České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra softwarového inženýrství

Bakalářská práce

Webová aplikace pro sdílení asistentů pro aplikaci DevAssistant

Miroslav Hrončok

Vedoucí práce: Mgr. Bohuslav Kabrda

15. května 2014

Poděkování

Na tomto místě bych rád poděkoval a vyslovil uznání všem lidem, kteří mi pomáhali při vzniku této práce. V prvé řadě Bohuslavu Kabrdovi, vedoucímu mé bakalářské práce, za cenné připomínky k textové části i implementaci. Dále také Tomášovi Radějovi za jeho užitečné připomínky a rady, když tady Slávek nebyl.

Děkuji také všem ostatním vývojářům projektu DevAssistant a firmě Red Hat za to, že vznik této bakalářské práce umožnili. Bez DevAssistantu by tato práce neměla smysl.

Děkuji Jakubovi Jirůtkovi za poskytnutí řešení k psaní textové části této práce v jazyce Markdown.

Děkuji svým spolužákům a přátelům Jakubovi Průšovi, Ondřejovi Brémovi a Ondřejovi Ezrovi za pomoc s ostatními školními povinnostmi, kterou jsem při psání této práce výrazně potřeboval.

V neposlední řadě děkuji své partnerce Barboře Havelkové za to, že mi v průběhu psaní těchto řádků zajistila dostatečný přísun potravin.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl 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 2014 ……

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Hrončok, Miroslav. *Webová aplikace pro sdílení asistentů pro aplikaci DevAssistant*. Bakalářská práce . Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Tato bakalářská práce si klade za cíl specifikovat a implementovat formát pro šíření asistentů do aplikace DevAssistant a navrhnout a implementovat webový repozitář, kde se dají asistenty v tomto formátu sdílet s ostatními.

Klíčová slova web, vývojář, katalog, formát, API, Python, Django, YAML

Abstract

This bachelor's thesis aims to specify and implement a format for distributing assistants for the DevAssistant application and design and implement a web repository to share assistants in that format with others.

Keywords web, developer, index, format, API, Python, Django, YAML

Obsah

Seznam ukázek kódu

Seznam adresářových struktur

Seznam obrázků

Úvod

Tato práce se věnuje implementaci formátu pro šíření asistentů do aplikace DevAssistant a webového repozitáře těchto asistentů.

Aplikace DevAssistant, která pomáhá automatizovat opakující se činnost programátorů jinou než programování samotné, umožňuje uživatelům vytvářet vlastní asistenty, tedy něco jako recepty na jednotlivé automatizované činnosti. Postrádá však vhodný způsob, jak tyto recepty šířit mezi ostatní uživatele. DevAssistant a princip asistentů je podrobněji popsán v části 1.1 na straně 21.

Cílem práce je implementovat formát pro šíření asistentů do aplikace DevAssistant a webový repozitář, kde budou uživatelé moci své asistenty sdílet s ostatními.

Text této bakalářské práce je členěn do tří kapitol. V první kapitole, *Status quo a požadavky*, je vysvětlen problém, který tato práce řeší, a jsou stanoveny požadavky na jeho řešení. V druhé kapitole, *Rešerše*, je proveden průzkum existujících řešení, které by teoreticky mohly být použity k vyřešení problému. V poslední kapitole, *Implementace*, je pak popsáno samotné řešení problému. Po kapitolách následuje závěr, kde práci shrnuji a uvádím možnosti dalšího rozvoje.

Nedílnou součástí práce je zdrojový kód implementace, dostupný na přiloženém médiu.

Kapitola 1

Status quo a požadavky

V této kapitole je osvětlena problematika této bakalářské práce s důrazem na důvody vedoucí k její potřebě.

1.1 DevAssistant

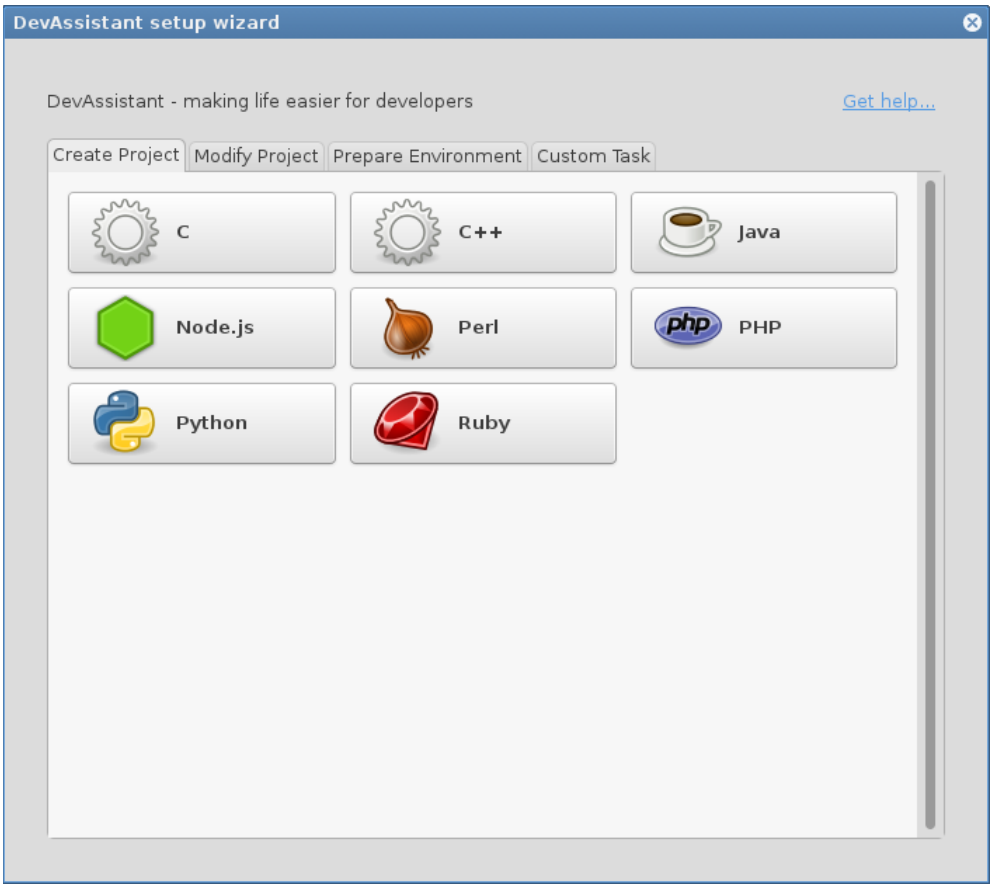
DevAssistant je aplikací, která programátorům pomáhá vytvářet nové vývojářské projekty, vydávat kód a dělat další věci, které programátory zdržují od toho nejdůležitějšího, od psaní softwaru. Nezáleží na tom, jestli jste právě objevili svět vývoje softwaru, nebo jestli už programujete dvě dekády, vždy se najde něco, s čím vám DevAssistant ulehčí život [58]. DevAssistant je možné používat z příkazové řádky nebo pomocí GUI, jak je vidět na obrázku 1.1 na straně 22.

Jedná se o svobodný software, je vydaný pod licencí GNU GPL verze 2 [14] nebo vyšší [58].

1.1.1 Asistenty

DevAssistant je vlastně nástroj umožňující spouštění takzvaných asistentů, které lze zařadit do čtyř kategorií:

- *vytvářecí asistenty* pomáhají uživateli začít nový projekt,
- *modifikační asistenty* pomáhají uživateli projekt upravit nebo do něj něco přidat,



Obrázek 1.1: Grafické rozhraní aplikace DevAssistant

- *připravovací asistenty* pomáhají uživateli zapojit se do nějakého projektu,
- *úkolové asistenty* pomáhají uživateli s věcmi, které se netýkají konkrétního projektu [33].

Jednotlivé asistenty jsou napsány ve speciálním DSL [34], který je založen na jazyce YAML.

> DSL (*Domain-specific language*) je programovací jazyk, který je prostřednictvím vhodné abstrakce a výrazového slovníku zaměřen na omezenou, konkrétní problémovou doménu. [49]

> YAML je formát pro serializaci strukturovaných dat. Výhodou tohoto formátu je, že je dobře čitelný nejen strojem, ale i člověkem. [51]

YAML soubory asistentu definují jak metadata (název asistentu, krátký popis, závislosti), tak samotný kód, který je vykonán aplikací (spuštění příkazů na příkazové řádce). Ke každému asistentu mohou patřit ikony pro zobrazení v grafickém rozhraní (jako na obrázku 1.1 na straně 22) a další soubory, které může asistent použít při vlastním spuštění. Speciálními soubory jsou pak tzv. snipety, které umožňují sdílení kódu mezi jednotlivými asistenty – jedná se, stejně jako v případě asistentů, o soubory ve formátu YAML.

V ukázce 1.1 můžete vidět příklad jednoduchého vytvářecího asistentu.

```
fullname: Argh Script Template
description: Create a template of simple script that uses argh library
project_type: [python]

dependencies:
- rpm: [python-argh]

files:
  arghs: &arghs
    source: arghscript.py

args:
  name:
    use: common_args

run:
- log_i: Hello, I'm Argh assistant...
- if $(test -e "$name"):
  - log_e: '"$name" already exists, cannot proceed.'
- cl: mkdir -p "$name"
- $proj_name~: $(basename "$name")
- cl: cp *arghs ${name}/${proj_name}.py
- cl: chmod +x *arghs ${name}/${proj_name}.py
- dda_c: "$name"
- cl: cd "$name"
- use: git_init_add_commit.run
- log_i: Project "$proj_name" has been created in "$name".
```

Ukázka kódu 1.1: Ukázka vlastního asistentu z dokumentace [32]

DevAssistant může načítat asistenty z více různých adresářů. Každý z těchto adresářů má pevně definovanou strukturu [32], která mimo jiné určuje, do které z výše uvedených kategorií asistent patří. Navíc je v této struktuře přesně definované místo pro ikony a případně další soubory asistentů.

Zatímco existuje základní sada asistentů, která je v některých případech distribuována společně s aplikací [35], existuje také možnost vytvářet a používat své vlastní asistenty [32]. Distribuování základní sady asistentů společně s aplikací přináší řadu potíží - například nutnost kompatibility se všemi platformami, na kterých aplikace běží, či snaha vyhovět požadavkům všech uživatelů (jejichž představy se pochopitelně různí).

> Jako příklad vlastního asistentu si můžeme představit vytvářecí asistent, který studentům prvního ročníku Fakulty informačních technologií ČVUT v Praze pomůže s vytvořením úloh z předmětu *Programování a algoritmizace 1*. Takový asistent by zajistil, že studenti mají k dispozici potřebné programy (kompilátor apod.), a pomohl jim zkompilovat úlohy pomocí programu make [18]. Přestože by tento asistent byl jistě přínosný pro zmíněné studenty, pro další uživatele by nedávalo smysl, aby takový asistent byl distribuovaný společně s aplikací DevAssistant.

Přestože je technicky možné distribuovat uživatelům asistent ve formě archivu, který je potřeba extrahovat na určité místo, jedná se o poměrně nepraktické řešení. Uživatelé mohou například archiv omylem rozbalit na špatné místo. Odstranění takového asistentu je problematické – je potřeba dohledat, které soubory jsou v archivu, a ze složky s asistenty je odstranit. Jednotlivé archivy spolu mohou navzájem kolidovat.

