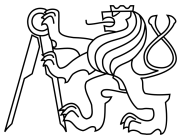
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro obsluhu 3D tiskových úloh
Student:	David Třebický
Vedoucí:	Ing. Marek Žehra
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Tato práce se zabývá automatizací 3D tisku a to primárně na 3D tiskárnách RepRap za účelem zjednodušení jejich použití koncovým uživatelem. Cílem práce je návrh a implementace systému tiskové fronty a distribuce 3D modelů na více cílových zařízení. Systém by měl umět rozhodnout na základě dat získaných analýzou souborů, zda jsou modely validní a podle uživatelem zadaných parametrů následně naplánovat tisk na konkrétní tiskárnu.

Požadavky:
- Proveďte řešerši současných řešení automatizace 3D tisku.
- Navrhněte a implementujte systém pro distribuci tiskové úlohy na cílová zařízení.
- Systém otestujte sadou tiskových úloh na virtuálních zařízeních (3DPrinter emulator, OctoPrint virtual printer, apod.), případně fyzických RepRap zařízeních.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 16. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Systém pro obsluhu 3D tiskových úloh

David Třebický

Vedoucí práce: Ing. Marek Žehra

15. května 2017

Poděkování

Děkuji všem z laboratoře 3D tisku na Fakultě informačních technologií ČVUT, zejména Mirovi Hrončokovi za rady ohledně programování v jazyku Python a Markovi Žehrovi za vedení práce. Dále děkuji Erikovi Švamberkovi, díky kterému jsem přišel k hlavní myšlence této práce.

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 15. května 2017 .....

Odkaz na tuto práci

Třebický, David. *Systém pro obsluhu 3D tiskových úloh*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se soustředí na zjednodušení použití open-source 3D tiskáren RepRap běžnými uživateli bez předchozích zkušeností s 3D tiskem. Literární rešerše detailně rozebírá současné možnosti automatizace 3D tisku a způsoby, jak modely tisknout z pohledu běžného uživatele. Cílem praktické části práce je návrh systému pro automatizované odbavování 3D tiskových úloh a implementace jeho stěžejní součásti – tiskové fronty.

Klíčová slova 3D tisk, automatizace, fronta, Django

Abstract

The thesis concentrates on simplifying the usage of RepRap open-source 3D printers by regular users with no previous experience with 3D printing. A literature research analyzes in detail the current automation capabilities of 3D printing and how to print models from the perspective of an ordinary user. The aim of the practical part is to design a system for automated handling of 3D printing tasks and implementation of its fundamental component – the print queue.

Keywords 3D printing, automatization, queue, Django

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Úvod

3D tisk se aktuálně stává velmi často vyhledávaným technologickým odvětvím, jehož praktické využití je však běžnému uživateli stále ještě vzdáleno či úplně skryto. Obzvlášť v digitálních médiích a ve světě internetu je na toto téma možno nalézt nepřeberné množství videomateriálů a článků, kde se lze přesvědčit, čeho všeho je možné za pomoci 3D tisku dosáhnout a kde všude ho lze využít v běžném životě. Ačkoli je mnoho lidí fascinováno výsledky, bohužel je také často zastáván názor, že používat tuto technologii znamená mít znalosti, které běžný uživatel nemá a vše zůstává určeno jen pro malou skupinu vyvolených.

Samotný 3D tisk, respektive jeho nejdostupnější forma – additivní výroba – funguje na velmi jednoduchém principu – postupně se na podložku nanáší tenké vrstvy hmoty, které vytváří prostorový objekt [48]. Toho lze za pomoci běžně dostupných 3D tiskáren dosáhnout velmi snadno a to bez laboratorních podmínek. Proto je dnes také možné si 3D tiskárnu koupit v obchodě, který ani nemusí být konkrétně specializován na 3D tisk. Přesto vyvstává sada otázek pro naprostou většinu potenciálních zájemců: „Budu umět 3D tiskárnu používat? Vyplatí se mi kupovat si vlastní tiskárnu?“ V mé práci se těmto otázkám budu věnovat, nicméně s velkou pravděpodobností můžeme předpokládat, že běžný uživatel bude odpovídat spíše negativně.

Běžní uživatelé zatím nemají možnost přijít k veřejně přístupné 3D tiskárně a jednoduše si model vytisknout. Často první shledání, pokud vůbec nastane, bývá zprostředkováno někým, kdo již tiskárnu vlastní. Takových lidí ale stále není mnoho a to zejména proto, že si také před zakoupením vlastní tiskárny položili stejné zmíněné otázky a mnoho si jich následně své rozhodnutí rozmyslelo. Já, ačkoli se o 3D tisk již dělší dobu zajímám, sám neznám nikoho (mimo komunitu 3D tisku), kdo by vlastnil svou osobní 3D tiskárnu.

Přestože je potenciální budoucí majitel ochotný naučit se 3D tiskárnu a s ní související znalosti ovládat, setkává se ještě s další překážkou. Tou bývají zejména ceny 3D tiskáren, které jsou v současnosti k dispozici. Ačkoli mohou být ceny opodstatněny náročností výroby a použitých komponent, pohybují se stále v desítkách až stovkách tisíců korun. Taková investice do jednoúčelového zařízení, třebaže s velmi všestranným využitím, je pro běžného uživatele stále příliš vysoká.

Já osobně jsem se k práci s 3D tiskárnami dostal díky fakultě, na které studuji, a přiznávám, že mě od té doby 3D tisk fascinuje ještě mnohem více. Mnoho přátel a známých mi také v posledních několika letech řeklo, že by je zajímalo, „jak vlastně ten 3D tisk funguje?“ Bohužel jsem však neměl žádnou možnost, jak jim proces předvést. Dlouho jsem přemýšlel a snažil se vymyslet způsob, jak 3D tiskárny dostat mezi veřejnost tak, aby je mohl každý přirozeně používat. Došel jsem k závěru, že jediná možnost jak v dnešní době dosáhnout něčeho takového, je zjednodušit celý proces na úroveň, kde bude potřeba jen minimální znalost technologie a zároveň rapidně snížit cenu i při malém objemu tisku.

Cíl práce

Cílem mé práce je zjistit, jaké mají v současnosti běžní uživatelé možnosti 3D tisku a vytyčit zásadní problémy s jeho dostupností. Následně navrhnout systém, který umožní obyčejnému uživateli s minimálními předchozími znalostmi technologií 3D tisku a bez předchozí práce s 3D tiskárnami vytisknout vlastní model s nízkými náklady na výrobu. Dále implementovat funkční koncept řešení, otestovat a připravit strategii nasazení.

Kapitola 1

Rešerše

V této kapitole se budu zabývat technologiemi 3D tisku (dále také jen tisku) a současnými uživatelskými možnostmi, u kterých zjistím jejich výhody a nevýhody. Dále se pokusím nalézt způsoby snížení ceny tisku a zpřístupnění pro běžného uživatele. Nakonec provedu analýzu možností automatizace procesu 3D tisku.

1.1 Technologie

Technologií 3D tisku je mnoho, cenově dostupné pro běžné uživatele jsou však u „osobních“ 3D tiskáren v současnosti primárně dvě. Stereolitografie – angl. *stereolitography* (zkr. SLA) a tavný proces tisku – angl. *fused deposition modeling* (zkr. FDM). První ze jmenovaných využívá k tisku tekuté fotopolymerové náplně. Tyto náplně jsou po dopadu paprsku světla určité vlnové délky vedeného po dané trajektorii po vrstvách vytvrzovány. Druhá technologie využívá zpravidla tuhé materiály – termoplasty, které taví a tryskou nanáší taktéž v jednotlivých vrstvách na podložku. [32]

Obě technologie jsou využívány pro své specifické výhody. Zatímco 3D tiskárny (dále také jen tiskárny) pro tisk technologií FDM jsou většinou levnější na výrobu a jsou velmi rozšířeny v rámci komunity 3D tisku, tiskárny využívající technologii SLA naopak produkují výtisky s mnohem vyšší úrovní detailu a hodí se rovněž díky cenové relaci, ve které se pohybují, spíše pro profesionální využití. [80]

Technologie lze porovnat v tabulce 1.1.

	FDM	**SLA**
Materiál	ABS[1], PLA[2], Nylon a další	Fotosensitivní pryskyřice
Kvalita tisku	nízká až střední	vysoká
Tloušťka vrstvy v mm	0.5 – 0.1	0.05 – 0.015
Povrch	hrubý (schodovitý)	hladký (často lesklý)
Cena materiálů	nízká	střední, ale může být i vysoká
Barvy materiálů	Mnoho barev průhledných i neprůhledných	Méně barev, často průhledných
Podpory při tisknutí převisů	vyžadovány	vyžadovány
Mechanické vlastnosti	Tvrdé nebo pružné, záleží na materiálu	Tvrdé, vznikají nové pružné materiály
Nepropustnost kapalin	Ne, bez úprav nemožné – vznikají malé mezery	Ano, mezery nevznikají
Post-process	Pro dosažení lesku nebo hladkého povrchu je potřeba, taktéž pro vyloučení malých mezer mezi vrstvami	Většinou není potřeba

Tabulka 1.1: Porovnání technologií 3D tisku [80]

O rozšíření 3D tisku mezi širší veřejnost se zasloužila především komunita okolo tiskáren založených na technologii FDM a cenová dostupnost těchto tiskáren. Zmíněná komunita se podílí na mnohých vylepšeních konstrukcí tiskáren, softwarových projektech a knihovnách, které usnadňují vývojářům i uživatelům práci s technologiemi 3D tisku.

Tento typ tiskáren – FDM, je rovněž možné nalézt na Fakultě informačních technologií ČVUT – v laboratoři 3D tisku [37], se kterou spolupracuji v rámci své práce. Dále se tedy budu zabývat výhradně tiskem touto technologií, která také výrazně přispěje ke snížení nákladů tisku pro koncové uživatele.

[1] Akrylonitrilbutadienstyren (zkr. ABS) je amorfní termoplastický průmyslový kopolymer, který je odolný vůči mechanickému poškození. [63]

[2] Polyactid Acid (zkr. PLA) je termoplastický polyester, který je získáván z obnovitelných zdrojů (např. z kukuřičného škrobu) a je biologicky odbouratelný. [49]

1.2 Tiskárny

Většina tiskáren, které jsou takzvaně „open-source[1] hardware“ [51], spadají pod projekt RepRap [62], nebo fungují na bázi těchto tiskáren a zpravidla je lze ovládat obdobným způsobem. Projekt RepRap vznikl na Univerzitě v Bathu ve Spojeném království[2]. Jedná se o hardwarové návrhy, ke kterým je volně k dispozici seznam součástek spolu se zdrojovými kódy a návod na sestavení. Kdokoli je tedy může vylepšovat a přispívat tak k vývoji 3D tiskáren. Do projektu jsou v současnosti zapojeny stovky vývojářů a desítky tisíc uživatelů po celém světě. [56] RepRap se také vyznačuje tím, že značnou část tiskárny lze zreplikovat na samotné tiskárně, jednoduše řečeno – lze si vytisknout náhradní díly či vylepšení. Odtud také pochází jméno projektu – **Rep**licating **Rap**id-prototyper. [62]

Výrobci jako jsou například Ultimaker [78], Makerbot [43], nebo český průkopník 3D tisku Josef Průša a jeho společnost Prusa Research [57], využívají návrhů těchto tiskáren, modifikují je a distribuují široké veřejnosti. Mnoho z nich taktéž přispívá k vývoji. Například firma Ultimaker vyvíjí software pro manipulaci s 3D modely a jejich přípravu pro tisk – Cura [77], který je open-source a pro veřejnost tedy volně dostupný.

V tabulce 1.2 je možné porovnat ceny některých FDM 3D tiskáren, které patří podle statistik portálu 3D Hubs [2] pro druhé čtvrtletí roku 2017, dostupných z [1], mezi nejpoužívanější. Ceny jsou platné k 30. 4. 2017 a přibližně přepočítané do CZK následujícími kurzy: 1 USD = 25 CZK, 1 EUR = 27 CZK. V případě zahraničních produktů se ceny v České republice mohou značně lišit od cen uváděných výrobci.

	Cena uvedená výrobcem	**Přibližná cena v CZK**
Ultimaker 2+ [79]	1 895 EUR	51 200 CZK
Makerbot Replicator+ [42]	2 499 USD	62 500 CZK
Zortrax M200 [83]	1 799 EUR	48 600 CZK
Flashforge Creator Pro [27]	899 USD	22 500 CZK
Prusa i3 MK2S [55]	26 990 CZK	26 990 CZK
Lulzbot TAZ 6 [12]	2 500 USD	62 500 CZK

Tabulka 1.2: Porovnání cen 3D tiskáren

[1] Projekty, které jsou takzvaně *open-source* mají veřejně dostupný zdrojový kód či jiný formát zdroje a lze většinou volně měnit jejich návrh

[2] Spojené království Velké Británie a Severního Irska

1.3 Současní a potenciální uživatelé

Ačkoli je model open-source hardware a software v oblasti 3D tisku velice žádaný, protože přispívá k celkovému vývoji, vyžaduje často znalosti, které nelze od běžných uživatelů očekávat. U současných uživatelů jsou nutné určité teoretické i praktické zkušenosti a schopnost používat softwarové nástroje potřebné k ovladání tiskáren a přípravě modelů. Použití některých tiskáren může být značně zjednodušeno, z vlastní zkušenosti však většinou platí, že čím jednodušší je tiskárnu používat, tím bývá vyšší její cena.

Cílového zákazníka, který by chtěl občasně vytisknout model, tato skutečnost zcela jistě odradí a přispěje tak k obecně rozšířenému názoru, že 3D tisk je jen pro profesionály či nadšence, případně ještě pro movitější vrstvu obyvatel. Potenciálních zájemců přitom zcela jistě není málo. S 3D modely pracuje mnoho oborů, z nichž můžeme jmenovat například architekturu, strojní průmysl, design, herní průmysl a existuje jich jistě mnohem více. Tyto modely by mnozí autoři jistě rádi viděli ve své „fyzické“ podobě. Dalšími zákazníky mohou být lidé, kteří si chtějí vytisknout náhradní díly či objekty, které nejsou sériově vyráběny a nelze je zakoupit.

Mnoho modelů je samozřejmě také k dispozici online a velká část je zdarma ke stažení. Jmenovat můžeme například portál Thingiverse [44], kde je k dispozici nespočetné množství modelů určených výhradně pro tisk na 3D tiskárnách.

1.4 Možnosti 3D tisku

Pro uživatele se nabízí několik možností na výběr. Pokud uživatel plánuje tisknout často či složitější modely a chce se 3D tisku dále věnovat, jako základní možnost se jeví pořízení tiskárny vlastní s tím, že potřebné znalosti se sám naučí z dostupných zdrojů. Jak již bylo ovšem řečeno, tato možnost nepokrývá cílovou skupinu, na kterou se zaměřuje má práce – tou jsou běžní uživatelé, kteří chtějí občas vytisknout model za příznivou cenu.

Vžijeme-li se do role takového běžného uživatele, zbývá nám pouze možnost přenechat tisk někomu jinému. Obvykle ovšem můžeme očekávat, že za takovou službu budeme muset zaplatit, a proto se nyní hodí zmínit možnosti, které máme.

1.4.1 Specializované firmy

Pomocí webových vyhledávačů lze najít specializované firmy, které na zakázku vytisknou námi zaslaný model. Objednávka je často formou poptávky a do-

mluva probíhá po e-mailu či telefonicky. Tyto firmy si účtují vysoké poplatky za samotné zpracování objednávky, přípravu modelu, materiál nebo dobu tisku. I malý model nás tak může vyjít na několik set korun, větší modely se pak pohybují i v řádech tisíců. Porovnání služeb takových firem může být velmi zdlouhavé a následný proces objednávky a tisku v námi vybrané firmě ještě delší. Navíc informace o průběhu zpracování objednávky jsou většinou nedostupné.

Vyzkoušel jsem tři náhodné české firmy, které jsem nalezl pomocí vyhledávače Google [31]. Popíši nyní zkušenosti s nalezenými firmami. Zde z pochopitelných důvodů zachovám jejich anonymitu.

U první firmy trvalo přes tři týdny, než jsem byl telefonicky kontaktován s otázkou, zda je má poptávka aktuální. Taková služba samozřejmě nesplnila má očekávání. U druhé firmy jsem skončil již u čtení ceníku, kde jen manipulační poplatky byly v řádech stokorun. Třetí firma se mi ozvala přibližně týden po zaslání modelu na uvedenou emailovou adresu. Bohužel jsem se dozvěděl, že momentálně nejsou schopni tisknout modely poptávanou kombinací materiálu a barvy (která byla naprosto běžná) a nabídli mi, že se mi ozvou až bude materiál k dispozici. Od té doby již uplynulo několik týdnů a žádnou zprávu jsem zatím nedostal.

Výsledky tedy nedopadly příliš uspokojivě a s každou další takovou firmou bych ztratil srovnatelné množství času, jako s již jmenovanými, a tudíž jsem se rozhodl, že budu hledat jiné možnosti.

