřátí před tiskem.

- Dávat tiskárně přímo příkazy ve formě G-code nebo pomocí ovládacích prvků.
- Zobrazit zbývající dobu tisku a procentuální zhotovení, pozastavovat, pokračovat a rušit tisk.
- Vizualizaci G-code souboru ve formě grafického znázornění pohybů.

Dále umí převést STL soubor přímo na G-code a základní přihlášení uživatele jménem a heslem se specifikací oprávnění.

OctoPrint má systém, který dává uživatelům možnost rozšířit jeho funkcionalitu připravenými balíčky. Tyto balíčky jsou jednoduše instalovatelné do základního OctoPrintu. Nejznámější příklady jsou: automatické posílání emailů o procesu tisku, integrace s komunikačním softwarem jako je Slack nebo podpora pro nestandardní typy tiskáren. [1]

Největší nevýhodou OctoPrintu je možnost připojení pouze k jedné tiskárně. Pokud chceme pracovat s více tiskárnami najednou, musíme mít více instancí OctoPrintu na jednom zařízení nebo více zařízení s OctoPrintem a tudíž je nutné pracovat s více okny v prohlížeči. [2]

Kapitola 2

Cíl práce

Cílem této práce je vytvořit webovou aplikaci pro správu více instancí aplikace OctoPrint s permisovní licencí MIT, aby tuto aplikaci mohl kdokoliv používat a dále rozvíjet. Součástí práce bude návod na spuštění serveru. Aplikace bude nenáročná na hardware a bude jednoduchá na spuštění. Z hlediska požadavků na funkčnost je nutné, aby aplikace umožnila:

- Přidávání instancí OctoPrint hromadně i jednotlivě jak v lokální síti tak na Internetu.
- Podporu skupin tiskáren, přepínání mezi nimi a správu příslušnosti ke skupině.
- Monitorování teplot a průběhu tisku.
- Posílání G-code souborů na jiné tiskárny.
- Přihlášení za použití univerzitního účtu se specifikací oprávnění osoby.
- Hromadné i jednotlivé nastavování teplot, spouštění tisku, nahrávání souborů a změny nastavení.

Bude vytvořena klientská a serverová část aplikace (tedy frontend a backend), kde uživatel bude moci po přihlášení ovládat tiskárny, ke kterým má práva. Jedná se pouze o monitorování a ovládání tisku. Uživatel aplikace s právy administrátora bude moci tiskárny i skupina nastavovat a dávat běžným uživatelům k tiskárnám přístup.

Důležitým rozhodnutím bude zvolení správného rozhraní pro komunikaci serveru s klientem a instancemi tiskáren, což je jedním z výzkumných hledisek, které jsem si v této práci zvolil. Dále bude prozkoumáno, zda-li je možnost automaticky detekovat tiskárny v lokální síti.

Kapitola 3

Analýza a návrh

V této části budou zkoumány stavební prvky pro vytvoření cílové aplikace. Jednotlivé sekce se týkají analýzy možných problémů a doporučení postupů, které je vhodné za daných okolností použít.

3.1 Komunikační rozhraní

Navrhovaný server bude působit jako propojovací článek mezi tiskárnami a uživateli. Aby se toho dalo docílit, je třeba zajistit komunikační rozhraní mezi serverem a uživatelem, a serverem a instancí OctoPrintu. V této sekci budou definovány rozhraní, které byly při výběru technologií zvažovány a které by se dle svých specifikací byly aplikaci prospěšné.

3.1.1 HTTP protokol

Hypertext Transfer Protocol (HTTP) je bezstavový protokol, působící jako aplikační vrstva pro komunikaci mezi oddělenými systémy. Aplikační vrstva je abstraktní pojem, který znamená umožnění komunikace po síti mezi systémy za pomoci definovaných protokolů a rozhraní. V současné době je HTTP základem pro moderní web. V protokolu HTTP jsou 2 role: hostitel a klient, kde hostitel je server nabízející možnost komunikace za pomocí HTTP a klient posílá požadavky na hostitele. Tato komunikace probíhá v párech, kde klient posílá hostiteli požadavek (request) a hostitel odesílá na základě požadavku klientu odezvu (response). Tento protokol je bezstavový právě proto, že komunikace probíhá pouze jako pár požadavek a odpověď, poté je uzavřena a není třeba, aby server uchovával informace o spojení. Spojení mezi klientem a serverem probíhá za pomoci Transmission Control Protocol (TCP) transportního protokolu na portu 80. [9]

Dle [10] verze HTTP/1.1 přidala oproti verzi 1.0 práci s mezipamětí, odesílání dat po částech a trvalá spojení, která zůstávají aktivní, dokud je klient

neuzavře. Nejaktuálnější verzí protokolu je HTTP/2, kde došlo k výrazným změnám a vylepšením mezi které patří:

Jediné připojení Je potřeba pouze jedno připojení pro načtení stránky, které je otevřené, dokud je webová stránka otevřená, což redukuje potřebu otevírat nová TCP připojení.

Multiplexování Je povoleno více požadavků na jednom připojení. V předchozí verzi HTTP/1.1 musel každý přenos čekat na konec předchozího.

Server push Posílání dat klientovi, která by mohl v budoucnu potřebovat a nezažádal si o ně, za účelem zrychlení odezvy.

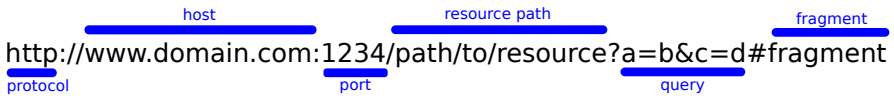
Prioritizace Možnost rychlejší reakce serveru podle vyšší priority požadavku.

Binární formát Změna oproti původní textové formě, která vynechává potřebu konverze textu na binární formu.

Komprese hlaviček Snížení velikosti přenášených dat kompresí hlaviček.

Hlavním přínosem HTTP/2 je zvýšení rychlosti přenosu, celkové odezvy, použití menšího počtu TCP připojení a snížení používané paměti serveru.

Aby bylo možné požadavek odeslat, musí mít cílového hostitele. HTTP se adresuje za pomoci Uniform Resource Locator (URL), které se skládá z:



Obrázek 3.1: Obrázek ukazuje rozdělení URL na části, ze kterých se skládá

Protocol Použitý protokol, v tomto případě HTTP, ale URL používají i jiné protokoly.

Host Adresa cílového zařízení v síti nebo doména v síti Internetu.

Port Číslo portu, port 80 je pro HTTP implicitní, 443 pro protokol Hypertext Transfer Protocol Secure (HTTPS) a není je v těchto případech nutno uvádět.

Resource path Umístění požadovaného zdroje na serveru.

Query Nepovinné parametry.

Fragment Nepovinný odkaz na zdroj na stránce.

Dalším stavebním prvkem HTTP jsou metody požadavků. Na základě těchto metod se liší struktura požadavku a každá tato metoda slouží pro vyvolání specifické akce na serveru.

GET Získá data o zdroji.

POST Odeslání dat a vytvoření nového zdroje.

PUT Odeslání dat a aktualizace existujícího zdroje.

DELETE Smazání existujícího zdroje.

Dalšími metodami jsou **HEAD**, **TRACE** a **OPTIONS**, které nejsou používány frekventovaně. [10]

Součástí odpovědi serveru na požadavek je návratový kód. Návratový neboli stavový kód je soubor unifikovaných číselných kódů v rámci HTTP protokolu. Stavový kód říká klientovi, jak interpretovat odpověď od serveru. Tyto kódy mají přesně definované kategorie:

1xx jsou pouze informační.

2xx sdělují, že server úspěšně zpracoval požadavek. Nejčastějším je `200 OK`.

3xx je nejčastěji přesměrování na jiný zdroj.

4xx jsou chyby na straně klienta, jedná se o neoprávnění uživatele, chybný požadavek nebo nenalezení zdroje na dané adrese.

5xx serverové chyby při zpracování požadavku, kde nejčastěji je použit kód `500 Internal Server Error`.

Požadavky a odpovědi jsou data v textovém nebo binárním formátu. Požadavky obsahují navíc URL a metodu, odpovědi obsahují navíc stavový kód. Oba typy zpráv se skládají z hlavičky a těla. V hlavičce jsou obsažena metadata o datech, komunikaci a spojení. V těle zprávy jsou obsažena samotná data posílaná za pomoci HTTP protokolu. [9]

3.1.2 REST API

Representational State Transfer (REST) je soubor principů a metod pro tvorbu unifikované struktury dat a Application Programming Interface (API). API je z pohledu aplikací komunikační rozhraní, které poskytují jiným knihovnám nebo aplikacím. Systém může za pomoci REST API jiné aplikace získat její data, aniž by k nim měl přímý přístup. REST je obvykle používaný s protokolem HTTP a je jedním z nejpoužívanějších API na webu pro výměnu dat mezi klientem a serverem. [11]

Základní principy architektury REST jsou:

- Konzistence struktury API.
- Bezstavovost v souvislosti s HTTP protokolem.
- Používání vhodných HTTP stavových kódů.

- Logická hierarchie URL koncových bodů.
- Verzování API přímo v URL.

K manipulaci s daty se používají HTTP metody GET, POST, PUT, DELETE, jak byly popsány v sekci 3.1.1. Většinou se REST API koncové body oddělují od zbytku předponou v cestě ke zdroji `/api`, za touto předponou mnohdy bývá verze kvůli zachování starého API a až poté cesta k datům `/api/v2/example`. Samotná cesta ke zdroji by se měla skládat z podstatných jmen, případně čísel, pokud je třeba přiložit Identification (ID) zdroje. Celkově by cesta měla odpovídat datové struktuře serverové aplikace. Klientu se vrací data většinou ve formátu JavaScript Object Notation (JSON), případně ve formátu Extensible Markup Language (XML). [11]

3.1.3 GraphQL

GraphQL je dotazovací jazyk pro API používající protokol HTTP [12]. Těmito dotazy se klient ptá přesně na všechna konkrétní data, která v daný okamžik potřebuje. Všechna data jsou dostupná na jediném koncovém bodu. Díky tomu může uživatel z jednoho koncového bodu systému získat více zdrojů v jednom požadavku. GraphQL API je popsáno jako sada typů (tříd) s atributy a jejich náležitých typů:

```
type Query {
  hero: Character
}
type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
}
type Planet {
  name: String
  climate: String
}
```

Ukázka kódu 1: Příklad těla požadavku pro GraphQL koncový bod

Tento zápis definuje všechna data, která lze získat z koncového bodu API. Klient je nucen přesně definovat všechny zdroje a jejich atributy, které chce vrátit. Ve výsledku GraphQL vytváří jednotné API s jedním koncovým bodem a nedefinuje nijak datovou strukturu ani způsob přechování dat. V GraphQL je třeba specifikovat všechna pole, které chce uživatel vrátit, což ve většině REST API koncových bodů není potřeba.

