ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Nástroj pro IT správu počítačových učeben |
| **Student:** | Bc. Karel Papež |
| **Vedoucí:** | Ing. Tomáš Kadlec |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce zimního semestru 2017/18 |

## Pokyny pro vypracování

Závěrečná práce se zaměřuje na podporu přípravy instalací software v učebnách a řízení provozu počítačových učeben na FIT. Výstupem bude nástroj pro správu a provoz učeben.

Vytvořte zprávu o stavu provozu učeben. Zjistěte požadavky na instalaci a provoz učeben od vyučujících, dalších pracovníků a studentů. Proveďte analýzu a navrhněte webovou aplikaci, která umožní sběr požadavků na software a jeho automatickou instalaci. Implementujte část pro sběr požadavků na instalaci software. Hlavním výstupem je backend a frontend zajišťující workflow požadavku (vytvořený, schválený, vrácený, nainstalovaný, ověřený). Součástí backendu bude RESTful API a případně bude využito pro uživatelské rozhraní. Backend aplikace bude pokryt automatickými jednotkovými, funkčními a integračními testy. Dodržte postupy a technologie používané v Oddělení ICT FIT ČVUT.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 9. září 2016

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

Diplomová práce

# Nástroj pro IT správu počítačových učeben

***Bc. Karel Papež***

Vedoucí práce: Ing. Tomáš Kadlec

4. května 2017

# Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Tomášovi Kadlecovi, za cenné rady, věcné připomínky a vstřícnost při konzultacích. Poděkování také patří mé rodině, přítelkyni a přátelům za podporu po celou dobu studia.

# 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 4. května 2017 .....

### Odkaz na tuto práci

Papež, Karel. *Nástroj pro IT správu počítačových učeben.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

# Abstrakt

Práce se zabývá problematikou přípravy instalací softwaru v počítačových učebnách na Fakultě informačních technologií ČVUT v Praze. Nejprve byla provedena rešerše stávajícího stavu provozu učeben a průzkum potřeb jejich uživatelů. S použitím těchto údajů byla následně navržena nová webová aplikace, jejíž hlavním účelem je sběr požadavků na software. Pro implementaci byl využit PHP framework Symfony.

**Klíčová slova** webová aplikace, software, počítačová učebna, uživatelský průzkum, prototypování, PHP, Symfony

# Abstract

This thesis focuses on the problem of software installation management in the computer labs at Faculty of Information Technology CTU in Prague. The initial part of the thesis consists of a research of the current installation procedures and a survey analyzing the users' needs. Using this information, a new web application for gathering software requirements has been designed. The application has been implemented using the Symfony PHP framework.

**Keywords** web application, software, computer lab, user survey, prototyping, PHP, Symfony

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Nedílnou součástí vzdělávacího procesu na Fakultě informačních technologií ČVUT v Praze je výuka v počítačových učebnách. Probíhají zde pravidelná seminární cvičení i jednorázové akce – konzultace, zkoušky, školení a další vzdělávací aktivity. Příprava počítačových učeben, která se provádí speciálně pro každý studijní semestr, zahrnuje pravidelnou aktualizaci programů a jejich obměnu dle aktuálních potřeb uživatelů.

Hlavní motivací této práce je podpořit správu fakultních počítačových učeben ve smyslu zefektivnění sběru uživatelských požadavků na software a automatizace jejich dalšího zpracování. Tomu by měla napomoci podpůrná webová aplikace, jejíž návrhem a implementací se tato práce zabývá. Samotnému návrhu předchází podrobnější analýza současného stavu a identifikace možných problémů a nedostatků.

Práce je členěna do tří hlavních kapitol. Úvodní kapitola se zaměřuje na analýzu současného stavu provozu fakultních počítačových učeben. Součástí této fáze je provedení průzkumu mezi uživateli, a to formou kvalitativní (prostřednictvím polostrukturovaných rozhovorů) i kvantitativní (dotazníkem), který je základem pro posouzení stávajícího stavu. Závěrem jsou stanoveny požadavky na funkce a kvality nové webové aplikace podporující správu počítačových učeben.

Druhá kapitola je zaměřena na návrh aplikace, založený na poznatcích z analýzy. V návrhu je definován životní cyklus požadavku na software a možné způsoby jeho zadání, a dále specifikovány případy užití aplikace, následované grafickými prototypy a popisem obrazovek. Součástí návrhu je rovněž doménový model aplikace.

Třetí kapitola je pak věnována seznámení s implementací webové aplikace. Náplní kapitoly je popis architektury aplikace a použitých technologií, uvedení konceptů řešení dílčích požadavků (funkčních i nefunkčních), popis RESTful aplikačního rozhraní a obeznámení s procesem testování aplikace.

Kapitola 1

# Analýza

Cílem analytické části práce bylo zmapovat současný stav počítačových učeben na FIT ČVUT – popsat možnosti jednotlivých typů učeben a způsob jejich řízení. Zároveň byl proveden uživatelský průzkum s cílem zjistit potřeby uživatelů počítačových učeben.

Průzkum ukázal, že současný systém je v některých ohledech neefektivní (zejména část týkající se sběru a zpracování požadavků na software) a nenaplňuje řadu uživatelských požadavků. Nedostatky identifikované v rámci průzkumu by měla odstranit nová aplikace, na níž jsou v závěru této kapitoly stanoveny funkční a kvalitativní požadavky.

## 1.1 Stávající stav

V kontextu nástroje pro IT správu počítačových učeben nejsou vyhledávána ani posuzována jiná řešení, neboť se jedná o velmi specifickou doménu. Na fakultní počítačové učebny se vztahují speciální požadavky a omezení ze strany oddělení ICT, které zajišťuje jejich provoz. Tato sekce se proto omezuje pouze na popis stávajícího stavu počítačových učeben.

### 1.1.1 Typy učeben a jejich možnosti

Většina fakultních počítačových učeben je pod správou oddělení ICT. Patří mezi ně seminární učebny, běžné počítačové učebny a některé specializované laboratoře. Další dvě učebny a ostatní laboratoře jsou pod správou jednotlivých kateder fakulty.

Všechny učebny jsou vybaveny počítačem pro vyučujícího a projektorem s možností připojit vlastní notebook. Běžné počítačové učebny a specializované laboratoře disponují počítači rovněž pro studenty.

#### 1.1.1.1 Běžné počítačové učebny

Běžné počítačové učebny používají předinstalované operační systémy. Jejich nastavení se v průběhu semestru pokud možno nemění. Studenti ani učitelé na ně nemají superuživatelský přístup, proto nelze provádět některé typy experimentů. Učebny jsou vybaveny operačními systémy **Gentoo GNU/Linux** (lokální instalace, síťový boot), **MS Windows** (lokální instalace) a dále systémy **Progtest** a **Moodle**, které slouží k ověřování studijních výsledků.

Běžné učebny se využívají také pro jednorázové akce (např. zkoušky, konference), pro které se často běžné instalace upravují a výjimečně i vytváří nové.

#### 1.1.1.2 Specializované laboratoře

Mezi specializované laboratoře patří tzv. bourací učebna, síťová laboratoř a Apple učebna, které jsou pod správou ICT oddělení, a rovněž hardwarové laboratoře, jež spravuje katedra číslicového návrhu (dále jen KČN).

**Bourací učebna a síťová laboratoř** (místnosti T9:345 a T9:344) slouží k výuce administrace systémů a sítí, kurzů CCNA (Cisco Certified Network Associate) aj. Obě laboratoře umožňují učitelům vytvářet a v průběhu semestru používat různé operační systémy, a provádět jejich úpravy dle potřeb výuky. Operační systém lze měnit jak mezi cvičeními, tak v průběhu jednotlivých cvičení.

**Apple učebna** (místnost TH:A-1142) se využívá především k výuce předmětů zabývajících se vývojem iOS aplikací, ale také bezpečnosti, kryptologie a dalších. Kromě operačního systému **macOS** umožňuje využít také běžné instalace Gentoo GNU/Linux a MS Windows.

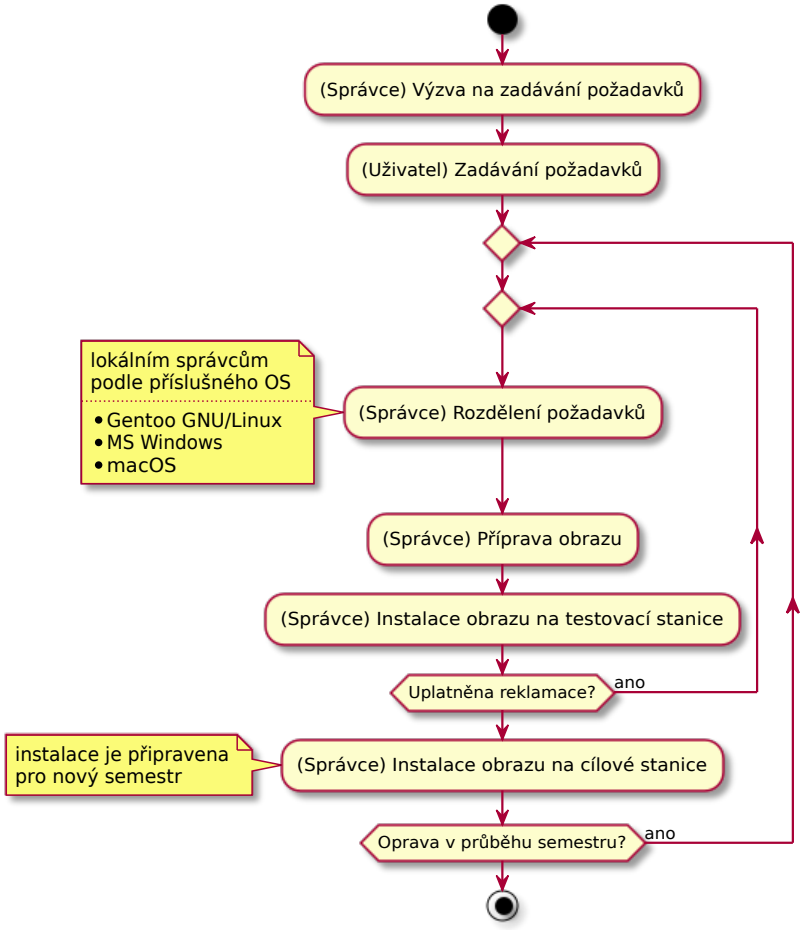
**Hardwarové laboratoře** (místnosti A-1042 a A-1048) jsou zaměřeny na výuku hardwarových předmětů. Uživatelé nemají superuživatelské oprávnění, ale mohou připojovat hardware či používat 3D tiskárny. Instalován je operační systém MS Windows připravený KČN. K dispozici je rovněž síťový boot systému Gentoo GNU/Linux.

### 1.1.2 Postup instalace učeben

Postup instalace zahrnuje sběr požadavků na software, vytvoření instalačních obrazů a jejich distribuci na počítačové stanice v učebnách. Tento proces se opakuje typicky dvakrát ročně, vždy před zahájením výukového semestru. Instalace běžných počítačových učeben a specializovaných laboratoří probíhá odlišným způsobem.

#### 1.1.2.1 Běžné počítačové učebny

Příprava běžných počítačových učeben sestává z několika kroků (viz obr. 1.1). Prvním z nich je vypsání výzvy na zadávání požadavků T. Kadlecem, vedoucím oddělení ICT a zároveň správcem běžných učeben. Výzva je zadána formou e-mailu do fakultního mailing listu pro vyučující, vždy jeden až dva měsíce před začátkem semestru.




Obrázek 1.1: Postup instalace běžných počítačových učeben (aktuální stav)

**Sběr požadavků** Vyučující, kteří mají speciální požadavky, je zasílají na helpdesk FIT (*helpdesk@fit.cvut.cz*). Typicky se jedná o volbu operačního systému a výčet aplikačního softwaru. Zkušenější uživatelé mohou přiložit např. předem vytvořený instalační skript. Požadavky si prostřednictvím tiketovacího systému RT[1] následně rozdělí správci učeben v závislosti na požadovaném OS.

[1] `https://bestpractical.com/request-tracker`

Rozdělení mezi správci je následující:

- Gentoo GNU/Linux: T. Kadlec a M. Václavík,
- MS Windows: L. Kudrna a J. Kadleček,
- macOS: M. Moravec.

Požadavek týkající se více operačních systémů je nutné duplikovat, neboť systém RT neumožňuje více souběžných řešitelů.

**Příprava obrazů** Postup přípravy instalačních obrazů se pro každý operační systém liší. Obraz systému Gentoo GNU/Linux se vytváří pomocí automatizačního nástroje Ansible[2]. Veškeré požadavky jsou zahrnuty do existujícího Ansible *playbooku*. Změny jsou prováděny na základě porovnání požadavků v RT a obsahu *playbooku*. Jsou instalovány aktuální stabilní verze softwarů.

Při instalaci OS Windows se obvykle vychází z obrazu pro předchozí semestr, podle potřeby je původní obraz nahrazen čistou instalací. Software je instalován ručně, nejsou využity automatizační nástroje ani balíčkovací systémy.

**Testování a reklamace** Nejpozději dva týdny před začátkem semestru je nainstalována jedna z učeben, kde si vyučující mohou instalaci vyzkoušet a ověřit, že splňuje jejich požadavky. V tomto období je vhodné uplatnit případné reklamace. Obdrží-li správce požadavek na reklamaci, instalaci opraví a přeinstaluje. V opačném případě je instalace pro nový semestr připravena.

Opravy lze výjimečně provádět i v průběhu semestru. Podle závažnosti problému se provede buď centralizovaná oprava zahrnující opětovnou distribuci obrazu na všechny počítače v učebnách, nebo pouze lokální oprava pomocí opravných skriptů.

#### 1.1.2.2 Specializované laboratoře

Laboratoře pro výuku administrace umožňují nativní provoz systémů z pevného disku. Vytváření a instalace obrazů operačních systémů se provádí prostřednictvím speciální aplikace dostupné z boot menu po startu počítače. Autorem obrazu může být správce učebny (J. Žďárek, nově M. Václavík) nebo vyučující, který se u správce předem zaregistroval.

Apple učebnu spravuje lokální správce M. Moravec. Vzhledem k omezenému počtu uživatelů řeší požadavky, testování i případné reklamace s uživateli osobně. Při instalaci vychází z čistého obrazu operačního systému macOS

[2] `https://www.ansible.com/`

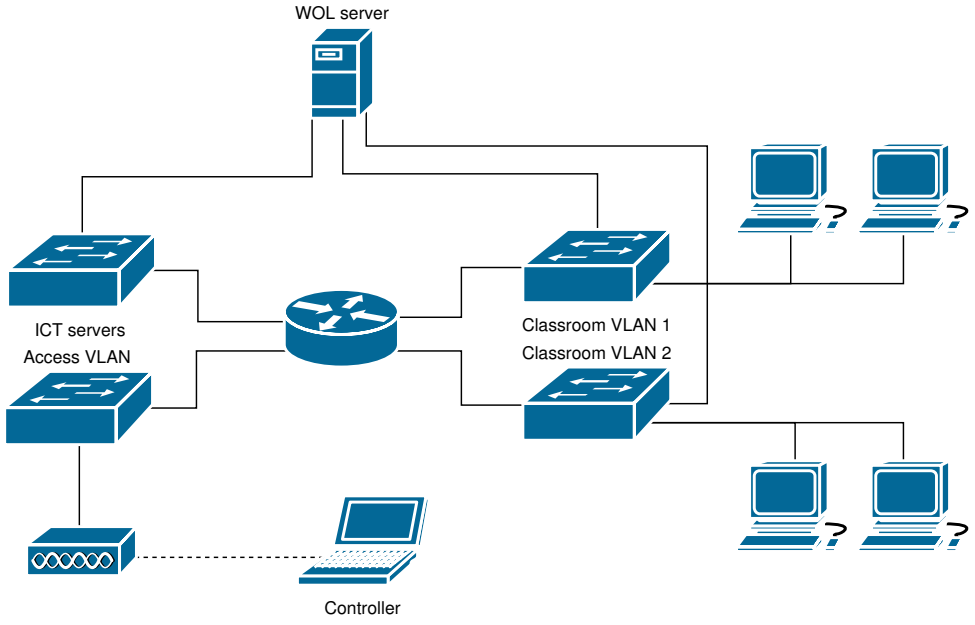
a repozitáře se softwarovými balíčky. Využívají se automatizační nástroje AutoPkg[3] a Munki[4].

Hardwarové laboratoře mají také svého lokálního správce, M. Vrátila, který přijímá požadavky a připravuje instalace pro potřeby uživatelů KČN. Vytváří se obraz operačního systému MS Windows, který již obsahuje všechny potřebné aplikace, ovladače apod. Tento obraz je následně distribuován na počítače v učebnách KČN.

Specializované laboratoře fungují nezávisle na běžných učebnách. Způsob jejich instalace umožňuje větší variabilitu a splňuje zvláštní nároky na jejich provoz. Z tohoto důvodu bude první fáze návrhu aplikace podporující přípravu instalací zaměřena pouze na standardní učebny a specializovaným laboratořím se nebude věnovat. Výjimku tvoří učebny KČN, které jsou způsobem instalace analogické běžným učebnám, a proto i pro ně bude nová aplikace relevantní.

### 1.1.3 Architektura systému

Stávající systém řízení počítačových učeben se skládá z několika prvků (viz obr. 1.2). Primárně se jedná o řídící počítač (*Controller*), Wake on LAN server (*WOL server*) a pracovní stanice v učebnách (*Workstations*). Řídící počítač je obsluhován správcem učebny a komunikuje s Wake on LAN serverem a pracovními stanicemi.

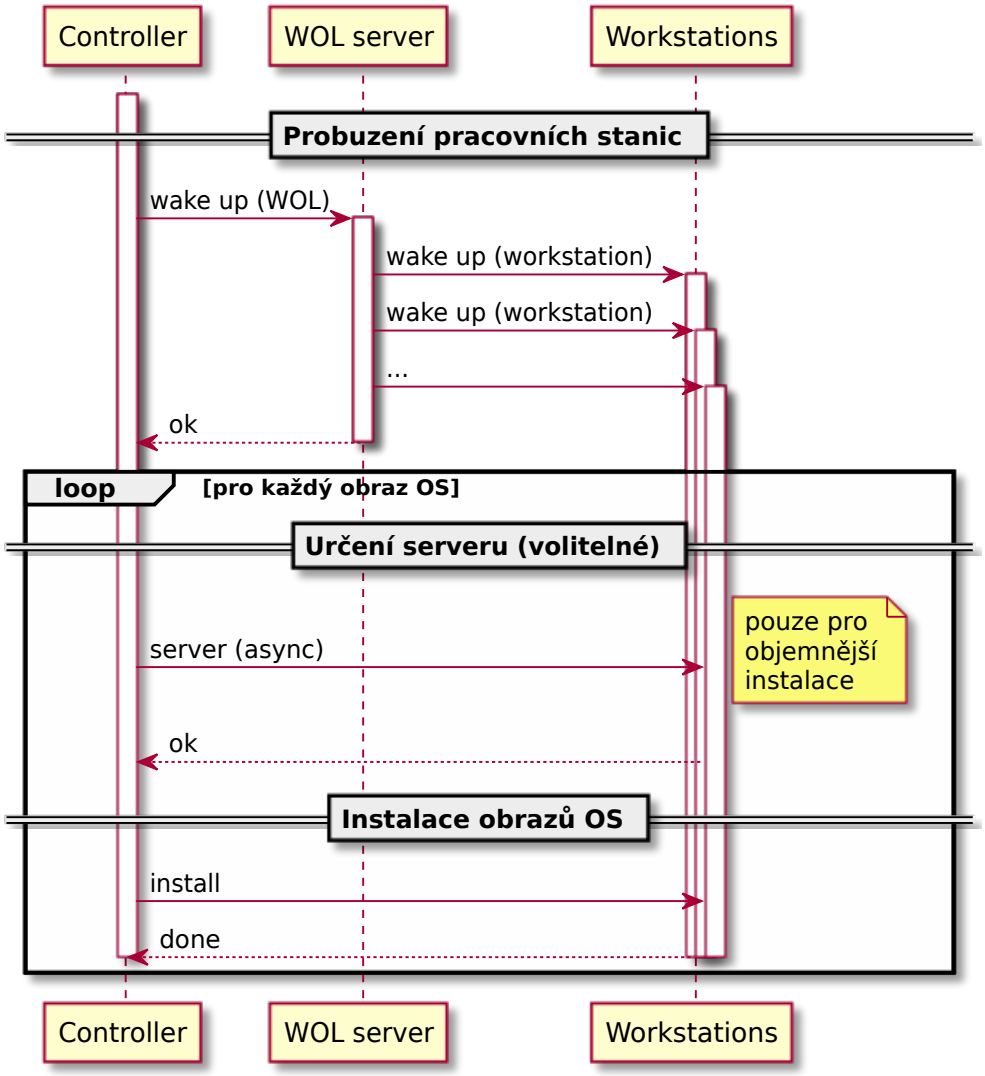



Obrázek 1.2: Síťová infrastruktura systému řízení počítačových učeben

[3] `https://github.com/autopkg/autopkg`
[4] `https://github.com/munki/munki`

Obrázek 1.3 představuje zjednodušený diagram komunikace jednotlivých prvků systému během distribuce instalací na cílové stanice. V první fázi řídící počítač iniciuje probuzení pracovních stanic prostřednictvím Wake on LAN serveru. Druhá, volitelná, fáze je určení serveru z množiny pracovních stanic. Při samotné instalaci (třetí fáze) zvolený server rozesílá prostřednictvím skupinového vysílání (multicast) data na zbylé stanice.

Fáze určení serveru se využívá zejména při objemných instalacích pro snížení datového toku (typicky všechny standardní obrazy OS). Druhá a třetí fáze se opakují pro každý operační systém.




Obrázek 1.3: Diagram komunikace prvků systému během instalace učeben

## 1.2 Uživatelský průzkum

Uživatelský průzkum zjišťoval, jaké jsou potřeby uživatelů při práci v počítačových učebnách na FIT ČVUT. Cílem průzkumu bylo zmapovat současný stav a zjistit potřeby vyučujících i správců učeben, od sběru požadavků na instalace přes samotnou přípravu instalací až po následné testování a případné reklamace.

Z průzkumu vyplývá, že uživatelé počítačových učeben postrádají základní informace, jako je seznam aktuálního softwarového vybavení počítačů či stav zpracování svých požadavků na software. Při specifikaci nových požadavků uživatelé často vychází z požadavků pro předchozí období, proto by využili funkci zobrazení historie požadavků s možností jejich automatického přenosu do nového období. Vyučující při specifikaci požadavků často spolupracují se svými kolegy.

### 1.2.1 Kvalitativní průzkum (rozhovory)

Na základě rozhovorů se zástupci uživatelských skupin (viz příloha A) vznikla sada hypotéz ke kvantitativnímu ověření (dotazníkem). Dotazovaní byli (přímo nebo kombinací) z řad akademických zaměstnanců (vyučujících) a neakademických zaměstnanců.

Aplikace pro podporu IT správy počítačových učeben by měla:

- umožnit uživateli specifikovat požadavky na software s možností:
  - volby verze (aktuální, starší),
  - přiložení návodů na složitější instalace,
  - přiložení instalačních skriptů;
- obsahovat historii požadavků s možností přenosu do nového období,
- nabídnout uživateli:
  - širší škálu operačních systémů, resp. linuxových distribucí,
  - možnost instalace vlastního obrazu OS;
- zobrazit správci aktuální seznam požadavků na systém, který spravuje,
- umožnit komentování jednotlivých požadavků,
- poskytovat informaci o stavu instalace,
- podpořit proces reklamace instalace,
- umožnit uživateli potvrzení správnosti instalace,

- notifikovat (např. formou e-mailu):
  - správce o nových požadavcích (včetně reklamací),
  - uživatele o proběhlé instalaci, resp. změně stavu instalace;
- poskytovat u standardních obrazů (Gentoo GNU/Linux, MS Windows) přehled:
  - aktuálně instalovaného softwaru,
  - změn oproti předchozí verzi instalace;
- poskytovat přehled nových verzí dostupného softwaru, včetně:
  - informace o změnách oproti původním verzím,
  - upozornění na bezpečnostní rizika a potenciální kolize s dalším SW;
- umožnit vazbu požadavků na vyučované předměty,
- podpořit spolupráci uživatelů (např. sdílení požadavků mezi vyučujícími jednoho předmětu),
- řešit deduplikaci požadavků,
- umožnit plánování zvolené instalace na konkrétní čas a místo (tj. skupinu počítačů),
- umožnit vzdálené testování instalace,
- poskytovat aplikační rozhraní (API) umožňující automatické řízení instalací.

Z kvalitativního průzkumu vyplynuly také následující poznatky:

- Uživatelé by ocenili možnost manipulovat s položkami bootovacího menu, jmenovitě:
  - navolit primární položku (na konkrétní čas a místo),
  - přeskupit položky,
  - skrýt nerelevantní položky.
- Uživatelé by uvítali možnost vzdálené interakce se stanicemi, zahrnující:
  - základní operace se zvolenou skupinou stanic (zapnutí, vypnutí, reset),
  - připojení na konkrétní stanici (za účelem zjištění stavu systému a poslední aktivity uživatele);

- Uživatelům chybí informace o stavu počítačů v učebnách, tj.:
  - základní statistiky, kolik nabootovalo apod.,
  - monitoring instalovaných prostředí.
- Vedle standardních obrazů OS by uživatelé využili také minimalistický obraz disponující pouze základním softwarovým vybavením (webový prohlížeč, textový editor, ...), bez nutnosti přihlášení, s rychlým startem.

Hypotézy vytvořené v rámci kvalitativního průzkumu jsou dále ověřeny kvantitativně, formou dotazníku.

### 1.2.2 Kvantitativní průzkum (dotazník)

Kvantitativní průzkum měl za cíl ověření hypotéz o sběru požadavků na učebnové instalace a plánování instalací, dále pak prioritizaci jednotlivých požadavků a funkcí. Průzkum proběhl formou elektronického dotazníku (viz příloha B) a zúčastnilo se jej 30 respondentů, především vyučujících FIT ČVUT.

V souladu s očekáváním se ukázalo, že značná část respondentů klade na instalace speciální požadavky. Při specifikaci těchto požadavků často vychází z historických požadavků (typicky z požadavků pro předchozí běh svého předmětu) a spolupracují se svými kolegy.

Největší zájem dotazovaní projevili o následující funkce:

- zobrazení seznamu aktuálně instalovaného softwaru (93 % respondentů),
- informace o stavu instalace (zpracování požadavků) (90 %),
- přehled změn v nových instalacích oproti původním (87 %),
- možnost výběru softwaru z nabídky (73 %),
- volba konkrétní verze softwaru (73 %),
- možnost vzdáleného otestování instalace (73 %),
- zobrazení historie požadavků s možností snadného znovupoužití (70 %).

Nejednoznačné názory jsou na následující funkce:

- přiložení textových návodů na složitější instalace (47 %),
- přiložení instalačních skriptů (47 %),
- nabídka více operačních systémů, linuxových distribucí (43 %),
- plánování na konkrétní čas a místo (43 %).

Nevalný zájem dotazovaní projevili o následující funkce:

- instalace OS „na míru“ (33 %),
- volba minimalistického OS (30 %),
- instalace vlastního obrazu OS (23 %).

Mezi zajímavé návrhy dalších funkcí patří:

- automatický přenos požadavků z předchozího do nového období,
- možnost využít návrhy změn kolegů, resp. možnost koordinace návrhů více učitelů,
- možnost doinstalací, případně oprav během semestru,
- zadávání požadavků prostřednictvím verzovacího systému Git,
- kontakt s živou bytostí během specifikace a dalšího případného řešení problémů s instalací.

Většina respondentů (87 %) během specifikace požadavků pravidelně nebo alespoň občas spolupracuje se svými kolegy.

Dotazník rovněž zjišťoval, zda a případně jakým způsobem uživatelé ověřují, že instalace splňuje jejich požadavky. Z průzkumu vyplynulo, že instalaci ověřuje pouze zhruba polovina dotazovaných. Současně většina dotazovaných by využila možnost vzdáleného ověření instalace. Tato funkce by tedy mohla řadu uživatelů přimět k včasnému ověření instalace, a tím snížit počet reklamací v průběhu semestru.

## 1.3 Posouzení stávajícího stavu

Posouzení stávajícího stavu (kap. 1.1) se vztahuje pouze na běžné počítačové učebny, neboť jak již bylo zmíněno, specializované laboratoře fungují nezávisle na běžných učebnách a nebudou pro tuto práci důležité; výjimku tvoří učebny KČN. Závěry vychází z uživatelského průzkumu (kap. 1.2).

**Přednosti** stávajícího řešení:

- Požadavky na instalace lze zadávat takřka libovolnou formou, včetně přiložení souborů (návody na složitější instalace, instalační skripty).
- Požadavky ve formě tiketů v RT je možné komentovat.
- Instalace operačních systémů Gentoo GNU/Linux a macOS využívají automatizační nástroje.

- Uživatelé mohou před začátkem semestru testovat instalace v učebnách.
- Je možné provádět (v rámci možností) úpravy instalací i průběhu semestru.

**Nedostatky** stávajícího řešení:

- Neexistuje veřejně dostupný přehled:
  - aktuálně instalovaného softwaru,
  - změn v nových instalacích oproti předchozím.
- Uživatelé nemají možnost snadno dohledat historii svých požadavků na instalace.
- Požadavky na více operačních systémů je nutné duplikovat.
- Chybí vazba požadavků na předměty.
- Uživatelé nemohou sdílet požadavky mezi sebou navzájem.
- Uživatelé nejsou informováni o stavu instalace (zpracování požadavků).
- Zpracování požadavků je komplikované, pro některé OS by šlo z požadavků rovnou generovat instalační skript.
- Uživatelé nemohou otestovat instalace přes vzdálený přístup.
- Nabídka operačních systémů v běžných učebnách je poměrně úzká.
- V běžných učebnách není uživatelům povoleno upravovat existující ani instalovat vlastní obrazy OS.
- Vyučující zaměstnaní na dohody o pracích konaných mimo pracovní poměr neobdrží výzvu na zadávání požadavků na instalace, neboť nejsou členy fakultní e-mailingové konference pro vyučující.

## 1.4 Požadavky na funkce

Funkční požadavky byly stanoveny na základě uživatelského průzkumu (kapitola 1.2). V rámci skupin funkcí jsou požadavky dělené na **základní** (výchozí úroveň nezbytnosti [1] *MUST*) a **rozšířené** (výchozí úroveň nezbytnosti *MAY* – není-li uvedeno jinak). Rozšířené požadavky jsou řazené sestupně podle uživatelských priorit.

### 1.4.1 Obecné

K **základním** funkcím patří:

- přehled aktuálně nainstalovaného softwaru,
- informování uživatele o stavu instalace (zpracování požadavků),
- zobrazení historie požadavků,
- možnost zobrazit správci aktuální seznam požadavků na systém který spravuje,
- [SHOULD] vazba požadavků na vyučované předměty,
- [SHOULD] sdílení požadavků mezi vyučujícími,
- [SHOULD] komentování existujících požadavků,
- [SHOULD] možnost vzdáleného testování instalace.

K **rozšířeným** funkcím patří:

- přehled změn v nových instalacích oproti předchozím,
- přehled nových verzí dostupného softwaru,
- plánování instalace na konkrétní čas a místo.

### 1.4.2 Volba softwaru

K **základním** funkcím patří:

- výběr softwaru z nabídky,
- volba konkrétní verze softwaru,
- [SHOULD] přenos požadavků z předchozího do nového období,
- [SHOULD] přiložení textových návodů (formulář/soubor)
- [SHOULD] přiložení instalačních skriptů.

**Rozšířenou** funkcí volby softwaru je integrace se službami přehledu balíčků jednotlivých distribucí:

- `https://packages.gentoo.org`,
- `https://packages.debian.org`.

### 1.4.3 Volba obrazu OS

K **základním** funkcím patří:

- výběr ze standardních učebnových instalací (Gentoo GNU/Linux, MS Windows),
- [SHOULD] vytvoření a instalace OS „na míru“ (např. testovací image).

K **rozšířeným** funkcím patří:

- nabídka více operačních systémů, linuxových distribucí,
- výběr minimalistického OS (základní SW, bez přihlášení, rychlý start),
- nahrání a instalace vlastního obrazu OS.

## 1.5 Požadavky na kvality

Požadavky na kvality se řídí dokumentem Pravidla a zásady projektů FIT (dále jen Pravidla) z 30. 9. 2016 (příloha C). Dále jsou uvedeny upřesnění a rozdíly v požadavcích. Výchozí úroveň nezbytnosti uvedených požadavků je *MUST*, dále se úroveň nezbytnosti řídí úrovněmi uvedenými v Pravidlech.

### 1.5.1 Dokumenty

Práce sleduje doporučený harmonogram podle Pravidel. Nachází se v samostatném dokumentu (příloha H). Práce dále zahrnuje:

- instalační příručku (příloha G),
- vývojářskou dokumentaci (dokumentaci vnitřního API).

Struktura a umístění dokumentace je v souladu s Pravidly.

### 1.5.2 Architektura a integrace do infrastruktury

Aplikace je implementována prostřednictvím jazyků:

- PHP 7 s využitím frameworku Symfony 3,
- JavaScript s využitím knihovny JQuery,
- HTML5,
- CSS3.

### 1.5.3 Webová přístupnost

Požadavky v této kategorii budou splněny v plném rozsahu.

### 1.5.4 Kód aplikace

Upřesnění požadavků na kód aplikace:

- Kód se řídí standardy jazyka PHP (PSR[5]) a je v souladu s konvencemi frameworku Symfony[6].
- [SHOULD] Pro aplikaci existuje kompletní anglická lokalizace.
- [SHOULD] V případě zveřejnění zdrojového kódu bude aplikace dostupná pod licencí MIT.