Proto vyvstává potřeba vytvořit jednotný formát distribuovatelných asistentů a místo, kde takové asistenty sdílet [31].

1.2 Požadavky

V této části jsou popsány požadavky na řešení problému popsaného na konci předchozí části.

1.2.1 Balík (dap)

Balík je jedna distribuovatelná jednotka obsahující několik asistentů. Pro naše potřeby musí splňovat následující požadavky:

- dodržení daného formátu a formy,
- obsažení distribuovaných asistentů a dalších souborů k nim náležícím,
  - společně s adresářovou strukturou
- obsažení metadat,
- možnost verzování.

V balíku by měla být nutně uložena tato metadata:

- název balíku,
- krátké slovní shrnutí obsahu,
- verze,
- licence,
- autor nebo autoři.

A volitelně pak také:

- domovská webová stránka,
- místo, kam hlásit chyby,
- delší text popisující obsah balíku a práci s asistenty v něm obsaženými.

Pro potřeby rozlišení balíku pro DevAssistant od jiných balíků (např. RPM) je tento balík pojmenován *DevAssistant Package*, zkráceně dap[1].

1.2.2 Repozitář

Repozitář je místo, kde mohou uživatelé sdílet a vyhledávat nasdílené dapy. Z důvodu uživatelské přívětivosti a dostupnosti je takovým místem webové úložiště (webová aplikace). Takové úložiště musí pro naše potřeby splňovat následující požadavky:

- Pro uživatele, který sdílí dap:
  - nahrání nového dapu,
  - nahrání nových verzí dapu,
  - kontrola správnosti[2],
  - klasifikace dapu[3],
  - přizvání dalších uživatelů ke správě dapu,

[1] Pro účely zjednodušení textu jsem se rozhodl toto slovo skloňovat podle vzoru *hrad*.
[2] Například jedná-li se skutečně o dap splňující specifikaci.
[3] Například pomocí tagů.

  - mazání jednotlivých verzí dapu nebo dapu jako celku.
- Pro uživatele, který chce dap získat:
  - vyhledávání dapů,
  - stažení dapů,
  - hodnocení dapů,
  - hlášení škodlivých dapů,
  - možnost prohledávání repozitáře a stahování dapů přímo z aplikace DevAssistant[4].
- Pro administrátora repozitáře:
  - mazání jednotlivých verzí dapu nebo dapu jako celku,
  - úprava nebo mazání uživatelských účtů a profilů,
  - vyhodnocování nahlášených dapů.
- Pro všechny uživatele:
  - autentizace,
  - úprava uživatelského profilu,
  - opuštění aplikace (smazání profilu a vytvořeného obsahu).

[4] V rámci této práce je implementováno pouze API toto umožňující.

Kapitola 2

Rešerše

V této kapitole jsou nastíněna možná řešení požadavků vyjmenovaných v části 1.2 na straně 24.

2.1 Balík (dap)

S ohledem na požadavky uvedené v části 1.2.1 na straně 24 jsou zde představeny možnosti, které vyvstávají při volbě vhodného formátu pro dap.

Vzhledem k tomu, že dap obsahuje řadu souborů v adresářové struktuře a metadata, nabízejí se dvě možnosti:

- vlastní binární formát vyvinutý pouze pro potřeby dapu,
- využití existujícího formátu pro archivaci souborů a přidání metadat
  - ve formě souboru v daném archivu,
  - nebo ve formě pozměněné hlavičky tohoto archivu.

2.1.1 Vlastní binární formát

Vlastní binární formát by dával smysl ve dvou případech:

Pokud by bylo žádoucí formát uzavřít a neprozradit nikomu, jak funguje – tato situace ale nenastává, jelikož vývoj aplikace DevAssistant a celý ekosystém kolem ní má být zcela otevřený.

Dalším důvodem je možnost navrhnout formát tak, aby byl optimalizovaný právě pro asistenty. Vzhledem k tomu, že asistenty sestávají převážně z textových souborů[5] a celková velikost jednoho dapu se v extrahované formě pohybuje řádově v desítkách kilobajtů, postrádá taková optimalizace smysl.

Implementace ryze vlastního formátu by pak přinášela mnoho problémů, například zvýšení rizika chyby či nutnost udržovat zpětnou kompatibilitu. Především by se jednalo o další kus kódu, který je potřeba napsat a udržovat – vzhledem k nulovým výhodám by se tedy jednalo o zbytečně složité řešení.

Závěr: Implementace ryze vlastního formátu je tedy pro účely dapu nevhodná.

2.1.2 Existující formát pro archivaci souborů

Zbývající možností je využití nějakého existujícího formátu určeného pro archivaci souborů. Takových formátů je mnoho a je třeba zvolit takový formát, který bude možné použít na všech platformách podporovaných aplikací. Po konzultaci s vedoucím práce jsme zvolili formát *tar.gz* [19][17], který je velmi rozšířený a otevřený.

Použitý archiv musí kromě souborů nést i metadata. Pro účely nezvyšování komplexnosti je vhodnější přidat do archivu soubor tyto metadata obsahující, než modifikovat hlavičku souboru – dap tak bude možné rozbalit obvyklým způsobem jako obyčejný *tar.gz* archiv bez ztráty těchto informací. Vzhledem k použití formátu YAML pro asistenty[6] je pak žádoucí použít stejný formát.

Závěr: Pro účely dapu bude použit archiv *tar.gz* obsahující YAML soubor s metadaty. Konkrétní implementace takového balíku je nastíněna v e-mailu, který tento formát navrhuje [31], a podrobně popsána v části 3.1 na straně 33.

2.2 Repozitář

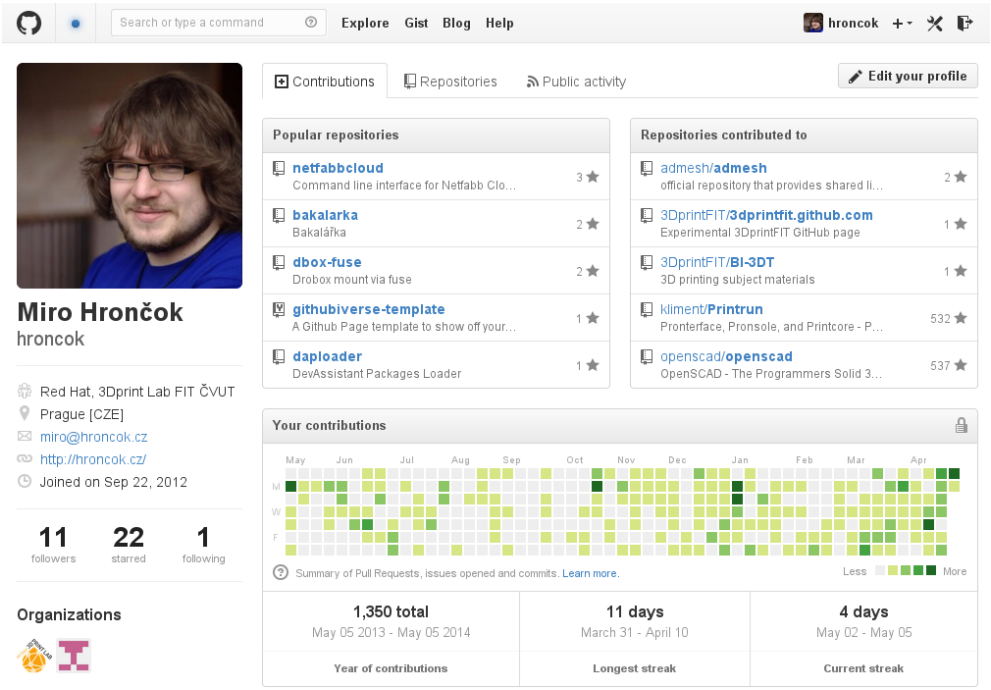
S ohledem na požadavky uvedené v části 1.2.2 na straně 25 jsou zde představeny služby a aplikace, které by teoreticky mohly být využity na sdílení dapů.

[5] YAML definice asistentů a různé šablony zdrojových kódů.

[6] Jak je popsáno v části 1.1.1 na straně 21.

2.2.1 GitHub

GitHub [21] (na obrázku 2.1) je poměrně známá a oblíbená služba určená k hostování zdrojových kódů aplikací pomocí verzovacího systému git [27]. Ačkoli jsou jednotlivé asistenty v dapu vlastně zdrojovým kódem a verzování tohoto kódu na GitHubu je pochopitelně možné, nesplňuje GitHub požadavky týkající se formátu dapu – je sice možné nahrát k repozitáři archivy, není ale možné zajistit jejich kontrolu správnosti. GitHub nadále nepodporuje uživatelské hodnocení ani hlášení škodlivého obsahu.



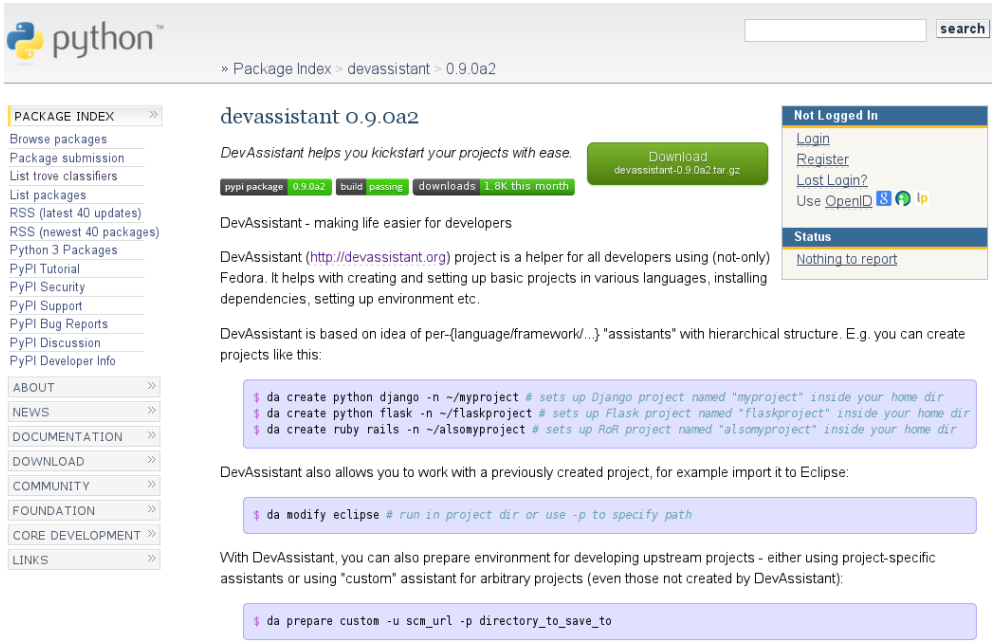
Obrázek 2.1: Snímek obrazovky ze služby GitHub

Z uživatelského hlediska pak není příliš přívětivé rozlišení, co na GitHubu je dap a co ne – procházení jednotlivých projektů na GitHubu z aplikace DevAssistant a vyhledávání dapů je tak nemožné, nebo by bylo příliš komplikované. Obsah by také nebyl pod kontrolou vývojářů DevAssistantu a případné odstranění škodlivých dapů by vyžadovalo vyjednávání s provozovateli GitHubu.

Závěr: Využití GitHubu nebo jakékoli podobné služby je tedy pro účely repozitáře dapů nevhodné.