V porovnání GraphQL a REST podle [13] se REST zaměřuje na stavbu jednotného, logického API místo optimalizace pro výkon. Jelikož je URL unikátní identifikátor pro každý zdroj, je možné odpovědi načítat z mezipaměti. Oproti tomu GraphQL je zaměřeno na výkon sjednocením celého API do jednoho koncového bodu. Jelikož tělo dotazu v GraphQL vždy obsahuje specifický datový formát, není možné nahrávat binární data, tedy média a kvůli tomu je třeba vytvořit jinou službu, která bude tato data zpracovávat.

3.1.4 WebSocket

WebSocket je protokol pro obousměrnou komunikaci po síti za použití jediného soketu [14]. Sokety umožňují komunikaci mezi 2 přístroji pomocí metod čtení a zápisu dat [15]. Obvykle je používaný webovými prohlížeči ve formě spojení klienta se serverem. Tento protokol byl představen společně s Hypertext Markup Language (HTML) 5.

WebSocket protokol pro zahájení komunikace používá HTTP protokol, kde komunikace se z HTTP protokolu změní na WebSocket. Klient pošle serveru dotaz s hlavičkou `Connection: Upgrade`, server odpoví s hlavičkou `Upgrade: WebSocket` a tím se uskuteční WebSocket spojení. HTTP spojení je nahrazeno WebSocket spojením, které používá stejné porty 80 a 443.

WebSocket je vytvořen pro komunikaci v reálném čase. V případě HTTP je buď nutné, aby klient posílal periodicky požadavky na server, kde je potřeba, aby se pokaždé otevírala nová komunikace, nebo server neuzavře komunikaci a neustále posílá informace klientu. V obou případech požadavky a odpovědi zahrnují HTTP hlavičky, což znamená posílání dalších dat, celkové zpomalení komunikace a zahlcování serveru. Pro HTTP oboustrannou komunikaci jsou zapotřebí 2 spojení. Nevýhodou WebSocketu je, že ne všechny prohlížeče ho podporují. [16]

3.1.5 Socket.IO

Socket.IO je knihovna umožňující obousměrnou komunikaci pro webové aplikace [17]. Zpřístupňuje aplikaci komunikaci pomocí WebSocket protokolu i HTTP metody long polling. Long polling je technika, kde klient udělá požadavek na server, který nechává požadavek otevřený, s tím i spojení, čeká než na nová data a odešle odpověd. Ihned poté klient udělá nový požadavek [18]. Oba druhy komunikace jsou zavedeny, aby vývojář nemusel řešit nekompatibilitu některých prohlížečů s WebSocket protokolem a zapouzdřuje je do jednoho API, pro zjednodušení vývoje.

Klientské i serverové rozhraní jsou si podobná. Pro server i klient je potřeba implementovat event listenery, což jsou funkce vyvolané například po získání připojení nebo určitých dat. Data se posílají vyvoláním eventů, které jsou na druhé straně komunikace zpracovány event listenery. Socket.IO umožňuje kromě posílání dat i streamování dat ze serveru na klient.

3.1.6 SockJS

SockJS je knihovna, která vznikla, jelikož WebSocket protokol není stále podporován všemi internetovými prohlížeči [19]. Tato knihovna emuluje WebSocket protokol a jejím účelem je co nejvíce se přiblížit komunikaci v reálném čase. Podporuje WebSocket, streamování založené na HTTP1.1, odesílání dat po částech a techniku polling (periodické opakování požadavku za účelem získání nových dat). Při vytváření připojení nejdříve klient vyzkouší WebSocket, pokud selže, SockJS otevře XmlHttpRequest (XHR), který vyzkouší, jestli je rozdělení dat na kusy podporováno proxy, pokud ne, je použita technika polling. Dle [20] je XHR API sloužící pro přenos a manipulaci XML dat za pomoci HTTP protokolu.

3.2 OctoPrint

Úvod k OctoPrintu je popsán v sekci 1.5, kde je popsáno, k čemu OctoPrint slouží a jaké jsou jeho funkcionality z pohledu uživatele.

Pro práci aplikace s více instancemi OctoPrint, jak po lokální síti tak Internetu je potřeba síťové komunikace podporované OctoPrintem. OctoPrint v základu podporuje zejména REST API, ale také SockJS.

REST API vyžaduje API klíč v každém požadavku. Tento klíč je řetězec nastavený v OctoPrintu, aby s tiskárnou mohl pracovat pouze oprávněný uživatel nebo aplikace. Dle dokumentace [21] REST API podporuje tyto akce:

- Získání informací o verzi, tiskárně a instanci OctoPrintu.
- Připojení OctoPrintu k tiskárně.
- Souborové operace: získání informací o souborech, nahrání souboru na tiskárnu, výběr a tisk souboru, převod STL na G-code, smazání souboru ze systému.
- Zapnutí tisku, přerušení, pozastavení, restart tisku, získání aktuálních informací o tisku (procentuální zhotovení, uplynulá doba, zbývající doba, jméno souboru).
- Informace o záznamových souborech (log files).
- Aktuální informace o tiskárně (teploty, stav), pohyb trysky tiskárny, nastavení teplot, vytlačení materiálu, poslání G-code příkazů.
- Získání nastavení profilů tiskárny (velikost, počet trysek), vytvoření profilu, úprava a smazání.
- Získání nastavení OctoPrintu, změna nastavení, generování nového API klíče.

- Informace o systémových příkazech a jejich vykonání.
- Uživatelské operace: získání uživatelů, přidání, mazání, úprava, resetování hesla, získání uživatelovo nastavení a jeho změna.

Pro získání dat v reálném čase OctoPrint používá SockJS pro neustálé posílání informací o svém stavu, teplotě a průběhu tisku. Po připojení k soketu OctoPrintu odešle nejdříve informace o spojení, historii stavu a poté aktuální informace o stavu teplotě a průběhu tisku. V základu jsou aktuální informace posílány každých 500 ms, to lze změnit posláním zprávy na soket s příkazem `throttle` s číselnou hodnotou, které násobí základní hodnotu 500 ms. [3]

Pro práci s API OctoPrintu existují 2 knihovny:

foosel/octoprint_client Jedná se o knihovnu přímo od autorky OctoPrintu napsanou v jazyce Python a vydanou pod licencí GNU AGPL [22]. Tato knihovna umožňuje SockJS komunikaci za pomoci WebSocket protokolu a také komunikaci s REST API bez předdefinovaných funkcí pro náležité koncové body. Vytváří tedy pouze základní rozhraní, s implementovanou komunikací.

hroncok/octoclient Autorem této knihovny [23] je vedoucí této práce Ing. Miroslav Hrončok. Je vydaná pod licencí MIT a je napsána v jazyce Python. Umožňuje práci s REST API za pomoci předdefinovaných funkcí pro jednotlivé koncové body a také komunikaci v reálném čase za pomoci SockJS metodou XHR streamováním dat. Tedy vytváří rozhraní s hotovými příkazy, ale komunikace v reálném čase je zastaralejší.

3.2.1 Testování

Pro vývoj a tetování aplikace, která bude komunikovat s nabízenými rozhraní OctoPrintu po síti, je potřeba mít spuštěnou službu OctoPrint v lokální síti nebo internetu. Jednou možností je testování na tiskárně s Raspberry Pi používající OctoPrint, tato možnost vyžaduje přístup k tiskárně, která není pro každého dostupná. Další možností využít možnosti OctoPrintu konfigurace do vývojového módu, který simuluje virtuální tiskárnu. Je možné využít předkompilovaného obrazu OctoPrintu běžícího ve službě Docker, který je dostupný jako součást bakalářské práce [24]. Docker je software, který umožňuje běh aplikací nezávisle na platformě či operačním systému v izolovaném prostředí. [25]. Virtuální OctoPrint lze jednoduše spustit s nainstalovanou službou Docker v systému:

```
docker run -p3200:5000 josefdolezal/virtuprint-docker
```

Ukázka kódu 2: Příkaz pro spuštění docker kontejneru pro virtuální prostředí OctoPrint

Tento příkaz spustí OctoPrint službu a přesměruje rozhraní na port 3200. Nevýhodou tohoto způsobu je chybějící funkcionalita umožňující detekci tiskáren v lokální síti za pomoci multicast Domain Name System (mDNS), která je popsána dále v sekci 3.4. Další možností je spuštění OctoPi operačního systému na bázi Raspbianu na Raspberry Pi s vývojářskou konfigurací a virtuální tiskárnou. Tato možnost zahrnuje práci s mDNS.

3.3 Přihlášení uživatelů

V zadání stojí, že do aplikace musí být přihlášení uživatelů pomocí uživatelských účtů s možností specifikace oprávněných osob. Fakulta nabízí přístup na FIT ČVUT OAuth 2.0 autorizační server, který vydává a validuje přístupové tokeny [26].

3.3.1 OAuth2

OAuth2 je autorizační framework. Funguje tak, že aplikace požádá o omezený přístup k uživatelskému účtu jinou službu používající HTTP, která má k tomuto účtu přímý přístup. Služba, ke které uživatel přistupuje, přesměruje uživatele pro přihlášení na autorizační server, kde se uživatel přihlásí a autorizuje původní službu pro použití přístupu k jeho účtu.

Dále je potřeba vysvětlit 3 role:

Uživatel Vlastník zdroje, který klientské aplikaci autorizuje přístup k jeho uživatelskému účtu v určeném rozsahu (scope).

Autorizační server Služba, na které se nachází uživatelský účet a která ověřuje uživatele a vydává přístupové tokeny.

Klientská aplikace Aplikace žádající autorizační server o přístup k uživatelskému účtu.