#### 1.5.4.1 Verzování

Kód aplikace je vyvíjen na revizním systému Git[7] využívající:

- veřejný repozitář na službě GitLab provozované oddělením ICT[8],
- branching model GitLab Flow[9],
- sémantické verzování[10].

#### 1.5.4.2 Kontrola kvality

Upřesnění požadavků na kontrolu kvality:

- Vývoj kódu se opírá o kontrolní nástroje, zejména kontrolu syntaxe (*php -l*) a kontrolu stylu (přes PHP CodeSniffer[11], příp. funkce použitého IDE). Linting je součástí testovacího procesu v continuous integration.
- Konkrétní metriky pro kód nejsou stanoveny; slouží primárně jako doporučení pro vývojáře.

### 1.5.5 Provoz, údržba a rozvoj aplikace a podpora uživatelů

Provoz je řešený ve dvou prostředích:

- *staging*, které je automaticky nasazované z větve master,
- *production*, které je nasazené z posledního otagovaného releasu.

[5] `http://www.php-fig.org/psr/`
[6] `http://symfony.com/doc/current/contributing/code/standards.html`
[7] `https://rozvoj.fit.cvut.cz/Main/Git`
[8] `https://ict.fit.cvut.cz/gitlab`
[9] `https://about.gitlab.com/2014/09/29/gitlab-flow/`
[10] `http://semver.org/`
[11] `https://github.com/squizlabs/PHP_CodeSniffer`

Logování chyb na back-endu i front-endu je integrováno s fakultní monitorovací službou Sentry[12].

Proces testování nových verzí nebude zahrnovat:

- provoz nezávislé (beta) verze kvůli zjednodušení procesu vývoje a nasazení,
- podporu A/B testování kvůli implementační náročnosti a obtížnému vyhodnocování výsledků v problémové doméně aplikace.

### 1.5.6 Bezpečnost a ochrana osobních údajů

Aplikace nepracuje s citlivými daty uživatele. Mezi zpracovávané osobní údaje patří pouze identita uživatele, tj.:

- jméno a příjmení,
- uživatelské jméno,
- e-mailová adresa,
- uživatelské role.

Konkrétní definice oprávnění, tj. jaká data jsou dostupná kterým uživatelům v závislosti na autorizaci uživatele, bude:

- v samostatném souboru (příloha implementační fáze),
- předmětem konfigurace.

### 1.5.7 Další koncepty webových aplikací (Web 2.0)

Úvaha nad dalšími koncepty webových aplikací je zpracovaná v příloze D.

[12] `http://sentry.fit.cvut.cz/`

Kapitola 2

# Návrh

Na základě poznatků z analýzy byla navržena webová aplikace pro podporu IT správy počítačových učeben, jejíž primárním účelem je sběr požadavků na instalaci softwaru.

Úvodem kapitoly jsou popsány možné způsoby zadání požadavku a definován jeho životní cyklus. Dále jsou uvedeny případy užití aplikace následované Lo-Fi prototypy a popisem obrazovek. V závěru kapitoly je prezentován doménový model aplikace.

Diagramy v této kapitole (vyjma Lo-Fi prototypů v sekci 2.3) byly vytvořeny pomocí open source nástroje PlantUML [2].

## 2.1 Softwarový požadavek a jeho životní cyklus

Primárním účelem navrhované aplikace je sběr uživatelských požadavků na software. Sekce popisuje možné způsoby zadání požadovaného softwaru a životní cyklus každého požadavku od jeho vytvoření až po instalaci a ověření uživatelem.

### 2.1.1 Forma zadání požadavku

Požadavek na software může být zadán několika způsoby – výběrem z nabídky, vyhledáním podle názvu balíčku nebo vlastním zadáním. Pro některé operační systémy však nemusí být všechny způsoby zadání požadavku dostupné.

#### 2.1.1.1 Výběr z nabídky

Prvním způsobem je výběr z nabídky předem připraveného softwaru. Nabídka softwaru se může pro každý podporovaný operační systém lišit.

Tento způsob zadání je pro uživatele jednoduchý a rychlý a z pohledu správce vyžaduje nejmenší úsilí během dalšího zpracování požadavku (např. je možné přeskočit krok schvalování požadavku, viz dále).

#### 2.1.1.2 Vyhledání podle názvu balíčku

Řada operačních systémů umožňuje instalaci softwaru prostřednictvím správce balíčků[13]. Z tohoto důvodu aplikace umožní zadat požadavek vyhledáním softwarového balíčku – skrze vyhledávací pole integrované se službou přehledu balíčků příslušného operačního systému, resp. linuxové distribuce.

Zadání požadavku touto formou využijí především uživatelé s hlubší znalostí cílového operačního systému. Instalace požadovaných balíčků je dobře automatizovatelná, nicméně oproti předchozímu způsobu je vyžadováno jejich manuální schválení příslušným správcem.

#### 2.1.1.3 Vlastní (částečně) strukturované zadání

Posledním způsobem je vlastní zadání strukturovanou formou. Požadavek je zadáván prostřednictvím formuláře s poli pro název požadovaného softwaru (povinné), jeho URL adresu, verzi a komentář.

Formulář rovněž umožňuje přiložit jeden či více souborů, včetně doplňujících metadat. Velikost souboru k nahrání je omezena. Je-li soubor větší než daná mez, je možné k jeho uložení využít externí službu a poskytnout URL adresu, klíč a typ použité šifry.

Výhodou posledního způsobu zadání je jeho obecnost – lze použít pro libovolný OS; pro některé operační systémy představuje jedinou možnou variantu. Prostor pro automatizaci instalace je v tomto případě však značně omezený.

### 2.1.2 Workflow požadavku

Životní cyklus požadavku podléhá časovému plánu přípravy učeben. Součástí plánu je datum zahájení semestru, od kterého se odvíjí počáteční termíny pro:

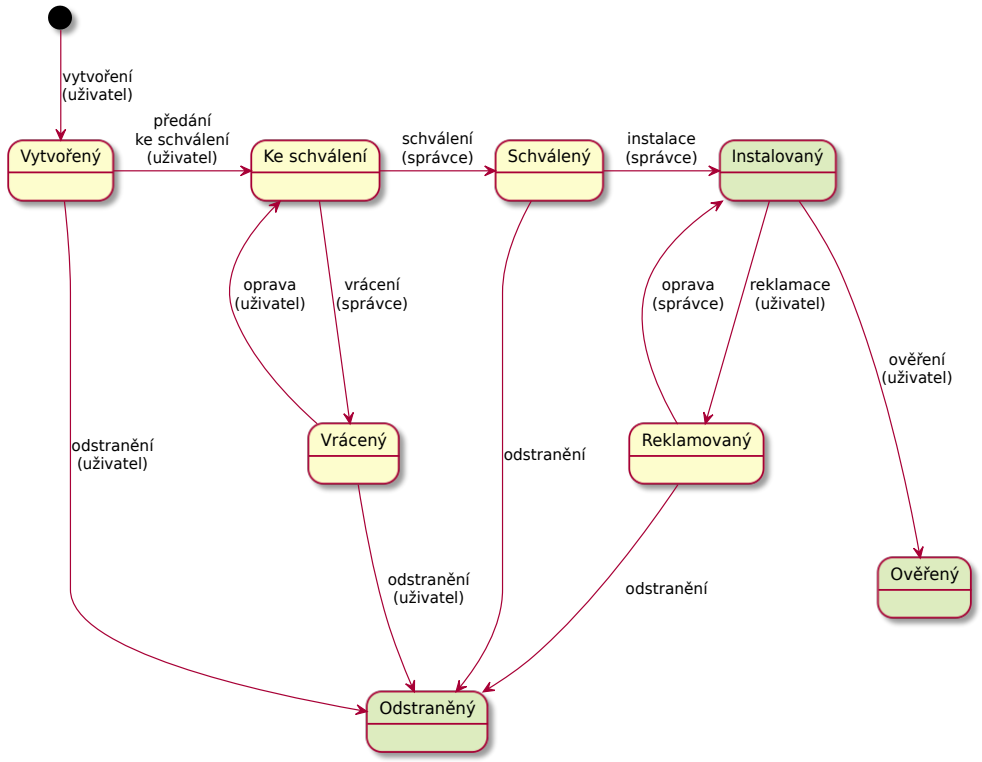
- zadávání požadavků (~ 2 měsíce předem),
- schválení požadavků a testovací instalace (~ 5 týdnů předem),
- uživatelské testování a reklamace (~ 3 týdny předem),
- finální instalace (~ 1 týden před zahájením semestru).

Proces, kterým požadavek prochází, musí být navržený tak, aby podpořil:

- vytvoření požadavku uživatelem,
- revizi požadavku správcem a jeho následnou instalaci,
- ověření instalovaného požadavku uživatelem s možností jeho reklamace.

Požadavek se nachází vždy v právě jednom stavu. Počátečním stavem je *Vytvořený*, koncové stavy jsou *Instalovaný*, *Ověřený* a *Odstraněný*.

[13] `https://en.wikipedia.org/wiki/Package_manager`




Obrázek 2.1: Životní cyklus požadavku na software (stavový diagram)

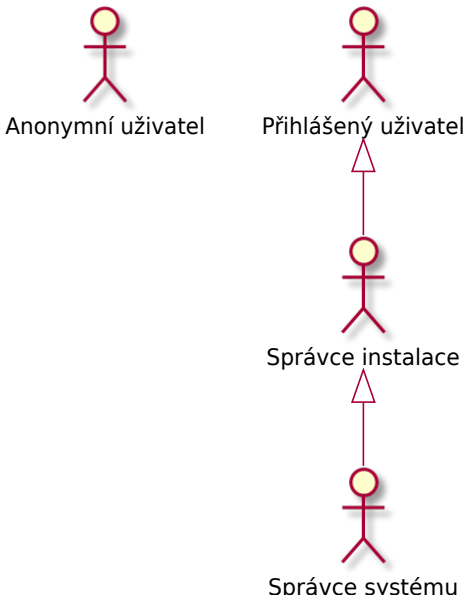
## 2.2 Případy užití

Sekce shrnuje případy užití aplikace pro správu počítačových učeben. Případy užití popisují chování systému z hlediska uživatele a vychází z požadavků na funkce, viz analytická část, kap. 1.4.

### 2.2.1 Aktéři

Existují čtyři typy aktérů, tj. rolí, které má uživatel ve vztahu k systému: **anonymní uživatel**, **přihlášený uživatel**, **správce instalace** a **správce systému**. Správce instalace je speciálním případem přihlášeného uživatele, může proto rovněž provádět všechny aktivity jako přihlášený uživatel. Systémový správce má ze všech aktérů práva nejvyšší a může provádět všechny níže definované aktivity.

Obrázky 2.3 a 2.4 znázorňují diagramy případů užití pro všechny definované aktéry. Detailním popisem scénářů jednotlivých případů užití se věnuje následující sekce.




Obrázek 2.2: Případy užití – aktéři

### 2.2.2 Scénáře případů užití

Každý případ užití obsahuje **název** včetně identifikátoru, **popis/cíl**, **aktéra** a **hlavní scénář** (případně jeho alternativy). Předpokladem pro každý scénář (vyjma prvního) je již přihlášený uživatel nacházející se na úvodní stránce aplikace.

#### 2.2.2.1 UC1: Přihlásit se

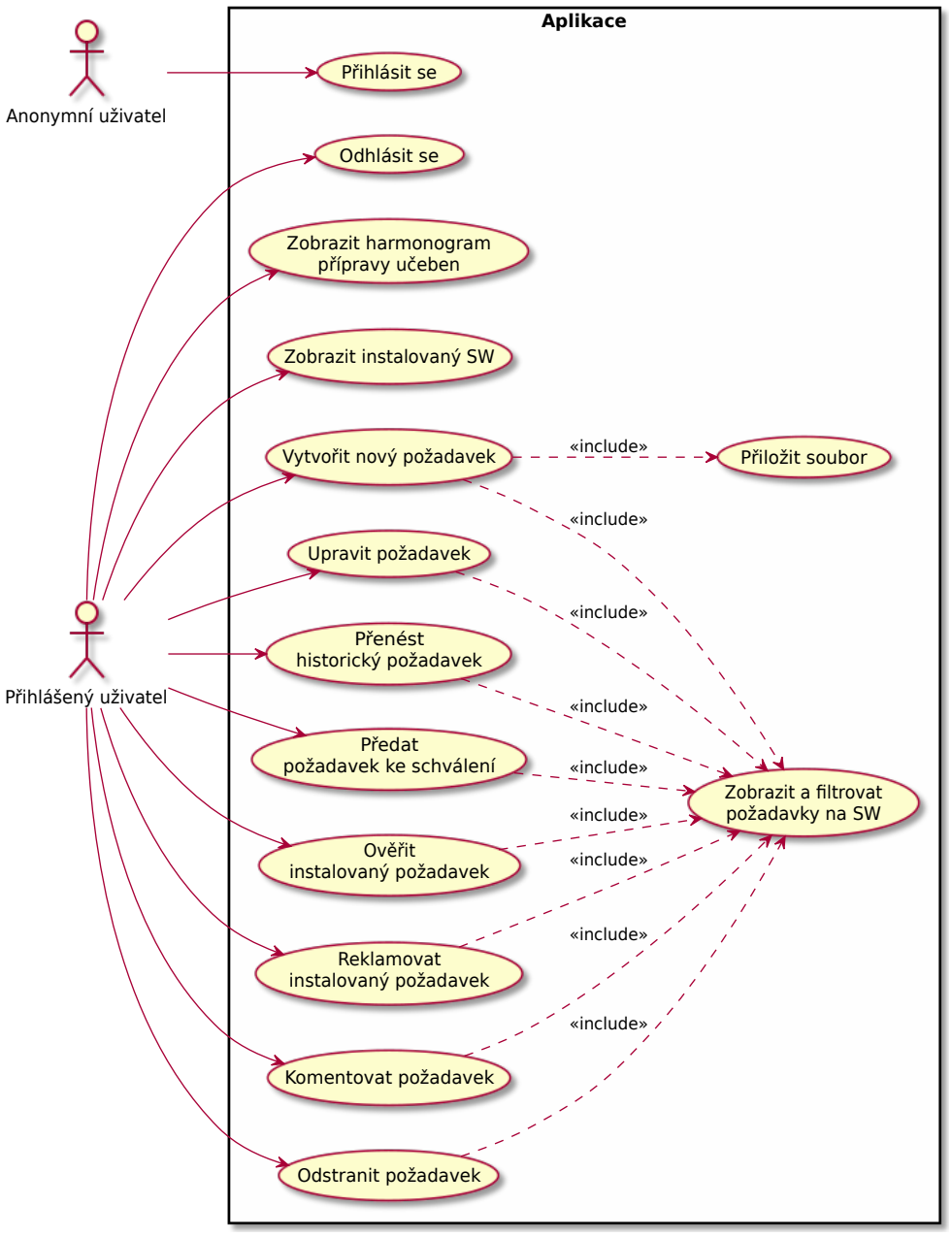
Případ užití popisuje proces přihlášení uživatele do aplikace.

**Aktér:** Anonymní uživatel

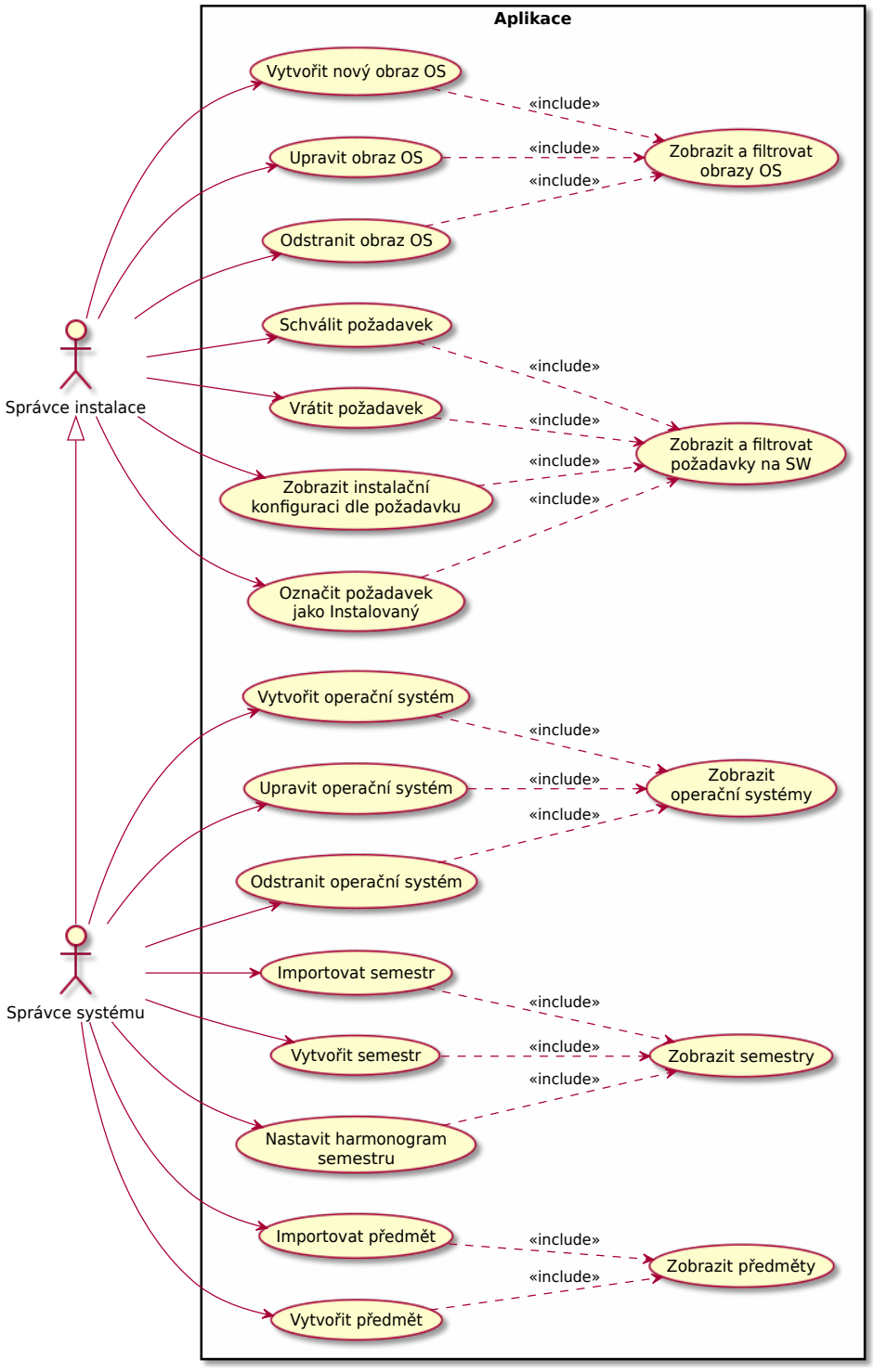
**Předpoklad:** Uživatel zná své uživatelské jméno a heslo.

**Scénář:**

1. Uživatel klikne na odkaz pro přihlášení.
2. Aplikace zobrazí přihlašovací formulář.
3. Uživatel zadá své uživatelské jméno a heslo.
4. Aplikace zvaliduje zadané přihlašovací údaje.
5. Uživatel je přesměrován na úvodní stránku jako přihlášený uživatel.




Obrázek 2.3: Případy užití – aktivity anonymního a přihlášeného uživatele




Obrázek 2.4: Případy užití – aktivity správce instalace a správce systému

#### 2.2.2.2 UC2: Odhlásit se

Případ užití popisuje proces odhlášení uživatele z aplikace.

**Aktér:** Přihlášený uživatel

**Scénář:**

1. Uživatel klikne na odkaz pro odhlášení.
2. Je přesměrován na domovskou stránku jako nepřihlášený uživatel.

#### 2.2.2.3 UC3: Zobrazit harmonogram přípravy učeben

Příprava programového vybavení počítačových učeben se řídí časovým plánem semestru, ve kterém probíhají zápisy do rozvrhu. Časový plán vytváří správce systému, viz UC30 (sekce 2.2.2.30) a přihlášenému uživateli je zobrazen na úvodní stránce aplikace.

**Aktér:** Přihlášený uživatel

**Scénář:**

1. Uživateli je na úvodní obrazovce zobrazen aktuální harmonogram.

#### 2.2.2.4 UC4: Zobrazit instalovaný software

Uživatel může zobrazit seznam instalovaného softwaru v počítačových učebnách a filtrovat jej podle semestru a operačního systému. Tuto funkci využije například vyučující pro ověření dostupnosti konkrétního softwaru, který si přeje využít v rámci svého cvičení v počítačových učebnách.

**Aktér:** Přihlášený uživatel

**Scénář:**

1. Uživatel klikne na položku menu „Instalovaný software“.
2. Aplikace zobrazí seznam instalovaného SW pro stávající semestr a výchozí volbu OS.
3. Uživatel upraví volbu semestru a operačního systému (volitelně).
4. Aplikace aktualizuje zobrazený seznam instalovaného SW.
5. Uživatel pohledem do seznamu či pomocí formulářového pole vyhledá potřebný SW.

#### 2.2.2.5 UC5: Zobrazit a filtrovat požadavky na software

Uživatel může zobrazit seznam svých historických či aktuálních požadavků na software. Každý požadavek je vztažen k výukovému semestru, obrazu operačního systému a jednomu či více vyučovaným předmětům, a nachází se v právě jednom stavu svého životního cyklu (viz sekce 2.1.2).

Podle výše zmíněných kritérií lze požadavky filtrovat. Uživatel dále může zobrazit pouze své požadavky, tj. požadavky jichž je autorem, nebo požadavky všech uživatelů.

**Aktér:** Přihlášený uživatel

**Scénář:**

1. Uživatel klikne na položku menu „Požadavky“.
2. Aplikace zobrazí seznam požadavků dle výchozích filtrů.
3. Uživatel upraví kritéria (semestr, obraz OS, předměty, stav, autorství).
4. Aplikace aktualizuje zobrazené požadavky.

#### 2.2.2.6 UC6: Vytvořit nový požadavek

Vytvoření nového požadavku na software je jedním z klíčových případů užití aplikace. Každý uživatel může vytvořit jeden nebo více požadavků. Požadavky lze zadávat pouze v určitém období (dáno harmonogramem), typicky několik týdnů před začátkem každého semestru.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Probíhá období zadávání požadavků (dle harmonogramu).

**Scénář:**

1. Případ užití začíná scénářem UC5, kde uživatel zvolil aktuální (tj. nejnovější) semestr.
2. Uživatel zvolí vytvoření nového požadavku.
3. Aplikace zobrazí formulář pro nový požadavek.
4. Uživatel upraví volbu obrazu OS (volitelně).
5. Uživatel vyplní předměty, pro které požadavek zadává.
6. Uživatel zadá pomocí formuláře požadavek software.

7. Uživatel uloží požadavek.

8. Aplikace zobrazí detail vytvořeného požadavku.

#### 2.2.2.7 UC7: Přiložit soubor

K požadavku vytvořenému vlastním, částečně strukturovaným, zadáním lze přiložit jeden nebo více souborů. Pro ostatní způsoby zadání požadavku (výběr z nabídky, vyhledání podle názvu balíčku) přiložení souboru postrádá smysl.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Požadavek se nachází ve stavu *Vytvořený* nebo *Vrácený*, uživatel je jeho autorem a byl vytvořen vlastním (částečně) strukturovaným zadáním.

**Scénář:**

1. Případ užití začíná scénářem UC5.

2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.

3. Aplikace zobrazí detail požadavku.

4. Uživatel zvolí přiložení souboru.

5. Aplikace zobrazí formulář pro přiložení souboru.

6. Uživatel vloží soubor a dokončí akci.

7. Aplikace zobrazí detail požadavku s přiloženým souborem.

#### 2.2.2.8 UC8: Upravit požadavek

Před odesláním vytvořeného požadavku ke schválení správci jej může uživatel upravit.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Požadavek se nachází ve stavu *Vytvořený* nebo *Vrácený* a uživatel je jeho autorem.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí upravení požadavku.
5. Aplikace zobrazí formulář požadavku s předvyplněnými hodnotami.
6. Uživatel upraví požadavek.
7. Uživatel uloží požadavek.
8. Aplikace zobrazí detail upraveného požadavku.

#### 2.2.2.9 UC9: Přenést historický požadavek

Aplikace podporuje přenos (zkopírování) historických požadavků do nového semestru. Požadavky lze přenést pouze jednotlivě.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Existuje alespoň jeden uživatelem vytvořený požadavek v některém z předchozích semestrů a probíhá období zadávání požadavků (dle harmonogramu).

**Scénář:**

1. Případ užití začíná scénářem UC5, kde uživatel zvolil libovolný semestr předcházející novému semestru.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí zkopírování požadavku do nového semestru.
5. Aplikace zobrazí detail přeneseného požadavku.

#### 2.2.2.10 UC10: Předat požadavek ke schválení

Nově vytvořený či přenesený požadavek musí být před vlastní instalací zrevidován příslušným správcem. Za tímto účelem je nutné předat požadavek ke schválení.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Požadavek se nachází ve stavu *Vytvořený* nebo *Vrácený* a uživatel je jeho autorem.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí předání požadavku ke schválení.
5. Aplikace přesune požadavek do stavu *Ke schválení* a aktualizuje zobrazený detail.

#### 2.2.2.11 UC11: Ověřit instalovaný požadavek

Před začátkem semestru může uživatel na testovací stanici vyzkoušet funkčnost požadovaného softwaru. Ověřením požadavku uživatel potvrzuje správnost instalace příslušného softwaru. K této akci lze volitelně připojit komentář.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Požadavek se nachází ve stavu *Instalovaný* a uživatel je jeho autorem.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí ověření požadavku.
5. Aplikace zobrazí formulář pro komentář.
6. Uživatel vloží komentář (volitelně) a dokončí akci.
7. Aplikace přesune požadavek do stavu *Ověřený* a aktualizuje zobrazený detail.

#### 2.2.2.12 UC12: Reklamovat instalovaný požadavek

V případě, že požadavek nebyl naplněn – software nebyl správně nainstalován nebo zcela chybí – jej může uživatel reklamovat. Tato akce vynucuje přiložení komentáře.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Požadavek se nachází ve stavu *Instalovaný* a probíhá období reklamace požadavků (dle harmonogramu).

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí reklamování požadavku.
5. Aplikace zobrazí formulář pro komentář.
6. Uživatel vloží komentář a dokončí akci.
7. Aplikace přesune požadavek do stavu *Reklamovaný* a aktualizuje jeho detail.

#### 2.2.2.13 UC13: Komentovat požadavek

Při neúspěšné instalaci nebo reklamaci chybné instalace je nutné umožnit komentování požadavku jak uživatelem, tak i správcem.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Uživatel je autorem požadavku či správcem příslušného obrazu OS.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí komentování požadavku.
5. Aplikace zobrazí formulář pro nový komentář.
6. Uživatel vloží komentář a uloží jej.
7. Aplikace aktualizuje detail požadavku s komentáři.

#### 2.2.2.14 UC14: Odstranit požadavek

Případ užití popisuje proces odstranění uživatelského požadavku z aplikace.

**Aktér:** Přihlášený uživatel

**Předpoklad:** Uživatel je autorem požadavku.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí odstranění požadavku.
5. Aplikace zobrazí potvrzovací dialog.
6. Uživatel potvrdí akci.
7. Aplikace přesune požadavek do stavu *Odstraněný* a zobrazí seznam požadavků.

#### 2.2.2.15 UC15: Zobrazit a filtrovat obrazy OS

Každý uživatelský požadavek na software je vztažen k právě jednomu obrazu operačního systému. Správa obrazů probíhá na speciální obrazovce, k níž běžný uživatel nemá přístup.

**Aktér:** Správce instalace

**Scénář:**

1. Uživatel klikne na položku menu „Obrazy OS“.
2. Aplikace zobrazí seznam obrazů OS pro stávající semestr.
3. Uživatel upraví volbu semestru (volitelně).
4. Aplikace aktualizuje zobrazený seznam obrazů.

#### 2.2.2.16 UC16: Vytvořit nový obraz OS

Před zadáváním požadavků pro nový semestr je nutné vytvořit speciální obraz pro každý podporovaný operační systém.

**Aktér:** Správce instalace

**Scénář:**

1. Případ užití začíná scénářem UC15.
2. Uživatel zvolí vytvoření nového obrazu.
3. Aplikace zobrazí formulář pro nový obraz.
4. Uživatel upraví volbu semestru (volitelně).
5. Uživatel zadá název obrazu a přiřadí operační systém.
6. Uživatel uloží obraz.
7. Aplikace zobrazí seznam obrazů.

#### 2.2.2.17 UC17: Upravit obraz OS

Vytvořený obraz je možné částečně editovat; není možné měnit semestr ani operační systém.

**Aktér:** Správce instalace

**Scénář:**

1. Případ užití začíná scénářem UC15.
2. Uživatel zvolí upravení obrazu.
3. Aplikace zobrazí formulář obrazu s předvyplněnými hodnotami.
4. Uživatel upraví obraz.
5. Uživatel uloží obraz.
6. Aplikace zobrazí seznam obrazů.

#### 2.2.2.18 UC18: Odstranit obraz OS

Při odstranění obrazu dojde rovněž k odstranění všech požadavků k němu vztažených.

**Aktér:** Správce instalace

**Scénář:**

1. Případ užití začíná scénářem UC15.
2. Uživatel zvolí odstranění obrazu.
3. Aplikace zobrazí potvrzovací dialog.
4. Uživatel potvrdí akci.
5. Aplikace odstraní obraz a zobrazí seznam existujících obrazů.

#### 2.2.2.19 UC19: Schválit požadavek

Nový požadavek na software podléhá schválení správcem příslušného obrazu OS. Je-li požadavek v pořádku, správce jej schválí. V opačném případě jej spolu s komentářem vrátí k přepracování, viz UC20.

**Aktér:** Správce instalace

**Předpoklad:** Požadavek se nachází ve stavu *Ke schválení*.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí schválení požadavku.
5. Aplikace zobrazí formulář pro komentář.
6. Uživatel vloží komentář (volitelně) a dokončí akci.
7. Aplikace přesune požadavek do stavu *Schválený* a aktualizuje zobrazený detail.

#### 2.2.2.20 UC20: Vrátit požadavek

Případ užití popisuje proces vrácení nevyhovujícího uživatelského požadavku.

**Aktér:** Správce instalace

**Předpoklad:** Požadavek se nachází ve stavu *Ke schválení*.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí vrácení požadavku.
5. Aplikace zobrazí formulář pro komentář.
6. Uživatel vloží komentář a dokončí akci.
7. Aplikace přesune požadavek do stavu *Vrácený* a aktualizuje zobrazený detail.

#### 2.2.2.21 UC21: Zobrazit instalační konfiguraci dle požadavku

Správce může z požadavku vygenerovat[14] a zobrazit instalační konfiguraci pro automatizační nástroj Ansible.

**Aktér:** Správce instalace

**Předpoklad:** Požadavek se nachází ve stavu *Schválený* a byl zadán vyhledáním ověřeného balíčku (pouze OS Linux).

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel zvolí zobrazení instalační konfigurace.
5. Aplikace zobrazí formulář s konfigurací.
6. Uživatel upraví konfiguraci (volitelně).
7. Uživatel zkopíruje konfiguraci do schránky.

#### 2.2.2.22 UC22: Označit požadavek jako instalovaný

Vlastní instalace softwaru dle požadavku uživatele probíhá mimo aplikaci. Aplikace disponuje pouze tlačítkem, pomocí něhož správce označí požadavek jako *Instalovaný*.

[14] Dle předpřipravené šablony pro každý operační systém.

**Aktér:** Správce instalace

**Předpoklad:** Požadavek se nachází ve stavu *Schválený* nebo *Reklamovaný*.

**Scénář:**

1. Případ užití začíná scénářem UC5.
2. Uživatel vyhledá požadavek v seznamu a zvolí zobrazení jeho detailu.
3. Aplikace zobrazí detail požadavku.
4. Uživatel označí požadavek jako instalovaný.
5. Aplikace zobrazí formulář pro komentář.
6. Uživatel vloží komentář (volitelně) a dokončí akci.
7. Aplikace přesune požadavek do stavu *Instalovaný* a aktualizuje zobrazený detail.

#### 2.2.2.23 UC23: Zobrazit operační systémy

Podmínkou pro vytvoření obrazu OS je předchozí vytvoření patřičného operačního systému. Obrazovku pro správu operačních systémů může zobrazit pouze správce systému.

**Aktér:** Správce systému

**Scénář:**

1. Uživatel klikne na položku menu „Operační systémy“.
2. Aplikace zobrazí seznam operačních systémů.

#### 2.2.2.24 UC24: Vytvořit nový operační systém

Každý operační systém je jednoznačně definován trojicí atributů: typ[15], verze a architektura. Dále je možné přiřadit vyhledávač softwarových balíčků (je-li pro daný OS k dispozici) a definovat šablonu instalační konfigurace. Posledním krokem je definování seznamu správců obrazů daného operačního systému.

**Aktér:** Správce systému

[15] Jedna položka ze seznamu podporovaných OS, definovaných v konfiguraci aplikace.

**Scénář:**

1. Případ užití začíná scénářem UC23.
2. Uživatel zvolí vytvoření nového operačního systému.
3. Aplikace zobrazí formulář pro nový operační systém.
4. Uživatel přiřadí typ, verzi a architekturu OS.
5. Uživatel přiřadí vyhledávač softwarových balíčků a definuje šablonu instalační konfigurace (volitelně).
6. Uživatel definuje seznam správců obrazů daného OS.
7. Uživatel uloží operační systém.
8. Aplikace zobrazí seznam operačních systémů.

#### 2.2.2.25 UC25: Upravit operační systém

Vytvořený operační systém je možné částečně editovat; není možné měnit atributy jednoznačně identifikující daný OS (typ, verze, architektura).