1.4.2 3D hub

Jako 3D hub se označuje místo, které je určené k tisku na 3D tiskárnách a zpravidla je buď provozováno jednotlivcem, skupinou, firmou, či nějakou neziskovou organizací. Za úplatu jsou zde nabízeny služby 3D tisku.

Existují také speciální 3D huby, takzvané *makerspaces* [1], kde je možné například výměnou za materiál či dokonce zdarma vytisknout modely, ovšem zde je nutná jistá příslušnost ke komunitě spjaté s 3D tiskem. Jsou tedy určeny převážně nadšencům.

Zbývá placená možnost tisku. K vyhledání 3D hubu v blízkém okolí nám může posloužit online nástroj 3D Hubs [2], který je pravděpodobně největším projektem sdružujícím tato místa.

[1] Slovo *makerspace* vzniklo jako analogie ke slovu *hackerspace*, které označuje místo podobné dílně, kde se schází přiznivci určitých technologií a sdílí mezi sebou zkušenosti či techniku

Zde jsem pocítil mnohem pohodlnější proces objednávky a možnost výběru z obrovského množství poskytovatelů. Velká výhoda je také v tom, že je zvykem zveřejňovat typ tiskárny, na které se model bude tisknout, což nám umožní filtrovat své požadavky a soustředit je jen na 3D huby vyhovující našim kritériím. Je zde mimo jiné možné porovnat cenu za tisk konkrétního modelu a z konkrétního materiálu, což je pro zákazníka naprosto zásadní.

Prověřil jsem, jakým způsobem je cena vypočítávána – mohlo by se totiž stát, že cenu udává objem modelu, který chceme tisknout. To však může být zavádějící, protože modely tištěné na 3D tiskárnách nebývají zcela vyplněné tiskovým materiálem. Často místo toho obsahují zpevňující vzor, který ovšem objemem málokdy přesahuje čtvrtinu celkového objemu vnitřní části, či mohou být modely dokonce duté. Sníží se tak čas tisku i objem spotřebovaného materiálu.

Příklad: Pokud bychom chtěli tisknout krychli o délce hrany 20 cm vyplněnou vzorem odpovídajícím 25 % objemu výplně, rázem bychom potřebovali místo 8000 cm^3 pouze přibližně třetinu.

Na serveru 3D Hubs dochází ke korektním výpočtům, přesto však cena za tisk jednoduchého modelu není nízká. Samotný portál si z každé objednávky bere provizi 12,5 % z celkové ceny. Každý hub může definovat jednorázový poplatek za tisk, který je minimálně 1 euro a cenu za tisk 1 cm^3 materiálu. Zde se jedná o množství materiálu potřebného k vytištění modelu s výplní nastavenou na 25 % celkového objemu a tloušťkou stěn 0,8 mm. V jednorázovém poplatku za tisk se skrývají položky, které se objevovaly i u firem, o kterých jsem psal již dříve.

V tabulce 1.3 můžeme porovnat ceny několika 3D hubů v Praze a okolí doporučených portálem 3D Hubs. Jako testovací modely pro tisk jsem zvolil krychle o délkách hran $\mathbf{a} = 10$ cm, $\mathbf{a} = 5$ cm a $\mathbf{a} = 2{,}5$ cm. U každého hubu byla vybrána nejlevnější možnost tisku. Ceny jsou platné k 30. 4. 2017, byly přepočítány z EUR do CZK kurzem 1 EUR = 27 CZK a zaokrouhleny na celá čísla dolů.

	a = 10 cm	**a = 5 cm**	**a = 2,5 cm**
3dtiskservice.cz's Hub [10]	1722 Kč	376 Kč	270 Kč
CreativeMedia's Hub [5]	1784 Kč	330 Kč	118 Kč
3D obiecto's Hub [9]	1928 Kč	328 Kč	162 Kč
Xcom3Dprint's Hub [7]	2077 Kč	337 Kč	96 Kč
Kinamico's Hub [4]	2320 Kč	371 Kč	101 Kč
atencom3D's Hub [3]	2508 Kč	420 Kč	132 Kč
COTU's Hub [6]	4509 Kč	816 Kč	257 Kč
3Dees's Hub [8]	4936 Kč	830 Kč	262 Kč

Tabulka 1.3: Porovnání cen tisku krychlí o hranách délky **a**

Vidíme, že ceny za tisk jsou velmi vysoké, ačkoli je část procesu již automatizována. Pokud by celková cena reflektovala pouze cenu materiálu a opotřebení tiskárny, byla by jistě mnohonásobně nižší. Pravděpodobně jsou tedy její součástí ještě další položky. Nyní některé z nich identifikuji.

1.4.3 Doplňující služby

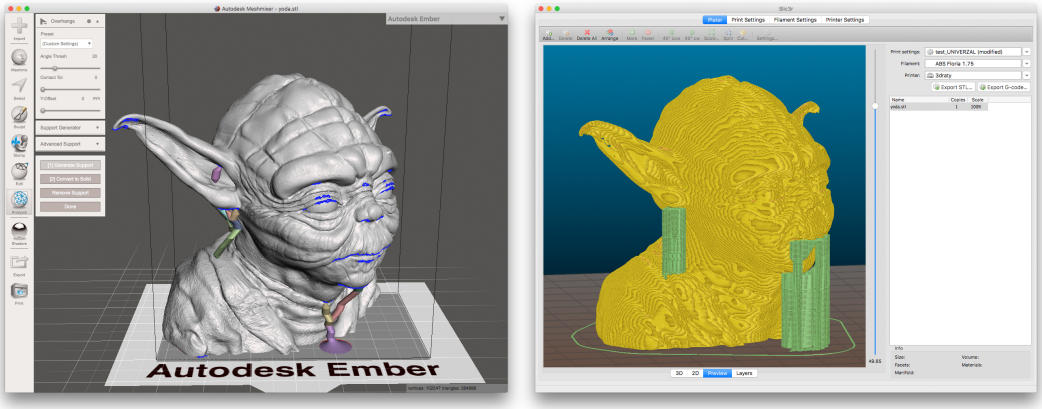
Tiskaři zpravidla nabízí doplňující služby. Mezi ně může patřit například povrchová úprava či barvení. Zde bych však rád zmínil některé pro mou práci stěžejní. Před tiskem je téměř vždy nutná kontrola modelu zkušeným tiskařem, zda je možné tisk provést. V případě negativního výsledku lze někdy využít technik, které i přesto dovolí modely vytisknout. V následující podkapitole popíši problémy, které mohou při tisku nastat.

1.5 Možné problémy s tiskem

Zásadní problém může nastat, pokud chceme tisknout model, který má stěnu tenčí, než je možné fyzicky na konkrétní tiskárně vytisknout. Je velmi pravděpodobné, že taková stěna bude při tisku ignorována. Model by následně mohlo být znemožněno vytisknout, pokud by taková stěna byla nosná.

Další úskalí je nemožnost tisknout převisy pod určitý sklon. Problém se týká tiskáren typu FDM i SLA a dá se řešit jedině vytisknutím podpor, které je možné buďto manuálně vymodelovat, často je však jednodušší je automaticky vygenerovat. Takové podpory samozřejmě přidají materiál navíc, nicméně jsou podmínkou pro správné vytisknutí modelu.

Podpory nám mohou pomoci překonat i první zmíněný problém.



Obrázek 1.1: Porovnání programů generujících podpory

Na obrázku 1.1 vidíme podpory vygenerované programem **Mesmixer** [15] (vlevo) a **Slic3r** (vpravo). Ačkoli Mesmixer dokáže vygenerovat podpory efektivněji, bohužel ho nelze jednoduše automatizovat.

1.6 Snížení ceny tisku

Jak jsme zjistili, 3D tisk není jednoduchý proces a vyžaduje obsluhu profesionálem, který musí umět ovládat velké množství programů pro přípravu modelů k tisku a samotnému ovládání tiskárny. Nutnost odborné obsluhy samozřejmě rapidně navyšuje cenu tisku. Kromě již zmíněného použití levnějších tiskáren typu FDM tedy potřebujeme eliminovat nutné zásahy obsluhy do práce s modely a tiskárnami. V následující podkapitole identifikuji oblasti, které lze automatizovat a které nikoli a uvedu příklady, jakým způsobem toho lze dosáhnout.

1.7 Automatizace procesů tisku

Zjednodušeně lze tisk rozdělit na tyto čtyři fáze:

- získání validního modelu,
- vygenerování tiskové dávky,
- odeslání dávky do tiskárny a zahájení tisku,
- odebrání vytištěného modelu před dalším tiskem.

Ve svém výzkumu jsem došel k závěru, že první tři body je možné uspokojivě automatizovat a to jen prostřednictvím softwarové automatizace. Pokud bychom se snažili automatizovat i poslední, téměř jistě bychom porušili podmínku, kterou je snížení nákladů na tisk. Operace spojené s manipulací s výtisky a tiskovou plochou tedy zůstávají výhradně v režii člověka. Podívejme se nyní detailně, jak je možné jednotlivé oblasti automatizovat.

1.7.1 Získání validního modelu

Modely formátu STL [66] se skládají ze sítě trojúhelníků – takzvaná *mesh*. Vlivem uživatelů nebo programů, které jsou k vytvoření modelu použity, mohou nastat v meshi chyby (například neuzavřená stěna, špatně definovaný normálový vektor). Ty mohou následně zapříčinit nemožnost model vytisknout.

Modely určené pro tisk na 3D tiskárnách bývají připraveny tak, aby byly validní. Nicméně se může stát, že model původně pro tisk určen nebyl, či není validní z jiného důvodu a pokud je to možné, je nutné podstoupit kroky, které jeho validitu zajistí.

Model můžeme opravit sami, což vyžaduje zkušenosti s 3D modelováním a ovládáním příslušného programu. V případě běžného uživatele tento proces musíme přenechat někomu zkušenějšímu. Existují ovšem i programy, které model analyzují, vyhodnotí jeho validitu a případně dokáží i některé chyby automaticky opravit. Takové programy jsou například **Netfabb** [16] či **ADMesh** [45]. **Netfabb**, ačkoli nabízí pokročilé techniky, je bohužel placený software a nelze ho ani jednoduše automatizovat, tudíž se zaměříme na druhý zmiňovaný program, který je zdarma, open-source a v komunitě 3D tisku hojně používán. **ADMesh** je program spouštěný z příkazové řádky, který nemá žádné grafické rozhraní. Je tedy ideální pro automatizaci procesu validace, ačkoli jen omezené.

Příkaz z ukázky 1.1 se pokusí načíst a opravit zdrojový soubor modelu, následně opravený model zapíše do výstupního souboru, v tomto případě do jeho textové formy. Mimo jiné z výpisu můžeme zjistit i informace o modelu, jako jsou jeho rozměry a objem.

```
$ admesh --write-ascii=output.stl yoda.stl

ADMesh version 0.98.2, Copyright (C) 1995, 1996 Anthony D.
↪  Martin
ADMesh comes with NO WARRANTY.  This is free software, and
↪  you are welcome to
redistribute it under certain conditions.  See the file
↪  COPYING for details.
Opening yoda.stl
Checking exact...
All facets connected.  No nearby check necessary.
No unconnected need to be removed.
No holes need to be filled.
Checking normal directions...
Checking normal values...
Calculating volume...
Verifying neighbors...

==== Results produced by ADMesh version 0.98.2 ====
Input file         : yoda.stl
File type          : Binary STL file
Header             : STL File created by netfabb
============== Size ==============
Min X = -39.810001, Max X =  29.764002
Min Y = -29.148001, Max Y =  31.424002
Min Z~=  0.008000, Max Z~=  63.744003
========= Facet Status ========== Original ======== Final
↪  ====
Number of facets                 : 198530           198530
Facets with 1 disconnected edge  :     0              0
Facets with 2 disconnected edges :     0              0
Facets with 3 disconnected edges :     0              0
Total disconnected facets        :     0              0
=== Processing Statistics ===     ===== Other Statistics
↪  =====
Number of parts       :     1      Volume   :  81343.72656
Degenerate facets     :     0
Edges fixed           :     0
Facets removed        :     0
Facets added          :     0
Facets reversed       :     0
Backwards edges       :     0
Normals fixed         :    16
```

Ukázka kódu 1.1: Ukázka výpisu programu ADMesh [45]

1.7.2 Vygenerování tiskové dávky

Každý model je nutné před tiskem převést do tiskové dávky. Pro FDM tiskárny je obvyklý formát GCode [61], který tiskárna dokáže zpracovat. Jedná se o sadu příkazů, které ovládají pozici tiskové hlavy a zda se tiskový materiál vytlačuje či nikoli. Samozřejmě je možné pomocí těchto řídících příkazů docílit i dalších možností, nicméně pro zjednodušení nám bude stačit znát tyto.

Je tedy nutné model převést do formátu GCode. K tomu slouží řezací software, neboli „slicer". Řešení, která jsou zdarma a dají se ovládat z příkazové řádky, jsou vesměs dvě. **Cura** [77] a **Slic3r** [70]. Pro ukázku jsem vybral program **Slic3r**. V ukázce kódu 1.2 je příklad výpisu programu.

```
$ slic3r --support-material --load config.ini --output
↪  output.gcode yoda.stl
=> Processing triangulated mesh
=> Generating perimeters
=> Preparing infill
=> Infilling layers
=> Generating support material
=> Generating skirt
=> Exporting G-code to output.gcode
Done. Process took 0 minutes and 29.176 seconds
Filament required: 5504.2mm (38.9cm3)
```

Ukázka kódu 1.2: Ukázka výpisu programu Slic3r

Na konci výpisu je možné zjistit, kromě úspěšného vygenerování souboru GCode, také délku struny materiálu či celkový objem, který bude potřeba k vytištění modelu při použití nastavení ze souboru `config.ini`. V tomto konfiguračním souboru je pro ukázku nastavena hustota výplně na 10 %, což je ze zkušenosti s 3D tiskem postačující. Objem tiskového souboru u testovacího modelu byl snížen o více než polovinu oproti objemu zjištěného programem **ADMesh**, tedy objemu zdrojového 3D modelu.

Všimněme si také, že program má možnost vygenerovat podpory pro modely s převisy. V ukázce 1.2 je již materiál na tisk podpor zahrnut v kalkulaci.

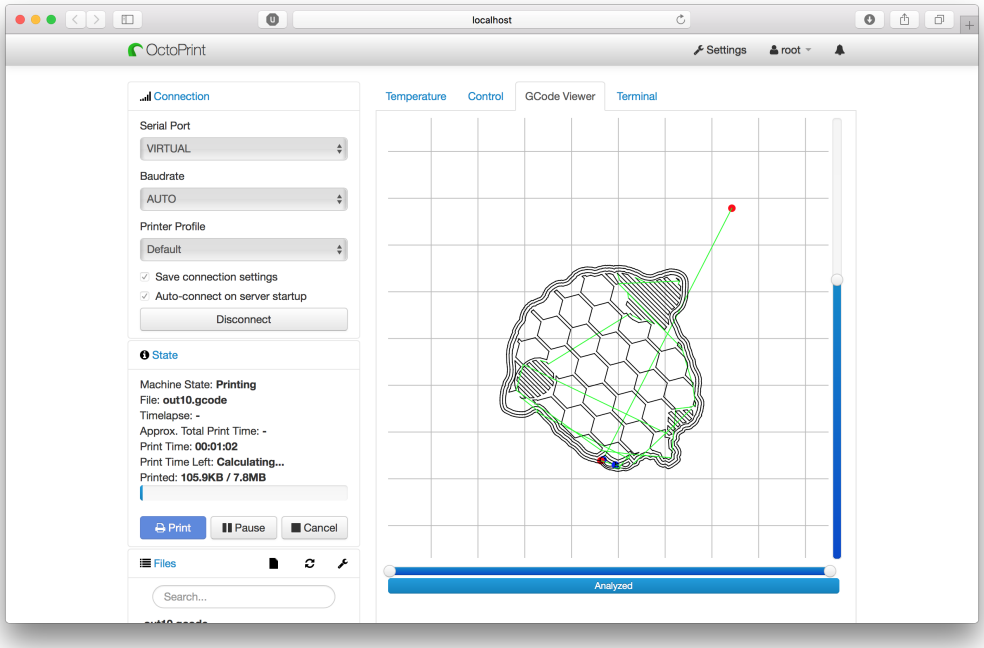
1.7.3 Odeslání dávky do tiskárny a zahájení tisku

Získaný seznam příkazů ve formátu GCode nyní potřebujeme odeslat do tiskárny. Některé tiskárny mají čtečku paměťových karet a ovládací panel s displayem, kde můžeme vybrat konkrétní tiskovou dávku a spustit tisk. Zjednodušuje se tím proces tisku pro běžného uživatele a nahrazuje se tak manipulace

s programy, které by umožnily tisk přímo z počítače.

Tyto programy fungují tak, že po sériové lince do tiskárny odesílají postupně jeden příkaz za druhým a ta je následně provádí. Jeden z takových programů je například **Printrun** [81]. Jeho verze pro použití z příkazové řádky – **Pronsole**, by se dala také automatizovat. Program ovšem musí být stále ve spojení s tiskárnou a je tedy blokující. Z toho plyne, že při více připojených tiskárnách můžeme jednou spuštěnou instancí programu v danou chvíli ovládat pouze jednu tiskránu.