Úspěšný průběh probíhá tak, že klientská aplikace požádá autorizační server o přístup k uživatelovým datům. Uživatel potvrdí požadavek na autorizačním serveru a klientské aplikaci je udělen autorizační grant. Klientská aplikace požádá autorizační server o přístupový token pomocí autorizačního grantu, registrovaným ID a tajným kódem. Autorizační server pošle klientské aplikaci přístupový token, čímž je autorizační dokonána. Za pomoci přístupového tokenu uživatele může aplikace přistupovat k zdrojům tohoto uživatele.

Průběh také záleží na autorizačním grantu, který byl pro aplikaci zvolen. OAuth2 nabízí 4 autorizační granty:

Authorization code Autorizační kód je nejvíce používaný grant. Je určen pro serverové aplikace, jelikož zdrojový kód není dostupný a klientský tajný kód je chráněn. Přístupový token vyprší po definované době, poté je třeba za pomoci obnovovacího tokenu získat nový přístupový token.

Password Tento grant funguje tak, že uživatel zadá své přístupové informace přímo aplikaci, která s jejich pomocí požádá o přístupový token. Měl by být použit pouze, pokud uživatel důvěřuje aplikaci, jelikož jí přímo sděluje své přístupové údaje.

Client credentials Tento grant nijak nezahrnuje uživatele, slouží pouze pro přístup do API souvisejícím s klientskou aplikací.

Implicit Implicitní grant je používaný pro mobilní zařízení a aplikace na straně klienta, tedy kde klientský tajný kód není chráněn.

Aby aplikace mohla autorizovat proti OAuth2 autorizačnímu serveru, je nejdříve potřeba aplikaci na server zaregistrovat. Pro registraci je použit název aplikace, stránky a URL pro přesměrování po přihlášení uživatele. Poté je obdrženo klientské ID a klientský tajný kód. Klientské ID je veřejný identifikátor aplikace, kdežto klientský tajný kód je formou hesla aplikace a musí být držen v tajnosti [26][27].

Autorizační server FIT ČVUT dle své dokumentace [28] nabízí pouze autorizační granty `authorization code`, `client credentials` a `implicit`. Nejvhodnější pro aplikaci bude grant `authorization code`, jelikož je potřeba pracovat s identitou uživatele a software bude webová aplikace na straně serveru.

Autorizační grant `authorization code` je nejpoužívanější metoda OAuth2 určená pro serverové aplikace, kde klientský tajný kód je bezpečně uchován na serveru. Nejdříve je uživatel přesměrován na autorizační koncový bod autorizačního serveru ve tvaru `https://auth.fit.cvut.cz/oauth/authorize?response_type=code&client_id=&redirect_uri=&scope=urn:zuul:oauth`, kde pole `request_type` záleží na typu autorizačního grantu, `scope` je úroveň požadovaného přístupu a klientovo ID a URL pro přesměrování lze získat z registrace klientské aplikace. Uživatel se zde přihlásí a poté bude požádán o udělení povolení pro klientskou aplikaci. Po udělení povolení autorizační server přesměruje uživatele na `redirect_uri` ve formě `http://clientapplication.com/home?code=p8pUAJ`. Server poté požádá o přístupový token, tím že zašle požadavek s obdrženým kódem na koncový bod pro vydávání přístupových tokenů ve tvaru `https://auth.fit.cvut.cz/oauth/token?client_id=&client_secret=&grant_type=authorization_code&code=p8pU6J&redirect_uri=`

Klientský tajný kód by se neměl posílat v parametrech URL kvůli bezpečnosti, ale společně s klientským ID zakódovaný v `Authorization` hlavičce požadavku algoritmem `Base64` [29]. Pokud bude celá autorizace úspěšná, autorizační server pošle klientské aplikaci přístupový token, obnovovací token, čas uplynutí platnosti přístupového tokenu, typ tokenu a úroveň přístupu ve tvaru JSON:

```
{
        "access_token": "ea173f10-babc-404f-b88d-f0ee8d95ff7c",
        "token_type": "bearer",
        "refresh_token": "aba6d32d-4f17-49e6-afcc-1f042f3e6d3c",
        "expires_in": 1209599,
        "scope": "urn:zuul:oauth"
}
```

Ukázka kódu 3: Ukázka JSON odpovědi s přístupovým a obnovovacím tokenem z FIT ČVUT OAuth2 autorizačního serveru

Nyní lze používat přístupový token v přístupu na zdroje na API autorizačního serveru. V případě, že přístupový token vyprší, musí klientská aplikace použít obnovovací token pro získání nového přístupového tokenu. [27][28]

3.3.2 JSON Web Token

JSON Web Token (JWT) je řetězec, který je používán pro poslání informací, které mohou být ověřeny digitálním podpisem [30]. Jedná se o objekt ve formátu JSON, který je kryptograficky podepsán. Výsledkem je řetězec, který často bývá posílán v `Authorization` hlavičce požadavků. Toto téma je zde uvedené, jelikož se obvykle při použití REST API používá tokenová autentizace.

Tento řetězec se skládá ze 3 částí oddělených tečkou:

Hlavička Je objekt formátu JSON s atributy `alg`, který obsahuje použitý šifrovací algoritmus a atribut `typ`, který deklaruje zakódovaný objekt jako JWT. Hlavička je zakódována algoritmem Base64, čímž vznikne první řetězec.

Vlastní obsah JSON objekt s rezervovanými atributy: `iss` zprostředkovatel tokenu, `sub` vlastník tokenu, `aud` pro koho je token určen, `exp` expirace ve formě časového razítka, `nbf` čas, před kterým je token neplatný ve formě časového razítka, `iat` kdy byl token vytvořen ve formě časového razítka, `jti` unikátní ID pro token. Další atributy mohou být pro vlastní potřebu. Celý objekt je zakódován Base64 z čehož vznikne druhý řetězec.

Podpis Třetí část JWT je vytvořena spojením již zakódované hlavičky, tečky a zakódovaného obsah tento celý řetězec je zašifrován silným algoritmem jako Keyed-hash Message Authentication Code Secure Hash Algorithm (HMAC SHA-256) nebo Rivest, Shamir, Adleman (RSA), kde tajný kód je bezpečně uložen na serveru, kde je token vytvořen.

Výhodou tokenové autentizace je zastoupení ukládání dočasných dat o uživateli v relaci na serveru, token může validovat více systému, tedy je znovupoužitelný. Uživatel pošle serveru požadavek s přihlašovacím jménem a heslem,

server vytvoří JSON web token s uživatelovými údaji a pošle ho v odpovědi. Při každé příležitosti, kdy uživatel přistupuje k zdroji s omezeným přístupem, pošle v hlavičce `Authorization` například řetězec `JWT VCJ9.lIn0.Q6C1`, kde první část `JWT` naznačuje o jaký druh autorizace se jedná a druhá část je JWT.

3.4 Detekce tiskáren v lokální síti

OctoPrint od verze 1.2 obsahuje v základu plugin pro detekci v síti [31]. Tento plugin umožňuje ihned nalézt instanci OctoPrintu na lokální síti bez znalosti Internet Protocol (IP) adresy. Využívá protokolu Universal Plug and Play (UPnP) pro zobrazení na zařízení s operačním systémem Windows a Zeroconf pro zařízení s macOS X a Linux. Zeroconf je podporován pouze pokud je na systému nainstalována Python knihovna `pybonjour`. OctoPrint používá jména služeb s příponou `_http._tcp.local` a `_octoprint._tcp.local`, kde první možnost je běžně používaná i jinými službami.

Zeroconf nebo-li Zero-Configuration protokol [32] je technologie pro automatické objevení systémů a služeb v lokální síti. Umožňuje zařízením získat IP adresu i v případě, že chybí Dynamic Host Configuration Protocol (DHCP) server a získat doménové jméno bez Domain Name System (DNS) serveru. Dává systémům možnost oznamovat svoji službu na síti a objevovat služby na lokální síti. Zeroconf přiřadí systému doménové jméno za pomoci mDNS, tím způsobem, že zařízení pošle DNS zprávy do lokální sítě a pokud jiné zařízení neodpoví, že vlastní doménové jméno, přivlastní si ho původní zařízení a oznámí ho v lokální síti. Zařízení musejí používat lokální doménu nejvyššího řádu `.local`. Služby se prezentují pod složeným jménem `name._octoprint._tcp.local`. Za pomoci záznamů jsou nalezena všechna dostupná zařízení s IP adresou, doménovým jménem a dalšími informacemi.

Při vývoji aplikace, umožňující detekci zařízení v lokální síti lze použít například knihovnu `python-zeroconf` na [33], která je vydána pod licencí GNU Lesser General Public License (LGPL). Tato knihovna umožňuje jak registraci služeb tak jejich detekci v lokální síti.

3.5 Databáze

Jelikož se v zadání práce objevují pojmy jako: tiskárna, skupina, uživatel bude potřeba v aplikaci nějaký druh persistence. Aplikace bude muset být také konfigurovatelná, hlavně kvůli nastavení OAuth2, jak je popsáno v sekci 3.3.1. Persistence je způsob uložení dat takovým způsobem, že data zůstanou i po skončení procesu, kde se s nimi pracovalo [34]. Nejvhodnější možností bude zvolení relační databáze kvůli vztahům mezi uživateli, skupinami a tiskárnami.