**Aktér:** Správce systému

**Scénář:**

1. Případ užití začíná scénářem UC23.
2. Uživatel zvolí upravení operačního systému.
3. Aplikace zobrazí formulář operačního systému s předvyplněnými hodnotami.
4. Uživatel upraví operační systém.
5. Uživatel uloží operační systém.
6. Aplikace zobrazí seznam operačních systémů.

#### 2.2.2.26 UC26: Odstranit operační systém

Odstranění operačního systému způsobí rovněž odstranění všech odvozených obrazů OS a požadavků k nim vztažených.

**Aktér:** Správce systému

**Scénář:**

1. Případ užití začíná scénářem UC23.
2. Uživatel zvolí odstranění operačního systému.
3. Aplikace zobrazí potvrzovací dialog.
4. Uživatel potvrdí akci.
5. Aplikace odstraní operační systém a zobrazí seznam existujících operačních systémů.

#### 2.2.2.27 UC27: Zobrazit semestry

Aplikace pracuje se studijními semestry. Data o aktuálních semestrech jsou importována ze studijního informačního systému KOS (UC28). V případě jeho nedostupnosti je možné nový semestr vytvořit také manuálně (UC29). Správu semestrů zajišťuje správce systému.

**Aktér:** Správce systému

**Scénář:**

1. Uživatel klikne na položku menu „Semestry“.
2. Aplikace zobrazí seznam semestrů.

#### 2.2.2.28 UC28: Importovat semestr

Import nových semestrů ze systému KOS je zautomatizovaný[16]. V případě potřeby je však možné jednorázově importovat semestr identifikovaný svým kódem dle KOS[17].

**Aktér:** Správce systému

**Předpoklad:** V databázi KOS existuje semestr s daným kódem a dosud nebyl importován.

**Scénář:**

1. Případ užití začíná scénářem UC27.
2. Uživatel zadá kód semestru a zvolí jeho import.
3. Aplikace importuje semestr a aktualizuje zobrazený seznam semestrů.

[16] Aplikace se periodicky synchronizuje s databází KOS.
[17] Např. kód B162 odpovídá letnímu semestru 2016/2017.

#### 2.2.2.29 UC29: Vytvořit semestr

Aplikace podporuje také manuální vytvoření semestru, čímž se stává nezávislou na systému KOS.

**Aktér:** Správce systému

**Předpoklad:** Semestr s daným kódem dosud nebyl importován ani ručně vytvořen.

**Scénář:**

1. Případ užití začíná scénářem UC27.
2. Uživatel zadá unikátní kód a název semestru, datum jeho zahájení a volitelně také datum ukončení.
3. Uživatel uloží semestr.
4. Aplikace aktualizuje zobrazený seznam semestrů.

#### 2.2.2.30 UC30: Nastavit harmonogram semestru

Před zahájením etapy sběru požadavků musí správce systému nastavit harmonogram přípravy učeben pro nový semestr. Harmonogram lze nastavit ručně nebo automaticky vygenerovat[18].

**Aktér:** Správce systému

**Předpoklad:** Semestr má již určené datum zahájení.

**Hlavní scénář:**

1. Případ užití začíná scénářem UC27.
2. Uživatel zvolí nastavení harmonogramu semestru.
3. Aplikace zobrazí formulář harmonogramu.
4. Uživatel ručně nastaví počáteční termíny jednotlivých fází harmonogramu.
5. Uživatel uloží harmonogram.
6. Aplikace zobrazí seznam semestrů.

[18]Dle data zahájení semestru a časových intervalů jednotlivých fází přípravy učeben specifikovaných v konfiguraci aplikace.

**Alternativní scénář:**

4. Uživatel zvolí automatické vygenerování harmonogramu.
5. Uživatel ručně upraví termíny (volitelně).
6. Uživatel uloží harmonogram.
7. Aplikace zobrazí seznam semestrů.

#### 2.2.2.31 UC31: Zobrazit předměty

Jak již bylo zmíněno, každý softwarový požadavek má vazbu na jeden nebo více vyučovaných předmětů, a je proto nutné umožnit jejich správu. Stejně jako v případě semestrů, také zde se pracuje s lokální kopií dat z KOS.

**Aktér:** Správce systému

**Scénář:**

1. Uživatel klikne na položku menu „Předměty“.
2. Aplikace zobrazí seznam předmětů.

#### 2.2.2.32 UC32: Importovat předmět

Proces importu nových předmětů je identický se semestry (UC28). Import/synchronizace s KOS probíhá automaticky v pravidelných časových intervalech. Předmět je možné importovat také jednorázově podle jeho kódu[19].

**Aktér:** Správce systému

**Předpoklad:** V databázi KOS existuje předmět s daným kódem a dosud nebyl importován.

**Scénář:**

1. Případ užití začíná scénářem UC31.
2. Uživatel zadá kód předmětu a zvolí jeho import.
3. Aplikace importuje předmět a aktualizuje zobrazený seznam předmětů.

[19]Např. kód BI-PA1 odpovídá předmětu Programování a algoritmizace 1.

#### 2.2.2.33 UC33: Vytvořit předmět

Poslední případ užití popisuje proces manuálního vytvoření předmětu – opět z důvodu zachování nezávislosti aplikace na systému KOS.

**Aktér:** Správce systému

**Předpoklad:** Předmět s daným kódem dosud nebyl importován ani ručně vytvořen.

**Scénář:**

1. Případ užití začíná scénářem UC31.
2. Uživatel zadá unikátní kód a název předmětu.
3. Uživatel uloží předmět.
4. Aplikace aktualizuje zobrazený seznam předmětů.

## 2.3 Lo-Fi prototypy a popis obrazovek aplikace

Sekce popisuje obrazovky aplikace a funkční prvky uživatelského rozhraní. Popis obrazovek názorně ilustrují jejich Lo-Fi[20] prototypy vytvořené pomocí open source nástroje Draw.io [3].

V průběhu tvorby prototypů byla odhalena (a posléze opravena) řada chyb v návrhu. Zde jsou uvedeny pouze finální verze prototypů, které pomohou představit si vzhled výsledné aplikace (zejm. po stránce rozložení ovládacích prvků, nikoli grafické).

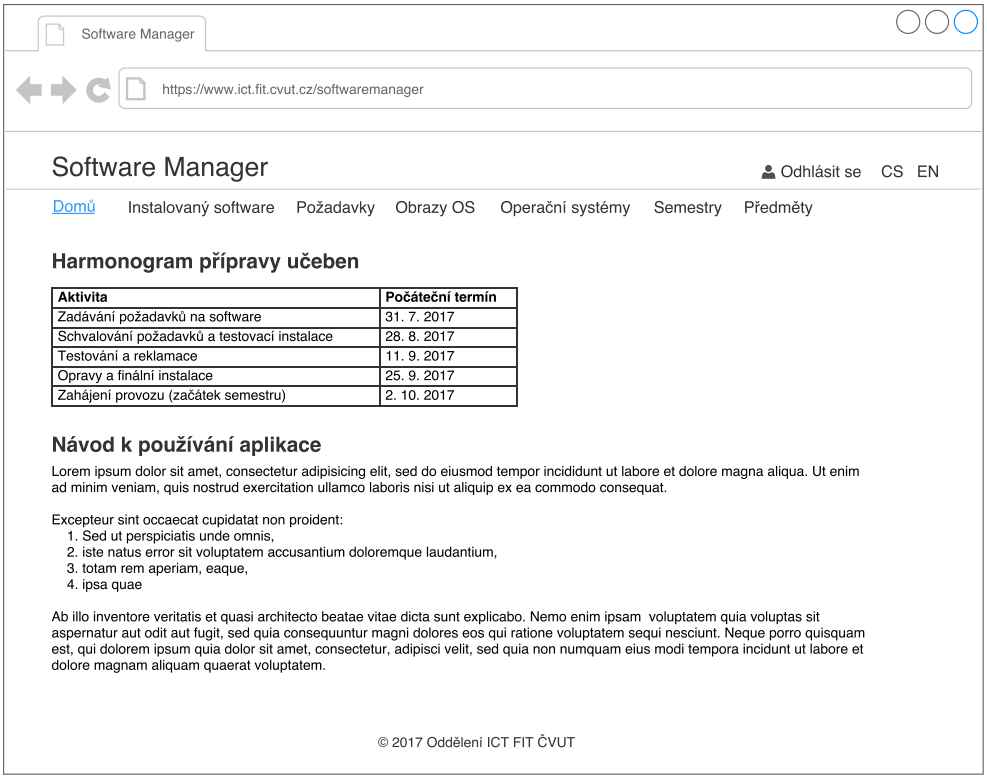
Aplikace obsahuje celkem třináct typů obrazovek:

- Domovská obrazovka,
- Instalovaný software,
- Požadavky,
- Nový požadavek,
- Přílohy požadavku,
- Detail požadavku,
- Operační systémy,
- Nový operační systém,
- Obrazy OS,
- Nový obraz,
- Semestry,
- Harmonogram semestru,
- Předměty.

[20] `http://www.usabilityfirst.com/glossary/low-fidelity-prototype/`

### 2.3.1 Domovská obrazovka

Domovská obrazovka aplikace obsahuje základní informace pro uživatele jako aktuální harmonogram přípravy učeben a návod k používání aplikace. Harmonogram je aktualizován při každé jeho změně na stránce Harmonogram semestru (sekce 2.3.9). Stejně jako ostatní obrazovky obsahuje hlavní navigaci pro přechod mezi jednotlivými obrazovkami aplikace, přihlášení, resp. odhlášení uživatele a změnu jazykové mutace aplikace.

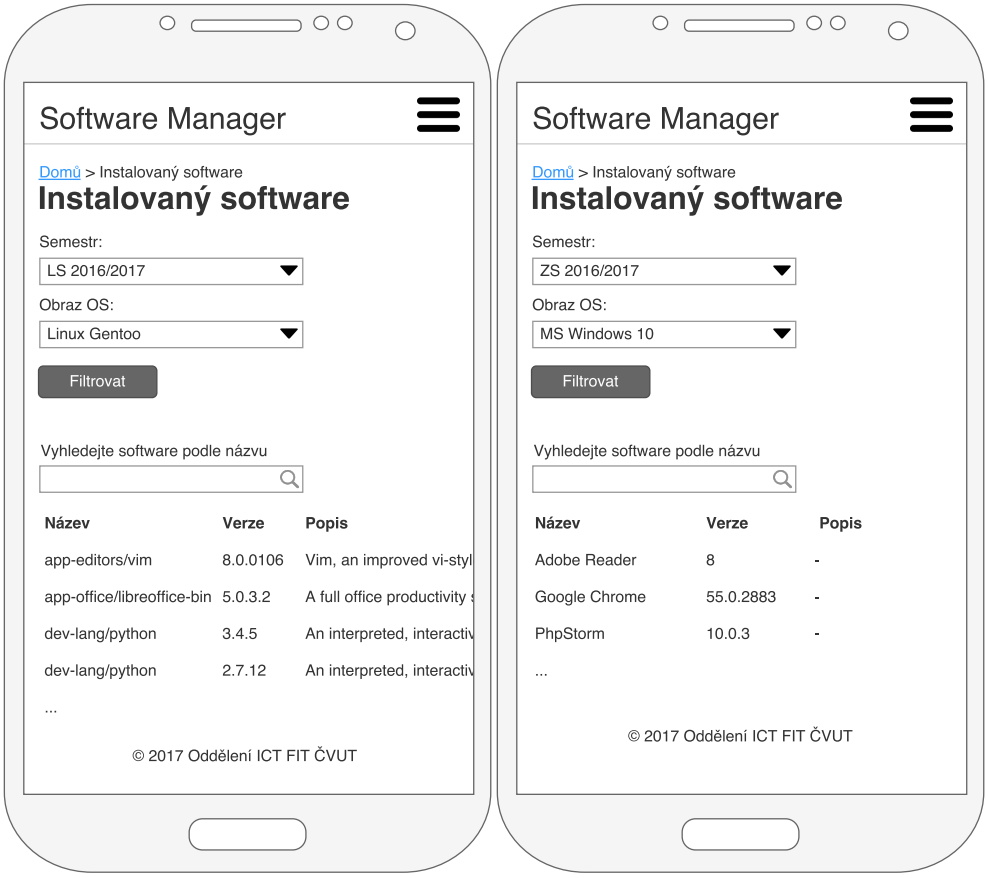



Obrázek 2.5: Lo-Fi prototypy – domovská obrazovka

### 2.3.2 Instalovaný software

Obrazovka zobrazuje instalovaný software v počítačových učebnách. Obsahuje formulář pro filtrování softwaru podle semestru a obrazu operačního systému (rozbalovací nabídka). Ve výchozím zobrazení je zvolen stávající semestr a nejpoužívanější obraz OS.

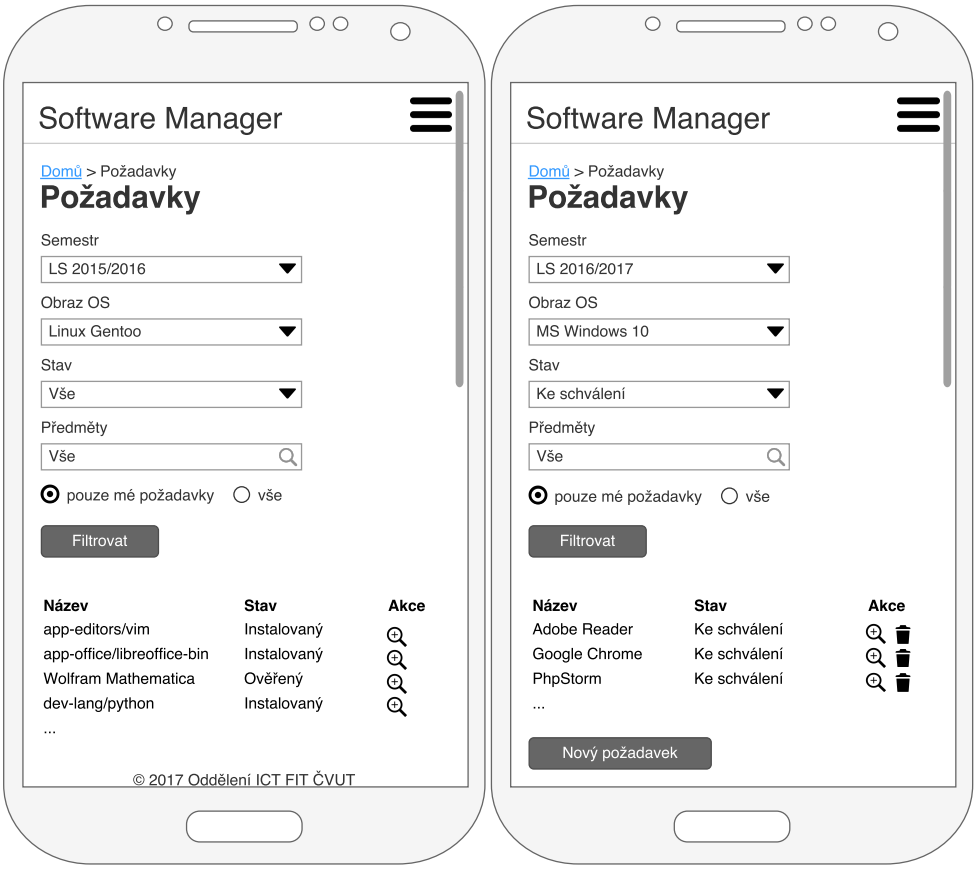
Sada dostupných obrazů se vytváří speciálně pro každý semestr, jejich nabídka se tedy napříč semestry může lišit. Z tohoto důvodu je vždy při změně preference semestru automaticky aktualizována nabídka obrazů.




Obrázek 2.6: Lo-Fi prototypy – obrazovka Instalovaný software

### 2.3.3 Požadavky

Obrazovka slouží uživateli k zobrazení a filtrování aktuálních i historických požadavků a zároveň je výchozím místem pro zadávání nových požadavků.




Obrázek 2.7: Lo-Fi prototypy – obrazovka Požadavky

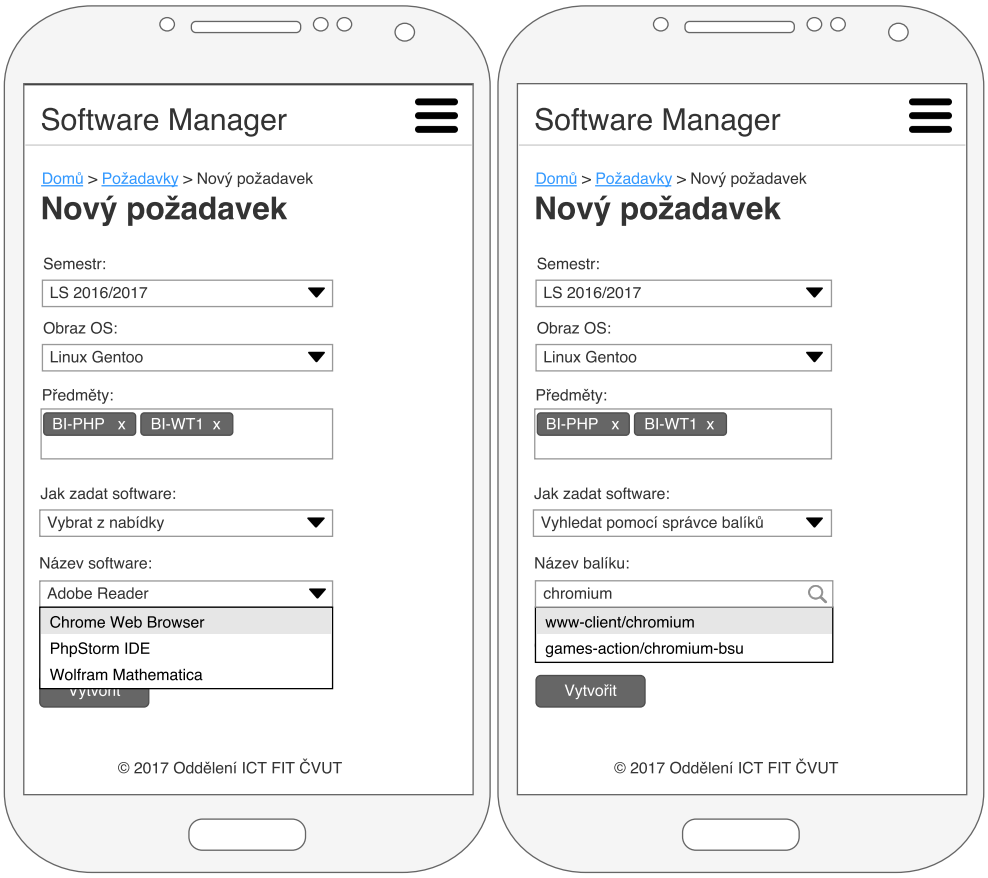
Požadavky lze filtrovat podle semestru, obrazu OS, předmětů a stavu. Dále je možné zvolit zobrazení požadavků všech uživatelů, nebo pouze aktuálně přihlášeného uživatele. Ve výchozím stavu aplikace zobrazuje všechny požadavky pro nový semestr.

Je-li vybrán nový semestr, jednotlivé požadavky lze editovat či odstranit. Obrazovka navíc obsahuje tlačítko pro přidání nového požadavku (obrazovku Nový požadavek popisuje následující sekce). Pokud je vybrán některý z předchozích semestrů, požadavky obsahují tlačítko pro jejich přenos do nového semestru. Editace existujících ani vytváření nových požadavků v tomto případě nejsou možné.

### 2.3.4 Nový požadavek a Přílohy požadavku

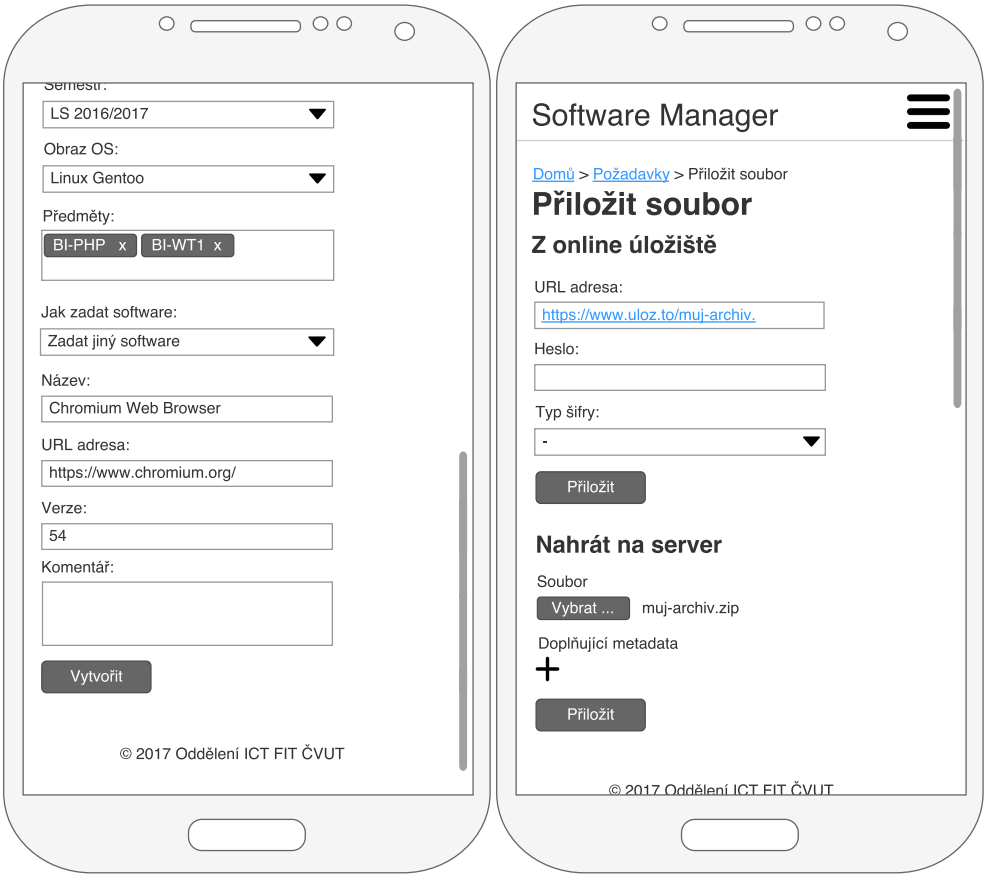
Obrazovka **Nový požadavek** umožňuje zadat požadavek na software pro nový semestr. Obsahuje formulář pro výběr obrazu OS, přiřazení souvisejících předmětů a volbu způsobu zadání samotného požadavku.

Dvojice obrazovek níže zobrazuje zadání požadavku výběrem z nabídky (vlevo) a vyhledáním pomocí názvu balíčku (vpravo). Druhá možnost je dostupná pouze pro obrazy operačních systémů s nastaveným vyhledávačem softwarových balíčků (viz sekce 2.3.6, obrazovka Nový operační systém).




Obrázek 2.8: Lo-Fi prototypy – obrazovka Nový požadavek

Třetí obrazovka zobrazuje vlastní, částečně strukturované, zadání požadavku. Formulář obsahuje čtyři textová pole pro název softwaru (povinné), URL adresu, verzi a komentář. K takto vytvořenému požadavku lze připojit jeden nebo více souborů. Přiložení souboru je předmětem obrazovky čtvrté.




Obrázek 2.9: Lo-Fi prototypy – obrazovky Nový požadavek a Přílohy požadavku

Aplikace podporuje dva způsoby přiložení souboru:

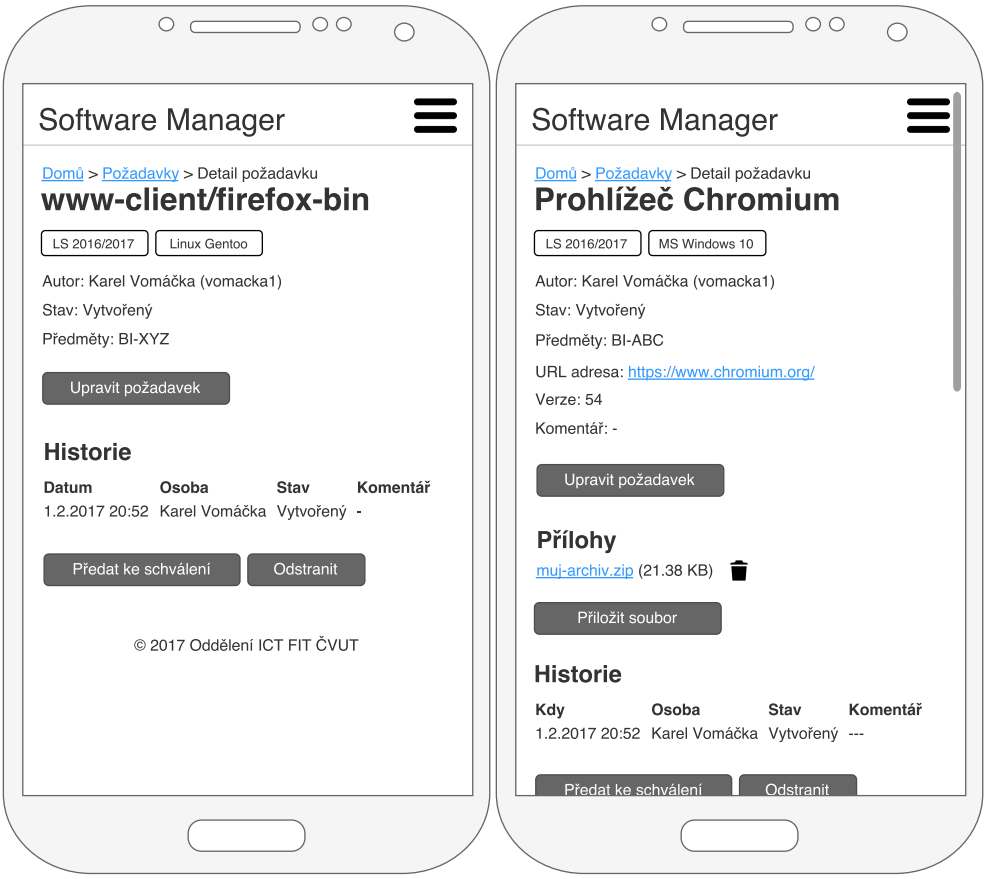
1. poskytnutí odkazu pro stažení souboru z online úložiště,
2. nahrání na server; velikost takového souboru je omezena.

Při zvolení první možnosti je nutné zadat URL adresu souboru a volitelně heslo, resp. klíčovou frázi a typ použité šifry. K souboru nahrávanému na server lze kromě povinného názvu volitelně připojit doplňující metadata (množina dvojic *klíč: hodnota*).

### 2.3.5 Detail požadavku

Obrazovka Detail požadavku zobrazuje následující informace:

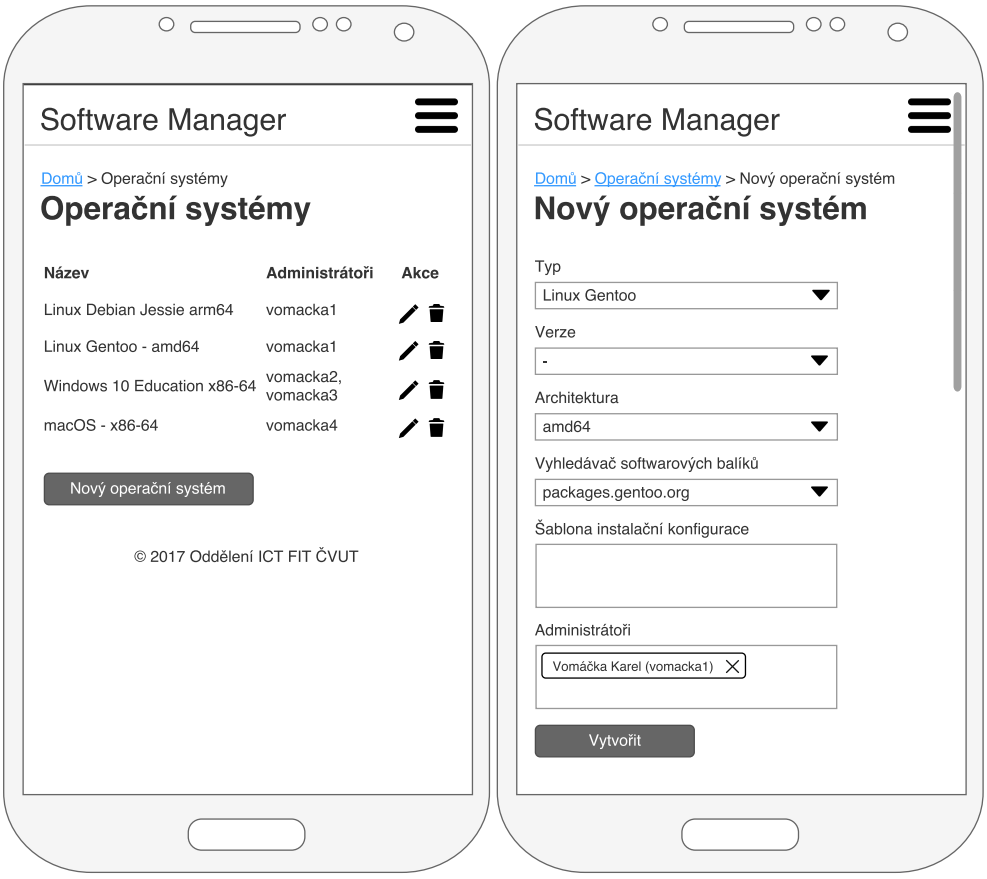
- název softwaru/balíčku (jako nadpis),
- přidružený semestr a obraz OS (ve formě štítků),
- jméno autora, stav a předměty, ke kterým se požadavek vztahuje,
- URL adresu, verzi a komentář (pouze pro vlastní požadavek),
- přílohy (pouze pro vlastní požadavek, viz obrazovka Přílohy požadavku),
- historii aktivit (změny stavu v čase, komentáře – formou tabulky),
- tlačítka akcí (v závislosti na stavu požadavku a roli uživatele).




Obrázek 2.10: Lo-Fi prototypy – obrazovka Detail požadavku

### 2.3.6 Operační systémy a Nový operační systém

Obrazovka **Operační systémy** zobrazuje přehled vytvořených operačních systémů ve formě tabulky. Každý řádek obsahuje název operačního systému složený z trojice atributů (typ, verze, architektura), uživatelská jména přiřazených správců a tlačítka akcí editace a odstranění. Obrazovka navíc obsahuje tlačítko pro přidání nového operačního systému, viz obrazovka Nový operační systém.



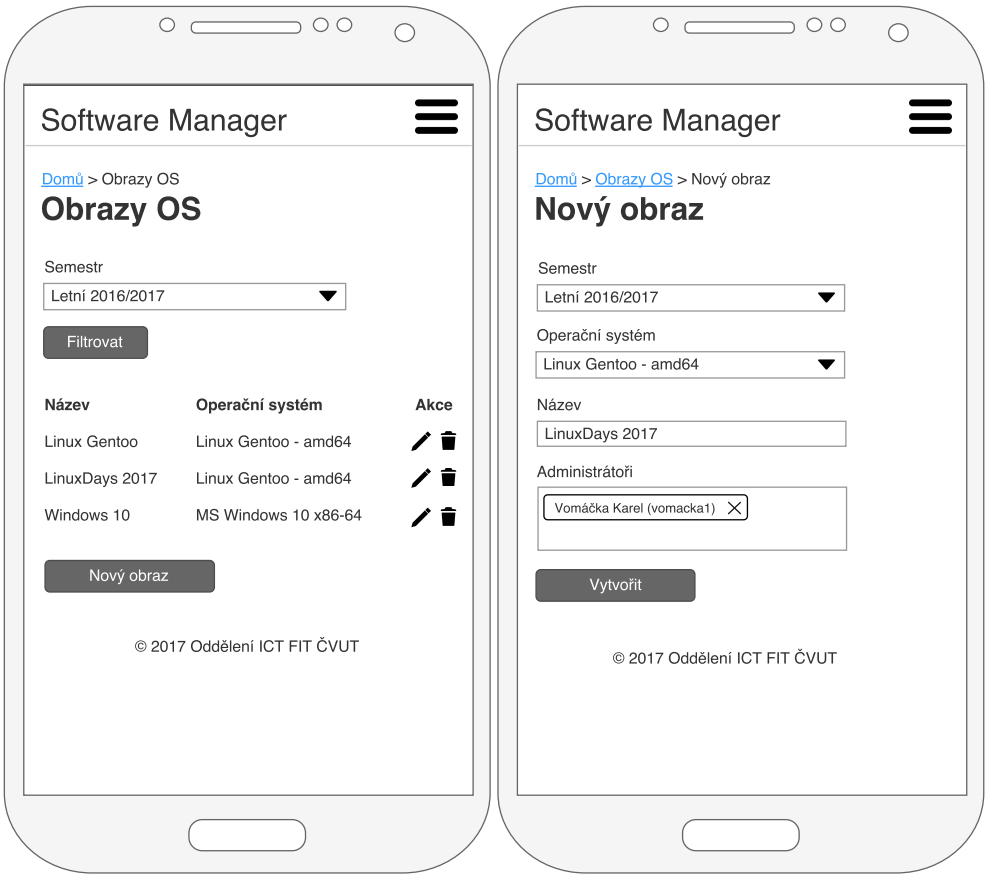

Obrázek 2.11: Lo-Fi prototypy – obrazovky Operační systémy a Nový OS

Obrazovka **Nový operační systém** slouží nejen pro vytvoření nového operačního systému, ale také pro editaci existujícího OS. Obsahuje formulář s poli pro typ, verzi, architekturu, vyhledávač softwarových balíčků, šablonu instalační konfigurace a přiřazení správců daného operačního systému. Při editaci nelze měnit typ, verzi ani architekturu operačního systému (formulářová pole jsou zablokovaná).