Nyní bych rád zmínil program **OctoPrint** [38], který je možné spustit jako webovou službu na počítači, ke kterému je připojena jedna nebo více tiskáren. Jeho součástí je i grafické rozhraní, ke kterému lze přistupovat přes internetový prohlížeč. Ukázka této aplikace je na obrázku 1.2.



Obrázek 1.2: Webové rozhraní programu OctoPrint

Tato aplikace je v současnosti široce využívána pro ovládání tiskáren. Zároveň podporuje téměř všechny nejpoužívanější FDM tiskárny, jako například zmiňované modely výrobců Makerbot, Ultimaker, Prusa Research a mnoho dalších. Zpravidla je k tiskárně trvale připojen mikropočítač, například Raspberry Pi

[71], který funguje jako server. Uživatel tak může k aplikaci přistupovat i vzdáleně. Aplikace disponuje také API, tedy rozhraním pro ovládání externími programy pomocí předdefinovaných metod. To nám pomůže automatizovat proces tisku.

1.8 Existující nástroje pro automatizaci

V rámci svého výzkumu jsem hledal projekty, které se automatizací procesu 3D tisku zabývají a mají funkční řešení. Našel jsem dvě takové služby a provedl jsem jejich analýzu.

1.8.1 Portál 3D Hubs

Nejrozšířenější projekt, který z části také automatizuje proces 3D tisku, je portál 3D Hubs [2]. Zde dochází k výpočtu ceny tisku, jehož postup byl vysvětlen již dříve, dále k základní kontrole, která prověří, zda-li se jedná o správný formát souboru modelu. Je zde také možnost zvolit si způsob dopravy k zákazníkovi a portál zprostředkovává i platby. Po provedení platby je soubor modelu odeslán do cílového 3D hubu a poté se již o tisk stará obsluha.

Zde bych rád dodal, že automatizována je zde pouze domluva mezi obchodníkem a zákazníkem a stanovení ceny, nikoli samotný proces tisku. Nicméně se jedná o projekt, který značně usnadní cílovému uživateli tisk modelu.

Výhody

- Jednoduché uživatelské rozhraní
- Mezinárodně rozšířená platforma

Nevýhody

- Neřeší automatizaci procesů 3D tisku
- Není open-source

Vyhodnocení

Jelikož projekt není open-source, není možné ho rozšířit o potřebné funkce.

1.8.2 BotQueue.com

Projekt, který se naopak zabývá automatizací celého procesu je open-source řešení BotQueue [34]. Funguje na principu klient-server, kde server ve formě

PHP [74] aplikace zpracovává uživatelské modely, automaticky je dokáže rozřezat a poslat do klientské aplikace, která je přímo propojená s tiskárnami. Zároveň je zde obsažena správa tiskáren či možnost zvolit si software k řezání modelů (na výběr je **Slic3r** [70] a **Cura** [77]).

Postup nasazení

Pokusím se nyní objasnit přesné použití aplikace.

- V první řadě se musíme zaregistrovat na portálu botqueue.com.
- Nyní zaregistrujeme svého bota (tak označuje autor 3D tiskárny), přiřadíme mu frontu a profil pro řezání.
- Nainstalujeme klientský program **bumblebee** [33], který je dostupný z balíčkovacího systému **pip** [58].
- Program vygeneruje odkaz, po jehož navštívení autorizujeme klienta k použití s naším účtem.
- Nyní přidáme úlohu ve formě modelu nebo tiskové dávky.
- Tiskárna, která je připojena k počítači, na kterém běží klient, by nyní měla začít tisknout.
- Po dokončení tisku je nutné potvrdit, zda tisk proběhl v pořádku.

U této aplikace se na první pohled zdá, že splňuje vše, co bylo popsáno v podkapitole 1.7. V minulosti jsem byl schopen autorizovat klientskou aplikaci, ačkoli jsem neměl jak vyzkoušet, zda se tisk opravdu spustí. Bohužel nyní se mi již nepodařilo projít autorizací. Pravděpodobně k tomu došlo, protože projekt není nadále udržován a samotný portál má prošlý HTTPS certifikát. Klientský software tedy nedokáže navázat zabezpečené spojení, a tudíž skončí chybou. Manuálně bohužel nelze klienta do aplikace přidat.

Jako první možnost se jeví opravit chyby v již pokročilém projektu a přidat funkce, které chybí. Nyní proto popíši výhody a nevýhody zmíněné aplikace.

Výhody

- Stanice s klientskou aplikací nemusí mít veřejnou IP adresu.
- Aplikace je open-source.

Nevýhody

- Není možné přerušit či jinak ovlivnit průběh tisku – vše je závislé na klientské aplikaci.
- Tisknout lze jen po jednom modelu.
- Projekt již není udržovaný.
- Podporovaných tiskáren je málo.
- Je nutný zásah do implementace, pokud chceme přidat ovladač pro model tiskárny, který ještě není kompatibilní.

Vyhodnocení

Ačkoli je patrná výhoda modelu klient-server, vše závisí na vývoji klientské aplikace. Ta bohužel nemá příliš mnoho přispěvovatelů a vývoj se zde prakticky zastavil. Zásadní problém je také nemožnost tiskárnu ovládat v průběhu tisku. To může být žádoucí v případě poruchy či chyby v průběhu tisku. Pokud navíc přidáme podporu malého množství tiskáren, nezbývá než posoudit, zda-li se vyplatí takový projekt dále vyvíjet, či se vydat jinou cestou.

Pro požadovanou funkčnost bychom museli přidat minimálně logiku pro ovládání tiskáren. V současnosti je klientský software program spouštěný pouze v příkazové řádce a ovládání by tudíž pro obsluhu bylo velmi složité. Byli bychom nuceni vytvořit API, či přímo grafické rozhraní. Takové API ale již existuje ve zmíněném open-source projektu **OctoPrint** [38], který dokáže tiskárny ovládat a lze tímto způsobem tisk automatizovat. Nevýhoda je, že bychom museli přidat logiku tiskové fronty, která je naopak v klientském programu aplikace BotQueue již implementována.

Po zvážení výhod a nevýhod jsem se rozhodl, že lepší bude upustit od modelu klient-server a navrhnout aplikaci tak, aby existoval odděleně modul tiskové fronty spolu s administrací a řídící moduly tiskáren v podobě programu **OctoPrint**. Budoucí aplikaci tak budeme schopni nasadit do již zaběhnuté infrastruktury využívající tento software.

Kapitola 2

Návrh

V kapitole věnované návrhu definuji, jak by měla aplikace fungovat, jaké jsou na ní funkční a nefunkční požadavky, její cílovou platformu a způsob použití.

2.1 Cílová platforma

Aplikace by měla splňovat architekturu takzvaných mikroslužeb - angl. *micro-services*. Znamená to, že každá služba, ať se jedná o databázi, webový server či například aplikační server, jsou od sebe odděleny a je možné je nezávisle nasadit [28].

V současné době nabývá na oblíbenosti trend virtualizace pomocí kontejnerů a pro naši aplikaci se tato platforma výborně hodí. Cílovou službou na které můžeme komponenty aplikace provozovat se pak může stát jedna z mnoha cloudových služeb. Jmenovat můžeme například Amazon Web Services [14], Google Cloud [30], Docker Cloud [24] či jiné. Jednotlivé služby budou zabaleny do obrazů formátu Docker image [23] (dále také jen image nebo Docker image).

2.2 Funkční požadavky

2.2.1 Automatizace procesů

Efektivní tisk více modelů

Jako jednu ze základních podmínek jsem stanovil možnost tisku více modelů v jedné tiskové úloze. Zajistíme tak efektivitu tisku a snížíme režii obsluhy, která by v případě tisku modelů po jednom musela velmi často tiskárny obsluhovat. Protože předpokládáme zapojení velkého množství tiskáren, minimalizujeme tak nároky na počet lidí, kteří musí tiskárny spravovat. Je tedy stěžejní efektivně kombinovat modely určené k tisku a seskupovat je podle jejich parametrů, kterými mohou být například materiál či barva.

Analýza validity modelu a získání informací o modelu

Model musí být automaticky validovaný, bez nutnosti zásahu obsluhy. Model může být vyhodnocen jako validní, nevalidní ale opravený, či zcela nevalidní. Dále je nutné generovat náhled modelu pro pozdější identifikaci. Součástí automatické analýzy je i výpočet množství materiálu, který bude na tisk modelu spotřebován. Na základě tohoto výpočtu bude možné stanovit cenu tištěného modelu.

Generování tiskové dávky

Z výsledné kombinace modelů musí být automaticky generována tisková dávka v závislosti na konkrétním nastavení cílové tiskárny. Dávka bude generována s podporami pro bezproblémový tisk i jinak nevytisknutelných objektů s převisy. Tuto funkci delegujeme na vybraný řezací software. Programy pro řezání modelů zpravidla nabízejí i generování podpor, jak jsme si předvedli v ukázce 1.2 programu **Slic3r**.

Odeslání dávky do tiskárny a zahájení tisku

Vygenerovaná tiskové dávka bude automaticky distribuována do příslušné volné tiskárny a následně bude zahájen tisk. Taková tiskárna bude označena jako nedostupná, aby nedošlo k odeslání další tiskové úlohy ještě před odejmutím hotových modelů. Obsluha následně manuálně nastaví, že tiskárna je opět k dispozici.

2.2.2 API

Celou aplikaci by mělo být možné ovládat přes HTTP API a to zejména z důvodu variability možností rozšíření systému a napojení na jiné služby.

2.2.3 Grafické rozhraní

Grafické rozhraní u implementace konceptu není vyžadováno. Jde především o funkční automatizační software a grafické rozhraní tedy není pro jeho běh důležité. Samozřejmě pro konkrétní případy užití bude grafické prostředí zcela jistě zapotřebí, to se ale velmi pravděpodobně bude u každého případu užití lišit. Ponecháme tedy implementaci na organizaci, která bude náš software používat.

2.3 Nefunkční požadavky

Zaměříme se také na vnitřní uspořádání, bezpečnost a kvalitu běhu aplikace.

2.3.1 Databáze objektů

Informace o objektech budou ukládány do relační databáze, kterou samozřejmě musíme ochránit standardními postupy, omezujícími následky případné havárie. Z předchozích funkčních požadavků vyplývá, které objekty budeme nutně muset ukládat a spravovat. Nyní je můžeme detailněji popsat.

Tiskárny, nastavení, materiály

Pro správnou funkčnost je potřeba evidovat samotné tiskárny, jejich specifické nastavení a materiály, které jsou k dispozici. Materiály i nastavení budeme přiřazovat konkrétním tiskárnám.

Uživatelé

Ačkoli bychom aplikaci mohli pojmout jako jednouživatelskou, či zcela bez uživatelských účtů, rozhodl jsem se, že lepší bude implementovat víceuživatelský přístup již v této vrstvě. Zajistíme tím tak povolení přístupu do administrační části pouze oprávněným uživatelům, které budeme umět rozlišit od uživatelů běžných. Přenechat správu práv uživatelů organizaci, která bude aplikaci používat, by mohlo vést k bezpečnostním rizikům.

Modely

Samozřejmě je nutné evidovat modely, které budou na tiskárnách tištěny. O každém nahraném modelu budou ukládány veškeré informace o jeho aktuálním stavu, zdrojovém souboru a parametrech tisku. Z informací o modelu bude možné také zjistit, kdo model nahrál.

2.3.2 Škálovatelnost

Jelikož aplikace může silně vytěžovat stroje, na kterých poběží, je nutno zajistit škálovatelnost kritických procesů. Tyto procesy již nyní můžeme identifikovat – jsou to všechny konverze souborů a síťové přenosy velkých objemů dat. Musíme zaručit, aby i při velkém množství uživatelů a tiskáren byla aplikace stále funkční a měla rychlou odezvu. Proto požadujeme delegaci výpočeních procesů na více strojů a možnost jednoduchého rozšíření.

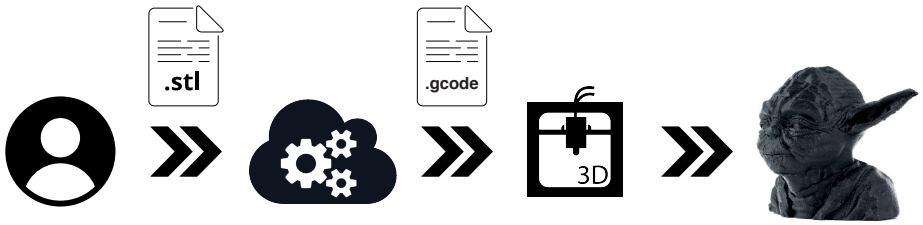
2.3.3 Perzistentní úložiště

Jelikož 3D modely mají často velký datový objem a potřebujeme s nimi často pracovat a uchovávat i jejich historii, je nutné abychom je měli bezpečně uložené na úložišti, které nám zajistí jejich stálou dostupnost. Protože předmětem cílové platformy je cloudová služba, použijeme některé z nabízených perzistentních úložišť. Můžeme jmenovat například Amazon Elastic Block Store [13], Google Persistent Disk [29] či další.

2.3.4 Bezpečnost komunikace

Pro zabezpečení přenosu dat mezi uživateli a aplikací použijeme HTTPS[1] certifikát. HTTPS certifikát taktéž použijeme při komunikaci s tiskárnami na straně aplikace OctoPrint.

2.4 Způsob použití aplikace



Obrázek 2.1: Diagram použití aplikace uživatelem

Voláním koncových bodů API (angl. *API endpoint*) budeme schopni aplikaci ovládat a rozlišit práva uživatelů odesílajících požadavky (prostřednictvím budoucího grafického rozhraní).

Obsluha tiskáren bude mít umožněno spravovat tiskárny, nastavení tiskáren a materiály. Dále bude mít přístup ke všem uživatelským účtům a nahraným modelům, které může v případě potřeby manuálně upravovat či jinak měnit jejich parametry.

Běžní uživatelé se budou moci zaregistrovat a poté nahrávat modely. Po nahrání modelu do aplikace není vyžadována další interakce uživatele, systém by měl sám umět rozhodnout, zda je možné tisk provést a případně model naplánovat k tisku na jedné z tiskáren (viz obrázek 2.1).

Potvrzení, zda budou modely vytisknuty, zůstává v režii obsluhy a v budoucnosti bude možné tento mechanismus rozšířit o jiné typy potvrzení – například pomocí plateb.

[1] HTTP over TLS/SSL – vrstva používaná k šifrování komunikace

Kapitola 3

Implementace

3.1 Použité technologie

3.1.1 Jazyk Python a framework Django

Při výběru vhodného jazyka pro implementaci jsem se rozhodoval mezi jazykem **C++** [19] a jazykem **Python** [60]. C++ je v mnoha ohledech výpočetně rychlejší. Jádro programu **Slic3r** je taktéž naprogramováno v C++ a program **ADMesh** v kompatibilním jazyce C, což by mohlo být výhodné při jejich automatizaci [69][46]. C++ ovšem není ideálním jazykem pro webovou aplikaci, kterou bezesporu naše výsledná aplikace bude.

Vybral jsem tedy jazyk Python. Kromě toho, že s programováním v tomto jazyce mám základní zkušenost, existují další důvody, které jsem bral v úvahu při jeho zvolení. Je často používán v komunitě 3D tisku a existuje například klientská knihovna pro komunikaci s aplikací OctoPrint – **OctoClient** [36] či knihovna programu ADMesh – **python-admesh** [35]. Python je také použit v mnoha projektech automatizujících periodické procesy (např. Ansible [72]) a je často používán ke skriptování. Navíc má jednoduchou a čitelnou syntaxi a je výborně zdokumentován. Další výhodou je, že pro Python existuje nespočet webových frameworků, které značně usnadňují implementaci webových aplikací.

Python je možné nainstalovat mnoha způsoby, většina linuxových distribucí ho již obsahuje, či lze jednoduše doinstalovat příslušným balíčkovacím manažerem. Pro aplikaci je zvolena (v té době aktuální) verze **3.5**.

Mezi mnoha webovými frameworky, které jsou pro jazyk Python k dispozici, jsem vybral dle mého názoru a z vlastní zkušenosti pro tento účel nejvhodnější – **Django** [21]. Framework Django přispěje k efektivnímu převedení návrhu do funkční formy.

Django je webový framework s architekturou *Model-View-Controller* [64] (zkr. MVC), což v praxi znamená oddělení datového modelu aplikace, uživatelského rozhraní a řídící logiky do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní [64]. Využiji ho mimo jiné k ukládání datových struktur, k operácím nad objekty, ke zpracování HTTP požadavků a k autentizaci. Framework Django nám také z velké části zajistí bezpečnost aplikace díky vestavěným kontrolám uživatelských vstupů.

3.1.2 Další použité knihovny

V projektu jsem použil mnoho doplňků a nyní bych rád zmínil nejdůležitější z nich. Jejich kompletní seznam je možno nalézt v souboru `requirements.txt`, který je umístěn v adresáři se zdrojovými kódy v příloze.