Databáze je systém pro přechovávání dat v předdefinovaném modelu, nejedná-li se o databázi beze schématu jako je NoSQL. Relační databáze je, taková kde entity nebo-li záznamy mohou mít mezi sebou vztahy, odkazovat

na sebe. Tyto databáze obsahují tabulky, které mají sloupce, což jsou atributy, které ukládají určitý druh informace. Každý řádek je entita nebo-li záznam, který povinně obsahuje v celé tabulce unikátní ID jako atribut. MySQL, PostgreSQL a SQLite jsou nejznámější a nejpoužívanější relační databáze. [35]

3.5.1 SQLite

SQLite je druh databáze, který je tvořen jediným souborem na disku, což zaručuje jednoduchou přenositelnost dat a jednoduché připojení aplikace bez použití soketové komunikace. Nevýhodou této databáze je absence správy uživatelů a přístupových privilegii. Na SQLite databázi se může připojit kdokoliv, kdo má přístup k danému souboru. SQLite se používá zejména pro vestavěné aplikace, které vyžadují přenositelnost nebo pro účely testování. Nepoužívá se, pokud k databázi přistupuje více uživatelů nebo aplikací a v aplikacích, které jsou náročné na zápis dat, jelikož SQLite je limitovaná pouze na jednu operaci zápisu najednou. [35]

3.5.2 MySQL

MySQL je nejpopulárnější relační databázový systém. Funguje jako samostatný server nebo služba, se kterou aplikace komunikují pro účel zápisu nebo čtení dat. Výhodou MySQL je díky své popularitě instalace, mnoho funkcionalit, správa uživatelů a přístupu a možnost škálovatelnosti. I když je MySQL projekt s otevřeným zdrojovým kódem, jeho vývoj se příliš nepohybuje vpřed. MariaDB dle [36] je nástupcem MySQL, který je nadále vyvíjen a to pod licencí GNU General Public License (GPL). MySQL by se mělo používat, pokud je zapotřebí samostatný databázový server, pokud přístup používá více uživatelů nebo aplikací. Nehodí se v případech, že je potřeba vyhledávat v fulltextu, nebo jsou potřeba jiné funkcionality, které nezahrnuje. [35]

3.5.3 PostgreSQL

PostgreSQL je objektově-relační databáze, což znamená, že kromě standardního přístupu persistence relací může také ukládat objekty. Výhodou PostgreSQL je mnoho funkcionalit a datových typů navíc, možnost programové rozšířitelnost a objektivní funkcionalita s možností vnořování. PostgreSQL je vhodná, pokud má databáze vykonávat vlastní procedury, čehož je docíleno rozšiřitelností nebo pokud je třeba v budoucnu databázi měnit. Nevýhodou je nižší rychlost čtení a podpora zejména komplexních databází. [35]

Kapitola 4

Realizace

V této kapitole je znázorněno, jaké technologie, knihovny, rozhraní a metody byly použity a proč. Nachází se zde popis struktury zdrojového kódu, použitá řešení a možné vylepšení. Na konci kapitoly je uvedeno jak byla aplikace testována.

4.1 Technologie

Aplikace je rozdělena na 2 celky:

Frontend Klientská aplikace, tedy to, co uživatel vidí, s čím přímo pracuje v internetovém prohlížeči. Je napsána v jazyce JavaScript se strukturou HTML a stylováním pomocí Cascading Style Sheets (CSS). Je použit strukturový framework AngularJS a framework komponent pro uživatelské rozhraní AngularJS Material. Pro vytváření balíčků, automatické obnovování při změně a minifikaci zdrojového kódu se zde používá služba gulp.

Backend Serverová aplikace, která poskytuje REST API a koncový bod pro získání frontendové aplikace. Je napsán v jazyce Python pro verzi 3, ve frameworku Flask. Pro databázi je použito SQLite a Object-relational mapping (ORM) knihovna SQLAlchemy.

4.2 Uživatelské role

V aplikaci mají uživatelé definovány 3 role:

Superadmin Má přístup ke všem skupinám a tiskárnám v aplikaci, může přidávat, měnit nastavení a odebírat tiskárny i skupiny. Může přidávat superadminy a měnit nastavení aplikace. Superadmin je role ve vztahu

k aplikaci, definovány přímo v tabulce uživatelů v databázi. Superadminem se uživatel stane, pokud je přidán pomocí příkazové řádky nebo jiným superadminem přímo v nastavení aplikace.

User Roli uživatele má jakýkoliv uživatel v základní formě, bez administrátorského vztahu ke skupině. Uživatel vidí skupiny a tiskárny, ke kterým mu byl poskytnut přístup, nemá přístup do nastavení aplikace.

Admin Role admin je definována ve vztahu uživatele ke skupině. Pokud má uživatel roli admin ke skupině, může stejné funkcionality jako uživatelská role a může měnit nastavení skupiny, smazat skupinu a měnit nastavení přiřazených tiskáren. Pokud má uživatel alespoň k jedné skupině roli admin, má přístup do nastavení aplikace bez funkcionalit pro superadmina.

4.3 Frontend

Frontend je vizuální stránka aplikace pro uživatele, zahrnuje práci v jazycích jako je HTML, CSS a JavaScript [37]. V tomto případě je frontend oddělitelný od backendu jako samostatná aplikace. Implementace softwaru začala frontendem, aby před začátkem práce na backendu byly definovány všechny potřebné koncové body. Mezitím byla použita platforma `Apiary`, která slouží mimo jiné pro simulaci a vytváření prototypů koncových bodů [38]. Skrze jednoduché předpisy jsou vytvořeny koncové body, které vracejí nadefinovaná statická data.

4.3.1 Zvolené technologie

AngularJS Framework AngularJS byl zvolen kvůli autorovým osobním preferencím a zkušenostem s tímto frameworkem. Jedná se o strukturový Model View Controller (MVC) framework, kde je používáno HTML a rozšířeno o předdefinované nebo vlastní komponenty. Dynamická data jsou oboustranně vázána s HTML dokumentem. MVC architektura dělí aplikaci na 3 logické části: model, který reprezentuje data a datovou logiku, view, který předtavuje veškeré uživatelské rozhraní a controller, který představuje spojení mezi modelem a view, manipuluje s daty a zabývá se všemi příchozími požadavky [39]. Angular slouží pro tvorbu single page aplikací, což jsou webové aplikace, které pracují čistě na straně klienta a se serverem pouze komunikují za pomoci definovaných rozhraní, aniž by se stránka musela po každém požadavku znovu načítat. [40][41]

AngularJS Material Tento framework je vytvořen jako rozšíření pro AngularJS. Jedná se pouze o framework obsahující komponenty pro uživatelské

rozhraní. Tyto komponenty jsou založené na návrhu Material Design s úpravou pro práci s AngularJS frameworkem. [42]

Gulp Je nástroj pro automatizace úloh. Slouží především pro sestavení a minimalizaci rozsáhlé javascriptové aplikace do menšího počtu souborů, pro nasazení aplikace, nebo pro automatické obnovení aplikace v prohlížeči po změně v zdrojovém kódu pro vývoj aplikací. Výhodou balíčku gulp je programovatelnost a mnoho různých rozšíření pro vykonávání úloh. [43]

NPM Node Package Manager (NPM) je software pro správu a instalaci javascriptových balíčků a knihoven z příkazové řádky. Slouží pro správu globálních balíčků ale i lokálních pro projekt. Pro projekt se knihovny nachází ve složce `node_modules` a závislosti jsou zapsané v souboru `package.json` v kořenové složce projektu. Pomocí tohoto souboru lze ihned nainstalovat všechny potřebné závislosti pro početí vývoje aplikace. [44]

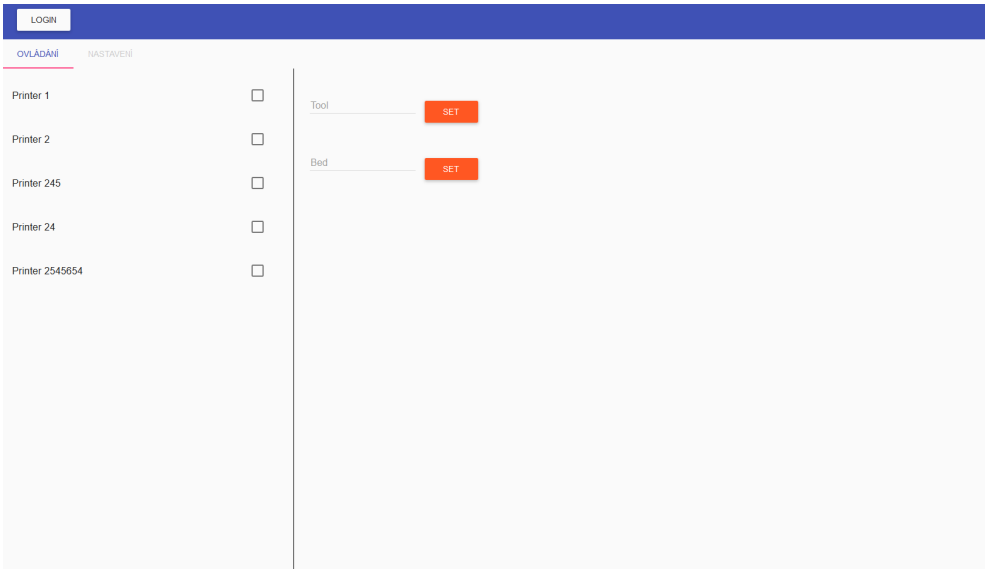
4.3.2 Inicializace projektu

Pro vytvoření kostry frontendové části projektu byl použit generátor Yeoman Fountain [45]. Výsledkem byla kostra aplikace s AngularJS frameworkem, preprocesorem Babel pro ECMAScript2015 JavaScript, již nakonfigurovaným softwarem gulp s možností sestavení pro nasazení a sestavení s obnovením v prohlížeči po změně v kódu. Za pomoci NPM byly přidány závislosti na frameworku Angular Material, knihovny pro vytváření a spravování vnořených cest (route) nazvané AngularUI router a knihovny pro vytváření HTTP požadavků pro interakci s REST API nazývající se Angular Resouce. Tyto moduly byly naimportovány do hlavního souboru `src/index.js`.