2.2.2 PyPI

PyPI [55] (na obrázku 2.2) je repozitář modulů do programovacího jazyka Python [44]. Zdrojový kód aplikace [54] je dostupný pod permisivní licencí BSD a je tedy teoreticky možné PyPI upravit a použít jako repozitář dapů.



Obrázek 2.2: Snímek obrazovky z hlavní instance PyPI

PyPI nesplňuje některé požadavky definované v části 1.2.2 na straně 25. V první řadě je navržen na sdílení modulů do Pythonu a vyžadoval by jisté úpravy, aby do něj bylo možné nahrávat dapy. PyPI nepodporuje uživatelské hodnocení ani hlášení škodlivého obsahu. Klasifikace je možná pouze pomocí kategorií definovaných přímo v nahrávaném balíku, je možné použít pouze předem dané kategorie [53].

Využití PyPI by vyžadovalo nemalou modifikaci jeho zdrojového kódu. To přináší řadu nevýhod, především nutnost prozkoumat cizí zdrojový kód a porozumět mu a následná nutnost synchronizování vlastních změn s aktuální verzí PyPI.

Závěr: Využití PyPI je tedy pro účely repozitáře dapů možné, ale nepříliš optimální.

2.2.3 RubyGems.org

RubyGems.org [56] (na obrázku 2.3) je repozitář gemů – modulů do jazyka Ruby. Pro Ruby plní stejnou funkci jako PyPI pro Python. Zdrojový kód aplikace [57] je dostupný pod permisivní licencí MIT a je tedy teoreticky možné RubyGems.org upravit a použít jako repozitář dapů, stejně jako tomu je u PyPI.

Obrázek 2.3: Snímek obrazovky z RubyGems.org

RubyGems.org ale také nesplňuje některé požadavky definované v části 1.2.2 na straně 25. Trpí stejným neduhem jako PyPI – je navržen na sdílení modulů do Ruby a vyžadoval by jisté úpravy, aby do něj bylo možné nahrávat dapy. RubyGems.org také nepodporuje uživatelské hodnocení ani hlášení škodlivého obsahu. Další nevýhodou je, že na rozdíl od PyPI nepodporuje žádnou možnost klasifikace.

Využití RubyGems.org by taktéž vyžadovalo nemalou modifikaci jeho zdrojového kódu. To přináší stejné nevýhody jako v případě PyPI.

Závěr: Využití RubyGems.org je tedy pro účely repozitáře dapů možné, ale nepříliš optimální, dokonce méně optimální než PyPI.

2.2.4 Další podobné služby

V rámci rešerše jsem prozkoumal další podobné služby (například npm [40] nebo CPAN [43]), všechny jsou ale příliš spjaty s konkrétním obsahem, který je na ně nahráván. Proto není jejich použití pro účely repozitáře dapů optimální.

2.2.5 Vlastní řešení

Z předchozích příkladů přímo vyplývá, že nejlepším řešením je řešení vlastní. To poskytuje tyto výhody:

- plná kontrola nad obsahem (oproti službám třetích stran)
- plná kontrola nad funkcionalitou
- plná kontrola nad architekturou aplikace
- plná kontrola nad použitými technologiemi

Závěr: Pro účely repozitáře dapů je nejlepší implementovat vlastní řešení.

Kapitola 3

Implementace

V této kapitole podrobně popisuji implementaci své bakalářské práce.

3.1 Specifikace dapu

Jak bylo popsáno v částech 1.2.1 na straně 24 a 2.1 na straně 27, dap je *tar.gz* archiv obsahující soubory a metadata jednoho nebo více asistentů. V této části práce je podrobně popsána specifikace formátu dap.

Jeden dap je *tar.gz* archiv [19][17] pojmenovaný `<název>-<verze>.dap`, konkrétně pak například `python-0.8.0.dap`. Název je zde uveden i s příponou – typická přípona *tar.gz* archivu `.tar.gz` či `.tgz` je nahrazena příponou `.dap`.

Nahrazení typické přípony je provedeno proto, aby byl dap uživateli jasně identifikovatelný a aby nedocházelo k omylům při pokusu nainstalovat místo dapu běžný *tar.gz* archiv.

3.1.1 Adresářová struktura uvnitř archivu

Dap má uvnitř archivu striktně danou adresářovou strukturu 3.1 na straně 34, která odpovídá adresářové struktuře, kterou očekává DevAssistant, doplněnou o další soubory a adresáře potřebné pouze pro dap.

Asistenty a snipety mají hierarchickou strukturu. Platí, že asistent či snipet nejvyšší úrovně musí mít vždy stejný název jako dap, ve kterém je obsažen.

```
<název>-<verze>/ ..... hlavní adresář
├── assistants/ ..... adresář s asistenty
│   └── crt|mod|prep|task/ ..... adresáře s asistenty daného typu
│       ├── <název>.yaml ..... asistent první úrovně
│       └── <název>/ ..... adresář s asistenty druhé úrovně
│           ├── *.yaml/ ..... asistenty druhé úrovně
│           └── */ ..... adresáře s asistenty další úrovně (dále obdobně)
├── doc/ ..... adresář s dokumentací
│   └── <název>/
│       └── * ..... libovolné soubory a adresáře s dokumentací
├── files/ ..... adresář s dalšími soubory
│   └── crt|mod|prep|task|snippets/
│       └── <název>/
│           ├── * ..... libovolné soubory
│           └── */ ..... další úrovně
├── icons/ ..... adresář s ikonami asistentů
│   └── crt|mod|prep|task|snippets/
│       ├── <název>.png|.svg ..... ikona pro asistent první úrovně
│       └── */ ..... další úrovně
├── snippets/ ..... adresář se snipety
│   ├── <název>.yaml ..... snipet první úrovně
│   └── <název>/ ..... adresář pro snipety druhé úrovně
│       ├── *.yaml ..... snipety druhé úrovně
│       └── */ ..... adresáře se snipety další úrovně (dále obdobně)
└── meta.yaml ..... soubor s metadaty
```

Adresářová struktura 3.1: Generický dap

Asistenty a snipety dalších úrovní se mohou jmenovat libovolně[7]. Pokud je v dapu asistent nižší úrovně, vždy v něm musí být i asistent úrovně vyšší. Například pro asistent `crt/python/flask.yaml` musí existovat asistent `crt/python.yaml` – ten ale může obsahovat jen metadata a nemusí sám o sobě nic vykonávat.

Ikony (`icons`) a soubory (`files`) náleží jednotlivým asistentům a je tedy nutné je patřičně zařadit. Například asistentu `crt/python/flask.yaml` náleží ikona `crt/python/flask.svg` či `crt/python/flask.png` ze složky `icons` a žádná jiná.

Adresáře, které jsou nevyužité (jsou prázdné), by dap neměl obsahovat. Veškerý obsah hlavního adresáře kromě souboru `meta.yaml` je volitelný. Teoreticky tak může existovat dap obsahující pouze metadata, prakticky však takový dap nedává příliš smysl. Jakýkoliv obsah mimo tuto danou strukturu je nepřípustný.

Adresář `doc` byl přidán pro účely dapu, aby bylo možné společně s asistenty distribuovat i dokumentaci. Například povinně uváděný text licence, za jejíchž podmínek je obsah dapu distribuován.

Adresářová struktura 3.2 na straně 36 ukazuje příklad u reálného dapu [35].

3.1.2 Soubor meta.yaml

Soubor `meta.yaml` obsahuje metadata dapu specifikovaná v části 1.2.1 na straně 24. Z důvodů popsaných v části 2.1.2 na straně 28 se jedná o YAML soubor.

Obsahuje tyto direktivy:

authors

> Seznam autorů obsahu dapu, povinný údaj. U každého autora je nejprve uvedeno celé jméno a poté volitelně e-mailová adresa v lomených závorkách, podobně jako při e-mailové komunikaci. Zavináč v adrese lze nahradit sekvencí `_at_`.

[7] V rámci pravidel souborového systému.

```
python-0.8.0/
├── assistants/
│   └── crt/
│       ├── python.yaml
│       └── python/
│           ├── django.yaml
│           ├── flask.yaml
│           ├── gtk3.yaml
│           └── lib.yaml
├── doc/
│   └── python/
│       └── LICENSE
├── files/
│   └── crt/
│       └── python/
│           └── lib/
│               └── setup.py…
├── icons/
│   └── crt/
│       └── python.svg
└── meta.yaml
```

Adresářová struktura 3.2: Reálný dap

Příklady validních autorů:

- Miroslav Hrončok <miroslav.hroncok@fit.cvut.cz>
- Miroslav Hrončok <miroslav.hroncok_at_fit.cvut.cz>
- Miroslav Hrončok
- Никола I Петровић-Његош

license

Licence obsahu dapu, povinný údaj. Používají se specifikátory licencí z RPM balíčků distribuce Fedora [9][10]. Je možné použít pouze ve Fedoře povolené licence [9][10], které zaručují svobodné šíření obsahu. Tagy lze kombinovat pomocí slov `and` a `or` – k tomuto

účelu je možné použít i závorky a vyhodnocení probíhá podobně jako v jiných logických výrazech. Slovo `and` se používá v případě, že část obsahu je šířena pod jednou a část pod druhou licencí, slovo `or` se používá, pokud je možné si licenci vybrat [7].

V případě zvýšené poptávky po možnosti uvedení nesvobodné licence [8], je možné později povolit v metadatech i tuto variantu. Pro účely repozitáře dapů ale bude nadále možné nahrávat jen svobodný obsah.

Příklady validních specifikátorů licence:

- AGPLv3
- GPL+ or Artistic
- GPLv2+
- LGPLv2+ and LGPLv2 and LGPLv3+ and (GPLv3 or LGPLv2) and (GPLv3+ or LGPLv2) and (CC-BY-SA or LGPLv2+) and (CC-BY-SA or LGPLv2) and CC-BY and BSD and MIT and Public Domain

package_name

Název dapu, povinný údaj. Smí obsahovat malá písmena obsažená v tabulce ASCII[8], číslice a znaky podtržítko (_) a spojovník (-), avšak první a poslední znak smí být pouze písmeno nebo číslice.

Délka názvu není nijak omezena, přesto není praktické vybírat extrémně dlouhé názvy, například kvůli možným omezením souborového systému, ale také kvůli uživatelské (ne)přívětivosti.

Příklady validních názvů:

- python
- foo
- fit-cvut
- 666
- pivo
- blok_6

[8]Tedy písmena anglické abecedy.

summary

Krátký popis dapu, povinný údaj. Slouží k rychlému obeznámení uživatele s obsahem dapu, měl by být napsaný v anglickém jazyce.

Příklady vhodných popisů:

- Python assistants for creating Flask, Djnago and GTK3 apps or pure Python libraries
- Set of assistants for students of FIT CTU in Prague

version

Verze dapu, povinný údaj. Verze musí obsahovat alespoň jedno nezáporné číslo a může obsahovat neomezeně dalších nezáporných čísel oddělených tečkou. Číslice se uvádí bez nadbytečných úvodních nul. Za číselnou verzí může být bez oddělení uveden text `dev`, `a`, nebo `b` značící v uvedeném pořadí blíže nespecifikovanou vývojovou verzi, alfa verzi a beta verzi.