Django REST framework

Django REST framework [18] je nadstavba frameworku Django, která umožňuje jednoduše vytvořit komplexní rozhraní HTTP API, se kterým budou uživatelé (prostřednictvím grafického rozhraní) komunikovat. Hlavní výhodou je možnost využití potenciálu objektového modelu, který nabízí framework Django.

Gunicorn

Gunicorn [17], neboli Green Unicorn, je webový HTTP server naprogramovaný v Pythonu. Jeho funkci blíže popíši v podkapitole nasazení.

RQ

RQ [25], neboli Redis Queue je implementace fronty pro ukládání úloh, které jsou delegovány pracovním procesům - angl. worker processes. Využívám tuto knihovnu pro asynchronní výpočty, které mohou trvat dlouho a omezily by tím běh hlavní aplikace. Nutnou podmínkou pro funkčnost této knihovny je databáze Redis.

RQ Scheduler

RQ Scheduler [50] je nadstavba knihovny RQ. Jak je z názvu patrné, jedná se o plánovač. Využívám ho pro periodické plánování kontrol tiskáren či dalších periodických procesů, které později detailně popíši.

3.1.3 Programy určené k automatizaci

Aplikace bude automatizovat několik programů, z nichž některé jsem již dříve zmínil a další teprve představím. Všechny jsou open-source a zdarma, což jsem kladl jako jednu ze základních podmínek.

- **ADMesh** [45] – program pro validaci modelů a jejich opravy,

- **Slic3r** [70] – program pro generování podpor a převod modelů na tiskové dávky,

- **simarrange** [82] – program umožňující efektivní umístění více modelů na podložku s definovanými rozměry,

```
simarrange --height=200 --width=200 --outputdir=combined/
↪  --limit=1 yoda{1..7}.stl
Generating plate 0
File: yoda1.stl minx: 111, miny: 36, minrot: 0
File: yoda2.stl minx: 151, miny: 91, minrot: 0
File: yoda3.stl minx: 46, miny: 96, minrot: 0
File: yoda4.stl minx: 101, miny: 141, minrot: 30
File: yoda5.stl minx: 36, miny: 161, minrot: 160
SKIP: yoda6.stl skipped for this plate
SKIP: yoda7.stl skipped for this plate
Stopping at limit of plate 1
```

Ukázka kódu 3.1: Ukázka výpisu programu simarrange

- **OpenSCAD** [40] – program primárně určený pro parametrické modelování 3D modelů, který také umožňuje generovat náhledy. Výstupní soubor příkazu `openscad -o image.png -render -imgsize=640,640 yoda.scad` je možné vidět na obrázku 3.1.

Obrázek 3.1: Výstupní soubor programu OpenSCAD

3.2 Služby

Nyní popíši služby, které je nutné spustit pro běh aplikace.

3.2.1 PostgreSQL

U frameworku Django je možné si vybrat mezi relačními databázemi MySQL [52] a PostgreSQL [75], které jsou použity pro ukládání informací o objektech. Vybral jsem databázi PostgreSQL, protože je doporučována autory framewroku. Ve výsledku by nemělo být důležité, jaká technologie je z programátorského pohledu zvolena, protože o manipulaci s daty v databázi se výhradně stará framework Django, a tudíž odpadá nutnost manuálních zásahů do definice databáze.

3.2.2 Redis

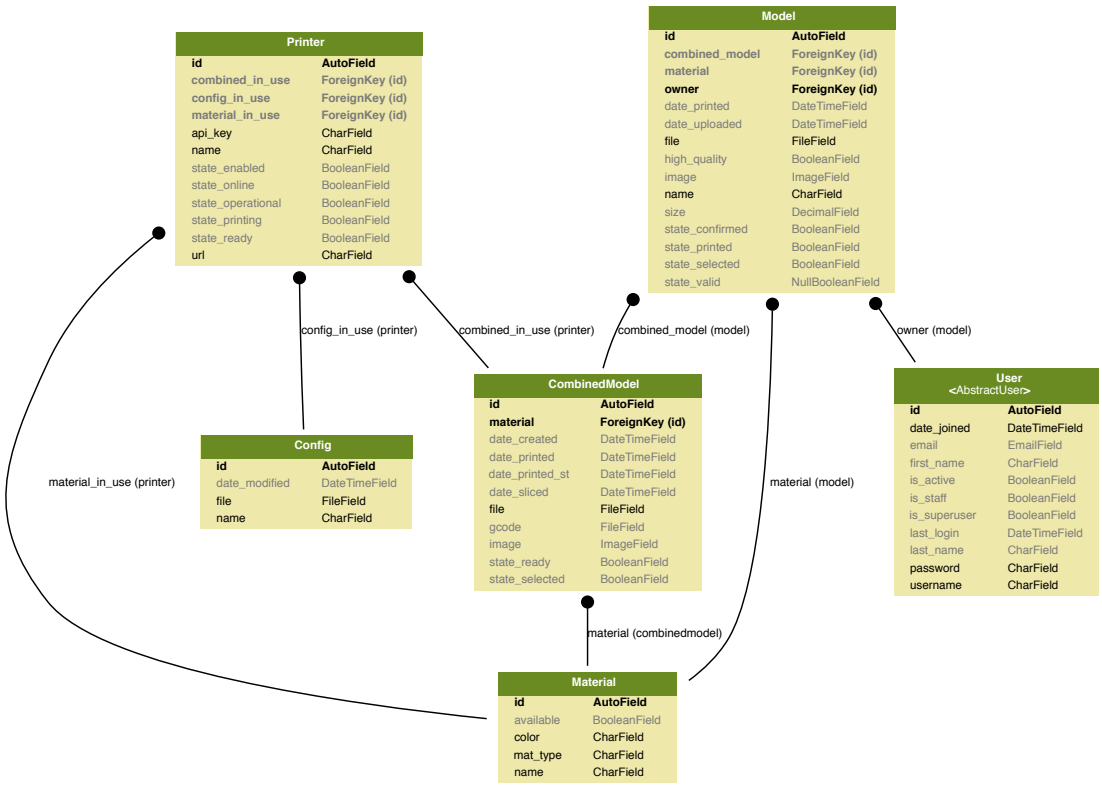
Redis [73] je NoSQL databáze, což znamená, že obsahuje pouze data typu „klíč-hodnota“ [65]. Tato databáze je použita pro plánování úloh a jejich delegování na pracovní procesy. V budoucnosti by mohla být také využita pro ukládání uživatelských sezení - angl. *sessions* a pro kešování obsahu – angl. *caching*[1].

[1] Ukládání dat, ke kterým je často přistupováno, do mezipaměti z důvodu šetření výpočetními prostředky

3.3 Architektura aplikace

3.3.1 Model tříd

V implementaci využívám objektově orientovaného stylu programování. Framework Django v tomto ohledu značně urychluje proces implementace. Mnoho metod pro manipulaci s objekty je totiž možné dědit přímo od tříd definovaných v samotném frameworku. Tyto třídy se nazývají *Django Modely*. Django se zároveň stará o objektově relační mapování, což prakticky znamená, že datový model odpovídá modelu tříd. Instance objektů aplikace je tedy možné jednoduše ukládat do databáze a zase je z ní načítat.



Obrázek 3.2: Model tříd

Třída User

Třída `User` je již definována v Django frameworku a využívám zde její implicitní implementace. Při vytváření objektů této třídy je nutné nastavit atributy

`username` a `password`. Django kontroluje, zda jsou v databázi unikátní uživatelská jména a zda-li jsou hesla dostatečně silná.

Třída Material

Tuto třídu jsem vytvořil, aby bylo možné spravovat seznam materiálů. Evidován je název materiálu, druh, barva a zda je aktuálně k dispozici.

Třída Config

Třída `Config` uchovává informace o konfiguračních souborech tiskáren. Ty mohou být použity ve více tiskárnách najednou.

Třída Printer

Třída `Printer` je použita pro ukládání informací o tiskárnách. Evidována je primárně URL adresa na které je dostupná spuštěná instance aplikace OctoPrint, dále API klíč pro autorizaci s aplikací OctoPrint, použitý materiál a konfigurační soubor. Pokud nebude zvolen materiál nebo konfigurační soubor, systém nedovolí tiskárnu nastavit jako povolenou k tisku. Tiskárna může být povolena i pokud není zrovna k dispozici, ve chvíli kdy bude možné se k ní připojit, automaticky se tak stane.

Třída Model

Tato třída uchovává uživatelské modely. Kromě cesty k samotnému souboru jsou v ní uloženy i údaje o validitě, velikosti, požadovaném materiálu či datu nahrání. Každý validní model obsahuje i náhled v podobě obrázku.

Třída CombinedModel

Pomocná třída, která uchovává seznam modelů určených k tisku v jedné dávce, model sestavený programem **simarrange** a tiskovou dávku ve formě souboru GCode. Další informace jako datum a čas zahájení tisku, náhled, zda již byl objekt vytisknut, nebo se zrovna tiskne a mnohé další jsou také k dispozici.

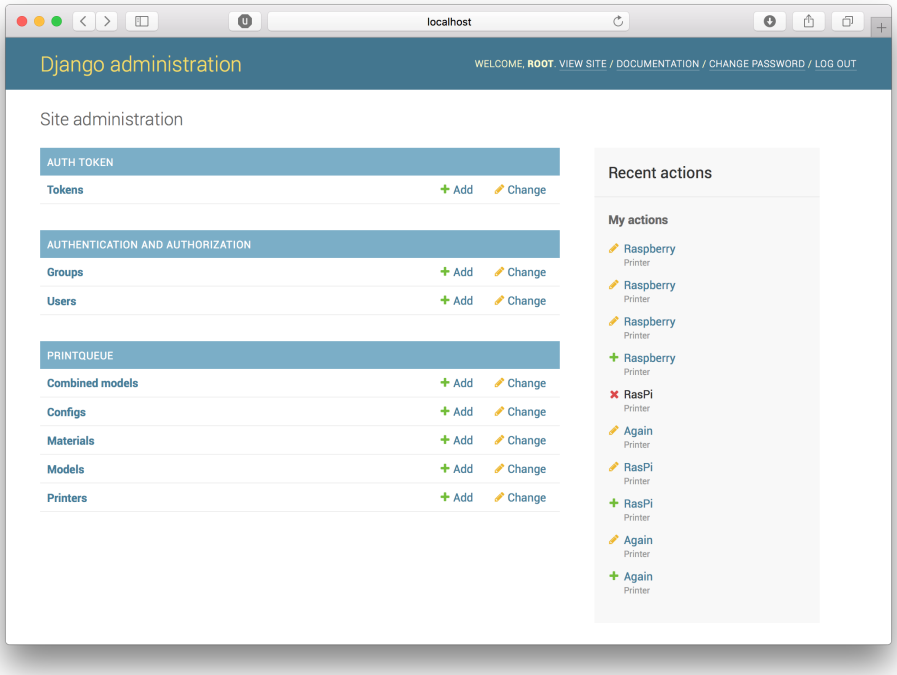
3.4 Komponenty aplikace

Aplikace se skládá ze tří komponent. Pro dosažení správného chodu aplikace musí být spuštěny všechny tři, přestože je možné je spustit nezávisle na sobě.

3.4.1 Hlavní proces

Základní komponentou aplikace je program v podobě jednoduchého webového serveru se kterým probíhá komunikace. Ten po spuštění přijímá a zpracovává

HTTP požadavky a generuje dynamický obsah na základě objektů v databázi a implementované logiky. Součástí frameworku Django je i takzvaná Django administrace, dynamicky generovaná webová aplikace sloužící ke správě objektů aplikace. Jelikož nemáme žádné vlastní grafické rozhraní, lze využít Django administrace, která ho bude prozatím suplovat.



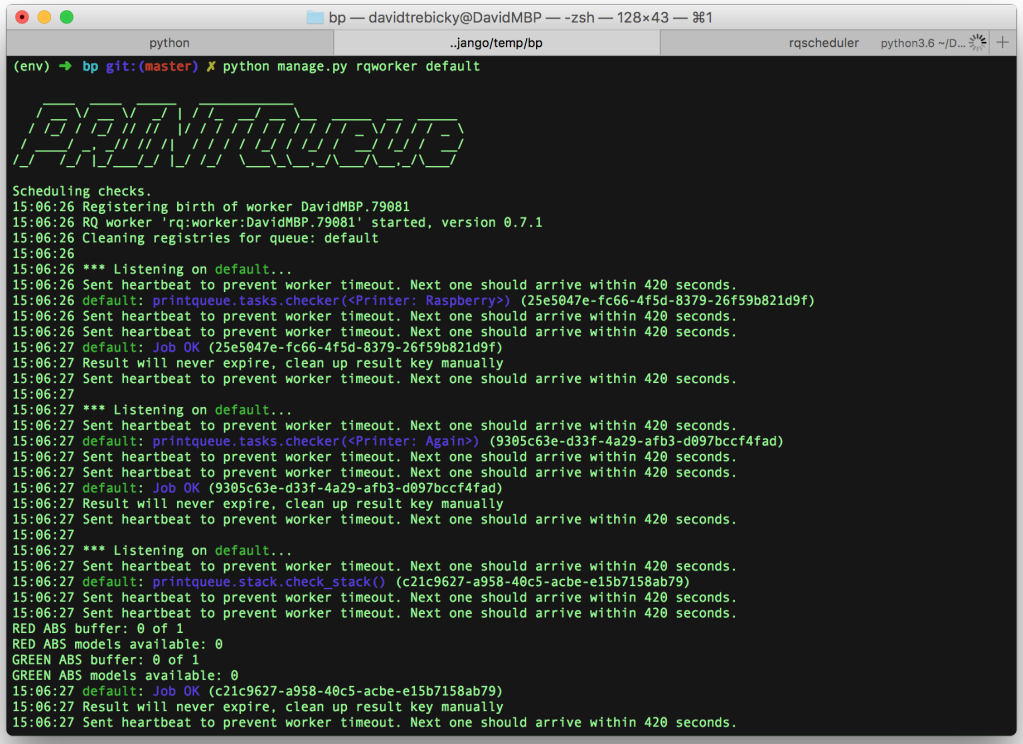
Obrázek 3.3: Webová aplikace Django admin

Na obrázku 3.3 je automaticky vygenerovaná HTML stránka s popisovanými třídami. Zobrazeny jsou i třídy `Users` a `Groups`, které jsou zabudované ve frameworku Django. Význam třídy `Token`, kterou zde vidíme popíši v podkapitole API. Vpravo nahoře nalezneme odkaz na programátorskou dokumentaci.

3.4.2 Worker

Pracovní proces, neboli worker, je komponenta, která zpracovává frontu úloh. Úlohy jsou spouštěny asynchronně a naprosto odděleně od hlavního procesu. Worker má na starost náročné operace, které mohou trvat i několik minut.

Z toho důvodu je možné spustit workerů neomezené množství. Jejich počet by se měl odvíjet od počtu uživatelů používajících aplikaci a tiskáren, které aplikací ovládáme. Pokud nebude dostatek workerů k dispozici, může se stát, že procesy analýzy modelů, či spouštění tiskových úloh budou vykonávány se spožděním, běh hlavního procesu tím však nijak negativně neovlivníme.



Obrázek 3.4: Ukázka běhu instance workeru

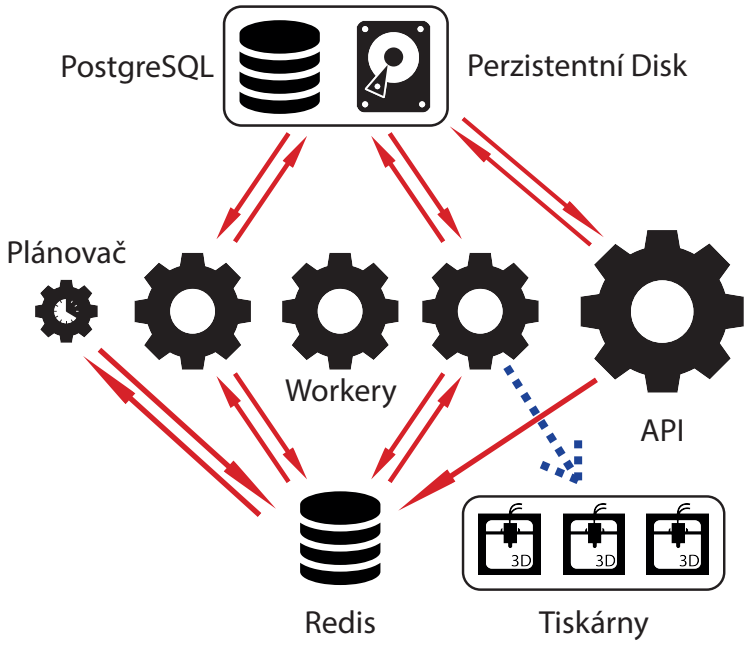
3.4.3 Plánovač

RQScheduler je jednoduchý plánovací program, který zajišťuje periodické vytváření úkolů pro workery. Funguje velmi přímočaře, záznamem v databázi Redis je plánovači předán název funkce a interval v jakém je potřeba ji provádět a ten do stejné databáze v daných intervalech vždy přidá úlohu, kterou si dále vyzvedne jeden z workerů.