### 2.3.7 Obrazy OS a Nový obraz

Obrazovky s obrazy OS jsou obdobou předchozích obrazovek s operačními systémy. Obrazy lze navíc filtrovat podle semestru, pro který byly vytvořeny; ve výchozím stavu je přednastaven aktuální semestr.

Obrazovka **Obrazy OS** zobrazuje formou tabulky přehled vytvořených obrazů. Každý řádek odpovídá jednomu obrazu a obsahuje jeho název, operační systém, seznam administrátorů a tlačítka akcí editace a odstranění. Ve spodní části obrazovky se nachází tlačítko „Nový obraz“ s patrným účelem.

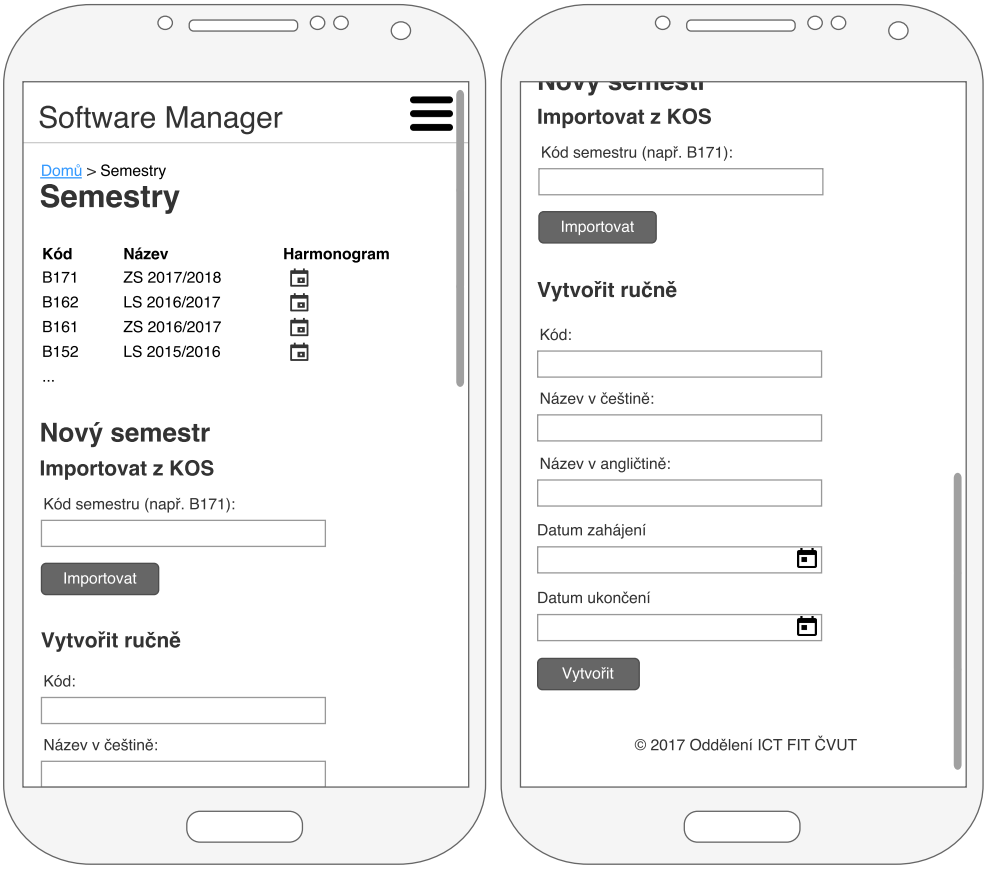



Obrázek 2.12: Lo-Fi prototypy – obrazovky Obrazy OS a Nový obraz OS

Formulář na obrazovce **Nový obraz** obsahuje pole pro volbu semestru a operačního systému (rozbalovací nabídka), název obrazu (textové pole) a přiřazení administrátorů obrazu (nabídka s výběrem více možností). Formulář slouží rovněž pro editaci obrazu, kde pole pro semestr a operační systém jsou zablokovaná.

### 2.3.8 Semestry

Obrazovka nabízí přehled existujících semestrů a je výchozím místem pro nastavení harmonogramu vybraného semestru, viz obrazovka Harmonogram semestru. Import semestrů probíhá automaticky; v případě potřeby však obrazovka umožňuje také jednorázový import či ruční vytvoření nového semestru.



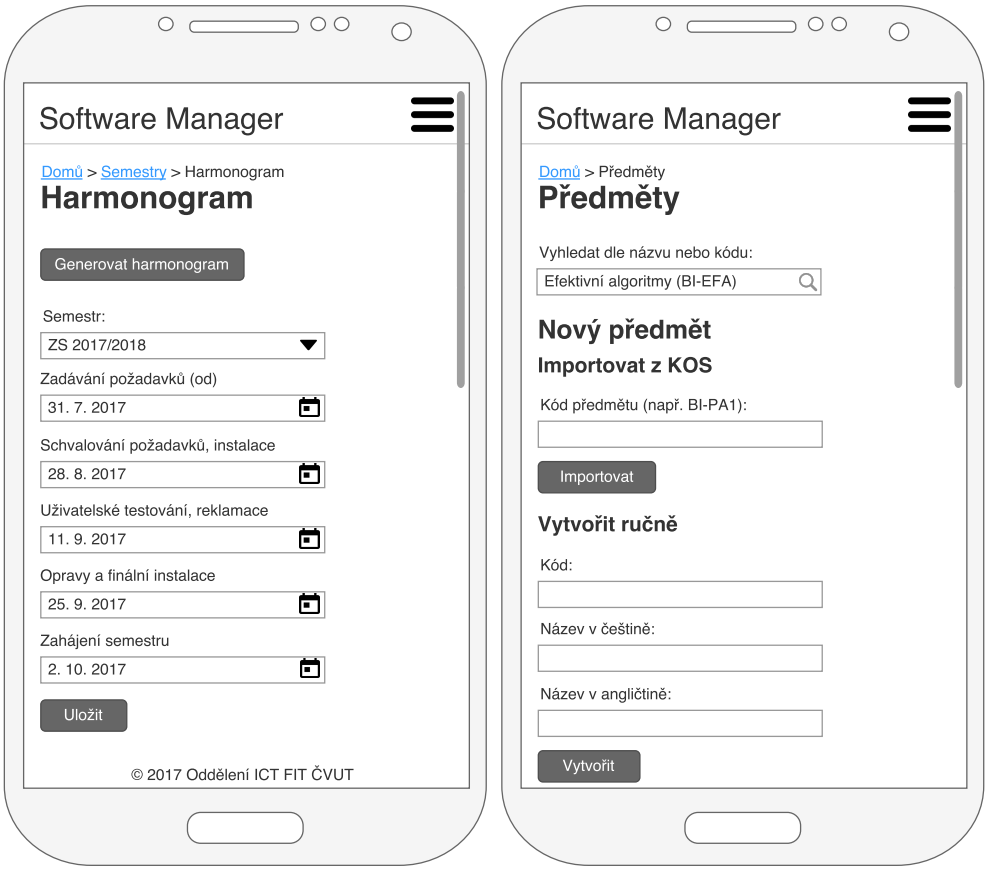

Obrázek 2.13: Lo-Fi prototypy – obrazovka Semestry

Pro import semestru stačí poskytnout jeho kód z KOS. Varianta ručního vytvoření vyžaduje kód, český a anglický název, datum zahájení a volitelně datum ukončení semestru. Po kliknutí na poslední dvě zmíněná pole je zobrazen kalendář pro výběr data.

### 2.3.9 Harmonogram semestru a Předměty

Obrazovka **Harmonogram semestru** slouží správci systému k nastavení časového plánu přípravy učeben pro daný semestr. Harmonogram nastavený na této obrazovce je zobrazen rovněž na obrazovce domovské.

- Tlačítko „Generovat harmonogram“ slouží k automatickému vygenerování harmonogramu (viz UC30, sekce 2.2.2.30).
- Pole pro výběr semestru má pouze informativní charakter, volba nelze změnit.
- Kliknutím na libovolné jiné pole se zobrazí kalendář pro výběr data.




Obrázek 2.14: Lo-Fi prototypy – obrazovky Harmonogram semestru a Předměty

Obrazovka **Předměty** obsahuje vyhledávací pole pro existující předměty. Dále umožňuje import nového předmětu dle jeho KOS kódu a ruční vytvoření zadáním kódu a názvu v češtině a angličtině.

## 2.4 Doménový model a popis entit

Sekce popisuje objekty (entity) aplikace, jejich atributy a relace. Doménový model, nazývaný též konceptuální nebo analytický, je znázorněn pomocí ER[21] diagramu na obrázku 2.15.

### 2.4.1 Requirement

Entita `Requirement` představuje uživatelský požadavek na software. Vystupuje v roli nadtypu v ISA hierarchii, kde podtypy jsou entity `SimpleRequirement`, `PackageRequirement` a `CustomRequirement`.

**Společné atributy:**

- `id` – unikátní identifikátor (číslo),
- `title` – název požadavku (text),
- `state` – stav životního cyklu (text).

Požadavek má vazbu na aktuální semestr (`Semester`), obraz operačního systému (`Image`) a předměty (`Course`). Každému požadavku je přiřazen autor a řešitel, v obou případech se jedná o entitu `User`. Každý požadavek obsahuje historii aktivit, reprezentovanou entitou `ActivityLog`.

#### 2.4.1.1 SimpleRequirement

Je-li požadavek zadaný výběrem z nabídky předem připraveného softwaru, jedná se o `SimpleRequirement`. Entita neobsahuje žádné specifické atributy.

#### 2.4.1.2 PackageRequirement

Požadavek vyhledaný pomocí názvu softwarového balíčku příslušné linuxové distribuce nabývá typu `PackageRequirement`. Entita rovněž neobsahuje žádné specifické atributy.

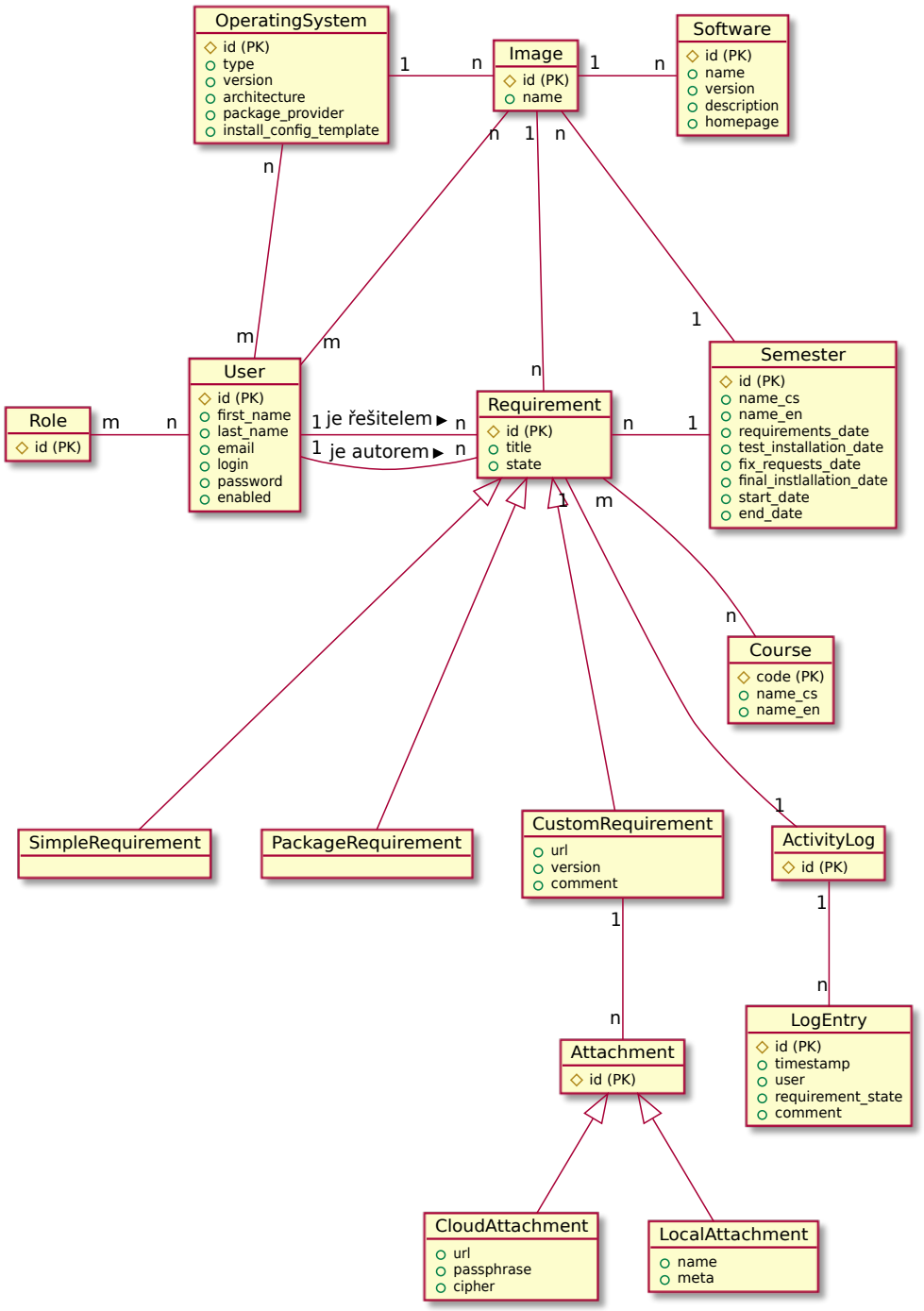
#### 2.4.1.3 CustomRequirement

`CustomRequirement` představuje požadavek zadaný třetím způsobem – prostřednictvím komplexního formuláře.

**Specifické atributy:**

- `url` – URL adresa požadovaného softwaru (text),
- `version` – verze softwaru (text),
- `comment` – komentář (text).

[21] `https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model`




Obrázek 2.15: Doménový model aplikace

### 2.4.2 Attachment

Požadavek typu `CustomRequirement` volitelně obsahuje jeden či více přiložených souborů (`Attachment`). Podobně jako požadavek, také přiložený soubor může být dvojího typu – `CloudAttachment`, nebo `LocalAttachment`.

**Společné atributy:**

- `id` – unikátní identifikátor (číslo).

#### 2.4.2.1 CloudAttachment

Podtyp `CloudAttachment` představuje soubor přiložený z externího úložiště. Typické využití je pro objemnější soubory, které není možné nahrát na server.

**Specifické atributy:**

- `url` – URL adresa souboru (text),
- `passphrase` – klíč pro dešifrování souboru (text),
- `cipher` – typ použité šifry (text).

#### 2.4.2.2 LocalAttachment

Naproti tomu `LocalAttachment`, jak už název napovídá, představuje soubor lokálně uložený na serveru. Velikost takového souboru nesmí překračovat maximální povolený limit.

**Specifické atributy:**

- `name` – název souboru (text),
- `meta` – doplňující metadata (text).

### 2.4.3 ActivityLog

Každý požadavek obsahuje historii aktivit – `ActivityLog`. Historie zahrnuje především záznamy o změnách stavu životního cyklu požadavku.

**Atributy:**

- `id` – unikátní identifikátor (číslo).

### 2.4.4 LogEntry

Jeden záznam v logu aktivit reprezentuje entita `LogEntry`. Každý log aktivit může mít libovolný počet záznamů `LogEntry`.

**Atributy:**

- `id` – unikátní identifikátor (číslo),
- `timestamp` – časové razítko aktivity (datum a čas),
- `user` – uživatelské jméno osoby, jež aktivitu provedla (text),
- `requirement_state` – stav požadavku (číslo),
- `comment` – komentář (text).

### 2.4.5 OperatingSystem

Operační systém je reprezentován entitou `OperatingSystem`. Mezi podporované operační systémy patří zejména Linux Gentoo, MS Windows a macOS. Každý operační systém má přiřazeného jednoho nebo více správců (`User`).

**Atributy:**

- `id` – unikátní identifikátor (číslo),
- `type` – typ operačního systému (text),
- `version` – verze operačního systému (text),
- `architecture` – podporovaná CPU architektura (text),
- `package_provider` – název služby zajišťující vyhledávání softwarových balíčků (text),
- `install_config_template` – šablona instalační konfigurace pro automatizační nástroj Ansible (text).

### 2.4.6 Image

Entita `Image` představuje obraz operačního systému. Každý obraz má vazbu na právě jeden operační systém (`OperatingSystem`) a semestr (`Semester`). Ke každému obrazu se vztahuje libovolný počet požadavků (`Requirement`). Podobně jako operační systém, také obraz má přiřazeného jednoho nebo více správců (`User`). Význam správce se v obou případech mírně liší a bude detailně popsán později.

**Atributy:**

- `id` – unikátní identifikátor (číslo),
- `name` – název obrazu (text).

### 2.4.7 Software

Entita `Software` představuje jednu položku v seznamu instalovaného softwaru, zobrazeného na obrazovce Instalovaný software. Každý software se vztahuje k právě jednomu obrazu OS.

**Atributy:**

- `id` – unikátní identifikátor (číslo),
- `name` – název softwaru (text),
- `version` – verze (text),
- `description` – popis (text),
- `homepage` – domovská stránka (text).

### 2.4.8 Course

Entita `Course` reprezentuje jeden vyučovaný předmět, pro který se požadavek zadává. Každý požadavek má přiřazen jeden nebo více předmětů.

**Atributy:**

- `id` – unikátní kód předmětu (text),
- `name_cs` – název v češtině (text),
- `name_en` – název v angličtině (text).

### 2.4.9 Semester

Výukový semestr je reprezentován entitou `Semester`. Vedle kódu a názvu obsahuje atributy související s harmonogramem přípravy učeben.

**Atributy:**

- `id` – unikátní kód semestru (text),
- `name_cs` – název v češtině (text),
- `name_en` – název v angličtině (text),
- `requirements_date` – poč. termín specifikace požadavků (datum),
- `test_installation_date` – poč. termín testovací instalace (datum),
- `fix_requests_date` – poč. termín přijímání reklamací (datum),
- `final_installation_date` – poč. termín finální instalace (datum),
- `start_date` – začátek semestru (datum),
- `end_date` – konec semestru (datum).

### 2.4.10 User

Entita `User` představuje uživatele v systému.

**Atributy:**

- `id` – unikátní identifikátor (číslo),
- `first_name` – rodné jméno (text),
- `last_name` – příjmení (text),
- `email` – e-mailová adresa (text),
- `login` – uživatelské jméno (text),
- `password` – heslo (text),
- `enabled` – příznak, zda je uživatel aktivní (true/false).

### 2.4.11 Role

Každému uživateli je přiřazena jedna nebo více uživatelských rolí (`Role`). Uživatelská role ovlivňuje to, jaký rozsah činností může konkrétní uživatel v aplikaci provádět. Mezi základní role patří uživatel (`ROLE_USER`) a administrátor (`ROLE_ADMIN`).

**Atributy:**

- `id` – unikátní název role (text).

Kapitola 3

# Implementace

Podle návrhu byla implementována webová aplikace pro sběr požadavků na instalaci softwaru. Na úvod je popsána architektura aplikace a použité technologie. Podstatná část kapitoly je zaměřena na koncepty řešení vybraných dílčích požadavků. Závěr kapitoly je věnován seznámení s implementací RESTful aplikačního rozhraní a procesem testování aplikace.

## 3.1 Architektura aplikace a použité technologie

Implementace byla realizována v jazyce PHP [4] verze 7 s využitím frameworku Symfony [5] (Standard Edition, verze 3.2), který je již několikátým rokem součástí výuky webových technologií na FIT ČVUT a zároveň prostředkem pro vývoj webových aplikací založených na PHP v oddělení ICT. Front-endová část aplikace je postavena na zavedených technologiích HTML5 a CSS3[22] a skriptovacím jazyce JavaScript s využitím knihovny JQuery [6]. Veškerý kód aplikace byl vyvíjen v PhpStorm IDE [7].

### 3.1.1 Symfony

Symfony je open source framework pro vývoj webových aplikací v jazyce PHP. Sestává ze sady samostatných, oddělených komponent[23], na jejichž vývoji se podílí široká komunita vývojářů a které jsou využity i v dalších nástrojích (Doctrine, Composer, Drupal, ...). Projekt je dále sponzorován francouzskou společností SensioLabs.

Framework klade důraz na princip Separation of Concerns[24] neboli rozdělení programu na různé části tak, aby se z hlediska funkcionality co možná nejméně překrývaly. Výsledný kód je lépe udržitelný, rozšiřitelný a znovupoužitelný.[8]

[22] `https://www.w3.org/standards/webdesign/htmlcss`
[23] `http://symfony.com/components`
[24] `http://effectivesoftwaredesign.com/2012/02/05/separation-of-concerns/`

Vývojáři webových aplikací v Symfony ocení Web Debug Toolbar[25] a Profiler[26], nástroje pro analýzu a zobrazení informací o aktuálním požadavku, komponentu Console[27] umožňující interagovat s aplikací pomocí příkazové řádky, logování a dobrou podporu pro jednotkové a funkční testování. V neposlední řadě je nutné zmínit rozsáhlou a kvalitní dokumentaci[28].

### 3.1.2 Architektura aplikace

Projekt sleduje doporučenou výchozí adresářovou strukturu[29].

```
/
├── app ..... konfigurace aplikace, globální šablony
├── bin ..... spustitelné soubory (např. bin/console)
├── src ..... zdrojový kód v PHP
├── tests ..... automatické testy (např. Unit testy)
├── var ..... generované soubory (cache, logy apod.)
├── vendor ..... zdrojové kódy knihoven třetích stran
└── web ..... kořenový adresář webu
```

#### 3.1.2.1 Bundles

Kód aplikace je členěn do logických celků, které jsou v Symfony nazývány „bundles“ (počeštěně „bundly“)[30]. Bundle je obyčejný adresář obsahující PHP třídy, konfigurační a další soubory, které dohromady zajišťují jednu konkrétní funkcionalitu. Koncept bundlů lze přirovnat k modulům v jiných systémech s tím rozdílem, že bundle není pouze komponenta rozšiřující základní funkčnost systému; v Symfony je vše složeno z bundlů – i samotné jádro frameworku.

Aplikace je dělena do tří bundlů:

- `BaseBundle` – doménová vrstva aplikace,
- `UiBundle` – prezentační vrstva, využívá `BaseBundle`,
- `ApiBundle` – RESTful API, využívá `BaseBundle`.

[25] `http://symfony.com/blog/new-in-symfony-2-8-redesigned-web-debug-toolbar`
[26] `http://symfony.com/blog/new-in-symfony-2-8-redesigned-profiler`
[27] `http://symfony.com/doc/current/components/console.html`
[28] `http://symfony.com/doc/current/index.html`
[29] `http://symfony.com/doc/current/quick_tour/the_architecture.html`
[30] `http://symfony.com/doc/current/bundles.html`

Každý bundle má svou vlastní adresářovou strukturu, kterou zachycuje následující adresářový strom:

```
/src/Cvut/Fit/Ict/SoftwareManager/
├── ApiBundle
│   └── Resources ..... konfigurace RESTful API
├── BaseBundle
│   ├── Api ..... třídy pro komunikaci s API třetích stran
│   ├── Command ..... příkazy symfony konzole
│   ├── DataFixtures ..... třídy pro vytvoření a nahrání testovacích dat
│   ├── DependencyInjection ..... třídy pro načtení konfigurace bundlu
│   ├── Entity ..... entitní (modelové) třídy
│   ├── EventListener ..... posluchači událostí (event listeners)
│   ├── Exception ..... vlastní výjimky
│   ├── Repository ..... repository třídy
│   ├── Resources ..... konfigurační soubory bundlu (definice služeb)
│   ├── Security ..... třídy zajišťující autentizaci a autorizaci
│   ├── Service ..... servisní třídy (business logika)
│   └── Utils ..... pomocné třídy
└── UiBundle
    ├── Controller ..... třídy controllerů
    ├── Form ..... formulářové třídy
    ├── Menu ..... třídy pro vytvoření navigace
    ├── Resources ..... definice služeb, překlady (i18n), šablony
    └── Twig ..... rozšíření do Twig šablon (funkce, filtry, ...)
```

#### 3.1.2.2 Entitní třídy

Entitní třídy vychází z doménového modelu (viz sekce 2.4) a jsou uloženy v adresáři `BaseBundle/Entity`. Typicky entitní třída reprezentuje tabulku v relační databázi, kde každý sloupec odpovídá jednomu atributu třídy a každý řádek jedné třídní instanci.

Konverzi dat mezi PHP objekty a relační databází zprostředkovává knihovna Doctrine [9], se kterou je framework Symfony integrován, prostřednictvím **objektově relačního mapování** (ORM). Pro zajištění funkčnosti mapování je nutné doplnit entity o jistá metadata pomocí DocBlock anotací (alternativně v YAML nebo XML).

Pro demonstrativní účely je uvedena entita `Image`, resp. její část s atributy `id`, `name` a `operatingSystem`. Anotace `@ORM\Entity` zajistí, že pro entitu bude vytvořena samostatná tabulka. Číselný atribut `id` bude sloužit jako primární klíč a jeho hodnota bude automaticky generovaná. Další sloupec tabulky typu *string* bude vyhrazen pro název obrazu; cílové datové typy jsou závislé na konkrétním zvoleném typu databáze. Vazba na operační systém je vyjádřena

anotací `@ORM\ManyToOne`, tím dojde k vytvoření cizího klíče odkazujícího do tabulky operačních systémů.

```
/**
 * @ORM\Entity
 */
class Image
{
    /**
     * @var int
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * The name of an image.
     *
     * @var string
     * @ORM\Column(type="string")
     */
    protected $name;

    /**
     * Related OS.
     *
     * @var OperatingSystem
     * @ORM\ManyToOne(
     *     targetEntity="OperatingSystem",
     *     inversedBy="images"
     * )
     */
    protected $operatingSystem;

    ...
}
```

Návrh aplikace počítá s dědičností entit (např. `Attachment` s potomky `CloudAttachment` a `LocalAttachment`), kterou relační databáze nepodporují. Doctrine nabízí několik strategií, jak třídy v hierarchii mapovat do databáze. Z možných řešení byla vybrána strategie s názvem **Single table inheritance**, kdy všechny třídy jsou mapovány do jedné společné tabulky. Pro rozlišení typu potomka slouží uměle vytvořený sloupec diskriminátor. Předností této metody je rychlost, neboť se zde neprovádějí žádné JOIN operace nad tabulkami.[10]

#### 3.1.2.3 Repository třídy

Ke každé entitní třídě existuje odpovídající repository třída, uložená v adresáři `BaseBundle/Repository`. Repository třídy zajišťují provádění tzv. CRUD (create, read, update, delete) operací nad entitami včetně pokročilejšího dotazování. CRUD operace zapouzdřuje trait `CRUDTrait`, který je využit v každé repository třídě.[11]

#### 3.1.2.4 Servisní třídy

Servisní třídy implementují business logiku aplikace a nachází se v adresáři `BaseBundle/Service`. Typicky provádějí autorizaci operací (více v sekci 3.2.7) a komunikují s repository třídami.

Veřejné metody každé servisní třídy jsou deklarovány v odpovídajícím rozhraní (např. `ImageService` implementuje rozhraní `ImageServiceInterface`). Servisní třídy jsou zaregistrovány jako služby[31] do DI Containeru, tím je lze snadno využívat například v controllerech (viz dále).

#### 3.1.2.5 Formulářové třídy

Formuláře, se kterými aplikace pracuje, je možné v principu rozdělit do dvou kategorií: formuláře pro **vytváření a editaci** entit a formuláře pro **filtrování a vyhledávání** entit.

Formuláře v první skupině jsou vždy vázané na entity – formulářová pole reflektují jejich atributy. Druhá skupina formulářů pracuje s pomocnými objekty pro filtrování (`RequirementFilter`, `SoftwareFilter`); řešení je inspirováno článkem „Filtering data by user input with Kdyby/Doctrine“ [12].

V souladu se Symfony best practices[32] je vytváření formulářů vyčleněno do speciálních formulářových tříd, které leží v adresáři `UiBundle/Form/Type`.

#### 3.1.2.6 Controllery

Controller je PHP funkce, jejíž úkolem je:

1. přijmout HTTP požadavek,
2. zvalidovat požadavek (volitelně),
3. učinit autorizační rozhodnutí (volitelně),
4. delegovat úkol na příslušnou servisní třídu (business logika),
5. vrátit HTTP odpověď (HTML stránka, JSON, ...).

[31] `http://symfony.com/doc/current/service_container.html#what-is-a-service`
[32] `http://symfony.com/doc/current/best_practices/index.html`

Třídy controllerů, ležící v adresáři `UiBundle/Controller`, jsou rozděleny podle entit, se kterými pracují (`OperatingSystemController`, `RequirementController` apod.). Každá třída typicky obsahuje následující metody (controllery):

- `indexAction()` – zobrazení kolekce entit,
- `showAction()` – zobrazení detailu entity,
- `createAction()` – vytvoření nové entity,
- `updateAction()` – editace entity,
- `deleteAction()` – odstranění entity.

#### 3.1.2.7 Šablony

Generování těla odpovědi controller deleguje na šablonovací systém. Symfony používá šablonovací systém Twig [13], který svou přívětivou syntaxí usnadňuje tvorbu šablon, jež následně kompiluje do optimalizovaného PHP.

Šablona je textový soubor s předpisem pro generování libovolného textového obsahu (HTML, XML, CSV, LaTeX, ...). Twig podporuje dědičnost šablon, vytváření vlastních maker a obsahuje řadu užitečných funkcí a filtrů.[14]

#### 3.1.2.8 Konfigurace

Konfigurace aplikace se nachází v adresáři `app/config/` a je zapsána ve formátu YAML[33]. Tento formát vyniká především dobrou čitelností a v Symfony je výchozí volbou; alternativně lze použít také XML nebo PHP.

Adresář obsahuje řadu konfiguračních souborů, tím hlavním je `config.yml`, obsahující všechny parametry, které souvisí s chováním aplikace (konfigurace jednotlivých bundlů). Parametry závislé na infrastruktuře, jako konfigurace databázového a mail serveru, jsou vyčleněny do souboru `parameters.yml`.

Další konfigurační soubory:

- `routing.yml` – routování (mapování URL k příslušnému controlleru),
- `security.yml` – zabezpečení aplikace (autentizace, autorizace),
- `services.yml` – definice služeb pro Dependency Injection Container[34].

Konfigurační údaje jsou definovány odděleně pro každé prostředí běhu aplikace (development, production, testing). Například konfigurace routování pro vývojové prostředí se nachází v souboru `routing_dev.yml`, který je importován souborem `config_dev.yml`.

[33] `http://yaml.org/`

[34] `http://symfony.com/doc/current/service_container.html`

## 3.2 Koncepty řešení dílčích požadavků

Sekce je zaměřena na popis řešení vybraných dílčích požadavků (funkčních i nefunkčních).

### 3.2.1 Workflow požadavku

Každý požadavek na software prochází životním cyklem (viz sekce 2.1.2). Jelikož počet definovaných stavů životního cyklu je konečný a požadavek se nachází vždy pouze v jednom stavu, lze toto chování modelovat pomocí **konečného automatu**.

#### 3.2.1.1 Výběr knihovny

V současnosti existuje celá řada knihoven implementujících konečný automat, proto je nasnadě některou z nich využít. Jednou z možností je knihovna, resp. bundle LexikWorkflowBundle[35], který je integrován v řadě novějších aplikací na frameworku Symfony provozovaných oddělením ICT FIT ČVUT. Vývoj této knihovny však v poslední době stagnuje[36], což je pádným důvodem k prozkoumání alternativ.

Framework Symfony s verzí 3.2 představil novou **komponentu Workflow**[37], která je součástí jeho jádra. Komponenta, navzdory poměrně stručné dokumentaci, implementuje všechny potřebné mechanismy konečného automatu, a proto se stala definitivní volbou.

Oficiální dokumentaci komponenty Workflow doplňuje velmi zdařilý tutoriál na serveru CodeReviewVideos.com [15].

#### 3.2.1.2 Definice workflow

Definice workflow je zapsána ve formátu YAML a pro zvýšení přehlednosti vyčleněna do speciálního souboru `app/config/workflows.yml`.

```
# app/config/workflows.yml

framework:
  workflows:
    requirement:
      type: 'state_machine'
      marking_store:
        type: 'single_state'
        arguments:
          - 'state'
```

[35] `https://github.com/lexik/LexikWorkflowBundle`

[36] Knihovna LexikWorkflowBundle byla k 31. 3. 2017 označena jako deprecated.

[37] `http://symfony.com/doc/current/components/workflow.html`

```
supports:
  - Cvut\Fit\Ict\SoftwareManager\BaseBundle\Entity\
      Requirement
initial_place: created
places:
  - created
  - needs_review
  - approved_by_admin
  - rejected_by_admin
  - installed_by_admin
  - returned_by_user
  - verified_by_user
  - deleted
transitions:
  to_review:
    from: [created, rejected_by_admin]
    to: needs_review
  approve:
    from: needs_review
    to: approved_by_admin
  reject:
    from: needs_review
    to: rejected_by_admin
  install:
    from: [approved_by_admin, returned_by_user]
    to: installed_by_admin
  verify:
    from: installed_by_admin
    to: verified_by_user
  return:
    from: installed_by_admin
    to: returned_by_user
  delete:
    from: [created, rejected_by_admin, returned_by_user]
    to: deleted
```

Soubor obsahuje definici jediného workflow s identifikátorem `requirement`. Workflow podporuje entitu `Requirement` a aktuální stav bude uložen v atributu `state`. Z definice lze dále pod klíčem `initial_place` vyčíst počáteční stav, množinu všech stavů (`places`) a přechodů (`transitions`).