Příklady verzí seřazené od nejstarší po nejnovější:

- 0.99
- 1
- 1.0.5
- 1.1dev
- 1.1a
- 1.1b
- 1.1

Následují nepovinné údaje.

bugreports

Místo, kam reportovat chyby, nepovinný údaj. Buďto validní URL podle stejných podmínek jako v případě *homepage* (popsáno dále), nebo e-mailová adresa. Zavináč v adrese lze nahradit sekvencí `_at_`, stejně jako v případě *authors*.

Příklady validních záznamů pro položku *bugreports*:

- https://github.com/hroncok/dap-travis/issues
- miroslav.hroncok_at_fit.cvut.cz
- devassistant@lists.fedoraproject.org

description

Delší, volitelný popis obsahu dapu. Slouží k podrobnějšímu obeznámení uživatelů s obsahem dapu a způsobu využití asistentů v něm obsažených. Obsah by se dal přirovnat k běžnému obsahu souboru README a měl by být v anglickém jazyce. V popisu lze použít formátování pomocí Markdownu [22].

Příklad je uveden v ukázce 3.2 na straně 40.

homepage

Webová stránka dapu, nepovinný údaj. Zadána ve formě platného URL. Je možné použít adresy využívající protokoly HTTP, HTTPS a FTP. Není možné použít adresy s IP adresou místo jména domény. Jako webová stránka dapu může sloužit například odkaz na repozitář s kódem.

Příklady validních URL webových stránek:

- https://github.com/hroncok/dap-travis
- http://users.fit.cvut.cz/~hroncmir/dap-travis
- ftp://users.fit.cvut.cz/hroncmir/dap-travis

V ukázce 3.1 můžete vidět příklad souboru `meta.yaml`, který obsahuje pouze povinné údaje, a v ukázce 3.2 na straně 40 pak příklad doplněný o údaje nepovinné.

```
package_name: travis
version: 0.0.1dev
license: ISC
authors: [Miro Hrončok <miro@hroncok.cz>]
summary: Adds Travis CI to your projects
```

Ukázka kódu 3.1: Minimální příklad souboru meta.yaml

```
package_name: travis
version: 0.0.1dev
license: ISC
authors: [Miro Hrončok <miro@hroncok.cz>]
homepage: https://github.com/hroncok/dap-travis
summary: Adds Travis CI to your projects
bugreports: https://github.com/hroncok/dap-travis/issues
description: |
    This mod assistant allows
    [Travis CI](https://travis-ci.org/) for you project.
    Your project has to be already on GitHub.

    Run `da mod travis [-p <path>]` and DevAssistant will
    check if the current working directory or path specified
    by `-p` is a git repository with GitHub *remote* specified.

    If so, it will allow Travis checks for your project and add
    `.travis.yml` (if not already present) with default content
    for language guessed from `.devassistant`, or blank when
    `.devassistant` is not available.

    ### More options:

     * `--url` - changes Travis CI URL, default https://travis-ci.org/
```

Ukázka kódu 3.2: Soubor meta.yaml s využitím všech volitelných položek

3.2 Knihovna pro načítání metadat dapů

Strojově lze číst metadata dapu následujícím postupem:

1. extrahovat dap jako *tar.gz* archiv,
2. najít `meta.yaml` v jediné složce, která byla vyextrahována,
3. parsovat YAML a číst potřebné údaje.

Rozhodl jsem se tedy napsat knihovnu, která umožní tento postup automatizovat a kontrolovat, že načítaný dap splňuje specifikaci. Knihovna umožní:

- načítat dapy a programově přistupovat k jejich metadatům,
- rozbalovat dapy na určité místo,
- kontrolovat správnost dapu a vypisování chyb a varovní v případě nesprávných dat.

Knihovnu jsem nazval *daploader*.

3.2.1 Použité technologie

Vzhledem k tomu, že aplikace DevAssistant je napsána v programovacím jazyce Python [44] a je žádoucí, aby knihovna šla použít přímo z této aplikace, zvolil jsem také programovací jazyk Python. Implementace v jiném jazyce by sice byla možná, ale bylo by pak nutné řešit komunikaci této knihovny s aplikací DevAssistant pomocí nějaké mezivrstvy [3], což by bylo zbytečně komplikované.

Protože DevAssistant je napsán tak, aby jej bylo možné interpretovat jak Pythonem ve verzi 2, tak Pythonem ve verzi 3, a protože je to považováno za vhodné [44], podporuje daploader taktéž obě používané verze Pythonu.

Pro parsování souboru `meta.yaml` jsem použil modul PyYAML [63], jinak jsem si vystačil se standardními moduly obsaženými v distribuci Pythonu.

Zvolil jsem metodu TDD a pro testování jsem použil nástroj pytest [36].

> Programování řízené testy (z anglického *Test-driven development* – TDD) je přístup k vývoji softwaru, který je založen na malých, stále se opakujících krocích, vedoucích ke zefektivnění celého vývoje. Prvním krokem je definice funkcionality a následné napsání testu, který tuto funkcionalitu ověřuje. Poté přichází na řadu psaní kódu a nakonec úprava tohoto kódu. [47]

3.2.2 Načtení dapu

Daploader poskytuje třídu *Dap*. Její konstruktor načte dap a pokusí se z něj získat obsah souboru `meta.yaml` – pokud se načtení nepodaří (nejedná se o *tar.gz* archiv nebo v archivu není právě jeden soubor `meta.yaml`), knihovna vyvolá výjimku. V případě správného načtení jsou jednotlivé položky ze souboru `meta.yaml` k dispozici ve formě asociativního pole[9].

3.2.3 Kontroly

Třída *Dap* obsahuje metodu `check()`, která spouští rutinu kontrol. Pokud některé kontroly selžou, chyby nebo varování jsou nahlášeny uživateli do

[9] V Pythonu nazvaného slovník – *dict*.

zvoleného místa výstupu (výchozí je standardní chybový výstup), případně je vyvolána výjimka.

Chybu vyvolá:

- chybějící povinná položka v souboru `meta.yaml`,
- nevalidní hodnota položky v souboru `meta.yaml`,
- přebytečná položka v souboru `meta.yaml` neobsažená ve specifikaci,
- špatně pojmenovaný hlavní adresář,
- soubor nebo adresář mimo hlavní adresář,
- špatně pojmenovaný soubor s dapem,
- soubor nebo adresář mimo povolenou strukturu,
- adresář s asistenty nebo snipety nižší úrovně bez patřičného asistentu nebo snipetu vyšší úrovně.

Varování vyvolá:

- dap obsahující pouze soubor `meta.yaml`,
- prázdný adresář v dapu,
- chybějící ikona pro asistent nebo snipet,
- vícenásobná ikona pro jeden asistent nebo snipet,
- přebytečná ikona pro neexistující asistent nebo snipet,
- přebytečné soubory pro neexistující asistent nebo snipet.

Kromě metody `check()` lze použít i program `daplint` (součást knihovny), který umožní provádět kontroly z příkazové řádky jako v ukázce 3.3.

```
$ daplint foo-1.0.0.dap
foo-1.0.0.dap: ERROR: out is outside of foo-1.0.0 top-level directory
foo-1.0.0.dap: WARNING: Only meta.yaml in dap
```

Ukázka kódu 3.3: Příklad výstupu programu daplint

3.2.4 Další funkce

Třída *Dap* také obsahuje metodu na extrahování dapu na určité místo. Knihovna nabízí porovnávací funkci verzí, kterou je možno použít například jako v ukázce 3.4.

```
from daploader import dapver
versions = ['1.0', '1.0.5', '1.1dev', '1.1a', '1.1', '1.1.1', '1.2']
assert versions == sorted(versions, cmp=dapver.compare)
```

Ukázka kódu 3.4: Použití porovnávače verzí z Pythonu 2

3.2.5 Testy

Díky zvolené metodě TDD [47] mají všechny kontroly testy, které ověřují, že správně fungují. Testy jsou součástí zdrojových kódů knihovny, které naleznete na přiloženém médiu.

3.2.6 Licence

Knihovna daploader je dostupná pod licencí GNU GPL verze 2 [14] nebo vyšší. Plné znění licence je součástí zdrojových kódů knihovny, které naleznete na přiloženém médiu.

Tato licence byla zvolena podle licence aplikace DevAssistant, aby bylo v budoucnu možné libovolně přesouvat kód mezi knihovnou daploader a DevAssistantem, případně do aplikace knihovnu začlenit.

3.2.7 Instalace

Knihovna daploader je dostupná v repozitáři PyPI [55] a je možné ji nainstalovat například pomocí programu `pip` [52] (v ukázce 3.5).

```
pip install daploader
```

Ukázka kódu 3.5: Instalace knihovny daploader

3.3 Repozitář

V této části je popsána implementace repozitáře dapů nazvaná *Dapi – DevAssistant Package Index*.

3.3.1 Použité technologie

Backend

Aby bylo jednoduché použít vytvořenou knihovnu daploader popsanou v části 3.3.4 na straně 55, zvolil jsem programovací jazyk Python [44]. Opět platí, že použít jiný programovací jazyk by bylo možné, ale zbytečně komplikované kvůli nutnosti přidat mezivrstvu [3].

Pro vytváření webových aplikací v programovacím jazyce Python existuje celá řada frameworků [4]. Výběr mezi nimi je záležitostí poskytovaných funkcí, množství dokumentace, dostupných modulů, ale i osobních preferencí.

Zvolil jsem framework Django [12][26], který poskytuje softwarovou architekturu MVC.

> Model-view-controller (MVC) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. [50]
>
> V terminologii Djanga se můžete setkat s pojmem MTV (*model*, *template*, *view*), který označuje stejnou technologii s jiným označením základních stavebních kamenů [13].

Pro Django navíc existuje celá řada doplňků, modulů do Pythonu, které umožňují řešit některé požadavky na repozitář specifikované v části 1.2.2 na straně 25:

- django-taggit [20] pro klasifikaci pomocí tagů,
- Haystack [37] a Whoosh [28] pro fulltextové vyhledávání,
- Django REST framework [29] pro API,
- Python Social Auth [1] pro autentizaci pomocí služeb třetích stran.

Použil jsem další moduly do Pythonu k řešení vzniklých problémů nespecifikovaných v požadavcích:

- South [23] pro migraci dat při změně modelů,
- Django Simple Captcha [6] pro zobrazení CAPTCHA u formulářů dostupných pro nepřihlášené uživatele,

- django-gravatar2 [66] pro zobrazení Gravatarů [5] uživatelů,
- markdown2 [39] pro zobrazení dlouhých popisů dapů.

A samozřejmě knihovnu daploader.

Aplikace je psaná pro Python 2.7, na podporu Pythonu 3 nebyl kladen důraz. Přesto je kód psán tak, aby případné portování na Python 3 probíhalo bez závažných problémů.

Databáze

Framework Django umožňuje použít databázové systémy SQLite [64], MySQL[10] [41] nebo PostgreSQL [45]. Díky dostatečné úrovni abstrakce z hlediska programování na použitém databázovém systému nezáleží.

Pro vývoj a běh aplikace na vlastním systému jsem použil SQLite – odlehčenou databázi uloženou v jednom souboru vhodnou právě pro tento účel [65].

Pro nasazení aplikace jsem pak použil PostgreSQL, které poskytuje oproti MySQL řadu výhod, především absolutně otevřený vývoj a lépe zajištěnou integritu dat [65] nebo speciální možnosti pro ukládání NoSQL dat [24].