3.5 Hlavní logika aplikace

3.5.1 Komunikace komponent aplikace

Na obrázku 3.5 jsou zachyceny komponenty a služby aplikace, jejichž vzájemnou komunikaci nyní popíši.



Obrázek 3.5: Komunikace komponent aplikace

Jako databázový backend pro framework Django slouží databáze PostgreSQL. Hlavní proces ukládá objekty, neboli instance Django Modelů, do databáze a opět je z ní dokáže načítat. Stejným způsobem s touto databází pracuje každá instance workeru. Všechny objekty jsou tedy sdíleny mezi workery a hlavním procesem. V těchto objektech jsou také uchovávány cesty k případným souborům umístěných na perzistentním úložišti, ke kterému mají workery i hlavní proces přístup. Těmito soubory mohou být například zdrojové soubory modelů ve formátu STL [66] nahraných uživateli, jejich vygenerované náhledy či tiskové dávky formátu GCode.

Do databáze Redis ukládáme veškeré periodické úlohy a blokující procesy, které delegujeme na workery – ty si úlohy automaticky přebírají. Databáze Redis nám zde tedy zastupuje frontu těchto úloh. Úlohy mohou být generovány všemi komponentami aplikace.

Hlavní proces vytváří úlohy na základě uživatelské interakce (např. validace nově nahraného modelu), plánovač periodicky přidává předdefinované úlohy (např. periodickou kontrolu ovládaných tiskáren) a worker může na základě zpracovávaných dat vytvořit další úlohy pro asynchronní zpracování (např. naplánování tisku na tiskárnu při zjištění její dostupnosti během kontroly všech tiskáren).

Žádné z komponent spolu tedy nekomunikují přímo. Procesy jsou spouštěny na základě záznamů v databázích, což nám zajišťuje oddělení jednotlivých komponent. Je možné například spustit libovolné množství instancí workerů a v případě potřeby i instancí hlavního procesu.

3.5.2 Zpracování modelu

Modely nahrávají uživatelé spolu s parametry tisku. Parametrem je barva a materiál z kterého požadujeme model vytisknout. Pokud blíže neurčíme materiál a barvu, model bude vytištěn z materiálu, který bude zrovna k dispozici. Bezprostředně po nahrání modelu se spustí automatická kontrola, případně se vygeneruje náhled a vypočítá se objem materiálu potřebného k tisku modelu.

Django implementuje takzvané „signály“, kterými lze odchytit různé akce a navázat na ně vlastní logiku - viz ukázka kódu 3.2.

```
1  @receiver(post_save, sender=Model)
2  def validate(sender, instance, **kwargs):
3      '''
4      Asynchronically validate item
5      '''
6      if kwargs['created']:
7          validate_item.delay(instance)
```

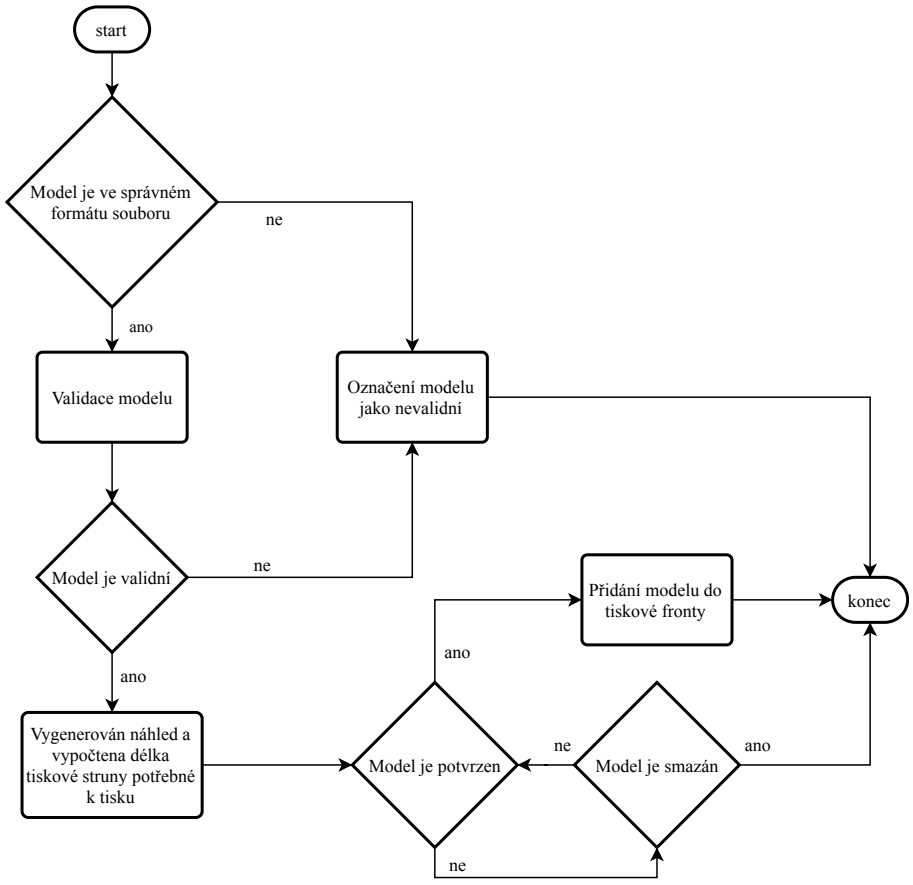
Ukázka kódu 3.2: Signál spouštějící validaci

Ve funkci `validate_item` je využito programu **ADMesh** pro kontrolu validity, programu **OpenSCAD** pro generování náhledu a programu **Slic3r** pro

zjištění délky tiskové struny použitím univerzálního konfiguračního souboru.

Konfigurační soubor je zvolen tak, aby co nejvíce odpovídal reálně spotřebovanému materiálu. Může to být například profil, který je použit na většině tiskáren ovládaných aplikací.

Na diagramu 3.6 lze vidět, co se děje s modely po tom, co je uživatel nahraje do aplikace.



Obrázek 3.6: Proces zpracování modelu systémem

Pokud je model vyhodnocen jako validní, je nutné jeho potvrzení. Tato procedura je zde přítomna z důvodu možnosti rozšíření systému o platby či jiné mechanismy určující, zda-li model bude přidán do tiskové fronty. Tento krok musí být proveden uživatelem s oprávněním obsluhy, které je nastaveno příznakem `is_staff = True`.

Pořadí tisku modelů je dané datem a časem potvrzení.

3.5.3 Periodické úlohy

Aplikace spouští dva typy periodických úloh – kontrolu všech ovládaných tiskáren a kontrolu zásobníku modelů pro tiskárny. Po spuštění komponenty hlavního procesu aplikace je automaticky vložen záznam do databáze Redis, který definuje jakou funkci je potřeba periodicky volat. Způsob uložení záznamu do databáze Redis je patrný z ukázky kódu 3.3.

Funkce `checker` s parametrem tiskárny je spouštěna každých 5 sekund a stará se o zjišťování stavu tiskáren.
Funkce `check_stack` je pouštěna každých 20 sekund a jejím úkolem je zajistit dostatečné množství připravených modelů pro tisk na volných tiskárnách.

```
1  # schedule periodic checks for each printer
2  for printer in Printer.objects.all():
3      scheduler.schedule(scheduled_time=datetime.utcnow(),
   ↪  func=checker, args=[printer], interval = 5)
4
5  # schedule periodic checks for the print queue
6  scheduler.schedule(scheduled_time=datetime.utcnow(),
   ↪  func=check_stack, interval = 20)
```

Ukázka kódu 3.3: Uložení seznamu periodicky volaných funkcí

Pokud nyní spustíme komponentu RQScheduler, z databáze Redis se vyzvednou názvy funkcí a začnou se periodicky plánovat ke spouštění na instancích workerů.

3.5.4 Funkce check_stack

Funkce `check_stack` kontroluje, zda je připraveno dostatečné množství modelů k tisku, respektive jestli existuje pro každý materiál dostatečný počet objektů typu `CombinedModel`. Tento objekt obsahuje konečný 3D model určený k tisku složený z jednotlivých modelů efektivně umístěných na tiskovou

plochu.

Počet dopředu sestavených objektů `CombinedModel` je možné nastavit v hlavním nastavení projektu – v souboru `settings.py` a proměnné `BUFFER`.

Pokud je pro konkrétní materiál připraveno méně tiskových úloh, než je daný limit, funkce přidá úlohu pro workery, kteří se pokusí vytvořit z dostupných modelů jejich kombinaci prostřednictvím programu **simarrange**.

```
1  def check_stack():
2      materials = Material.objects.all()
3      ...
4      for material in materials:
5          combined_count = CombinedModel.objects
   ↪   .filter(material=material)
   ↪   .filter(state_selected=False).count()
6
7          models_count =
   ↪   Model.objects.filter(state_confirmed=True)
   ↪   .filter(state_valid=True)
   ↪   .filter(Q(state_selected=False, material=material) |
   ↪   Q(state_selected=False, material=None)).count()
8
9          if (combined_count < settings.BUFFER and
   ↪   models_count > 0):
10             new_combined = CombinedModel()
11             new_combined.material = material
12             new_combined.save()
13             combine_by_material.delay(material,
   ↪   new_combined)
```

Ukázka kódu 3.4: Funkce `check_stack`

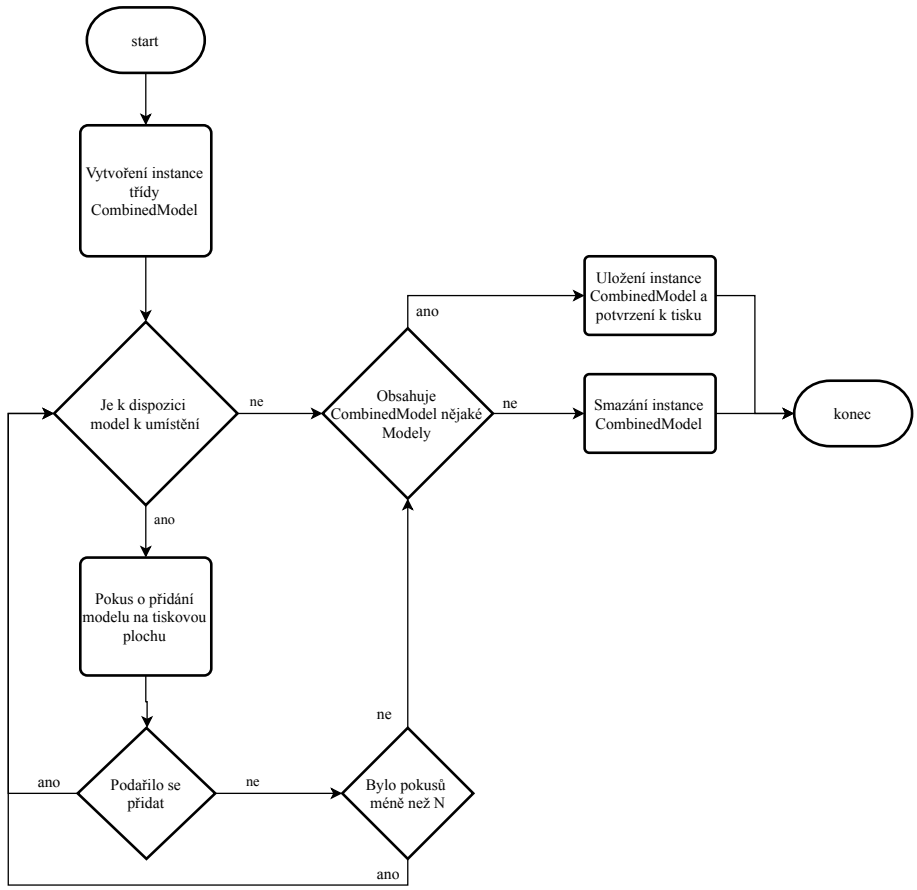
Poznámka: Metoda `Model.objects.filter(...).count()` vnitřně odpovídá SQL dotazu `SELECT COUNT(*) FROM printqueue.Models WHERE ...`, je tedy velmi efektivní, protože výpočet probíhá již v datové vrstvě a to bez nutnosti vybírat data z databáze. [20]

Z ukázky 3.4 je patrné, jakým způsobem je rozhodováno, zda se mají modely začít kombinovat – funkce `check_stack` se pokusí vždy pro každý materiál nalézt počet modelů, které ještě nebyly vytisknuty a mají být z daného materiálu vytištěny, případně nemají materiál stanoven. Pokud nějaké existují a objektů `CombinedModel` pro tento konkrétní materiál je méně než je stanoven limit, vytvoří úlohu pro workery, kteří objekt `CombinedModel` začnou sestavovat. K tomu slouží funkce `combine_by_material`.

Funkce combine_by_material

Funkce `combine_by_material` hledá vždy nejstarší model daného materiálu v databázi a odkazovaný soubor se pokusí umístit na (virtuální) tiskovou plochu prostřednictvím programu **simarrange**. Takto zkouší postupně všechny modely, dokud se jich předem daný počet nepodaří umístit.

Pokud se model programu **simarrange** nepodaří umístit, znamená to, že model je příliš velký, aby se vešel na aktuální tiskovou plochu a je přeskočen. Tím zčásti odfiltrujeme modely, které by svými rozměry blokovaly modely menší.



Obrázek 3.7: Proces umisťování modelů na tiskovou plochu

Velikost tiskové plochy lze nastavit v hlavním nastavení aplikace – v proměnných `BED_WIDTH` a `BED_HEIGHT` v souboru `settings.py`.

Program **simarrange** dokáže efektivně umístit modely na předem nastavenou velikost plochy. Tento program je napsán v jazyce C a obsahoval chybu způsobující nemožnost specifikovat výstupní adresář, do kterého se mají výstupní soubory zapisovat. Tuto chybu jsem opravil ve zdrojovém kódu a zaslal autorovi provedené změny [76].

3.5.5 Funkce checker

Ve funkci `checker` je využita knihovna **OctoClient** [36], pomocí které se worker připojuje k tiskárně. Příklad je v ukázce kódu 3.5.

```
1  def checker(printer):
2      ...
3      try:
4          o~= OctoClient(url=printer.url,
   ↪  apikey=printer.api_key)
5          res = o.printer()
6      except:
7          ...
```

Ukázka kódu 3.5: Zjištění stavu tiskárny

V proměnné `res` je nyní informace o stavu tiskárny. Pokud tiskárna splňuje všechny podmínky pro zahájení tisku, systém se pokusí najít objekt typu `CombinedModel`, odpovídající materiálu, který je v danou chvíli nastaven pro danou tiskárnu. V případě, že je takový model nalezen, je naplánována úloha `prepare_and_print`, které se opět ujme jeden z workerů.

```
1      ...
2      combined = get_file_for_printer(printer)
3      if (combined):
4          printer.state_enabled = False
5          printer.save()
6          prepare_and_print.delay(printer, combined)
```

Ukázka kódu 3.6: Získání objektu CombinedModel a příprava na tisk

Předtím, než se začne připravovat soubor pro tisk, nastaví se příznak tiskárny `state_enabled` na `False`, čímž je docíleno, že ostatní workery nebudou do této tiskárny odesílat žádná další data.

Funkce prepare_and_print

Tato funkce se stará o převod zkombinovaného modelu na tiskovou dávku v podobě souboru GCode. K tomu využívá program **Slic3r**. Ke konverzi používá konfigurační soubor, který je aktuálně přiřazen k tiskárně. Pokud je tiskárna stále dostupná po dokončení konverze, odešle tiskovou dávku a spustí tisk.

```
1  try:
2      print ("Uploading GCode.")
3      ret = o.upload(combined.gcode.file.name, select=True,
   ↪  print=True)['done']
4  except:
5      ...
```

Ukázka kódu 3.7: Odeslání tiskové dávky do tiskárny

V proměnné `ret` máme nyní uloženou návratovou hodnotu, která nám říká, zda odeslání proběhlo v pořádku.

3.6 API

Základním ovládacím prvkem aplikace je HTTP REST API. Využívám knihovnu Django REST framework [18], která značně zjednodušuje její implementaci.

API se skládá z několika koncových bodů, které odpovídají definovaným třídám `User`, `Model`, `Config`, `Printer` a `Material`. Každý z těchto koncových bodů slouží pro manipulaci s objekty daného typu.

S API je možné komunikovat standadním způsobem pomocí příslušných HTTP požadavků **POST**, **GET**, **PATCH** a **DELETE**. Data jsou předávána v těle požadavku ve formátu JSON [39]. Odpověď aplikace obsahuje navíc ještě HTTP stavový kód.