#### 3.2.1.3 Řízení workflow

S takto zapsanou definicí lze načíst a používat službu s názvem `state_machine.requirement`, pomocí níž je možné zjistit, které akce/přechody jsou pro

daný objekt a stav povoleny (metody `can($requirement, 'transition')` a `getEnabledTransitions($requirement)`), a případně přesunout objekt do dalšího stavu (metoda `apply($requirement, 'transition')`).

Akce změny stavu softwarového požadavku (např. předání požadavku ke schválení) obsluhuje controller `advanceWorkflowAction()` ve třídě `RequirementController`. Akce jsou delegovány na servisní třídu `RequirementService` s metodou `advanceWorkflow()`, která vnitřně volá výše zmíněnou metodu `apply(...)`.

Autorizační rozhodnutí pro jednotlivé akce/přechody, tj. zda je uživatel oprávněn provést danou akci, provádí třída `RequirementWorkflowVoter` ve spolupráci s třídou `RequirementWorkflowListener`, která odebírá a obsluhuje události workflow.[16]

### 3.2.2 Komentování požadavku

V kontextu funkce komentování požadavku byly uvažovány dva možné způsoby realizace. Prvním z nich je vyčlenění komentářů do samostatného vlákna, druhým pak zahrnutí komentářů do logu aktivit požadavku (v tabulce přibude sloupec *komentář*).

**Samostatné vlákno** Výhodou komentářů v samostatném vlákně je vyšší přehlednost a též rozšířenost tohoto řešení. Nevýhodou je oddělení informací o změně stavu požadavku od samotných komentářů (není patrné v jakém stavu se požadavek v době přidání komentáře nacházel).

**Součást logu aktivit** Předností druhé varianty je na první pohled patrná souvislost mezi stavem požadavku a komentářem. Komentář lze navázat přímo na akci změny stavu požadavku (schválení, reklamace, ...). Další výhodou je nižší implementační náročnost. Nevýhodou pak může být menší přehlednost, zejména na mobilních zařízeních s nižším rozlišením. Především s ohledem na implementační náročnost byl pro první fázi implementace zvolen právě tento způsob.

### 3.2.3 Přílohy požadavku

Přikládání souborů se provádí ve druhém kroku po vytvoření požadavku. Soubory lze přiložit pouze k požadavku vytvořenému vlastním zadáním (entita `CustomRequirement`), a to jedním z následujících způsobů:

1. poskytnout odkaz pro stažení souboru z online úložiště,

2. nahrát soubor na server; velikost takového souboru je značně omezena.

#### 3.2.3.1 Využití online úložiště

Při využití online úložiště je vyžadována URL adresa souboru, volitelně lze zadat heslo/klíč a typ použité šifry. Klíč a typ šifry je zobrazen pouze autorovi požadavku a správci příslušného obrazu.

Seznam povolených šifer je předmětem konfigurace:

```
# app/config/parameters.yml

software_manager.attachment.ciphers: [aes-256, aes-256-cbc, ...]
```

Pro zlepšení uživatelského prožitku lze klíč přiloženého souboru jedním kliknutím na patřičné tlačítko zkopírovat do schránky. Tuto funkci zajišťuje JavaScriptová knihovna clipboard.js [17].

#### 3.2.3.2 Upload na server

Při nahrání na server je nutné vybrat cestu k souboru, volitelně lze připojit doplňující metadata (množina dvojic *klíč: hodnota*).

Implementace využívá dva existující bundly: VichUploaderBundle [18] pro usnadnění uploadu souborů a OneupFlysystemBundle [19], který integruje do Symfony projektu Flysystem – knihovnu pro abstrakci souborového systému (umožňuje snadno vyměnit lokální souborový systém za vzdálený).

Soubory jsou nahrávány do adresáře `var/upload/attachments/` (cesta je konfigurovatelná), dále členěného do podadresářů pojmenovaných podle semestrů (např. `B162`). Názvy podadresářů jsou generovány třídou `Attachment-Service/DirectoryNamer`. Soubory jsou ukládány pod vlastními jmény s náhodně generovaným prefixem (funkcí `uniqid()`). Díky tomu lze nahrát na server více souborů se stejným jménem.

### 3.2.4 Harmonogram semestru

Příprava učeben se řídí harmonogramem semestru. Vytvoření harmonogramu znamená nastavení počátečních dat jednotlivých fází (viz sekce 2.3.9, obrazovka Harmonogram semestru).

Kliknutím na formulářové pole se zobrazí kalendář pro výběr data, alternativně lze datum zadat ručně ve formátu *RRRR-MM-DD*. Jako kalendářová komponenta pro výběr data byla zvolena JavaScriptová knihovna Pikaday [20] spolu s knihovnou Moment.js [21] zajišťující potřebné formátování data.

Pro urychlení nastavení harmonogramu lze data předem automaticky vygenerovat – dle data zahájení semestru a časových intervalů jednotlivých fází specifikovaných v konfiguraci aplikace.

Konfigurační parametry jsou následující (fáze harmonogramu: délka trvání ve dnech):

```
# app/config/parameters.yml

software_manager.schedule.requirements: 28
software_manager.schedule.test_installation: 14
software_manager.schedule.fix_requests: 14
software_manager.schedule.final_installation: 7
```

### 3.2.5 Import z KOS

Aplikace využívá vybrané zdroje z databáze systému KOS[38], jmenovitě předměty (zdroj *Course*) a semestry (zdroj *Semester*). Data jsou získávána prostřednictvím KOSapi[39] a ukládána do lokální databáze.

#### 3.2.5.1 Zabezpečení KOSapi

KOSapi je zabezpečeno protokolem OAuth 2.0[40] a využívá fakultní autorizační server. Pro jeho využití bylo v první řadě nutné provést následující kroky:

1. přihlásit se do aplikace AppsManager[41],
2. vytvořit novou aplikaci *Software Manager* typu *Service Account* – aplikace se autentizuje sama za sebe (grant *client credentials*),
3. aktivovat službu čtení dat z KOS (scope *cvut:kosapi:read*).

#### 3.2.5.2 Implementace

Pro import dat byla vytvořena sada příkazů, které je možné (ručně nebo automatizovaně) spouštět prostřednictvím Symfony Console:

```
$ php bin/console software-manager:kos:courses:import <code>
$ php bin/console software-manager:kos:courses:sync
$ php bin/console software-manager:kos:semesters:import <code>
$ php bin/console software-manager:kos:semesters:sync
```

Každému z výše definovaných příkazů odpovídá právě jedna PHP třída uložená v adresáři `BaseBundle/Command/KOS/`:

- `CoursesImportCommand` – import libovolného předmětu dle jeho kódu,

[38] `https://kos.cvut.cz`
[39] `https://kosapi.fit.cvut.cz/`
[40] `https://rozvoj.fit.cvut.cz/Main/oauth2`
[41] `https://auth.fit.cvut.cz/manager/`

- `CoursesSyncCommand` – synchronizace předmětů s KOS; pro účely aplikace dochází k importu všech FIT předmětů pro rozvrhovaný semestr,
- `SemestersImportCommand` – import libovolného semestru dle kódu,
- `SemestersSyncCommand` – synchronizace semestrů s KOS ~ import aktuálně rozvrhovaného semestru.

Všechny třídy využívají servisní třídu `KosService` implementující rozhraní `KosServiceInterface` s metodami `importCourse(code)`, `syncCourses()`, `importSemester(code)` a `syncSemesters()`. Tato třída zajišťuje získání dat z KOSapi a jejich uložení do lokální databáze (pokud již neexistují).

**Získání dat** Pro vykonávání samotných dotazů na KOSapi byl využit PHP HTTP klient Guzzle [22] a knihovna Sainsbury's guzzle-oauth2-plugin [23] poskytující OAuth2 middleware pro Guzzle.

**Konverze** Zdroje získané z KOSapi jsou ve formátu Atom (XML)[42]. Konverzi XML obsahu do PHP objektů (instancí třídy `Course`, resp. `Semester`) zajišťuje třída `XMLParser`, která dále využívá Symfony komponentu DomCrawler[43].

**Persistence** Úspěšně konvertované objekty jsou třídou `KosService` předány servisní třídě `CourseService`, resp. `SemesterService`, která prostřednictvím repository třídy `CourseRepository`, resp. `SemesterRepository` uloží objekty do databáze.

Komunikaci všech výše zmíněných tříd (resp. její podstatnou část) na příkladu importu předmětu zachycuje sekvenční diagram na obrázku 3.1.

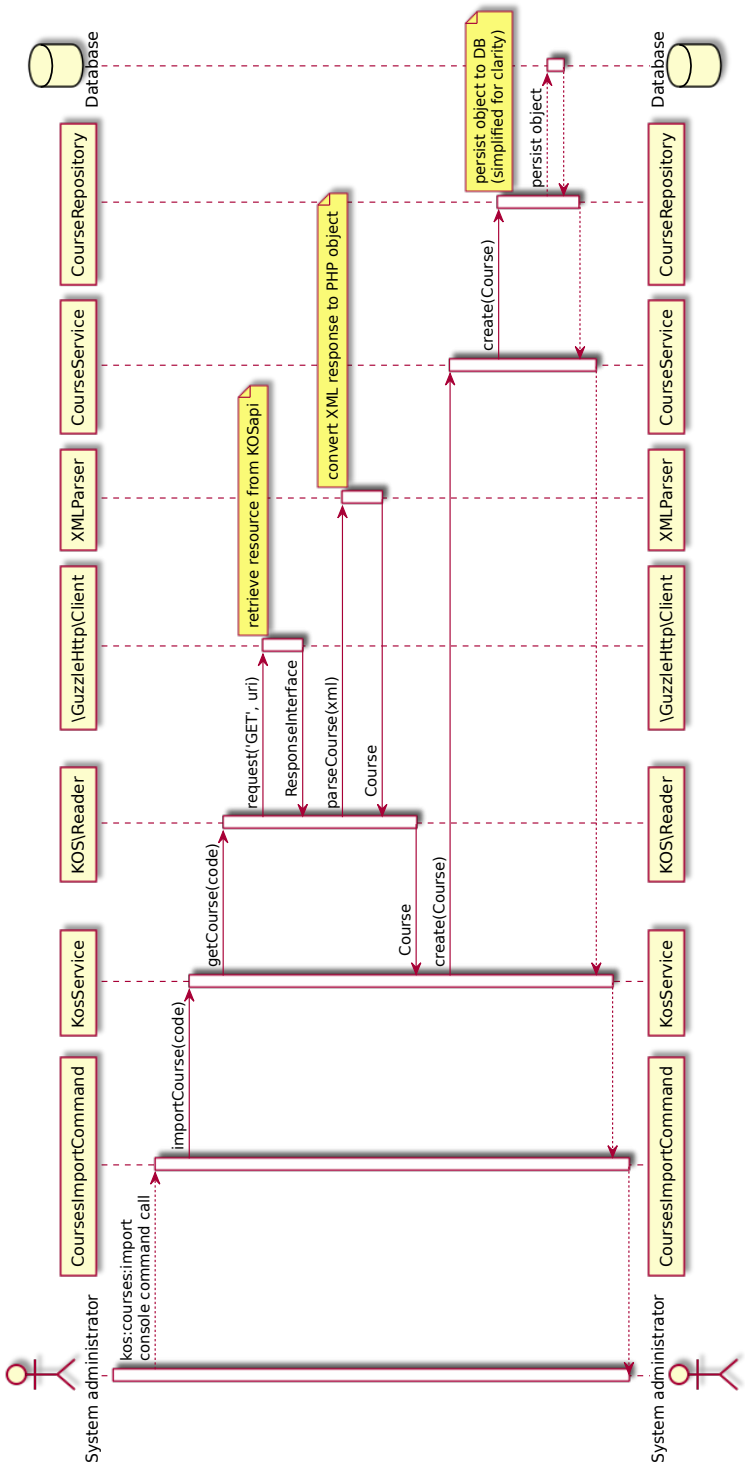
#### 3.2.5.3 Konfigurace

Konfigurační parametry pro komunikaci s KOSapi jsou následující (z formátovacích důvodů vynechán prefix `software_manager.kos.`):

```
# app/config/parameters.yml

api_endpoint:   'https://kosapi.fit.cvut.cz/api/3/'
token_endpoint: 'https://auth.fit.cvut.cz/oauth/oauth/token'
client_id:      client_id        # ziskane z AppsManager
client_secret:  client_secret    # ziskane z AppsManager
```

[42] `https://www.rfc-editor.org/info/rfc4287`
[43] `http://symfony.com/doc/current/components/dom_crawler.html`




Obrázek 3.1: Import předmětu z KOS (sekvenční diagram)

### 3.2.6 Vyhledávání softwarových balíčků

Pro vybrané operační systémy lze nový požadavek zadat vyhledáním softwarového balíčku. Aplikace je v základu integrována se dvěma službami přehledu softwarových balíčků:

- `https://packages.debian.org/` (Debian GNU/Linux),
- `https://packages.gentoo.org/` (Gentoo GNU/Linux).

#### 3.2.6.1 Back-endová část

Pro každou z integrovaných služeb existuje servisní třída implementující rozhraní `PackageProviderServiceInterface` s metodami `getName()` (vrací název vyhledávače balíčků) a `retrieve()` (vrací kolekci balíčků v jednotném formátu, viz dále).

Metoda `retrieve()` obsahuje parametry:

- `$term` – povinný, hledaný výraz (název balíčku nebo jeho část),
- `$osVersion` – verze operačního systému,
- `$osArchitecture` – architektura operačního systému,
- `$limit` – maximální počet vrácených balíčků.

Vrácená kolekce balíčků musí mít následující formát (atributy `id` a `text` jsou povinné, atribut `description` je volitelný):

```
[
  {
    id: 'value',
    text: 'text to display',
    description: 'description'
  },
  ...
]
```

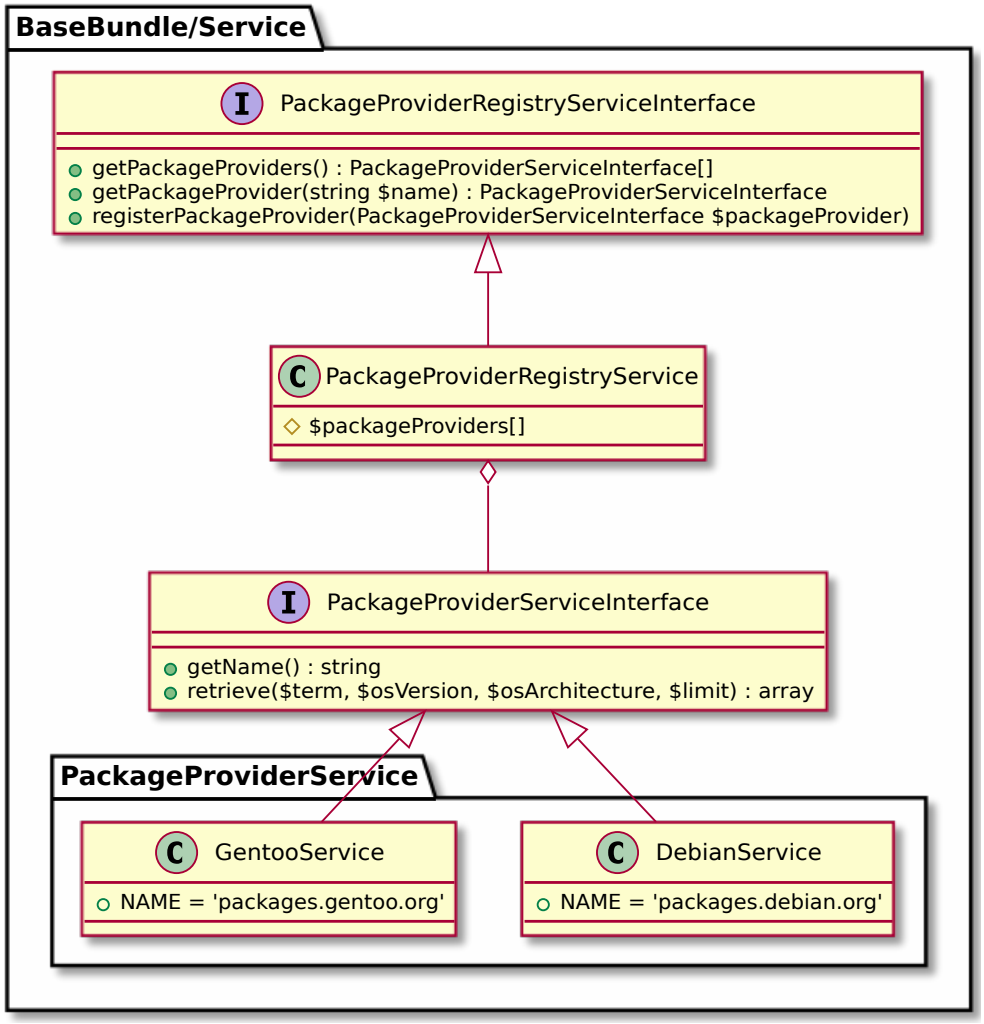
**Registr vyhledávačů** Všechny definované servisní třídy pro vyhledávání balíčků centralizuje služba `PackageProviderRegistryService` implementující rozhraní `PackageProviderRegistryServiceInterface` s metodami:

- `getPackageProviders()` – vrátí všechny registrované vyhledávače,
- `getPackageProvider(string $name)` – vrátí vyhledávač podle názvu,

- `registerPackageProvider(PackageProviderServiceInterface $packageProvider)` – zaregistruje nový vyhledávač.

Servisní třídy vyhledávačů jsou zaregistrovány jako služby do DI Containeru, s tagem `cvut_fit_ict_software_manager_base.service.package_provider_service`. Do registru vyhledávačů jsou zaregistrovány v době sestavení Containeru.[24]

Diagram na obrázku 3.2 představuje statický pohled na vazby mezi výše popsanými třídami a rozhraními.




Obrázek 3.2: Vyhledávání softwarových balíčků (diagram tříd)

**Packages.debian.org** Vyhledávání balíčků pro Debian GNU/Linux zajišťuje třída `DebianService`. Data jsou získávána ve formátu JSON prostřednictvím API, které server poskytuje[44]. Pro vykonávání samotných dotazů je využit opět HTTP klient Guzzle. Vyhledávání pro Debian je omezeno na konkrétní verzi, architektura se zde nebere v úvahu.

Příkladem může být dotaz na URL adresu `https://sources.debian.net/api/search/php/?suite=stretch`, který vrátí všechny balíčky pro verzi *Stretch*, jejichž název obsahuje výraz „php“.

**Packages.gentoo.org** Vyhledávání balíčků pro Gentoo GNU/Linux zajišťuje třída `GentooService`. Ačkoli server nenabízí rozsáhlé možnosti dotazování nad balíčky, poskytuje strojově zpracovatelná data ve formátu JSON. Vyhledávány jsou pouze aktuální stabilní verze balíčků pro danou architekturu, verze je v tomto případě irelevantní.

Základní informace o balíčcích, jejichž název obsahuje výraz „php“, jsou v tomto případě vystaveny na URL adrese `https://packages.gentoo.org/packages/suggest.json?q=php`. Pro omezení vyhledávání na konkrétní architekturu je nutné se dotázat na detail každého balíčku – např. informace o balíčku „dev-lang/php“ se nachází na adrese `https://packages.gentoo.org/packages/dev-lang/php.json` – a na základě získaných informací z výsledku odfiltrovat nežádoucí balíčky.

#### 3.2.6.2 Front-endová část

Formulářové pole pro vyhledávání balíčků je integrované s jQuery pluginem Select2 [25], který podporuje načítání dat pomocí AJAX volání[45].

Vyhledávání balíčků probíhá následujícím způsobem:

1. Uživatel zadá do vyhledávacího pole název balíčku nebo jeho část.
2. Select2 odešle asynchronní HTTP POST dotaz na controller `retrieveAction()` třídy `PackageProviderController` (formát dat viz dále).
3. `PackageProviderController` deleguje dotaz na příslušnou servisní třídu (`DebianService`, nebo `GentooService`).
4. Servisní třída získá data z datového zdroje a předá je controlleru.
5. Controller vrátí odpověď s daty v JSON formátu (`JsonResponse`).
6. Select2 zparsuje JSON data a aktualizuje nabídku balíčků.

[44] `https://sources.debian.net/doc/api/`
[45] `https://select2.github.io/examples.html#data-ajax`

Data, která jsou součástí dotazu ve 2. kroku, mají následující formát:

```
provider: packages.gentoo.org
term: php
osVersion: -
osArchitecture: amd64
limit: 5
```

Hodnoty `provider`, `osVersion` a `osArchitecture` jsou získány ze speciálních atributů `data-package-provider`, `data-os-version` a `data-os-architecture` předem uložených k HTML elementu formulářového pole. Data jsou přenášena jako typ *application/x-www-form-urlencoded*.

#### 3.2.6.3 Přidání nového vyhledávače

Pro přidání nového vyhledávače softwarových balíčků je nutné provést následující kroky:

1. Vytvořit novou servisní třídu implementující rozhraní `PackageProviderServiceInterface`.
2. V metodě `retrieve()` implementovat logiku pro získání balíčků z datového zdroje a jejich konverzi do požadovaného výstupního formátu.
3. Zaregistrovat servisní třídu jako službu s tagem `cvut_fit_ict_software_manager_base.service.package_provider_service`.

Nově přidaný vyhledávač je nyní možné vybrat z nabídky „Vyhledávač softwarových balíků“ na obrazovce Nový operační systém (viz obr. 2.11). Název vyhledávače v nabídce odpovídá návratové hodnotě funkce `getName()`.

### 3.2.7 Autentizace uživatelů a řízení přístupu

Správa identit a rolí uživatelů je zajišťována fakultním systémem IdM (Identity Management). Aplikace obsahuje lokální databázi uživatelů (s omezenou sadou atributů), která je plněna ze systému IdM.

#### 3.2.7.1 Autentizace uživatelů

Autentizace je řešena zvlášť pro produkční a vývojové prostředí. V **produkčním** prostředí je zajištěna přes SSO Shibboleth[46]. Přihlašování neprobíhá přes vstupní formulář aplikace, ale na společném SSO serveru zvaném Identity Provider (IdP); aplikace s autentizačními údaji uživatelů vůbec nepracuje. Integraci s SSO zajišťuje třída `ShibbolethAuthenticator`.

[46] `https://ict.fit.cvut.cz/~web/current/web/identity/shibboleth-sp/`

Pro účely **vývoje a testování** je zprovozněna autentizace pomocí HTTP Basic[47]. Uživatel přistupující k aplikaci – ať už přes webové rozhraní, nebo k vystavenému API (viz sekce 3.3) – se prokáže svým uživatelským jménem a heslem.

Autentizace pomocí HTTP Basic je využívána rovněž v produkčním prostředí, a to výhradně pro zabezpečení vystaveného API. Přístup je povolen pouze prostřednictvím servisního účtu; jehož přihlašovací jméno a heslo je předmětem konfigurace (viz dále). Běžný uživatel k API nemá přístup, s jeho využitím se počítá v navazujících službách oddělení ICT.

#### 3.2.7.2 Řízení přístupu

Autorizace uživatelů probíhá na základě:

- rolí poskytovaných systémem IdM (`ROLE_USER` a `ROLE_ADMIN`),
- pseudo-rolí definovaných frameworkem Symfony (`IS_AUTHENTICATED_ANONYMOUSLY`, `IS_AUTHENTICATED_FULLY`, ...),
- vlastní logiky, kterou pokrývají následující třídy:
  - `RequirementVoter` – autorizace pro požadavky,
  - `RequirementWorkflowVoter` – autorizace akcí workflow požadavku,
  - `AttachmentVoter` – autorizace pro přílohy požadavku,
  - `ImageVoter` – autorizace pro obrazy OS.

Symfony Voters (třídy se sufixem `Voter`) jsou použity v případech, kdy není možné učinit autorizační rozhodnutí jednoduše na základě rolí uživatele. Každá ze tříd se váže k jednomu zdroji (entitě), např. `Requirement`, a na základě jednoduchých podmínek rozhoduje o oprávnění provádět operace s tímto zdrojem. (Např. uživatel má právo editovat požadavek, pouze je-li jeho autorem.) Více informací o Symfony Voters v oficiální dokumentaci [26].

Konkrétní definice oprávnění, tj. jaká data jsou dostupná / jaké operace jsou povolené kterým uživatelům v závislosti na autorizaci uživatele, je uvedena v příloze E.

#### 3.2.7.3 Konfigurace

Konfigurace zabezpečení aplikace se nachází v souborech:

- `app/config/security.yml` – produkční prostředí,
- `app/config/security_dev.yml` – vývojové prostředí.

[47] `https://cs.wikipedia.org/wiki/Basic_access_authentication`

Každý ze souborů obsahuje konfiguraci možných způsobů autentizace, načtení objektu aktuálního uživatele, metody pro šifrování uživatelských hesel a další bezpečnostní nastavení.

Konfigurace přihlašovacích údajů servisního uživatele pro přístup k API:

```
# app/config/parameters.yml

software_manager.api.user: api_user
software_manager.api.password: api_password
```

### 3.2.8 Jazyková lokalizace

Aplikace obsahuje kompletní českou a anglickou lokalizaci, další jazykové mutace je možné doplnit.

Symfony poskytuje dobrou podporu pro internacionalizaci (často zkráceně i18n). Kód aplikace je od veškerých textových výstupů odstíněn, používají se zástupné klíče – např. místo zprávy „Předmět byl úspěšně vytvořen“ je použit klíč `course.create.msg.success`.

Překlady všech použitých klíčů se nacházejí v následujících souborech:

```
UiBundle/Resources/translations/
├── messages.(cs|en).yml ..... hlavní soubor s překlady
├── forms.(cs|en).yml ..... překlady formulářů
├── navigation.(cs|en).yml ..... překlady navigace
└── validators.(cs|en).yml ..... chybové validační zprávy
```

Ukázka klíčů s překlady:

```
# UiBundle/Resources/translations/messages.en.yml

course:
  index:
    new: New course
  create:
    header: Create manually
    msg:
      success: Course has been successfully created.
      entry_already_exists: Course '%code%' already exists.
```

Ačkoli doporučeným formátem pro ukládání překladů je XLIFF (založený na XML), aplikace používá formát YAML, a to především z důvodu lepší čitelnosti. YAML podporuje vnořené identifikátory (viz ukázka), ty jsou následně zploštěny, přičemž původní úrovně jsou odděleny tečkami.

Aplikace obsahuje speciální menu pro změnu jazykové preference. Jeho sestavení zajišťuje třída `LangMenu`. Informace o volbě jazyka je obsažena v URL (např. */en/requirements*).

### 3.2.9 Vzhled aplikace

Pro uživatelské rozhraní aplikace byl využit CSS framework Milligram [27]. Mezi jeho přednosti patří:

- minimalistický vzhled,
- responzivita (správné zobrazení stránek na různých typech zařízení),
- snadná rozšiřitelnost,
- minimální velikost (jednotky kB),
- aktivní vývoj projektu.

#### 3.2.9.1 Vlastní pravidla stylů

Kaskádové styly definované frameworkem Milligram jsou doplněny o vlastní pravidla (layout stránky, barevné schéma, přepisy pravidel knihoven třetích stran, ...), která jsou zapsána do souboru `web/assets/css/style.css`.

#### 3.2.9.2 Navigace

Aplikace obsahuje hlavní navigaci pro přechod mezi obrazovkami definovanými v sekci 2.3, sekundární navigaci (přihlášení a odhlášení uživatele) a navigaci pro volbu jazyka.

Standardně jsou tyto navigace zobrazeny horizontálně při horním okraji stránky. Na mobilních zařízeních s nižším rozlišením obrazovek toto řešení není příliš vhodné, proto jsou zde všechny tři typy navigací spojeny do jednoho vertikálního menu, které je skryté pod ikonou tří vodorovných pruhů (tzv. hamburger menu). Sestavení mobilního menu zajišťuje knihovna SlickNav [28].

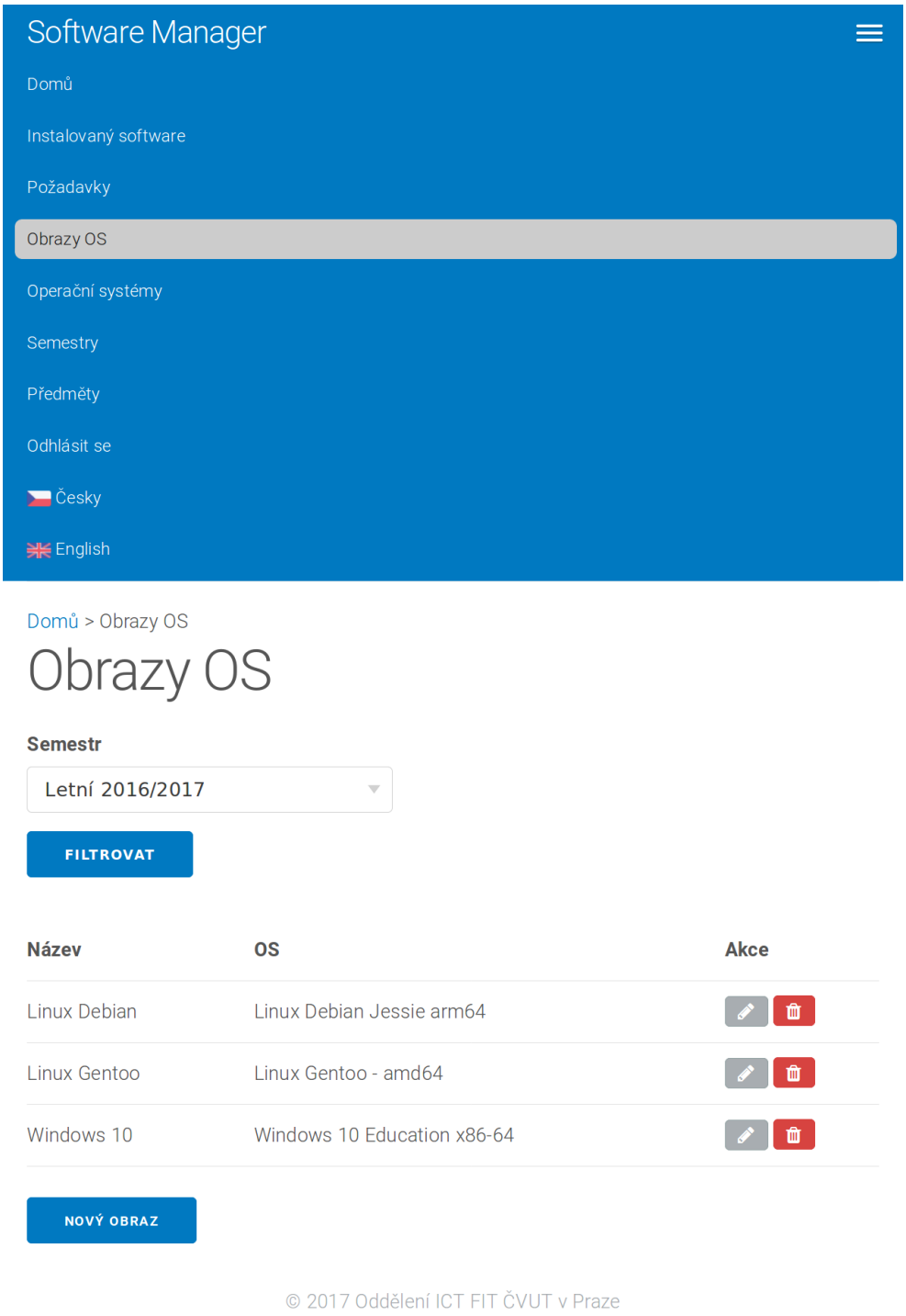
Aplikace dále disponuje drobečkovou navigací a navigační lištou stránkování, které je využito na obrazovkách Instalovaný software a Požadavky.

#### 3.2.9.3 Ikony a favicon

Aplikace využívá sadu ikon (navigace, editace a odstranění záznamů, kopírování do schránky, kalendář, ...) převzatých z knihovny Font Awesome [29].

Výchozí ikona webu favicon[48] (`web/favicon.ico`) s logem frameworku Symfony byla nahrazena ikonou převzatou ze serveru iconsDB.com [30].

[48] `https://en.wikipedia.org/wiki/Favicon`




Obrázek 3.3: Ukázka aplikace – obrazovka Obrazy OS, rozbalené menu

#### 3.2.9.4 Správa závislostí

Správu front-end závislostí (stahování, instalace, vyhodnocování vzájemných závislostí) obstarává Bower [31]. Závislosti jsou specifikovány ve verzovaném souboru `bower.json` a po provedení příkazu `bower install` staženy do adresáře `web/assets/vendor/`.

Pro zrychlení načítání aplikace se provádí minimalizace a zkombinování jednotlivých JavaScript a CSS souborů pomocí filtrů nástroje Assetic [32].

Obrázek 3.4: Favicon

## 3.3 RESTful API

Aplikace poskytuje aplikační rozhraní (API) v podobě RESTful webových služeb. REST služby jsou orientované na „zdroje“, každý z dostupných zdrojů má svou unikátní URL adresu; základní URL aplikačního rozhraní je `/api`. Komunikace s API probíhá prostřednictvím protokolu HTTP.

### 3.3.1 Realizace

Pro realizaci byl využit PHP framework API Platform [33], který je postaven na frameworku Symfony; jeho integrace do jiného Symfony projektu je tak velmi snadná.

Konfigurace pro API Platform se nachází v hlavním konfiguračním souboru `app/config/config.yml`. Převážná část realizace sestává pouze ze dvou dalších konfiguračních souborů, které jsou vyčleněny do speciálního bundle pro aplikační rozhraní – `ApiBundle`. Prvním ze souborů je `/Resources/config/api_resources/resources.yml` obsahující definici mapování entit na zdroje.

Pro serializaci objektů do požadovaného formátu[49] API Platform používá Symfony komponentu Serializer[50]. V souboru `/Resources/config/serialization.yml` se nachází pravidla pro serializaci, resp. definice serializačních skupin.