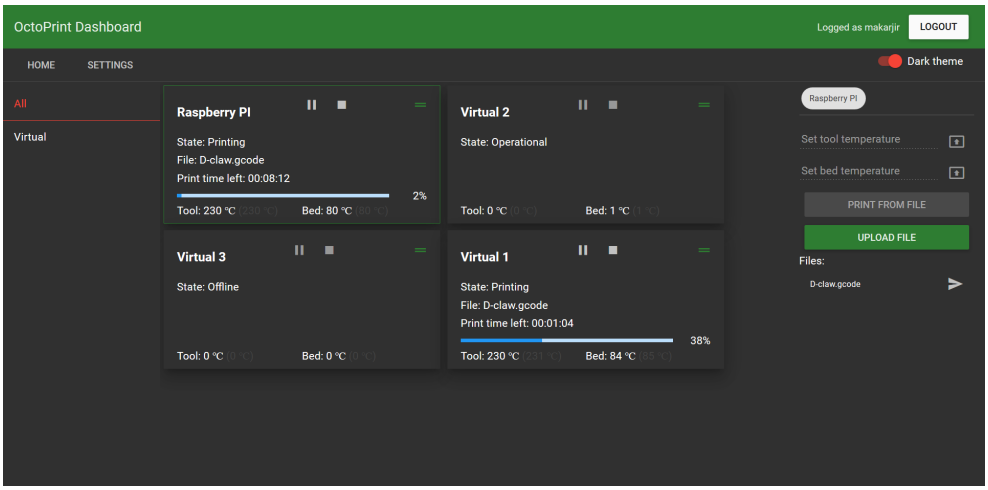
4.3.3 Návrh uživatelského rozhraní

První návrh uživatelského rozhraní je zachycen na obrázku 4.1. Kde v horní části je lišta s tlačítkem pro přihlášení. Na levé straně je seznam tiskáren, kde každá tiskárna má zaškrtávací políčko. Na pravé straně je prostor s ovládacími prvky tiskárny, kde zatím nejsou doimplementované prvky jako seznam souborů, tlačítko pro tisk ze souboru a tlačítko pro nahrání souboru. Po zaškrtnutí jedné nebo více tiskáren se uživateli objeví prvky pro ovládání, se kterými může pracovat s označenými tiskárnami. Nevýhodou tohoto návrhu je velké prázdné místo v pravé části, kde jsou ovládající prvky, které se zobrazí pouze po zaškrtnutí tiskáren a málo místa pro tiskárny a data z monitorování.

Při první prezentaci tohoto návrhu byl vznesen požadavek pro možnost přeřazovat pořadí tiskáren a vylepšit celkové rozdělení prostoru. Druhý návrh uživatelského rozhraní na obrázku 4.2 zvětšuje prostor pro tiskárny na šířku 75 % okna prohlížeče a ovládací prvky na zbylých 25 % okna napravo. Seznam



Obrázek 4.1: Obrázek zachycující vzhled prvního návrhu uživatelského rozhraní



Obrázek 4.2: Obrázek zachycující vzhled konečné formy uživatelského rozhraní hlavní stránky

tiskáren byl předělán na mřížku s možností přetahování pro změnu pořadí tiskáren v mřížce. Funkcionalita přetahování elementů v AngularJS není v základu implementovaná. Proto byla použita knihovna `angular-sortable-view`, která podporuje mřížku s tahacími prvky [46].

Prvek tiskárna obsahuje informace o tiskárně a tisku, má také akce pro pozastavení, pokračování a zastavení tisku. Pro výběr je použito zvýraznění okraje elementu barvou po stisknutí namísto zaškrtnutí neoznačeného zaškr-

távacího tlačítka. V levé části je svislá lišta se seznamem dostupných skupin tiskáren a možností přepínat mezi tiskárnami s základní skupinou `All`, která obsahuje všechny uživateli dostupné tiskárny. Na konec lišty s menu byl přidán přepínač pro tmavé a světlé téma. Pravá svislá lišta slouží pro ovládací prvky, kde je seznam zaškrtnutých tiskáren, nastavení teplot, nahrání souborů, tisk a v případě vybrání jedné tiskárny seznam souborů na tiskárně s akcemi.

Do záložky nastavení má přístup pouze superadmin nebo uživatel s alespoň jednou rolí admin ve vztahu ke skupině. Na levé straně jsou prvky související s tiskárnami. Jsou zde zobrazeny všechny dostupné tiskárny pro uživatele, kterým je možné měnit nastavení případně je mazat. Tlačítka pro smazání a přidání tiskárny jsou přístupná pouze superadminovi. Tlačítko nastavení otevře pro zvolené tiskárny modální okno, kde je možné skupinově měnit teplotní přednastavení tiskáren. Pokud byla vybrána pouze jedna tiskárna pro nastavení a uživatel je superadmin, zpřístupní se hlavní nastavení tiskárny a je možné měnit přístupové informace k tiskárně.

Ve střední části obrazovky pro nastavení se nachází ovládání pro skupiny. Součástí je tlačítko pro přidání skupiny, které může použít pouze superadmin. V seznamu skupin jsou všechny odstupné skupiny, které uživatel může upravovat nebo mazat. Po akci úprava skupiny se otevře modální okno, kde lze měnit jméno skupiny, příslušné tiskárna a uživatele a jejich role. Pro superadmina se na pravé straně zobrazí tlačítko pro přidání superadmina a nastavení pro periodu obnovení informací o stavu tiskáren na serveru a klientu.

4.3.4 Komponenty

Aplikace je rozdělena na AngularJS komponenty, kde každá komponenta se skládá z HTML šablony a JavaScript controlleru, čímž splňuje architekturu MVC, jelikož modelovou vrstvu tvoří služby pro komunikaci se serverem [39]. Komponenty jsou v aplikaci zavedeny kvůli logickému rozdělení aplikace na funkční celky, které mezi sebou budou komunikovat. Každá komponenta je nějaký druh prvku na obrazovce uživatele. V této sekci budou uvedeny pouze nejdůležitější komponenty.

Toolbar Komponenta `toolbar`, je horní vodorovnou lištou v aplikaci. Tato komponenta je umístěná v základní šabloně aplikace a je neměnná pro jakékoliv URL aplikace. Obsahuje tlačítko pro přihlášení, které při akci přihlásí uživatele. Přihlášení uživatele probíhá autorizačním frameworkem OAuth2 na server FIT ČVUT grantem `Authorization code`. Proces přihlášení spravuje modul do AngularJS s názvem `Satellizer` [47]. Aby tento modul správně fungoval, bylo třeba nakonfigurovat: klientské ID, URL pro přesměrování, autorizační koncový bod a úroveň přístupu pro OAuth2, tyto informace lze získat po registraci na [48]. Dále byl potřeba nastavit koncový bod, na který má být získaný autorizační kód poslán.

Po stisknutí přihlašovacího tlačítka je zavolána funkce `$auth.authenticate('CVUT')`, která vytvoření promise, který otevře nové okno prohlížeče s přihlašovací stránkou autorizačního serveru, po přihlášení a odsouhlasení autorizace klientské aplikace se okno zavře a pošle se požadavek na koncový bod definovaný v nastavení. Promise je objekt reprezentující vyhodnocení nebo selhání asynchronní operace [49]. Tento promise po vyhodnocení vrátí odpověď z koncového bodu serveru, na kterou se reaguje uložením JWT pomocí registrované služby `$auth` vytvořené modulem `Satellizer` do uživatelova lokálního uložiště. V této fázi je uživatel v kontextu aplikace přihlášen a přihlašovací tlačítko je nahrazeno tlačítkem pro odhlášení, současně nastavením JWT tokenu službou `$auth` se automaticky token posílá při každém HTTP požadavku na server v hlavičce `Authorization`.

Součástí této komponenty je stav přihlášení, tedy kdo je přihlášen, navigační lišta s cestami `HOME` a `SETTINGS` a prvek pro přepínání mezi tmavým módem uživatelského rozhraní a světlým.

printerGrid Komponenta uchovávající mřížku s tiskárnami. Tiskárny je možné přeřazovat díky modulu `angular-sortable-view` [46]. Controller této komponenty při inicializaci zažádá o dostupné tiskárny, skupiny a spustí periodické vykonávání požadavků na stav dostupných tiskáren. Příchozí data jsou slučována do pole s lokálními daty, tak aby se při každé aktualizace neměnilo pořadí tiskáren v poli a tedy v zobrazení pro uživatele. Periodické požadavky na server jsou vykonávány dle nastavení. Tento problematika je řešena v sekci 4.5

4.3.5 Služby

Modul služeb byl zaveden pro oddělení znovupoužitelných funkcionalit do samostatných modulů a pro oddělení logiky:

eventListeners Jedná se o funkci spuštěnou při načtení AngularJS aplikace v prohlížeči. Obsahuje funkci, která kontroluje, zda-li je uživatel při přístupu na cesty s omezeným přístupem přihlášen a má roli admin nebo superadmin, jinak je přesměrován na hlavní stránku. Dále obsahuje HTTP požadavek pro získání konfigurace ze serveru. po získání konfigurace nastaví modul `Satellizer` umožňující OAuth2 proces.

filters Obsahuje filtry pro formátování dat v šablonách AngularJS aplikace.

restServices Obsahuje služby a funkce využívající modul `$resource` pro vytváření JavaScipt promise objektů za účelem komunikace s koncovými body REST API serverové částí aplikace.

4.4 Backend

Backend aplikace je všechno co uživatel nevidí. Backendová nebo serverová aplikace je oddělená datová vrstva. Jedná se hlavně o práci s daty, připojení k databázi a řízení přístupu k jednotlivých koncovým bodům. [37] Implementace backendu navázala na frontend, který stanovil potřebné koncové body pro běh aplikace.

4.4.1 Zvolené technologie

Flask Aplikace je vyvíjena v jazyce Python 3, kvůli autorovým osobním preferencím, jelikož existují 2 knihovny pro Python pro práci s API OctoPrintu popsané v sekci 3.2 a jelikož OctoPrint je sám napsán v programovacím jazyce Python. Flask je odlehčený framework pro vytváření webových aplikací. Výhodou Flask frameworku je minimální struktura projektu, tedy základní forma projektu nemá složitou strukturu. Díky tomu se v malých projektech napsaných ve Flask frameworku dá lépe vyznat. [50]

SQLAlchemy Knihovna poskytující ORM a nástroje pro práci s databází pro Python. Tato knihovna využívá pro ORM data mapper pattern. Data mapper pattern je architektura, kde existuje služba pro obousměrné převádění mezi objekty v paměti a záznamy v databázi. [51] [52]

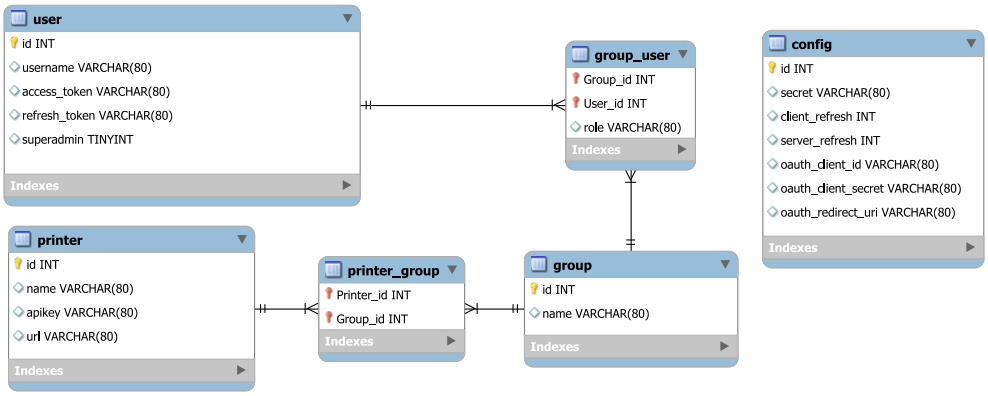
python-zeroconf Je knihovna pro objevování služeb na mDNS [33]. Dále je princip popsán v sekci 3.4.