3.6.1 Autentizace

Z pohledu uživatele začíná interakce s aplikací registrací uživatelského účtu. Ten je vytvořen pomocí HTTP požadavku **POST** na koncový bod `/api/users/`

a je popsán v ukázce 3.8.

```
POST /api/users/ HTTP/1.1
Host: 127.0.0.1:8000
Content-Type: application/json
{"username": "user", "password": "user1234", "first_name":
↪  "Test", "last_name": "User", "email": "test@user.com" }
------------------------------
201 Created
{
  "id": 2,
  "username": "user",
  "first_name": "Test",
  "last_name": "User",
  "email": "test@user.com",
  "auth_token": "601cc31cf719bc05e303a2ed2b33b2b8fce95390"
}
```

Ukázka kódu 3.8: Registrace uživatele voláním koncového bodu API

Uživatelům je při registraci vygenerován API token, kterým se dále autentizují a jsou stejným způsobem také autorizováni k provedení určitých operací. Tento token je odesílán s HTTP požadavky v jeho hlavičce.

3.6.2 Autorizace

Uživatelé jsou rozděleni na běžné uživatele a obsluhu, podle toho do jaké skupiny patří, jsou definována jejich práva na operace s objekty aplikace.

Běžní uživatelé mohou nahrávat a následně upravovat nahrané modely do chvíle, než jsou potvrzeny k zařazení do tiskové fronty. Změnit mohou jejich materiál nebo název. Dále mohou také upravit svůj uživatelský profil, zobrazit dostupné materiály, seznam svých modelů či jejich stav. Na další operace oprávnění nemají a při pokusu o jejich provedení dojde k chybě – viz ukázka 3.9.

Obsluha má oprávnění zasahovat do všech modelů a nastavovat jim různé příznaky. Uživatelé této skupiny tedy mohou například potvrdit uživatelské modely, aby byly zařazeny do tiskové fronty, ale také nastavit, aby byly znovu vytištěny v případě selhání tisku. Kromě toho mohou spravovat tiskárny a jejich konfigurační profily, přidávat nové materiály či nastavovat jejich dostupnost.

```
GET /api/printers/ HTTP/1.1
Host: 127.0.0.1:8000
Authorization: Token
↪  601cc31cf719bc05e303a2ed2b33b2b8fce95390
Content-Type: application/json
------------------------------

403 Forbidden

{
  "detail": "You do not have permission to perform this
↪  action."
}
```

Ukázka kódu 3.9: Přístup k informacím bez oprávnění

```
GET /api/printers/ HTTP/1.1
Host: 127.0.0.1:8000
Authorization: Token
↪  02bddcd068641c18ca381eaaf7c6c3341da1cfc8
Content-Type: application/json
------------------------------

200 OK

[
  {
    "id": 1,
    "name": "Printer",
    "url": "http://printer:5000",
    "api_key": "BB3F2FBE30494E298645F22D30927B14",
    "state_enabled": true,
    "config_in_use": 1,
    "material_in_use": 1,
    "state_operational": true,
    "state_ready": true,
    "state_printing": false
  }
]
```

Ukázka kódu 3.10: Přístup k informacím s oprávněním

3.6.3 Příprava pro GUI

API je navrženo tak, aby bylo možné ji jednoduše napojit na případné grafické uživatelské rozhraní.

3.7 Možnosti rozšíření

3.7.1 Vylepšení návrhu

Během implementace jsem objevil několik možných problémů, které mohou nastat.

Nyní upřednostňuji především rychlost generování tiskových dávek, a proto jsou modely sestavovány předem a následně jen konvertovány do tiskových dávek pro konkrétní tiskárny. Tato verze aplikace nepočítá s různými velikostmi tiskových ploch tiskáren a používá jen jedno globální nastavení. Aplikaci je možné poměrně malým zásahem do implementace přizpůsobit i scénáři použití více druhů tiskáren.

Další problém může nastat, pokud nebude dostatek spuštěných instancí workerů. Zjišťování stavu tiskáren pak může být velmi zpomaleno. Všechny asynchronní úlohy totiž uchovávám pouze v jedné frontě. Tento nedostatek by ovšem také neměl být problém opravit a dalo by se například využít dvou separátních front. Jedna by byla určena pro velmi pomalé procesy a druhá pro procesy rychlé, do kterých periodické kontroly tiskáren spadají. Případně by se dal také ještě oddělit provoz tiskáren od interakce uživatelů. Analýza nově nahraných modelů by tak negativně neovlivňovala provoz tiskáren a obráceně.

Ve fázi validace je sice zaručeno, že model bude možné převést na tiskovou dávku, ovšem není zcela jisté, že ho opravdu lze vytisknout. Bylo by vhodné zapojit ještě další mechanismy, které by rozeznaly nevyhovující modely. Mohl by se například počítat poměr obsahu podstavy k ostatním vrstvám modelu a alespoň z části tak odfiltrovat modely, které by při tisku nemusely stabilně držet na tiskové podložce. Takové modely by například bylo nutné manuálně odsouhlasit obsluhou.

3.7.2 Rozšíření

Ačkoli jsem pro převod modelů na tiskové dávky vybral program **Slic3r** [70], mnoho uživatelů preferuje software Cura [77]. Cura disponuje také rozhraním pro ovládání z příkazové řádky – CuraEngine [41], a nebyl by tedy problém tuto možnost přidat – ideálně tak, aby mohly být volitelně použity oba řezací programy.

Místo volání programu **ADMesh** [45] by bylo přehlednější použít nativní knihovnu **python-admesh** [35]. Při jejím použití jsem ovšem zdaleka nedosahoval rychlosti jako při volání programu z příkazové řádky. Nejvíce se zpomalení projevilo při opravě chybných modelů. Nejdříve by tedy bylo nutné analyzovat, zda by se tato změna vyplatila.

Kapitola 4

Testování a nasazení

4.1 Testování

V této podkapitole se věnuji testování aplikace. K testování využívám platformu Docker, která umožňuje spustit několik instancí virtualizovaného operačního systému. V jednotlivých instancích budou spuštěny služby potřebné pro běh aplikace.

Cílem je zautomatizovat systém tak, aby ho bylo možné spustit na libovolném počítači, který disponuje programem Docker a to bez nutnosti nastavovat jednotlivé služby. Bude tak umožněno systém omezeně testovat potencionálními uživateli ještě před jeho nasazením. Aplikaci otestuji nejdříve manuálně a následně sérií HTTP požadavků z aplikace Postman [54].

4.1.1 Testovací tiskárny

Aplikaci lze samozřejmě otestovat na fyzických tiskárnách, nicméně pro účel zjištění jejího správného fungování budou stačit tiskárny virtuální. Souborem `Dockerfile` je definován server se spuštěnou aplikací OctoPrint a virtuální tiskárnou, kterou lze následně využívat stejným způsobem jako tiskárnu fyzickou.

Využijeme již dostupného Docker image [23] `mrwyss/octoprint`, jehož zdrojový soubor `Dockerfile` je k dispozici z [47].

Příkazem `docker run -d -p 5000:5000 mrwyss/octoprint` stáhneme image a spustíme kontejner. Pomocí přepínače `-p` je zde mapován port na kterém poslouchá aplikace OctoPrint na port lokální adresy. Spuštěná instance aplikace OctoPrint je nyní dostupná na adrese `http://localhost:5000`. Zde je také možné zjistit API klíč pro vzdálené ovládání.

Pokud bychom chtěli takto spustit více virtuálních instancí aplikace OctoPrint, museli bychom proces vždy opakovat. Proto jsem vytvořil vlastní zdrojový soubor Dockerfile a API klíč i uživatelský účet aplikace definuji předem v konfiguračních souborech.

Pro vlastní zdrojový soubor `Dockerfile` využiji stejný image `mrwyss/octoprint` a rozšířím ho o zmíněné nastavení. V ukázce 4.1 je jeho obsah.

```
FROM mrwyss/octoprint

ADD config.yaml /data/config.yaml
ADD users.yaml /data/users.yaml
```

Ukázka kódu 4.1: Dockerfile pro aplikaci OctoPrint

4.1.2 Docker Compose

Součástí programu Docker je ve většině linuxových distribucí i pomocný nástroj Docker Compose [22], případně je možné ho nainstalovat balíčkovacím manažerem, či nástrojem *pip*. Pro Windows a macOS je součástí oficiálního instalačního souboru.

Pomocí nástroje Docker Compose jsme schopni nadefinovat nastavení jednotlivých virtuálních serverů a jejich závislosti, či další pokročilé nastavení. Definice se zapisují do zdrojového souboru formátu YAML [26], který je zpravidla pojmenován `docker-compose.yml`.

Nejdříve definuji služby, které budou potřeba pro běh aplikace. Jsou to databáze PostgreSQL a Redis. Pro otestování přidávám také instanci aplikace OctoPrint, která bude představovat jednu virtuální tiskárnu. Službě mapuji konfigurační soubory o kterých jsem psal dříve. Výsledný soubor by mohl vypadat jako v ukázce 4.2.

```
version: '2'
services:
  postgres:
    image: postgres:latest
    env_file: .env
    expose:
      - "5432"
    volumes:
      - /var/lib/postgresql/data/

  redis:
    image: redis:latest
    expose:
      - "6379"

  printer:
    image: mrwyss/octoprint
    volumes:
      - ./printer/config.yaml:/data/config.yaml
      - ./printer/users.yaml:/data/users.yaml
    ports:
      - "3100:5000"
    expose:
      - "5000"
  ...
```

Ukázka kódu 4.2: Zdrojový soubor nástroje Docker Compose – služby

Nyní přidávám definici aplikace, tedy všech jejích komponent. Hlavní proces, worker i plánovač mají společný Docker image, jehož zdrojový soubor `Dockerfile` je v podsložce `web/`, jak je vidět z ukázky 4.3.

```
  ...
  web:
    build: ./web
    ports:
      - "8000:8000"
    links:
      - redis
      - postgres
    volumes:
      - /usr/src/app/static
      - /usr/src/app/media
    env_file: .env
    command: /usr/local/bin/python manage.py runserver

  scheduler:
    build: ./web
    volumes_from:
      - web
    links:
      - redis
    command: /usr/local/bin/rqscheduler -i 2 --host redis
↪   --port 6379 --db 0

  worker:
    build: ./web
    links:
      - redis
      - postgres
    env_file: .env
    volumes_from:
      - web
    command: /usr/local/bin/python manage.py rqworker
↪   default
```

Ukázka kódu 4.3: Zdrojový soubor nástroje Docker Compose – aplikace

Při vytváření image pro aplikaci jsem narazil na problém. Server na kterém je spuštěna aplikace totiž běží v režimu headless, což znamená, že není připojeno žádné zobrazovací zařízení a jsme odkázáni pouze na vzdálený přístup z příkazové řádky. Tento režim má u virtuálních systémů spouštěných jako kontejnery pochopitelný význam. Služby v systému většinou běží na pozadí a není tudíž potřeba žádné grafické zobrazovací zařízení.

Program OpenSCAD [40], kterým je generován náhled modelů, však při vykreslování zobrazovací zařízení potřebuje. Jako řešení jsem nalezl spuštění virtuálního X serveru [67]. V našem případě tuto funkci plní program **xvfb** [68]. Program supervisord [11] zajišťuje, že je X server vždy spuštěn a v případě jeho neočekávaného ukončení je restartován. V ukázce 4.4 je možné si pro-

hlédnout kompletní definici. Všechny soubory jsou k dispozici i na přiloženém médiu.

```
FROM python:3.5

COPY xorg.conf /xorg.conf

ENV DISPLAY=:10
ENV DEBIAN_FRONTEND=noninteractive

COPY keyboard /etc/default/keyboard

RUN apt-get update && \
    apt-get -y install libhighgui-dev libopencv-dev
↪   libargtable2-dev libadmesh-dev && \
    git clone https://github.com/trebidav/simarrange.git &&
↪   \
    cd simarrange && \
    make && \
    make install && \
    apt-get -y install xvfb && \
    apt-get -y install supervisor && \
    apt-get -y install admesh slic3r openscad && \
    mkdir -p /root/.local/share && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

COPY supervisord/*.conf /etc/supervisor/conf.d/

COPY run.sh /run.sh
RUN chmod +x /run.sh

ENTRYPOINT ["/run.sh"]
```

Ukázka kódu 4.4: Soubor Dockerfile pro běh aplikace

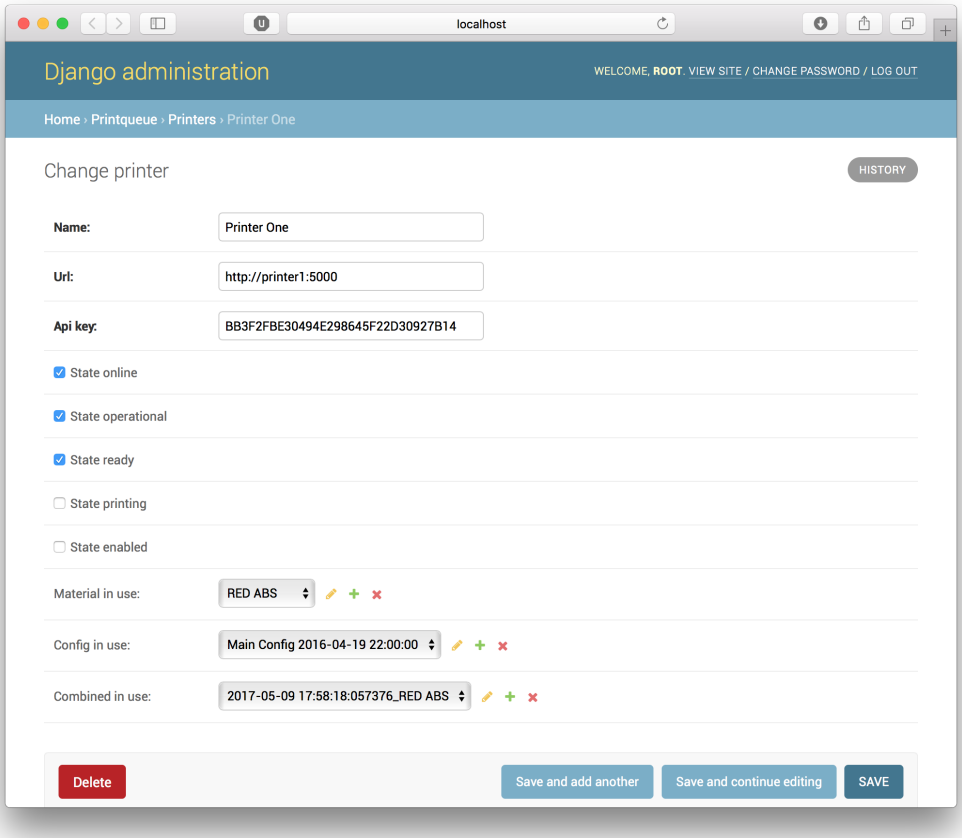
4.1.3 Průběh testování

Následuje seznam příkazů, kterým dostaneme systém do stavu, kdy je možné začít s testováním.

- Spustíme databázi PostgreSQL příkazem `docker-compose up -d postgres`.
- Do databáze nahrajeme její schéma příkazem `docker-compose run web python manage.py migrate`.
- Nahrajeme testovací data příkazem `docker-compose run web python manage.py loaddata initial_data`.

- Spustíme celou aplikaci příkazem `docker-compose up`.

Nyní se přihlásíme do Django administrace na adrese `http://localhost:8000/admin` s účtem superuživatele. Jeho předdefinované přihlašovací údaje ze souboru `initial_data.yaml` jsou `root:root1234`. Využitím Django administrace bychom nyní mohli aplikaci otestovat. Můžeme například zobrazit detaily připojené tiskárny – viz obrázek 4.1.