Frontend

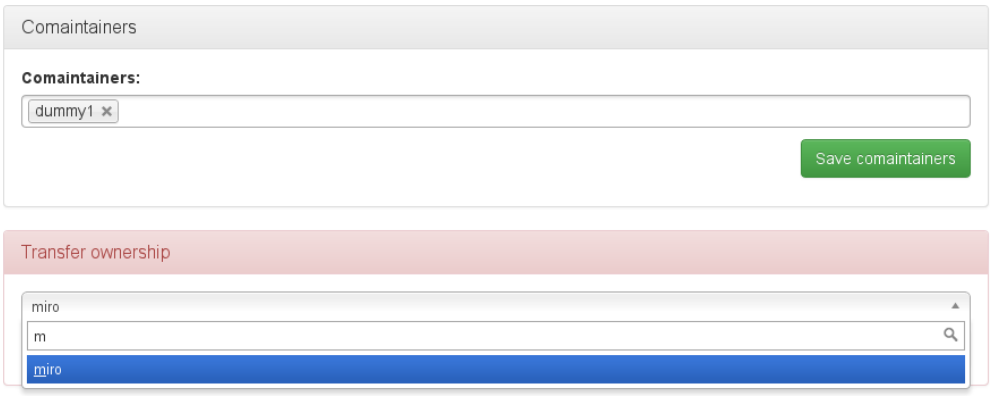
V části, která interaguje s uživatelem, jsem použil knihovny jQuery [30] a Bootstrap [42], abych se vzhledem přiblížil k webové stránce aplikace DevAssistant [58] a abych mohl řešit layout webového rozhraní dostatečně progresivně. Dále jsem použil javascriptové knihovny chosen [25] pro uživatelsky přívětivější formuláře (na obrázku 3.1 na straně 46) a toc [2] pro vygenerování obsahu stránky.

Nasazení

Aplikaci jsem nasadil na cloudovou platformu OpenShift [60], která mi umožní:

- nasazení pomocí gitu [27],
- automatickou instalaci závislostí z PyPI [55],

[10] Případě kompatibilní náhrady jako MariaDB [38].



Obrázek 3.1: Knihovna chosen v praxi

- použití Pythonu [44] a PostgreSQL [45],
- pravidelné spouštění skriptů pomocí cronu.

  Cron je softwarový démon, který v operačních systémech automatizovaně spouští v určitý čas nějaký příkaz, resp. proces (skript, program apod.). Jedná se vlastně o specializovaný systémový proces, který v operačním systému slouží jakožto plánovač úloh, jenž umožňuje opakované spouštění periodicky se opakujících procesů (např. noční běhy dávkových úloh při hromadném zpracování dat apod.). [46]

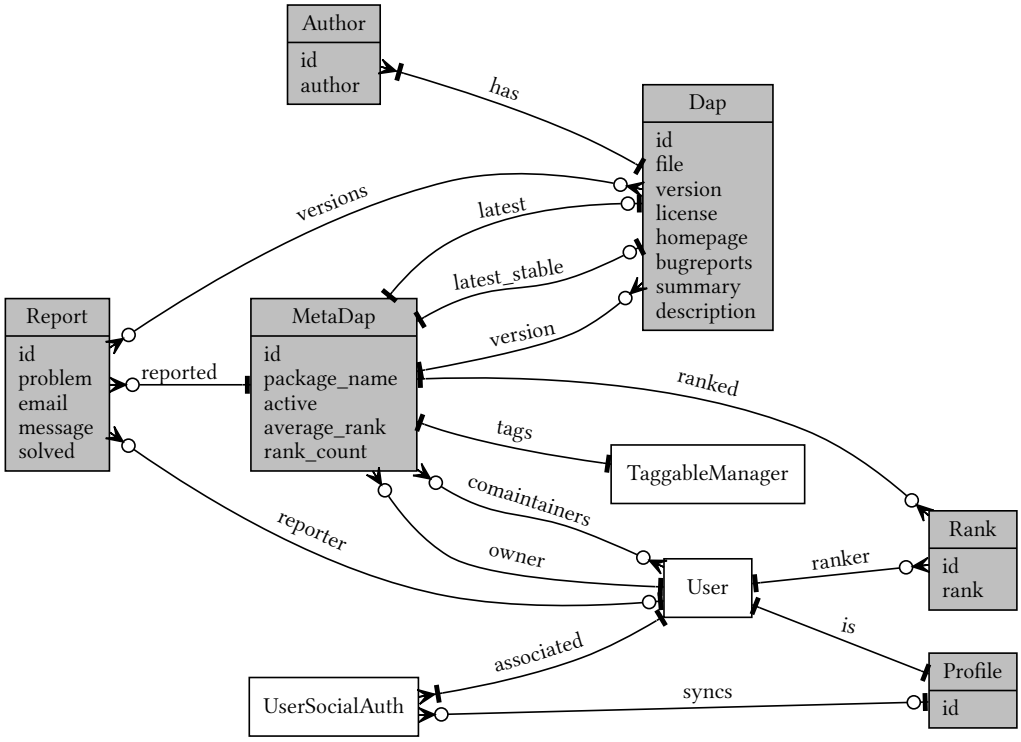
Základní využití OpenShiftu je navíc zcela zdarma.

3.3.2 Architektura

Dapi obsahuje několik modelů reprezentující dapy, uživatele apod. Jejich vztahy jsou znázorněny na obrázku 3.2 na straně 47. Bílé obdélníčky znázorňují modely převzaté z Djanga nebo z některých použitých modulů.

Author

Author představuje jednoho autora dapu. Je vázán na konkrétní verzi dapu, tedy na model *Dap*.



Obrázek 3.2: ERM diagram aplikace

Dap

Dap představuje dap v jedné konkrétní verzi. Uchovává metadata dapu (kromě názvu), odkaz na *MetaDap* a cestu k souboru s dapem.

MetaDap

MetaDap uchovává informace o dapu, bez ohledu na jeho konkrétní verzi. Tedy název dapu, vlastníka, spoluvlastníky, hodnocení, tagy, hlášení a informace o tom, je-li *MetaDap* aktivní[11]. Dále obsahuje odkaz na poslední a poslední stabilní verzi dapu, pokud je k dispozici.

Informace o celkovém počtu hodnocení a průměrném hodnocení je uchována v databázi a přepočítává se, až když dojde k nějaké změně. Jedná se o vědomé porušení třetí normální formy [48]. Je to proto, aby se při každém načtení

[11]Neaktivní MetaDap plní roli smazaného dapu bez nutnosti ho úplně smazat.

stránky s dapem nemusela z databáze načítat všechna jeho hodnocení. Stejným způsobem fungují odkazy na poslední a poslední stabilní verzi dapu.

Profile

Django poskytuje model *User* reprezentující uživatele aplikace. Do tohoto modelu není možné přidávat atributy, proto existuje model *Profile*, který se váže právě na jednoho uživatele a který uchovává dodatečné informace [13].

V případě Dapi je to odkaz na služby, pomocí kterých když se uživatel přihlásí, tak přepíší data uživatele (jméno a e-mailovou adresu). To je nutné proto, že uživatel se může přihlašovat přes více služeb, které mohou poskytovat různé údaje. Takto si může vybrat, které údaje mají platit. Ve výchozím stavu takto přepisuje data první použitá služba.

Profile nadále může obsahovat další data chybějící v modelu *User*, pokud by bylo rozhodnuto, že to je potřeba – například telefonní číslo apod. Podobně jsou metody, které by se normálně implementovaly v modelu *User*, implementovány v modelu *Profile*.

Rank

Rank představuje hodnocení jednoho uživatele jednoho *MetaDapu*. Uchovává odkaz na *MetaDap* a uživatele (model *User* z Djanga) a výši hodnocení (1 až 5 „hvězdiček“).

Report

Report představuje hlášení o škodlivém dapu. Uchovává se odkaz na *MetaDap*, druh škodlivosti, obsah hlášení, informace, jestli je hlášení vyřízeno, a odkaz na uživatele, který dap nahlásil, případě volitelně e-mailová adresa, pokud šlo o nepřihlášeného uživatele a ten ji vyplnil. Dále je uchováván odkaz na příslušné verze (modely *Dap*), pokud byly při hlášení uvedeny.

V aplikaci existují tyto druhy škodlivosti:

- legální problém (problémy s autorským právem, nesvobodný nebo patentovaný obsah),
- malware (škodlivý kód),

- nenávistný nebo jinak nevhodný obsah (rasismus, sexismus, podněcování k nenávisti, pornografie apod.),
- spam.

3.3.3 Stránky

Uživatel interaguje s aplikací prostřednictvím jednotlivých stránek. Zde je popsán jejich obsah, případně zvolený způsob implementace, pokud není triviální.

Součástí všech stránek je navigační prvek – horní lišta obsahující odkazy na jednotlivé části aplikace, dokumentaci a přihlášení či odhlášení a vyhledávací políčko. V případě, že je uživatel přihlášen, obsahuje navigace odkaz na stránku s jeho profilem a na stránku, kde může svůj profil upravit.

Hlavní stránka aplikace

Hlavní stránka aplikace je místem, kudy uživatel na stránku vstupuje, pokud nepoužil odkaz vedoucí na konkrétní obsah. Obsahuje velmi stručnou informaci o aplikaci a seznam nejlépe hodnocených, nejčastěji hodnocených a nejnověji nahraných dapů.

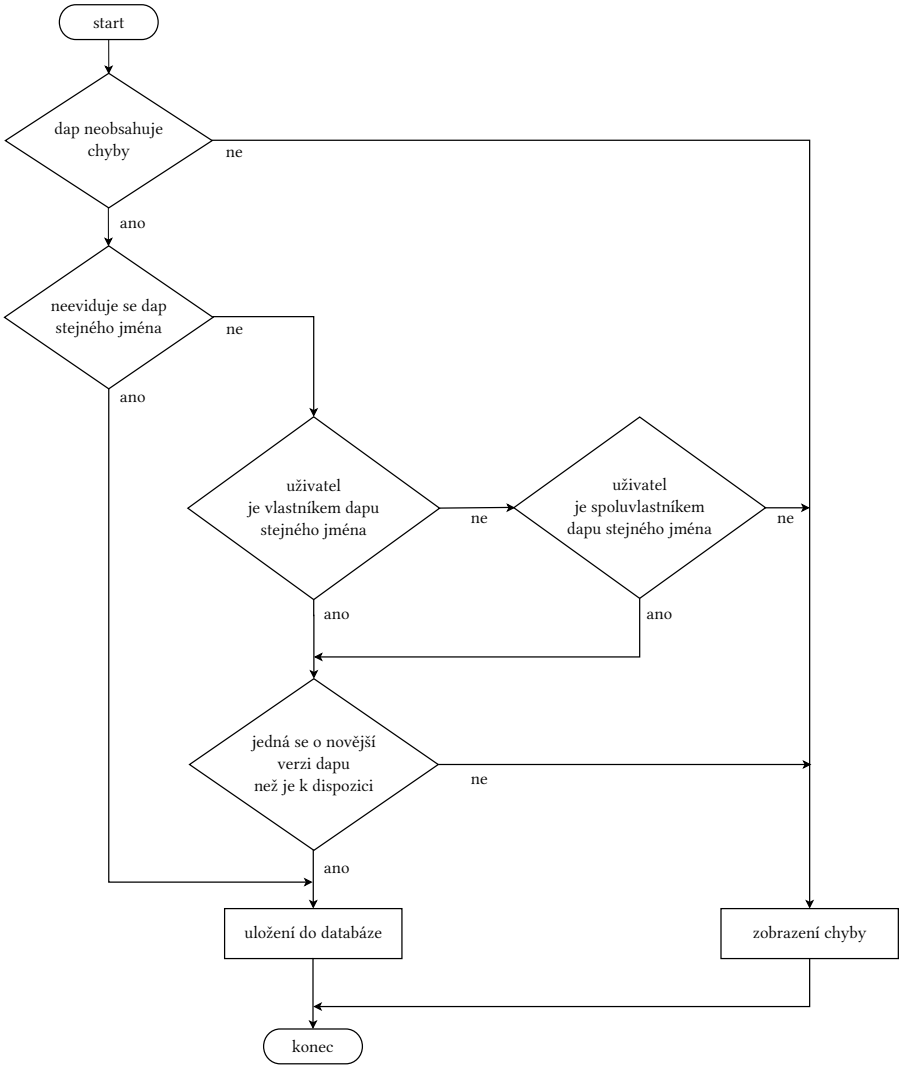
Přihlašovací stránka

Přihlašovací stránka nabízí uživateli přihlášení pomocí služeb třetích stran, konkrétně dle zadání GitHub [21] a Fedora [59]. Uživateli je zobrazena v případě, že se nepřihlášený pokusí zobrazit stránku, kde je přihlášení vyžadováno.