API Platform poskytuje popis rozhraní ve Swagger formátu[51], ze kterého je pomocí Swagger UI[52] generována dokumentace. Ta se nachází na adrese `/api/docs` a zahrnuje kompletní seznam zdrojů, podporovaných formátů a návratových kódů odpovědí a také sandbox pro testování dotazů.

[49] Seznam podporovaných formátů se nalézá v sekci 3.3.3.

[50] `http://symfony.com/doc/current/components/serializer.html`

[51] `http://swagger.io/specification/`

[52] `http://swagger.io/swagger-ui/`

### 3.3.2 RESTful zdroje

Sekce popisuje seznam poskytovaných zdrojů; z URL adres je vynechán prefix `/api`.

#### 3.3.2.1 Instalovaný software

Dotazy na zdroj `Software`:

- `GET /software` – vrátí všechny položky softwaru,
- `POST /software` – vytvoří nový software,
- `GET /software/{id}` – vrátí konkrétní software podle jeho ID,
- `DELETE /software/{id}` – odstraní software podle jeho ID.

Příklad těla odpovědi pro HTTP GET požadavek na URL `/api/software/1`:

```
{
  "id": 1,
  "name": "www-client/google-chrome",
  "version": "55.0.2883",
  "description": "Chrome Web Browser",
  "homepage": "https://www.google.com/chrome",
  "image": "/api/images/2"
}
```

(Odpověď je v tomto případě ve formátu JSON.)

#### 3.3.2.2 Požadavky

Dotazy na zdroj `Requirement`:

- `GET /requirements` – vrátí všechny požadavky,
- `GET /requirements/{id}` – vrátí konkrétní požadavek podle jeho ID.

#### 3.3.2.3 Operační systémy

Dotazy na zdroj `OperatingSystem`:

- `GET /operating-systems` – vrátí všechny operační systémy,
- `GET /operating-systems/{id}` – vrátí konkrétní OS podle jeho ID.

#### 3.3.2.4 Obrazy OS

Dotazy na zdroj `Image`:

- `GET /images` – vrátí všechny obrazy,
- `GET /images/{id}` – vrátí konkrétní obraz podle jeho ID.

#### 3.3.2.5 Semestry

Dotazy na zdroj `Semester`:

- `GET /semesters` – vrátí všechny semestry,
- `GET /semesters/{id}` – vrátí konkrétní semestr podle jeho kódu.

#### 3.3.2.6 Předměty

Dotazy na zdroj `Course`:

- `GET /courses` – vrátí všechny předměty,
- `GET /courses/{id}` – vrátí konkrétní předmět podle jeho kódu.

### 3.3.3 Podporované formáty

Framework API Platform podporuje širokou škálu formátů[53]. Aplikace se omezuje na následující podmnožinu:

- JSON (MIME typ application/json),
- XML (application/xml, text/xml),
- YAML (application/x-yaml),
- HTML (text/html).

Požadovaný formát se odesílá v HTTP hlavičce *Accept* (např. *Accept: application/json*).

### 3.3.4 Filtrování výstupu

API podporuje filtrování výstupu. V současné době je možné filtrovat pouze požadavky na software, a to podle názvu nebo jeho části (`title`), typu (`type`) a stavu (`state`). Tyto parametry se zapisují do URL jako query string[54], tedy za otazníkem.

Např. dotaz na `/api/requirements?title=editor&state=needs_review` vrátí všechny požadavky, které mají v názvu slovo „editor“ a nachází se ve stavu „Ke schválení“.

[53] `https://api-platform.com/docs/core/content-negotiation`
[54] `https://en.wikipedia.org/wiki/Query_string`

## 3.4 Testování aplikace

Poslední část implementační kapitoly popisuje proces testování aplikace, který je zaměřen především na automatické testy. Automatické testování je součástí kontinuální integrace na GitLabu (GitLab CI).

Základ aplikace byl rovněž podroben code review (revize kódu další osobou) prostřednictvím požadavků na začlenění[55].

### 3.4.1 Kvalita kódu

Kontrola kvality kódu se opírá o statickou analýzu, která v první řadě zahrnuje kontrolu syntaxe a kontrolu stylu. Níže popsaný postup pro zajištění kvality kódu je inspirován článkem „The Three Pillars of Static Analysis in PHP“ [34].

#### 3.4.1.1 Kontrola syntaxe

Syntaktické chyby jsou nejjednodušší na vytvoření, ale také na odhalení. Kontrolu syntaxe provádí již vývojové prostředí PhpStorm, díky tomu je naprostá většina chyb programátorem bezprostředně opravena.

Za účelem zautomatizování lze kontrolu syntaxe libovolného PHP souboru spustit příkazem `php -l`. Aplikace využívá nástroj **PHP Parallel Lint** [35], který je nadstavbou nad `php -l`. Jeho přední výhodou je možnost paralelní kontroly více souborů.

Kontrola syntaxe pomocí nástroje PHP Parallel Lint lze spustit následujícím příkazem:

```
$ php vendor/bin/parallel-lint --exclude app --exclude vendor .
```

#### 3.4.1.2 Kontrola stylu

Proběhne-li syntaktická kontrola úspěšně, tj. aplikace obsahuje validní PHP kód, je vhodné zkontrolovat dodržení konvencí pro psaní kódu (správné formátování, pojmenování tříd, metod a proměnných, dokumentace, ...), tzv. coding standards. Dodržování zavedených pravidel velmi zvýší čitelnost kódu.

Aplikace dodržuje konvence frameworku Symfony[56]. Pro automatickou kontrolu stylu se používá nástroj **PHP Coding Standards Fixer** [36]. Definice stylistických pravidel a množiny souborů ke kontrole se nachází v konfiguračním souboru `.php_cs.dist`.

Kontrola stylu dle konfigurace lze spustit následujícím příkazem:

```
$ php vendor/bin/php-cs-fixer fix
```

[55] `https://yalantis.com/blog/code-review-via-gitlab-merge-requests-code-review-must/`

[56] `http://symfony.com/doc/current/contributing/code/standards.html`

Tento příkaz zároveň opraví chybné soubory. Spustíme-li stejný příkaz s přepínačem `--dry-run`, pouze se vypíší navrhované opravy.

#### 3.4.1.3 Pokročilá statická analýza

Posledním článkem statické analýzy je nástroj **PHPStan** [37]. PHPStan hledá chyby, aniž by bylo třeba daný kód spouštět, čímž se blíží kompilátorům staticky typovaných jazyků. Nabízí různé nastavení striktnosti (úroveň 0–5, přičemž 5. úroveň je zatím experimentální), díky tomu lze odstranit chyby postupně od těch nejzávažnějších.

Analýza kódu pomocí PHPStan lze spustit následujícím příkazem:

```
$ php vendor/bin/phpstan analyse src tests -l 4
```

Vedle nástroje PHPStan byl v průběhu implementace využíván též PhpStorm plugin **Php Inspections** [38].

### 3.4.2 Jednotkové a funkční testy

Jednotkové testy se zaměřují na testování samostatných tříd, které jsou co možná nejvíce izolovány od zbytku aplikace. Testují se především správné návratové hodnoty metod v závislosti na vstupních parametrech. Pomocí funkčních testů je testována korektnost chování celé aplikace.[39]

Automatické jednotkové a funkční testy aplikace byly realizovány s využitím frameworku **PHPUnit** [40].

#### 3.4.2.1 Struktura a umístění testů

Všechny implementované testy se nacházejí v adresáři `tests/`. Testy backendové části aplikace jsou uloženy v podadresáři `BaseBundle/`, testy frontendu jsou umístěny pod `UiBundle/`. Další členění kopíruje strukturu jednotlivých bundlů (pro `BaseBundle` tedy `Entity/`, `Repository/`, `Service/`, ...).

Každý test typicky testuje jednu konkrétní metodu. Například test metody `create()` třídy `RequirementService` pro vytvoření nového požadavku se nachází v metodě `testCreate()` v `RequirementServiceTest`. Součástí každého testu jsou pak asertační metody, tzv. assertions, které slouží k ověřování předpokladů. Typickým předpokladem je, že metoda testované třídy vrací očekávané hodnoty.

#### 3.4.2.2 Testovací databáze a data fixtures

Pro účely testování není vhodné využívat produkční databázi. Z toho důvodu byla pro testování vyhrazena samostatná databáze, jejíž konfigurace se nachází v souboru `app/config/config_test.yml`.

Samotnému spuštění testů předchází nahrání základních dat, tzv. fixtures, do testovací databáze za pomoci DoctrineFixturesBundle[57]. Vytvoření testovacích dat zajišťují třídy v adresáři `BaseBundle/DataFixtures/ORM/`.

Nahrání dat do testovací databáze lze provést příkazem:

```
$ php bin/console doctrine:fixtures:load --env=test
```

#### 3.4.2.3 Spuštění automatických testů a code coverage

Ke spuštění testů slouží příkaz `$ php vendor/bin/phpunit`. Pomocí přepínače `-c` lze načíst XML konfigurační soubor `phpunit.no-coverage.xml`, který definuje dvě sady testů – jednu pro `BaseBundle` a druhou pro `UiBundle`.

Jednou z dalších možností, které framework PHPUnit nabízí, je generování reportu pokrytí kódu testy, tzn. code coverage. Pro vygenerování reportu je nutné načíst konfigurační soubor `phpunit.coverage.xml`. Report je po ukončení běhu testů uložen do adresáře `var/coverage/` ve formátu HTML; úvodní stránka je `index.html`.

Ukázka spuštění automatických testů:

```
$ php vendor/bin/phpunit -c phpunit.no-coverage.xml

PHPUnit 5.7.19 by Sebastian Bergmann and contributors.

.............................................  47 / 187 ( 25%)
.............................................  94 / 187 ( 50%)
............................................. 141 / 187 ( 75%)
.............................................  187 / 187 (100%)

Time: 13.78 seconds, Memory: 66.00MB

OK (187 tests, 924 assertions)
```

[57] `https://github.com/doctrine/DoctrineFixturesBundle`

# Závěr

V rámci práce byla provedena analýza současného stavu provozu počítačových učeben – popis typů učeben, jejich možností a postupů instalace – a uskutečněn průzkum mezi uživateli s cílem zjistit jejich potřeby. Hypotézy vytvořené na základě rozhovorů v kvalitativní části průzkumu byly následně ověřeny formou elektronického dotazníku. V závěru analytické části práce byly stanoveny funkční a kvalitativní požadavky na novou webovou aplikaci, která částečně odstraní zjištěné nedostatky.

Na základě poznatků z analýzy byla navržena a implementována webová aplikace, jejímž primárním účelem je sběr požadavků na software. Aplikace dále umožňuje zobrazit seznam aktuálně instalovaného softwaru pro jednotlivé obrazy operačních systémů v aktuálním semestru. Správce instalace pak pomocí aplikace spravuje operační systémy a jejich obrazy, a provádí další úkony potřebné pro její provoz, jako import semestrů a předmětů nebo nastavení harmonogramu přípravy učeben pro aktuální semestr.

V první fázi implementace se podařilo naplnit všechny prioritní funkční požadavky, řada dalších podnětů ovšem stále čeká na svou realizaci. Jmenovitě se jedná o možnost přenosu historických požadavků do nového semestru, notifikaci uživatelů o změně stavu jejich požadavků či umožnění vzdáleného otestování instalace. Rovněž prozatím nebyla zprovozněna možnost zadání požadavku výběrem z nabídky, pro což v současnosti nejsou k dispozici datové zdroje. Mezi další návrhy na vylepšení patří například přehled aktuálních požadavků uživatele na domovské obrazovce aplikace, přesunutí komentářů k existujícím požadavkům do samostatných vláken a zpřístupnění importu předmětů běžným uživatelům.

# Literatura


[1] Bradner, S.: Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, RFC Editor, March 1997. Dostupné z: https://www.rfc-editor.org/info/rfc2119

[2] Roques, A.: PlantUML. [online], 2017, [cit. 2017-05-01]. Dostupné z: http://plantuml.com/

[3] JGraph Ltd: draw.io. [online], 2005-2017, [cit. 2017-05-01]. Dostupné z: https://www.draw.io/

[4] PHP Group: PHP. [online], 2001-2017, [cit. 2017-05-01]. Dostupné z: http://php.net/

[5] Potencier, F.: Symfony. [online], 2017, [cit. 2017-05-01]. Dostupné z: http://symfony.com/

[6] The jQuery Team: jQuery. [online], 2017, [cit. 2017-05-03]. Dostupné z: https://jquery.com/

[7] JetBrains: PhpStorm. [online], 2000-2017, [cit. 2017-05-01]. Dostupné z: https://www.jetbrains.com/phpstorm/

[8] Potencier, F.: What is Symfony2? [online], Říjen 2011, [cit. 2017-05-01]. Dostupné z: http://fabien.potencier.org/article/49/what-is-symfony2

[9] Doctrine Team: Doctrine. [online], 2006-2017, [cit. 2017-05-01]. Dostupné z: http://www.doctrine-project.org/

[10] Doctrine Team: Inheritance Mapping. [online], 2006-2017, [cit. 2017-05-01]. Dostupné z: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/inheritance-mapping.html

[11] Potencier, F.: How to Create custom Repository Classes. [online], [cit. 2017-05-01]. Dostupné z: http://symfony.com/doc/current/doctrine/repository.html

[12] Pudil, J.: Filtering data by user input with Kdyby/Doctrine. [online], Červen 2015, [cit. 2017-05-01]. Dostupné z: https://jiripudil.cz/blog/filtering-data-by-user-input-with-kdyby-doctrine

[13] SensioLabs: Twig. [online], 2010-2017, [cit. 2017-05-01]. Dostupné z: https://twig.sensiolabs.org/

[14] Potencier, F.: Creating and Using Templates. [online], [cit. 2017-05-01]. Dostupné z: http://symfony.com/doc/current/templating.html

[15] CodeReviewVideos.com: Symfony Workflow Component Tutorial. [online], [cit. 2017-05-01]. Dostupné z: https://www.codereviewvideos.com/course/symfony-workflow-component-tutorial

[16] Potencier, F.: How to Use the Workflow: Using Events. [online], [cit. 2017-05-01]. Dostupné z: http://symfony.com/doc/current/workflow/usage.html#using-events

[17] Rocha, Z.: clipboard.js. [online], 2017, [cit. 2017-05-01]. Dostupné z: https://clipboardjs.com/

[18] Dobervich, D.: VichUploaderBundle. [online], 2011, [cit. 2017-05-03]. Dostupné z: https://github.com/dustin10/VichUploaderBundle

[19] 1up GmbH: OneupFlysystemBundle. [online], 2017, [cit. 2017-05-03]. Dostupné z: https://github.com/1up-lab/OneupFlysystemBundle

[20] Bushell, D.; Rikkert, R.: Pikaday. [online], 2014, [cit. 2017-05-01]. Dostupné z: https://github.com/dbushell/Pikaday

[21] JS Foundation and other contributors: Moment.js. [online], 2017, [cit. 2017-05-01]. Dostupné z: http://momentjs.com/

[22] Dowling, M.: Guzzle, PHP HTTP client. [online], 2015, [cit. 2017-05-03]. Dostupné z: docs.guzzlephp.org/

[23] Macedo, D.: Sainsbury's guzzle-oauth2-plugin. [online], 2016, [cit. 2017-05-03]. Dostupné z: https://github.com/Sainsburys/guzzle-oauth2-plugin

[24] Potencier, F.: How to Work with Service Tags. [online], [cit. 2017-05-01]. Dostupné z: http://symfony.com/doc/current/service_container/tags.html

[25] Brown, K.; Vaynberg, I.: Select2. [online], 2012-2015, [cit. 2017-05-01]. Dostupné z: https://select2.github.io/

[26] Potencier, F.: How to Use Voters to Check User Permissions. [online], [cit. 2017-05-01]. Dostupné z: http://symfony.com/doc/current/security/voters.html

[27] CJ Patoilo: Milligram. [online], 2017, [cit. 2017-05-01]. Dostupné z: https://milligram.github.io/

[28] Cope, J.: SlickNav. [online], 2016, [cit. 2017-05-01]. Dostupné z: http://slicknav.com/

[29] Gandy, D.: Font Awesome. [online], 2016, [cit. 2017-05-01]. Dostupné z: http://fontawesome.io/

[30] iconsDB.com: iconsDB. [online], 2011-2017, [cit. 2017-05-01]. Dostupné z: http://www.iconsdb.com/custom-color/software-box-icon.html

[31] Twitter and other contributors: Bower. [online], 2016, [cit. 2017-05-01]. Dostupné z: https://bower.io/

[32] OpenSky Project Inc: Assetic. [online], 2010-2015, [cit. 2017-05-01]. Dostupné z: https://github.com/kriswallsmith/assetic

[33] Dunglas, K.: API Platform. [online], 2016, [cit. 2017-05-01]. Dostupné z: https://api-platform.com/

[34] Mirtes, O.: The Three Pillars of Static Analysis in PHP. [online], Prosinec 2016, [cit. 2017-05-01]. Dostupné z: https://medium.com/@ondrejmirtes/three-pillars-of-static-analysis-in-php-f3f5d7bfd61b

[35] Onderka, J.: PHP Parallel Lint. [online], 2012, [cit. 2017-05-01]. Dostupné z: https://github.com/JakubOnderka/PHP-Parallel-Lint

[36] Potencier, F.; Rumiński, D.: PHP Coding Standards Fixer. [online], 2012-2017, [cit. 2017-05-01]. Dostupné z: https://github.com/FriendsOfPHP/PHP-CS-Fixer

[37] Mirtes, O.: PHPStan – PHP Static Analysis Tool. [online], 2016-2017, [cit. 2017-05-01]. Dostupné z: https://github.com/phpstan/phpstan

[38] Reznichenko, V.: Php Inspections (EA Extended). [online], 2017, [cit. 2017-05-01]. Dostupné z: https://plugins.jetbrains.com/plugin/7622-php-inspections-ea-extended-

[39] Zamrzla, J.: Testování a tvorba testovatelného kódu v PHP. [online], Srpen 2012, [cit. 2017-05-03]. Dostupné z: https://www.zdrojak.cz/clanky/testovani-a-tvorba-testovatelneho-kodu-v-php/

[40] Bergmann, S.: PHPUnit. [online], 2001-2017, [cit. 2017-05-01]. Dostupné z: https://phpunit.de/

[41] FIT ČVUT: Pravidla a zásady projektů FIT. [online], 2016, [cit. 2017-05-01]. Dostupné z: https://docs.google.com/document/d/1umkLCuvYY1EYMat8jLnYfUj5WyccC3s-30thPvePot4/edit?usp=sharing

Příloha **A**

# Rozhovory v rámci kvalitativního průzkumu

Cílem bylo provést kvalitativní průzkum formou polostrukturovaného rozhovoru[58] především mezi vyučujícími a dalšími zaměstnanci FIT ČVUT v Praze, kteří ke své činnosti využívají fakultní počítačové učebny.

## A.1 Výběr respondentů

Respondenti se dělí primárně do dvou skupin. První jsou **běžní uživatelé**, typicky vyučující, druhou pak **správci učeben**.

Uživatelé využívají připravené softwarové prostředí v učebnách. Před zahájením každého výukového semestru mají možnost specifikovat své požadavky na software. Úlohou správců je sběr a zpracování těchto požadavků. Příprava učeben zahrnuje instalaci operačních systémů a aplikačního softwaru včetně následné distribuce na cílové stanice v učebnách.

**Uživatelské skupiny:**

- běžní uživatelé
  - vyučující
  - ostatní neakademičtí zaměstnanci
  - studenti
- správci učeben
  - technici

[58] `https://cs.wikipedia.org/wiki/Polostrukturovan%C3%BD_rozhovor`

## A.2 Scénář rozhovorů

Sekce shrnuje témata, kterých se otevřené rozhovory dotýkají. Témata jsou přizpůsobena roli uživatele. Respondent by měl být veden formou hloubkového rozhovoru.[59].

### A.2.1 Obecné

1. Popis projektu
    a) „Výuka v počítačových učebnách fakulty čím dál častěji vyžaduje specifický režim v provozu (specializované instalace, virtualizace, aktuální software apod.). Požadavky lze se stávající infrastrukturou pouze těžko plnit. Naším cílem je vylepšit systém řízení učeben a nabídnout řešení na míru výuce a dalším akcím. Zajímá nás, jaké jsou Vaše potřeby a jak si tento nový systém představujete.“
2. Prioritizace klíčových témat rozhovoru (závěrem)
    a) „Co je pro Vás ze všeho uvedeného nejdůležitější a co méně?“

### A.2.2 Uživatel (vyučující)

1. Potřeby uživatele
    a) „K jakým účelům využíváte počítačové učebny?“
    b) „Jaké operační systémy v rámci své činnosti v učebnách používáte?“
    c) „Jaké máte speciální požadavky na požadované instalace?“
2. Stávající postup instalace prostředí
    a) „Jak postupujete, když v učebnách potřebujete zajistit instalaci potřebného prostředí?“ (shromáždění a specifikace požadavků, testování instalace, reklamace)
    b) „Vyhovuje Vám stávající postup instalace?“
    c) „Čím je pro Vás tento postup limitující?“
    d) „Co byste na současném postupu změnil(a)?“
3. Možnosti vylepšení
    a) Zobrazení historie požadavků pro předchozí semestry (web UI)
    b) Plánování instalace na konkrétní čas a místo

[59] `https://cs.wikipedia.org/wiki/Hloubkov%C3%BD_rozhovor`

c) Výběr jiného OS vedle aktuálně nabízených (Gentoo GNU/Linux, MS Windows, macOS)

d) Instalace OS „na míru“ (navolení specifických požadavků na SW)

e) Instalace vlastního obrazu OS

f) Vzdálené ověření instalace (na virtuálním stroji)

### A.2.3 Správce učebny

1. Obecné

   a) „Jaké fakultní počítačové učebny spravujete?“ (standardní/specializované)

   b) „Jaké operační systémy spravujete?“

   c) „S jakými problémy se jako správce nejčastěji potýkáte?“

2. Požadavky na instalace

   a) „Jakým způsobem vyzýváte uživatele ke specifikaci požadavků?“

   b) „Jaké jsou typické požadavky uživatelů?“

   c) „Máte od uživatelů požadavky, kterým v současnosti nelze vyhovět?“

   d) „Jakým způsobem s jejich požadavky pracujete?“ (skladování, verzování, . . . )

   e) „Jakým způsobem řešíte reklamace či požadavky na úpravy v průběhu semestru?“

3. Instalace a testování

   a) „Jak postupujete při instalaci požadovaného prostředí?“

   b) „Jakým způsobem si uživatelé mohou instalace otestovat?“

   c) „Vyhovuje Vám stávající postup instalace?“

   d) „Co byste na současném postupu změnil(a)?“

   e) „Co by se dalo v procesu instalace zautomatizovat?“

## A.3 Zápisy z rozhovorů

Z rozhovorů se zástupci jednotlivých uživatelských skupin vznikly strukturované zápisy. Každý z níže uvedených zápisů byl po uskutečnění rozhovoru odeslán příslušnému respondentovi k revizi.

### A.3.1 Vyučující

*30. 5. 2016, 14:00, budova FIT, kancelář A-1331*

- Počítačové učebny využívá v rámci předmětů, které na fakultě vyučuje.
- K výuce využívá OS Gentoo Linux s dodatečnou instalací potřebného softwaru, který základní instalace neobsahuje.
- Instalaci potřebného prostředí v minulosti řešil individuálně se správcem učebny. V současné době má předpřipravené instalační skripty.
- V návaznosti na plánované změny prostředí pro účely výuky by chtěl mít větší kontrolu nad instalací OS.
- Větší kontrolu nad instalací by uvítal také v průběhu semestru. Pro vybraná cvičení by tak využil upravené instalace obsahující pouze relevantní software.
- Ověření správnosti instalace v současnosti probíhá ve 2 fázích:
  1. na testovacím stroji v době specifikace požadavků,
  2. přímo v učebnách před zahájením semestru.
- Funkci otestování instalace přes vzdálený přístup by mimo jiné využil také v průběhu semestru pro otestování konkrétních postupů pro následující cvičení.
- Uvítal by možnost výběru z dalších linuxových distribucí.
- Smysl mu dává také možnost instalace vlastního obrazu OS.

### A.3.2 Vyučující, správce HW laboratoře, student doktorského programu FIT

*1. 6. 2016, 13:30, budova FIT, kancelář A-1054*

- Jako vyučující využívá fakultní hardwarové laboratoře.
- Primárně využívá operační systém MS Windows.
- K výuce potřebuje dodatečný software – výhradně od společnosti Xilinx.
- Své požadavky na instalace řeší s M. Vrátilem, správcem učeben a laboratoří katedry číslicového návrhu.
- Testování instalace provádí přímo v hardwarových laboratořích. V případě potřeby testuje i v průběhu semestru s cílem ověřit konkrétní postupy pro cvičení a zadané úlohy.

- Příležitostně využívá standardní, síťově bootovaný systém Gentoo Linux (bez dalších požadavků na dodatečný software).
- Navrhuje umožnit studentům instalaci vlastního image (např. zálohy v případě výpadku).

### A.3.3 Vyučující, zástupce vedoucího laboratoře 3D tisku FIT, student navazujícího Mgr. programu FIT

*3. 6. 2016, 11:30, budova FIT, kancelář A-1153*

- Výuka předmětu 3D Tisk probíhá ve fakultní hardwarové laboratoři.
- Používá se standardní, síťově bootovaná, linuxová distribuce Gentoo.
- Jako vyučující má rozsáhlejší požadavky na speciální software k 3D tisku (příprava modelů apod.)
- Součástí požadavků je také běžně používaný software (Git, ...), aby bylo zaručeno, že bude obsažen i v nové instalaci.
- Pro zajištění instalace potřebného softwaru postupuje následovně:
  1. reaguje na výzvu ke specifikaci požadavků obdrženou e-mailem
  2. v historii odeslaných zpráv dohledá své požadavky pro předchozí semestr
  3. požadavky zkontroluje a případně aktualizuje
  4. nové požadavky zasílá na Helpdesk FIT
- Při tomto postupu identifikuje následující problém: výzva k specifikaci požadavků mu někdy nepřijde. Přesto, že je vyučující, jako zaměstnanec na dohodu o provedení práce nedostává v období letních prázdnin zprávy zaslané do fakultního učitelského mailing listu *ucitele-l*. Vhodnou dobu pro odeslání požadavků na Helpdesk FIT tak mnohdy odhaduje.
- Současný způsob zajištění potřebných instalací mu nevyhovuje.
- Navrhuje zřídit společný Git repozitář s adresáři podle programů a konfiguračním souborem pro každý předmět zvlášť. Konfigurační soubory by pak obsahovaly požadovaný software. Požadavky na instalace či přímo instalační skripty by se zadávaly formou požadavku na začlenění (pull request).
- Uživatelé, kteří nepracují s verzovacím systémem Git, by své požadavky mohli zadávat přes uživatelské rozhraní webové aplikace, která by s Gitem komunikovala na pozadí.

- Testování instalace provádí začátkem semestru přímo v některé z fakultních hardwarových laboratoří. Při testování ověřuje úspěšný boot systému a správnost instalace požadovaného softwaru.
- V minulosti již narazil na chybějící instalaci požadovaného softwaru.
- Možnost vzdáleného otestování instalace se mu líbí a využil by ji.
- Rád by využil také jiné linuxové distribuce (např. Fedora), tuto možnost však nepovažuje za podstatnou.
- Vytvoření a instalace vlastního obrazu mu přijde jako dobrý nápad, ale obává se, že bude problematické zajistit jednotné přihlašování a přístup k domovskému adresáři.

### A.3.4 Správce fakultních hardwarových laboratoří (KČN)

*8. 6. 2016, 16:05, budova FIT, kancelář A-1054*

- Spravuje počítačové učebny pro katedru číslicového návrhu.
- Používá se primárně operační systém MS Windows.
- Pro potřeby některých předmětů (3D Tisk, ...) je k dispozici síťově bootovaná linuxová distribuce Gentoo pod správou ICT oddělení.
- Požadavky na instalace řeší pouze v rámci učeben a laboratoří katedry číslicového návrhu. Vzhledem k omezenému počtu vyučujících na KČN řeší požadavky s vyučujícími osobně. Může tak operativně vyřešit případné reklamace či usměrnit nerealizovatelné požadavky.
- OS Windows vyžaduje pro potřeby uživatelů KČN rozsáhlé úpravy, vytváří se tzv. tlustý obraz (~120 GB), který již obsahuje všechny potřebné aplikace, ovladače apod. Tento obraz je následně distribuován na počítače v HW laboratořích.
- Testování obrazu probíhá obratem po instalaci s konkrétním žadatelem. Instalaci je rovněž možné do předem určeného data otestovat na jedné vybrané stanici. Poté je obraz naklonován na zbývající stanice.
- Podotýká, že aplikace s přístupem k hardware je obtížnější testovat.
- Reklamace i návrhy na drobné změny a vylepšení přijímá a snaží se jim vyhovět i v průběhu semestru.
- Upozorňuje, že při testování obrazu přes vzdálenou plochu může být problém s aplikacemi pro HW, specifickými ovladači či aplikacemi vyžadujícími 3D akceleraci. Toto platí pouze při využití Remote Desktop Protocol (RDP), nikoliv pro přenos bitmapové plochy, např. Virtual Network Computing (VNC).

### A.3.5 Vyučující, správce laboratoře pro výuku administrace a síťové laboratoře

*9. 6. 2016, 15:30, budova FIT, kancelář A-1233*

- Spravuje laboratoř pro výuku administrace, tzv. bourací učebnu, a síťovou laboratoř.
- V těchto učebnách poskytuje pouze infrastrukturu. Požadavky na instalace (OS, aplikační software apod.) neřeší. Uživatelům bourací učebny je povoleno v průběhu semestru vytvářet a používat různé operační systémy a provádět jejich úpravy dle vlastního uvážení.
- S konceptem sjednocení principu fungování specializovaných učeben a standardních počítačových učeben, kdy potřebné instalace jsou připraveny před začátkem semestru a v jeho průběhu se mění pouze v akutních případech, identifikuje následující problémy:
  1. Obě specializované laboratoře vyžadují zvýšené nároky na bezpečnost, proto je jejich síťová infrastruktura oddělena od zbytku sítě.
  2. Druhý problém může nastat z hlediska výkonu. Pro specializované učebny jsou vyhrazeny dedikované linky, které jsou schopné vyhovět datovým tokům během instalace obrazů OS na jednotlivé stanice.
- V návaznosti na předchozí bod navrhuje se v první fázi návrhu nového systému omezit pouze na standardní počítačové učebny.
- Ve standardních učebnách by pak postup mohl být následující:
  1. Uživatelé (vyučující) kdykoli v průběhu semestru navolí své požadavky na instalace, včetně času a místa instalace.
  2. Během následující noci se dávkově spustí instalace všech nově navolených instalací na požadované stroje. Tím je zaručeno nenarušení provozu učeben.
- Za důležité považuje, aby si nový systém pamatoval historii požadavků, kterou by si uživatel mohl zobrazit, upravit pro aktuální potřeby a použít pro nový semestr.
- Při výběru softwaru by bylo dobré umožnit zafixovat jeho konkrétní verzi, neboť ne vždy musí být žádoucí verze nejaktuálnější.
- Možnost vzdáleného otestování instalace by využil. Zároveň si přeje ponechat možnost testování přímo na cílové stanici, především kvůli odlišnostem v hardware.
- Jako vyučující by chtěl mít možnost navolit pro konkrétní učebnu a čas primární položku v bootovacím menu.

- Dále by uvítal schopnost vzdáleně ovládat stanice (reset, vypnutí) i mimo specializované laboratoře. Navíc by si přál mít možnost se na jednotlivé stanice vzdáleně připojit v administrátorském režimu tak, aby mohl v případě potřeby prozkoumat stav systému a aktuální aktivity uživatele. To se osvědčilo při předcházení nedorozuměním během testů či soutěží, kdy může dojít k reklamaci skutečné nebo domnělé nefunkčnosti prostředí a z uživatelského konta není z různých důvodů možné nebo vhodné stav prozkoumat.

- Z vlastní zkušenosti ví, že někteří uživatelé sami nevědí, co ve skutečnosti potřebují (nedokáží specifikovat své požadavky). Proto by bylo výhodné mít možnost méně zkušeným uživatelům poskytnout stavební bloky systému, u nichž bude známo, jaké typy SW obsahují, ale bez detailní specifikace. (Něco jako skupiny, známé např. z instalací RedHat distribucí.) Zkušenější by naopak měli mít možnost specifikovat vše přesně dle svých potřeb.

### A.3.6 Vyučující, správce systému ProgTest

*15. 6. 2016, 16:45, budova FIT, kancelář A-1331*

- Počítačové učebny využívá pro účely předmětů programování a algoritmizace (BI-PA1, BI-PA2).

- Při specifikaci požadavků na nové instalace vychází vždy z požadavků pro předchozí semestr.

- Testování potřebné instalace provádí osobně s T. Kadlecem na testovací stanici. Testování je vždy značně specifické a trvá v řádu minut až hodin.

- Vzdálený přístup pro testování instalovaného prostředí se mu jeví jako dobrý nápad, nicméně tuto možnost na základě vlastní zkušenosti označil za nepříliš spolehlivou.

- V průběhu semestru obvykle nemá potřebu do instalací zasahovat.

- Funkce zobrazení historie požadavků pro něj není klíčová, nicméně dokáže si představit její využití.

- Podobný postoj zaujímá k možnosti zafixování verze (zvolení konkrétní verze) softwaru.

- Dále zmiňuje tzv. testovací image využívanou při testování studentů. Testovací image se od standardní liší tím, že obsahuje pouze omezenou sadu software.

- Nový systém si představuje ve stylu skládání komponent. Jinými slovy, existoval by jeden image se základním programovým vybavením. Ostatní obrazy by pak obsahovaly tento společný základ s přidaným specifickým softwarem.