Obrázek 4.1: Detail tiskárny v Django administraci

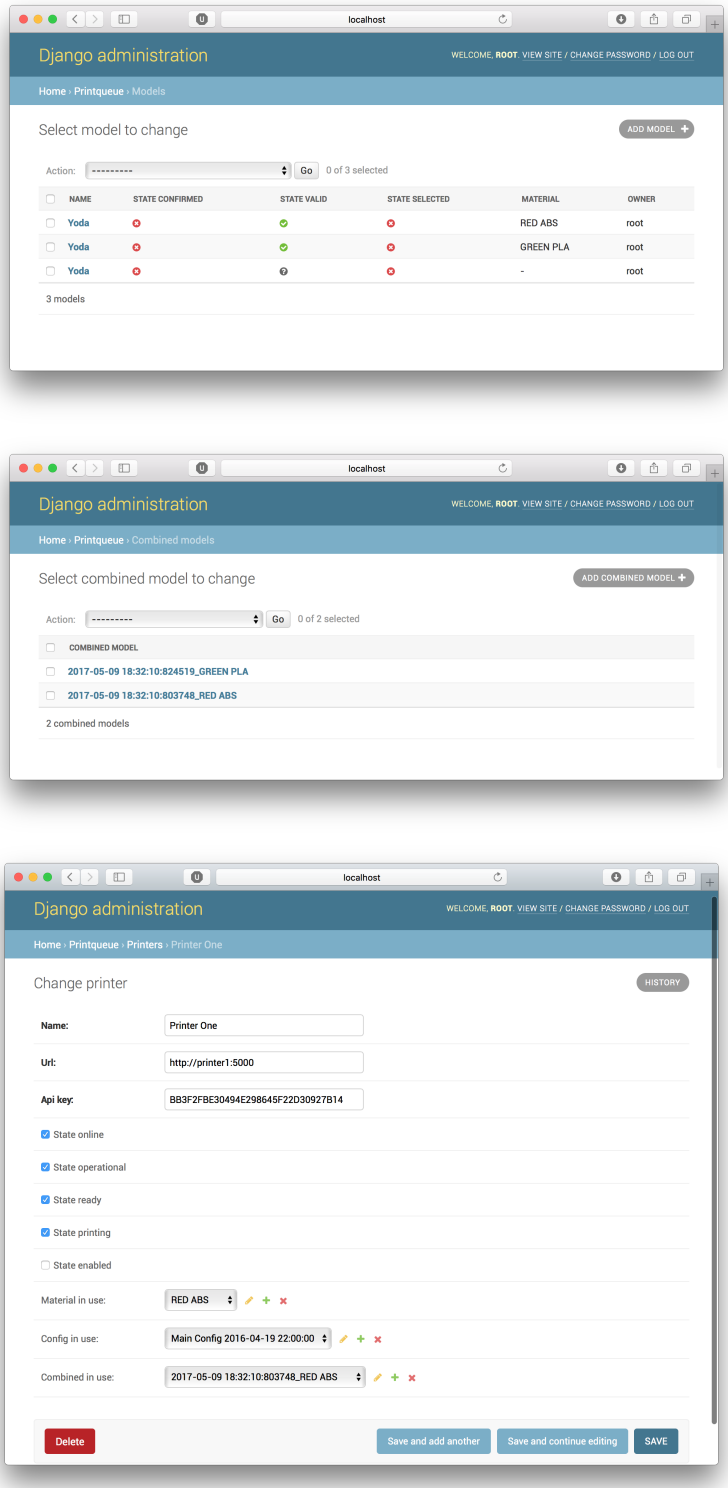
Aplikaci je však vhodnější otestovat způsobem, kterým bude běžně ovládána, tedy prostřednictvím API. Z Django administrace zjistíme pouze Token uživatele `root`. Nyní je možné odeslat požadavek z ukázky 4.5, pro povolení tiskárny k tisku.

```
PATCH /api/printers/1/ HTTP/1.1
Host: 127.0.0.1:8000
Authorization: Token
↪  ece3f3c224eec18ff1a6a3e3982f81ad7f091016
Content-Type: application/json
{
  "state_enabled": true
}
------------------------------

200 OK

{
  "id": 1,
  "name": "Printer One",
  "url": "http://printer1:5000",
  "api_key": "BB3F2FBE30494E298645F22D30927B14",
  "state_enabled": true,
  "config_in_use": 1,
  "material_in_use": 1,
  "state_operational": true,
  "state_ready": true,
  "state_printing": false
}
```

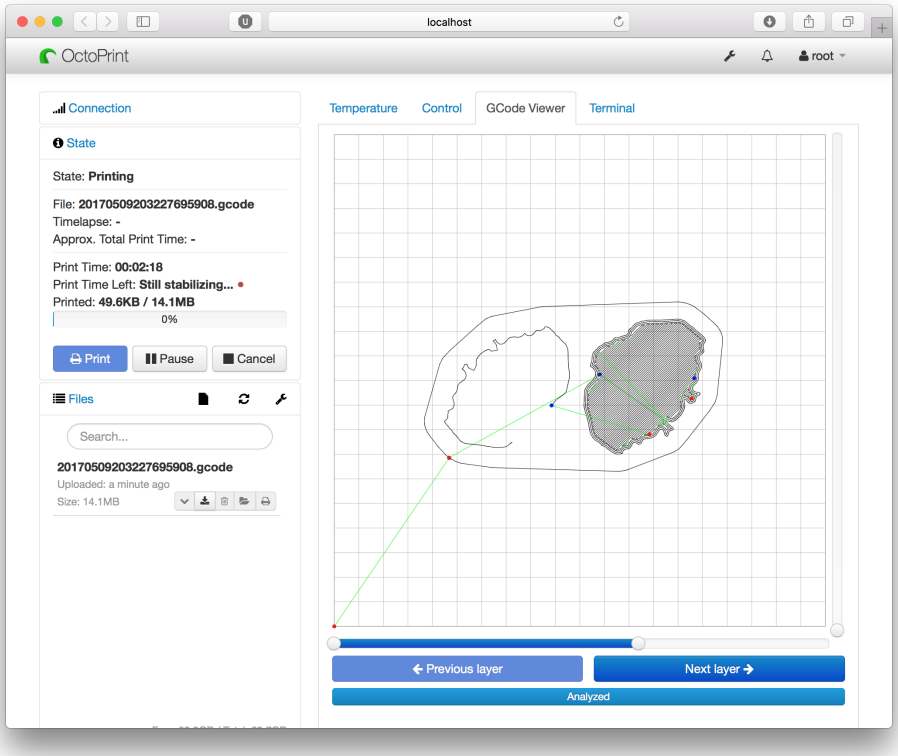
Ukázka kódu 4.5: Povolení tiskárny k tisku

Nahrajeme tři testovací modely `yoda.stl`. Prvnímu nastavíme `material=1`, druhému `material=2` a třetímu `material=null`, tedy ponecháme ho bez specifikovaného materiálu. Všechny tři potvrdíme k tisku voláním metody **PATCH**, obdobně jako při úpravě nastavení tiskárny v ukázce 4.5.

Nyní modely projdou validací, umístí se na tiskovou plochu a odešlou se k tisku na příslušnou tiskárnu. Jelikož je k dispozici jen tiskárna s materiálem `id=1`, umístí se model bez přiřazeného materiálu tak, aby byl vytištěn z libovolného materiálu co nejdříve – tedy materiálem s `id=1`. V Django administraci můžeme sledovat průběh procesu – viz obrázky v ukázce 4.2. V aplikaci **OctoPrint** se následně můžeme přesvědčit, že byl tisk spuštěn (obrázek 4.3).



Obrázek 4.2: Proces automatizovaného tisku



Obrázek 4.3: Tisk spuštěný na cílové tiskárně

Provedené testy je nyní možné automatizovat, je ovšem nutné počítat s prodlevami mezi operacemi, které jsou testovány. Tiskárna například nezačne tisknout ihned po jejím povolení. Systém musí nejdříve připravit tiskovou dávku a tu odeslat po síti.

V ukázce 4.4 jsou výsledky testů ověřující funkcionalitu API pro koncové uživatele provedené pomocí programu **newman** [53], který je určen pro spouštění testů aplikace **Postman** z příkazové řádky. Jejich definice jsou dostupné ve formátu JSON v souboru na přiloženém médiu. Tyto testy je vhodné využívat například při nasazování nových verzí aplikace.

Obrázek 4.4: Automatizované testy API

4.1.4 Výsledky testování

Po otestování jsme zjistili, že aplikace funguje podle předpokladů. V tomto rozsahu však výsledky nemají příliš vysokou vypovídající hodnotu o plynulosti běhu aplikace v produkčním prostředí se stovkami uživatelů a tiskáren. K takovému testování bohužel nedostačuje výkon běžného osobního počítače, na kterém jsem testy prováděl. V následující kapitole popíši mechanismy, které v reálném případě užití pomohou dosáhnout ideálních výsledků i při vysoké zátěži.

4.2 Nasazení

V této podkapitole se věnuji návrhu nasazení. Rozšířím aplikaci o mechanismy, které by měly zaručit bezchybný běh i při vysokém zatížení a popíšeme způsob, jakým by aplikace mohla být nasazena do reálného produkčního prostředí.

Do teď jsme spouštěli aplikaci ve vývojářském módu příkazem `python manage.py runserver`, který suploval plnohodnotný webový server. V produkčním nasazení budou ovšem nutné mnohem sofistikovanější mechanismy pro kontrolu přístupu a zabezpečení aplikace. Běžnou praxí při nasazování Django aplikací je použití platformy WSGI [59].

4.2.1 Gunicorn

Gunicorn [17], nebo-li Green Unicorn je HTTP WSGI server naprogramovaný v Pythonu. WSGI [59] je rozhraní, které umožňuje komunikaci mezi webovým serverem a webovou aplikací a je součástí i všech Django projektů. Hlavním úkolem webového serveru Gunicorn je naslouchání příchozích požadavků a jejich delegování na naši aplikaci způsobem definovaným ve specifikaci WSGI. Naše aplikace naopak odpoví přes rozhraní WSGI webovému serveru a ten odešle HTTP odpověď klientovi. [59] V ukázce kódu 4.6 lze vidět změnu ve spouštěném příkazu.

```
  web:
    restart: on-failure
    build: ./web
    expose:
      - "8000"
    ...
    command: /usr/local/bin/gunicorn bp.wsgi:application -w
↪   4 -b :8000
```

Ukázka kódu 4.6: Upravený zdrojový soubor Docker compose

4.2.2 NGINX

Je běžná praxe, že se k serveru Gunicorn nepřistupuje přímo. Jelikož je potřeba, kromě jiného, odstínit požadavky na statické soubory a odlehčit tak zátěž serveru Gunicorn, přidávám „před“ něj nyní ještě jeden webový server. Protože se zaměřuji na škálování a dostupnost, zvolil jsem webový server NGINX, který je pro tyto funkce přímo určen. Zajistí zároveň bezpečnost spojení přidáním HTTPS cerifikátu, kterým se nyní bude šifrovat komunikace s klienty. Bude poslouchat na standardních portech 80 a 443. Ostatní

požadavky, které vyžadují dynamické generování jsou pomocí funkce reverzní proxy delegovány na server Gunicorn a potažmo hlavní proces aplikace. Pokud by v budoucnosti aplikace nestíhala vyřizovat dostatek požadavků, bylo by možné do konfigurace serveru NGINX přidat ještě roli rozložení zátěže - angl. *load balancing*. Stačilo by tak přidat jen IP adresu, či název další instance aplikace do příslušné direktivy. Příklad konfigurace lze vidět v ukázce kódu 4.7.

```
upstream app {
        server app1:8000;
        server app2:8000;
    }
server {
    listen       80;
    server_name  example.org;
    return 301   https://$server_name$request_uri;
}
server {
    listen       443 ssl;
    server_name  example.org;

    ssl_certificate       /etc/ssl/server.crt;
    ssl_certificate_key   /etc/ssl/server.key;

    charset               utf-8;
    client_max_body_size 64M;

    location /static {
        alias /usr/src/app/static;
    }

    location / {
        proxy_pass       http://app;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
↪   $proxy_add_x_forwarded_for;
    }
}
```

Ukázka kódu 4.7: Konfigurační soubor webového serveru NGINX

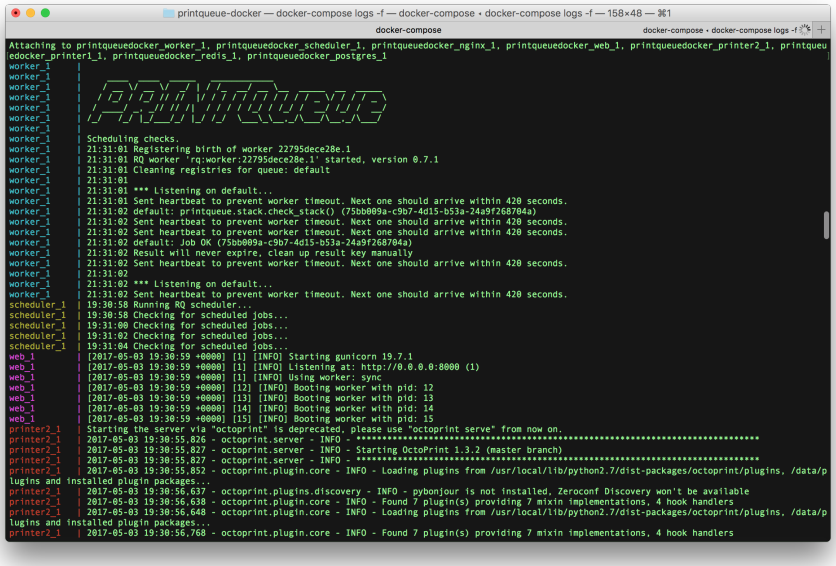
Definice serveru NGINX v souboru `docker-compose.yml` by vypadala například jako v ukázce 4.8.

```
nginx:
    restart: always
    build: ./nginx/
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /www/static
    volumes_from:
      - web
    links:
      - web:web
```

Ukázka kódu 4.8: Definice serveru NGINX v souboru `docker-compose.yml`

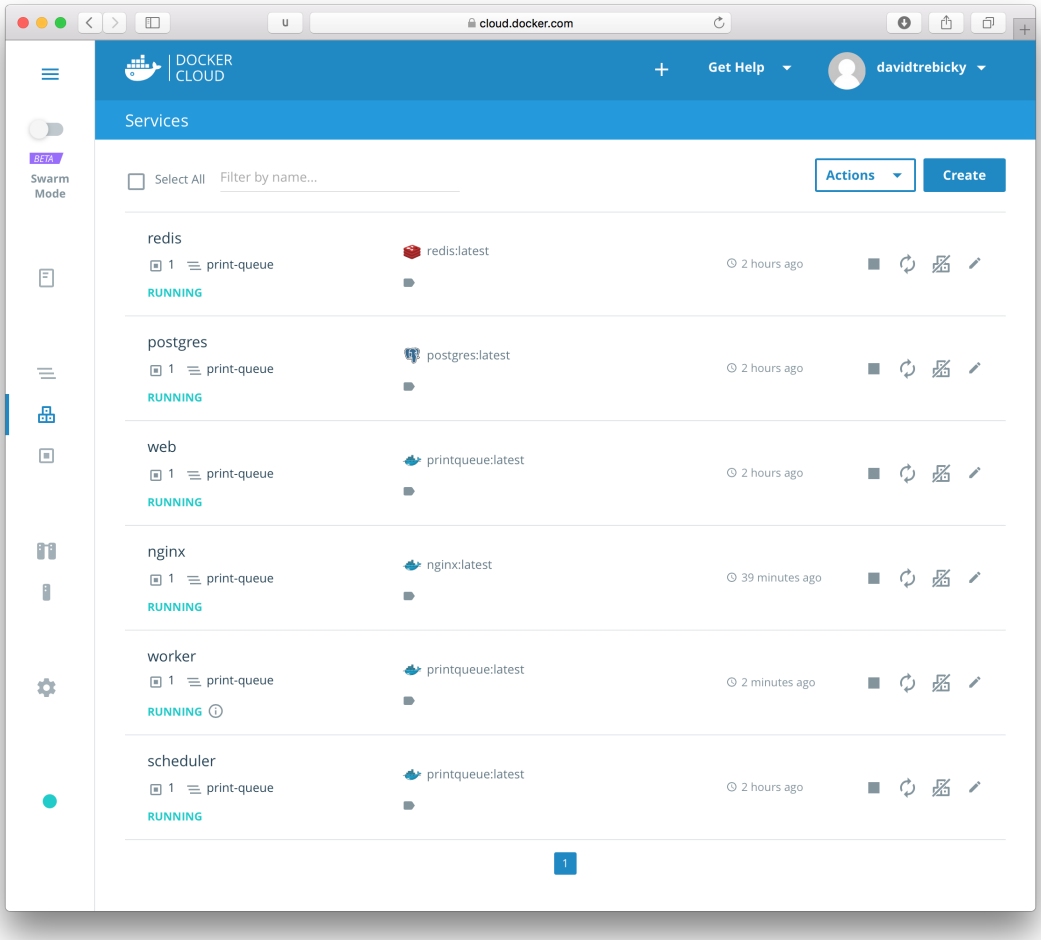
4.2.3 Další postupy nasazení

Nyní jsou všechny služby zabalené do formátu Docker image a připravené k nasazení na libovolnou (například cloudovou) službu podporující spouštění kontejnerů. Pro otestování funkčnosti je vhodné spustit všechny služby lokálně nástrojem Docker Compose. Výpisy služeb spuštěného systému můžeme vidět v ukázce 4.5. Rozhraní API je nyní dostupné na standardní adrese `https://localhost/api/`.



Obrázek 4.5: Výpis jednotlivých služeb v nástroji Docker compose

V případě platformy Docker Cloud [24] by byly nutné jen minimální zásahy do souboru `docker-compose.yml` s definicemi služeb. U jiných služeb by bylo nutné definice upravit, aby odpovídaly konkrétním specifikacím. Vyžadováno by bylo také perzistentní úložiště, sdílené mezi komponentami aplikace. Takové perzistentní úložiště nabízí například Amazon nebo Google ke svým cloudovým službám [13] [29].



Obrázek 4.6: Aplikace nasazená službou Docker Cloud

Osobně jsem byl schopen aplikaci úspěšně nasadit do produkčního prostředí (obrázek 4.6) a vyzkoušel jsem ji také na fyzických tiskárnách RepRap na Fakultě informačních technologií ČVUT.

Závěr

Ačkoli je proces 3D tisku zpravidla považován za velmi komplexní, přesvědčili jsme se, že lze efektivně zjednodušit tak, že i nezkušený uživatel je schopen vytisknout model na 3D tiskárně.