Analogicky existuje stránka odhlašovací, ta však nemá žádný obsah a odhlášeného uživatele přesměruje na hlavní stránku.

Nahrání dapu

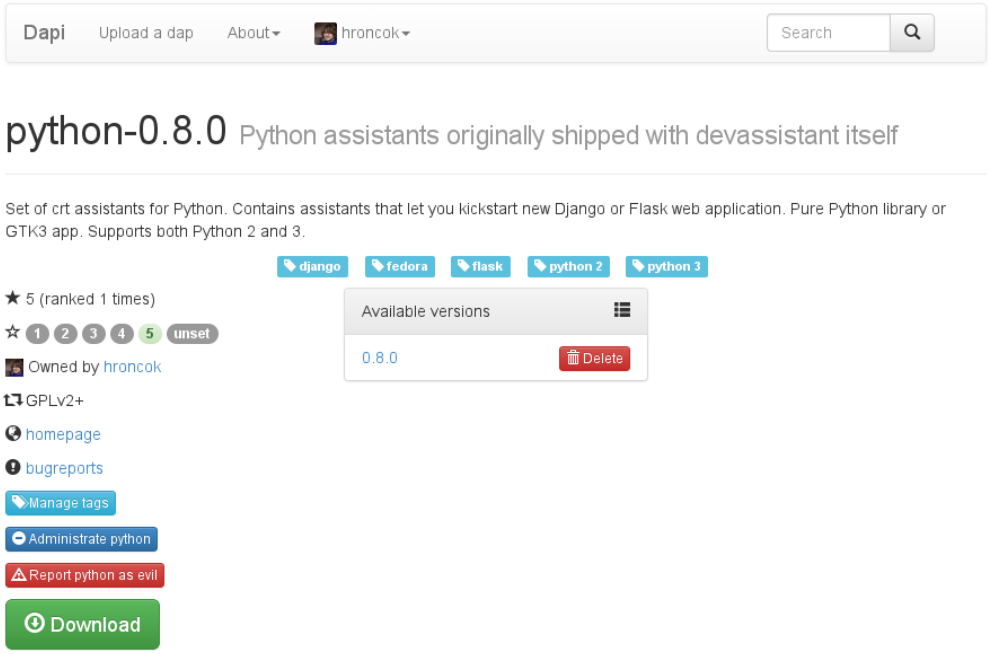
Stránka s formulářem sloužícím k nahrání dapu. Je zobrazena pouze přihlášeným uživatelům. Po nahrání je dap zkontrolován knihovnou daploader a v případě, že na to má uživatel oprávnění, je zařazen do repozitáře – tento proces je zobrazen na obrázku 3.3.



Obrázek 3.3: Vývojový diagram nahrávání dapu

Zobrazení dapu

Stránka s detaily dapu – je zobrazena na obrázku 3.4. Zobrazuje informace z modelu *Dap* a *MetaDap*. Pokud není určeno specificky, jakou verzi zobrazit, je zobrazena poslední stabilní verze. Pokud žádná stabilní verze není dostupná, je zobrazena poslední vývojová. Pokud dap neobsahuje žádnou verzi, jsou zobrazeny pouze informace z modelu *MetaDap*.



Obrázek 3.4: Stránka s dapem

Vlastníkovi a spoluvlastníkovi dapu a administrátorovi repozitáře je umožněno mazat jednotlivé verze dapu. Všem návštěvníkům jsou v některých případech zobrazena varování:

- Toto není nejnovější stabilní verze dapu.
- Toto není nejnovější verze dapu.
- Toto není stabilní verze dapu.
- Tento dap má nevyřízená hlášení.
- Tento dap není aktivní.

Administrace dapu

Stránka je dostupná pouze vlastníkovi dapu, případně administrátorovi repozitáře. Umožňuje spravovat spoluvlastníky, označit dap jako neaktivní, úplně jej smazat, případně darovat jinému uživateli.

Správa tagů dapu

Vlastníkovi a spoluvlastníkovi dapu a administrátorovi repozitáře je zde umožněno nastavit tagy pro dap.

Opuštění dapu

Spoluvlastníkovi dapu je zde umožněno na pozici spoluvlastníka rezignovat.

Nahlášení dapu

Návštěvník stránky zde může nahlásit dap jako škodlivý. Pro nepřihlášené uživatele je zde možnost zadat i e-mailovou adresu a nutnost opsat CAPTCHA kód z obrázku. Po nahlášení je odeslán informační e-mail vlastníkovi dapu a administrátorovi či administrátorům repozitáře.

Seznam hlášení dapu

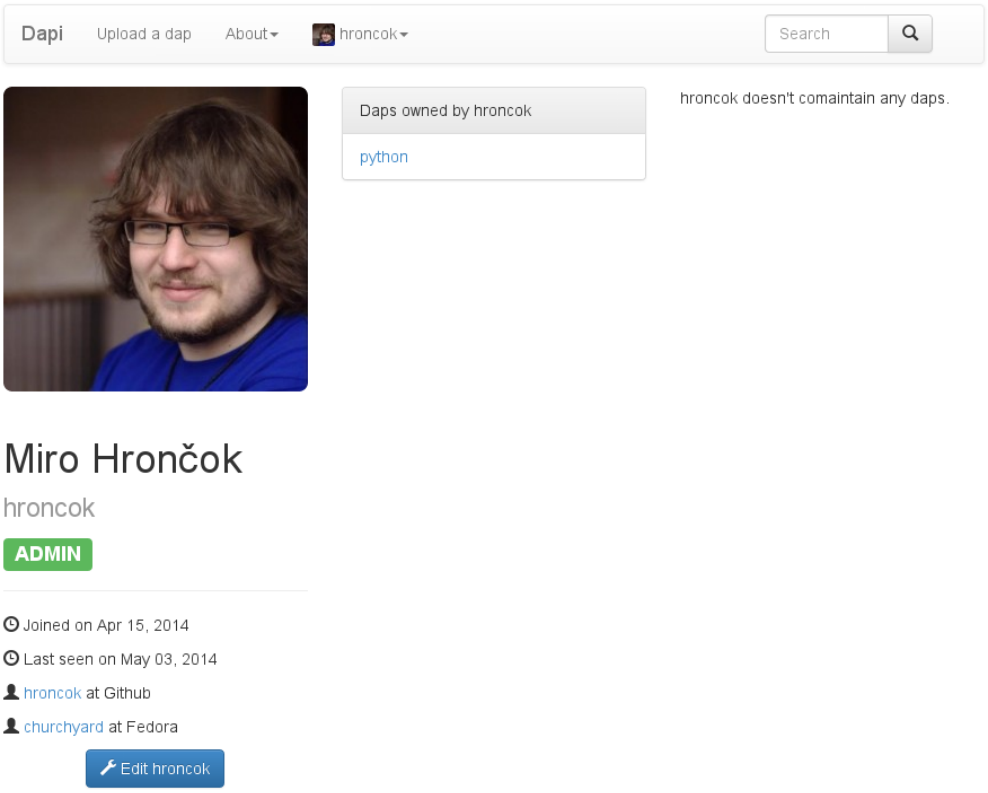
Seznam nezpracovaných hlášení daného dapu. Administrátor může jednotlivá hlášení označit jako zpracovaná/vyřízená – tato hlášení pak nikdo jiný nevidí.

Zobrazení tagu

Stránkovaný seznam všech dapů se zvoleným tagem – je zobrazen při kliknutí na tag. Obsahuje název, krátký popis, vlastníka a hodnocení dapu. Název dapu slouží jako odkaz na jeho detailní zobrazení.

Zobrazení uživatele

Stránka s profilem uživatele – je zobrazena na obrázku 3.5. Obrázek uživatele je načten ze služby Gravtar [5].



Obrázek 3.5: Stránka s uživatelským profilem

Úprava uživatele

Úprava uživatelského profilu – je dostupná pouze danému uživateli nebo administrátorovi repozitáře. Umožňuje změnu uživatelských údajů (přihlašovacího jména, křestního jména, příjmení, e-mailové adresy), asociování a deasociování jednotlivých služeb pro přihlášení, správu služeb, které přepisují uživatelské údaje při přihlášení, a smazání uživatele, pokud nevlastní žádné dapy.

Výsledky vyhledávání

Stránkovaný seznam s výpisem všech dapů odpovídajících hledané frázi, je zobrazen stejně jako seznam dapů se zvoleným tagem. Vyhledávání je realizováno přes souborovou databázi Whoosh [28] – při každém uložení nebo smazání dapu je tato databáze aktualizována. V případě většího provozu na webové aplikaci je možné databázi místo toho aktualizovat asynchronně pomocí služby cron [46].

Podmínky služby

Statická stránka zobrazující podmínky použití služby. Pro větší přehlednost je zobrazen obsah pomocí javascriptové knihovny toc [2]. Text podmínek dodal Richard Fontana, zaměstnanec firmy Red Hat [61].

Podmínky zajišťují, že provozovatel služby nezískává žádná speciální práva na obsah nahraný uživateli (kromě práva tento obsah zobrazit a šířit uživatelům), ale také za daný obsah nepřejímá zodpovědnost.

3.3.4 API

Dapi nabízí API pro práci s repozitářem například z aplikace DevAssistant. Na základě pokynů od vedoucího práce jde zatím pomocí API provádět jen neautorizované čtecí operace. Tedy:

- získat seznam uživatelů,
- získat podrobnosti uživatele,
- získat seznam *MetaDapů*,
- získat podrobnosti *MetaDapu*,
- získat seznam *Dapů*,
- získat podrobnosti *Dapu*,
- získat výsledky vyhledávání.

API je realizováno pomocí modulu Django REST framework [29]. Díky tomu je možné API procházet přímo v prohlížeči v rámci aplikace Dapi. Použití Django REST frameworku také zjednodušuje budoucí možnou implementaci zapisovací části API.