APScheduler Knihovna, která umožňuje plánovat nebo periodicky spouštět zdrojový kód, plánované spuštění lze ukládat do databáze, aby nedošlo k narušení při restartování APScheduleru. Funkce lze spouštět na pozadí běžící aplikace. Běžící procesy lze spravovat pomocí API knihovny, jedná se hlavně o změnu času spuštění a mazání procesů. [53]

pip Jedná se o Python nástroj pro jednoduché stahování, instalaci a odinstalaci balíčků a knihoven. Pip v základu vyhledává balíčky v repozitáři Python Package Index (PyPI) [54]. Na tento repozitář lze volně registrovat vlastní knihovny a balíčky pro Python. [55]

4.4.2 Model

Zvolený druh databáze je SQLite3. Python podporuje v základu ovladač pro SQLite of verze Python2.5 [56]. SQLite byla vybrána kvůli zjednodušení nasazení aplikace, i přes své nedostatky pro účely webové aplikace. Došlo k závěru, že tyto nedostatky nebudou rizikem jelikož se pro server pro ovládání více 3D tiskáren neočekává velký uživatelský provoz.



Obrázek 4.3: Obrázek zachycující vzhled databázového modelu backendu

Model aplikace tvoří pojmy: Uživatel, Skupina a Tiskárna, které tvoří jednoduchý databázoví model se vztahy Uživatel-Skupina a Skupina-Tiskárna, kde všechny vztahy jsou typu M:N, a vztah Uživatel-Skupina také zahrnuje atribut `role`, což je role uživatele v kontextu skupiny. V databázi se ukládá také tabulka pro konfiguraci, která obsahuje pouze jeden záznam, kde jsou všechna nastavení aplikace, aby mohly být měněna za běhu serveru. Výsledný model je zachycen na obrázku 4.3.

Při spuštění serveru je soubor s databází automaticky vytvořen a schéma společně sním. Databázové schéma je definované přímo v třídách v modelu. Pro získání záznamů z databáze ve formě objektů je použito API `Query` poskytované knihovnou SQLAlchemy.

4.4.3 Přihlášení

Proces přihlášení uživatele ze strany serveru začne posláním požadavku na vystavený koncový bod `/auth` společně s kódem získaným z autorizačního serveru. Kód je vyměněn s autorizačním server za přístupový token a identitu uživatele. Pro komunikaci je posílán klientovo ID a secret kód zakódovaný v hlavičce `Authorization` ve formě `"Basic "+b64encode("ID:SECRET")`, ostatní data jsou poslána v těle požadavku. Uživatel je poté uložen do databáze a je mu vytvořen JWT, který je poslán v HTTP odpovědi.

Pro kontrolu přístupu jsou v aplikaci zavedeny 2 dekorátory. Dekorátory umožňují dělat transformace funkcí. [57] V tomto případě jsou použity pro obalení funkce vykonávané koncovým bodem, tak aby se funkcionalita v dekorátoru vykonala před ní. Důvodem tohoto stylu implementace je časté užití v zdrojovém kódu.

login_required Zkontroluje, zda-li je v hlavičce `Authorization` JWT a validuje ho. Uživatele vyhledá v databázi a nastaví ho do globální kontextové proměnné v rámci požadavku, aby se sním dalo pracovat dále ve

vlastní funkci koncového bodu. Pokud uživatel není přihlášen nebo token není validní, odpoví dekorátor, před vykonáním samotného controlleru.

superadmin_required Má stejnou funkcionalitu, ale kontroluje zda-li uživatel má roli superadmin

4.4.4 Rozhraní OctoPrintu

Pro komunikaci s OctoPrintem byla použita knihovna `octoclient`, která byla v rámci vývoje softwaru rozšířena [23]. Tato knihovna využívá REST API požadavků za pomoci předdefinovaných funkcí pro jednotlivé koncové body z dokumentace OctoPrintu, umí také využít metody `xhr_streaming` v rámci SockJS, pro streamování dat z OctoPrintu v reálném čase. Dle [21] je SockJS API nestabilní, dalším problémem této metody je posílání velkého množství dat každých 500 ms, což při velkém množství tiskáren mohlo nevýkonný server zatížit a celkově komunikaci zpomalit, proto bylo použito periodické volání REST API OctoPrintu s nastavitelnou dobou mezi požadavky. Služba APScheduler má pro každou tiskárnu zaregistrované periodické volání funkce, která získá informace o stavu tiskárny a průběhu tisku. Tyto informace jsou uloženy do třídní proměnné služby `Printer`, která slouží zároveň jako Data Mapper Object pro SQLAlchemy.

4.4.5 REST API

Aplikace poskytuje nestandardní REST API, uzpůsobené přímo potřebám frontendové aplikace za účelem zefektivnění a zrychlení komunikace a snížení nároků na server. Za pomoci REST API jsou pokryty všechny funkcionální požadavky ze zadání. Pro získání informací o tiskárnách v reálném čase na frontendu ze serveru je také použito periodických HTTP požadavků REST API ze stejného důvodu, jako pro rozhraní mezi OctoPrintem a serverem aplikace a pro zjištění následků použití architektury, která není určena pro streamování dat.

ClientConfigApi Slouží pro poskytnutí konfiguračních hodnot, jako je klientovo ID a přesměrovací adresa pro OAuth2 a perioda požadavků na stav tiskáren. Nevyžaduje žádnou úroveň přihlášení.

ConfigApi Tento koncový bod umožňuje s přístupem superadmina, získat a měnit aktuální konfiguraci aplikace. Jedná se o změnu periody obnovení stavu tiskáren na serveru i klientovi a přístupu na OAuth2 autorizační server FIT ČVUT. Pokud dojde ke změně periody obnovení stavu tiskáren na serveru, všechny vlákna, které vyvolávají funkci pro získání dat, jsou přeplánována na novou hodnotu.

FileApi Umožňuje POST požadavkem nahrát soubor na tiskárny definované dle ID s možností započetí tisku.

FileIdApi Koncový bod pro získání informací o souborech na tiskárně, mazání souborů na tiskárně a přeposlání souboru z jedná tiskárny na cílovou množinu tiskáren. Při přeposlání souborů je nejdříve soubor stažen ze zdrojové tiskárny na server do paměti a poslán na cílové tiskárny.

GroupApi GroupApi je pro získání uživateli dostupných skupin, případně skupin ke kterým má uživatel roli admin. Superadmin má automaticky roli admin ke každé skupině. Díky tomuto API lze navíc přidávat nové skupiny, pokud je uživatel superadmin a mazat skupiny, pokud uživatel má ke skupině roli administrátora.

GroupSettingsApi Slouží pro získání nastavení skupiny, jedná se hlavně o seznam tiskáren a uživatelů skupiny. Další funkcionalitou je úprava nastavení skupiny. K oběma akcím je zapotřebí, aby uživatel měl roli administrátora ke skupině.

LocalOctoPrintServiceApi Toto API slouží pro získání tiskáren detekovaných na mDNS pomocí Zeroconf protokolu, které se nenachází v databázi. Na toto API má přístup pouze superadmin.

PrinterApi Účelem tohoto koncového bodu je získání tiskáren, ke kterým má uživatel roli admin, hromadné mazání a hromadné přidávání tiskáren. Součástí přidávání tiskáren je validace, která uživatele sdělí, zda-li je tiskárna s danými údaji serverem dosažitelná. Přidávat, validovat a mazat tiskárny může pouze superadmin. Po přidání nové tiskárny, je tiskárna zaregistrována do třídy `Scheduler`, která se stará o periodické požadavky na status tiskáren.

PrinterIdApi Umožňuje úpravu názvu a přístupových údajů tiskárny, jako je URL a klíč k API. Po změně přístupových údajů je odstraněno vlákno, které se staralo o získání stavu tiskárny a je vytvořeno nové s aktualizovanými daty.

PrinterSettingsApi PrinterSettingsApi je pro hromadné získání nastavení instance OctoPrintu tiskárny, konkrétně se jedná o teplotní profily pro jednotlivé materiály. Pokud je uživatel superadmin, součástí těchto dat jsou přístupové informace k tiskárně URL a klíč k API. Pomocí POST požadavku lze teplotní profily upravovat, pokud má uživatel administrátorské právo alespoň k jedné skupině, která vlastní tiskárnu.

PrinterStatusApi Metodou GET lze získat aktuální stav na serveru všech uživateli přístupných tiskáren. Jedná se o data o teplotách, stav, jméno tiskárny, příslušnost do skupin a průběh tisku. Metodou POST lze provádět hromadné příkazy tiskárnám. Tyto příkazy jsou nastavení teplot, pozastavení, pokračování a přerušení tisku.

SuperAdminApi Na tento koncový bod má přístup pouze uživatel s rolí superadmin a slouží k přidání role superadmin jinému uživateli.

UserApi Toto API slouží pro získání přihlašovacích jmen všech uživatelů v databázi a je používáno pro napovídání při přidávání uživatelů do skupin na frontendu.

4.4.6 Zeroconf

Pro detekci tiskáren v síti publikovaných za pomoci mDNS je použita knihovna `python-zeroconf`. Při spuštění serveru se na pozadí zapne vyhledávač služeb, který nové služby typu `_octoprint._tcp.local.` registruje do své třídní proměnné a zpřístupňuje je tak pro aplikaci. Tyto data jsou poté využita na koncovém bodě `LocalOctoPrintServiceApi` REST API, kde jsou vynechány již registrované tiskárny a odeslány adresy možných tiskáren v lokální síti serveru.