V rešeršní části jsem analyzoval, jaké možnosti 3D tisku mají v současnosti běžní uživatelé. Výsledky byly podle předpokladu neuspokojivé. Ceny za služby jsou velmi vysoké i v případě tisku malých modelů. Zjistil jsem, co má na cenu největší vliv – technologie a nutnost odborné obsluhy. Ačkoli jsem při porovnání služeb volil levnější technologii tisku, stále se v ceně nejvíce odrážela práce, kterou musí odborná obsluha vykonat. Identifikoval jsem procesy, které nyní obsluha běžně provádí a mohly by být automatizovány. Zjistil jsem, že některé programy, které jsou používány pro přípravu modelů k tisku a k ovládání tiskáren, lze téměř zcela automatizovat a pro manipulaci s tiskovou plochou již nejsou nutné odborné znalosti.

V části práce věnované návrhu jsem navrhl systém, který automatizuje zmíněné procesy a umožní tak uživatelům tisknout na 3D tiskárnách bez předchozích zkušeností. Určil jsem také, jakým způsobem se bude systém ovládat a na jaké platformě by mohl být ideálně nasazen.

V implementační části jsem prostřednictvím moderních nástrojů vytvořil webovou aplikaci, která naplnila kritéria návrhu. Stěžejní byla automatizace existujících programů a logika odbavování uživatelských modelů k cílovým tiskárnám. Z implementace jsem záměrně vynechal grafické rozhraní, které se pravděpodobně bude lišit v závislosti na tom, kde bude aplikace použita. K budoucímu napojení na grafické rozhraní jsem připravil standardizované rozhraní API.

V průběhu implementace jsem odhalil některé chyby a navrhl jsem možné budoucí řešení, jak je eliminovat.

Aplikaci jsem manuálně otestoval nejen na virtuálních tiskárnách, ale také na fyzických zařízeních RepRap. Vytvořil jsem také automatizované testy rozhraní aplikace.

Pomocí doporučených postupů jsem připravil systém pro vysokou zátěž, danou velkým počtem tiskáren a uživatelů. Zohlednil jsem zejména bezpečnost komunikace a možnost škálovatelnosti v závislosti na aktuálním vytížení. Navrhl jsem způsob nasazení, kde jsem se soutředil zejména na moderní technologie cloudových služeb. Sám jsem byl schopen aplikaci na cílovou platformu nasadit.

Výsledkem je tedy funkční a reálně nasaditelná aplikace, pomocí které lze rychle a bez nutnosti odborné obsluhy tisknout se sníženými náklady. Náklady je možné téměř ihned stanovit koncovým uživatelům s velkou přesností. Projekt hodlám i nadále rozvíjet.

Literatura

[1] 3D Hubs B.V. 3D Printing Trends Q2-2017. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/trends#printers`

[2] 3D Hubs B.V. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com`

[3] atencom3D's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/150554`

[4] Kinamico's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/Kinamico`

[5] creativemedia's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/143101`

[6] COTU's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/56568`

[7] Xcom3Dprint's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/231853`

[8] 3Dees's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/66753`

[9] 3D obiecto's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/27001`

[10] 3dtiskservice.cz's Hub. *3D Hubs: Browse Online 3D Printing Services* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.3dhubs.com/service/172267`

[11] Agendaless Consulting. *Supervisord: A Process Control System* [online]. [cit. 3.5.2017]. Dostupné z: http://supervisord.org

[12] Aleph Objects, Inc. LulzBot TAZ 6. *LulzBot | 3D Printer, Parts and Filament* [online]. [cit. 3.5.2017]. Dostupné z: https://www.lulzbot.com/store/printers/lulzbot-taz-6

[13] Amazon Web Services, Inc. Amazon Elastic Block Store. *Amazon Web Services (AWS) - Cloud Computing Services* [online]. [cit. 3.5.2017]. Dostupné z: https://aws.amazon.com/ebs/

[14] Amazon Web Services, Inc. *Amazon Web Services (AWS) - Cloud Computing Services* [online]. [cit. 3.5.2017]. Dostupné z: https://aws.amazon.com

[15] Autodesk, Inc. *Autodesk Meshmixer* [online]. [cit. 3.5.2017]. Dostupné z: http://www.meshmixer.com

[16] Autodesk, Inc. *Netfabb* [online]. [cit. 3.5.2017]. Dostupné z: http://www.netfabb.com

[17] CHESNEAU, B. et al. *Gunicorn - Python WSGI HTTP Server for UNIX* [online]. [cit. 3.5.2017]. Dostupné z: http://gunicorn.org

[18] CHRISTIE, T. et al. *Django REST* framework. 2016, [cit. 3.5.2017]. Dostupné z: http://www.django-rest-framework.org/

[19] A Brief Description. *cplusplus.com - The C++ Resources Network* [online]. [cit. 3.5.2017]. Dostupné z: http://www.cplusplus.com/info/description/

[20] Django Software Foundation. QuerySet API reference | Django documentation. *Django - The web framework for perfectionists with deadlines.* [online]. [cit. 3.5.2017]. Dostupné z: https://docs.djangoproject.com/en/1.11/ref/models/querysets/#django.db.models.query.QuerySet.count

[21] Django Software Foundation. *Django - The web framework for perfectionists with deadlines* [online]. [cit. 3.5.2017]. Dostupné z: https://www.djangoproject.com

[22] Docker, Inc. Docker Compose. *Docker Documentation* [online]. [cit. 3.5.2017]. Dostupné z: https://docs.docker.com/compose/

[23] Docker, Inc. Docker images. *Docker documentation* [online]. [cit. 3.5.2017]. Dostupné z: https://docs.docker.com/engine/reference/commandline/images/#extended-description

[24] Docker, Inc. *Docker Cloud* [online]. [cit. 3.5.2017]. Dostupné z: `https://cloud.docker.com/`

[25] DRIESSEN, V. et al. *RQ: Simple job queues for Python* [online]. [cit. 3.5.2017]. Dostupné z: `http://python-rq.org`

[26] EVANS, C. C. *The Official YAML Web Site* [online]. [cit. 3.5.2017]. Dostupné z: `http://yaml.org`

[27] FlashForge Corporation. Creator Pro 3D Printer. *FlashForge 3D Printers* [online]. [cit. 3.5.2017]. Dostupné z: `http://www.flashforge.com/product/creator-pro-3d-printer/`

[28] FOWLER, M.. *Microservices* [online]. [cit. 3.5.2017]. Dostupné z: `https://martinfowler.com/articles/microservices.html`

[29] Google, Inc. Persistent Disk. *Google Cloud Platform* [online]. [cit. 3.5.2017]. Dostupné z: `https://cloud.google.com/persistent-disk/`

[30] Google, Inc. *Google Cloud Platform* [online]. [cit. 3.5.2017]. Dostupné z: `https://cloud.google.com`

[31] Google, Inc. *Google* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.google.com/`

[32] GRIESER, F. FDM vs SLA: Explained and compared. *All3DP - The Leading 3D Printing Magazine Website* [online]. [cit. 3.5.2017]. Dostupné z: `https://all3dp.com/fdm-vs-sla/`

[33] HOEKEN, Z. Hoektronics/bumblebee: BotQueue's client Bumblebee. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: `https://github.com/kliment/Printrun`

[34] HOEKEN, Z. *Botqueue.com: Internets + Digital Fabrication = Win* [online]. [cit. 3.5.2017]. Dostupné z: `https://www.botqueue.com`

[35] HRONČOK, M. admesh/python-admesh: Cython wrapper around admesh. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: `https://github.com/admesh/python-admesh`

[36] HRONČOK, M. hroncok/octoclient: Python client library for OctoPrint REST API. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: `https://github.com/hroncok/octoclient`

[37] HRONČOK, M., ŽEHRA, M. et al. *3D tisk na FIT - Laboratoř 3D tisku* [online]. [cit. 3.5.2017]. Dostupné z: `http://3dprint.fit.cvut.cz`

[38] HÄUßGE, G. et al. *OctoPrint.org* [online]. [cit. 3.5.2017]. Dostupné z: `http://octoprint.org`

[39] *JSON* [online]. [cit. 3.5.2017]. Dostupné z: http://www.json.org

[40] KINTEL, M. *OpenSCAD* [online]. [cit. 3.5.2017]. Dostupné z: http://www.openscad.org

[41] KUIPERS, T. et al. Ultimaker/Cura - CuraEngine is a powerful, fast and robust engine for processing 3D models into 3D printing instruction for Ultimaker and other GCode based 3D printers. It is part of the larger open source project called "Cura". *GitHub*. [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/Ultimaker/CuraEngine

[42] MakerBot® Industries, LLC. Shop Replicator+ Desktop 3D Printer. *Connected 3D Printing Solutions | Makerbot* [online]. [cit. 3.5.2017]. Dostupné z: https://store.makerbot.com/printers/replicator2x/

[43] MakerBot® Industries, LLC. *Connected 3D Printing Solutions - Makerbot* [online]. [cit. 3.5.2017]. Dostupné z: https://www.makerbot.com

[44] MakerBot® Industries, LLC. *Thingiverse.com* [online]. [cit. 3.5.2017]. Dostupné z: http://www.thingiverse.com

[45] MARTIN, A. D. *ADMesh* [online]. [cit. 3.5.2017]. Dostupné z: http://www.varlog.com/admesh-htm

[46] MARTIN, A. D., HRONČOK, M., RANELLUCCI, A. et al. admesh/admesh: CLI and C library for processing triangulated solid meshes. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/admesh/admesh

[47] mrwyss. mrwyss/octoprint. *Docker Hub* [online]. [cit. 3.5.2017]. Dostupné z: https://hub.docker.com/r/mrwyss/octoprint/

[48] Nová média. 3D tisk. *3D tisk* [online]. [cit. 3.5.2017]. Dostupné z: http://www.3d-tisk.cz/3d-tisk/

[49] Nová média. PLA. *3D tisk* [online]. [cit. 3.5.2017]. Dostupné z: http://www.3d-tisk.cz/3d-tisk/

[50] ONG, S. ui/rq-scheduler: A light library that adds job scheduling capabilities to RQ. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/ui/rq-scheduler

[51] Open Source Hardware Association. *Open Source Hardware Association* [online]. [cit. 3.5.2017]. Dostupné z: https://www.oshwa.org

[52] Oracle Corporation. *MySQL* [online]. [cit. 3.5.2017]. Dostupné z: https://www.mysql.com

[53] Panse, A., Nagpal, K. et al. newman. *npm* [online]. [cit. 3.5.2017]. Dostupné z: https://www.npmjs.com/package/newman

[54] Postdot Technologies, Inc. *Postman | Supercharge your API workflow* [online]. [cit. 3.5.2017]. Dostupné z: https://www.getpostman.com

[55] Prusa Research. 3D tiskárna Original Prusa i3 MK2S. *Prusa Research s.r.o. - 3D tisk a 3D tiskárny od Josefa Průši* [online]. [cit. 3.5.2017]. Dostupné z: http://shop.prusa3d.com/cs/3d-tiskarny/53-3d-tiskarna-original-prusa-i3-mk2s.html

[56] Prusa Research. Co je RepRap? *Prusa Research s.r.o. - 3D tisk a 3D tiskárny od Josefa Průši* [online]. [cit. 3.5.2017]. Dostupné z: http://www.prusa3d.cz

[57] Prusa Research. *Prusa Research s.r.o. - 3D tisk a 3D tiskárny od Josefa Průši* [online]. [cit. 3.5.2017]. Dostupné z: http://www.prusa3d.cz

[58] Python Packaging Authority. *pip — pip 9.0.1 documentation* [online]. [cit. 3.5.2017]. Dostupné z: https://pip.pypa.io/en/stable/

[59] Python Software Foundation. PEP 3333 – Python Web Server Gateway Interface v1.0.1. *Python.org* [online]. [cit. 3.5.2017]. Dostupné z: https://www.python.org/dev/peps/pep-3333/

[60] Python Software Foundation. *Python.org* [online]. [cit. 3.5.2017]. Dostupné z: https://www.python.org

[61] Přispěvovatelé projektu RepRap. G-code. *RepRapWiki* [online]. [cit. 3.5.2017]. Dostupné z: http://reprap.org/wiki/G-code

[62] Přispěvovatelé projektu RepRap. *RepRapWiki* [online]. [cit. 3.5.2017]. Dostupné z: http://www.reprap.org

[63] Přispěvovatelé Wikipedie. Akrylonitrilbutadienstyren. *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://cs.wikipedia.org/wiki/Akrylonitrilbutadienstyren

[64] Přispěvovatelé Wikipedie. Model-view-controller. *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://cs.wikipedia.org/wiki/Model-view-controller

[65] Přispěvovatelé Wikipedie. NoSQL. *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://cs.wikipedia.org/wiki/NoSQL

[66] Přispěvovatelé Wikipedie. STL (file format). *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://en.wikipedia.org/wiki/STL_(file_format)

[67] Přispěvovatelé Wikipedie. X Window System. *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://en.wikipedia.org/wiki/X_Window_System

[68] Přispěvovatelé Wikipedie. Xvfb. *Wikipedie, otevřená encyklopedie* [online]. [cit. 3.5.2017]. Dostupné z: https://en.wikipedia.org/wiki/Xvfb

[69] RANELLUCCI, A. Slic3r/README.md alexrj/Slic3r. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/alexrj/Slic3r/blob/master/README.md

[70] RANELLUCCI, A. et al. *Slic3r* [online]. [cit. 3.5.2017]. Dostupné z: http://slic3r.org/

[71] Raspberry Pi Foundation. *Raspberry Pi - Teach, Learn and Make with Raspberry Pi* [online]. [cit. 3.5.2017]. Dostupné z: https://www.raspberrypi.org

[72] Red Hat, Inc. *Ansible is Simple IT Automation* [online]. [cit. 3.5.2017]. Dostupné z: https://www.ansible.com

[73] SALVATORE, S. et al. *Redis* [online]. [cit. 3.5.2017]. Dostupné z: http://redis.io

[74] The PHP Group. *PHP: Hypertext Preprocessor* [online]. [cit. 3.5.2017]. Dostupné z: https://secure.php.net

[75] The PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open source database* [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/ui/rq-scheduler

[76] TŘEBICKÝ, D. Files are written to the correct outputdir. by trebidav - Pull request #14 - kliment/simarrange. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: https://github.com/kliment/simarrange/pull/14

[77] Ultimaker B.V. Cura. *Ultimaker: 3D Printers* [online]. [cit. 3.5.2017]. Dostupné z: http://wiki.ultimaker.com/Cura

[78] Ultimaker B.V. *Ultimaker: 3D Printers* [online]. [cit. 3.5.2017]. Dostupné z: https://www.ultimaker.com

[79] Ultimaker B.V. Ultimaker 2+ 3D Printer. *Ultimaker: 3D Printers* [online]. [cit. 3.5.2017]. Dostupné z: https://ultimaker.com/en/products/ultimaker-2-plus

[80] WARGNY, M. Battle of the 3D technologies: SLA vs FDM vs SLS. *3D Printing Blog / Sculpteo* [online]. [cit. 3.5.2017]. Dostupné z: https://www.sculpteo.com/blog/2016/08/31/fdm-vs-sla-vs-sls-battle-of-the-3d-technologies/

[81] YANEV, K. kliment/Printrun: Pronterface, Pronsole, and Printcore - Pure Python 3d printing host software. *GitHub* [online]. [cit. 3.5.2017]. Dostupné z: `https://github.com/kliment/Printrun`

[82] YANEV, K. kliment/simarrange - STL 2D plate packer with collision simulation. *Github* [online]. [cit. 3.5.2017]. Dostupné z: `https://github.com/kliment/simarrange`

[83] Zortrax Spółka Akcyjna. 3D Printers. *Zortrax 3D Printers - Online Store* [online]. [cit. 3.5.2017]. Dostupné z: `https://store.zortrax.com/3d-printers`

Příloha **A**

Seznam použitých zkratek

3D Three-dimensional

ABS Akrylonitrilbutadienstyren

API Application Programming Interface

ČVUT České vysoké učení technické

FDM Fused deposition modeling

FIT Fakulta informačních technologií

GUI Graphical user interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol over TLS/SSL

IP Internet Protocol

JSON JavaScript Object Notation

MVC Model-View-Controller

NoSQL Not Only SQL

PHP Hypertext Preprocessor

PLA Polylactid Acid

REST Representational State Transfer

SLA Stereolitography

SSL Secure Sockets Layer

STL Stereolithography

TLS Transport Layer Security

YAML YAML Ain't Markup Language

WSGI Web Server Gateway Interface

Příloha **B**

Obsah přiložené SD karty

- `src/`
  - `impl/` ..... zdrojové kódy implementace
    - `README.md` ..... Uživatelská příručka
    - `bp/` ..... zdrojové kódy Django projektu
    - `docker/` ..... zdrojové kódy Docker
    - `tests/` ..... zdrojové kódy automatizovaných testů
  - `thesis/` ..... zdrojová forma práce ve formátu LaTeX
- `text/`
  - `BP_TREBICKY_DAVID_2017.pdf` ..... text práce ve formátu PDF