Jednotlivé objekty a jejich seznamy jsou serializovány pomocí YAMLu – to je sice pro API poměrně netradiční řešení[12], ale vzhledem k tomu, že DevAssistant již závisí na knihovnách, které YAML parsují, je YAML nejvhodnější volbou. Příklad serializovaného *Dapu* můžete vidět v ukázce 3.6 na straně 55 – tam, kde je to vhodné, jsou serializována i data *MetaDapu* (například název dapu).

[12]Tradičnější je použití JSONu nebo XML.

Při práci na Dapi jsem přispěl i do projektu Django REST framework, aby bylo možné vhodně a čitelně serializovat v YAMLu i řetězce se znaky mimo ASCII tabulku, například moje příjmení.

```
$ curl http://dapi.devassistant.org/api/daps/python-0.8.0/
active: true
api_link: http://dapi.devassistant.org/api/daps/python-0.8.0/
authors: [Bohuslav Kabrda <bkabrda@redhat.com>]
bugreports: https://github.com/bkabrda/devassistant/issues
description: 'Set of crt assistants for Python.

  Contains assistants that let you kickstart new Django or
  Flask web application. Pure Python library or GTK3 app.

  Supports both Python 2 and 3.'
download: http://dapi.devassistant.org/download/python-0.8.0.dap
homepage: https://github.com/bkabrda/devassistant-assistants-fedora
human_link: http://dapi.devassistant.org/dap/python/0.8.0/
id: 4
is_latest: true
is_latest_stable: true
is_pre: false
license: GPLv2+
metadap: http://dapi.devassistant.org/api/metadaps/python/
package_name: python
reports: 0
summary: Python assistants originally shipped with devassistant itself
version: 0.8.0
```

Ukázka kódu 3.6: Příklad použití API

Testy

Základní funkcionalita API je při každém nasazení aplikace na OpenShift automaticky ověřena službou Runscope [62].

Daploader

Do knihovny daploader jsem doplnil funkce na práci s API Dapi. Při kontrole je možné volitelně rozhodnout, zda kontrolovat i přítomnost dapu stejného jména na Dapi. Kontrola v případě pozitivního nálezu vyvolá varování.

Zároveň jsem knihovnu rozšířil o program `dapi`, který umožňuje zobrazovat data z Dapi a instalovat jednotlivé dapy do adresářů, kde je DevAssistant očekává. Jedná se ale pouze o technologické demo - metadata z instalovaných dapů nejsou nikam ukládána a není tedy možné sledovat, zda jsou nainstalované dapy aktuální, případně jestli byly nainstalovány z Dapi, či nikoli. Program `dapi` umožňuje:

- Vyhledávat dapy
- Zobrazit informace o dapu v poslední nebo konkrétní verzi
- Nainstalovat dap poslední nebo konkrétní verze
- Aktualizovat[13] dap poslední nebo konkrétní verze
- Nainstalovat nebo aktualizovat dap ze souboru
- Aktualizovat všechny nainstalované dapy
- Odinstalovat dap
- Zobrazit všechny nainstalované dapy

Příklad práce s `dapi` můžete vidět v ukázce 3.7 na straně 57. Nápověda je pak součástí programu (přepínač `--help`).

3.3.5 Licence

Z důvodu licenční kompatibility s knihovnou daploader musí být kód Dapi vydán pod licencí kompatibilní s GNU GPL verze 2 [14] nebo vyšší. Nabízí se použití stejné licence, zvolil jsem ale raději licenci GNU AGPL verze 3 [15], která je kompatibilní s GNU GPL verze 3[14] [16]. Licence AGPL na rozdíl od GPL upravuje podmínky při vzdálenému přistupování k aplikaci – tedy například přes webové rozhraní nebo API. Poskytování vzdáleného přístupu je u AGPL vyhodnoceno jako šíření aplikace. Plné znění licence je součástí zdrojových kódů Dapi, které naleznete na přiloženém médiu.

Jakýkoliv obsah, který není kódem, například texty apod., je pak vydán pod licencí Creative Commons Attribution-ShareAlike 4.0 International [11], není-li uvedeno jinak.

[13] Tedy nahradit nainstalovanou verzi jinou.
[14] AGPL verze 2 není kompatibilní s GPL verze 2.

```
$ dapi search flask
python - Python assistants originally shipped with devassistant itself
$ file .devassistant
.devassistant: ERROR: cannot open `.devassistant' (No such file or dir
ectory)
$ dapi install python
$ tree .devassistant
.devassistant
├── assistants
│   └── crt
│       ├── python
│       │   ├── django.yaml
│       │   ├── flask.yaml
│       │   ├── gtk3.yaml
│       │   └── lib.yaml
│       └── python.yaml
├── doc
│   └── python
│       ├── LICENSE
│       └── NOTICE
├── files
(...zkráceno...)
└── icons
    └── crt
        └── python.svg

14 directories, 22 files
$ dapi uninstall python
$ tree .devassistant
.devassistant
├── assistants
│   └── crt
├── doc
├── files
│   └── crt
└── icons
    └── crt

7 directories, 0 files
```

Ukázka kódu 3.7: Příklad práce s dapi

Závěr

Cílem práce bylo implementovat formát pro šíření asistentů do aplikace DevAssistant a jejich webový repozitář. Tento cíl byl úspěšně naplněn.

Podrobně jsem definoval formát dap, který je vhodný pro distribuování asistentů. Naimplementoval jsem nástroj, který s daným formátem pracuje.

Naimplementoval jsem repozitář Dapi, který umožňuje sdílet asistenty ve formátu dap. Jednotlivé dapy lze vyhledávat, procházet a stahovat. Dapi obsahuje API pro možnou integraci do aplikace DevAssistant. Instance Dapi je nasazena v cloudu na adrese `http://dapi.devassistant.org` a připravena k používání.

Možnosti rozvoje

V rámci dalšího rozvoje aplikace vidím následující možnosti:

- Zapisovací API pro nahrání dapu – tedy především možnost nahrání dapu z příkazové řádky. Toto vyžaduje také autentizaci v případě použití API, například pomocí tokenu.
- Závislosti dapů mezi sebou. Časem jistě vyvstane potřeba, aby jeden dap vyžadoval ke své funkcionalitě dap jiný – například pokud chce využívat k běhu svých asistentů snipety v něm obsažené.

- Deklarace podporovaných platforem. Jednotlivé asistenty mohou fungovat jen na některých platformách – například kvůli specifikování závislostí pouze pomocí seznamu RPM balíčků nebude asistent plně fungovat na distribuci s jiným balíčkovacím systémem, či asistent používající linuxový příkaz `sed` nebude vhodně fungovat na systému Mac OS.
- Kontrola YAML DSL z daploaderu. Daploader nyní neprovádí žádné kontroly obsahu jednotlivých asistentů – DevAssistant ale obsahuje kód, který kontroluje validnost asistentů. Tento kód by bylo vhodné vyextrahovat do samostatné knihovny, kterou by používal jak DevAssistant, tak daploader.
- Použití placené nebo vlastní instance OpenShiftu – aktuálně použitá veřejná instance OpenShiftu se zdá být přetížena množstvím uživatelů, což může mít za následek nepříjemné časy načítání jednotlivých stránek nebo odpovědí API.
- Asistent na tvorbu asistentů a dapů.
- Integrace do aplikace DevAssistant – do grafického i příkazového rozhraní.
- Pokročilé vyhledávání – filtrování podle uživatelů, licence, hodnocení apod.

Zdroje

1 AGUIRRE, Matías. *Python Social Auth* [online]. 2012 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://psa.matiasaguirre.net/⟩.

2 ALLEN, Greg. *TOC: jQuery Table of Contents Plugin* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://projects.jga.me/toc/⟩.

3 ALTIS, Kevin; BEHNEL, Stefan; MONTANARO, Skip; ROVNER, Mike et al. Integrating Python With Other Languages. *Python Wiki* [online]. 2014, [cit. 2014-05-08]. Dostupný z WWW: ⟨https://wiki.python.org/moin/IntegratingPythonWithOtherLanguages?action=recall&rev=75⟩.

4 ATHANASIAS, Dale; WESSELS, Rutger; COULTON, Simon; CLENDI, Joe et al. Web Frameworks for Python. *Python Wiki* [online]. 2014, [cit. 2014-05-09]. Dostupný z WWW: ⟨https://wiki.python.org/moin/WebFrameworks?action=recall&rev=426⟩.

5 AUTOMATTIC, Inc. *Gravatar: Celosvětově uznávané Avatary* [online]. 2014 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://cs.gravatar.com/⟩.

6 BONETTI, Marco. *Django Simple Captcha* [online]. 2013 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://django-simple-captcha.readthedocs.org/⟩.

7 CALLAWAY, Tom. Combined Dual and Multiple Licensing Scenario. *Fedora Project* [online]. 2013, [cit. 2014-05-07]. Dostupný z WWW: ⟨http://fedoraproject.org/wiki/Packaging:LicensingGuidelines#Combined_Dual_and_Multiple_Licensing_Scenario⟩.

8 CALLAWAY, Tom. Bad Licenses. *Fedora Project* [online]. 2014, [cit. 2014-05-07]. Dostupný z WWW: ⟨https://fedoraproject.org/wiki/Licensing:Main?rd=Licensing#Bad_Licenses⟩.

9 CALLAWAY, Tom. Good Licenses. *Fedora Project* [online]. 2014, [cit. 2014-05-07]. Dostupný z WWW: ⟨https://fedoraproject.org/wiki/Licensing:Main?rd=Licensing#Good_Licenses⟩.

10 CALLAWAY, Tom. Good Licenses. *Fedora Project* [online]. 2014, [cit. 2014-05-07]. Dostupný z WWW: ⟨https://fedoraproject.org/wiki/Licensing:Main?rd=Licensing#Good_Licenses_2⟩.

11 CREATIVE COMMONS CORPORATION. *Creative Commons: Attribution-ShareAlike 4.0 International* [online]. 2014 [cit. 2014-05-12]. Dostupný z WWW: ⟨https://creativecommons.org/licenses/by-sa/4.0/⟩.

12 DJANGO SOFTWARE FOUNDATION. *Django: The Web framework for perfectionists with deadlines* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://www.djangoproject.com/⟩.

13 DJANGO SOFTWARE FOUNDATION. *FAQ: Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don't use the standard names?* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names⟩.

14 FREE SOFTWARE FOUNDATION, Inc. *GNU General Public License* [online]. 1991 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://www.gnu.org/licenses/old-licenses/gpl-2.0.html⟩.

15 FREE SOFTWARE FOUNDATION, Inc. *GNU Affero General Public License* [online]. 2007 [cit. 2014-05-11]. Dostupný z WWW: ⟨http://www.gnu.org/licenses/agpl-3.0.html⟩.

16 FREE SOFTWARE FOUNDATION, Inc. *GNU General Public License* [online]. 2007 [cit. 2014-05-11]. Dostupný z WWW: ⟨http://www.gnu.org/licenses/gpl-3.0.html⟩.

17 FREE SOFTWARE FOUNDATION, Inc. *Gzip* [online]. 2010 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://www.gnu.org/software/gzip/gzip.html⟩.

18 FREE SOFTWARE FOUNDATION, Inc. *Make* [online]. 2013 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://www.gnu.org/software/make/⟩.