Při spuštění serveru je vytvořena třída `ZeroconfBrowser`, na kterou je zavolána metoda `start`. Tím se vytvoří prohlížeč mDNS, kterému je zaregistrována funkce, která se vykoná při jakékoliv změna služby na mDNS. Prohlížeč mDNS je nastaven na vyhledávání služeb typu `_octoprint._tcp.local.` Registrovaná funkce při přidání služby na mDNS zapíše tuto tiskárnu do své instanční proměnné společně s jejími informacemi. Jedná se hlavně o IP adresu, doménové jméno a název tiskárny.

Tyto data jsou poté dostupné superadminovi při snaze přidat do aplikace nové tiskárny. Zobrazí se mu ve formuláři seznam tiskáren, které nebyly do aplikace přidány, se svým doménovým jménem a případně jménem tiskárny, pokud bylo nastaveno v instanci OctoPrintu. Uživatel musí stále znát API klíč jednotlivých tiskáren, aby je bylo možné do aplikace přidat.

4.5 Monitorování v reálném čase

Aby uživatel znal stav jemu dostupných tiskáren a zároveň měl přístup pouze k tiskárnám, ke kterým mu byl dán přístup systémem skupin dle modelu, který je popsán v sekci 4.4.2, musejí data cestovat z tiskáren přes backend na frontend. Pro obě rozhraní bylo použito periodicky opakovaných požadavků na REST API. Počet sekund mezi uskutečněním jednotlivých požadavků je nastavitelný v backendu aplikace pro obě rozhraní zvlášť. Přímé získání dat

z tiskárny na frontend bez serveru jako prostředníka bylo vyloučeno, jelikož tiskárna, která je v lokální síti serveru, nemusí být klientovy přístupná.

Problém, který se naskytl při zvolení této architektury, je nestabilní doba odezvy, pokud uživatel na frontendu provede nějakou akci s tiskárnou. Pokud je doba opakování volání zvolena na příliš vysokou hodnotu, může uživatel dostat správnou odezvu ihned, ale existuje riziko, že uživatel bude čekat několik sekund, než se mu zobrazí správná hodnota. Možným řešením tohoto problému by bylo nastavení doby mezi požadavky na server i na tiskárny na nízkou hodnotu jako například 1 sekundu. Tento způsob přináší problém zvýšení zátěže pro server, jelikož musí vytvářet mnoho nových připojení při HTTP požadavcích.

Pro vyřešení problému dlouhé odezvy bylo použito metody, kdy frontend předvídá odezvu tiskárny a aniž by znal správná data, zobrazí uživateli odezvu, kterou očekává. S tímto řešením se naskytl problém, kde po zobrazení správné hodnoty ihned po vykonání akce, je následně vytvořen požadavek na stav tiskáren, ale data na serveru stále nebyla aktualizována na novou hodnotou po již vykonané akci uživatelem a na frontendu se data přepíší na chybnou hodnotu.

Lepším řešením monitorování tiskáren v reálném čase by bylo použít nestabilního SockJS API OctoPrintu dle [21] pro získání dat na serveru a Socket.IO knihovny pro přeposílání dat z backendu na frontend přes soketovou komunikaci, kde připojení zůstává otevřené, server není tolik zatěžován a data mohou být posílána častěji.

4.6 Testování použitelnosti

Testování použitelnosti je metoda testování aplikace. Uživatelé z cílové skupiny testují aplikaci formou předdefinovaného scénáře dle cílů uživatele a je pozorováno, zda-li uživatel nemá potíže při práce, či vykonání úkolů a jestli uživatel ví, kde se nachází a k čemu prvky slouží. [58]

Testování proběhlo na 5 lidech, kde 3 neměli žádnou zkušenost s 3D tiskem, OctoPrintem ani testováním uživatelského rozhraní. Jeden uživatel měl zkušenost s 3D tiskem i testováním User Interface (UI) a jeden měl zkušenost s 3D tiskem, OctoPrintem i testováním a byl to budoucí uživatel na administrátorské úrovni. Zadané úkoly byly: přihlášení, předehřátí tiskárny, výtisk modelu ze souboru, přidání nové tiskárny do aplikace a přidání tiskárny do skupiny.

Hlavními vyřešenými problémy nalezenými těmito testujícími uživateli byly neoznačené prvky jako štítky pro nastavení teplot, nadpisy pro seznamy skupin a tiskáren, nepřihlášenému uživateli se zobrazily prvky, se kterými mohl manipulovat, ale nevykonávali žádnou akci, takže docházelo k zmatení uživatele, označení vybraných tiskáren bylo původně formou zaškrtávacího políčka bez popisku což bylo změněno na zvýraznění celého prvku tiskárny změnou

barvy rámečku, reakce na tyto změny byla kladná ze strany uživatelů. Dále byly na vyžádání uživateli přidány nápovědy k ovládacím prvkům a zašedění nepoužitelných prvků, například příkazu tisknutí během tisku. Prvky jako pozastavení, zastavení tisku, nastavení teplot, tiskárny, skupin, mazání byly nahrazeny odpovídajícími ikonami pro větší přehlednost.

Při použití aplikace zobrazené na projektoru, bylo odhaleno, že tmavé pozadí s bílým textem promítané na bílou zeď je pro uživatele nečitelné. Jako reakce na tento problém, byl do frontendu přidán přepínač mezi tmavým a světlým tématem, pro případ užití aplikace se zobrazením na projektoru.

4.7 Nasazení

Kvůli jednoduchému nasazení na jakoukoliv platformu podporující Python, byla aplikace zaregistrována na repozitář PyPI. Aplikace by měla být spuštěna ve virtuální prostředí. Pro to se používá Python modul `venv`, který slouží pro vytvoření odděleného virtuální prostředí pro Python. Pro nasazení nejnovější distribuce aplikace je nutné spustit z příkazové řádky příkazy uvedené v bloku se zdrojovým kódem 4.

```
# vytvoří složku env s virtuálním prostředím
python -m venv env
# aktivuje v příkazovém řádku platformy Linux virtuální prostředí
. env/bin/activate
# stáhne balíček aplikace do virtuálního prostředí
pip install octoprint_dahboard
# nastaví proměnou prostředí na platformě Linux pro Flask
export FLASK_APP=octoprint_dashboard
# příkaz pro vytvoření nebo úpravu konfigurace aplikace
python -m flask config
# přidá do aplikace uživatele s rolí superadmin
python -m flask add_superadmin USERNAME
# spustí server s argumenty pro definici hostitele a portu
python -m flask run [--host=] [--port=]
```

Ukázka kódu 4: Sada příkazů pro stažení, spuštění a konfiguraci aplikace ve virtuálním prostředí

Závěr

Cílem této práce bylo vytvořit webovou aplikaci pro správu více instancí aplikace OctoPrint. Aplikace měla být vytvořena tak, aby byla nenáročná na hardware a jednoduchá na spuštění. Měla býti vytvořena klientská a serverová část aplikace (tedy frontend a backend), kde uživatel může po přihlášení ovládat tiskárny, ke kterým má práva. V případě administrátora může tiskárny i skupiny nastavovat a dávat běžným uživatelům přístup k tiskárnám.

Všechny funkční požadavky pro ovládání a monitorování tiskáren byly splněny, vytvořením frontendu v JavaScript frameworku AngularJS a backendu v Python frameworku Flask, aplikace spolu komunikují skrze rozhraní REST API. Server využívá REST API instancí OctoPrintu. REST API bylo použito i pro získání dat potřebných v reálném čase a z pozorování bylo zjištěno proč je toto řešení nevhodné. Funkcionalita automatické detekce tiskáren v lokální síti byla přidána navíc k požadavkům a umožňuje rychle přidat všechny dostupné tiskárny v síti.

V budoucnu se vývoj aplikace bude zaměřovat hlavně na změnu rozhraní pro monitorování v reálném čase. Bude přidána možnost konfigurace přihlášení pro jiné metody, jiné autorizační servery a možnost bez řízení přístupu. Aplikace je vydána pod licencí MIT, může se proto na vývoji aplikace podílet každý. Jelikož je aplikace zadána pro účely laboratoře 3D tisku na FIT ČVUT, předpokládá se, že se členové laboratoře zapojí do aktivního vývoje aplikace.

Literatura

[1] Häußge, G.: *OctoPrint*. [online]. [cit. 2017-05-20]. Dostupné z: http://octoprint.org/

[2] Häußge, G.: Multiple printers #113. *Github*, [online diskuse]. [cit. 2017-05-20]. Dostupné z: https://github.com/foosel/OctoPrint/issues/113

[3] Häußge, G.: *OctoPrint's documentation*. [online]. [cit. 2017-05-20]. Dostupné z: http://docs.octoprint.org/en/master/index.html

[4] Cummins, K.: The rise of additive manufacturing. *The Engineer*, Květen 2010, [online]. [cit. 2017-05-24]. Dostupné z: https://www.theengineer.co.uk/issues/24-may-2010/the-rise-of-additive-manufacturing/

[5] Palermo, E.: Fused Deposition Modeling: Most Common 3D Printing Method. *LiveScience*, Září 2013, [online]. [cit. 2017-05-24]. Dostupné z: http://www.livescience.com/39810-fused-deposition-modeling.html

[6] Jones, R.; Haufe, P.; Sells, E.; aj.: RepRap – the replicating rapid prototyper. *Robotica*, ročník 29, č. 1, 2011: str. 177–191, doi:10.1017/S026357471000069X, [cit. 2017-05-30].