- V novém systému by bylo vhodné zavést monitoring stanic a instalovaných prostředí. Kontrolu by automatizovaně prováděl řídící server např. 1x týdně.

- Chtěl by mít možnost přeskupit či skrýt nerelevantní položky v bootovacím menu.

- Systém řízení chodu učeben by měl poskytovat aplikační rozhraní (API) umožňující strojové řízení.

### A.3.7 Vyučující

*28. 6. 2016, 10:30, budova FIT, kancelář A-1331*

- Vyučuje předměty týkající se programování v shellu, webu a multimédií.

- Během výuky využívá jak standardní počítačové učebny, tak i bourací učebnu.

- Upřednostňuje OS Linux, studenti v jeho předmětech však mohou fungovat i v prostředí MS Windows.

- S aktuální nabídkou operačních systémů si vystačí.

- Testování instalací provádí přímo v učebnách dle potřeby.

- V případě vážnějších problémů se obrací na správce učeben, a to i v průběhu semestru.

- Možnost vzdáleného otestování instalace (např. prostřednictvím VNC) mu přijde zajímavá. Problém vidí v testování softwaru využívající 3D akceleraci (např. Blender, software pro 3D modelování).

- Plánování instalace na konkrétní čas a místo by využil – například při testování studentů, kdy se využívá speciální image. Zvolit automatickou instalaci testovací image na vybrané učebny ve zvolený čas (a následně vše vrátit do původního stavu) by mu pomohlo.

- Jako vyučující by také chtěl mít informaci o stavu počítačů v učebně (statistika, kolik počítačů nabootovalo apod.). Dále by chtěl mít možnost počítače hromadně zapnout/vypnout.

- Při používání PC určeného vyučujícím se často potýká s problémy jako nefunkčnost projektoru nebo zobrazení černé obrazovky po zadání uživatelského jména a hesla. Tento problém by mohla řešit speciální, lokálně instalovaná image.

- Přál by si, aby existovala jakási minimalistická image, která by nevyžadovala autentizaci, obsahovala pouze základní software (webový prohlížeč, ...) a neuchovávala data. Výhodou takové image by byla především rychlost bootování a stabilita.

- V potenciální aplikaci pro sběr požadavků na instalace navrhuje nevztahovat požadavky k jednotlivým uživatelům, nýbrž ke skupině uživatelů či přímo vyučovaným předmětům. Na výuce každého předmětu se většinou podílí více vyučujících, bylo by tedy vhodné sdílet požadavky s kolegy.

- Zmiňuje v minulosti existující webovou stránku, která zobrazuje aktuálně instalovaný software v počítačových učebnách, včetně verzí.

### A.3.8 Správce běžných počítačových učeben

*11. 7. 2016, 11:00, budova FIT, kancelář A-1331*

- Spravuje standardní počítačové učebny na fakultě, tj. všechny učebny mimo bouracích. Také správu hardwarových laboratoří ponechává jejich lokálnímu správci M. Vrátilovi.

- Spolu s J. Kadlečkem odpovídá za instalaci a údržbu obrazu operačního systému MS Windows. Konkrétně se jedná o instalaci softwaru pro zmíněný operační systém dle uživatelských požadavků a distribuci výsledného obrazu na počítačové stanice v učebnách.

- Požadavky na instalace jsou dostupné v systému Request Tracker formou tiketů. Je možné zde dohledat i historické požadavky, které jsou součástí uzavřených tiketů.

- Při instalaci vychází z obrazu pro předchozí výukový semestr a se zohledněním nových požadavků jej zaktualizuje. Jednou za čas původní obraz zahodí a vychází z čisté instalace.

- Instalace typicky řeší jednou za půl roku vždy před začátkem semestru. Méně často pak mimo výuku pro účely různých akcí (např. v období letních prázdnin). Pro tyto akce vytváří speciální krátkodobou instalaci.

- Testování instalace umožňuje provést ve své kanceláři. Následně je možné instalaci otestovat přímo v učebně.

- Na současném řešení mu vadí především to, že instalace se nespouští automaticky. Při instalaci např. nemůže hromadně nastavit, jaký operační systém má při následujícím spuštění stanice nabootovat. Rád by měl možnost zvolit operační systém MS Windows jako defaultní bootovací záznam.

- V nové aplikaci podporující správu počítačových učeben by chtěl mít možnost automatického informování uživatelů o proběhlé instalaci (např. formou e-mailu) a její zpětné kontroly od uživatele. Tím by si ověřil, že instalace je v pořádku a za další případné reklamace nenese vinu.

- Nová aplikace by v rámci sběru požadavků měla umožnit přikládání návodů na složitější instalace (komplikované licenční podmínky apod.).

- Dále by měla řešit deduplikaci požadavků.

- Uvítal by také možnost základního hromadného ovládání zvolené skupiny počítačů (zapnutí, reset, ...).

- Zmiňuje v minulosti fungující webovou aplikaci zobrazující požadavky na software a stav instalace.

- Dále poznamenává, že i pro MS Windows jsou dostupné nástroje pro správu balíčků. Žádný takový software ale momentálně nepoužívá.

### A.3.9 Správce Apple učebny

*13. 7. 2016, 15:00, budova FIT, kancelář A-1328*

- Spravuje fakultní Apple učebnu, která se využívá především k výuce předmětů týkajících se vývoje iOS aplikací, ale také bezpečnosti, kryptologie a dalších. Přístup do této učebny je možný pouze pod dozorem, případně v době výuky po domluvě s vyučujícím.

- Pro účely výuky v Apple učebně připravuje operační systém macOS.

- V učebně je možné rovněž využít běžné instalace Gentoo GNU/Linux a MS Windows, jež jsou pod správou ICT oddělení. Změny těchto běžných instalací se promítnou také do Apple učebny.

- Požadavky na instalace řeší osobně s vyučujícími příslušných předmětů, které využívají Apple učebnu a operační systém macOS.

- Při instalaci vychází z čistého obrazu operačního systému macOS a repozitáře se softwarovými balíčky. Využívá automatizační framework na stahování nových verzí softwaru.

- Instalaci chce mít pod kontrolou, proto instalovaný software následně testuje. Pokud by instalace probíhala plně automaticky, neměl by jistotu, že instalovaný software je funkční a v systému nevzniknou kolize. Při plně automatické instalaci vznikají také bezpečnostní rizika.
- Vyučující mají možnost otestovat potřebný software přímo v Apple učebně.
- Reklamacím v průběhu semestru, vzhledem k nízkému počtu předmětů využívajících macOS, obvykle vyhoví.
- V rámci konsolidace systému řízení počítačových učeben by si přál zlepšit především automatizaci instalace OS Linux a Windows.
- Smysl nové aplikace vidí především pro standardní počítačové učebny.
- Jakožto správce by v aplikaci pro sběr požadavků chtěl vidět aktuální seznam požadavků na operační systém který spravuje.
- Přál by si být notifikován (např. e-mailem) o nových požadavcích.
- Zároveň by uvítal schopnost se k požadavkům vyjádřit; např. ke každému požadavku přiřadit status „připravuje se“, „instalováno“, „won't fix“ apod.
- Podotýká, že při automatizované instalaci dle uživatelských požadavků může nastat problém v případě, že instalovaný software vyžaduje pro zajištění plné funkcionality spuštění s administrátorským oprávněním.
- Navrhuje přidat funkci zobrazení přehledu nových verzí softwaru včetně informace o změnách oproti původním verzím (changelog).
- Zajímavá mu přijde také možnost vzdáleného otestování instalace.

### A.3.10 Ing. Tomáš Kadlec, vedoucí ICT oddělení, správce běžných počítačových učeben, vyučující

*8. 8. 2016, 8:00, budova FIT, kancelář A-1337*

- Počítačové učebny využívá k výuce a jednorázovým akcím.
- K těmto účelům používá výhradně OS Gentoo GNU/Linux, buď lokálně instalovaný nebo síťový boot (v závislosti na dostupnosti v konkrétní učebně).
- Požaduje následující SW vybavení:
  - IDE PhpStorm,

  - PHP v aktuální verzi s rozšířeními a podporou frameworku Symfony,
  - podpora SQLite.

- Tyto požadavky, vzhledem ke své roli správce instalace, zahrnuje přímo do existujícího playbooku nástroje Ansible.
- Zmíněný postup instalace naplňuje potřeby jeho výuky. V případě nutnosti může snadno provést úpravu.
- Přeje si možnost zobrazení historie požadavků pro předchozí semestry.
- Plánování instalace na konkrétní čas a místo nepovažuje za nutnost. Instalaci využívají studenti pro práci na semestrálních úlohách. Je výhodné, že je k dispozici ve všech učebnách.
- Pro stávající výuku by pravděpodobně rovněž nevyužil možnosti výběru jiného OS, instalace OS „na míru“ ani instalaci vlastního obrazu OS.
- Vzdálené ověření instalace považuje za zajímavou možnost, která by usnadnila proces validace obrazů.
- V roli správce zajišťuje:
  - vytváření a instalaci obrazů (skript, nověji pomocí Ansible),
  - správu běžných učeben,
  - správu projekce.
- Z operačních systémů spravuje Gentoo GNU/Linux.
- Častým problémem, se kterým se jako správce potýká, je nefunkční software či některé jeho vlastnosti. Typicky je to způsobeno nedostatečnou validací před začátkem semestru. Následná oprava představuje riziko zavlečení jiné chyby a nefunkčnosti jiného SW, který před opravou funkční byl (regrese). Bez zopakování celého validačního procesu či automatizace tomuto nelze předejít.
- Výzvu ke specifikaci požadavků zasílá uživatelům e-mailem. Výzva platí pro učebny s výjimkou HW laboratoří.
- Uživatelé většinou vyžadují běžnou instalaci, takže požadavky ani nepíší. Uživatelů, kteří požadavky zasílají je relativně málo 5-20.
- Požadavky se zasílají na helpdesk@fit.cvut.cz do aplikace RT. Je jim přiřazen řešitel a stav.
- Reklamace a další požadavky na úpravy v průběhu semestru řeší stejným způsobem – e-mailová výzva vyučujícím, správa reklamací v RT.

- Při instalaci vytváří Ansible playbook, který obsahuje veškeré požadavky. Změny jsou prováděny na základě porovnání požadavků v RT a obsahu playbooku. Jsou instalovány aktuální stabilní verze SW.
- Cca 14 dnů před začátkem semestru je nainstalována jedna z učeben, kde si vyučující mohou vyzkoušet instalaci.
- Stávající postup instalace mu nevyhovuje. Zejména zpracování požadavků je komplikované. Pro některé OS by šlo z požadavků rovnou generovat instalační skript.
- V současném postupu by dále uvítal lepší správu požadavků, jejich vazbu na předměty a sdílení (validaci) mezi vyučujícími.

Příloha B

# Dotazník ke kvantitativnímu průzkumu

Na základě kvalitativního průzkumu (kap. 1.2.1) vznikl interaktivní webový dotazník[60]. Tato příloha shrnuje znění dotazníku s hrubým shrnutím a prioritizací odpovědí.

Dotazník byl cílen především na akademické zaměstnance FIT ČVUT. Z tohoto důvodu byla žádost o jeho vyplnění rozeslána e-mailem do fakultního mailing listu pro vyučující. Dotazník byl rozeslán také prostřednictvím sociálních sítě Twitter, kde jej měli možnost vyplnit vedle zaměstnanců také studenti fakulty. V době vyhodnocení (26. 9. 2016) měl dotazník 30 odpovědí.

## B.1 Znění dotazníku s hrubým vyhodnocením odpovědí

Dotazník ověřuje potřeby uživatelů fakultních počítačových učeben. Cílem je vylepšit způsob sběru požadavků na instalace, plánování instalací a nabídnout řešení na míru výuce a dalším akcím. Dotazník obsahuje 13 otázek a jeho vyplnění zabere 5 až 10 minut.

### B.1.1 Otázky dotazníku

1. **Využíváte ke své činnosti na fakultě počítačové učebny?**
   (Jedna možná odpověď)

   a) Ano (30)

   b) Ne (0)

[60] Dotazník byl vytvořen pomocí služby Formuláře Google (`https://www.google.cz/intl/cs/forms/about/`).

2. **K jakým účelům počítačové učebny využíváte?**
   (Více možných odpovědí)
   a) K výuce (30)
   b) K jednorázovým akcím (školení apod.) (7)
   c) Počítačové učebny spravuji (2)
   d) Jiné (Volná odpověď) (0)
3. **Jaká je Vaše role v předmětu?**
   (Více možných odpovědí)
   a) Cvičící (27)
   b) Přednášející (24)
   c) Garant (9)
   d) Proseminující (5)
   e) Vedoucí laboratoří (4)
   f) Jiné (Volná odpověď) (2)
      i. Student
      ii. Testující
   g) Žádná (0)
4. **Kladete na instalace v učebnách speciální požadavky?**
   Např. specifický software pro předmět, který vyučujete.
   (Jedna možná odpověď)
   a) Ano (18)
   b) Ne, vystačím si s běžnými instalacemi (Gentoo GNU/Linux, MS Windows) (12)
5. **Je pro Vás důležité vidět historii Vašich požadavků?**
   Např. seznam požadavků pro předchozí semestr.
   (Jedna možná odpověď)
   a) Rozhodně ano (7)
   b) Spíše ano (10)
   c) Spíše ne (7)
   d) Rozhodně ne (6)
6. **Spolupracujete při specifikaci požadavků se svými kolegy?**
   Např. spolupráce vyučujících jednoho předmětu.
   (Jedna možná odpověď)
   a) Ano, pravidelně (15)

b) Jen občas (11)

c) Ne, nikdy (4)

d) Jiné (Volná odpověď) (0)

7. **Využili byste při specifikaci požadavků následující funkce?**
(Rozhodně ano / Spíše ano / Nevím, možná / Spíše ne / Rozhodně ne)

   a) Rozhodně ano

      i. Seznam aktuálně nainstalovaného softwaru
      (22 / 6 / 1 / 0 / 1)

      ii. Informace o stavu instalace (zpracování Vašich požadavků)
      (16 / 11 / 1 / 1 / 0)

      iii. Možnost výběru softwaru z nabídky
      (14 / 8 / 5 / 2 / 1)

      iv. Zobrazení historie požadavků s možností snadného znovupoužití
      (13 / 8 / 4 / 3 / 2)

      v. Přehled změn oproti původním verzím běžných instalací
      (12 / 14 / 2 / 2 / 0)

      vi. Volba verze softwaru (nejaktuálnější, starší)
      (12 / 10 / 4 / 3 / 1)

   b) Spíše ano

      i. Přiložení textových návodů na složitější instalace
      (9 / 5 / 10 / 4 / 2)

      ii. Přiložení instalačních skriptů
      (6 / 8 / 9 / 4 / 3)

8. **Jaké další funkce byste využili v rámci specifikace požadavků na instalace?**
(Volná odpověď)

   Odpovědi zahrnovaly následující náměty:

   a) automatický přenos požadavků z předchozího do nového období,

   b) možnost podívat se a využít návrhy změn kolegů, resp. možnost koordinace návrhů více učitelů,

   c) možnost doinstalací, případně oprav během semestru,

   d) zadávání požadavků prostřednictvím verzovacího systému Git,

   e) možnost veta v upgradu OS nebo aplikačního softwaru, pokud změna má zásadní dopady na výukovou dokumentaci (EDUX),

   f) kontakt s živou bytostí během specifikace a dalšího případného řešení problémů s instalací.

9. **Využili byste následující možnosti počítačových učeben?**
(Rozhodně ano / Spíše ano / Nevím, možná / Spíše ne / Rozhodně ne)

   a) Nabídka více operačních systémů, linuxových distribucí
   (7 / 6 / 6 / 10 / 1)

   b) Volba minimalistického OS (základní software, bez přihlášení, rychlý start)
   (2 / 7 / 9 / 9 / 3)

   c) Instalace OS „na míru“ (speciální prostředí dle Vašich požadavků)
   (4 / 6 / 6 / 8 / 6)

   d) Instalace vlastního obrazu OS
   (3 / 4 / 6 / 10 / 7)

10. **Využili byste možnost plánování instalace na konkrétní čas a místo?**
Např. naplánování instalace speciálního prostředí pro jednorázovou akci na konkrétní čas a učebnu.
(Jedna možná odpověď)

   a) Rozhodně ano (5)

   b) Spíše ano (8)

   c) Spíše ne (12)

   d) Rozhodně ne (5)

11. **Ověřujete před zahájením semestru, resp. akce, že instalace splňuje Vaše požadavky?**
(Jedna možná odpověď)

   a) Ano, instalaci ověřuji v některé z učeben (15)

   b) Ano, instalaci ověřuji osobně se správcem (2)

   c) Ne, instalaci neověřuji (10)

   d) Jiné (Volná odpověď) (3)

      i. Jeden respondent uvádí, že instalace je často připravena až první den výuky, což znemožňuje její včasné otestování. Jinak by ji rád ověřil.

      ii. Další podotýká, že jako externímu pracovníkovi mu nezbývá, než se nechat vždy překvapit.

      iii. Jiný instalaci ověřuje ve všech učebnách, které následně využívá.

12. **Využili byste možnost vzdáleného otestování instalace?**
Např. přes vzdálenou plochu.
(Jedna možná odpověď)

    a) Rozhodně ano (10)
    b) Spíše ano (12)
    c) Spíše ne (6)
    d) Rozhodně ne (1)

13. **Chcete se podílet na projektu počítačových učeben?**
Zanechte nám prosím svoji e-mailovou adresu.
(E-mail)

    Zájem o spolupráci vyjádřilo zanecháním své e-mailové adresy 7 respondentů.

Příloha C

# Pravidla a zásady projektů FIT

Příloha obsahuje kompletní a nezměněné znění dokumentu „Pravidla a zásady projektů FIT“ [41] platné ke dni 30. 9. 2016. Dokument slouží jako podklad pro stanovení požadavků na kvality aplikace (kap. 1.5).

*FIT ČVUT, 2016*

# Pravidla a zásady projektů FIT

Tento dokument popisuje preferovaný způsob řešení projektů FIT. Pokud některá část tohoto dokumentu není vůči konkrétnímu projektu efektivní nebo na ní není dostatek prostředků, je jí snížena úroveň nezbytnosti[1] nebo je od ní zcela odstoupeno. O takovém rozhodnutí musí existovat záznam.

Tento dokument je nedílnou součástí projektové dokumentace jako příloha ke kapitole Požadavky na kvalitu. Veškeré oblasti tohoto dokumentu, které uvedená kapitola nezmíní, se předpokládají a požadují. Výchozí úrovní nezbytnosti požadavků (není-li uvedeno jinak) je [MUST].

## *Obsah dokumentu*



[1] Key words for use in RFCs to Indicate Requirement Levels

# Dokumenty

Ke každému projektu vzniká sada samostatných dokumentů v čele s harmonogramem. Dokumenty jsou (až na výjimky) psané česky. Výchozím dokumentem je Harmonogram, který podléhá schvalování. Není-li uvedeno jinak, o projektu rozhoduje vedení fakulty (grémium děkana).

## Harmonogram

Stěžejním rozcestníkem projektu je iterativní harmonogram. V každém běhu se předpokládá realizace jen takových funkcí a vlastností aplikace, které náleží do daného běhu. Každý běh (iterace) obsahuje všechny uvedené fáze harmonogramu. Pro každou fázi je stanoven termín předpokládaného dokončení. Každá fáze podléhá schvalování vedením písemnou formou. Obsah harmonogramu se s postupem času upřesňuje.

1. **Analytická fáze**
    - Průzkum existujících řešení (dále jen SOTA[2]).
    - Vytvoření hypotéz (kvalitativní průzkum).
        i. Uživatelské skupiny
        ii. Potřeby uživatelů
    - Ověření hypotéz (kvantitativní průzkum dotazníkem).
    - Sestavení požadavků na kvality (vycházející z tohoto dokumentu).
    - Sestavení požadavků na funkce (prioritní seznam rozdělený dle úrovní nezbytnosti).
    - Posouzení SOTA vůči požadavkům na funkce.
2. **Návrhová fáze**
    - Prototypování (paralelní).
    - Diagramy (procesní, aktivit).
    - Scénáře průchodu, user-stories, případové studie
        i. pro všechny vznikající funkce
        ii. testování (např. formou storyboarding, inspekce).
    - Hi-fi prototypy (grafický návrh, …).
3. **Implementační fáze**
    - Koncepty řešení dílčích požadavků (funkční / nefunkční).
    - Technická specifikace (API) [příloha dokumentace projektu].
    - Testování (inspekce, heuristika).
4. **Vyhodnocení**
    - Testování, logování.
    - Vyhodnocení (feedback, statistiky, logy).

## Ostatní dokumenty projektu

V rámci vývoje vznikají další typy dokumentů s níže uvedenými náležitostmi pro různé skupiny čtenářů. Obsah každého dokumentu je cílený na příslušnou skupinu uživatelů a zohledňuje jejich schopnosti a možnosti.

- **Projektová dokumentace**

[2] State of the Art

    - je množina samostatných dokumentů vznikajících pro jednotlivé iterace agilního vývoje aplikace[3],
    - má strukturu podle fází harmonogramu,
    - odkazuje na související legislativní úpravu problémové domény aplikace,
    - je určená pro zadavatele, návrháře a vývojáře projektu a případně pro uživatele.
- **Instalační a provozní příručka**
    - obsahuje kompletní postup pro sestavení (build) a nasazení (deployment) aplikace a nových verzí,
    - popisuje dostupná prostředí (staging/produkční verze) v návaznosti na kap. Údržba a rozvoj,
    - popisuje provozuschopnost v případě nedostupnosti souvisejících služeb,
    - popisuje proces obnovení provozu v případě výpadku.
- **Uživatelská dokumentace**
    - je průběžně udržovaný samostatný dokument,
    - obsahuje informaci, k čemu a komu aplikace souží,
    - je určená koncovým uživatelům frontendové aplikace[4], resp. aplikačního rozhraní (RESTful API)[5],
    - je dostupná z webu FIT (stačí odkazem)[6],
    - zahrnuje CHANGELOG (viz kap. Verzování).
- **Dokumentace vnitřního API**
    - je sada dokumentů generovaná z kódu průběžně udržovaná společně s kódem aplikace,
    - je psaná anglicky v příslušné standardizované syntaxi[7], přičemž dokumentace veřejných entit zahrnuje minimálně:
        - souhrnný popis dokumentované entity (funkce, třídy, metody, proměnné, …),
        - souhrnný popis parametrů (funkce/metody) nebo typových proměnných (generické typy),
        - popis vyhazovaných výjimek (které výjimky a kdy vznikají),
        - popis návratové hodnoty (a její význam).

# Architektura a integrace do infrastruktury

Projekt je webovou aplikací, která efektivně využívá existující technologie a služby FIT. Aplikace je členěná na nezávislé části, které je možné vyměnit a je provozuschopná i v případě výpadků souvisejících služeb.

- Projekt je webovou aplikací s
    - uživatelským rozhraním (UI) pro webový prohlížeč, nebo
    - RESTful API s upřesněním standardu vč. formátu[8].
- Architektura aplikace
    - striktně odděluje frontend a backend,
    - správa uživatelů (user-management) je zajištěna fakultním IDM,
    - využívá maximum dostupných služeb (např. notifikace),
    - [SHOULD] podporuje použití pro více fakult na jediné instanci.

[3] Agile software development
[4] 10 Examples of Great End User Documentation
[5] Doporučené nástroje pro dokumentaci RESTful API: RAML, Swagger / OpenAPI
[6] Návod ke psaní dokumentace ICT FIT
[7] Např. JavaDoc nebo DoxyGen
[8] Doporučujeme vycházet ze standardu JSON API

- Aplikace
  - je součástí katalogu služeb FIT[9] od počátku práce na projektu (stav „připravuje se“),
  - využívá mezipaměť pro urychlení obsluhy požadavků,
  - je provozuschopná i v případě nedostupnosti (zpomalení) souvisejících služeb[10].
- [MAY] Aplikace je implementovaná na platformě/jazyku:
  - Ruby,
  - JavaScript, resp. izomorfní JavaScript[11],
  - Groovy/Java na Spring Frameworku,
  - Python,
  - PHP na frameworku Symfony.

# Webová přístupnost

Aplikace je přístupná pro uživatele bez ohledu na jejich omezení a zařízení, kterým k aplikaci přistupují.

- Aplikace respektuje požadavky WCAG 2.0 AA[12], zejména
  - sémantické značkování výstupu HTML,
  - jednoznačné perzistentní URL jednotlivých stránek[13],
  - podpora tisku,
  - *progressive enhancement*[14].
- Výstup aplikace (HTML) je v souladu s principem *mobile-first*[15], *media-first*[16], zejména
  - přizpůsobivé uživatelské rozhraní[17],
  - použitelnost ovládacích prvků pro manipulaci prsty,
  - nenáročnost s ohledem na výkon CPU a spotřebu baterie,
  - minimalizace přenesených dat.
- [MAY] Aplikace je odolná vůči výpadkům připojení a funkčnost bez připojení k Internetu.[18]
- [MAY] Aplikace podporuje *Web App Manifest*[19] a integraci do operačního systému[20] zahrnující
  - podporu push notifikací[21],
  - synchronizaci na pozadí přes *Service Workers*[22].

[9] Katalog služeb ICT FIT
[10] Např. bez datového, resp. autentifikačního, zdroje, informace z mezipaměti, resp. zobrazí jen veřejné informace (s příslušným upozorněním).
[11] Viz Isomorphic JavaScript.
[12] Web Content Accessibility Guidelines (WCAG) 2.0
[13] Viz Cool URIs a Why JavaScript web applications should embrace traditional URLs.
[14] Poskytnout klientovi úplnou funkcionalitu i v případě, že nepodporuje dynamické technologie; viz článek na Gov.UK.
[15] Viz kniha Mobile First (Luke Wroblewski).
[16] Společná definice zobrazení od sémantického obsahu pro čtečky a textové interprety, přes tisk a malé obrazovky až po velké obrazovky.
[17] Viz Responsive Web Design.
[18] Tzv. Offline-First.
[19] Viz Web App Manifest.
[20] Viz články Progressive Web Apps: Escaping Tabs Without Losing Our Soul a Getting started with Progressive Web Apps.
[21] Viz Push Notifications on the Open Web.
[22] Viz Introduction to Service Worker.

# Kód aplikace

Veškerý kód je psaný kompletně v angličtině s prioritou udržitelnosti a čitelnosti. Vývoj kódu přehledně odděluje provozní větev od vývojových. Před nasazením prochází každý nově vzniklý kód kontrolou kvality na několika úrovních.

- Aplikace respektuje *best practices* pro psaní udržitelného a čitelného kódu[23]; zejména
  - logické členění kódu do modulů podle funkcionality,
  - specifikace konvencí používaných technologií (např. CSS[24], JavaScript[25], Java[26]),
  - dodržování stylu autora při editaci cizího kódu,
  - minimalizace importů[27] (import, include, using, atd.),
  - používání existujících knihoven[28], kdykoli je to efektivní a smysluplné,
  - používání návrhových vzorů[29],
  - komentování potenciálně nejasných částí.
- Veškerý kód je psaný
  - v UTF-8 s unixovým koncem řádek (řídící znak LF / 0x0A),
  - anglicky (názvy funkcí a proměnných, komentáře, systémová a jiná hlášení).
- Veškeré výstupy (texty pro uživatele) podporují lokalizaci a internacionalizaci.
- Pro aplikaci existuje kompletní česká lokalizace.

## Verzování

Vývoj kódu je organizovaný s přehledným oddělením provozní a vývojové větve. Umožňuje operativní opravy kritických chyb (hotfix) a nezávislý vývoj nových funkcí. Podporuje bezpečný model nasazování nových verzí[30] pro účely testování a ladění (akceptační testy).

- Kód aplikace je vyvíjen na revizním systému Git[31] využívající
  - repozitář na službě GitLab provozované fakultou[32] nebo oddělením ICT[33],
  - standardní branching model Git Flow[34]
    (nástroje OMGF[35] nebo Git-Flow Cheatsheet[36]) a
  - sémantické verzování[37].
- Používání revizního systému se řídí pravidly *commitování*, zejména
  - *commit* každé dílčí změny funkcionality,
  - zachování funkcionality celku přes jednotlivé *commity*,

[23] Viz Best Practices a kniha The Pragmatic Programmer.
[24] Konkrétně konvenci SUIT CSS.
[25] JavaScript Quality Guidelines and Recommendations
[26] Code Conventions for the Java Programming Language
[27] Např. neimportovat celý balíček, když z něj bude použita jen malá část.
[28] Pod svobodnými nebo open-source softwarovými licencemi a respektovat podmínky těchto licencí.
[29] Gang of Four Design Patterns či Design Patterns na Wiki
[30] Deployment environment
[31] Jak na Git
[32] GitLab FIT ČVUT
[33] GitLab ICT
[34] Git Flow
[35] OMGF
[36] Git-Flow Cheatsheet
[37] Semantic Versioning 2.0.0

    - používání rozkazovacího tvaru v přítomném čase[38].
- Součástí vývoje je udržování aktuálního souboru CHANGELOG[39] dostupného z webu na úrovni
    - nových funkcí (či inovací) vždy při jejich začlenění do vývojové větve,
    - nových MINOR verzí vždy při začlenění vývojové větve do provozní.
- Na společných (sdílených) větvích není povoleno přepisování historie.
- Veškeré texty verzování jsou anglicky.

## Kontrola kvality

Veškerý kód se před nasazením patřičně kontroluje na úrovni automatizovaných nástrojů a dílčích (jednotkových a dalších) testů. Součástí kontroly kódu je (jednoduchý) schvalovací proces. Alternativně se kód vyhodnocuje, zda splňuje stanovené kvalitativní metriky.

- Vývoj kódu se opírá o kontrolní nástroje jako zejména
    - příslušný *linter*[40].
- Veškeré nasazování změn kódu (merge) procházejí kontrolním procesem[41] s následujícími pravidly.
    - Veškeré merge jsou prováděny formou požadavků na začlenění[42] (dále PR).
    - Veškeré PR (bez ohledu na svou podstatu a závažnost) budou potvrzované minimálně druhým členem týmu – programátorem, alternativně nadřízeným.
    - PR kritického požadavku si může jeho řešitel sám akceptovat. O takovém úkonu neprodleně vyrozumí členy týmu. Povinnost potvrzení podle předchozího bodu zůstává, však může být učiněno dodatečně (bez zbytečného prodlení).
    - Součástí kontrolního procesu nasazování je continuous integration[43] na GitLabu[44].
- Kód obsahuje automatické testy, mezi které patří zejména
    - jednotkové testy,
    - integrační testy (API, resp. automatizované průchody).
- [SHOULD] Vývoj kódu je řízený testy[45].
- [SHOULD] Minimální požadované hodnoty metrik[46] pomocí fakultní služby Sonar[47] jsou stanoveny následujícím způsobem:
    - Method Total Length (< 30 lines)
    - Class Total Length (< 300 lines)
    - Unit Tests Line coverage (> 70 %)
    - Unit Tests Branch coverage (> 70 %)
    - Density of duplicated lines (< 5 %)
    - Lack of cohesion of methods (< 3)
    - Average complexity by method (< 5)
    - Rules compliance index (žádné závady úrovně „blocker“ ani „critical“)

[38] How to Write a Git Commit Message
[39] Keep a Changelog
[40] Platí zejm. pro dynamické a značkovací jazyky; viz např. doporučení pro JavaScript a CSSLint.
[41] Best Practices for Code Review a Code reviews v praxi
[42] Code Review Via GitLab Merge Requests
[43] Continuous integration
[44] GitLab Continuous Integration
[45] Test-Driven Development
[46] Sonar Metric Definitions
[47] Sonar FIT ČVUT

## Provoz, údržba a rozvoj aplikace a podpora uživatelů

Veškerá (nově vznikající) funkcionalita je uživatelům dostupná přehledně a jednoduše. Aplikace (nová verze) se nasazuje do provozu po splnění akceptačních testů. Součástí údržby a dlouhodobého rozvoje aplikace je sběr informací o používání a jejich pravidelné vyhodnocování následované patřičným zapracováním do aplikace.

- Vzhled uživatelského rozhraní (UI) aplikace je moderní, přehledný, vzdušný a tvořený obsahem.[48]
- Sada akceptačních testů je specifikovaná pro účely testování všech dostupných a nově vznikajících funkcí.
- Sběr dat se provádí na základě
  - zpětné vazby uživatelů prostřednictvím
    - funkce issue tracking v rámci GitLabu a
    - e-mailu na helpdesk,
  - logování[49] a integrace s monitorovacími službami na úrovni
    - chyb (fatal, warning),
    - informačních zpráv o používání[50] (používanost funkcí, doby trvání, přístupy) a
    - systémových zpráv a dalších výstupů.
- Proces vyhodnocování dat zahrnuje
  - podporu uživatelů a
  - opravy chyb včetně klasifikace jejich závažnosti.
- [SHOULD] Proces rozvoje od návrhu po realizaci za účelem
  - vylepšování stávajících funkcí (optimalizace chodu a procesů),
  - přidávání nových funkcí.
- [SHOULD] Proces testování nových verzí aplikace zahrnuje
  - provoz nezávislé (beta) verze,
  - podporu AB testování,
  - provádění inspekcí a heuristik,
  - pozorování.

## Bezpečnost a ochrana osobních údajů

Aplikace je standardně zabezpečená; zejména nepracuje s hesly uživatelů a veškerá komunikace probíhá přes šifrovaný protokol. Aplikace také respektuje nařízení rektora o ochraně osobních údajů.

- Aplikace respektuje principy bezpečných webových aplikací[51], jmenovitě
  - veškerá komunikace (S2S, S2C) probíhá přes HTTPS,
  - jako API poskytuje různé úrovně oprávnění pomocí *scopes*[52],
  - nepracuje s hesly uživatelů; autentizace, resp. autorizace, probíhá přes Shibboleth (není-li potřeba autorizace), resp. autorizační server FIT (protokol OAuth 2.0)[53].