19 FREE SOFTWARE FOUNDATION, Inc. *Tar* [online]. 2013 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://www.gnu.org/software/tar/tar.html⟩.

20 GAYNOR, Alex et al. *django-taggit's documentation* [online]. 2014 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://django-taggit.readthedocs.org/en/latest/⟩.

21 GITHUB, Inc. *GitHub*. 2014. Dostupný také z WWW: ⟨https://github.com/⟩.

22 GITHUB, Inc. *Markdown Basics* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://help.github.com/articles/markdown-basics⟩.

23 GODWIN, Andrew. *South documentation* [online]. 2010 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://south.readthedocs.org/⟩.

24 HAAS, ROBERT. *Why The Clock is Ticking for MongoDB* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://rhaas.blogspot.ch/2014/04/why-clock-is-ticking-for-mongodb.html⟩.

25 HARVEST. *Chosen: A jQuery Plugin by Harvest to Tame Unwieldy Select Boxes* [online]. 2013 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://harvesthq.github.io/chosen/⟩.

26 HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. *The Definitive Guide to Django: Web Development Done Right*. Berkeley, CA: Apress, 2008. ISBN 978-1-59059-725-5.

27 CHACON, Scott. *Pro Git*. Praha: CZ.NIC, z. s. p. o., 2009. Dostupný také z WWW: ⟨http://knihy.nic.cz/files/nic/edice/scott_chacon_pro_git.pdf⟩. ISBN 978-80-904248-1-4.

28 CHAPUT, Matt; WALDMANN, Thomas. *About Whoosh* [online]. 2013 [cit. 2014-05-09]. Dostupný z WWW: ⟨https://bitbucket.org/mchaput/whoosh/wiki/Home⟩.

29 CHRISTIE, Tom. *Django REST framework: APIs made easy* [online]. 2014 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://www.django-rest-framework.org/⟩.

30 JQUERY FOUNDATION, The. *jQuery* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://jquery.com/⟩.

31 KABRDA, Bohuslav. [rfc] DevAssistant Packaging Format + Assistants Repository. *DevAssistant mailing list* [online]. 2013, [cit. 2014-05-06]. Dostupný z WWW: ⟨https://lists.fedoraproject.org/pipermail/devassistant/2013-November/000006.html⟩.

32 KABRDA, Bohuslav; HRÁČEK, Petr. Tutorial: Creating Your Own Assistant. *DevAssistant 0.9.0a2 documentation* [online]. 2013, [cit. 2014-05-06]. Dostupný z WWW: ⟨http://doc.devassistant.org/en/v0.9.0a2/developer_documentation/tutorial_creating_assistant.html⟩.

33 KABRDA, Bohuslav; HRÁČEK, Petr. User Documentation. *DevAssistant 0.9.0a2 documentation* [online]. 2013, [cit. 2014-05-06]. Dostupný z WWW: ⟨http://doc.devassistant.org/en/v0.9.0a2/user_documentation.html⟩.

34 KABRDA, Bohuslav; HRÁČEK, Petr. Yaml Assistant Reference. *DevAssistant 0.9.0a2 documentation* [online]. 2013, [cit. 2014-05-06]. Dostupný z WWW: ⟨http://doc.devassistant.org/en/v0.9.0a2/developer_documentation/yaml_assistant_reference.html⟩.

35 KABRDA, Bohuslav; HRÁČEK, Petr; KONČICKÝ, Jaromír; HRONČOK, Miroslav; RADĚJ, Tomáš. *devassistant-assistants-fedora* [online]. 2014 [cit. 2014-05-04]. Dostupný z WWW: ⟨https://github.com/bkabrda/devassistant-assistants-fedora⟩.

36 KREKEL, Holger. *pytest: helps you write better programs* [online]. 2013 [cit. 2014-05-08]. Dostupný z WWW: ⟨http://pytest.org/⟩.

37 LINDSLEY, Daniel. *Haystack: Search for Django* [online]. 2011 [cit. 2014-05-09]. Dostupný z WWW: ⟨http://haystacksearch.org/⟩.

38 MARIADB FOUNDATION. *MariaDB: An enhanced, drop-in replacement for MySQL* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨https://mariadb.org/⟩.

39 MICK, Trent. *markdown2* [online]. 2014 [cit. 2014-05-09]. Dostupný z WWW: ⟨https://pypi.python.org/pypi/markdown2⟩.

40 NPM, Inc. et al. *npm* [online]. [cit. 2014-05-07]. Dostupný z WWW: ⟨https://www.npmjs.org/⟩.

41 ORACLE CORPORATION. *MySQL: The world's most popular open source database* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://www.mysql.com/⟩.

42 OTTO, Mark et al. *Bootstrap* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://getbootstrap.com/⟩.

43 PERL.ORG; HIETANIEMI, Jarkko. *The Comprehensive Perl Archive Network* [online]. 2013 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://www.cpan.org/⟩.

44 PILGRIM, Mark. *Ponořme se do Python(u) 3*. Praha: CZ.NIC, z. s. p. o., 2010. Dostupný také z WWW: ⟨http://knihy.nic.cz/files/nic/edice/mark_pilgrim_dip3_ver3.pdf⟩. ISBN 978-80-904248-2-1.

45 POSTGRESQL GLOBAL DEVELOPMENT GROUP, The. *PostgreSQL: The world's most advanced open source database* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://www.postgresql.org/⟩.

46 PŘISPĚVATELÉ WIKIPEDIE. Cron. *Wikipedie: Otevřená encyklopedie* [online]. 2013, [cit. 2014-05-10]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=Cron&oldid=9966819⟩.

47 PŘISPĚVATELÉ WIKIPEDIE. Programování řízené testy. *Wikipedie: Otevřená encyklopedie* [online]. 2013, [cit. 2014-05-08]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=Programov%C3%A1n%C3%AD_%C5%99%C3%ADzen%C3%A9_testy&oldid=10974780⟩.

48 PŘISPĚVATELÉ WIKIPEDIE. Třetí normální forma. *Wikipedie: Otevřená encyklopedie* [online]. 2013, [cit. 2014-05-12]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=T%C5%99et%C3%AD_norm%C3%A1ln%C3%AD_forma&oldid=10429607⟩.

49 PŘISPĚVATELÉ WIKIPEDIE. Doménově specifický jazyk. *Wikipedie: Otevřená encyklopedie* [online]. 2014, [cit. 2014-05-06]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=Dom%C3%A9nov%C4%9B_specifick%C3%BD_jazyk&oldid=11183746⟩.

50 PŘISPĚVATELÉ WIKIPEDIE. Model-view-controller. *Wikipedie: Otevřená encyklopedie* [online]. 2014, [cit. 2014-05-09]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=Model-view-controller&oldid=11132009⟩.

51 PŘISPĚVATELÉ WIKIPEDIE. YAML. *Wikipedie: Otevřená encyklopedie* [online]. 2014, [cit. 2014-05-06]. Dostupný z WWW: ⟨http://cs.wikipedia.org/w/index.php?title=YAML&oldid=11256298⟩.

52 PYPA. *pip 1.5.5 documentation* [online]. 2014 [cit. 2014-05-08]. Dostupný z WWW: ⟨https://pip.pypa.io/en/1.5.X/⟩.

53 PYTHON SOFTWARE FOUNDATION. *List trove classifiers* [online]. 2014 [cit. 2014-05-06]. Dostupný z WWW: ⟨https://pypi.python.org/pypi?:action=list_classifiers⟩.

54 PYTHON SOFTWARE FOUNDATION. *pypa / pypi* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://bitbucket.org/pypa/pypi/overview⟩.

55 PYTHON SOFTWARE FOUNDATION. *PyPI* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://pypi.python.org/pypi⟩.

56 QUARANTO, Nick; MICHAELS-OBER, Erik; DOLLAR, David; MEIKLEJOHN, Christopher et al. *RubyGems.org* [online]. 2014 [cit. 2014-05-07]. Dostupný z WWW: ⟨https://rubygems.org/⟩.

57 QUARANTO, Nick; MICHAELS-OBER, Erik; DOLLAR, David; MEIKLEJOHN, Christopher et al. *rubygems/rubygems.org* [online]. 2014 [cit. 2014-05-06]. Dostupný z WWW: ⟨https://github.com/rubygems/rubygems.org⟩.

58 RED HAT, Inc. *DevAssistant: Homepage* [online]. 2013 [cit. 2014-05-07]. Dostupný z WWW: ⟨http://devassistant.org/⟩.

59 RED HAT, Inc. *Fedora Accounts System* [online]. 2013 [cit. 2014-05-10]. Dostupný z WWW: ⟨https://admin.fedoraproject.org/accounts⟩.

60 RED HAT, Inc. *OpenShift by Red Hat* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨https://www.openshift.com/⟩.

61 RED HAT, Inc. *Red Hat: The World's Open Source Leader* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨http://www.redhat.com/⟩.

62 RUNSCOPE, Inc. *Runscope: Debug and test your API, webhook and mobile back end service integrations* [online]. 2014 [cit. 2014-05-11]. Dostupný z WWW: ⟨https://www.runscope.com/⟩.

63 SIMONOV, Kirill. *PyYAML* [online]. 2014 [cit. 2014-05-08]. Dostupný z WWW: ⟨https://pypi.python.org/pypi/PyYAML⟩.

64 SQLITE CONTRIBUTORS. *SQLite* [online]. [cit. 2014-05-10]. Dostupný z WWW: ⟨http://www.sqlite.org/⟩.

65 TEZER, O. S. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [online]. 2014 [cit. 2014-05-10]. Dostupný z WWW: ⟨https://www.digitalocean.com/community/articles/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems⟩.

66 WADDINGTON, Tristan. *django-gravatar2* [online]. 2013 [cit. 2014-05-09]. Dostupný z WWW: ⟨https://pypi.python.org/pypi/django-gravatar2⟩.

Dodatek A

Seznam použitých zkratek

AGPL	Affero General Public License
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BSD	Berkeley Software Distribution
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CPAN	Comprehensive Perl Archive Network
CTU	Czech Technical University
ČVUT	České vysoké učení technické
DSL	Domain-specific language
ERM	Entity-relationship model
FIT	Fakulta informačních technologií
FTP	File Transfer Protocol
GNU	GNU's Not Unix!
GPL	General Public License
GTK	GNOME Toolkit neé GIMP Toolkit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISC	Internet Systems Consortium
JSON	JavaScript Object Notation
LGPL	Lesser General Public License

MIT	Massachusetts Institute of Technology
MVC	Model-view-controller
NoSQL	Not Only SQL
npm	Node Packaged Modules
PDF	Portable Document Format
PyPI	Python Package Index
REST	Representational state transfer
RPM	RPM Package Manager
SQL	Structured Query Language
TDD	Test-driven development
URL	Uniform Resource Locator
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

Dodatek B

Obsah přiloženého média

- `README.md` ..... stručný popis obsahu média
- `src`
  - `dapi` ..... zdrojové kódy aplikace Dapi
  - `daploader` ..... zdrojové kódy knihovny daploader včetně testů
  - `bakalarka` ..... zdrojová forma práce ve formátu Markdown a LaTeX
- `BP_Hroncok_Miroslav_2014.pdf` ..... text práce ve formátu PDF

Adresářová struktura B.1: Obsah přiloženého média