[7] Mozdřeň, K.: *G-Kód (G-code)*. [online]. [cit. 2017-06-20]. Dostupné z: http://homel.vsb.cz/~moz017/G-code/

[8] Čevela, L.: Raspberry Pi 3: Raspbian na desktopu. *LinuxEXPRES*, ISSN 1801-3996, [online]. [cit. 2017-06-20]. Dostupné z: https://www.linuxexpres.cz/hardware/raspberry-pi-3-raspbian-na-desktopu

[9] Podila, P.: *HTTP: The Protocol Every Web Developer Must Know*. Duben 2013, [online]. [cit. 2017-05-31]. Dostupné z: https://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177

[10] Stox, P.: Why Everyone Should Be Moving To HTTP/2. *Search Engine Land*, Listopad 2015, [online]. [cit. 2017-06-02]. Dostupné z: http://searchengineland.com/everyone-moving-http2-236716

[11] Rocheleau, J.: The Basics of REST and RESTful API Development. *Hongkiat*, [online]. [cit. 2017-06-02]. Dostupné z: http://www.hongkiat.com/blog/rest-restful-api-dev/

[12] *GraphQL | A query language for your API.* [online]. [cit. 2017-06-03]. Dostupné z: http://graphql.org/

[13] Sturgeon, P.: *GraphQL vs REST: Overview.* Leden 2017, [online]. [cit. 2017-06-03]. Dostupné z: https://philsturgeon.uk/api/2017/01/24/graphql-vs-rest-overview/

[14] *About HTML5 WebSocket.* [online]. [cit. 2017-06-05]. Dostupné z: https://www.websocket.org/aboutwebsocket.html

[15] *What is a Socket?* [online] [cit. 2017-06-20]. Dostupné z: https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm

[16] Lubbers, P.; Greco, F.: *HTML5 WebSocket: A Quantum Leap in Scalability for the Web.* [online]. [cit. 2017-05-31]. Dostupné z: https://www.websocket.org/quantum.html

[17] Kelleher, F.: Understanding Socket.IO. *Nodesource*, Srpen 2014, [online]. [cit. 2017-06-05]. Dostupné z: https://nodesource.com/blog/understanding-socketio/

[18] Hanson, J.: What is HTTP Long Polling? *PubNub*, Prosinec 2014, [online]. [cit. 2017-06-05]. Dostupné z: https://www.pubnub.com/blog/2014-12-01-http-long-polling/

[19] Lövdahl, S.: *SockJS - WebSocket emulation done right.* Duben 2016, [online]. [cit. 2017-06-05]. Dostupné z: https://github.com/sockjs/sockjs-client/wiki/%5BArticle%5D-SockJS:-WebSocket-emulation-done-right

[20] *AJAX - XMLHttpRequest.* [online]. [cit. 2017-06-26]. Dostupné z: https://www.tutorialspoint.com/ajax/what_is_xmlhttprequest.htm

[21] Häußge, G.: *REST API - OctoPrint's documentation.* [online]. [cit. 2017-06-20]. Dostupné z: http://docs.octoprint.org/en/master/api/index.html

[22] Häußge, G.: *octoprint_client - foosel/OctoPrint.* [online]. [cit. 2017-06-20]. Dostupné z: https://github.com/foosel/OctoPrint/tree/master/src/octoprint_client

[23] Hrončok, M.; Makarius, J.: *Python client library for OctoPrint REST API.* [online]. [cit. 2017-06-20]. Dostupné z: https://github.com/hroncok/octoclient

[24] Doležal, J.: *iOS aplikace k ovládání 3D tiskáren.* Bakalářská práce, České vysoké učení technické, Fakulta informačních technologií, Praha, 2017, [cit. 2017-06-20].

[25] *What is docker?* [online]. [cit. 2017-06-06]. Dostupné z: https://www.docker.com/what-docker

[26] Jirůtka, J.: *OAuth 2.0.* Leden 2017, [online]. [cit. 2017-06-07]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/oauth2

[27] Anicas, M.: An Introduction to OAuth 2. *DigitalOcean*, Červenec 2014, [online]. [cit. 2017-06-07]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[28] *APPS Manager Beta.* [online]. [cit. 2017-06-08]. Dostupné z: https://auth.fit.cvut.cz/manager/app-types.xhtml

[29] Jirutka, J.: *Obtain an Access Token - cvut/zuul-oaas.* [online]. [cit. 2017-06-20]. Dostupné z: https://github.com/cvut/zuul-oaas/wiki/Authorization-Code-Grant#obtain-an-access-token

[30] Tkalec, T.: JSON Web Token Tutorial: An Example in Laravel and AngularJS. *Toptal*, [online]. [cit. 2017-06-09]. Dostupné z: https://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs

[31] Häußge, G.: *Discovery Plugin.* [online]. [cit. 2017-06-08]. Dostupné z: http://docs.octoprint.org/en/master/bundledplugins/discovery.html

[32] MacDonald, A.: *How Zero-Conf Works.* Červen 2013, [online]. [cit. 2017-06-09]. Dostupné z: https://angus.nyc/2013/zero-conf-bootstrapping-the-network-layer/

[33] Stasiak, J.: *A pure python implementation of multicast DNS service discovery - jstasiak/python-zeroconf.* [online]. [cit. 2017-06-20]. Dostupné z: https://github.com/jstasiak/python-zeroconf

[34] Pfeil, M.: What is persistence and why does it matter? *Datastax*, Říjen 2010, [online]. [cit. 2017-06-09]. Dostupné z: http://www.datastax.com/dev/blog/what-persistence-and-why-does-it-matter

[35] SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. *DigitalOcean*, Únor 2014, [online]. [cit. 2017-06-09]. Dostupné z: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[36] *Learn - mariadb.org.* [online]. [cit. 2017-06-20]. Dostupné z: https://mariadb.org/learn/

[37] What's the Difference Between the Front-End and Back-End? *Pluralsight*, [online]. [cit. 2017-06-10]. Dostupné z: https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end

[38] *How Apiary Works: Fast-track your API Design.* [online]. [cit. 2017-06-12]. Dostupné z: https://apiary.io/how-apiary-works

[39] *MVC Framework - Introduction.* [online]. [cit. 2017-06-10]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[40] *What Is AngularJS?* [online] [cit. 2017-06-10]. Dostupné z: https://docs.angularjs.org/guide/introduction

[41] Danylko, J.: What is AngularJS good for? *Quora*, Únor 2016, [online]. [cit. 2017-06-10]. Dostupné z: https://www.quora.com/What-is-AngularJS-good-for

[42] *AngularJS Material - Introduction.* [online]. [cit. 2017-06-10]. Dostupné z: https://material.angularjs.org/latest/

[43] Clapp, B.: *What is gulp.js and why use it?* Duben 2015, [online]. [cit. 2017-06-10]. Dostupné z: http://brandonclapp.com/what-is-gulp-js-and-why-use-it/

[44] Wanyoike, M.; Dierx, P.: A Beginner's Guide to npm — the Node Package Manager. *Sitepoint*, Červen 2017, [online]. [cit. 2017-06-10]. Dostupné z: https://www.sitepoint.com/beginners-guide-node-package-manager/

[45] *Yeoman Fountain generators.* [online]. [cit. 2017-06-21]. Dostupné z: http://fountainjs.io/

[46] Pękala, K.: *Fully declarative (multi)sortable for AngularJS - kamilkp/angular-sortable-view.* [online]. [cit. 2017-06-21]. Dostupné z: https://github.com/kamilkp/angular-sortable-view

[47] Yalkabov, S.: *Token-based AngularJS Authentication - sahat/satellizer.* [online]. [cit. 2017-06-21]. Dostupné z: https://github.com/sahat/satellizer

[48] *Start building apps on CTU APIs!* [online]. [cit. 2017-06-21]. Dostupné z: https://auth.fit.cvut.cz/manager/index.xhtml

[49] *Promise - Javascript.* [online]. [cit. 2017-06-10]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript/Reference/Global_Objects/Promise

[50] *Introduction to Flask.* [online]. [cit. 2017-06-12]. Dostupné z: http://pymbook.readthedocs.io/en/latest/flask.html

[51] *SQLAlchemy - The Database Toolkit for Python.* [online]. [cit. 2017-06-12]. Dostupné z: https://www.sqlalchemy.org/

[52] *Data Mapper Pattern.* [online]. [cit. 2017-06-12]. Dostupné z: https://www.js-data.io/v3.0/docs/data-mapper-pattern

[53] Grönholm, A.: *Advanced Python Scheduler.* [online]. [cit. 2017-06-12]. Dostupné z: https://apscheduler.readthedocs.io/en/latest/

[54] *PyPI - the Python Package Index.* [online]. [cit. 2017-06-22]. Dostupné z: https://pypi.python.org/pypi

[55] Common Python Tools: Using virtualenv, Installing with Pip, and Managing Packages. *DigitalOcean*, [online]. [cit. 2017-06-13]. Dostupné z: https://www.digitalocean.com/community/tutorials/common-python-tools-using-virtualenv-installing-with-pip-and-managing-packages

[56] *sqlite3 - DB-API 2.0 interface for SQLite databases.* [online]. [cit. 2017-06-12]. Dostupné z: https://docs.python.org/2/library/sqlite3.html

[57] Smith, K. D.; Jewett, J. J.; Montanaro, S.; aj.: *PEP 318 – Decorators for Functions and Methods.* Červen 2003, [online]. [cit. 2017-06-10]. Dostupné z: https://www.python.org/dev/peps/pep-0318/

[58] Usability testing. *UTest*, [online]. [cit. 2017-06-13]. Dostupné z: https://www.utest.com/articles/usability-testing

Příloha **A**

Seznam použitých zkratek

AGPL Affero General Public License.

API Application Programming Interface.

CSS Cascading Style Sheets.

DHCP Dynamic Host Configuration Protocol.

DNS Domain Name System.

FDM Fused Deposition Modeling.

GPL General Public License.

HMAC SHA-256 Keyed-hash Message Authentication Code Secure Hash Algorithm.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ID Identification.

IP Internet Protocol.

JSON JavaScript Object Notation.

JWT JSON Web Token.

LGPL Lesser General Public License.

mDNS multicast Domain Name System.

MVC Model View Controller.

NPM Node Package Manager.

ORM Object-relational mapping.

PyPI Python Package Index.

REST Representational State Transfer.

RSA Rivest, Shamir, Adleman.

STL Stereolithography.

TCP Transmission Control Protocol.

UI User Interface.

UPnP Universal Plug and Play.

URL Uniform Resource Locator.

XHR XmlHttpRequest.

XML Extensible Markup Language.

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... Návod pro spuštění a instalaci softwaru
- `src` ..... Zdrojový kód aplikace
  - `octoprint_dashboard` ..... Zdrojový kód backendu aplikace
  - `frontned` ..... Zdrojový kód frontendu aplikace
- `thesis`
  - `src` ..... Zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... Text práce ve formátu PDF