[48] UXMyths: You don't need the content to design a website.
[49] Logging Best Practices
[50] Google Analytics s využitím událostí (events)
[51] Viz principy OWASP.
[52] Securing Access with OAuth2: How to deal with OAuth Scopes
[53] Viz Autorizační server FIT (OAuth 2.0).

- Aplikace respektuje nařízení rektora o ochraně osobních údajů[54], zejména dokumentace definuje,
    - jaké informace jsou citlivé/osobní,
    - jaká data jsou dostupná kterým uživatelům v závislosti na autorizaci uživatele (např. anonymní uživatel, přihlášený uživatel, student, vyučující, administrátor),
    - které citlivé/osobní informace jsou přístupné v rozporu s nařízením.

# Další koncepty webových aplikací (W2.0)

Aplikace explicitně zohledňuje možnosti využití níže uvedených webových konceptů a případně dalších. Vzhledem k omezení jednotlivých projektů mohou být využití konceptů pouze součástí projektové dokumentace – byť jen jako potenciální rozšíření funkcionality s uvedenými přínosy a konkrétními příklady.

- Dokumentace (např. v příloze) popisuje možnosti a úroveň nezbytnosti využití všech následujících konceptů:
    - RSS,
    - personalizace,
    - customizace,
    - folksonomie (tagování),
    - social networking,
    - real-time web[55],
    - crowdsourcing[56],
    - kolaborace,
    - průvodce (wizardy),
    - konfigurátory (rozšířených dotazů vyhledávání, parametrů služby),
    - gamifikace[57],
    - mikrodata[58].

[54] Příkaz rektora č. 5/2015  Ochrana osobních údajů na ČVUT v Praze
[55] Real-time web (Wiki)
[56] Crowdsourcing (Wiki)
[57] Gamification (Wiki)
[58] Microdata (Wiki)

Příloha D

# Další koncepty webových aplikací

Příloha popisuje koncepty moderního webu (známého jako Web 2.0) a současně ukazuje, které z nich budou využity (úroveň nezbytnosti MUST, není-li uvedeno jinak) a které mohou mít uplatnění v budoucích iteracích (MAY). Přehled konceptů vychází z obecných pravidel a zásad projektů FIT (příloha C). Závěry se opírají o funkční požadavky na aplikaci (kap. 1.4).

Následující koncepty budou v aplikaci uplatněné:

- customizace (volba OS a aplikačního softwaru, jazykové preference),
- crowdsourcing (sběr SW požadavků na standardní obrazy OS),
- konfigurátory (dotazy nad zadanými požadavky),
- [SHOULD] personalizace (zobrazení relevantních předmětů vyučujícím a seznamu požadavků správcům instalací),
- [SHOULD] kolaborace (sdílení požadavků mezi uživateli),
- [MAY] RSS,
- [MAY] průvodce.

Následující koncepty nebudou prozatím využity:

- folksonomie,
- social networking,
- real-time web,
- gamifikace,
- mikrodata.

## RSS

Formát RSS je vhodný pro odebírání novinek a informování o změnách webového obsahu. Používá se především u internetových stránek s dynamickým obsahem. Uživatelé aplikace pro podporu řízení počítačových učeben by mohli mít zájem být informováni o změnách učebnových instalací a případně dalších provozních opatřeních právě prostřednictvím RSS.

## Personalizace

Softwarové požadavky pro účely výuky by měly být vztažené k vyučovaným předmětům. Na základě role přihlášeného uživatele by aplikace nabídla vyučujícímu pouze ty předměty, které v aktuálním období vyučuje. Další aplikací personalizace, tentokrát pro správce učeben, může být například zobrazení kompletního seznamu požadavků na obrazy OS, které spravuje. V závislosti na uživatelských rolích by aplikace mohla personalizovat také výchozí jazykové preference (čeština, angličtina).

Vhodným podkladem pro funkce personalizace v budoucích iteracích bude sledování aktivity uživatelů se zohledněním jejich uživatelských rolí. Budou-li patrné korelace mezi rolemi a využíváním konkrétních funkcí, jedná se o užitečný vstup pro implementaci personalizace, například skrze algoritmy strojového učení.

## Customizace

Customizace v aplikaci pro správu počítačových učeben bude zahrnovat (úroveň nezbytnosti MUST):

- volbu operačního systému, resp. jeho obrazu pro aktuální semestr,
- volbu aplikačního softwaru,
- nastavení jazykových preferencí (čeština, angličtina).

Rozšířenými možnostmi customizace jsou:

- [SHOULD] instalace OS „na míru“,
- [MAY] instalace vlastního obrazu OS,
- [MAY] plánování customizované instalace na konkrétní čas a místo.

## Folksonomie (tagování)

Folksonomie v kontextu aplikace podporující správu počítačových učeben by mohla znamenat kategorizaci (štítkování) požadavků na instalace. Uživatelé by mohli například vytvářet a sdílet kolekce požadavků pro své kolegy, případně vytvářet pokročilé vyhledávací dotazy („seznamy“ požadavků). Tyto dotazy a seznamy by byly dále přístupné dalším uživatelům.

Proces vytváření nových kategorií či kategorizace stávajících požadavků přímo uživateli by vyžadoval moderaci, což zvyšuje implementační náročnost. Z tohoto důvodu nebude folksonomie v této iteraci nevyužita.

## Social networking

V rámci aplikace podporující správu počítačových učeben by Social networking představoval podporu propojování uživatelů, např. vyučujících jednotlivých předmětů, možnost komentování a hodnocení požadavků na software apod. S ohledem na doménu aplikace však není cílem vytvářet virtuální komunitu uživatelů.

Aplikace bude podporovat součinnost uživatelů a správců instalací a také uživatelů navzájem (viz Kolaborace).

## Real-time web

Za real-time informace, se kterými bude aplikace pracovat, lze považovat zadané softwarové požadavky, přidružené komentáře a také aktuální stav instalace požadovaného softwaru (operačních systémů a aplikačního softwaru). Tato data by bylo vhodné zobrazit bez prodlevy a nutnosti aktualizovat webovou stránku. Z důvodu implementační náročnosti však nepočítáme s využitím konceptu real-time webu v této iteraci.

## Crowdsourcing

Nové obrazy běžně používaných OS (Gentoo GNU/Linux, MS Windows) vznikají – počínaje výzvou pro uživatele – inkrementální instalací softwaru dle uživatelských požadavků. Vytvoření těchto instalací, coby softwarového prostředí pro potřeby výuky a jednorázových akcí, lze chápat jako crowdsourcing.

## Kolaborace

Koncept kolaborace by mohl nalézt uplatnění především ve sdílení požadavků na software mezi uživateli. Uživatelský průzkum (kap. 1.2) ukázal, že vyučující během specifikace požadavků často spolupracují se svými kolegy, a to

primárně s těmi, kteří se podílí na výuce stejného předmětu. Každý předmět by obsahoval společnou historii požadavků a rovněž návrhy aktuálních požadavků.

Kolaborace zde znamená také spolupráci uživatele a správce počítačové učebny. Uživatelem je typicky vyučující, který klade požadavky na operační systém a aplikační softwaru pro nadcházející období. Požadavky následně podléhají kontrole správcem příslušného operačního systému. Správce rovněž reaguje na případné pozdější reklamace.

## Průvodce (wizardy)

Aplikace by mohla obsahovat „průvodce“ uživatelským rozhraním, minimálně pro pokročilé funkce (např. filtrování požadavků, vyhledávání softwarových balíčků). Základní funkce aplikace však musí být natolik snadné, že potřeba průvodce by ukazovala na chybu v návrhu. Ani případná existence průvodce zároveň neodstraňuje potřebu uživatelské dokumentace.

## Konfigurátory

Pokud aplikace bude zahrnovat možnost složitějšího dotazování nad daty, měla by obsahovat i rozhraní pro vytváření dotazů. Může se například jednat o pomocná tlačítka pro vkládání parametrů dotazů, nebo o formulář pro sestavení parametrů vyhledávání.

## Gamifikace

Motivací uživatelů k používání aplikace bude zajištění softwarového prostředí pro potřeby výuky, resp. jednorázových akcí. Není proto nutné motivovat uživatele k používání aplikace začleněním herních prvků.

## Mikrodata

Mikrodata, podobně jako mikroformáty[61] nebo RDFa[62], umožňují rozšířit sémantiku jazyka HTML a vkládat do webových stránek strojově čitelné informace. Tyto informace mohou být dále zpracovány dalšími aplikacemi, např. webovými prohlížeči a vyhledávači.

Aplikace nebude poskytovat data pro nepřihlášené uživatele a tím pádem ani pro vyhledávače. Z tohoto důvodu prozatím nebudou mikrodata v aplikaci využita.

[61] `https://en.wikipedia.org/wiki/Microformat`
[62] `https://en.wikipedia.org/wiki/RDFa`

Příloha E

# Oprávnění uživatelů aplikace

Příloha definuje oprávnění uživatelů aplikace v závislosti na úrovni autorizace. Tj. jaká data jsou kterým uživatelům dostupná a jaké operace mohou provádět. Definice oprávnění je zpracována v tabulce E.1 na následující stránce.

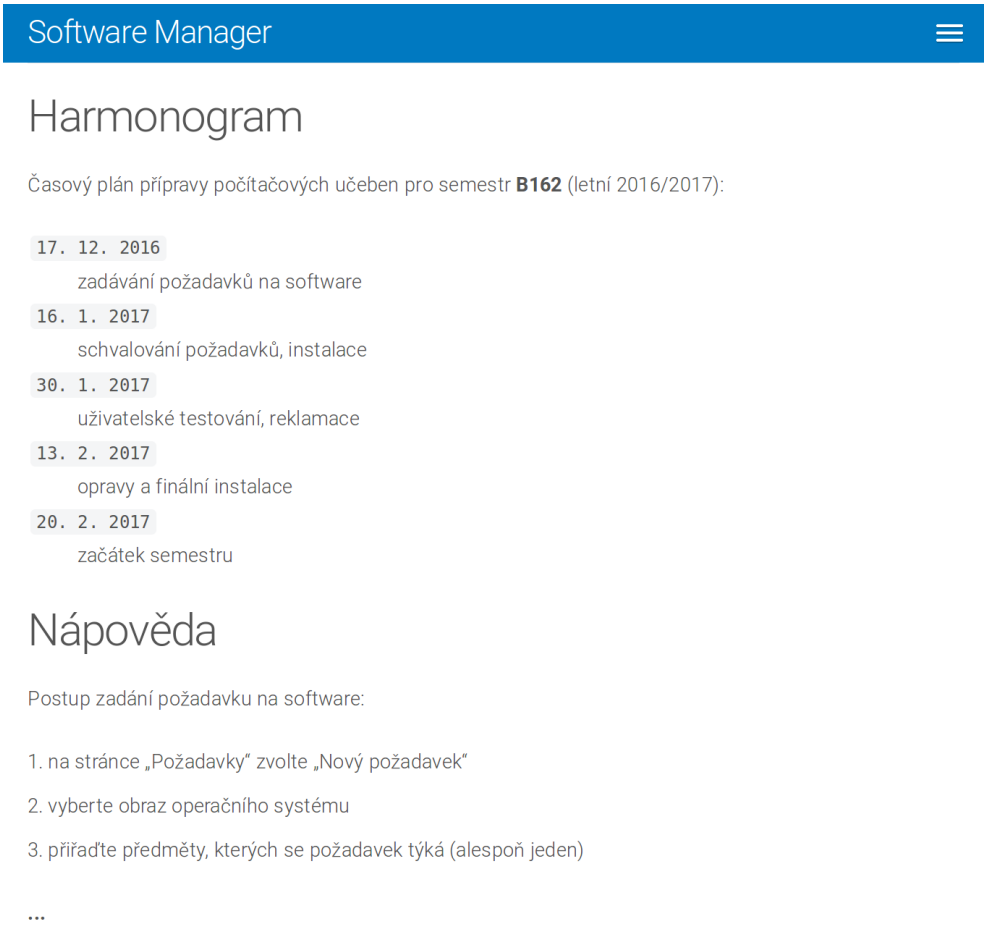
Tabulka E.1: Oprávnění uživatelů aplikace

| | **Běžný uživatel** (`ROLE_USER`) | **Správce instalace** (`ROLE_USER`) | **Správce systému** (`ROLE_ADMIN`) |
|---|---|---|---|
| Zobrazit domovskou obrazovku | ano | ano | ano |
| Zobrazit instalovaný software | ano | ano | ano |
| Vytvořit požadavek | ano | ano | ano |
| Upravit / komentovat požadavek / přiložit soubor | ano[a] | ano | ano |
| Předat ke schválení / ověřit / reklamovat požadavek | ano[a] | ano | ano |
| Odstranit požadavek | ano[a] | ano | ano |
| Schválit / vrátit / instalovat požadavek | ne | ano[b] | ano |
| Zobrazit obrazovku Obrazy OS | ne | ano | ano |
| Vytvořit obraz | ne | ano[c] | ano |
| Upravit / odstranit obraz | ne | ano[d] | ano |
| Zobrazit obrazovku Operační systémy | ne | ne | ano |
| Vytvořit / upravit / odstranit OS | ne | ne | ano |
| Zobrazit obrazovku Semestry | ne | ne | ano |
| Importovat / ručně vytvořit semestr | ne | ne | ano |
| Nastavit harmonogram semestru | ne | ne | ano |
| Zobrazit obrazovku Předměty | ne | ne | ano |
| Importovat / ručně vytvořit předmět | ne | ne | ano |

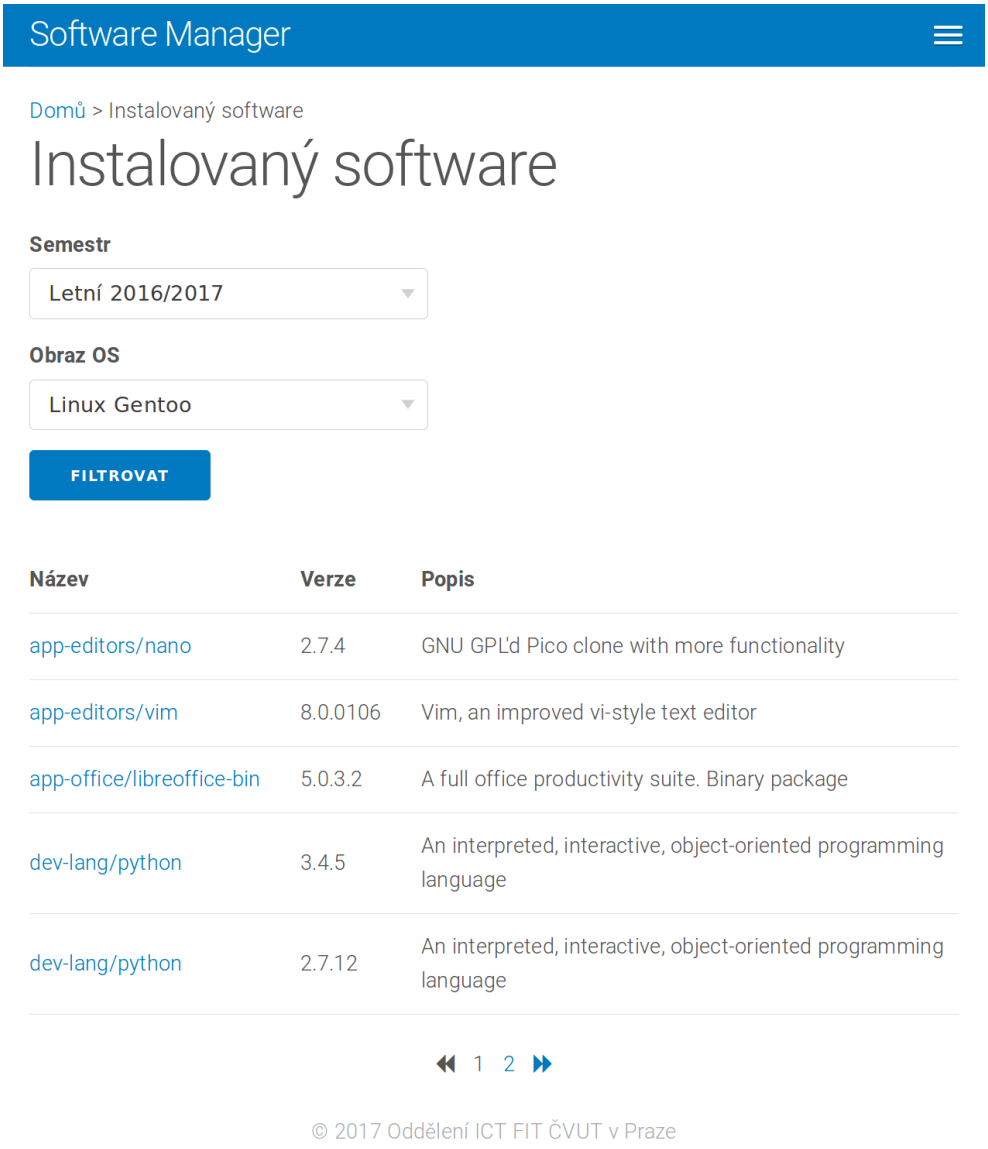
[a] Pouze autor požadavku.
[b] Správce obrazu, pro který byl požadavek zadán.
[c] Libovolný ze správců daného operačního systému.
[d] Libovolný ze správců daného obrazu.

Příloha F

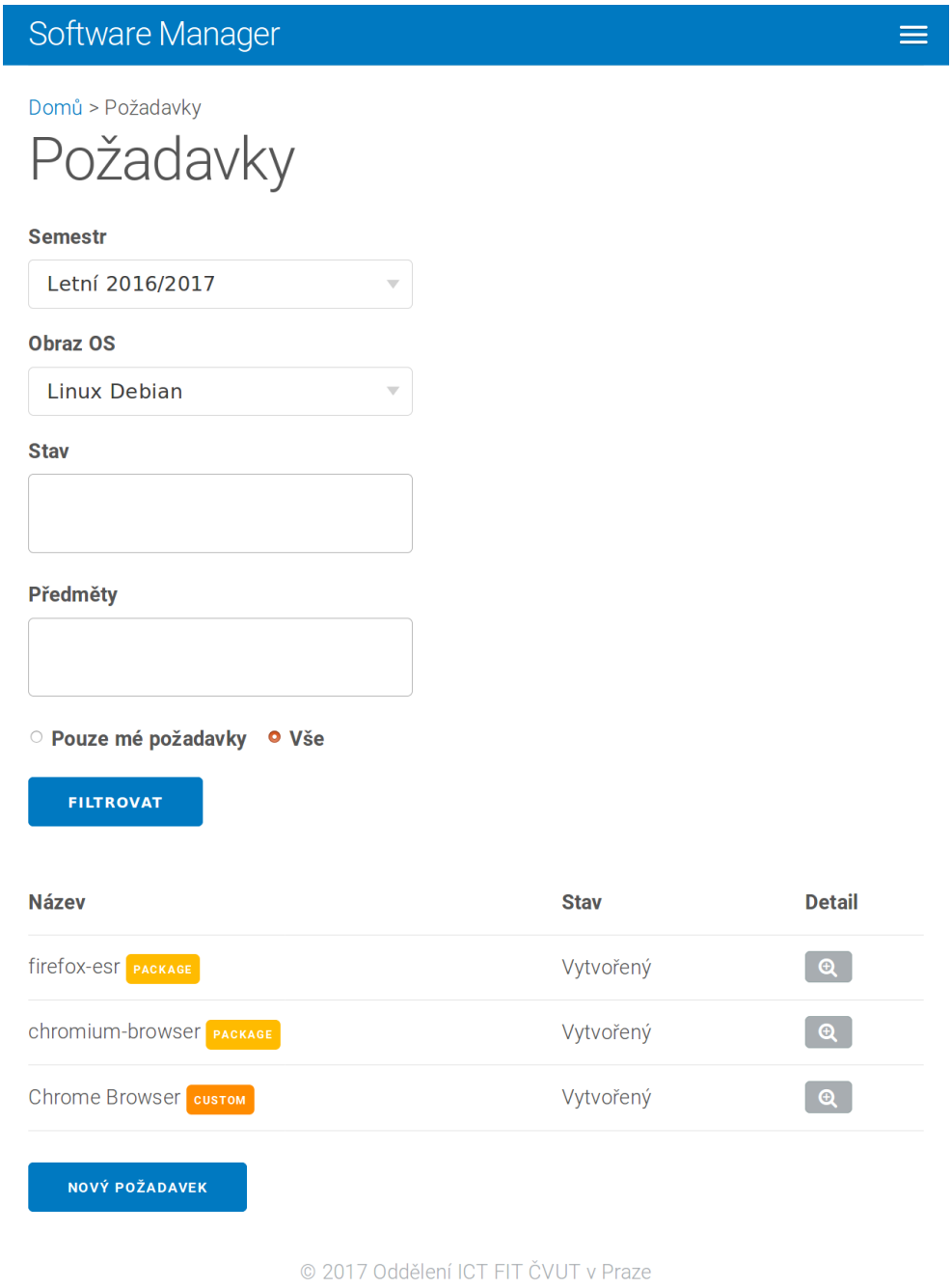
# Ukázky aplikace



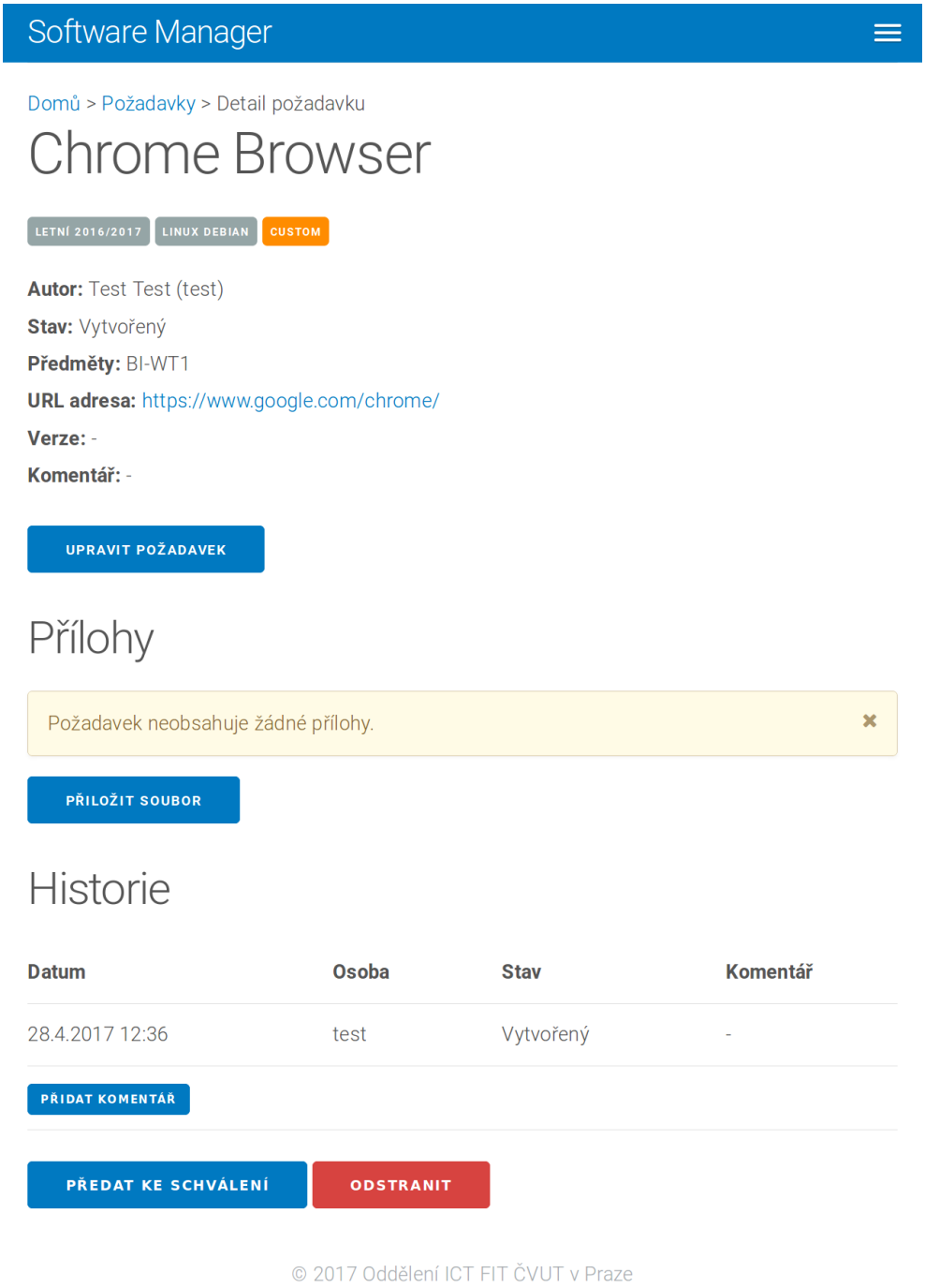

Obrázek F.1: Ukázka aplikace – domovská obrazovka



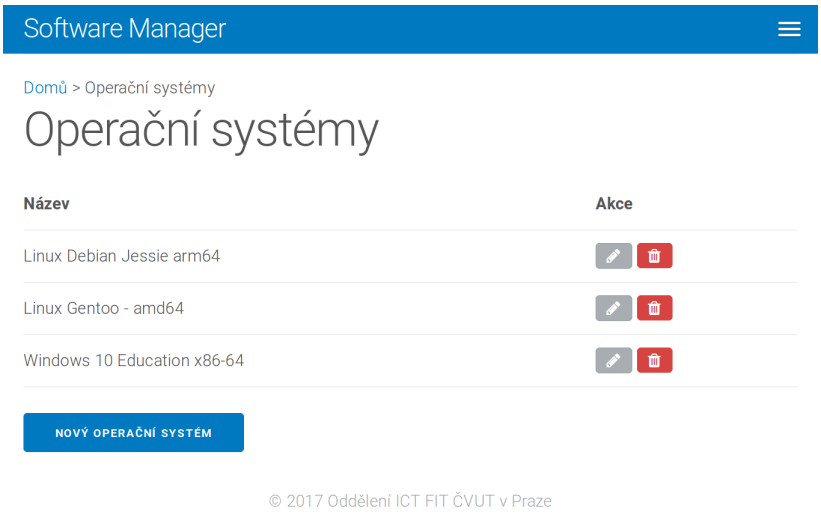

Obrázek F.2: Ukázka aplikace – obrazovka Instalovaný software



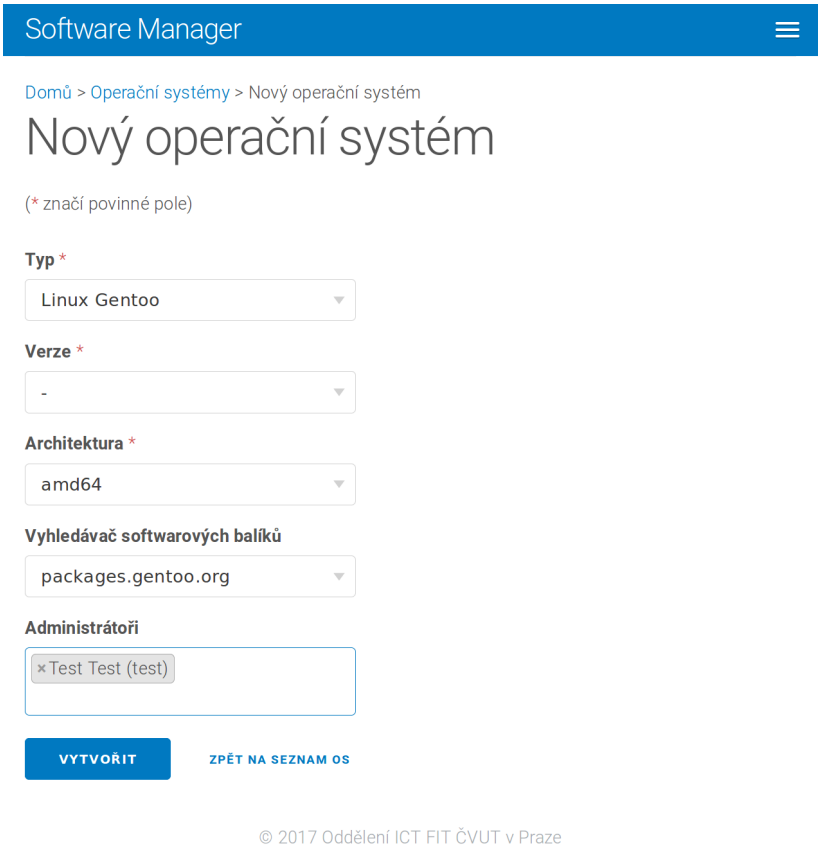

Obrázek F.3: Ukázka aplikace – obrazovka Požadavky




Obrázek F.4: Ukázka aplikace – obrazovka Detail požadavku




Obrázek F.5: Ukázka aplikace – obrazovka Operační systémy




Obrázek F.6: Ukázka aplikace – obrazovka Nový operační systém

Příloha G

# Instalační příručka

Instalace aplikace sestává z několika málo kroků. Následující postup je platný pro systémy Linux/Unix s instalací PHP >= 7.0.

1. Zkopírujte obsah adresáře `impl/src/` z přiloženého CD na disk počítače a přejděte do tohoto adresáře.

2. Instalace back-end závislostí pomocí nástroje Composer[63]; při instalaci nastavte parametry připojení k databázi:

   ```
   $ ./getcomposer.sh
   ```

   ```
   $ php composer.phar install
   ```

3. (volitelně) Ověření, že systém splňuje všechny požadavky pro běh Symfony aplikace:

   ```
   $ php bin/symfony_requirements
   ```

4. Vytvoření databáze dle parametrů v `app/config/parameters.yml` a databázového schématu:

   ```
   $ php bin/console doctrine:database:create
   ```

   ```
   $ php bin/console doctrine:schema:create
   ```

5. (volitelně) Nahrání testovacích dat do databáze:

   ```
   $ php bin/console doctrine:fixtures:load
   ```

[63] `https://getcomposer.org/`

6. Instalace front-end závislostí pomocí nástroje Bower[64] a jejich kompilace pomocí Assetic[65]:

   ```
   $ bower install
   ```

   ```
   $ php bin/console assetic:dump
   ```

7. Spuštění vestavěného PHP serveru na `localhost:8000`:

   ```
   $ php bin/console server:start
   ```

8. Existují-li v databázi testovací data (krok 5), je možné se po přistoupení na `http://localhost:8000/cs/` přihlásit pod uživatelským jménem `admin` a heslem `admin` (prostřednictvím HTTP Basic).

[64] `https://bower.io/`
[65] `http://symfony.com/doc/current/assetic/asset_management.html`

Příloha H

# Harmonogram projektu

Příloha obsahuje časový plán vypracování předkládané diplomové práce. Údaje v [závorkách] u aktivit značí data jejich ukončení.

1. Analytická fáze
    - stávající stav [26. 9. 2016]
    - kvalitativní uživatelský průzkum (rozhovory) [9. 8. 2016]
    - kvantitativní uživ. průzkum – zveřejnění dotazníku [1. 9. 2016]
    - kvantitativní uživ. průzkum – uzavření dotazníku [26. 9. 2016]
    - kvantitativní uživ. průzkum – vyhodnocení odpovědí [3. 10. 2016]
    - posouzení stávajícího stavu [18. 10. 2016]
    - sestavení požadavků na funkce [18. 10. 2016]
    - sestavení požadavků na kvality [19. 10. 2016]
2. Návrhová fáze
    - stanovení procesů, případy užití [5. 1. 2017]
    - lo-fi prototypy [5. 1. 2017]
    - doménový model aplikace, popis entit [21. 12. 2016]
3. Implementační fáze
    - rozmyšlení a implementace dílčích konceptů [1. 4. 2017]
    - implementace RESTful API [18. 4. 2017]
    - testování aplikace [4. 5. 2017]
4. Odevzdání práce [9. 5. 2017]

Příloha I

# Seznam použitých zkratek

**AJAX** Asynchronous JavaScript and XML

**API** Application Programming Interface

**CI** Continuous Integration

**CRUD** Create, Read, Update, Delete

**CSS** Cascading Style Sheets

**CSV** Comma-separated values

**ČVUT** České vysoké učení technické v Praze

**DI** Dependency Injection

**ER** Entity-relationship

**FIT** Fakulta informačních technologií

**GNU** GNU's Not Unix

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**HW** Hardware

**ICT** Information and Communication Technologies

**IDE** Integrated Development Environment

**IT** Information Technology

**JSON** JavaScript Object Notation

**LAN** Local Area Network

**Lo-Fi** Low-Fidelity

**MIT** Massachusetts Institute of Technology

**ORM** Object-Relational Mapping

**PHP** PHP: Hypertext Preprocessor

**PSR** PHP Standards Recommendations

**RDFa** Resource Description Framework in attributes

**REST** Representational State Transfer

**RSS** Rich Site Summary

**RT** Request Tracker

**SSO** Single Sign-On

**SW** Software

**UC** Use Case

**UI** User Interface

**URL** Uniform Resource Locator

**XLIFF** XML Localisation Interchange File Format

**XML** eXtensible Markup Language

**YAML** YAML Ain't Markup Language

Příloha J

# Obsah přiloženého média

- `README.txt` ..... stručný popis obsahu média
- `impl`
  - `install.txt` ..... instalační příručka aplikace
  - `doc`
    - `api` ..... vygenerovaná dokumentace vnitřního API
    - `coverage` ..... vygenerovaný report pokrytí kódu testy
  - `src` ..... zdrojové kódy implementace
- `text`
  - `DP_Papež_Karel_2017.pdf` ..... text práce ve formátu PDF
  - `src` ..... zdrojová forma práce ve formátu LaTeX